Online Aggregation über Datenströmen mit Verfahren der mathematischen Statistik in grossen Datenbanksystemen

Dissertation

zur

Erlangung des Doktorgrades

der Naturwissenschaften
(Dr. rer. nat.)

dem
Fachbereich Mathematik und Informatik
der Philipps-Universität Marburg
vorgelegt von

Björn Blohsfeld
aus Ehringshausen

Marburg/Lahn 2002

Vom Fachbereich Mathematik und Informatik

der Philipps-Universität Marburg als Dissertation am 6. Mai 2002

angenommen.

Erstgutachter Prof. Dr. Bernhard Seeger

Zeitgutachter Prof. Dr. Manfred Sommer

Tag der mündlichen Prüfung am 8. Mai 2002

Vorwort

Die Speicherung und Verarbeitung großer Datenmengen ist eine Kernkompetenz heutiger Datenbanksysteme. Trotzdem steigt die Zeit, die Anwender mit dem Warten auf die Ergebnisse ihrer Anfragen verbringen. Dies liegt vor allem an der schneller wachsenden Flut von Daten und Informationen im Vergleich zum technischen Fortschritt.

Eine spätere Analyse und Verarbeitung dieser gespeicherten Datenberge ist mit heutigen Technologien kaum mehr zu bewältigen. Neue Verfahren und Ansätze werden daher benötigt, die es ermöglichen, die Daten bei ihrer Entstehung direkt an der Quelle zu verarbeiten. Somit muss der Weg weg von einer nachträglichen Analyse der Daten hin zu einer Verarbeitung vor der Speicherung gehen. Zudem häufen sich die Anwendungsfelder, in denen die Daten nicht mehr (oder noch nicht) in persistenter Form vorliegen. Auch hier ist eine Verarbeitung direkt auf dem Datenstrom unerläßlich.

Viele Verfahren, die aus der mathematischen Statistik bekannt sind, haben auch in praktischen Anwendungen ihr Potenzial gezeigt. Doch orientieren sich die in diesen Anwendungen verwendeten Modelle zu stark an den Verfahren selbst und zu wenig am Anwender oder der Anwendungsumgebung. So muss in der Regel ein Experte die Ergebnisse bzw. das Ergebnis eines Verfahrens interpretieren und die entsprechenden Schlüsse ziehen. Und der Einsatz solcher Verfahren der mathematischen Statistik in Datenbanksystemen krankt vor allem in der fehlenden Unterstützung für die dort vorhandenen und stark einschränkenden Gegebenheiten. Ein Beispiel dafür ist die in der Regel fehlende Unterstützung der höchst optimierten Datenzugriffstechniken, die Sekundärspeicherzugriffe minimieren und optimieren.

Ist also von mathematischer Statistik oder einfach Statistik und Datenbanksystemen in einem Atemzug die Rede, bezieht sich dies meist auf die junge Disziplin des *Data Mining*. Mit Hilfe von Verfahren des *Data Mining* können zum Beispiel Informationen sichtbar gemacht werden, die mit klassischen Techniken übersehen worden wären. Doch viele Verfahren beruhen auf experimentell validierten Heuristiken, die keine theoretische Fundierung besitzen. Somit existieren in diesen Fällen auch keinerlei Aussagen zur Qualität und Richtigkeit der Ergebnisse eines Modells. Doch diese Verfahren berücksichtigen die besonderen Umstände und Zielkriterien, die die Verarbeitung von Daten innerhalb eines DBMS mit sich bringen.

Die Berechnung von Aggregationen ist eine Standardfunktionalität heutiger DBMS. Alle gängigen Systeme können Anzahl, Minimum, Maximum, Summe, Mittelwert und in der Regel auch Varianz der (numerischen) Daten einer Relation berechnen. Liegen die Daten jedoch nicht persistent vor und soll die Aggregation *online*, d. h. auf

einem Datenstrom, berechnet werden, ist dies in der Regel nicht möglich. Die dafür notwendigen Techniken sind von Haas et al. in [HHW97] erstmals vorgestellt worden. Auf diese Techniken wird kurz motivierend eingegangen und die grundlegenden Prinzipen der *Online Aggregation* werden anhand von notwendigen und hilfreichen Kriterien formuliert. Aufgrund dieser Kriterien kann der Begriff der Online Aggregation auch auf induktive im Gegensatz zu bisher betrachteten deskriptiven Statistiken erweitert werden.

Einen ersten universellen Ansatz zur Berechnung beliebiger Aggregate liefert das Verfahren des speicherbasierten Stichprobenziehens (*reservoir sampling*) nach Vitter [Vit85]. Diese Technik liefert zu jedem Zeitpunkt eine gültige *i.i.d.* Stichprobe über einem Datenstrom für die bis dahin verarbeiteten Daten. In Kombination mit Kernschätzern zur Dichte- und Selektivitätsschätzung werden so Dichte- und Selektivitätsschätzer für Bereichsanfragen über Datenströmen möglich.

Die universelle Einsetzbarkeit wird jedoch mit einigen Schwächen dieser Technik bezahlt. So wird zum Beispiel jedes Datum verarbeitet, jedoch in der Regel nur ein Bruchteil davon zur Berechnung der Aggregation herangezogen. Mit Hilfe einer adaptiven Technik in Kombination mit einem geeigneten Komprimierungsverfahren können diese Schwächen jedoch erheblich reduziert werden. So können zum Beispiel numerische Daten auf einer Intervallskala effizient mit Hilfe von Verfahren der nichtparametrischen Statistik segmentweise geschätzt und diese Segmente anwendergesteuert kombiniert werden. Die Leistungsfähigkeit aller entwickelten speicherbasierten und adaptiven Techniken wird mit Hilfe ausgiebiger und praxisnaher Tests validiert und in Relation zu einem Referenzverfahren gesetzt. Hier wird sowohl der Grad an Sortiertheit eines Datenstroms wie auch das Verhältnis der Größe der Grundgesamtheit zum verwendeten Speicher berücksichtigt. Der Einsatz eines hybriden Verfahrens auf Basis von Kernschätzern zur Selektivitätsschätzung macht es zudem notwendig, auch Sprungstellen in den Daten zu lokalisieren und die Wahrscheinlichkeit ihres Auftretens zu bestimmen.

Im Mittelpunkt aller Verfahren steht ein für die Verfahren lediglich begrenzt zur Verfügung stehender Speicherplatz. Sowohl das Referenzverfahren als auch die neu entwickelten Techniken werden ausschliesslich bei gleicher Speichernutzung verglichen. Dabei ist die Leistungsfähigkeit des klassischen *Sampling* (empirische Stichprobenfunktion) in Kombination mit einer Komprimierung auf Basis kubischer Splines besonders bemerkenswert.

Eine hohe Transparenz der verwendeten und implementierten Verfahren besitzt einen hohen Stellenwert. So stehen alle verwendeten Techniken *ready-to-use* und voll dokumentiert in der Java-Bibliothek XXL [1] zur Verfügung. Die Weiterentwicklung der Bibliothek ist ein Schwerpunkt der Arbeitsgruppe Datenbanksysteme des Fachbereichs Mathematik und Informatik der Universität Marburg.

[1] siehe [XXL]

Inhaltsverzeichnis

1 Einleitung **1**

1.1 Mathematische Statistik und Datenbanksysteme 1

1.2 Gliederung der Arbeit 3

2 Einordnung der Arbeit **5**

2.1 Motivation 6

2.1.1 Verarbeitung aktiver Datenquellen 6

2.1.2 Operatorbäume und Zwischenergebnisse 7

2.2 Begriffsbildung 8

2.2.1 Skalen und Ordnungen 8

2.2.2 Diskret versus Stetig 9

2.2.3 Deskriptive und induktive Statistik 10

2.2.4 Parametrische und nichtparametrische Statistik 11

2.3 Datenströme 12

2.4 Data Mining und Mathematische Statistik 13

2.5 XXL 14

2.5.1 Funktionen und Prädikate 15

2.5.2 Iteratoren und Cursor 16

2.5.3 Das Statistik-Paket 18

3 Online Aggregation **21**

3.1 Grundlagen der Online Aggregation 22

3.2 Bisherige Ansätze 24

3.3 Statistik und Online Aggregation 25

3.4 Voraussetzungen zur Online Aggregation 26

3.5 Online Aggregation mit klassischen Aggregaten 27

3.5.1 Mittelwert 27

3.5.2 Summe 29

3.5.3 Varianz und Standardabweichung 30

3.5.4 Eignung klassischer Aggregate zur Online Aggregation 33

3.5.5 Klassische Online Aggregation in XXL . . . 33
3.6 Aggregation und Kennzahlen . . . 34

4 Speicherbasierte Induktive Aggregation auf Datenströmen 35
4.1 Induktive Statistiken und Stichproben . . . 35
4.2 Speicherbasiertes Stichprobenziehen . . . 36
4.2.1 Typ R . . . 38
4.2.2 Typ X, Typ Y und Typ Z . . . 40
4.2.3 Implementierung in XXL . . . 43
4.2.4 Vergleich der verschiedenen Ansätze . . . 44
4.3 Induktive Statistiken auf Datenströmen . . . 45
4.3.1 Kernschätzer . . . 47
4.3.1.1 Kerndichteschätzer . . . 47
4.3.1.2 Kernfunktionen . . . 49
4.3.1.3 Bandbreitenwahl für Kernschätzer . . . 50
4.3.1.4 Randbehandlung und Konfidenzintervall . . . 53
4.3.2 Hybrider Kerndichteschätzer . . . 55
4.3.3 Kernselektivitätsschätzer . . . 57
4.3.4 Hybrider Selektivitätsschätzer . . . 59
4.3.5 Schätzer zur Sprungstellenentdeckung . . . 59
4.3.6 Kernschätzer in XXL . . . 63
4.4 Probleme und Grenzen des Ansatzes . . . 65

5 Online Aggregation Induktiver Statistiken 67
5.1 Allgemeiner Adaptiver Schätzer . . . 67
5.2 Komprimierung reeller Funktionen mit Splines . . . 75
5.2.1 Interpolation mit natürlichen kubischen Splines . . . 76
5.2.2 Konstruktion und Berechnung von kubischen Splinefunktionen . . . 78
5.2.3 Komprimierung mit Splines . . . 81
5.2.4 Splines in XXL . . . 82
5.3 Vom speicherbasierten zum größenbeschränkten adaptiven Schätzer . . . 83
5.3.1 Splinekomprimierte adaptive Schätzer . . . 85
5.3.2 Adaptive Schätzer in XXL . . . 87
5.3.3 Komprimierte adaptive Schätzer in XXL . . . 90
5.4 Evaluation der Verfahren . . . 92
5.4.1 Testkriterien und Modellannahmen . . . 92
5.4.2 Fehler und Fehlermaße . . . 93
5.4.3 Testdaten . . . 94
5.4.3.1 Erzeugung von Daten beliebiger Verteilung . . . 94

5.4.3.2 Teilsortierte Daten 96
5.4.4 Fensteranfragen 97
5.4.5 Die empirische Verteilungsfunktion 97
5.5 Ergebnisse der Tests 99
5.5.1 Einfluss der Speichergröße auf speicherbasierte Dichteschätzer . 100
5.5.2 Einfluss des Grads der Sortiertheit auf speicherbasierte Dichteschätzer 101
5.5.3 Einfluss der Verfahrensparameter auf speicherbasierte Schätzer . 103
5.5.4 Einfluss der Gewichte auf Adaptive Dichteschätzer 104
5.5.4.1 Arithmetische Gewichte und teilsortierte Daten 105
5.5.4.2 Geometrische Gewichte und teilsortierte Daten 106
5.5.4.3 Teilsortierte Daten und unterschiedliche Gewichtsfunktionen 107
5.5.4.4 Speicherbasierter und adaptiver Difference Kernel Estimator 109
5.6 Evaluation der angewandten Schätzer 112
5.6.1 Vergleich der Verfahren 112
5.6.2 Einfluss der Speichergröße auf adaptives Sampling 114
5.6.3 Quantitative und qualitative Evaluation der Verfahren 115
5.7 Diskussion des allgemeinen adaptiven Schätzers 119

6 Diskussion und Ausblick **121**

A Testdaten **127**
A.1 Erzeugte Datensätze 127
A.1.1 Klassische Verteilungen 127
A.1.2 Daten mit Sprungstellen 128
A.2 Reale Datensätze 130
A.3 Statistische Eigenschaften der Testdaten 131
A.4 Teilsortierte Datensätze 131

B Algorithmen **135**
B.1 Dichteschätzer 135
B.2 Selektivitätsschätzer 137

C Ergebnisse der Tests **141**
C.1 N(0,1)-Daten in zufälliger Reihenfolge 141
C.2 CP2-Daten in teilsortierter Reihenfolge 145
C.3 CP1-Daten in fast sortierter Reihenfolge 147
C.4 STR-Daten in nicht veränderter Reihenfolge 150

Kapitel 1

Einleitung

Die Speicherung und Verarbeitung großer Datenmengen ist eine Kernkompetenz heutiger Datenbanksysteme (DBS). Doch trotz ständiger Weiterentwicklung der Systeme und stetig wachsender Hardwarekapazitäten steigen oft die Zugriffs- und Anfragezeiten auf extrem großen Datenbeständen. Dies liegt unter anderem an der noch schneller wachsenden Flut von Daten und Informationen im Vergleich zum technischen Fortschritt. Speichermedien, die problemlos mehrere *Terabyte* an Daten aufnehmen können, verführen dazu, alles zu speichern und aufzuzeichnen. Eine Analyse dieser so enstehenden Datenberge ist aber kaum noch zu bewältigen, da die dazu notwendigen komplexen Verfahren höhere Kapazitäten benötigen als die Speicherung. Daher ist es nur noch eine Frage der Zeit, bis der Punkt erreicht ist, ab dem Daten ungenutzt und unverarbeitet auf der Datenmüllhalde landen.

Somit muss der Weg weg von einer nachträglichen Analyse der Daten hin zu einer Verarbeitung vor der Speicherung gehen. Dazu ist es notwendig, die Daten als Strom an der Quelle direkt zu verarbeiten. Auch in Anwendungsfeldern, in denen die Daten nicht in persistenter Form vorliegen, sondern nacheinander erzeugt bzw. gemessen werden, müssen Datenströme betrachtet werden. Speziell bei der Übertragung großer Datenbestände wie Videodaten nutzt man das Konzept des *Streamings*.

Doch die Nutzung des Konzepts der Datenströme bedingt ein Umdenken in der Verarbeitung der Daten. So ist eine wahlfreier Zugriff auf beliebige Bereiche der Daten nicht mehr möglich. Dazu kommt eine zeitliche Komponente, die den Einsatz zu gering skalierbarer Verfahren unmöglich macht.

1.1 Mathematische Statistik und Datenbanksysteme

Ist von mathematischer Statistik oder einfach Statistik und Datenbanksystemen in einem Atemzug die Rede, bezieht sich dies meist auf die junge Disziplin des *Data Mining*. Für die einen ist es die Lösung aller Probleme und eine Quelle neuer unentdeckter Informationen in Datengräbern, für die anderen lediglich bekannte Verfahren der Statistik mit neuem Namen.

Doch beides ist zugleich falsch und richtig. Sicher können mit Hilfe von Verfahren des *Data Mining* Informationen sichtbar gemacht werden, die ansonsten übersehen worden wären. Auch können mit ihrer Hilfe zunehmend qualitative Aussagen zu Datenbeständen gemacht werden und Tendenzen in den Daten früher als bisher oder überhaupt erst erkannt werden.

Viele Verfahren beruhen auf experimentell validierten Heuristiken, die jedoch keine theoretische Fundierung besitzen. Somit existieren auch keinerlei Aussagen zur Qualität und Richtigkeit der Ergebnisse eines Modells. Doch diese Verfahren berücksichtigen die besonderen Umstände und Zielkriterien, die die Verarbeitung von Daten innerhalb eines DBMS mit sich bringen. Minimale Anzahl von Sekundärspeicherzugriffen, beste Lösung zu festem Speicherplatz, spezielle Zugriffsmethoden (zum Beispiel durch Datenströme), sequentielles Lesen und Schreiben der Daten, Ausnutzung von Cache-Effekten und subquadratische Laufzeiten sind nur einige Modellparameter für Algorithmen, die in bzw. auf einem DBMS arbeiten.

Dem stehen viele Verfahren der mathematischen Statistik mit ihren mathematischen Modellen und oft genauen Qualitätsaussagen gegenüber. Es liegt nahe, diese Quelle an wohlbekannten Verfahren zu nutzen. Doch die Methoden der mathematischen Statistik sind kein Allheilmittel und haben ihre Schwächen. Ein großes Handicap ist das oftmals notwendige Expertenwissen zur Anwendung vieler Verfahren und zur Interpretation der Ergebnisse. Selbst wenn dieses Wissen vorhanden ist, kann es oftmal nicht eingesetzt werden. Denn die heutigen großen Datenbestände bedingen eine unüberwachte und automatische Datenanalyse. Kritische Parameter der Verfahren müssen ohne Eingriff von außen bestimmt und evaluiert werden.

Dazu kommt eine oftmals mangelnde Skalierbarkeit statistischer Verfahren. Doch Datenbanksysteme sind Multi-User-Systeme mit beschränkten Kapazitäten für jeden Anwender und jeden zugreifenden Prozess. Gering skalierbare Verfahren sind in einer solchen Umgebung in Verbindung mit den extremen Datenmengen nicht einsetzbar. Viele Verfahren der Statistik sind nur für kleine Stichproben aus kleinen Grundgesamtheiten entwickelt und evaluiert worden und damit für die Datenmengen heutiger Datenbanksysteme nicht gerüstet.

Somit ist *Data Mining* mehr als die Verwendung statistischer Methoden in Datenbanken. Vielmehr bedingen die Besonderheiten der involvierten Disziplinen ein Umdenken im Umgang mit großen Datenbeständen.

Neben der fehlenden theoretischen Fundierung vieler experimentell validierter Heuristiken in der Datenbankwelt besteht oft ein weiteres Problem. So basiert die Evaluation vieler neuer Verfahren lediglich auf einer experimentellen Auswertung in verschiedenen Modellszenarien in einem prototypischen, geschlossenen System. Die Ergebnisse sind in der Regel besser als die konkurrierender Algorithmen. Für Außenstehende sind diese Experimente jedoch aus vielen Gründen nicht transparent und damit nicht nachvollziehbar. Dies schränkt den Nutzen und die Bedeutung dieser Verfahren stark ein.

Hohe Transparenz implementierter Verfahren und eine breite Basis zur Evaluation und Entwicklung neuer Algotihmen waren damit zwei wichtige Ziele in der Entwicklung der prototypischen Java-Bibliothek XXL . Die Weiterentwicklung der Bibliothek ist zu einem Schwerpunkt der Arbeitsgruppe Datenbanksysteme des Fachbereichs Ma-

thematik und Informatik der Universität Marburg geworden. Auch sämtliche Verfahren dieser Arbeit stehen transparent in XXL zur Verfügung und können damit leicht nachvollzogen werden.

1.2 Gliederung der Arbeit

In vielen Bereichen nimmt heute die Statistik einen wichtigen Platz ein. Täglich nutzen wir Begriffe der Statistik selbst oder werden mit ihnen durch die Medien konfrontiert. Doch auch uns vertraute Begriffe haben je nach Zusammenhang und Kontext oft verschiedene Bedeutungen und bergen damit eine trügerische Sicherheit im Umgang mit Verfahren der mathematischen Statistik. Dazu kommt eine oft modifizierte Bedeutung ursprünglich klar definierter Begriffe im Sprachgebrauch anderer Disziplinen. Da die vorliegende Arbeit sowohl im Kontext der mathematischen Statistik als auch in der Datenbankwelt angesiedelt ist, gibt Kapitel 2 eine Einführung der in dieser Arbeit verwendeten Begriffe aus dem Gebiet der Datenbanklehre und der Statistik.

Kapitel 3 gibt einen Überblick zu klassischen Aggregaten und ihrer Verwendung über Datenströmen. Darüber hinaus werden notwendige und nützliche Kriterien entwickelt, welche Aggregate über Datenströmen berechenbar und wie die Ergebnisse zu interpretieren sind. Kapitel 4 zeigt einen allgemeinen Ansatz zur Verwendung beliebiger induktiver Statistiken über Datenströmen zu einem vorgegebenen festen und beschränkten Speicherplatz. Dazu werden einige bekannte Verfahren der nichtparametrischen Statistik auf das verwendete Modell angepasst und bezüglich der Kriterien aus Kapitel 3 bewertet.

In Kapitel 5 wird ein neuer Ansatz zur Aggregation mittels der in Kapitel 4 vorgestellten Verfahren entwickelt und mit Hilfe verschiedener Tests untersucht. Dazu werden alle Verfahren untereinander und zusätzlich mit einem Referenzverfahren verglichen. Weiterhin werden die Fehler bezüglich einiger Testmengen von Anfragen bestimmt. Kapitel 6 fasst die Ergebnisse dieser Arbeit zusammen, gibt einen Ausblick auf zukünftige Arbeiten und formuliert offen gebliebene Fragen.

Alle Verfahren dieser Arbeit stehen mit der Java-Bibliothek XXL [1] transparent und dokumentiert zur Verfügung. Damit alle Verfahren und Tests nachvollziehbar sind, werden in jedem Kapitel die Aufrufe der verschiedenen Verfahren in XXL erläutert und diskutiert.

[1] vgl. Abschnitt 2.5

Kapitel 2

Einordnung der Arbeit

Der Begriff des Aggregats ist seit Jahren wohlbekannt in der Datenbankwelt. Die Berechnung einfacher statistischer Kennzahlen von Datensätzen in Datenbanken, wie zum Beispiel der Mittelwert oder die Varianz, werden zum Beispiel in *SQL*[1] angeboten. Doch die zunehmende Vernetzung geht mit einer aufkommenden Datenflut einher, die ein Umdenken in der Verarbeitung großer Datenmenge verlangt. Es reicht schon lange nicht mehr, bekannte Verfahren der mathematischen Statistik nach vorgegebenem Schema anzuwenden und auf ein brauchbares Ergebnis zu hoffen. Lange Laufzeiten „naiv“ gestellter Anfragen und hohe Kosten für eine nicht automatisierte Auswertung erfordern neue Ansätze und Verfahren.

Mit dem Aufkommen neuer Techniken wie des *Data Warehousing* und neuer Disziplinen wie des *Data Mining* ist ein erster Schritt in diese Richtung getan. Doch trotz aller Popularität dieser Schlagworte sind die Möglichkeiten und Chancen dieser Techniken noch längst nicht ausgeschöpft. Dies liegt sicherlich zum Teil in einem Kommunikationsproblem zwischen den beiden am meisten beteiligten Disziplinen, die der Datenbanklehre und der mathematischen Statistk, begründet. Jedoch bedarf es auch neuer Konzepte und Ansätze, um das Potenzial dieser Techniken auszuschöpfen.

Eine Erweiterung des Aggregationsbegriff weg von einfachen deskriptiven Statistiken hin zu induktiven Verfahren der mathematischen Statistik ist ein Schritt in die richtige Richtung. Darüber hinaus bedarf es eines verstärkten Fokus auf die immer wichtiger werdende Verarbeitung von Datenströmen über großen Datenmengen. Kapitel 3 widmet sich diesem Problem und Kapitel 4 stellt einen ersten Ansatz zur Implementierung solcher Techniken vor.

Die folgenden Abschnitte geben eine kurze Motivation für die in dieser Arbeit entwickelten Techniken und Verfahren. Danach werden die wichtigsten Begriffe der Statistischen Methodenlehre in Bezug zu Datenbanken gebracht und es wird mit einigen populären Irrtümern und Fehlinformationen aufgeräumt.

[1] vgl. SQL92-Norm (ISO/IEC 9075:1992 bzw. DIN 66315) und [DD93] Seite 86 ff.

2.1 Motivation

Eine zentrale Anwendung in einem klassischen relationalen Datenbankmanagementsystem (DBMS) ist die Verarbeitung der von Benutzern in *SQL* formulierten Anfragen. Auf Grund der ständig anwachsenden Datenmengen, die durch das Internet verfügbar sind, besteht jedoch auch großes Interesse, Anfragen direkt auf der Datenquelle Internet zu formulieren. Daraus ergeben sich jedoch besondere Anforderungen.

2.1.1 Verarbeitung aktiver Datenquellen

Mit der fortschreitenden Entwicklung des Internets und der firmeninternen Netze ist es für jedermann möglich, selbst Informationen unabhängig von Anderen zu jeder Zeit zu veröffentlichen. Dazu kommen Millionen Nutzer der Netze weltweit. Somit ist es ein kaum zu lösendes Problem unter all diesen Daten relevante Informationen zu finden oder gar Änderungen auf bekannten Quellen lückenlos zu erfassen.

So muss zum Beispiel ein Nutzer, der regelmäßig Seiten verschiedener Nachrichtendienste besucht und an Änderungen, d. h. neuen Nachrichten, interessiert ist, diese Seiten regelmäßig manuell auf relevante Änderungen prüfen. Einfacher ist dieses allgemeine Problem des Überwachens einer Datenquelle zu lösen, wenn die Datenquelle bei Änderungen selbst aktiv wird. Somit braucht ein Beobachter nicht mehr in regelmäßigen Abständen Daten holen (*poll*), sondern wartet lediglich auf das Eintreffen neuer Daten, die die Quelle an ihn übermittelt (*push*).

Mit diesem auch unter dem Begriff *publish/subscribe* bekannten Prinzip gehen einige Vorteile gegenüber der konventionellen Methode einher. So ist mit geringerem Netzwerkverkehr zu rechnen, da nur bei Änderungen tatsächlich eine Kommunikation aufgebaut wird. Zudem werden die Empfänger zeitnah über Änderungen informiert und die kritische Größe der Aktualisierungsabstände zwischen zwei Anfragen wird eliminiert.

Einige Vorteile aktiver Quellen bzw. aktiver Datenbanken können auch im Kontext klassischer Anwendungen eines Datenbankmanagementsystems genutzt werden. Gerade Datenbanken im operativen Umfeld großer Konzerne verzeichnen extrem viele neue Einträge in kurzer Zeit, da in der Regel alle Transaktionen aufgezeichnet werden. Somit ist eine Überwachung (*monitoring*) dieser Daten sehr aufwendig. Klassische Systeme führen regelmäßige[2] vordefinierte Anfragen auf den Daten aus und starten eine anschließende statistische Untersuchung der Antworten. Die damit einhergehenden Probleme liegen auf der Hand. Zunächst sind die Zeitabstände zwischen den Anfragen kritisch. Zu kurze Intervalle bedingen zu hohe Kosten und zu viele gebundene Kapazitäten. Zu lange Abstände erhöhen das Risko, wichtige Änderungen in den Daten zu spät zu erkennen. Dazu kommt eine oft zu geringe Skalierbarkeit der Verfahren zur Erkennung von Änderungen[3].

Werden jedoch die Daten zum Zeitpunkt der Einfügeoperation direkt mit statistischen Methoden untersucht und relevante Ergebnisse direkt an die Nutzer weitergege-

[2] Oft auch durch *Trigger* ausgelößt

[3] vgl. dazu auch Abschnitt 2.2 und vor allem 2.3

ben, können zumindest die oben beschriebenen Probleme gelöst werden. Die eingehenden Daten können in einem solchen Szenario logisch als Datenstrom gesehen werden, die aktiv an einen „statistischen Operator“ geliefert werden. Eine Anforderung, die ein solcher statistischer Operator erfüllen muss, ist sicherlich eine nicht-blockierende Verarbeitung der Daten. Weitere notwendige und wünschenswerte Bedingungen für diese Operatoren werden in Kapitel 3 erläutert.

2.1.2 Operatorbäume und Zwischenergebnisse

Neben der Beobachtung aktiver Quellen gewinnen nicht-blockierende Operatoren auch in der klasssichen Anfrageverarbeitung immer mehr an Bedeutung. Trotz allem Leistungszuwachs heutiger Systeme werden die Laufzeiten komplexer Anfragen immer länger. Dies liegt in der noch stärker ansteigenden Datenflut, die die Basis für viele Anfragen bildet, begründet. Somit wird es immer wichtiger, dass komplexe Anfragen schnell Zwischenergebnisse liefern. Auf diese Weise können „falsch“ formulierte Anfragen ebenso schnell anhand der Zwischenergebnisse erkannt werden wie eine zu restriktive bzw. zu umfangreiche Anwortmenge.

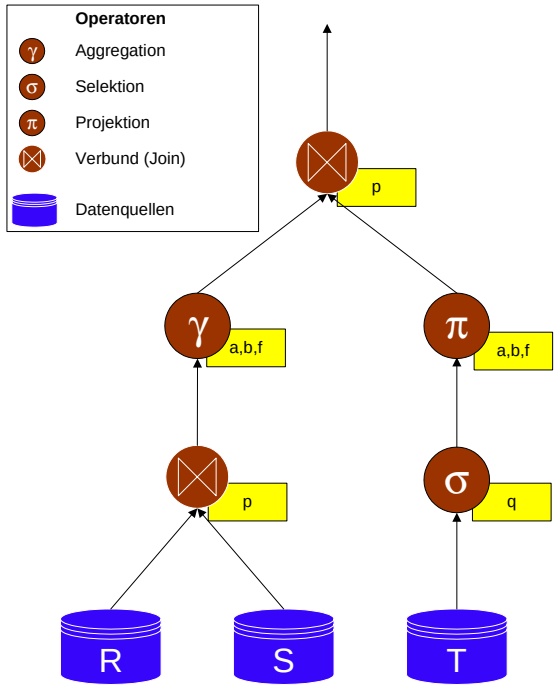

Abbildung 2.1: Operatorbaum über drei Datenquellen mit Aggregation

Solche neuen Anforderungen an die Anfrageverarbeitung in Datenbanksystemen bedingen auch neue Operatoren und Verfahren. So darf kein Operator Verwendung finden, der den Datenfluss des Operatorbaums blockiert, und jeder Operator muss eine Überwachung des Datenflusses erlauben. Setzt zum Beispiel ein Operator eine Sortierung der Eingabe voraus, so kann dieser erst dann Ergebnisse liefern, wenn alle Daten der Eingabe verarbeitet sind.

Abbildung 2.1 zeigt einen Operatorbaum über drei Datenquellen mit verschiedenen Operatoren. Selektionen und Projektionen sind in diesem Zusammenhang unproblematisch. Einen guten Überblick über nicht-blockierende Verbundoperationen im allgemeinen und deren Einsatz im kommerziellen Datenbankmanagementsystem *Informix* geben Hellerstein et al. in [HAR00]. Somit bleibt die Frage offen, ob Aggregationsoperatoren auf Datenströmen nicht-blockierend implementiert sein können. Diese Frage wird für die klassischen Aggregatsfunktionen im Kapitel 3 ausführlich beantwortet. Im Kontext des *Data Mining* ist es darüberhinaus interessant, weitere deskriptive Statisti-

ken des Datenstroms innerhalb eines Operatorbaums zu sammeln und zu verwerten. Ein Häufigkeitshistogramm über einer diskreten Eingabemenge oder ein Dichteschätzer für eine stetige Datenbasis können logisch ebenso als Aggregationsfunktion betrachtet und implementiert werden. Diese Sichtweise bedingt jedoch neben einem über einfache beschreibende Kennzahlen hinausgehenden Aggregationsbegriff auch Verfahren zur effizienten Erzeugung und Speicherung induktiver Aggregate.

2.2 Begriffsbildung

Praktische Anforderungen im Bereich von Datenbanksystemen führen oft zu pragmatischen und heuristischen Lösungen für eigentlich komplexe Probleme. Dagegen versucht die mathematische Statistik durchaus praktische Probleme mit Modellen zu lösen, die oftmals in der Realität nicht haltbar sind. Nun ist gerade *Data Mining* im Grenzbereich zwischen diesen beiden Disziplinen angesiedelt und versucht die unterschiedlichen Ansätze zusammen zu bringen. Zu diesen verschiedenen Herangehensweisen kommt die unterschiedliche Sprache bzw. Begriffsbildung der beiden Welten.

Doch *Data Mining* ist mehr als das Anwenden bekannter Verfahren der Statistik auf Probleme der Datenbankwelt. So existieren oft keine oder nur unzureichende Betrachtungen der Komplexität und der benötigten Ressourcen eines Verfahrens der mathematischen Statistik. Auch die Skalierbarkeit[4] der Methoden wird oft nicht oder nur unzureichend betrachtet. Viele Beispiele aus der Literatur zu nichtparametrischen Verfahren der mathematischen Statistik verwenden den *Old Faithful*-Datensatz mit insgesamt gerade 107 Beobachtungen[5] und eines der wichtigsten Bücher im Bereich der kernbasierten Methoden von Wand/Jones [WJ95] enthält gerade neun Seiten im Anhang über die (rechnergestützte) Berechnung der in diesem Buch behandelten Methoden[6]. Diese Defizite der mathematischen Statistik im Umgang mit extremen Datenmengen versucht das *Data Mining* mit Hilfe bekannter und neuer Ansätze aus dem Bereich der Datenbanken zu kompensieren.

Ein weiteres Problem stellt der unterschiedliche Sprachgebrauch der Disziplinen dar. Daher wird im folgenden Abschnitt kurz auf wichtige Begriffe und deren Bedeutung eingegangen.

2.2.1 Skalen und Ordnungen

Das Meßniveau der zu verarbeitenden Daten spielt eine wichtige Rolle in der Wahl des Verfahrens. Der Begriff des Messens in der Statistik ist weit gefasst und entspricht den Anforderungen der verwendeten Modelle[7]. So bezeichnet der Begriff „Messen“ die

[4] Skalierbarkeit bezeichnet die Fähigkeit von Verfahren, mit unterschiedlich großen Datenmengen umzugehen; d. h. es existiert keine Schwelle, ab der das Verfahren nicht mehr oder nur mit großem Aufwand läuft.

[5] Quelle: [Sco92] Seite 278

[6] vgl. [WJ95] Seite 182-191

[7] vgl. [BT94]

Zuordnung von Zahlen oder Symbolen zu Objekten unter Berücksichtigung von modellspezifischen Verträglichkeitsbedingungen. Um zu verhindern, dass beim Messen kein Informationsverlust bezüglich des Modells entsteht, aber auch nicht „zu viel“ gemessen wird, muss das verwendete Modell verträglich zur Meßskala sein. Damit ist die Zugehörigkeit der im Modell verwendeten Daten zu einer Skala wichtig für die Auswahl der Methoden die verwendet werden können.

In der Regel geht man in der Statistk von fünf verschiedenen Skalen aus, die zu zwei Kategorien zusammen gefasst werden können[8].

Die erste Kategorie der Topologischen Skalen oder auch Kategorial-Skalen umfasst die Nominal- und die Ordinalskala. Die Nominalskala hat klassifizierenden Charakter und umfasst Daten, die lediglich bezüglich einer Regel in Klassen zusammengefasst werden können, aber keine Wertung und Anordnung erlauben. In einer Ordinalskala werden Daten erfasst, die im Gegensatz zu nominalen Daten in eine oftmals natürliche Rangordnung gebracht werden können. So lassen sich zum Beispiel Blutgruppen lediglich zu Klassen zusammenfassen aber nicht anordnen, während Schulnoten eine Anordnung erlauben.

In der Kategorie der Kardinalskalen werden die Intervall-, Verhältnis- und Absolutskala zusammengefasst. Da hier ausschließlich numerische Daten erfasst werden, sind diese Skalen auch unter dem Begriff der metrischen Skalen zusammengefasst. Die Namen der einzelnen Skalen leiten sich aus den zulässigen Transformationen und Operationen auf den Daten der jeweiligen Skala ab. So hat im Gegensatz zur Ordinalskala der Abstand (das Intervall) zwischen zwei Werten einer Intervallskala einen Informationsgehalt. Während sich die Schulnote „sehr gut“ von der Note „befriedigend“ um zwei Notenpunkte (d. h. den Rang) unterscheidet, ist jedoch der Abstand zwischen den Noten nicht definiert. Der Abstand zweier in *Grad Celsius* gemessener Temparaturen hat aber sehr wohl eine Bedeutung. Ist zusätzlich ein eindeutiger Nullpunkt vorhanden, wird von einer Verhältnisskala gesprochen. Ein Beispiel für Daten auf einer solchen Skala sind Temperaturen gemessen in *Grad Kelvin*. Eine Absolutskala besitzt zusätzlich noch ein natürliches Einselement.

2.2.2 Diskret versus Stetig

Neben der Skala bzw. dem Messniveau der Daten ist die Einteilung von Daten in stetige und diskrete Merkmale sehr wichtig. Ein Merkmal heißt diskret, „wenn seine Ausprägungen eine diskrete Zahlenmenge bilden“[9]. Daten auf nominalem Niveau und im Allgemeinen Daten, die durch Zählen erhoben worden sind, sind typische Beispiele. Bei stetigen Merkmalen hingegen können auch alle Zwischenwerte zweier Merkmalsausprägungen realisiert sein.

[8] An manchen Fällen wird auch eine Skala mit vier verschiedenen Niveaus verwendet. Vergleiche dazu auch [BT94] und [BB98].

[9] Quelle: [BB98], Seite 7. Der Begriff „diskrete Zahlenmenge“ bezeichnet hier die Eigenschaft der Abzählbarkeit.

Aus dieser Definition stetiger Daten kann nun aber nicht gefolgert werden, alle in einem Rechnersystem gespeicherten Merkmale seien diskret, da nicht alle Zwischenwerte aufgrund der Rechnerarchitektur realisiert sein können. Zum einen muss hier zwischen den Daten selbst und ihrer Repräsentation in einem Rechnersytem unterschieden werden. Zum anderen haben stetige Daten im allgemeinen die Eigenschaft, dass trotz beschränkter Auflösung annähernd jeder Wert genau einmal auftritt. Treten bei diskreten Daten sehr viele verschiedene Merkmalsausprägungen auf, so werden diese Daten oft als stetig angesehen, um entsprechende Verfahren der Statistik anwenden zu können. Die Interpretation der Ergebnisse bezüglich solcher quasistetigen Merkmale bedarf jedoch großer Vorsicht.

Sicherlich ist es auch möglich, stetige Daten zu diskretisieren. Im Prinzip passiert dies bereits beim Speichern stetiger Daten im Computer (vgl. oben). Trotz allem werden in diesem Fall bei einer feinen Auflösung viele Klassen der diskreten Datenmenge keinerlei Werte enthalten. Eine grobe Diskretisierung hingegen produziert bereits einen oft nicht akzeptablen Fehler[10].

Eine gutes Beispiel für die Wichtigkeit der richtigen Wahl der Skala und der stetigen bzw. diskreten Merkmalsausprägung liefern Korn et al. in [KJJ99]. Die Autoren sprechen von einem bis zum fünffachen niedrigeren Fehler bei der Verwendung nichtparametrischer Verfahren auf einer Intervallskala gegenüber Histogrammen, die auf die Verarbeitung nominaler Daten spezialisiert sind. Dieser Vergleich kann nicht positiv für das auf Histogrammen arbeitende Verfahren ausfallen, da die Daten zuerst diskretisiert werden müssen. Der dabei entstehende Fehler dominiert in der Regel alle durch das Verfahren selbst entstehende Fehler.

Die Wahl der richtigen Skala und des dazu passenden Verfahrens ist somit grundlegend für den Einsatz von Verfahren der mathematischen Statistik.

2.2.3 Deskriptive und induktive Statistik

Neben den zu untersuchenden Daten bestimmt das verwendete statistische Modell maßgeblich die Vorgehensweise und Aussagekraft der Ergebnisse. Am Beispiel der bekanntesten Klasse von Verfahren der mathematischen Statistik in der Datenbankwelt wird dies deutlich. Histogramme können auf der einen Seite zur Beschreibung einer Grundgesamtheit von Daten einer beliebigen Skala genutzt werden. Mit ihnen sind Häufigkeitshistogramme ebenso möglich wie eine Diskretisierung und Beschreibung stetiger Daten. Diese Verwendung im Bereich der deskriptiven Statistik[11] führt zu der einfachen Regel, je mehr Histogrammklassen, desto besser die Beschreibung der Daten.

Es ist jedoch auch möglich, ein Histogramm über einer repräsentativen Teilmenge (Stichprobe) der Daten aufzubauen. Diese Anwendung der induktiven oder schließenden Statistik zielt darauf ab, ausgehend von einer Teilmenge Rückschlüsse auf die

[10] vgl. [KJJ99]

[11] Die deskriptive oder auch beschreibende Statitik versucht im allgemeinen eine Datenmenge mit Kennzahlen möglicht gut zu charakterisieren. Dies kann im Extremfall lediglich ein Wert oder eine vollständige Häufigkeitstabelle sein.

gesamte Datenmenge zu ziehen. Damit muss bei der Interpration eines Histogramms klar sein, welche Verwendung vorliegt. Zudem führt die im deskriptiven Fall eindeutige Regel „je mehr Klassen, desto besser die Qualität“, bei einer induktiven Verwendung nicht zum Ziel. Vielmehr existiert ein Zielkonflikt zwischen Varianz und Verzerrung des Schätzers bei der Wahl der Anzahl der Klassen. So existiert eine optimale Wahl der Zahl der Klassen zwischen den möglichen Extremen, die zu einem lokalen Minimum in der Fehlerfunktion führt[12].

Gerade im Bereich der Datenbanken und des *Data Mining* werden jedoch oft die Grenzen zwischen deskriptiven und induktiven Verfahren verwischt. Ein gutes Beispiel liefern die oft unreflektierten Interpretationen extrahierter Assoziationsregeln. Dabei wird ein Trainingsdatenbestand[13] auf mögliche und wahrscheinliche „wenn-dann“-Beziehungen zwischen unterschiedlichen Dimensionen untersucht. In den Datenbanken eines Unternehmens kann so zum Beispiel die Regel „Wenn ein Kunde in einem Alter über 30 Jahren ein Handy kauft, so kauft er mit einer Wahrscheinlichkeit von 60% auch ein Headset.“ gefunden werden. Diese Aussage gilt jedoch deskriptiv bestenfalls für den Trainingsdatensatz. Stellt der Trainingsdatensatz eine einfache Zufallsstichprobe einer Grundgesamtheit dar, so kann diese Regel induktiv auf diesen Datenbestand angewendet werden. Damit wird eine beschreibende Aussage bezüglich einer Teilmenge eines Datensatzes zur Einschätzung der Gesamtdaten genutzt. Dabei müssen mindestens das verwendete statistische Modell, die Auswahl und Größe des Trainingsdatensatzes und die Algorithmen zur Extraktion der Regeln bekannt sein, damit eine seriöse und theoretisch fundierte Aussage über die Qualität der Regel getroffen werden kann. Oft jedoch verwechseln ungeübte Anwender diese Regeln mit Tatsachen und viele Veröffentlichungen in diesem Gebiet verführen sie dazu.

2.2.4 Parametrische und nichtparametrische Statistik

Viele Verfahren der schließenden Statistik lassen sich grob einer von zwei[14] Kategorien zuordnen. Ein Modell, dass eine einer Familie von Funktionen zugehörige Modellfunktion verwendet, wird als parametrisch bezeichnet. Der Name rührt von der Tatsache her, dass durch die Kenntnis der Funktionenfamilie nur noch eine begrenzte Anzahl an Parametern bestimmt werden muss. Wird diese Modellannahme nicht getroffen und kann somit nicht einfach die Parameter des Modells bestimmen, gehört die Statistik in die Kategorie der nichtparametrischen Verfahren.

Einen Überblick über Vor- und Nachteile der unterschiedlichen Ansätze geben Bü-

[12] vgl. [BT94] Seite 258 f.

[13] Trainingsdatensätze werden vor allem aus zwei Gründen verwendet. Zum einen ist der gesamte Datenbestand oft zu groß um vollständig verarbeitet zu werden, zum anderen sollen einmal generierte Regeln nicht nur für den erzeugenden Datensatz, sondern auch für ähnliche und zukünftige Daten verwendet werden.

[14] Neben den parametrischen und nichtparametrischen Methoden wird oft eine dritte Kategorie, die der verteilungsfeien Statistiken, angeführt. Da diese Unterscheidung für die hier vorliegende Arbeit und dem Verständnis der verwandten Methoden nicht notwendig ist, sei auf weiterführende Literatur verwiesen ([Ras95], [BT94], [BB98]). Im Folgenden werden diese verteilungsfreien der Gruppe der nichtparametrischen Statistiken implizit zugeordnet.

ning und Trenkler in [BT94]. Zwei der wichtigsten Vorteile nichtparametrischer Methoden sind zum einen die in der Regel schwachen Annahmen. Damit stellt sich das Problem der Robustheit einer Statistik nicht in dem Maße wie bei parametrischen Modellen. Des Weiteren sind nichtparametrische Verfahren durch die fehlende Verteilungsannahme universeller und ohne besondere Vorabinfomationen einsetzbar. Damit sind sie die richtige Wahl in einer dynamischen Umgebung wie Datenbanken sie heute darstellen und im Rahmen dieser Arbeit betrachtet werden.

Die Nachteile gegenüber parametrischen Methoden zeigen sich jedoch oft erst auf den zweiten Blick. So ist es in der Regel schwierig, die Güte eines nichtparametrischen Tests in analytisch geschlossener Form anzugeben. Auch gibt es wenige Untersuchungen zur Robustheit von Modellen mit abhängigen Variablen und multivariaten Fragestellungen[15]. Oft sind nur asymptotische Aussagen unter gewissen Regularitätsbedingungen möglich.

2.3 Datenströme

Datenströme und deren Verarbeitung werden mit zunehmender Rechenleistung und Speicherkapazität moderner Systeme immer wichtiger. Heutzutage wird annähernd jede Transaktion eines Unternehmen aufgezeichnet. So produziert zum Beispiel der Walmart-Konzern täglich 20 Millionen Datensätze alleine durch Verkaufstransaktionen und Google wickelt 70 Million Suchanfragen ab[16]. Auch das Sammeln wissenschaftlicher Daten (zum Beispiel durch erdbeobachtende Satelliten) produziert eine nicht abreissende Flut von unstrukturierten Daten. Diese Daten können nicht mehr unreflektiert für eine spätere Bearbeitung gespeichert werden, da heutige Methoden des *Data Mining* nicht in der Lage, sind solche Datenmenge in akzeptabler Zeit zu verarbeiten[17]. Auch ein Beobachten (*monitoring*) oder Filtern (*filtering*) solcher stetig eingehenden Daten verlangt nach neuen zeitkritischen Verfahren. Braucht die Verarbeitung eines Elements eines Datenstroms länger als der Abstand zum nachfolgenden Element, so werden sich diese Verzögerungen summieren und früher oder später zum Abbruch der Verarbeitung führen[18].

Klassische Datenbanksysteme erwarten, dass alle Daten in persistenter Form vorliegen und verwaltet werden. Typischerweise wird eine Anfrage genau einmal über einem Datensatz ausgeführt. Dieser Ansatz ist aber nur für Daten, die sich wenig oder relativ selten änderen bzw. auf die unregelmäßig zugegriffen wird, zweckmäßig[19]. Ändern sich die Daten jedoch ständig bzw. werden immer neue Datensätze hinzugefügt, ist dieser Ansatz nicht mehr anwendbar. Vielmehr müssen die Daten bereits bei ihrer Ankunft, das bedeutet als Datenstrom, verarbeitet werden. Dieses Konzept des *continuous query processing* erzeugt damit einmal eine Anfrage und führt diese im logischen Sinne ständig über den Daten aus.

[15] vgl. [BT94] Seite 3 ff.
[16] vgl. [DH01]
[17] vgl. [DH01] und [BW01]
[18] vgl. [BW01]
[19] vgl. [BW01]

Auch in der Anfrageverarbeitung selbst werden Datenströme immer wichtiger. Damit noch zur Laufzeit einer Anfrage bereits Zwischenergebnisse produziert werden können, müssen die Daten als Datenstrom verarbeitet werden und alle Operation müssen nicht-blockierend sein.

Damit gibt es zwei verschiedene Typen von Datenquellen. Zum einen kann die Quelle selbst aktiv sein bzw. werden und Daten in das Anfragesystem *pushen*[20]. Des Weiteren kann das Anfragesystem nach Verarbeitung eines Datensatzes den nächsten anfordern. In dieser Arbeit wird sich auf die zweite Möglichkeit der passiven Quellen beschränkt, da der Fokus auf einer Anfrageverarbeitung der Daten mittels Operatoren liegt. Damit ist ein Datenstrom definiert als Folge von Daten $x_1, x_2, x_3, \ldots$ über einer Grundgesamtheit G, die nicht notwendigerweise persistent existieren muss. Vielmehr stellt der Datenstrom mittels des *ONC*-Prinzips (*open-next-close*) einen Zugriff auf die Daten zur Verfügung. Der Index i eines Datums kann damit auch als diskreter Zeitpunkt der Verarbeitung angesehen werden.

Sicherlich ist es möglich, mit Hilfe eines Puffers oder *Caches* die zeitkritische Komponente der Verarbeitung zu entschärfen. Jedoch ist ein solcher Pufferspeicher immer beschränkt und die genaue Zahl der zu verarbeitenden Daten ist oft nicht bekannt. Damit kann lediglich ein temporäres Gedächtnis mit Hilfe solcher Mechanismen erzielt werden. Zudem wird in dieser Arbeit davon ausgegangen, dass alle Daten lediglich einmal zur Verarbeitung zur Verfügung stehen[21]. Somit müssen alle Verfahren mit einem einzigen Durchlauf der Daten (*one-pass*) und einem festen und beschränkten Speicherplatz arbeiten.

Die Wichtigkeit der effizienten und skalierbaren Verarbeitung von Datenströmen ist in den letzten jahren vor allem im Bereich des *Data Mining* erkannt worden. So sind vor allem Verfahren zur automatischen Erkennung zusammengehöriger Daten (*Clustering*) auf Datenströmen[22] entwickelt worden, die nicht mehr eine persistente Form der Daten benötigen und mit beschränkten Ressourcen, d. h. konstantem Aufwand pro Datum, auskommen.

2.4 Data Mining und Mathematische Statistik

Der Begriff des *Data Mining* bekommt seit einigen Jahren vor allem auf dem Gebiet der Datenbanklehre immer mehr Bedeutung. Dabei ist eine Definition des Begriffs oder eine Abgrenzung gegenüber verwandten Gebieten sehr schwierig. Hand gibt in [Han98] eine mögliche Definition des Begriffs mittels „*(...) data mining as the process of secondary analysis of large databases aimed at finding unsuspected relationships which are of interest or value to the database owners*“.

Damit sieht Hand das Ziel des *Data Mining* in einer Wiederverwertung bereits

[20] Typischerweise im Falle des *continuous query processing*. Vergleiche dazu auch [BW01].

[21] Mit Ausnahme der gepufferten Datenwerte zu einem gegebenen festen Speicherplatz.

[22] O'Callaghan et al. liefern in [OMM+02] einen Überblick über aktuelle Verfahren und ihre Leistungsfähigkeit über Datenströmen.

gespeicherter Daten zum Auffinden bzw. Extrahieren neuer Erkenntnisse. Diese Sicht des Begriffs ist meiner Ansicht nach nicht weit genug gefasst, da dabei zu stark auf das Auffinden von „*wenn-dann*"-Regeln (*association rules*) abgezielt wird. Vielmehr kann jede Verarbeitung von großen Datenmengen, mit dem Ziel nicht triviale bzw. nicht offensichtliche Informationen über die Daten zu erhalten, als eine Art „schürfen" in den Daten angesehen werden. So kann das einfache Sammeln deskriptiver Kennzahlen einer Datenmenge bereits in das Gebiet des *Data Mining* gehören. Dazu kommt, dass nicht notwendigerweise eine Wiederverwertung von Daten stattfinden muss. Vielmehr können auch dort, wo Daten entstehen, Metainformationen direkt gesammelt werden.

Eine solche Sicht macht aber eine Abgrenzung des *Data Mining* gegenüber der Statistik schwierig. Oft wird kritisiert, dass bereits wohlbekannte Methoden der Statistik im Kontext des *Data Mining* neu wiedererfunden werden. Doch neben neuen Algorithmen der Wissensextraktion aus Datenbanken (*knowledge discovery in databases (KDD)*)[23] zeigen sich viele Unterschiede in den Disziplinen. Eine Diskussion der Gemeinsamkeiten und Unterscheide gibt Hand in [Han99]. Er stellt vor allem die nicht Abgeschlossenheit des *Data Mining* der Statistik gegenüber, d. h. das Suchen nach Metainformationen ist nie beendet, da neue Erkenntnisse immer wieder neue Fragen nach sich ziehen. Selbst dann nicht, wenn die Daten ideal[24] vorliegen.

2.5 XXL

Alle in dieser Arbeit vorgestellten und neu eingeführten Verfahren stehen in der plattformunabhängigen Java-Bibliothek XXL (*eXtensible and fleXible Library*) zur Verfügung. Die Bibliothek stellt sowohl ein *Framework* als auch fertige Applikationen unter Wahrung des Prinzips der physischen Datenunabhängigkeit bei der Anfrageverarbeitung zur Verfügung. Im Gegensatz zu klassischen, monolithisch aufgebauten Datenbanksystemen und ihren eingeschränkten Möglichkeiten, eigene Funktionalität zu integrieren, bietet XXL die einfache Möglichkeit, sowohl die Bibliothek selbst zu erweitern und um neue Algorithmen zu ergänzen, als auch eine transparente Anbindung an bestehene Systeme zu etablieren.

Die Bibliothek ist in verschiedenen Paketen (*packages*) organisiert, die unterschiedliche Funktionalitäten und Anwendungsgebiete repräsentieren. Die zentralen Pakete `xxl.cursors`, `xxl.collections`, `xxl.io`, `xxl.functions` und `xxl.predicates` stellen eine breite Basis für die Entwicklung spezialisierter Verfahren zur Verfügung. Dabei stehen sowohl hauptspeicherbasierte Datenstrukturen als auch Sekundärspeicherverfahren transparent zur Verfügung. In `xxl.relational` findet sich eine relationale Erweiterung der allgemeinen Anfrageoperatoren aus `xxl.cursors` basierend auf der JDBC 2.0 Spezifikation. Die Pakete `xxl.spatial`, `xxl.indexStructures`, `xxl.binarySearchTrees` und `xxl.statistics` enthalten Schnittstellen und Anwendungen für ein abgegrenztes Gebiet und nutzen die zentralen Pakete der Bibliothek

[23] Oft wird dieser Begriff als Synonym für *Data Mining* verwendet. Aus Sicht des Autors jedoch ist KDD vielmehr als ein Teilgebiet anzusehen (vgl. hierzu [Man97]).

[24] D. h. vollständig, richtig und abschließend

intensiv. In `xxl.util` finden sich Hilfsmittel und Schnittstellen, die keinem der anderen Pakete thematisch zuzuordnen sind.

Die Bibliothek wird stetig weiterentwickelt durch die Arbeitsgruppe Datenbanken an der Philipps-Universität Marburg unter Leitung von Herrn Professor Dr. Bernhard Seeger. Die Wurzeln des Projektes gehen zurück auf Herrn Professor Dr. Seeger und Herrn Jochen van den Bercken, der die zentralen Strukturen der Bibliothek maßgeblich gestaltet hat.

Zum Zeitpunkt dieser Arbeit liegt die Bibliothek in der Version 0.99b Build 2002-02-13 vor. Aktuelle Versionen, eine vollständige Dokumentation der API (*Advanced Programmers Interface*) und weitere Informationen finden sich unter [XXL].

Im Folgenden werden die wichtigsten Prinzipien der Bibliothek, die zum Verständnis der im Rahmen dieser Arbeit integrierten Verfahren notwendig sind, kurz erläutert. Eine umfassende Präsentation der Bibliothek mit Anwendungsbeispielen findet sich in [BBD+01].

2.5.1 Funktionen und Prädikate

Eines der zentralen Konzepte von XXL stellen die Schnittstellen `Function` und `Predicate` zur Verfügung. Prädikate sind eine Sonderform des allgemeineren Prinzips der Funktionsparameter, die ausschließlich boolesche Werte[25] zurückgeben. Alle im Folgenden getroffenen Aussagen zu Funktionen in XXL können daher direkt auf Prädikate übertragen werden. In vielen Implementierungen sind Objekte vom Typ `Function` und vom Typ `Predicate` mittels spezieller *Wrapper* austauschbar.

Funktionen und funktionale Datentypen bieten ein bekanntes und wichtiges Hilfsmittel, das Prinzip der Abstraktion in einer Programmiersprache umzusetzen. Funktionale Datentypen und damit auch Funktionale höherer Ordnung sind jedoch kein Bestandteil der Kernfunktionalität von Java[26]. XXL schließt diese Lücke elegant mit Hilfe der Schnittstelle `xxl.functions.Function`. Eine von `Function` erbende Klasse kann entweder direkt den allgemeinen n-stelligen Funktionsaufruf

```
public Object invoke( Object [] arguments);
```

überladen oder mit Hilfe eines in `Function` integrierten Rekursionsmechanismus lediglich einen der null-, ein- oder zweistelligen Methoden verwenden. Durch das in XXL integrierte Konzept stehen auch Kompositionen und damit Funktionen höherer Ordung direkt zur Verfügung. Im Rahmen der Bibliothek sind Funktionen oft mittels anonymer innerer Klassen definiert. Dieses spezielle Konzept, das in Java seit dem SDK 1.2 zur Verfügung steht, unterstützt die Implementierung einer Klasse an der Stelle, wo die Instanzen der Objekte dieses Typs erzeugt werden. Dadurch kann die Zahl der Klassendefinitionen stark eingeschränkt und Funktionalität dort plaziert werden, wo sie benötigt wird.

[25] Prädikate geben den Basistyp `boolean` zurück und können daher nicht von Funktionen erben, die Instanzen des allgemeinen Typs *Object* zurückgeben. Die parallele Entwicklung von Prädikaten und Funktionen ist durch einen Gewinn an Geschwindigkeit und einer klaren Trennung der Semantik motiviert.

[26] Die Bibliothek XXL ist zu 100% in reinem Java und kompatibel mit dem Java SDK 1.2 realisiert.

Instanzen von Funktionen können in XXL im Gegensatz zur mathematischen Definition einer Funktion einen Status besitzen. Dies ist vor allem für die später noch näher zu betrachtenden Aggregationsfunktionen wichtig. Doch auch mathematische Funktionen lassen sich mit dem oben erläuterten Konzept nachbilden. So kann zum Beispiel der Tangens (tan) einfach durch die Komposition des Sinus (sin) und des Kosinus (cos) mit Hilfe der Division (div) implementiert werden mittels

```
Function tan = div.compose( sin, cos);
```

Hierbei ist vorausgesetzt, dass die Instanzen `sin`, `cos` und `div` mit der entsprechenden Funktionalität bereits erzeugt worden sind.

Aufgrund der zentralen Bedeutung und der großen Anzahl an vordefinierten Klassen stehen für bereits vorhandene Implementierungen die beiden Pakete `xxl.functions` und `xxl.predicates` zur Verfügung. Einen Sonderfall stellt das Unterpaket `xxl.functions.aggregates` dar, das ausschließlich spezielle Funktionen zur Aggregation enthält. Im folgenden Abschnitt wird näher auf solche Aggregatsfunktionen eingegangen.

2.5.2 Iteratoren und Cursor

Die effiziente Verarbeitung und Manipulation von Datenströmen nimmt gerade in Datenbanksystemen einen immer größeren Stellenwert ein. Dabei ist es essentiell notwendig, die physische Unabhängigkeit zwischen der dem Datenstrom zugrunde liegenden Datenstruktur, den Metainformationen der Daten und der datenverarbeitenden Algebra zu gewährleisten. Java stellt bereits mit dem Konzept der *Iteratoren* eine Abstraktion zur Verarbeitung von Datenströmen unabhängig von der Datenquelle zur Verfügung. In XXL wird dieses Konzept aufgegriffen und um wichtige Funktionalität erweitert. Iteratoren sind gemäß der Java SDK 1.2 API (*Advanced Programming Interface*) definiert durch

```
interface Iterator{
    boolean hasNext();
    Object next();
    void remove();
}
```

Dabei wird die *open*-Phase des Datenstroms implizit nach der Instanziierung eines Objektes von Typ `Iterator` ausgeführt. Die *next*-Phase ist durch die Methode `Object next()` realisiert. Dabei zeigt die Methode `hasNext()` die Verfügbarkeit weiterer Objekte im Datenstrom an. Die optionale Methode `remove()` kann von implementierenden Klassen verwendet werden, um das letzte mit `next()` gelieferte Objekt im Datenstrom zu löschen. In XXL findet eine erweiterte Schnittstelle Verwendung, die eine explizite *close*-Phase unterstützt.

```
interface Cursor extends Iterator{
    Object peek();
    void update( Object o);
    void reset();
```

```
    void close();
}
```

Dabei müssen implementierende Klassen die Methode `close` zur Verfügung stellen, die eine Datenquelle schließt und alle verwendeten Ressourcen freigibt. Die zusätzlichen Methoden bieten optionale Funktionalität an, die nicht von allen Datenquellen unterstützt werden kann.

Das Paket `xxl.cursors` stellt eine Vielzahl von *Cursors* zur Datenmanipulation, Dateneingabe, Anfrageverarbeitung und Datenerzeugung zur Verfügung. So ist eine vollständige Algebra zur Anfrageverarbeitung mit benutzerdefinierten Prädikaten ebenso enthalten, wie die Möglichkeit, von verschiedensten Datenquellen zu abstrahieren. Wichtige spezielle *Cursor* sind zum Beispiel der *Mapper*, der die Objekte eines Datenstroms mittels einer durch den Benutzer spezifizierten `Function` transformiert und wiederum als Datenstrom zur Verfügung stellt. Dazu gibt es die Möglichkeit, Daten beliebig zu filtern, zu gruppieren, die Gruppierung wieder aufzulösen[27], etc. Alle in XXL integrierten Implementierungen sind soweit möglich *demand-driven*, d. h. ein Zugriff auf die Datenquelle bzw. eine Manipulation der Daten erfolgt erst bei der expliziten Anforderung der Daten durch die nächste *next*-Phase. Wenige Verfahren benötigen eine strikte (*strict*) oder halb-strikte (*semi-strict*) Implementierung.

Die Klasse `Aggregator` stellt einen speziellen Cursor (Aggregator-Cursor) zur Verfügung, der eine induktive Verarbeitung von Daten eines Datenstroms unterstützt. Ein *Aggregator* ist selbst wieder ein *Cursor*, der die Ergebnisse der Aggregation als Datenstrom liefert. Die zur Aggregation verwendete `Function` f muss dazu einen binären Funktionsaufruf der Form $a_n = f(a_{n-1}, x_n)$ unterstützen. Die Aggregationsfunktion erhält damit den „alten“ Aggregationswert a_{n-1} und das nächste Datum x_n und liefert den „neuen“ Aggregationswert a_n zurück.

Betrachten wir das Beispiel der Minimum-Funktion `Minimum` auf einem Datenstrom mit den Elementen $x_1, x_2, \ldots$ Die Funktion wird mit dem ersten Element des Datenstroms initialisiert und dann induktiv auf alle weiteren Daten angewendet. Für eine Folge $8, 6, 8, 5, 4, 5$ liefert somit ein Aggregator mit der `Minimum`-Funktion die Sequenz $8, 6, 6, 5, 4, 4$, indem das „alte“ Minimum mit dem nächsten Wert der Folge verglichen wird.

Für Funktionen, die eine mehrstufige Initialisierungsphase benötigen, muss eine Aggregationsfunktion nach außen hin dem Aggregator mitteilen, wann die Initialisierung abgeschlossen ist bzw. ob sie noch andauert. Dies geschieht durch die Rückgabe von `null`-Objekten. Das bedeutet, solange eine Aggregationsfunktion ein `null`-Objekt als Ergebnis liefert, ist es der Funktion noch nicht möglich, ein gültiges Aggregat zu berechnen. Des Weitern führt diese Konvention zu einer gleichen Anzahl von Elementen im zu aggregierenden und im aggregierten Datenstrom.

Das Beispiel der Aggregatsfunktion `LastNthAverage`, die den Mittelwert der letzten n Datenwerte zurückliefert, soll das Prinzip verdeutlichen. Der Aufruf

[27] Diese Operationen werden im Kontext von Datenbanksystemen auch mit *nest* und *unest* bezeichnet.

```
Aggregator agg = new Aggregator( input, new LastNthAverage( 3));
```

erzeugt eine Instanz eines Cursors, der den gleitenden Mittelwert der letzen drei Datenwerte des Datenstroms `input` liefert. Enthält der Eingabe-Cursor die Sequenz $5, 6, 7, 6, 6, 3, 4$, so liefert der Aggregator die Folge `null`, `null`, $6, 6.33, 6.33, 5, 4$. Zur Initialisierung werden somit zwei bereits verarbeitete Werte benötigt. Bei Verarbeitung des dritten Datums ist dann eine gültige Aggregation möglich.

Der Aggregator-Cursor bietet nicht nur die Möglichkeit, genau eine Aggregation auf genau einem Datenstrom vorzunehmen. Vielmehr ist es möglich, beliebig viele Aggregationen gleichzeitig in einem einzigen Durchlauf zu berechnen. Die gewünschten Aggregationsfunktionen werden dazu in einem Array zusammengefasst. Um gleichzeitig das Minimum der Daten, das Maximum und den gleitenden Durchschnitt der letzten drei Werte zu berechnen, genügt ein Aufruf der Form

```
Aggregator agg = new Aggregator(
    input,
    new Function [] {
        new Minimum(),
        new Maximum(),
        new LastNthAverage( 3)
    }
);
```

Dabei liefert der Aggregator-Cursor Objekte von Typ `Object []` zurück, die die aggregierten Werte in der Reihenfolge der Aggregationsfunktionen im Funktionen-Array enthalten. Selbst unterschiedliche Längen der Initialisierungsphasen sind so kein Problem, da jedes Aggregat unabhängig von den anderen berechnet wird. Enthält der Eingabedatenstrom im obigen Beispiel die Folge $5, 6, 7, 6, 6, 3, 4$, so liefert der Aggregator eine Sequenz von Tripeln der Form $\{5, 5, \texttt{null}\}, \{5, 6, \texttt{null}\}, \{5, 7, 6\}, \{5, 7, 6.33\}, \{5, 7, 6.33\}, \{3, 7, 5\}, \{3, 7, 4\}$.

Die Berechnung mancher Aggregate benötigt zusätzlich zum alten Aggregationswert Informationen über bisher verarbeitete Werte. So muss zum Beispiel die Aggregatsfunktion `Average`, die den laufenden Mittelwert eines Datenstroms berechnet, wissen, wieviele Datenwerte bereits in die Berechnung des Mittelwerts eingegangen sind. Solche Aggregatsfunktionen besitzen einen internen Status, der zur Berechnung herangezogen wird. Durch einen direkten Aufruf der Funktion ausserhalb des Aggregator-Cursors kann der Status der Funktion bezüglich des Aggregats inkonsistent werden.

Die klassischen Aggregate eines DBMS stehen in XXL im Paket `xxl.functions.aggregates` zur Verfügung. Weitere Aggregatsfunktionen, die komplexe Aggregate mit Hilfe von Verfahren der mathematischen Statistik berechnen, finden sich im Statistikpaket `xxl.statistics`.

2.5.3 Das Statistik-Paket

Im Paket `xxl.statistics` finden sich alle in dieser Arbeit entwickelten und zu Vergleichen herangezogenen Verfahren, die die Methoden der mathematischen Statistik direkt oder indirekt verwenden. So bietet zum Beispiel die Klasse `EmpiricalCDF` eine Im-

plementierung der empirischen Stichprobenfunktion, die direkt zur Selektivitätsschätzung von Fensteranfragen (*window queries*) auf metrischen Attributen genutzt werden kann[28]. Darüber hinaus stehen gängige Strategien Histogramme aufzubauen ebenso zur Verfügung wie die Möglichkeit, maschinenerzeugte Zufallszahlen (*pseudo random numbers*) beliebiger Verteilungen zu generieren. Zusätzlich bietet das Statistik-Paket die Möglichkeit, reellwertige Funktionen mit $f : \mathbb{R} \to \mathbb{R}$ effizient zu implementieren und deren Stammfunktion und Ableitung zur Verfügung zu stellen.

Das Unterpaket `xxl.statistics.kernels` beinhaltet elementare Verfahren und eine umfangreiche Basisfunktionalität zur effizienten Implementierung nichtparametrischer Verfahren auf Basis von Kernfunktionen[29].

[28] vgl. hierzu auch [Blo98] und [Kor99]

[29] Zu einer Einführung in die Theorie der Kernschätzer sei hier auf [Sco92] verwiesen.

Kapitel 3

Online Aggregation

Aggregate werden in einem relationalen Datenbankmanagementsystem (*DBMS*) üblicherweise zum Sammeln deskriptiver Statistiken großer bzw. komplexer Datensätze verwendet.[1] In den heute üblichen monolitischen und historisch gewachsenen Datenbankmanagementsystemen laufen diese Aggregationen dabei in der Regel in einem *Stapelprozess*, auch mit *batch-mode* bezeichnet, ab. Die Anfrage, zumeist in *SQL* formuliert, wird dabei an den Teil des DBMS übergeben, der für die Anfrageverarbeitung zuständig ist. Jetzt wird eine größere Menge Daten über einen längeren Zeitraum verarbeitet, und am Ende dieses Prozesses bekommt der Anwender, der die Anfrage formuliert hat, ein Ergebnis präsentiert, das nicht notwendigerweise die Erwartungen erfüllt[2].

Laut Haas [Haa97] und Hellerstein [Hel96] jedoch sind viele Anwender nicht am genauen Ergebnis der Aggregation interessiert. Das Ziel vieler Anfragen mit Aggregationen sei lediglich, einen groben Überblick über die Daten zu bekommen, um das weitere Vorgehen, d. h. weitere in die Tiefe gehende Anfragen des Anwenders, zu bestimmen. Hier genügt oft eine Antwort, die nur den ungefähren Wert einer Aggregation liefert. Dazu muss das Ergebnis der Aggregation jedoch zum einen sehr schnell zur Verfügung stehen, zum anderen muss eine Aussage über die Qualität des ungefähren Aggregates möglich sein. Vor allem im Bereich des *Data Warehousing* stellen Aggregationen einen großen Teil der zur Laufzeit formulierten und nicht vorberechneten Anfragen. Hier ist eine schnelle Antwort um so wichtiger, da Teile der möglichen Aggregate zum Zeitpunkt der Erzeugung des Data Warehouse vorberechnet werden, und somit in der Analysephase sofort zur Verfügung stehen.

Da die Präferenzen der Benutzer eines DBMS oder eines Data Warehouses sehr unterschiedlich sind, muss es dem Benutzer möglich sein, die Genauigkeit und den zur Berechnung verwendeten Aufwand der Aggregate zu bestimmen. Dies ist aber nur dann möglich, wenn die Aggregate *progressiv* berechnet und dem Anwender zur Laufzeit, d. h. *online*, präsentiert werden. Heutige DBMS sind mit solchen Anforderungen überfordert, da nicht nur die Anfrageverarbeitung angepasst, sondern auch von der Oberfläche

[1] vgl. u. a. [Haa97] und [Hel96]

[2] vgl. hierzu auch [HÖT88], [HÖT89] und [LNS90]

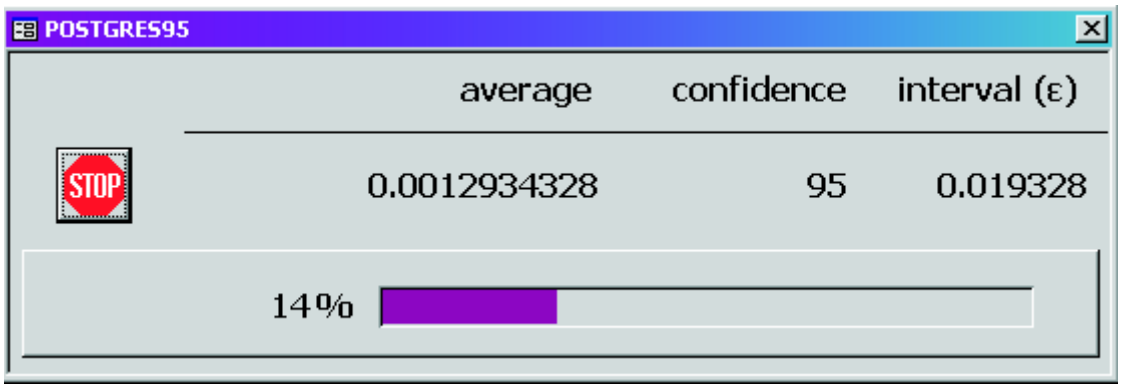

Abbildung 3.1: Mögliche GUI zur Unterstützung von Online Aggregation. Nachbildung der *POSTGRES95* Oberfläche aus [HHW97]

Werkzeuge zum Steuern und Überwachen der Aggregatsberechnung zur Verfügung gestellt werden müssen[3].

3.1 Grundlagen der Online Aggregation

Wie bereits oben gesagt sind die heutigen DBMS in der Regel nicht in der Lage, die Voraussetzungen der *Online Aggregation* zu erfüllen. Damit ein System Verwendung finden kann, müssen auf zwei unterschiedlichen Implemetierungsebenen Schnittstellen zur Verfügung stehen. Dies wäre zum einen das Benutzer-Frontend bzw. die graphische Benutzerschnittstelle (*GUI-graphical users interface*). Hellerstein et at. zeigen in [HHW97], wie eine solche GUI aussehen könnte. Zum einen sollte der aktuelle Wert des progressiv berechneten Aggregats zu sehen sein. Des Weiteren muss es möglich sein, die Anfrage zu stoppen. Dazu kommen noch Informationen über die Genauigkeit und über die Laufzeit des Aggregates. Denkbar wären auch prioritätsgesteuerte Aggregate in Kombination mit *GROUP BY* Klauseln. Abbildung 3.1 zeigt, wie für ein einfaches Mittelwert-Aggregat eine solche GUI aussehen könnte.

Aber nicht nur das Frontend muss Werkzeuge zur Unterstützung einer *Online Aggregation* zur Verfügung stellen, auch der Datenzugriff muss gewissen Anforderungen genügen. Damit klar wird, welche das sind und wieso sie gebraucht werden, muss man sich die Zusammenhänge zwischen *Online Aggregation* und Statistik etwas genauer ansehen.

Die induktive Statistik versucht, mit Hilfe einer Teilmenge der zu untersuchenden Grundgesamtheit Schlüsse über die Grundgesamtheit selbst zu ziehen und getroffene Aussagen zu begründen[4]. Eine induktive Statistik $T = T(X)$ hängt direkt von der Stichprobe X, also einer Teilmenge der Grundgesamtheit N, ab. Der Wert der Statistik berechnet sich jedoch mit Hilfe einer Realisation x der Teilmenge X. Damit ist

[3] vgl. [HHW97]
[4] vgl. Abschnitt 2.2.3

$T(X)$ eine Zufallsvariable und $T(x)$ eine Funktion mit einem konkreten Wert[5]. Damit nun allgemeine Aussagen über eine beliebige Realisation x einer solchen Teilmenge gemacht werden können, muss diese Teilmenge X gewissen Anforderungen genügen. Man sagt, die Stichprobe muss *i.i.d. (identically independent distributed)* sein. Das bedeutet, alle gezogenen Stichprobenwerte müssen die gleiche Verteilung besitzen und paarweise unabhängig sein. Während die erste Forderung nach gleicher Verteilung aller gezogenen Stichprobenwerte trivialerweise erfüllt ist, wenn alle Werte direkt aus der gleichen Grundgesamtheit gezogen werden, ist die Forderung nach Unabhängigkeit schwieriger zu erfüllen. Vor allem in DBMS mit Anfragesystemen, die auf Datenströmen arbeiten, ist diese Forderung im allgemeinen nicht zu erfüllen, da der Datenstrom in der Regel ein Zwischenergebnis in einem Anfragebaum ist. Somit hat eine Vorverarbeitung, oft eine *Selektion* oder ein *Join*, der Daten stattgefunden, die eine Sortierung oder Gruppierung der Daten bezüglich eines Attributs zur Folge hat. Damit sind die ersten n Datenwerte eines Datenstroms in der Regel nicht in zufälliger Reihenfolge und bilden somit keine *i.i.d.* Stichprobe des Datenstroms.

Damit wird klar, dass Qualitätsaussagen zur Laufzeit über die progressiv berechneten Aggregate nur dann möglich sind, wenn die Reihenfolge der Daten nicht mit dem Aggregat bzw. dem zu aggregierenden Attribut korreliert. Jedoch auch im Falle einer vorliegenden Sortierung bzw. einer Korrelation der Reihenfolge der Daten zu der Spalte, auf der das Aggregat berechnet werden soll, gibt es verschiedene Möglichkeiten, die Reihenfolge, in der die Daten verarbeitet werden, so zu variieren, dass eine fast zufällige Verarbeitungsreihenfolge erreicht wird. Hellerstein et al. nennen hier drei verschiedene Techniken[6], die Reihenfolge der Datenverarbeitung den Anforderungen der *Online Aggregation* anzupassen.

1. **Heap Scans**[7]
   In der Regel speichern Methoden auf Basis eines *heap file* Zugriffverfahrens die Daten in einer nicht spezifizierten Reihenfolge ab. Damit kann oft mit einfachen *heap scans* eine zufällige Reihenfolge der Daten gewährleistet werden. Vorsicht ist geboten, falls die logische Ordnung der Daten, die von einem *bulk insertion* Verfahren oder einer Clusteranalyse herrühren kann, mit den zu aggregierenden Daten korreliert. In diesem Fall sollte eine der anderen Techniken zum Einsatz kommen.

2. **Index Scans**
   Da ein Index in der Regel eine Ordnung, wie im Falle der B-Bäume, widerspiegelt, oder zumindest Daten in Gruppen anordnet, wie im Falle von *hash-basierten* Verfahren, eignen sich *index scans* oftmals nicht zur *Online Aggregation*. Sind die Daten, bezüglich derer der Index aufgebaut worden ist, jedoch nicht mit den zu aggregierenden Daten korreliert, liefern *index scans* einen schnellen und geeigneten Datenstrom.

[5] Zur Stichproben- und Testtheorie sei hier auf die einschlägige Literatur verwiesen, vgl. u. a. [Ras95] Seite 231 ff. und 300 ff. und [BB98].

[6] vgl. [HHW97]

[7] Der Name ist insofern irreführend, als keinesfalls die Datenstruktur *Heap* bei dieser Strategie Verwendung findet, sondern die Verwaltung der Daten ohne spezielle Struktur realisiert ist.

3. **Sampling from Indices**
   In [Olk93] und [OR95] sind von Olken bzw. Olken und Rotem verschiedene Techniken vorgestellt worden, vorhandene Indices zu nutzen, um effizient Stichproben in Datenbanken zu ziehen. Diese Verfahren sind sehr mächtig, um zufällige Reihenfolgen in Datenströmen zu gewährleisten, bezahlen dies aber oft mit einem hohen Aufwand an Rechenzeit. Zudem verhindern diese Techniken den Einsatz von Eingabe/Ausgabe-Optimierungen wie *pre-fetching* und Clusterung der Daten.

Damit sind laut Hellerstein et al. in den meisten Fällen *heap scans* effizient. Gerade im Falle von großen zu aggregierenden Datenmengen entsteht so kein Leistungsverlust. Ist das *heap file* jedoch sortiert oder die Qualität der Reihenfolge essentiell, so bieten *index scans* eine performante Alternative.

3.2 Bisherige Ansätze

Erste Ansätze, Anfragen an Datenbanken im allgemeinen und im besonderen Anfragen zur Berechnung von Aggregaten nicht dem *black box* Prinzip folgend als Stapelprozess zu sehen, finden sich recht früh in der Literatur. So entwickeln Hou et al. in [HÖT88] einen Schätzer, der auf einer speziellen Technik zum Ziehen einer Stichprobe basiert, der die Anzahl der Antworten einer Anfrage zu einem Ausdruck der relationalen Algebra liefert. Dieser Schätzer arbeitet jedoch nicht *online* im hier betrachteten Sinn, sondern vor Ausführung der eigentlichen Anfrage wird bereits ein Schätzwert berechnet. Der Anwender hat im weiteren Verlauf der Berechnung der Anfrage jedoch keine Möglichkeiten zum Steuern und Überwachen. Die Notwendigkeit zur Bestimmung eines Fehlers für die approximative Antwort der Anfrage wird von den Autoren als wichtig erkannt, sie geben jedoch nur asymptotische Fehleraussagen wie z. B. die Konsistenz des Schätzers an. In [HÖT89] gehen Hou et al. einen Schritt weiter und betrachten als einschränkenden Parameter die Zeit, die gebraucht wird, eine approximative Antwort zu finden. Diese *time-constraint* Verfahren liefern somit in einer gegebenen Zeitspanne die beste zu findende Approximation für eine Anfrage. Zu diesen Verfahren geben die Autoren auch Konfidenzaussagen der Ergebnisse an. Die Konfidenzlevel müssen jedoch vor Berechnung des Schätzers spezifiziert werden. Vielen dieser Verfahren[8] liegt das folgende Prinzip zugrunde:

> *Ziehe solange eine Stichprobe, bis das Ergebnis des Schätzers in einem vorher spezifizierten Konfidenzbereich liegt, dann höre auf.*[9]

Zur Laufzeit dieser Verfahren ist in der Regel keine Einflußnahme des Anwenders mehr möglich. Zudem terminieren diese Verfahren nach dem Erreichen ihrer *a priori* gegebenen Genauigkeit und bieten so keinerlei Möglichkeiten, den wahren Wert des Aggregats

[8] vgl. auch [LNS90] und [LNSS93]

[9] Auch hier muss natürlich die gezogenen Stichprobe *i.i.d.* sein, damit die Schätzverfahren gültige Ergebnisse liefern.

zu berechnen bzw. den bisherigen Aufwand zu nutzen um eine genauere Schätzung zu berechnen.

3.3 Statistik und Online Aggregation

Wie bereits in Abschnitt 3.1 angeklungen ist, kann man *Aggregation* und speziell *Online Aggregation* nicht isoliert von der mathematischen Statistik betrachten. Die klassischen Aggregate in einem DBMS nach der SQL92-Norm (ISO/IEC 9075:1992 bzw. DIN 66315) sind *Mittelwert*, *Summe*, *Minimum*, *Maximum* und *Anzahl*. Jedes dieser Aggregate ist eine Kennzahl für die Daten, auf die es angewendet worden ist. Somit bezeichnet ein Aggregat nichts anderes als eine *deskriptive Statistik* im Kontext eines DBMS. Allen der fünf klassischen Aggregatsfunktionen ist zudem gemein, dass sie mit nur einem Durchlauf durch die Daten progressiv berechenbar und damit ideal zur *online* Verarbeitung geeignet sind[10].

Jedes Aggregat ist somit eine Kennzahl für die verwendeten Daten. Der SQL/92-Standard[11] definiert nur die bereits erwähnten fünf Aggregate. Jedoch bieten alle heute etablierten DBMS weitere Aggregate wie z. B. *Varianz* und *Standardabweichung*. Die mathematische Statistik bietet für diese wohlbekannten Kennzahlen eine breite Palette an effizienten und leistungsstarken Schätzern. So ist zum Beispiel der wirksamste (oder auch *effizienteste*) Schätzer für den Mittelwert einer Grundgesamtheit G unter gewissen Voraussetzungen[12] der Mittelwert der *i.i.d.* Stichprobe $X = (X_1, \ldots, X_n)$ mit

$$T_n(X) = \frac{1}{n} \sum_{i=1}^{n} X_i. \tag{3.1}$$

Im Falle einer endlichen Grundgesamtheit der Größe N kann man nun die Größe der Stichprobe solange erhöhen bis $n = N$ gilt und erhält somit den wahren Mittelwert der Daten. Aber selbst im Falle einer unendlichen Grundgesamtheit bzw. einer stetigen Verteilung der Daten liefert Formel (3.1) den besten Schätzer für den Erwartungswert. Lässt man nun die Größe der Stichprobe gegen die Größe der Grundgesamtheit laufen, d. h. $n \to N$, so gilt

$$lim_{n \to N} T_n(X) = \mu \tag{3.2}$$

mit μ der Erwartungswert der den Daten zugrunde liegenden Verteilung. Damit ist $T_n(X)$ nicht nur effizientester Schätzer, sondern konvergiert sogar gegen den tatsächlichen Wert.

Zur Laufzeit ist jedoch $T_n(x)$ eine induktive Statistik, die auf einer Realisation $x = (x_1, \ldots, x_n)$ einer Stichprobe X basierend einen Schätzwert für den Erwartungswert der Verteilung bzw. den Mittelwert der Daten liefert. Damit ist $T_n(X)$ eine Zufallsvariable abhängig von der Stichprobe X. Konfidenzaussagen und andere Gütekriterien müssen sich somit im wesentlichen auf die mathematische Statistik stützen.

[10] vgl. Abschnitt 3.4

[11] vgl. SQL92-Norm (ISO/IEC 9075:1992 bzw. DIN 66315) und [DD93] Seite 86 ff.

[12] vgl. u. a. [BB98] und [Ras95]

3.4 Voraussetzungen zur Online Aggregation

Der Begriff der *Online Aggregation* ist maßgeblich von Hellerstein [Hel96] und Hellerstein et al. [HHW97] geprägt worden. In ihren Arbeiten wird zum ersten Mal auf die Notwendigkeit eingegangen, auch Aggregate durch den Anwender überwacht und gesteuert ablaufen zu lassen, damit die Berechnung nicht relevanter Aggregate frühzeitg erkannt und abgebrochen werden kann. Jedoch ist der Begriff der Aggregation und insbesondere der *Online Aggregation* bei Hellerstein et al. auf die bekannten deskriptiven Aggregate[13] beschränkt. Zudem fehlt eine klare Definition, wann ein Aggregat *online* berechnet wird bzw. berechnet werden kann, d. h. welche Voraussetzungen erfüllt sein müssen, damit es überhaupt möglich ist, ein spezielles Aggregat *online* zu berechnen. So ist es z. B. für den Mittelwert einer Grundgesamtheit klar wie er zu berechnen ist, bei der Varianz einer Grundgesamtheit stellt sich schon die Frage nach der rekursiven Berechenbarkeit und bei induktiven Statistiken, wie z. B. Dichteschätzern, ist es völlig unklar, welche Eigenschaften sie erfüllen müssen. Alleine die Möglichkeit einer rekusiven Berechnung reicht offensichtlich nicht aus, wenn nach der Verarbeitung der gesamten Datenmenge der tatsächliche Aggregatswert nicht mit dem Ergebnis der rekusiven Berechnung übereinstimmt.

Somit ist klar, dass nicht jedes beliebige Aggregat, welches sich rekursiv berechnen lässt, auch *online* berechnet werden kann. Einen Ansatz für die notwendigen Bedingungen zur *online* Berechnung eines Aggregats liefern folgende Kriterien.

1. Der Schätzer für das Aggregat muss induktiv, d. h. schrittweise berechenbar sein. Eine Steuerung der Schrittweite sollte möglich sein, da im Falle trivialer Aggregate wie dem Mittelwert der maximale Abstand der Rechenschritte gleich eins ist. Komplexe Aggregate wie eine Dichte bzw. der Schätzer zu einer Dichte benötigen höhere Schrittweiten, die in der Regel *a priori* festgelegt werden müssen.

2. Der Schätzer für das Aggregat muss bei einer endlichen Grundgesamtheit zum tatsächlichen Wert konvergieren bzw. der tatsächliche Wert des Aggregats muss sich nach Verarbeitung der gesamten Datenmenge aus dem aktuellen Schätzwert berechnen lassen. Bei unendlichen Grundgesamtheiten und induktiven Aggregaten reicht auch eine Konvergenz nach Wahrscheinlichkeit (*Konsistenz*).

3. Das Aggregat muss auf einem Datenstrom berechenbar sein, d. h. auf alle Datenpunkte darf höchstens einmal zugegriffen werden und die Datenwerte müssen in der „gelieferten“ Reihenfolge verarbeitet werden. Daraus folgt, dass ein wahlfreier Zugriff auf die Daten nicht möglich ist.

4. Der benötigte Speicherplatz zur Berechnung des Aggregats muss konstant sein und damit unabhängig von der Gesamtzahl der zu verarbeitenden Daten.

Zu diesen notwendigen Eigenschaften eines *Online-Aggregats* kommen noch einige, welche die Möglichkeiten und Vorzüge der online Aggregation erst richtig nutzbar machen.

[13] vgl. Abschnitt 3.3

1. In jedem Berechnungsschritt sollte eine Konfidenzaussage zur Güte des Schätzers zum aktuellen Zeitpunkt möglich sein.

2. Der Schätzer sollte bei einem Abbruch der Berechnung ein gültiges Ergebnis zum Zeitpunkt des Abbruchs liefern.

3. Der Schätzer sollte ohne bzw. mit möglichst wenigen Vorinformationen, wie z. B. die Gesamtzahl der Daten oder Extremwerte, auskommen.

Die klassischen Aggregate eines DBMS erfüllen in der Regel alle oben formulierten Kriterien, wie Haas in [Haa97] zeigt. Oft lassen sich sogar zusätzliche Gütekriterien wie deterministische Konfidenzintervalle angeben oder Selektionen auf den Daten berücksichtigen. In [Haa97], [HHW97], [Hel96] und [Hel97] werden für alle etablierten Aggregatsfunktionen entsprechende Aussagen entwickelt. Im folgenden Abschnitt werden die wichtigsten Aggregate mit ihren rekursiven Berechnungsvorschriften und laufenden Kofidenzintervallen vorgestellt.

3.5 Online Aggregation mit klassischen Aggregaten

In diesem Abschnitt werden die wichtigsten Aggregate eines DBMS mit den zur online Aggregation notwendigen Formel vorgestellt. Dazu seien $x_1, x_2, x_3, \ldots$ die Werte des zu verarbeitenden Datenstroms bzw. die Ergebnisse eines Ausdrucks angewendet auf die Werte des Datenstroms. Sei G die Grundgesamtheit, seien F die wahre Verteilung und f die wahre Dichte bzw. bei diskreter Verteilung die Wahrscheinlichkeitsfunktion von G und bezeichne P ein Wahrscheinlichkeitsmaß.

3.5.1 Mittelwert

Der Mittelwert einer Grundgesamtheit G der Größe N berechnet sich bekanntermaßen durch

$$\text{avg}(G) = \frac{1}{N} \sum_{i=1}^{N} x_i. \tag{3.3}$$

Der Mittelwert zum Zeitpunkt $n \leq N$ ist definiert durch

$$\text{avg}_n = \frac{1}{n} \sum_{i=1}^{n} x_i. \tag{3.4}$$

Diese Form ist aber gleichzeitig ein effizienter Schätzer für den Mittelwert bzw. bei stetig verteilten Grundgesamtheiten für den Erwartungswert der Verteilung. Der Mittelwert lässt sich sehr leicht induktiv berechnen mittels

$$\text{avg}_n = \frac{n-1}{n} \cdot \text{avg}_{n-1} + \frac{1}{n} x_n \text{ für } n \geq 1 \text{ und } \text{avg}_0 = 0. \tag{3.5}$$

In [Haa97] entwickelt Haas neben dem auf dem Zentralen Grenzwertsatz (ZGS) basierenden bekannten Konfidenzintervall zu (3.4) weitere Konfidenzaussagen. So gilt nach dem ZGS für ein Konfidenzintervall $[\text{avg}_n - \epsilon_n, \text{avg}_n + \epsilon_n]$ zum Zeitpunkt n zum Konfidenzniveau $p = 1 - \alpha$

$$\epsilon_n = \left(\frac{z_p^2 \hat{\sigma}_n^2}{n} \right)^{\frac{1}{2}} \tag{3.6}$$

mit z_p das $\frac{1+p}{2}$-Quantil der Standard Normal-Verteilung $N(0,1)$, d. h. mit $\Phi(z_p) = \frac{1+p}{2}$.

Diese Aussage gilt aber nur für große n, da bei unbekannter wahrer Varianz σ^2 die normierte t-Statistik erst bei großen n ausreichend durch die Standardnormalverteilung $N(0,1)$ approximiert wird[14]. Zur Bestimmung des Konfidenzintervalls aus (3.6) ist es notwendig einen Schätzer $\hat{\sigma}^2$ für die in der Regel unbekannte Varianz ebenfalls *online* zu berechnen. Dazu kann die rekursive Form aus Formel (3.19) verwendet werden.

Sind aber bei einer endlichen Grundgesamtheit G der Größe N zwei Werte a und b mit $a \leq \min(G)$ und $b \geq \max(G)$ bekannt[15], so folgt für den wahren Mittelwert μ aus

$$n \cdot \text{avg}_n - (N-n)a \leq N \cdot \mu \leq n \cdot \text{avg}_n + (N-n)b \tag{3.7}$$

direkt

$$\text{avg}_n - \underline{\epsilon}_n \leq \text{avg}_N \leq \text{avg}_n + \overline{\epsilon}_n \tag{3.8}$$

mit

$$\begin{aligned} \overline{\epsilon}_n &= \frac{(N-n)(b - \text{avg}_n)}{N} \\ \underline{\epsilon}_n &= \frac{(N-n)(\text{avg}_n - a)}{N}. \end{aligned}$$

Diese Form ist jedoch durch die benötigte Kenntnis über die Anzahl der Daten auf unendlichen Datenströmen nicht anwendbar. Zudem liefert Formel (3.8) durch seinen deterministischen Charakter in der Regel ein breiteres Intervall im Vergleich zur stochastischen Aussage von Formel (3.6).

Sind jedoch lediglich die zwei Werte a und b mit $a \leq \min(G)$ und $b \geq \max(G)$ bekannt, so kann mit Hilfe von Höffdings Ungleichung[16] ein konservatives Konfidenzintervall angegeben werden. Mit μ der wahre Mittelwert folgt

$$P\left(|\text{avg}_n - \mu| \leq \epsilon\right) \geq 1 - 2e^{\frac{-2n\epsilon^2}{(b-a)^2}} \tag{3.9}$$

für $\epsilon > 0$. Daraus folgt direkt, dass mit einer Wahrscheinlichkeit p das laufende Aggregat avg_n in einem Intervall ϵ_n liegt mit

$$\epsilon_n = (b-a)\left(\frac{1}{2n} \ln\left(\frac{2}{1-p} \right) \right)^{\frac{1}{2}}. \tag{3.10}$$

[14] Nach [BB98] ist jedoch $n \geq 30$ für die meisten Anwendungen ausreichend

[15] Im Allgemeinen sind solche a priori Informationen nicht bekannt. Heutige DBMS sammeln jedoch Metainformationen zu gespeicherten Daten, so dass die Annahme über die Kenntnis der Extremwerte für Datenströme mit einer Quelle in einem DBMS gemacht werden kann.

[16] vgl. [Hoe63]

Der Unterschied in den verschiedenen angegebenen Konfidenzintervallen liegt in dem zugrunde liegenden Ansatz begründet. Ein Intervall gemäß Formel (3.6) basiert auf einer Anwendung des Zentralen Grenzwertsatzes. Damit ist eine Verwendung eines solchen Intervalls dann zulässig, wenn n so klein ist, dass der Einfluss des Ziehens ohne Zurücklegen vernachlässigbar ist, aber andererseits auch groß genug, um den Satz anwenden so können.

Damit dürfen Konfidenzintervalle, die auf einer Anwendung des ZGS beruhen, nicht unreflektiert verwendet werden. Jedoch ist in der Regel ein solches Intervall erheblich schmaler als ein immer anwendbares konservatives Intervall basierend auf Höffdings Ungleichung.

3.5.2 Summe

Die laufende Summe zum Zeitpunkt n ist gegeben durch

$$\text{sum}_n = \sum_{i=1}^{n} x_i.$$

Zur Berechnung eines laufenden Schätzers für die Gesamtsumme muss jedoch die Größe N der Grundgesamtheit im voraus bekannt sein. Damit ist es nicht möglich, die Summe online für unendliche Grundgesamtheiten zu berechnen. Ist aber N bekannt, so kann mit der Beziehung

$$\text{sum}(G) := \text{sum}_N = N \cdot \text{avg}_N = \sum_{i=1}^{N} x_i.$$

zum Zeitpunkt n ein Schätzer für die Summe[17]angegeben werden mittels

$$\text{sum}_n^e = N \cdot \text{avg}_n = \frac{N}{n} \sum_{i=1}^{n} x_i. \tag{3.11}$$

Damit lassen sich die im Abschnitt 3.5.1 gemachten Aussagen direkt übertragen. Somit gilt nach dem ZGS für ein Konfidenzintervall $[\text{sum}_n^e - \epsilon_n, \text{sum}_n^e + \epsilon_n]$ zum Zeitpunkt n zum Konfidenzniveau $p = 1 - \alpha$

$$\epsilon_n = \left(\frac{\hat{z}_p^2 \hat{s}_{(n,\Sigma)}^2}{n} \right)^{\frac{1}{2}} \tag{3.12}$$

mit

$$\hat{s}_{(n,\Sigma)}^2 = \frac{1}{n-1} \sum_{i=1}^{n} (N x_i - \text{sum}_n^e)^2 \tag{3.13}$$

einem Schätzer für die mittlere quadratische Abweichung $s_{(n,\Sigma)}^2$ der Summe. Zu beachten ist, dass der Schätzer aus Formel (3.11) nicht die aktuelle Summe zum Zeitpunkt

[17]Im Folgenden kennzeichnet der Index s^e für eine Größe s einen Schätzer für die Größe.

n liefert, sondern die erwartete Summe aller x_i. Es gilt aber

$$\text{sum}_n = \frac{n}{N} \cdot \text{sum}_n^e,$$

d. h. kennt man den Berechnungszeitpunkt n, so kann aus dem Schätzer sum_n^e die aktuelle Summe sum_n berechnet werden.

Der Schätzer für die Varianz der Summe aus Formel (3.13) zum Zeitpunkt n lässt sich rekursiv berechnen mit[18]

$$(n-1)s^2_{(n,\Sigma)} = (n-2)s^2_{(n-1,\Sigma)} + \frac{(n-1)}{n}\left(\text{sum}^e_{n-1} - Nx_n\right)^2 \quad (3.14)$$

$$= (n-2)s^2_{(n-1,\Sigma)} + \frac{N^2(n-1)}{n}\left(\text{avg}_{n-1} - x_n\right)^2 \text{ für } n \geq 2. \quad (3.15)$$

Zudem lässt sich mit Hilfe von Formel (3.7) direkt ein deterministisches Intervall für die Summe angeben mit

$$\text{sum}_n - (N-n)a \leq \text{sum}_N \leq \text{sum}_n + (N-n)b. \quad (3.16)$$

3.5.3 Varianz und Standardabweichung

Einen effizienten Schätzer für die Varianz

$$\text{var}(G) = \text{var}_N = \frac{1}{N}\sum_{i=1}^{N}\left(x_i - \text{avg}_N\right)^2 \quad (3.17)$$

einer Grundgesamtheit G liefert die Stichprobenvarianz einer Stichprobe der Größe n mit

$$\text{var}_n^e = \frac{1}{n-1}\sum_{i=1}^{n}\left(x_i - \text{avg}_n\right)^2. \quad (3.18)$$

In [CGL83] geben die Autoren durch

$$(n-1)\text{var}_n^e = (n-2)\text{var}_{n-1}^e + \frac{1}{n(n-1)}\left(\text{sum}_{n-1} - (n-1)x_n\right)^2 \text{ für } n \geq 2 \quad (3.19)$$

einen numerisch stabilen Algorithmus zur *one-pass* Berechnung der Stichprobenvarianz var_n^e einer Stichprobe vom Umfang n an. Auch hier gibt der Schätzer wie schon bei der Summe zum Zeitpunkt n nicht die wahre Varianz der Daten, sondern lediglich eine erwartungstreue Schätzung dieser an. Jedoch durch

$$\frac{n-1}{n}\text{var}_n^e = \text{var}_n = \frac{1}{n}\sum_{i=1}^{n}\left(x_i - \text{avg}_n\right)^2$$

kann wie bereits bei der erwarteten Summe bei Kenntnis des Zeitpunkts n die Varianz aus dem Schätzwert berechnet werden.

[18] vgl. [Haa97] Seite 11

Auch beim zweiten Moment lässt sich mit Hilfe des ZGS eine Konfidenzaussage zum Zeitpunkt n für den Schätzer machen. So gilt für ein Konfidenzintervall $[\text{var}_n^e - \epsilon_n, \text{var}_n^e + \epsilon_n]$ zum Zeitpunkt n zum Konfidenzniveau $p = 1 - \alpha$

$$\epsilon_n = \left(\frac{z_p^2 (m_{(n,4)}^e - \text{var}_n^{e\,2})}{n} \right)^{\frac{1}{2}} \tag{3.20}$$

mit

$$m_{(n,4)}^e = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \text{avg}_n)^4 . \tag{3.21}$$

einem Schätzer für das vierte zentrale Moment der Daten zum Zeitpunkt n.

Zur Berechnung der laufenden Konfidenzintervalle muss nun das vierte zentrale Moment bzw. der Schätzer $m_{(n,4)}^e$ rekursiv berechenbar sein. Haas gibt in [Haa97] eine gut konditionierte Form durch

$$\begin{aligned} (n-1)m_{(n,4)}^e &= (n-2)m_{(n-1,4)}^e + \frac{4m_{(n-1,3)}^e(\text{avg}_n - x_n)}{n} \\ &+ \frac{6\text{var}_{n-1}^e(\text{avg}_n - x_n)^2}{n^2} + \frac{(n^2-3n+3)(n-1)(\text{avg}_n - x_n)^4}{n^3} \end{aligned} \tag{3.22}$$

an. Dabei bezeichnet $m_{(n,3)}^e$ einen Schätzer für das dritte Moment, welches ebenfalls rekursiv berechenbar ist durch

$$\begin{aligned} (n-1)m_{(n,3)}^e &= (n-2)m_{(n-1,3)}^e + \frac{3\text{var}_{n-1}^e(\text{avg}_n - x_n)}{n} \\ &- \frac{(n-2)(n-1)(\text{avg}_n - x_n)^3}{n^2}. \end{aligned} \tag{3.23}$$

Neben diesem auf dem Zentralen Grenzwertsatz basierenden Konfidenzintervall kann auch für die Varianz bei Kenntnis der Schranken a und b ein deterministisches Konfidenzintervall angegeben werden. So gilt[19] für jedes n für die Varianz $\text{var}(G)$ der Grundgesamtheit

$$\text{var}_n^e - \underline{\epsilon}_n \leq \text{var}(G) \leq \text{var}_n^e + \overline{\epsilon}_n$$

mit

$$\overline{\epsilon}_n = \lambda_n - \frac{N-n}{N-1}\text{var}_n^e \tag{3.24}$$

und

$$\underline{\epsilon}_n = \frac{N-n}{N-1}\text{var}_n^e. \tag{3.25}$$

Dabei gilt für

$$\lambda_n = \left(1 - \frac{n}{N}\right) \left(\frac{(b-a)^2}{4} + \frac{n}{N} \left(\max\left(\text{avg}_n - a, b - \text{avg}_n\right)\right)^2 \right). \tag{3.26}$$

Mit Kenntnis der Schranken a und b lässt sich zudem entsprechend der Summe und dem Mittelwert ein konservatives Konfidenzintervall mit Hilfe der Ungleichung von Hoeffding[20] angeben. So gilt $\text{var}_n^e \in [\text{var}_n^e - \epsilon_n, \text{var}_n^e + \epsilon_n]$ zu einem Konfidenzlevel

[19] vgl. [Haa97] Seite 5

[20] vgl. [Hoe63], [HHW97] und Formel (3.9).

a_n	ZGS	Konservativ	Deterministisch
avg_n	$\epsilon_n = \left(\frac{z_p^2 \text{var}_n}{n}\right)^{\frac{1}{2}}$	$\epsilon_n = \left(\frac{(b-a)^2}{2n} \ln\left(\frac{2}{1-p}\right)\right)^{\frac{1}{2}}$	$\underline{\epsilon}_n = \frac{(N-n)(\text{avg}_n - a)}{N}$ $\overline{\epsilon}_n = \frac{(N-n)(b - \text{avg}_n)}{N}$
sum_n^e	$\epsilon_n = \left(\frac{z_p^2 s^2_{(n,\Sigma)}}{n}\right)^{\frac{1}{2}}$	n. V.	$\underline{\epsilon}_n = (N-n)a$ $\overline{\epsilon}_n = (N-n)b$
var_n^e	$\epsilon_n = \left(\frac{z_p^2 (m^e_{(n,4)} - \text{var}_n^{e\,2})}{n}\right)^{\frac{1}{2}}$	$\epsilon_n = \frac{N^*(b-a)^2}{4(N^*-1)^2} + \frac{N^* B_n^{\frac{1}{2}}}{N^*-1}$	$\underline{\epsilon}_n = \frac{N-n}{N-1}\text{var}_n^e$ $\overline{\epsilon}_n = \lambda_n - \frac{N-n}{N-1}\text{var}_n^e$
std_n^e	$\epsilon_n = \left(\frac{z_p^2 (m^e_{(n,4)} - \text{var}_n^{e\,2})}{4n\sqrt{\text{var}_n^e}}\right)^{\frac{1}{2}}$	$\epsilon_n = \frac{N^*(b-a)^2}{4(N^*-1)^2} + \frac{N^* B_n^{\frac{1}{4}}}{N^*-1}$	$\underline{\epsilon}_n = \sqrt{\text{var}_n^e} - \left(\frac{n-1}{N-1}\text{var}_n^e\right)^{\frac{1}{2}}$ $\overline{\epsilon}_n = \left(\frac{n-1}{N-1}\text{var}_n^e + \lambda_n\right)^{\frac{1}{2}} - \sqrt{\text{var}_n^e}$

Tabelle 3.1: Konfidenzintervalle zu den wichtigsten klassischen Aggregaten

$\alpha = 1 - p$ mit

$$\epsilon_n = \frac{N^*(b-a)^2}{4(N^*-1)^2} + \frac{N^* B_n^{\frac{1}{2}}}{N^*-1}.$$

Dabei bezeichnet N^* eine untere Schranke für die Größe der Grundgesamtheit N[21] und

$$B_n = \frac{1}{\lfloor \frac{n}{2} \rfloor} \ln\left(\frac{2}{1-p}\right) \cdot \begin{cases} \frac{(b-a)^4}{8} \text{ für } 0 \le a < b \text{ oder } a < b \le 0 \\ \max\left(\frac{(b-a)^4}{8}, \frac{1}{2}\max\left(|a|, b\right)(|a|+b) - \frac{1}{4}\max\left(|a|, b\right)^2\right) \text{ für } a < 0 < b \end{cases}. \tag{3.27}$$

Da sich die Standardabweichung einer Grundgesamtheit direkt aus der Varianz berechnen lässt mit

$$\text{std}(G) = \sqrt{\text{var}(G)} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \text{avg}_N)^2}$$

und für den wirksamsten Schätzer der Standardabweichung

$$\text{std}_n^e = \sqrt{\text{var}_n^e} = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \text{avg}_n)^2}$$

gilt, können alle oben gemachten Aussagen zur Varianz auch für die Standardabweichung getroffen werden. Die entsprechenden Formeln finden sich neben denen für den Mittelwert, die Summe und die Varianz in Tabelle 3.1. Ein ϵ_n bezeichnet dabei ein zweiseitiges Intervall für das Aggregat a_n zum Zeitpunkt n der Form $a_n \in [a_n - \epsilon_n, a_n + \epsilon_n]$ zu einem Konfidenzlevel $\alpha = 1 - p$ und $\overline{\epsilon}_n$ und $\underline{\epsilon}_n$ die jeweils obere bzw. untere Grenze

[21] Ist N bekannt, so kann natürlich $N^* = N$ gesetzt werden.

	Mittelwert		Summe		Varianz & Std.-Abw.	
Aggregat	ident.		$\text{sum}_n = \frac{n}{N} \cdot \text{sum}_n^e$		$\text{var}_n = \frac{n-1}{n}\text{var}_n^e$	
Variablen	laufende	a priori	laufende	a priori	laufende	a priori
ZGS	n, var_n^e	-	n, $s^2_{(n,\Sigma)}(\text{avg}_n)$	N	n,m_4^e,$m_3^e(\text{avg}_n)$	-
Konservativ	n	a,b	n.V.	n.V.	n,B_n	a,b,N^*
Deterministisch	n	a,b,N	n	a,b,N	n,$\lambda_n(\text{avg}_n)$	a,b,N

Tabelle 3.2: Klassische Aggregate mit ihren laufenden und a priori benötigten Größen

des Intervalls. Die Formeln in der Spalte „ZGS“ entsprechen den mit Hilfe des Zentralen Grenzwertsatzes ermittelten Konfidenzintervallen. Die mit „Konservativ“ bezeichnete Spalte bezieht sich auf die mit Hoeffdings Ungleichung bestimmten Konfidenzen und in der Spalte „Deterministisch“ finden sich die analytisch abgeleiteten Intervalle bzgl. der Schätzer. Des Weiteren sind a und b a priori bekannte Schranken mit $a \leq x_i \leq b$ für alle x_i mit $i = 1, 2, \ldots, N$, N die Größe der Grundgesamtheit, N^* eine untere Schranke für ein unbekanntes N und z_p das p-Quantil der Standardnormalverteilung. Die zudem benötigten Größen λ_n, B_n, $m^e_{(n,4)}$ und $s^2_{(n,\Sigma)}$ finden sich in (3.26), (3.27), (3.22)+(3.23) bzw. in (3.14).

3.5.4 Eignung klassischer Aggregate zur Online Aggregation

Die in den obigen Abschnitten vorgestellten laufenden Schätzer für die wichtigsten Aggregatfunktionen erfüllen alle die in Abschnitt 3.4 formulierten Eigenschaften onlinefähiger Aggregate. Jedoch setzt die Berechnung einiger von Haas in [Haa97] und Hellerstein et al. in [HHW97] entwickelten Konfidenzintervalle die Kenntnis der Gesamtzahl der zu verarbeitenden Daten und das Wissen über a priori Schranken $a \leq x_i \leq b$ für $i = 1, 2, \ldots, N$ voraus. Diese sind somit nur eingeschränkt verwendbar. In Tabelle 3.2 sind für die Aggregate und den jeweiligen Schätzer noch einmal die Beziehung zwischen beiden Größen und die für Konfidenzaussagen benötigten a priori Informationen zusammengestellt. Alle Aggregate, die die Kenntnis über die Gesamtzahl der Daten N benötigen, können per se keine unendlichen Datenströme verarbeiten.

Die Spalten für die laufenden und a priori Variablen in Tabelle 3.2 geben die zur Berechnung der laufenden Konfidenzintervalle explizit (und implizit) mitzuführenden und im voraus benötigten Größen an. Die Zeilen beziehen sich jeweils auf eine der drei verschiedenen Verfahren zur Bestimmung der Konfidenzintervalle. Dazu ist in Zeile zwei noch einmal der Zusammenhang zwischen dem laufenden Schätzer und dem laufenden Aggregat dargestellt.

3.5.5 Klassische Online Aggregation in XXL

In XXL stehen alle von Haas in [Haa97] und Hellerstein et al. in [HHW97] vorgeschlagene Aggregate zur Verfügung. Während die Aggregatsfunktion `Variance` die laufende Varianz berechnet, gibt `VarianceEstimator` einen Schätzwert für die Varianz der

Daten des gesamten Datenstroms zurück. Dem gleichen Prinzip folgen die Klassen `StandardDeviation` bzw. `StandardDeviationEstimator` für die Standardabweichung und `Sum` bzw. `SumEstimator` für die laufende Summe. Da der laufende Mittelwert eines Datenstroms gleichzeitig auch effizientester Schätzer unter gewissen Voraussetzungen ist[22], gibt es für den Schätzer des Mittelwerts aller Daten eines Datenstroms keine eigene Implementierung.

Darüberhinaus stellt die Klasse `ConfidenceAggregationFunction` eine Schnittstelle für Aggregatsfunktionen zur Verfügung, die Konfidenzwerte für einen Schätzer berechnen. Zusätzlich bietet die Klasse eine Reihe von *Factory*-Methoden, die Instanzen von Schätzern mit verschiedenen Strategien zur Berechnung der Konfidenzintervalle erzeugen. Auf diese Weise sind alle oben aufgeführten Schätzer mit allen zur Verfügung stehenden Konfidenzaussagen implementiert. So erzeugt zum Beispiel der Aufruf

```
Aggregator agg = new Aggregator( input, new SumEstimator( N));
```

einen Cursor, der eine Folge von Schätzwerten für die Summe zu einem Eingabedatenstrom der Länge N ohne Berechnung eines Konfidenzintervalls liefert. Ein Aufruf der Form

```
ConfidenceAggregationFunction f
    = deterministicConfidenceSumEstimator( a, b, N);
Aggregator agg = new Aggregator( input, f);
```

erzeugt einen Cursor, der ebenso eine Folge von Schätzwerten zur Summe der Werte eines Datenstroms liefert, jedoch zusätzlich die Möglichkeit bietet, sich die Werte $\underline{\epsilon}_n$ und $\overline{\epsilon}_n$ zu einem deterministischen Konfidenzintervall ausgeben zu lassen.

3.6 Aggregation und Kennzahlen

Das Prinzip der Online Aggregation bietet erhebliche Vorteile bei der Bestimmung von Kennzahlen über Datenströmen gegenüber einer klassischer Stapelverarbeitung. Doch reicht es gerade im Bereich des *Data Mining* schon lange nicht mehr aus lediglich Kennzahlen einer Grundgesamtheit zu bestimmen. Vielmehr stellt sich immmer häufiger die Frage nach Lageparametern der Verteilung und der Verteilung bzw. Dichte der Daten selbst. Auch sind zunehmend Ausreißer (*outliers*) oder andere besondere Punkte oder Bereiche der Daten interessant.

Im folgenden Kapitel wird dieses Problem konkretisiert und versucht mit Hilfe einer speziellen Technik zum Ziehen von Stichproben eine Lösung zu erarbeiten.

[22] vgl. Abschnitt 3.3

Kapitel 4

Speicherbasierte Induktive Aggregation auf Datenströmen

In Kapitel 3 sind die notwendigen Kriterien für Verfahren zur Online Aggregation entwickelt und Verfahren zur online Berechnung klassischer Aggregate vorgestellt worden. In diesem Kapitel wird der Begriff der Online Aggregation über die Berechnung deskriptiver Kennzahlen über Datenströmen hinaus erweitert.

4.1 Induktive Statistiken und Stichproben

Zunächst soll in diesem Abschnitt erläutert werden, warum induktive Statistiken nicht in natürlicher Weise Online Aggregation unterstützen. Dies erfordert einige grundlegende Kenntnisse im Bereich der schließenden Statistik. In dieser Arbeit wird davon ausgegengen, dasss ein induktive Statistik $T(x; X)$ auf einer Realisation einer Stichprobe X mit den Stichprobenwerten $x_1, \ldots, x_n$ arbeitet. Zur Auswertung dieser Statistik T in einem Punkt x muss in der Regel die gesamte Stichprobe herangezogen werden. Zudem ist es oft nicht möglich, eine Statistik $T^{(t+i)}(x; X)$ mit $i \geq 1$ zu einem Zeitpunkt $t+i$ auf ihre Vorgänger $T^{(t+i-1)}(x; X), \ldots, T^{(t)}(x; X)$ zurückzuführen. Ein weiteres Problem ist die Berechnung eines theoretischen und eines praktischen Fehlers. Beide Fehler hängen in der Regel von der unbekannten Verteilung F der Daten ab, was eine exakte Berechnung nicht möglich macht. Damit stellen sich im Zusammenhang mit den für Online Aggregation in Abschnitt 3.4 formulierten Kriterien einige Fragen:

1. Wie kann man auf unendlichen Datenströmen die für induktive Statistiken benötigten *i.i.d.* Stichproben ziehen?

2. Wie kann die Konvergenz zum wahren Aggregat sichergestellt werden?

3. Ist die Statistik bei einem Abbruch zu einem beliebigen Zeitpunkt gültig, d. h. liefert sie eine Aussage zum Aggregat und eventuell zur Güte?

4. Gibt es Aussagen über die Qualität der Statistik zum aktuellen Zeitpunkt?

Die Liste der offenen Fragen ließe sich sicherlich noch weiter fortsetzen. Diese Fragen sind jedoch nicht allgemein zu klären und hängen direkt von dem verwendeten Schätzer ab. Lediglich zu Punkt 1 lässt sich eine Aussage treffen, die in Abschnitt 4.2 diskutiert wird.

Ein einfaches Beispiel für die Probleme induktiver Statistiken im Umfeld der Online Aggregation liefert ein Histogramm, dessen Klassen (*bins*) mit Hilfe einer *MaxDiff*-Strategie[1] aufgebaut worden sind. Bei diesem Typus von Histogrammen werden die Klassen so gewählt, dass zu einer vorgegebenen Anzahl von k *bins* die Grenzen der einzelnen Klassen gerade zwischen die Datenpunkte mit den $(k-1)$ größten Abständen fallen. Die Klassenbildung ist damit datenabhängig. Ein neu einzufügendes Element x kann somit die optimale Klassenbildung beeinflussen. Damit müsste man das Histogramm mit allen schon verarbeiteten Daten neu aufbauen, um eine Optimalität bzgl. der Daten und der gegebenen Anzahl von Klassen zu erhalten[2]. Eine Lösung wäre, die Klassenbildung a priori festzulegen; entweder aufgrund einer Modellannahme wie z. B. Gleichverteilung oder aufgrund einer Vorverarbeitung der Daten. Es ist jedoch sofort klar, dass beide Lösungen entscheidende Nachteile haben. Eine sinnvolle Verteilungsannahme ist oft nicht möglich, und eine Vorverarbeitung ist auf einem Datenstrom in einem Operatorbaum blockierend bzw. nicht möglich.

Ein anderes Problem stellen unendliche Datenströme dar. Während ein Mittelwert-Aggregat immer nur einen Wert speichern muss und somit der Speicherbedarf a priori bekannt ist, ist es bei Statistiken, die Schätzfunktionen liefern[3], in der Regel nicht gleichzeitig möglich den Speicherbedarf und ein Gütekriterium festzulegen. Dieses Problem wird in Kapitel 5 näher untersucht.

4.2 Speicherbasiertes Stichprobenziehen

Ein elementares Prinzip der induktiven Statistik ist die Verwendung einer möglichst kleinen Stichprobe einer Datenmenge. In klassischen Anwendungen ist die Größe der Datenmenge generell im voraus bekannt. Damit ist das Problem des Ziehens einer Stichprobe der Größe n aus einer Grundgesamtheit der Größe N zum Beispiel einfach zu formulieren:

Algorithmus 4.1 *Berechne n über der Menge $1, 2, \ldots, N$ gleichverteilte Zufallszahlen $z_i, i = 1, \ldots, n$ ohne Zurücklegen. Füge das Datum an der Stelle z_i in die Stichprobe ein.*

Ist jedoch die Größe der Grundgesamtheit nicht bekannt, so sind alle Verfahren zum Ziehen von Stichproben, die auf diese oder ähnliche Weise arbeiten, nicht ohne weiteres anwendbar.

[1] vgl. [Poo97]

[2] Durch die Verwendung eines B-Baumes für die Datenwerte und eines Heaps für die Abstände zweier benachbarter Punkte ist dieses Problem in $\Theta(\log n)$ zu lösen, benötigt jedoch für stetige Daten $\Theta(n)$ Speicher. Damit kann ein *MaxDiff*-Histogramm nur dann online verwendet werden, wenn die Partitionierung über n Datenwerten erfolgt und n unabhängig von der Gesamtzahl der Daten N ist.

[3] Zum Beispiel Schätzer für die Dichte einer Verteilung.

Ein erster Ansatz wäre, N mittels eines einfachen Durchlaufens der Grundgesamtheit zu bestimmen. Da jedoch im hier betrachteten Kontext die Grundgesamtheit in der Regel eine zur Laufzeit erzeugte Datenmenge als Ergebnis bzw. Ausgabemenge eines Operators und gleichzeitig die Eingabemenge eines weiteren Operators ist, kann eine solche Zählung der Datenmenge in der Regel nicht durchgeführt werden. Zum anderen muss der Operator, der die Grundgesamtheit als Ausgabemenge liefert, alle Teilergebnisse verarbeitet und weitergegeben haben. Damit blockiert eine Zählung der Daten den direkt folgenden Operator. Eine zusätzliche Schwierigkeit ist die notwendige Zwischenspeicherung der Daten. Die erzeugten Zwischenergebnisse in Operatorbäumen sind oftmals zu umfangreich, um sie im Hauptspeicher zu halten. Eine Auslagerung auf den Sekundärspeicher verbietet sich in der Regel aber aus Gründen der Performanz. Insgesamt ist dieser Ansatz nicht möglich oder zu ineffizient.

Eine zweite Möglichkeit besteht darin, ein Verfahren zum Ziehen einer Stichprobe anzuwenden, welches die Größe nicht im voraus wissen muss. Eine Klasse sehr interessanter und performanter Verfahren stellt Vitter in [Vit85] vor. Allen vier dort vorgestellten Verfahren ist gemein, dass sie in einem Schritt k mit $n \leq k \leq N$ eine gültige *i.i.d.* Stichprobe der Größe n für alle bis zum Schritt k verarbeiteten Daten liefern, ohne Kenntnis der Größe der Grundgesamtheit N. Die vier Verfahren werden mit *Typ R*, *Typ X*, *Typ Y* und *Typ Z* bezeichnet. Das Verfahren vom *Typ R* unterscheidet sich von den anderen Verfahren dahingehend, dass für jedes verarbeitete Datum eine Zufallszahl generiert werden muss. Alle anderen Verfahren „springen“ innerhalb der Daten zum jeweils nächsten Stichprobenkandidaten.

Sei im folgenden die Größe N der Grundgesamtheit G unbekannt, wobei N nicht notwendigerweise endlich sein muss. Sei ein Speicher (*Reservoir*) der Größe $n << N$ gegeben. Dabei stellt $x_1, x_2, x_3, \ldots, x_t, \ldots$ eine Folge von Objekten der Grundgesamtheit als Datenstrom dar[4].

Alle hier verwendeten Verfahren arbeiten in zwei Schritten:

1. Initialisierungsphase

2. Arbeitsphase

In der bei allen Verfahren identischen Initialisierungsphase wird das Reservoir mit den ersten n Elementen des Datenstroms gefüllt. Diese ersten n Elemente bilden trivialerweise eine gültige *i.i.d.* Stichprobe der ersten n Elemente des Datenstroms. In der Arbeitsphase wird nun für jedes weitere Element entschieden, ob es in die Stichprobe aufgenommen wird und welches bereits gezogene Element es aus der bisherigen Stichprobe verdrängt. Dabei soll die Invariante, dass das Reservoir eine *i.i.d.* Stichprobe der bisher verarbeiteten Elemente ist, erhalten bleiben. Im Folgenden werden die unterschiedlichen Stategien für die zweite Phase erläutert.

[4] vgl. Abschnitt 2.3

4.2.1 Typ R

Ein intuitiver Ansatz ist sicherlich, für jedes zum Zeitpunkt t ankommende Element neu zu entscheiden, ob es in die Stichprobe bzw. das Reservoir aufgenommen wird. Ist diese Entscheidung positiv ausgefallen, d. h. das Element wird aufgenommen, muss noch festgelegt werden, an welche Position es im Reservoir eingefügt wird und damit das dort schon vorhandene Element verdrängt. Der folgende Satz legitimiert dieses Vorgehen.

Satz 4.2 *Enthält ein Reservoir zum Zeitpunkt t eine gültige i.i.d. Stichprobe, wird ein Element zum Zeitpunkt $t+1$ mit einer Wahrscheinlichkeit von $\frac{n}{t+1}$ in das Reservoir aufgenommen und ist die „Verdrängungswahrscheinlichkeit" für jedes Element im Reservoir gleich $\frac{1}{n}$, so enthält das Reservoir auch zum Zeitpunkt $t+1$ eine gültige i.i.d. Stichprobe der bisher verarbeiteten $t+1$ Elemente.*

Beweis:
(Mittels Induktion über k mit $t = n + k$)
Der Beweis folgt im wesentlichen der Beweisidee von Vitter in [Vit85], ist jedoch an einigen Stellen ausführlicher und einfacher gehalten.

Zum Beweis des Satzes muss gezeigt werden, dass die Wahrscheinlichkeit im Reservoir zu liegen, für jedes x_i mit $i = 1, \ldots, t$ zum Zeitpunkt t für alle x_i gleich ist.
Sei nun die Zufallsvariable $X_i(t)$ definiert durch:

$$X_i(t) = \begin{cases} 1 & \text{das } i\text{-te Element liegt nach Bearbeitung} \\ & \text{von } t \text{ Elementen im Reservoir} \\ 0 & \text{sonst} \end{cases}$$

Für $k = 0$ ist nichts zu zeigen. Damit gilt für den

Induktionsanfang ($k = 1$):
Für $i = n + 1$ gilt:

$$P(X_{n+1}(n+1) = 1) = \frac{n}{n+1}$$

und für $i = 1, \ldots, n$ gilt:

$$\begin{aligned} P(X_i(n+1) = 1) &= P(X_{n+1}(n+1) = 0) + P(X_{n+1}(n+1) = 1) \cdot \frac{n-1}{n} \\ &= \frac{1}{n+1} + \frac{n}{n+1}\frac{n-1}{n} \\ &= \frac{1}{n+1} + \frac{n-1}{n+1} \\ &= \frac{n}{n+1}. \end{aligned}$$

Die Behauptung gelte für alle $j \leq k$, zu zeigen für $k+1$ ($k \longrightarrow k+1$):
Für $i = n + k + 1$ gilt:

$$P(X_{n+k+1}(n+k+1) = 1) = \frac{n}{n+k+1}$$

und für $i = 1, \ldots, n+k$ *gilt:*

$$\begin{aligned} P(X_i(n+k+1)=1) &= P(X_i(n+k)=1) \\ &\cdot \Big(P(X_{n+k+1}(n+k+1)=0) \\ &\quad + P(X_{n+k+1}(n+k+1)=1)\frac{n-1}{n}\Big) \\ &= \frac{n}{n+k}\left(\frac{k+1}{n+k+1} + \frac{n}{n+k+1}\frac{n-1}{n}\right) \\ &= \frac{n}{n+k+1} \end{aligned}$$

q.e.d.

Damit lässt sich nun ein Algorithmus für ein speicherbasiertes Verfahren zum Ziehen einer Stichprobe des Typs R angeben.

Algorithmus 4.3 *Sei* `R` *ein Reservoir der Größe* $n \leq N$ *und* t *der laufende Index der Elemente* `x[t]` *des Datenstroms. Speicherbasiertes Ziehen vom Typ R einer Stichprobe geht wie folgt:*

```
 1    LET t = 0
 2    WHILE( t <= n)
 3        INCREMENT t
 4        R[t] = x[t]
 5    WHILE( NOT Datenstromende OR Abbruch)
 6        INCREMENT t
 7        LET X = NEW RANDOM NUMBER with X ~ U[1,t]
 8        IF ( X <= n)
 9            LET Y = NEW RANDOM NUMBER with Y ~ U[1,n]
10            R[Y] = x[t]
```

mit $U[1,n]$ *einer (diskreten) Gleichverteilung über der Menge* $\{1, 2, \ldots, n\}$.

In der ersten Phase wird das Reservoir mit den ersten n Elementen x_t gefüllt. Die zweite Phase läuft solange weiter, bis der Datenstrom zu Ende ist, oder der Benutzer das Verfahren abbricht. Hier wird für jedes x_t zum Zeitpunkt t mit Hilfe einer Zufallszahl X entschieden, ob es in die Stichprobe aufgenommnen wird. Fällt die Entscheidung positiv aus, so wird mit Hilfe der Zufallszahl Y die Position im Reservoir bestimmt, und damit auch das Element, das verdrängt wird.

Da das Generieren einer Zufallszahl in der Regel viel CPU-Zeit beansprucht[5], sollte das unnötige Erzeugen von Zufallszahlen vermieden werden. In Algorithmus 4.3 wird jedoch für jedes Element x_t eine Zufallszahl erzeugt, egal ob dieses Element in die Stichprobe aufgenommen wird oder nicht. Alternativ ist es möglich, zu einem Zeitpunkt t einen Wert $S(n,t)$ zu bestimmen, der angibt, wie viele Elemente übersprungen werden sollen, bevor wieder ein Element in die Stichprobe eingefügt wird.

[5] vgl. [Vit85]

4.2.2 Typ X, Typ Y und Typ Z

Die Typen X, Y und Z verwenden den gleichen Algorithmus zum Ziehen einer Stichprobe. Sie unterscheiden sich lediglich in der Berechnung der zu überspringenden Elemente.

Algorithmus 4.4 *Sei R ein Reservoir der Größe $n \leq N$ und t der laufende Index der Elemente x_t des Datenstroms. Speicherbasiertes Ziehen vom Typ X, Y und Z einer Stichprobe ist möglich mittels:*

```
 1    LET t = 0
 2    LET S = 0
 3    WHILE( t <= n)
 4        INCREMENT t
 5        R[t] = x[t]
 6    S = NEW RANDOM NUMBER with S ~ S(n,t)
 7    WHILE( NOT Datenstromende OR Abbruch)
 8        INCREMENT t
 9        IF( S == 0 )
10            LET Y = NEW RANDOM NUMBER with Y ~ U[1,n]
11            R[Y] = x[t]
12            S = NEW RANDOM NUMBER with S ~ S(n,t)
13        DECREMENT S
```

Auch hier finden sich wieder die zwei Phasen wie in Algorithmus 4.3. Wichtig ist hier in der zweiten Phase die Berechnung der zu überspringenden Elemente $S(n,t)$. Die Verteilung von S hängt sicherlich von der Größe des Reservoirs n und dem Zeitpunkt t ab. Zudem muss für alle x_t, und damit auch für die übersprungenen Elemente, weiterhin Satz 4.2 gelten. Dies bedeutet, die Verteilung der Zufallsvariablen $S(n,t)$ muss so gewählt werden, dass die Wahrscheinlichkeit für ein x_t zum Zeitpunkt t in das Reservoir aufgenommen zu werden, weiterhin $\frac{n}{t}$ ist bzw. dass die Wahrscheinlichkeit im Reservoir zu liegen für alle x_i, $i = 1, \ldots, t$ gleich ist.

Satz 4.5 *Seien $a, b \in \mathbb{N}$ und gelte*

$$a^{\underline{b}} = a(a-1)\cdots(a-b+1) = \frac{a!}{(a-b)!} \quad \textit{und} \quad a^{\overline{b}} = a(a+1)\cdots(a+b-1) = \frac{(a+b+1)!}{(a-1)!}.$$

Ist nun $S(n,t)$ verteilt gemäß $F(s,n,t)$ mit

$$F(s,n,t) = 1 - \frac{t^{\underline{n}}}{(t+s+1)^{\underline{n}}} = 1 - \frac{(t+1-n)^{\overline{s+1}}}{(t+1)^{\overline{s+1}}} \tag{4.1}$$

bzw. besitzt $S(n,t)$ die Wahrscheinlichkeitsfunktion $f(s,n,t)$ mit

$$f(s,n,t) = \frac{n}{t+s+1}\frac{t^{\underline{n}}}{(t+s)^{\underline{n}}} = \frac{n}{t-n}\frac{(t-n)^{\overline{s+1}}}{(t+1)^{\overline{s+1}}}, \tag{4.2}$$

so liefert Algorithmus 4.4 zu jedem Zeitpunkt t eine i.i.d. Stichprobe der ersten t Elemente eines Datenstroms.

Beweis:
(Mittels Induktion über k mit $t = n + k$)
Es ist zu zeigen, dass die Wahrscheinlichkeit im Reservoir zu liegen für alle x_i, $i = 1, \ldots, t$ gleich ist. Für $k = 0$ ist nichts zu zeigen. Damit gilt für den

Induktionsanfang ($k = 1$):
Für $i = n + 1$ gilt

$$\begin{aligned} P(X_{n+1}(n+1) = 1) &= f(0, n, n) \\ &= \frac{n}{n+1} \end{aligned}$$

und für $i = 1, \ldots, n$ gilt:

$$\begin{aligned} P(X_i(n+1) = 1) &= P(X_{n+1}(n+1) = 0) + P(X_{n+1}(n+1) = 1)\frac{n-1}{n} \\ &= \frac{1}{n+1} + \frac{n}{n+1}\frac{n-1}{n} \\ &= \frac{1}{n+1} + \frac{n-1}{n+1} \\ &= \frac{n}{n+1} \end{aligned}$$

Die Behauptung gelte für alle $j \leq k$, zu zeigen für $k+1$ ($k \longrightarrow k+1$):
Für $i = n + k + 1$ gilt

$$\begin{aligned} P(X_{n+k+1}(n+k+1) = 1) &= \sum_{j=0}^{k} P(X_{n+j}(n+j) = 1) f(k-j, n, n+j) \\ &= \sum_{j=0}^{k} \frac{n}{(n+j)} \frac{n}{((n+j)+(k-j)+1)} \frac{(n+j)^{\underline{n}}}{((n+j)+(k-j))^{\underline{n}}} \\ &= \sum_{j=0}^{k} \frac{n}{(n+j)} \frac{n}{(n+k+1)} \frac{(n+j)^{\underline{n}}}{(n+k)^{\underline{n}}} \\ &= \frac{n}{n+k+1} \underbrace{\sum_{j=0}^{k} \frac{n}{(n+j)} \frac{(n+j)^{\underline{n}}}{(n+k)^{\underline{n}}}}_{=1} \\ &= \frac{n}{n+k+1} \end{aligned}$$

und für $i = 1, \ldots, n+k$ gilt

$$\begin{aligned} P(X_i(n+k+1)=1) &= P(X_i(n+k)=1) \\ &\cdot \Big(P(X_{n+k+1}(n+k+1)=0) \\ &\quad + P(X_{n+k+1}(n+k+1)=1)\frac{n-1}{n}\Big) \\ &= \frac{n}{n+k}\left(\frac{k+1}{n+k+1} + \frac{n}{n+k+1}\frac{n-1}{n}\right) \\ &= \frac{n}{n+k+1}. \end{aligned}$$

q.e.d.

Bemerkung 4.6 *Die Zufallsvariable $S(n,t)$ hat den Erwartungswert*

$$E\left(S(n,t)\right) = \frac{t-n+1}{n-1}.$$

Die drei Algorithmen vom Typ X, Y und Z unterscheiden sich nun in der Berechnung der gemäß $S(n,t)$ verteilten Sprungweite s.

Satz 4.7 [6] *Sei $u \sim U[0,1]$ rechteckverteilt auf $[0,1]$. Dann besitzt s mit*

$$s = \textit{ kleinste ganze Zahl mit } u \leq F(s,n,t) \textit{ und } s \geq 0 \tag{4.3}$$

die Verteilung (4.1).

Das Verfahren nach Typ X prüft nun für jedes $s \in \mathbb{N}$ inkrementell und beginnend mit $s = 0$ die Bedingung

$$1 - \frac{(t+1-n)^{\overline{s+1}}}{(t+1)^{\overline{s+1}}} \leq U,$$

wogegen Typ Y eine diskrete Variante des Newton Verfahrens verwendet. Hier sei zum genauen Vorgehen auf [Vit85] verwiesen. Typ Y wird im Folgenden nicht weiter betrachtet, da die Berechnung der Sprungweite s nach Typ Y häufig aufwendiger ist als eine sequentielle Suche nach Typ X[7].

Neben dem Basisalgorithmus für den Typ Z existieren drei weitere verbesserte Varianten mit unterschiedlichen Eigenschaften. Jede der drei Optimierungen stellt darauf ab, die Zahl der zu erzeugenden Zufallszahlen signifikant zu verringern, da der Autor im Erzeugen der benötigten Zufallszahlen den größten Kostenfaktor der Algorithmen sieht. Für die verschiedenen Möglichkeiten die Verteilung $S(n,t)$ effizient zu berechnen, sei hier auf [Vit85] verwiesen.

[6] Für den Beweis vgl. [Vit85]

[7] vgl. [Vit85] Abschnitt 5

4.2.3 Implementierung in XXL

In XXL stehen Typ R, Typ X und eine Variante von Typ Z zur Verfügung. Die Implementierung aller Varianten orientiert sich am Iteratorprinzip der Bibliothek. Alle Algorithmen arbeiten auf einem Datenstrom, stellen selbst wieder einen Datenstrom zur Verfügung und arbeiten damit *lazy* bzw. nicht blockierend.

Da ein speicherbasiertes Verfahren zum Ziehen einer Stichprobe zum Zeitpunkt t immer eine gültige *i.i.d.* Stichprobe der bisher verarbeiteten Daten liefert, liegt es nahe, das Reservoir als Aggregat der Daten zum Zeitpunkt t zu sehen. Demzufolge stehen die Verfahren als binäre Aggregatsfunktion[8] zur Verfügung, die zusammen mit dem *Aggregator*[9] verwendet werden. Die Klasse `xxl.functions.aggregates.ReservoirSample` besitzt eine innere Schnittstelle, welche von allen Varianten implementiert werden muss. Die Varianten sind alle als innere Klassen realisiert. Eine neue Aggregatsfunktion zum Ziehen einer Stichprobe der Größe n vom Typ R erhält man mittels:

```
Function sampling = new ReservoirSample(n, new ReservoirSample.RType(n));
```

Die Instanz `sampling` vom Typ `ReservoirSample` ist wie alle Aggregatsfunktionen eine binäre Funktion und erwartet als erstes Argument entweder `null` als Indikator, dass die Initialisierungsphase noch nicht abgeschlossen ist, oder ein `Object[]` mit der aktuell gültigen Stichprobe. Das zweite Argument ist das Datum des Datenstroms zum aktuellen Zeitpunkt.

Da somit die Verfahren selbst als binäre Funktion implementiert sind, können sie mit anderen Aggregatsfunktionen kombiniert werden und gemeinsam auf dem gleichen Datenstrom arbeiten. Um z. B. gleichzeitg den Mittelwert (`Average`), die Anzahl der Daten (`Count`) und eine Stichprobe der Größe n nach Typ X auf dem Cursor `input` zu berechnen, genügt folgender Aufruf:

```
Aggregator agg = new Aggregator{
    input,
    new Function []{
        new Average(),
        new Count(),
        new ReservoirSample( n, new XType( n))
    }
};
```

Ein solcher Aggregator liefert mit jedem `next`-Aufruf ein `Object`-Array, das im ersten Feld den aktuellen Mittelwert und im zweiten die aktuelle Anzahl enthält. Das dritte Feld enthält entweder `null`, wenn das Reservoir noch nicht vollständig initialisiert ist, oder ein `Object[]` mit einer *i.i.d.* Stichprobe.

Soll lediglich eine Stichprobe einer Datenmenge erzeugt werden, so bietet sich die Verwendung des speziellen Cursors `xxl.cursors.ReservoirSampler` an. Dieser Cursor ist ein „Makro“ zur einfachen Verwendung der `ReservoirSample`-Funktion. Als Argumente erwartet der `ReservoirSampler` den Eingabedatenstrom `input`, die Größe

[8] vgl. Abschnitt 2.5.1
[9] vgl. Abschnitt 2.5.2

Verfahren	Mittlere Anzahl zu erzeugender Zufallszahlen	Mittlere CPU-Zeit
R	$N - n$	$O(N)$
X	$\approx 2n \ln \frac{N}{n}$	$O(N)$
Y	$\approx 2n \ln \frac{N}{n}$	$O\left(n^2 \left(1 + \log \frac{N}{n} \log\log \frac{N}{n}\right)\right)$
Z	$\approx 3n \ln \frac{N}{n}$	$O\left(n \left(1 + \log \frac{N}{n}\right)\right)$

Tabelle 4.1: Theoretische Performanz der Verfahren R, X, Y und Z (Quelle: [Vit85] Abs. 1)

der Stichprobe `n` und einen Indikator `type` vom Typ `int` für das zu nutzende Verfahren.

```
Cursor cursor = new ReservoirSampler( input, n, type);
```

Die volle Funktionalität, wie eine Kombination mit anderen Aggregatsfunktionen oder die Verwendung eines performanten Zufallszahlengenerators[10], erreicht man jedoch nur bei Verwendung der `ReservoirSample`-Funktion.

4.2.4 Vergleich der verschiedenen Ansätze

Die theoretischen Kosten der Verfahren aus den Abschnitten 4.2.1 und 4.2.2 nach Vitter finden sich in Tabelle 4.1. Das Laufzeitverhalten der verschiedenen Verfahren ist vom Autor empirisch bestimmt worden und entspricht dem, was nach der Analyse der theoretischen Kosten zu erwarten war. Jedoch zeigen Zeitmessungen der in XXL implementierten Verfahren ein anderes Bild. Hier unterscheiden sich die verschiedenen Verfahren sowohl in ihrem zeitlichen Aufwand als auch in der Qualität der Stichprobe kaum. Für den Test sind drei künstlich erzeugte Datenmengen mit jeweils 1 Million Datenwerten verwendet worden. Die Datenmengen waren binomial- ($\sim B(n = 100, p = 0.4)$), Poisson- ($\sim P(\lambda = 30)$) und normalverteilt ($\sim N(\mu = 0, \sigma = 1)$). Aus jeder Grundgesamtheit sind jeweils mit den drei Typen R, X und Z 50 Stichproben mit jeweils 100, 500, 1000, 2000, 5000, 10000, 20000, 25000 und 50000 Stichprobenwerten gezogen worden. Alle Grundgesamtheiten lagen vollständig im Hauptspeicher vor, um einen Einfluss von Sekundärspeicherzugriffen zu vermeiden. Abbildung 4.1 zeigt die benötigte mittlere Zeit in Millisekunden (ms) zur Erzeugung der Stichproben mit jeweils gleicher Größe. Hier sieht man, dass die Wahl des Verfahrens keinen signifikanten Einfluss auf die Laufzeit hat. Die erzeugten Stichproben sind neben der Zeitmessung einer „Qualitätskontrolle“ unterzogen worden. Dazu sind verschiedene Kennzahlen (u. a. Mittelwert, Varianz, Schiefe, Minimum, Maximum) der Stichproben berechnet und mit den entsprechenden Werten der Grundgesamtheit verglichen worden. Die Statistiken zeigen keinerlei Auffälligkeiten in allen berechneten Kennzahlen. Exemplarisch zeigt Abbildung 4.2 die relativen Fehler der Mittelwerte der Stichproben. Selbst eine Stichprobe der Größe 100, dies entspricht 0.01 Prozent der Grundgesamtheit, zeigt lediglich eine mittlere Abweichung von 0.16 Prozent. Über die Qualität der Verfahren in einem

[10] Zum Beispiel mit Hilfe der Schnittstellen `xxl.util.ContinuousRandomWrapper` und `xxl.util.DiscreteRandomWrapper` bzw. der Implementierungen `connectivity.colt.ColtContinuousRandomWrapper` und `connectivity.colt.ColtDiscreteRandomWrapper`, die die Möglichkeit bieten Zufallszahlengeneratoren anderer Bibliotheken wie z. B. Colt in XXL einzubinden.

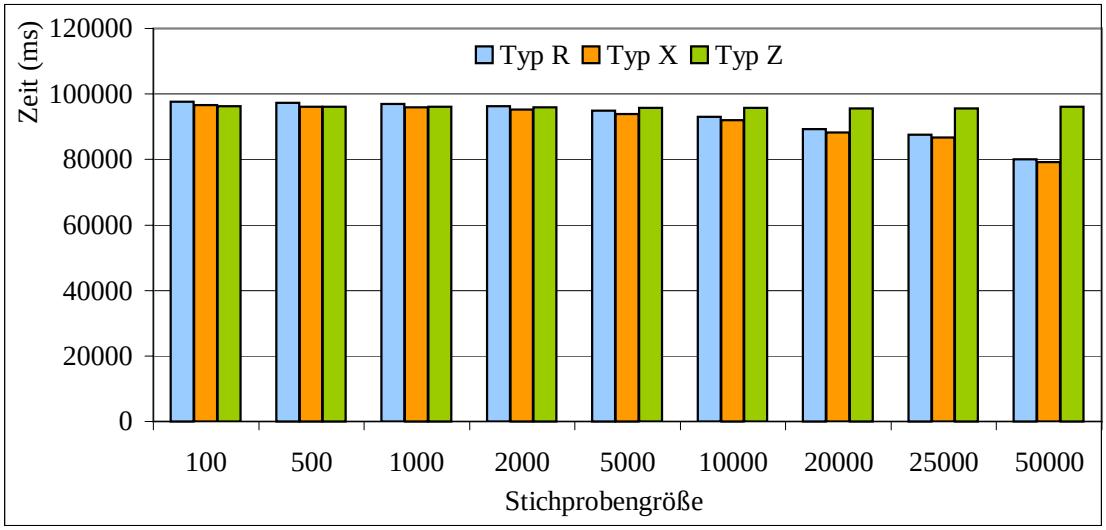

Abbildung 4.1: Mittlere Laufzeit der *Reservoir Sampling* Techniken vom Typ R, X und Z

Vergleich untereinander lässt sich somit keine Wertung zugunsten eines Verfahrens abgeben.

Der Grund für die Unterschiede in den theoretischen Qualitäten und den Testergebnissen der Verfahren liegt vor allem in zwei Punkten begründet. Zum einen sind heutige Zufallszahlengeneratoren viel performanter[11] und die Leistung heutiger Prozessoren erheblich größer. Damit entfällt das Argument des Autors, dass die gesamten Kosten inklusive der Eingabe- und Ausgabeoperationen von der benötigten CPU-Zeit dominiert werden. Zum anderen ist XXL eine Bibliothek, die generisch im Kontext von Datenbanksystemen arbeitet. Damit werden die Kosten der verschiedenen Verfahren durch solche Kosten dominiert, die durch die generische und offene Implementierung anfallen. In den folgenden Algorithmen, die speicherbasiertes Ziehen einer Stichprobe verwenden, wird, soweit nicht anders vermerkt, immer das Verfahren vom Typ X eingesetzt, um eine Vergleichbarkeit der Verfahren zu gewährleisten.

4.3 Induktive Statistiken auf Datenströmen

In Abschnitt 4.1 ist die Beziehung einer induktiven Statistik zu der Stichprobe einer Grundgesamtheit aufgezeigt worden. Wie eine *i.i.d.* Stichprobe effizient auf einem Datenstrom gezogen werden kann, zeigt Abschnitt 4.2. Jetzt stellt sich die Frage, wie kann mit Hilfe des speicherbasierten Stichprobenziehens ein beliebiger Schätzer über einem Datenstrom etabliert werden.

Nach Abschluss der Initialisierungsphase enthält das Reservoir der Größe n eine

[11] Die Java-Bibliothek Colt [Col] stellt neben einigen anderen den sehr performanten Zufallszahlengenerator *Mersenne Twister* zur Verfügung.

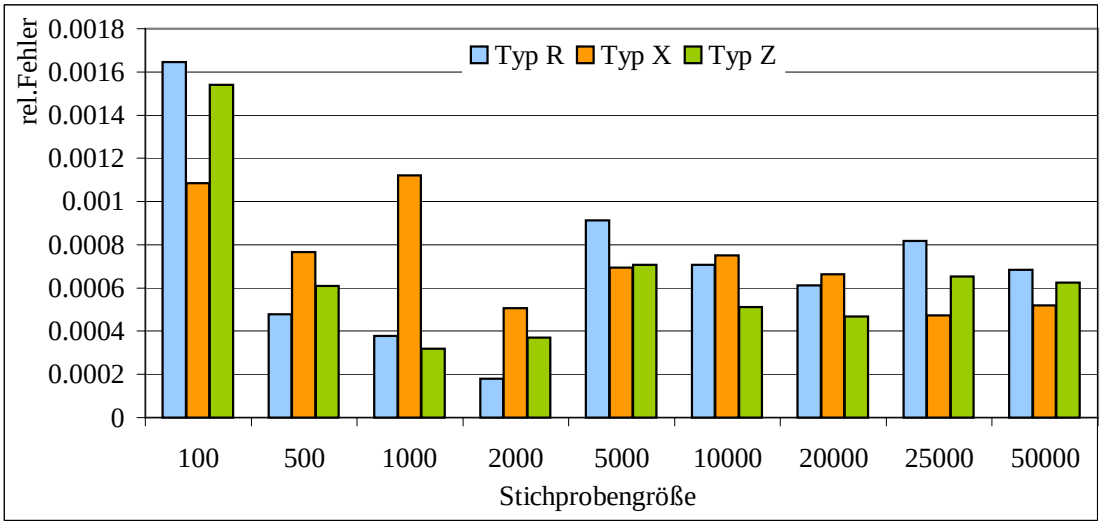

Abbildung 4.2: Mittlerer relativer Fehler zum Mittelwert der *Reservoir Sampling* Techniken vom Typ R, X und Z

gültige *i.i.d.* Stichprobe der bisher verarbeiteten Datenobjekte. Damit kann das Reservoir in einer induktiven Statistik als Stichprobe der ersten k verarbeiteten Daten verwendet werden. Abbildung 4.3 zeigt einen Datenstrom und das Reservoir, welches die Stichprobe enthält, zum Zeitpunkt k. Der laufende Schätzer $T_k(\cdot; X_k)$ basiert dabei auf dem Reservoir X_k zum Zeitpunkt k. Wichtig ist, dass die Statistik T_k einen gültigen Schätzer zum Zeitpunkt k, und in der Regel[12] nicht für den gesamten Datenstrom liefert. Hier bieten sich aufgrund der Voraussetzungen Verfahren der nichtparametrischen Statistik an[13]. Eine Klasse effizienter und in der Praxis bewährter Statistiken sind die Kernschätzer. Diese auf Kernfunktionen basierenden Schätzer liefern mittels einer lokalen Gewichtung in der Regel glatte Schätzfunktionen. In der mathematischen Statistik werden Kernschätzer zumeist zur Dichteschätzung und Regression eingesetzt, in den letzen Jahren jedoch auch in anderen Gebieten der Statistik wie im Bereich der *Change-Point estimation* und der Visualisierung.

Wichtig bei allen Verfahren ist die Möglichkeit, die Statistik bzw. die zur Berechnung des Schätzwertes benötigten Daten zu jedem Zeitpunkt t, an dem die Statistik ausgewertet werden soll, aktualisieren zu können. Dies schließt insbesondere die zum Zeitpunkt t gültige Stichprobe und eventuell benötigte Kennzahlen des bisher verarbeiteten Datenstroms mit ein.

[12] Liefert der Datenstrom die Datenobjekte in zufälliger und bezüglich des Schätzers unkorrelierter Reihenfolge, so enthält das Reservoir eine gültige *i.i.d.* Stichprobe und damit T_k eine gültige Statistik für den gesamten Datenstrom.

[13] vgl. Abschnitte 2.2.4 und 2.3

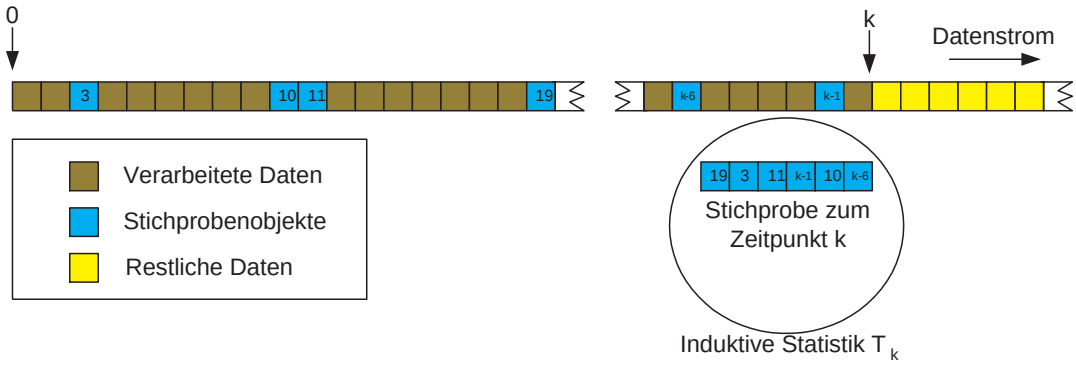

Abbildung 4.3: Speicherbasierte induktive Statistik auf einem Datenstrom zum Zeitpunkt k

4.3.1 Kernschätzer

In den letzen Jahren etablieren sich Kernschätzer und verwandte Verfahren auch in der Datenbankwelt. Hier sind Kernschätzer zunächst zur Selektivitätsschätzung ein- und mehrdimensionaler Merkmale verwendet worden[14]. Ein umfassender Vergleich von Kernschätzern zu den bis dahin dominierenden Verfahren auf Basis von Histogrammen und anderen nichtparametrischen Verfahren zur Selektivitätsschätzung findet sich in [Kor99]. Gunopulos et al. vergleichen in [GKTD00] spezielle Histogramme mit entsprechenden Kernschätzern zur Selektivitätsschätzung hochdimensionaler Bereichanfragen über numerischen Daten. Zudem finden im Bereich der Clusteranalyse[15] und der Visualisierung von Daten Kernschätzer heute Verwendung.

Einen umfassenden Überblick über die Theorie der Kernschätzer im Bereich der Regression und der Dichteschätzung liefern Wand et al. in [WJ95]. Scott zeigt in [Sco92] die Verwendung nichtparametrischer Techniken zur Visualisierung und Dichteschätzung stetiger Verteilungen.

4.3.1.1 Kerndichteschätzer

Vor allem im Bereich der Dichteschätzung haben sich Kernschätzer als effizient und leistungsstark erwiesen. Im eindimensionalen Fall sind Kerndichteschätzer in der mathematischen Literatur ausführlich diskutiert worden. Im Folgenden werden wichtige Sätze und Definitionen aus verschiedenen Quellen zusammengetragen, die zum elementaren Verständnis einiger Abschnitte notwendig sind und wichtige Ideen motivieren. Daher wird auch in der Regel auf Beweise verzichtet und es werden lediglich Quellen zum Nachlesen angegeben.

Im Falle einer stetig verteilten univariaten Zufallsvariable ist ein Kerndichteschätzer über eine Stichprobe X definiert durch

[14] vgl. vor allem [BKS99] und [Blo98].
[15] vgl. [HK98]

Definition 4.8 *Sei $X = \{X_1, \ldots, X_n\}$ eine i.i.d. Stichprobe und $n \in \mathbb{N}$. Dann heißt*

$$\hat{f}_h(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - X_i}{h}\right) \tag{4.4}$$

***Kerndichteschätzer** oder **Kernschätzer für die Dichte** mit der Kernfunktion K mit $\int K(x)dx = 1$ und der Bandbreite h.*

Unter gewissen Annahmen hat ein Dichteschätzer gemäß Formel (4.4) einige wichtige Eigenschaften. So gilt z. B. asymptotisch für den *mean integrated squared error (MISE)* folgender

Satz 4.9 [16] *Ist K eine Kernfunktion mit*

$$(a) \quad \int K(x)dx = 1$$

$$(b) \quad \int xK(x)dx = 0$$

$$(c) \quad \int x^2K(x)dx = \sigma_K^2 > 0,$$

so gilt für den Bias und die Varianz des Kerndichteschätzers

$$\begin{aligned} Bias(x) &= \frac{1}{2}\sigma_K^2 h^2 f''(x) + O(h^4) \\ Var(x) &= \frac{f(x)R(K)}{nh} - \frac{f(x)^2}{n} + O\left(\frac{h}{n}\right) \end{aligned}$$

mit $R(g) = \int g(x)^2 dx$ die roughness einer Funktion g und damit für den AMISE

$$AMISE(x) = \frac{R(K)}{nh} + \frac{1}{4}\sigma_K^4 h^4 R(f'').$$

Dieser Ausdruck wird minimal für ein

$$h^* = \left(\frac{\mathrm{R}(K)}{n\sigma_K^4 \mathrm{R}(f'')}\right)^{\frac{1}{5}}. \tag{4.5}$$

Für die Ableitungen vom Grad $r \in \mathbb{N}$ des Schätzers gilt unter den Voraussetzungen, dass

(a) $K(x)$ ist symmetrisch,

(b) $\int x^r K'(x)\mathrm{dx} = 0$ für gerade r,

(c) $\int xK'(x)\mathrm{dx} = -1$ und $\int x^3 K'(x)\mathrm{dx} = -3\sigma_K^2$,

[16] Für weitere Eigenschaften eines Kerndichteschätzers und den Beweis siehe [Sco92] Seite 130 ff.

für den AMISE der Ableitung von Grad r

$$\text{AMISE}\left(\hat{f}^{(r)}(x)\right) = \frac{\mathrm{R}(K^{(r)})}{nh^{2r+1}} + \frac{1}{4}\sigma_K^4 h^4 \mathrm{R}(f^{(r+2)}).$$

Auch dieser Ausdruck kann für eine Bandbreite h mit

$$h_r^* = \left(\frac{(2r+1)\mathrm{R}(K^{(r)})}{n\sigma_K^4 \mathrm{R}(f^{(r+2)})}\right)^{\frac{1}{2r+5}} \tag{4.6}$$

minimiert werden. Diese Eigenschaften eines Kernschätzers bzw. seiner Ableitungen lassen sich nutzen, einfache Regeln zur Berechnung einer Bandbreite h anzugeben. Jedoch hängen die asymptotisch optimalen Bandbreiten noch von der zweiten Ableitung bzw. im Falle der Ableitungsschätzer sogar von höheren Ableitungen der unbekannten wahren Dichte f ab. In Abschnitt 4.3.1.3 werden Vorschläge erläutert, wie trotzdem diese asymptotischen Eigenschaften der Schätzer zur Bandbreitenwahl genutzt werden können.

Damit die in Abschnitt 3.4 geforderten Eigenschaften eines online-fähigen Aggregats erfüllt sind, werden allgemeine Qualitätsaussagen für Kernschätzer benötigt. Eine liefert folgender

Satz 4.10 [17] *Gilt für die Kernfunktion*[18] K

$$\int_{-\infty}^{\infty} |K(x)|dx < \infty$$
$$\sup_{x\in\mathbb{R}} |K(x)| < \infty$$
$$\lim_{x\to\infty} |xK(x)| = 0,$$

so ist der Kernschätzer $\hat{f}_n(x)$ gemäß Formel (4.4) asymptotisch erwartungstreu, d. h. es gilt

$$\lim_{n\to\infty} E(\hat{f}_n(x)) = f(x). \tag{4.7}$$

Gilt zusätzlich $nh \to \infty$ für $n \to \infty$, d. h. läuft n „schneller" gegen unendlich als h gegen 0, so ist $\hat{f}_n(x)$ konsistent im quadratischen Mittel, also

$$\lim_{n\to\infty} MSE(\hat{f}_n(x)) = \lim_{n\to\infty} E\left((\hat{f}_n(x) - f(x))^2\right) = 0. \tag{4.8}$$

Die obigen Betrachtungen zeigen, dass die Qualität eines Kernschätzers im wesentlichen von der Wahl der Kernfunktion und der Wahl der Bandbreite abhängt. Hier spielt die „richtige" Wahl der Bandbreite eine erheblich größere Rolle als die Wahl der Kernfunktion[19].

4.3.1.2 Kernfunktionen

Im univariaten Fall gibt es eine Fülle von Kernfunktionen, die sich in ihren theoretischen Eigenschaften kaum unterscheiden. Ein wichtiges Kriterium einer Kernfunktion,

[17] vgl. [BT94] Seite 263 ff.
[18] Alle gängigen und alle in XXL verfügbaren Kernfunktionen erfüllen diese Eigenschaften.
[19] vgl. [Sco92] und [WJ95] Seite 58

Kernfunktion	Formel	pos. Träger	$R(K)$	σ_K^2	Klassenname
Gauss-Kern	$K(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}$	unbeschränkt	$\frac{1}{2\sqrt{\pi}}$	1	`GaussianKernel`
Epanechnikow	$K(x) = \frac{3}{4}\left(1 - x^2\right)$	$[-1, 1]$	$\frac{3}{5}$	$\frac{1}{5}$	`EpanechnikowKernel`
Biweight	$K(x) = \frac{15}{16}\left(1 - x^2\right)^2$	$[-1, 1]$	$\frac{5}{5}$	$\frac{1}{7}$	`BiweightKernel`
Triweight	$K(x) = \frac{35}{32}\left(1 - x^2\right)^3$	$[-1, 1]$	$\frac{350}{429}$	$\frac{1}{9}$	`TriweightKernel`
Kosinus	$K(x) = \frac{\pi}{4} \cos \frac{\pi}{2} x$	$[-1, 1]$	$\frac{\pi^2}{16}$	$1 - \frac{8}{\pi^2}$	`CosineArchKernel`

Tabelle 4.2: In XXL verfügbare eindimensionale Kernfunktionen

die unüberwacht, d. h. ohne ein Eingreifen eines Anwenders, in einem System Verwendung finden soll, ist eine schnelle und effiziente Berechenbarkeit. Eine auf einem Intervall beschränkte Kernfunktion hat den Vorteil, dass die Auswertung des Kernschätzers im Vergleich zur Verwendung einer unbeschränkten Funktion erheblich effizienter implementiert werden kann[20]. Vor allem in der Theorie spielt der unbeschränkte Gauss-Kern[21] eine wichtige Rolle. Ein Vorteil dieser Kernfunktion liegt in der beliebigen Differenzierbarkeit. In praktischen Anwendungen spielen jedoch höhere Ableitungen als die dritten Grades kaum eine Rolle. Der wichtigste beschränkte ist der Epanechnikow-Kern. Unter gewissen Voraussetzungen kann gezeigt werden, dass diese Kernfunktion optimal für den univariaten Kernschätzer der Dichte f bezüglich des AMISE ist. Dieser Kern gehört zu einer Familie von Kernfunktionen, die alle Polynome niederen Grades und optimal bezüglich einer Ordnung[22] sind. Zu dieser Klasse der *Beta*-Kerne gehören auch der Biweight- und der Triweight-Kern. Sogar der Gauss-Kern kann asymptotisch dieser Klasse zugeordnet werden.

Die wichtigsten und in XXL verfügbaren Kernfunktionen sind in Tabelle 4.2 zusammengestellt. Alle Eigenschaften, die für eine praktische Anwendung der Kerne wichtig sind, werden dort aufgeführt. So bezeichnet $R(K)$ die *roughness* des Kerns und berechnet sich mittels

$$R(K) = \int K^2(x)\mathrm{dx} \tag{4.9}$$

und σ_K^2 das zweite Moment der Kernfunktion mit

$$\sigma_K^2 = \int x^2 K(x)\mathrm{dx}. \tag{4.10}$$

Über den Klassennamen lassen sich die Implementierungen der Kernfunktionen, welche dem Prinzip der Funktionen in XXL folgen, im Paket `xxl.statistics.kernels` finden. Die Abbildungen 4.4 bis 4.7 zeigen die wichtigsten beschränkten Kerne. Alle in XXL verwendeten beschränkten Kerne besitzen einen positiven Träger auf $[-1, 1]$.

4.3.1.3 Bandbreitenwahl für Kernschätzer

Die in Abschnitt 4.3.1.1 genannten Eigenschaften können genutzt werden, verschiedene Strategien zur Bandbreitenberechnung zu entwickeln. Im wesentlichen wird dazu das

[20] vgl. [Blo98]
[21] Oft auch als Normaler Kern oder *normal-kernel* bezeichnet
[22] vgl. [Sco92] Seite 138 ff.

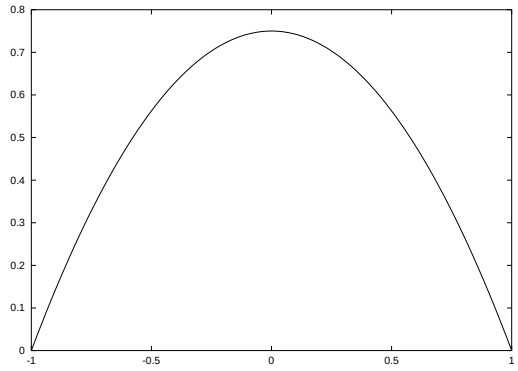

Abbildung 4.4: Epanechnikow Kern

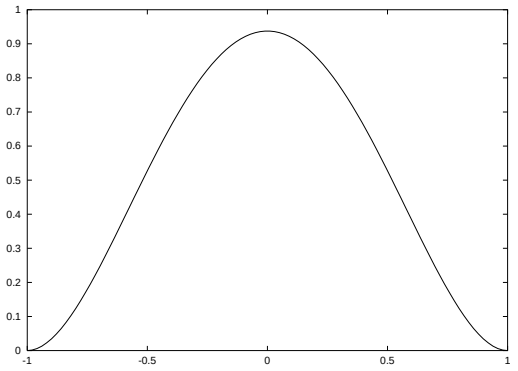

Abbildung 4.5: Biweight Kern

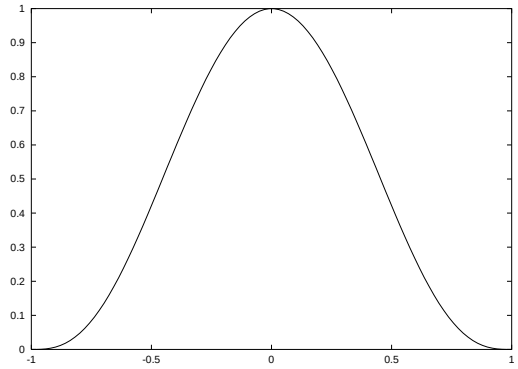

Abbildung 4.6: Triweight Kern

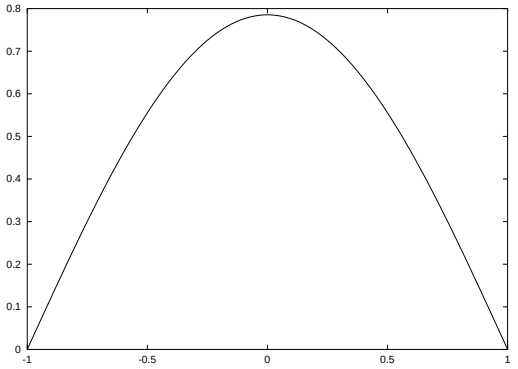

Abbildung 4.7: Kosinus Kern

asymptotisch optimale h aus Formel (4.5) genutzt. Wie bereits gesagt, hängt dieses h jedoch von der unbekannten Dichte bzw. der zweiten Ableitung der unbekannten wahren Dichte f direkt ab. Alle hier genannten Verfahren zur Bandbreitenbestimmung nutzen nun verschiedene Techniken, den unbekannten Term $R(f'')$ in Formel (4.5) zu schätzen.

Die einfachste Regel zur Bandbreitenbestimmung ist die Normalskalierungsregel (*normal scale rule*). Hier wird einfach angenommen, die wahre Dichte f sei $N(\mu, \sigma^2)$ verteilt. So lässt sich $R(f'')$ einfach berechnen und man erhält mit der Kernfunktion K eine Bandbreite gemäß

$$h_{ns} = \left(\frac{8\sqrt{\pi} R(K)}{3\sigma_K^2 n} \right)^{\frac{1}{5}} s. \tag{4.11}$$

Der Parameter s bezeichnet hierbei ein Skalierungs- bzw. Streuungsmaß wie z.B. die Standardabweichung oder den Quartilsabstand.

Für das asymptotisch optimale h lässt sich eine obere Schranke angeben. Diese kann als Startwert für induktiv arbeitende Verfahren genutzt werden. Ein direktes Verwenden dieser Schranke als Bandbreite führt zu einer „Überglättung“ des Schätzers. Die Bandbreite nach der Überglättungsregel (*oversmoothing rule* oder *maximum principle*

rule) berechnet sich mittels

$$h_{os} = \left(\frac{243R(K)}{35\sigma_K^2 n} \right)^{\frac{1}{5}} s \tag{4.12}$$

mit s wiederum ein Maß für die Streuung.

Ein ganze Klasse von Verfahren versucht mit Hilfe der Technik der Kreuzvalidierung (*cross validation*) einen Bandbreitenparameter zu bestimmen. Hier wird in der Regel nicht direkt $R(f'')$ geschätzt, sondern versucht den wahren MISE zu schätzen und zu minimieren. Die wichtigsten Verfahren sind die *biased CV*, die *least squares CV* und die *maximum likelihood CV*. Für eine ausgiebige Diskussion dieser Verfahren zur Bandbreitenbestimmung sei hier vor allem auf [WJ95] und [Sco92] verwiesen.

Der Versuch, das unbekannte $R(f'')$ selbst mit Hilfe eines Kernschätzers zu bestimmen, liefert ein zum asymptotisch optimalen h^* ähnliches Ergebnis. Als Schätzer zum Funktional $R(f^{(r)}) = \int f^{(r)}(x)^2 \mathrm{dx}$ bietet sich

$$\hat{\psi}_r(g) = \frac{1}{gn^2} \sum_{i=1}^{n} \sum_{j=1}^{n} K^{(r)} \left(\frac{X_i - X_j}{g} \right) \tag{4.13}$$

an[23]. Die Kernfunktion K muss dabei genügend oft differenzierbar sein und g bezeichnet den zu nutzenden Glättungsparameter. Der Parameter g hängt nun aber seinerseits wieder von der unbekannten Dichte f durch den Term $R(f^{(4)})$ ab. Schätzt man diesen Term wiederum mit $\hat{\psi}_r(g)$, so folgt abermals eine Abhängigkeit von der wahren Dichte; nun aber in der Form $R(f^{(6)})$. Solange die Ableitungen der wahren Dichte f existieren, kann dies fortgeführt werden. In der Praxis reicht aber in der Regel ein zweistufiges Vorgehen mit einem Startwert für g, der ähnlich der Normalskalierungsregel bestimmt werden kann. Der zu nutzende Kern muss lediglich genügend oft[24] differenzierbar sein. Hier bietet sich der Gauss-Kern an, dessen Ableitungen beliebigen Grades sich zudem einfach über das *Hermite-Polynom* berechnen lassen.

Der für die Praxis wichtigste Algorithmus, die zweistufige Einsetzungsregel[25] (*2-stage direct plug in rule* oder DPI_2), beginnt mit der Berechnung von

$$\hat{\psi}_8^{ns} = \frac{105}{32\sqrt{\pi} s^9}. \tag{4.14}$$

Damit lässt sich ein

$$g_1 = \left(\frac{-2K^{(6)}(0)}{\sigma_K \hat{\psi}_8^{ns} n} \right)^{\frac{1}{9}} \tag{4.15}$$

berechnen, welches als Bandbreite g zur Berechnung von $\hat{\psi}_6(g_1)$ mittels Formel (4.13) verwendet wird. Damit lässt sich nun ein

$$g_2 = \left(\frac{-2K^{(4)}(0)}{\sigma_K \hat{\psi}_6(g1) n} \right)^{\frac{1}{7}} \tag{4.16}$$

[23] vgl. [WJ95] Seite 67 ff.

[24] Für eine zweistufige Regel muss die Kernfunktion damit mindestens siebenmal nicht-trivial differenzierbar sein.

[25] vgl. [HSJM91] und [Tur93]

Schätzer	Bias	Varianz	h^*(MISE)	MISE($\hat{f}_{h^*}$)
Histogramm	$\sim h$	$\sim (nh)^{-1}$	$\sim n^{-\frac{1}{3}}$	$\sim n^{-\frac{2}{3}}$
Kernschätzer	$\sim h^2$	$\sim (nh)^{-1}$	$\sim n^{-\frac{1}{5}}$	$\sim n^{-\frac{4}{5}}$

Tabelle 4.3: Vergleich der Konvergenzgeschwindigkeiten zwischen Kernschätzern und Histogrammen (Quelle: [Här91] Seite 62)

berechnen, welches wiederum zur Berechnung von $\hat{\psi}_4(g_2)$ mittels Formel (4.13) genutzt wird. Die Bandbreite für den Dichteschätzer berechnet sich nun durch

$$h_{DPI2} = \left(\frac{R(\tilde{K})}{\sigma^2_{\tilde{K}} \hat{\psi}_4(g2)n} \right)^{\frac{1}{5}}. \tag{4.17}$$

Die Kernfunktion K, die bei der Berechnung der Schätzer für das Funktional $R(f^{(r)})$ verwendet wird, muss nicht gleich der Kernfunktion $\tilde{K}$ sein, die für den Dichteschätzer Verwendung findet. Da für die Funktion K Ableitungen bis zum Grad sechs berechnet werden müssen, bietet sich der Gauss-Kern an. Mit Hilfe des Hermite-Polynoms lassen sich beliebige Ableitungen ausdrücken mittels

$$\phi^{(r)}(x) = (-1)^r H_r(x)\phi(x) \tag{4.18}$$

mit dem Hermite-Polynom

$$H_r(x) = xH_{r-1}(x) - (r-1)H_{r-2}(x) \text{ für r} = 2, 3\ , \ldots \tag{4.19}$$

und

$$\begin{aligned} H_0(x) &= 1 \\ H_1(x) &= x. \end{aligned}$$

Oft wird in der Literatur vorgeschlagen, die Bandbreite aus einem vorgegebenen Bereich durch Sichtung der dadurch erzeugten Kurven zu bestimmen. Ein Experte mit Erfahrung kann auf diese Weise sicherlich wichtige Strukturen der unbekannten wahren Dichte sichtbar machen, die ein automatisch ablaufendes Verfahren „übersehen“ hätte. Im hier betrachteten Kontext ist eine solche Vorgehensweise nicht praktikabel.

4.3.1.4 Randbehandlung und Konfidenzintervall

Die in Abschnitt 4.3.1.1 genannten Merkmale der Kernschätzer belegen die guten theoretischen Eigenschaften. Mit Hilfe der dort gemachten asymptotischen Aussagen lassen sich die Konvergenzgeschwindigkeiten für die beiden Fehlerkomponenten Bias und Varianz und damit auch für den gesamten L_2-Fehler angeben. Ein interessanter Vergleich der Konvergenzgeschwindigkeiten verschiedener Größen zum Konkurrenzverfahren der Histogramme findet sich in Tabelle 4.3. Doch trotz der im Vergleich zu den Histogrammen höheren Konvergenzgeschwindigkeit und der Konsistenz der Kernschätzer[26]

[26] vgl. Formeln (4.7) und (4.8)

treten in praktischen Anwendungen oft große Fehler vor allem am Rand beschränkter Dichten auf[27]. Diese Randfehler haben ihre Ursache in einer dicht am Rand nicht mehr sichergestellten Konsistenz des Schätzers für eine feste Bandbreite h. In [Sco92], [Här91] und [WJ95] finden sich verschiedene Ansätze, diese Fehler zu minimieren. Eine einfache und effiziente Möglichkeit bietet die Technik der Spiegelung (*reflection*), die auch für mehrdimensionale Kernschätzer geeignet ist.

Definition 4.11 *Seien $X = \{X_1, \ldots, X_n\}$ eine Stichprobe und h eine Bandbreite zu einem Kernschätzer gemäß Definition 4.8 zu einer auf dem Intervall $[a, b]$ beschränkten Dichte f. Dann heißt*

$$\hat{f}_h(x) = \frac{1}{nh}\left(\sum_{i=1}^{n} K\left(\frac{x - X_i}{h}\right) + K\left(\frac{x - X_i^a}{h}\right) I(X_i)_{[a,a+h]} + K\left(\frac{x - X_i^b}{h}\right) I(X_i)_{[b-h,b]}\right) \tag{4.20}$$

gespiegelter Kernschätzer *mit $X_i^a = 2a - X_i$ und $X_i^b = 2b - X_i$ und $I(x)_{[l,r]}$ der Indikatorfunktion.*

Ein Kernschätzer gemäß Formel (4.20) ist auch am Rand konsistent. Eine höhere Konvergenzgeschwindigkeit am Rand bringt die Verwendung von speziellen Randkernen[28]. Hier wird im kritischen Bereich anstatt der Kernfunktion K eine mit $x \in [a, a + h]$ bzw. $x \in [b - h, b]$ parametrisierte Schar von Randkernen verwendet. Jedoch ist im praktischen Vergleich[29] mit der Technik der Spiegelung kein Vorteil zu erkennen, der den erheblich höheren Berechnungsaufwand rechtfertigt.

Die in Abschnitt 3.4 entwickelten Eigenschaften, die Statistiken zur online Aggregation erfüllen sollten, fordern eine Qualitätsaussage der Statistik zu jedem Berechnungsschritt. Unter gewissen Voraussetzungen kann eine solche Aussage auch für Kernschätzer gemacht werden. So gibt der folgende Satz ein asymptotisches Konfidenzintervall für einen Kerndichteschätzer an.

Satz 4.12 [30] *Existiert $f''(x)$ und gilt für die Bandbreite $h = h_n = cn^{-\frac{1}{5}}$ mit einer Konstanten $c \in \mathbb{R}$, so ist der Schätzer $\hat{f}_h(x)$ asymptotisch normalverteilt. Damit lässt sich ein asymptotisches Konfidenzintervall angeben der Form*

$$f(x) \pm \epsilon_n \approx \hat{f}_h(x) \mp \frac{1}{n^{\frac{2}{5}}}\left(\frac{c^2}{2} f''(x)\sigma_K \pm z_{1-\frac{\alpha}{2}}\sqrt{\frac{1}{c} f(x) R(K)}\right) \tag{4.21}$$

mit $z_{1-\frac{\alpha}{2}}$ das $1 - \frac{\alpha}{2}$-Quantil der Standardnormalverteilung.

Auch hier findet sich wieder die Abhängigkeit des Ausdrucks von der wahren Dichte $f(x)$ und der zweiten Ableitung der wahren Dichte $f''(x)$. Zur praktischen Anwendung müssen die beiden benötigten Funktionen geschätzt und in Formel (4.21) eingesetzt werden. Der für diesen Zweck bereits in Abschnitt 4.3.1.3 verwendete Gauss-Kern bietet sich aufgrund seiner Eigenschaften zum Schätzen der zweiten Ableitung an. Zudem

[27] vgl. hierzu vor allem [Blo98]

[28] Oft auch mit *floating kernels* bezeichnet; vgl. hierzu vor allem [WJ95] Seite 46 ff.

[29] vgl. [Blo98]

[30] Zur Herleitung und zum Beweis sei auf [Här91] Seite 62 verwiesen.

sollte die Bandbreite g dieses Schätzers größer der Bandbreite h des Kerndichteschätzers gewählt werden[31]. Mit g gemäß der *oversmoothing rule* (4.12) und

$$\hat{f}_g''(x) = \frac{1}{ng^3} \sum_{i=1}^{n} \phi'' \left(\frac{x - X_i}{g} \right)$$

lässt sich nun ein Schätzer für das Konfidenzintervall zum Konfidenzlevel $1 - \alpha$ im Punkt x mit $\hat{f}_h(x) = y$ angeben mit

$$\hat{\epsilon}_n^{(-)} = \frac{1}{n^{\frac{2}{5}}} \left(\frac{c^2}{2} \hat{f}_g''(x) \sigma_K + z_{1-\frac{\alpha}{2}} \sqrt{\frac{1}{c} y R(K)} \right) \tag{4.22}$$

für die untere Intervallgrenze $y - \hat{\epsilon}_n^{(-)}$ und

$$\hat{\epsilon}_n^{(+)} = -\frac{1}{n^{\frac{2}{5}}} \left(\frac{c^2}{2} \hat{f}_g''(x) \sigma_K - z_{1-\frac{\alpha}{2}} \sqrt{\frac{1}{c} y R(K)} \right) \tag{4.23}$$

für die obere Intervallgrenze $y + \hat{\epsilon}_n^{(+)}$.

4.3.2 Hybrider Kerndichteschätzer

Erfüllt eine Verteilung die Modellannahmen eines Kerndichteschätzers nicht, so können die obigen Aussagen über die Konvergenz und die Qualität eines solchen Schätzers nicht aufrecht erhalten werden. So gilt z. B. für Verteilungen, deren Dichte f nicht für alle $x \in \mathbb{R}$ zweimal stetig differenzierbar ist, dass an diesen Punkten der Fehler des Schätzers unter Umständen beliebig groß werden kann. So ist der unüberwachte Einsatz von Kernschätzern für nicht glatte Verteilungen problematisch.

Auf der anderen Seite bieten Histogramme, bei Kenntnis der Lage der entsprechenden Punkte, eine natürliche Unterstützung für nicht glatte Verteilungen. Histogramme zur Dichteschätzung sind aufgrund ihres Konstruktionsprinzips eine Treppenfunktion mit Sprüngen an den Rändern der Histogrammklassen (auch mit *bins* oder *buckets* bezeichnet) und arbeiten innerhalb einer Klasse unabhängig von den anderen Klassen. Ist nun die Zahl solcher „interessanter“ Punkte beschränkt und deren Lage bekannt, können die *bins* eines Histogramms so aufgebaut werden, dass die Grenzen der Klassen genau auf diese Punkte fallen.

Bei diesem Ansatz gibt man aber die höhere Konvergenzgeschwindigkeit der Kernschätzer gegenüber den Histogrammen auf. Zudem können dabei einzelne Klassen so groß werden, dass eine weitere Unterteilung in zusätzliche *bins* notwendig wird. Damit entstehen wiederum, nur an anderer Stelle, die Probleme, die beim (unüberwachten) Aufbau der Klassen in der Regel auftreten. Jedoch kann bei Kenntnis der Lage dieser interessanten Punkte auch ein Kernschätzer an diese Situation angepasst werden, indem man das den Histogrammen zugrunde liegende Prinzip mit der Theorie der Kernschätzer kombiniert.

[31] vgl. [Här91] Seite 221

Sei im Folgenden F die unbekannte stetige Verteilung mit der unbekannten Dichte f, für die

$$f(x) = \begin{cases} f_0(x) & \text{für } x \leq x_{\Delta_1} \\ f_1(x) & \text{für } x_{\Delta_1} < x \leq x_{\Delta_2} \\ \vdots & \vdots \\ f_v(x) & \text{für } x_{\Delta_v} \leq x \end{cases} \tag{4.24}$$

mit

$$\lim_{x \to x_{\Delta_i}} f_i(x) \neq \lim_{x_{\Delta_i} \leftarrow x} f_{i+1}(x) \text{ für } i = 0, \ldots, v-1 \tag{4.25}$$

und

$$f_i''(x) \text{ existiert für alle } x \in (x_{\Delta_{i-1}}, x_{\Delta_i})$$

gilt. Dabei bezeichnen die Pfeile in Formel (4.25) den links- bzw. rechtsseitigen Grenzwert der Funktion. Die Dichte f besteht somit aus $v+1$ zweimal differenzierbaren „Teildichten“, in deren Endpunkten Sprungstellen liegen. Des Weiteren sei

$$\Delta_i := |f_i(x_{\Delta_i}) - f_{i+1}(x_{\Delta_i})|$$

die Höhe der Sprünge in der Dichte f. Nun gelten die Voraussetzungen für Kerndichteschätzer für die Teildichten und können ohne Qualitätsverlust darauf angewendet werden.

Definition 4.13 *Seien $x_{\Delta_1} < \ldots < x_{\Delta_v}$ bekannt und $X = \{X_1, \ldots, X_n\}$ eine i.i.d. Stichprobe der Größe n. Dann heißt*

$$\hat{f}(x) = \begin{cases} \hat{f}_0(x) & \textit{für } x \leq x_{\Delta_1} \\ \hat{f}_1(x) & \textit{für } x_{\Delta_1} < x \leq x_{\Delta_2} \\ \vdots & \vdots \\ \hat{f}_v(x) & \textit{für } x_{\Delta_v} \leq x \end{cases}$$

Hybrider Kerndichteschätzer *mit*

$$\hat{f}_i(x) = \frac{1}{h_i n} \sum_{l=1}^{n_i} K_i \left(\frac{x - X_l^{(i)}}{h_i} \right)$$

für eine Kernfunktion K_i, eine Bandbreite h_i und Stichprobenwerte $X_l^{(i)} \in X^{(i)} = \{X_l : X_l \in (x_{\Delta_{i-1}}, x_{\Delta_i}]\}$ und $|X^{(i)}| = n_i$.

Setzt man nun Unabhängigkeit zwischen den Klassen voraus, entstehen an allen Sprungstellen x_{Δ_i} die gleichen Probleme, wie am Rand einer Verteilung. Jedoch kann dies wie auch im klassischen Fall durch die Verwendung des Prinzips der Spiegelung gelöst werden.

Da aber die Sprungstellen in der Regel nicht bekannt sind, werden Schätzer für die Lage der Sprünge benötigt. Im Abschnitt 4.3.5 wird auf dieses Problem eingegangen[32].

[32] Hybride Kernschätzer sind in [BKS99] zum ersten Mal zur Selektivitätsschätzung eingesetzt worden. Jedoch ist die dort empfohlene Vorgehensweise zur Bestimmung der Klassen nicht immer anwendbar und nicht sehr performant.

4.3.3 Kernselektivitätsschätzer

In [Blo98] und [Kor99] sind Kernschätzer zum ersten Mal zur Selektivitätsschätzung in Datenbanksystemen verwendet worden. Die Selektivität einer Datenbankanfrage ist definiert als Verhältnis der Anzahl der sich bezüglich eines Prädikats qualifizierender zu der Gesamtzahl aller möglichen Tupel. Um die Selektivität bezüglich einer Fensteranfrage[33] zu bestimmen, kann man den Zusammenhang zwischen der Wahrscheinlichkeit, dass ein Datum in einen vorgegebenen Bereich fällt, mit der tatsächlichen Anzahl der Daten in diesem Bereich ausnutzen.

Seien nun G eine Grundgesamtheit, deren numerische Daten stetig gemäß F verteilt sind, und $[a, b]$ ein Intervall mit $a < b$. Die Selektivität sel einer Anfrage $Q([a, b])$ ist definiert durch

$$\text{sel}(Q([a,b])) = \frac{|Q([a,b])|}{|G|}.$$

Unter der Annahme, dass die in der Datenbank realisierte Menge G nur eine Stichprobe der tatsächlichen gemäß F verteilten Datenmenge ist, kann die Selektivität auch mittels

$$\text{sel}(Q([a,b])) = F(b) - F(a) = \int_a^b f(x)\text{dx}$$

definiert werden. Mit einem Schätzer $\hat{f}$ für die unbekannte Dichte f kann so unter gewissen Voraussetzungen[34] ein Schätzer sel^e für die Selektivität einer Anfrage $Q([a, b])$ etabliert werden durch

$$\text{sel}^e(a,b) = \int_a^b \hat{f}(x)\text{dx}. \tag{4.26}$$

Damit kann nun ein speicherbasierter Selektivitätsschätzer für eine Bereichsanfrage als laufendes Aggregat konstruiert werden. Es ist also möglich, die Selektivität einer dem aktuellen Operator im Operatorbaum[35] folgenden Selektion bezüglich eines Intervalls online zu schätzen. Wie bereits bei den nichtparametrischen Schätzern auf Basis von Kernfunktionen für die Dichte einer stetigen Verteilung[36], können auch hier Randeffekte den Schätzer stark beeinflussen.

Einen ersten einfachen Schätzer für die Selektivität einer Bereichsanfrage erhält man durch die Anwendung des Modells gemäß Formel (4.26) auf den einfachen Kerndichteschätzer.

Definition 4.14 *Seien $X = \{X_1, \ldots, X_n\}$ eine Stichprobe einer stetig verteilten Grundgesamtheit, $h > 0$ eine Bandbreite und K eine auf $[-1, 1]$ beschränkte Kernfunktion. Dann liefert*

$$sel_N^e(a,b) = \frac{1}{n}\sum_{i=1}^{n} \int_{\max\left(-1, \frac{a-X_i}{h}\right)}^{\min\left(1, \frac{b-X_i}{h}\right)} K\left(\frac{x - X_i}{h}\right) dx \tag{4.27}$$

[33] Oft auch mit Bereichsanfrage oder *range query* bezeichnet.
[34] vgl. [Blo98] Seite 50 ff.
[35] vgl. Abschnitt 2.1
[36] vgl. Abschnitt 4.3.1

einen ***Selektivitätsschätzer*** *zu einer Bereichsanfrage $Q([a,b])$. Der Schätzer $sel^e(Q)_N$ heißt* ***einfacher Kernselektivitätsschätzer*** *oder auch nativer Kernselektivitätsschätzer (native kernel selectivity estimator).*

Die Definition kann ohne weiteres auch auf nicht beschränkte Kernfunktionen erweitert werden. Jedoch muss im Falle eines unbeschränkten Kerns, wie zum Beispiel des Gauss-Kerns, die gesamte Stichprobe der Größe n durchlaufen werden, um den Schätzer für eine Anfrage bezüglich eines Intervalls $[a,b]$ auszuwerten, da alle Stichprobenwerte einen Beitrag zur Masse in einem Punkt x liefern. Damit ist der Aufwand zur Auswertung einer Anfrage $\Theta(n)$.

Bei der Verwendung einer auf $[-1,1]$ beschränkten Kernfunktion kann durch eine einmalige Sortierung der Stichprobe der Aufwand aus folgenden Gründen auf $\Theta(\log n)$ reduziert werden. Durch den lokal beschränkten Einfluss eines Stichprobenwertes X_i auf $[X_i - h, X_i + h]$ müssen lediglich die zwischen den beiden Werten $a-h$ und $b+h$ liegenden Stichprobenwerte in die Berechnung mit einbezogen werden. Die entsprechenden Stichprobenwerte X_{i_a} und X_{i_b} mit $X_{i_a} \leq a-h < b+h \leq X_{i_b}$ können effizient mittels zweifacher binärer Suche[37] in der sortierten Stichprobe bestimmt werden. Zusätzlich gehen alle Stichprobenwerte X_i mit $a+h \leq X_i \leq b-h$ mit der Masse 1 in die Berechnung ein[38], was eine zusätzliche Optimierung möglich macht.

Den unter Umständen auftretenden Randeffekten kann im Falle einer Selektivitätsschätzung mit den gleichen Mitteln begegnet werden, wie bei einer Dichteschätzung mit Kernschätzern.

Definition 4.15 *Seien $X = \{X_1, \ldots, X_n\}$ eine Stichprobe einer auf $[l,r]$ beschränkten stetig verteilten Grundgesamtheit, $h > 0$ eine Bandbreite und K eine auf $[-1,1]$ beschränkte Kernfunktion. Dann heißt*

$$
\begin{aligned}
sel_R^e(a,b) \;=\; & \frac{1}{n}\sum_{i=1}^{n}\left(\int_{\max\left(-1,\frac{a-X_i}{h}\right)}^{\min\left(1,\frac{b-X_i}{h}\right)} K\left(\frac{x-X_i}{h}\right)dx\right. \\
& + \int_{\max\left(-1,\frac{a-X_i^l}{h}\right)}^{\min\left(1,\frac{b-X_i^l}{h}\right)} K\left(\frac{x-X_i^l}{h}\right)dx \cdot I(X_i)_{[a,a+h]} \\
& \left. + \int_{\max\left(-1,\frac{a-X_i^r}{h}\right)}^{\min\left(1,\frac{b-X_i^r}{h}\right)} K\left(\frac{x-X_i^r}{h}\right)dx \cdot I(X_i)_{[b-h,b]}\right),
\end{aligned}
$$

mit $X_i^l = 2l - X_i$ und $X_i^r = 2r - X_i$ und $I(x)_{[l,r]}$ der Indikatorfunktion, ***gespiegelter Kernselektivitätsschätzer*** *(reflection kernel selectivity estimator) für eine Bereichsanfrage $Q([a,b])$ mit $l \leq a < b \leq r$.*

[37] Die binäre Suche in einem sortierten Array hat den Aufwand $\Theta(\log n)$.

[38] Da Kernfunktionen gleichzeitig die Eigenschaften einer Dichte besitzen, integrieren sie zu 1 über ihrem Definitionsbereich.

4.3.4 Hybrider Selektivitätsschätzer

Da die im obigen Abschnitt 4.3.3 vorgestellten Selektivitätsschätzer auf Basis integrierter Kernfunktionen die gleichen Voraussetzungen benötigen, wie die entsprechenden Dichteschätzer, gelten auch die gleichen Einschränkungen bei Verteilungen mit Sprüngen[39]. Jedoch können auch in diesem Fall die oben vorgestellten Selektivitätsschätzer mittels einer hybriden Methode angepasst werden.

Definition 4.16 *Seien F eine Verteilung mit einer Dichte f gemäß Formel (4.24), die $x_{\Delta_1} < \ldots < x_{\Delta_v}$ bekannt und $X = \{X_1, \ldots, X_n\}$ eine i.i.d. Stichprobe von F der Größe n. Dann heißt*

$$sel^e_H(a,b) \quad = \quad \sum_{i=0}^{v} \frac{n_i}{n} sel^e_i(a_i, b_i)$$

Hybrider Kernselektivitätsschätzer *für eine gegebene Bereichanfrage $Q([a,b])$ mit $sel^e_i(a_i, b_i)$ ein Selektivitätsschätzer über einer Stichprobe $X^{(i)}$ für die i-te Klasse. Dabei ist zu beachten, dass die Anfrage $Q([a,b])$ für jede Klasse i, die von $[a,b]$ geschnitten wird, mittels*

$$\begin{aligned} a_i &:= \max(a, x_{\Delta_i}) \\ b_i &:= \min(b, x_{\Delta_{i+1}}) \end{aligned}$$

angepasst werden muss. Die Selektivität für Klassen, deren Schnitt mit der Bereichsanfrage leer ist, braucht nicht berechnet zu werden, da sie 0 ist. Zudem liefern Klassen, die vollständig von der Anfrage überdeckt werden, die Masse 1. Dabei gilt $X^{(i)} = \{X_l : X_l \in (x_{\Delta_{i-1}}, x_{\Delta_i}]\}$ und $|X^{(i)}| = n_i$.

Auch hier gelten die für hybride Dichteschätzer gemachten Aussagen bezüglich der Grenzen der einzelnen Klassen. Die dort auftretenden Randeffekte können durch die Wahl eines gespiegelten Kernselektivitätsschätzers nach Definition 4.15 für jede Klasse weitestgehend vermieden werden.

Im folgenden Abschnitt wird ein Verfahren zur Schätzung der Lage und der Höhe von Sprungstellen in stetigen Verteilungen mit einer Dichte gemäß Formel (4.24) entwickelt.

4.3.5 Schätzer zur Sprungstellenentdeckung

Nicht nur zur effizienten Verwendung hybrider Verfahren in der nichtparametrischen Statistk[40] ist es wichtig, besondere Punkte in einer Datenverteilung zu kennen. In vielen Fällen markieren solche Punkte eine Veränderung der Daten selbst, der Datenverteilung oder anderer Eigenschaften. Eine wichtige Klasse solcher besonderen Punkte ist die der *change points*. Diese markieren einen Wechsel der den Daten zugrunde liegenden

[39] vgl. Abschnitt 4.3.2
[40] vgl. Abschnitte 4.3.2 und 4.3.4

Verteilung. Besonders im Bereich der stochastischen Prozesstheorie kommen parametrische Modelle zur Bestimmung solcher *change points* in einem oft stationären Prozess zum Einsatz. Neben einem Wechsel der Verteilungsfunktion sind gerade für Verfahren, die das Prinzip der lokalen Glättung verwenden, Sprünge in der Verteilungs- bzw. Dichtefunktion der Daten besonders wichtig.

Im folgenden habe die unbekannte wahre Dichte f einer stetigen Verteilung F eine Form gemäß Formel (4.24). Ziel ist es nun, die Punkte x_{Δ_i} mit $i = 1, \ldots, v$ und die Anzahl solcher Sprungstellen v zu bestimmen.

In der Literatur[41] finden sich verschiedene Schätzer, die alle auf der gleichen Idee basieren. Befindet sich eine Sprungstelle in einer Kurve, so wird die Höhe dieser Sprungstelle durch Glättungsverfahren lokal „verteilt". Trotzdem wird die erzeugte glatte Kurve eine relativ hohe Steigung in diesem Bereich besitzen. Damit liegt es nahe, die Differenz zweier einseitiger Kernschätzer zur Erkennung solcher Sprünge zu verwenden. Der erste Kernschätzer nutzt einen linksseitigen Kern, während der zweite einen rechtsseitigen Kern verwendet[42]. Aufgrund des den Kernschätzern zugrunde liegenden Prinzips der lokalen gewichteten Durchschnittsbildung wird die Differenz nahe Null sein für den Fall, dass die zu schätzende Kurve in diesem Punkt glatt ist. Andernfalls wird der Schätzer einen Wert nahe der tatsächlichen Sprunghöhe annehmen. Damit stellt eine solche Differenz ein gutes Kriterium für die Existenz einer Sprungstelle dar.

Eine Kernfunktion kann zur Berechnung eines solchen *Difference Kernel Estimators* verwendet werden, wenn sie einen beschränkten Support besitzt. Hier werden wir uns auf solche Kernfunktionen beschränken, die auf $[-1,\ 1]$ einen positiven Träger besitzen.

Definition 4.17 *Sei $K(t)$ mit $t \in \mathbb{R}$ eine auf $[-1,\ 1]$ beschränkte Kernfunktion mit $K(t) \geq 0$ für alle t, so heißt*

$$K^T(t) = \begin{cases} \frac{2}{c} K\left(\frac{2}{c}(x-l)-1\right) & , \text{ mit } l \leq t \leq l+c \\ 0 & , \text{ sonst} \end{cases} \tag{4.28}$$

transformierter Kern *oder* **transformierte Kernfunktion** *zum Intervall $[l,\ l+c]$.*

Diese Transformation erhält offensichtlich viele Eigenschaften der transformierten Kernfunktion.

Satz 4.18 *Sei $K^T_{[l,\ l+c]}(t)$ ein mittels $K(t)$ auf $[l,\ l+c]$ transformierter Kern, so gilt:*

1. $\int_l^{l+c} K^T_{[l,\ l+c]}(t)\ dt\ = \int_{-1}^{1} K(t)\ dt\ = 1$
2. *Ist $K(t)$ symmetrisch, so auch $K^T_{[l,\ l+c]}(t)$*
3. *Ist $K(t)$ differenzierbar, so gilt*

$$\frac{d\ K^T(t)}{dt} = \left(\frac{2}{c}\right)^2 K'\left(\frac{2}{c}(x-l)-1\right) \tag{4.29}$$

[41] vgl. vor allem [Qiu94] und [Mül92]
[42] vgl. [Qiu94]

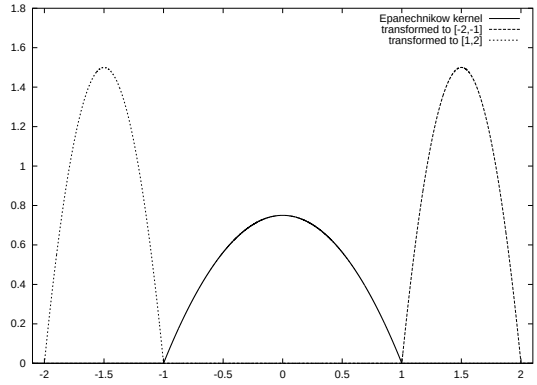

Abbildung 4.8: Epanechnikow Kern mit transformierten Kernen

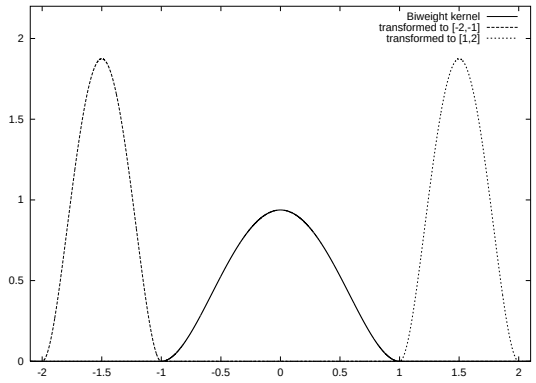

Abbildung 4.9: Biweight Kern mit transformierten Kernen

4. Es gilt

$$\int_l^a K^T_{[l,\ l+c]}(t)d\ t\ = \int_{-1}^{\frac{2}{c}(a-l)-1} K(t)\ dt \quad mit\ l \leq a \leq l+c \tag{4.30}$$

Beweis:
Die Behauptungen ergeben sich direkt aus Definition 4.17.
1. und 2. folgen direkt aus Definition 4.28, 3. folgt direkt aus Definition 4.28 mit Hilfe der Kettenregel und 4. folgt direkt aus Definition 4.28 mit Hilfe der Substitutionsregel.
q.e.d.

Die Abbildungen 4.8 und 4.9 zeigen zwei der wichtigsten Kernfunktionen und ihre links- und rechtsseitigen Transformationen zu den Intervallen $[-2,-1]$ und $[1,2]$. Mit Hilfe dieser transformierten Kernfunktionen kann nun ein Schätzer für Sprungstellen in unbekannten Verteilungen bzw. in deren Dichtefunktionen angegeben werden.

Definition 4.19 *Sei $K(t)$ eine Kernfunktion mit auf $[-1,\ 1]$ beschränktem Support und $X = X_1, X_2, \cdots, X_n$ eine i.i.d. Stichprobe der Größe n. Dann heißt*

$$\hat{dke}_h(x;X) = \frac{1}{nh}\sum_{i=1}^{n}\left(K^{(r)}\left(\frac{X_i - x}{h}\right) - K^{(l)}\left(\frac{X_i - x}{h}\right)\right) \tag{4.31}$$

Difference Kernel Estimator[43] *mit $K^{(l)}(t) := K^T_{[-2,1]}(t)$ und $K^{(r)}(t) := K^T_{[1,1]}(t)$ zur Bandbreite h.*

Die Konsistenz eines Schätzers gemäß Definition 4.19 mit $l = -2$, $r = 1$ und $c = 1$ ist durch Müller [Mül92] und Qiu et al. [QAL91] in verschiedenen Szenarien gezeigt worden. Damit werden im Folgenden nur noch auf die Intervalle $[-2,-1]$ und $[1,2]$ transformierte Kerne betrachtet. Abbildung 4.10 zeigt einen *Difference Kernel Estimator* (DKE) für eine standardnormalverteilte Grundgesamtheit und die tatsächliche Dichte. Der DKE basiert auf einer Stichprobe der Größe $n = 2000$, dem Biweight-Kern und einer Bandbreite h, die mittels der *oversmoothing rule* berechnet worden ist. Die beiden lokalen Extrema des DKE liegen an denjenigen Stellen der Glockenkurve mit der

[43] vgl. [Qiu94]

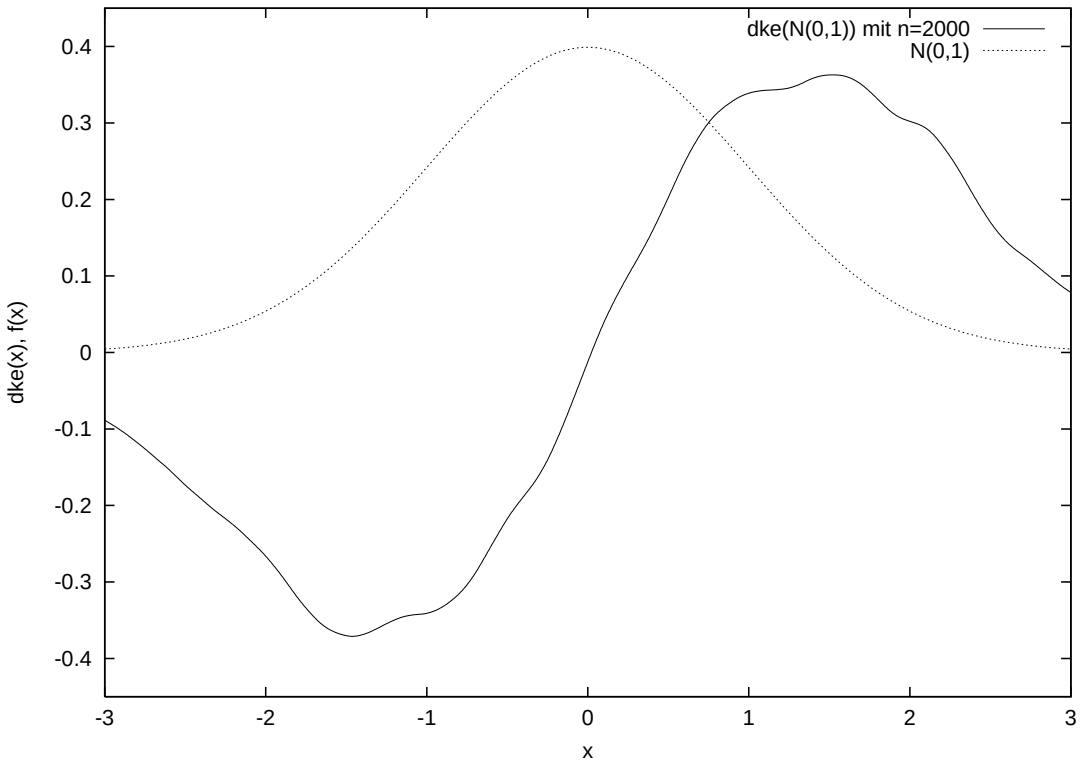

Abbildung 4.10: *Difference Kernel Estimator* der Dichte einer Stichprobe der Größe $n = 2000$ einer gemäß $N(0, 1)$ verteilten Grundgesamtheit mit wahrer Dichte f

jeweils absolut größten Steigung bzw. den größten Ableitungswerten. Dies zeigt aber auch die Schwierigkeiten, die in der praktischen Verwendung dieses Prinzips auftreten. Im obigen Beispiel fände ein unüberwachtes[44] Verfahren entweder zwei Kandidaten für Sprungstellen oder bei Angabe eines zu niedrigen Schwellenwertes für die wahrscheinliche Höhe der Sprünge keinen Kandidaten. Somit ist in der Regel die Kenntnis der Anzahl der Sprünge oder die eines Schwellenwertes notwendig. Die Verwendung eines Schwellenwertes hat den entscheidenden Nachteil einer Abhängigkeit von den verwendeten Parametern, d. h. die Größe der Stichprobe n und die Wahl der Bandbreite h beeinflussen direkt die Höhe des DKE in einem Punkt x und damit auch die Wahl eines Schwellenwertes.

Qiu stellt in [Qiu94] eine konsistente Lösung zum obigen Problem vor. Dieses mit *Difference Apart Kernel Estimator* (DAKE) bezeichnete Verfahren arbeitet in zwei Schritten. Zuerst wird mit Hilfe des DKE und einer Folge von Schwellenwerten b_j eine Kandidatenmenge $\{a_i^j\}$ bestimmt, aus der dann sukzessive einzelne Werte eliminiert werden, bis für einzelne isolierte Bereiche j nur ein Kandidat übrig bleibt.

[44] Alle Verfahren im Kontext eines Datenbanksystems sollten unüberwacht bzw. zumindest ohne zusätzlich benötigtes Expertenwissen ablaufen.

Seien im Folgenden f eine auf $[l, r]$ beschränkte Dichte und

$$a_i = l + \frac{i(r-l)}{m} \text{ für } i = 1, \ldots, m.$$

Mit

$$m = \lfloor \frac{r-l}{2h_n} \rfloor + 1$$

gilt für $n \to \infty$ und damit $h_n \to 0$ auch $m = m_n \to \infty$. Damit ist die Folge $\{a_i : i = 1, \ldots, m\}$ dicht in $[l, r]$.

Qiu beschränkt sich bei der Auswertung des DKE auf die Punkte a_i und reduziert damit die Komplexität erheblich. Seien nun

$$a_{i_1} < a_{i_2} < \ldots < a_{i_c}$$

diejenigen a_i mit

$$\left| dke_n(a_{i_j}) \right| \geq \frac{b_0}{2} \text{ mit } j = 1, 2, \ldots, c,$$

d. h. diejenigen Punkte in $[l, r]$, für die der DKE ausgewertet wird und deren Wert oberhalb einer Schranke $\frac{b_0}{2}$ liegt. Nun gilt aber, dass mit einer hohen Wahrscheinlichkeit benachbarte Punkte a_i und a_{i+1} auf diese Weise in die Kandidatenmenge aufgenommen werden. Damit wird es notwendig, einzelne Werte, die dicht beieinander liegen, zu eliminieren. Gilt nun für zwei Werte j_l und j_r, dass

$$\begin{aligned} a_{i_j} - a_{i_{j+1}} &= \frac{1}{m} \text{ für } j = j_l, j_l + 1, \ldots, j_r \\ a_{i_{j_l}} - a_{i_{j_l - 1}} &> \frac{1}{m} \\ a_{i_{j_r+1}} - a_{i_{j_r}} &> \frac{1}{m} \end{aligned}$$

gilt, werden alle Kandidaten $\{a_{i_j}\}_{j=j_l}^{j_r}$ eliminiert und durch ihren Mittelwert $\frac{a_{i_{j_r}} - a_{i_{j_l}}}{2}$ ersetzt. Die verbleibenden Kandidaten repräsentieren nun die Schätzwerte für die unbekannten Sprungstellen x_{Δ_i}.

In der praktischen Anwendung stellen sich jedoch einige Fragen. So ist die Wahl der Bandbreite h_n essentiell. Zudem muss der Schwellenwert b_0 richtig bestimmt bzw. ausgewählt werden. Qiu schlägt in [Qiu94] eine Bandbreite $h = \frac{r-l}{5} n^{-\frac{1}{3}}$ und einen Schwellenwert $b_0 = 2(r-l)n^{-\frac{1}{4}}$ vor und verweisst darauf, dass die Zahl der Auswertungspunkte m nicht zu klein gewählt werden darf[45].

4.3.6 Kernschätzer in XXL

Die generische Bibliothek XXL bietet sowohl Schnittstellen zur einfachen Implementierung eigener Kernschätzer als auch vordefinierte Klassen für kernbasierte Dichte- und Verteilungsschätzer.

[45] Direkt geht Qiu in [Qiu94] nur auf die Wahl der entsprechenden Parameter für den Fall einer auf $[0, 1]$ normierten Funktion ein.

Alle bereits in Abschnitt 4.3.1.2 vorgestellten Kernfunktionen implementieren `xxl.statistics.kernels.KernelFunction` und sind in einer eigenen Klasse realisiert. Dies ist notwendig, obwohl Kernfunktionen keinen Status besitzen und damit statisch (`static`) implementiert sein sollten[46], um kernbasierten Verfahren die zu verwendende Kernfunktion als Parameter übergeben zu können. Somit ist zusätzlich sichergestellt, dass die zur einfachen Bandbreitenbestimmung benötigten Größen $R(K)$ und σ_K^2 von allen Kernen zur Verfügung gestellt werden. An mehrdimensionalen Kernfunktionen sind zur Zeit[47] in XXL lediglich Produktkerne implementiert. Eigene mehrdimensionale Kernfunktionen und sphärische Kerne können jedoch leicht mit Hilfe der Schnittstelle `xxl.statistics.kernels.KernelFunctionND` realisiert werden.

Die Klasse `xxl.statistics.kernels.KernelBandwidths` stellt zur Zeit sechs verschiedene Möglichkeiten zur Bandbreitenbestimmung zur Verfügung. Für mehrdimensionale Kernschätzer stehen *Scott's rule*[48] und *Ada-Band*[49] zur Verfügung. Die *Ada-Band*-Regel basiert auf einer k-Nächster-Nachbar-Suche auf der gesamten Datenmenge und ist somit blockierend in der Anwendung. *Scott's rule* benötigt lediglich die Varianzen der Merkmale in den verschiedenen Dimensionen, die sehr leicht auf einem Datenstrom berechnet werden können[50].

Im eindimensionalen Fall sind die einfachen Bandbreiten der *normal scale rule* und der *oversmoothing rule* implementiert. Zusätzlich stehen die zwei-stufige *direct-plug-in rule* und eine Variante der *cross validation* Regeln, die *maximum likelihood CV*[51], zur Verfügung. Der Aufwand für die einfachen Regeln zur Bandbreitenbestimmung ist lediglich $O(1)$, während der Aufwand für die *plug-in*-Verfahren und die der *cross validation* bei $O(n^2)$ liegt.

Zur Dichteschätzung stehen mit den Klassen `NativeKernelDensityEstimator` und `ReflectionKernelDensityEstimator` bereits zwei Kernschätzer in XXL zur Verfügung. Während der erste einen einfachen Dichteschätzer gemäß Formel (4.4) implementiert, stellt der zweite einen Schätzer für eine auf das Intervall $[a, b]$ beschränkten Dichte gemäß Formel (4.20) bereit. Beide Klassen ermöglichen zudem die Berechnung eines Konfidenzintervalls nach den Formeln (4.22) und (4.23) zu einem gegebenen Konfidenzlevel $1 - \alpha$. Bei beiden Implementierungen ist es möglich, die Parameter des Schätzers zur Laufzeit zu ändern und sie damit zur Online Aggregation zu nutzen.

Des Weiteren bieten die Klassen `NativeKernelCDF` und `ReflectionKernelCDF` Unterstützung zur Selektivitätsschätzung von Fensteranfragen mit Hilfe von Kernschätzern.

[46] In der Klasse `xxl.util.Statistics` finden sich die statischen Implementierungen aller in XXL verfügbaren Kernfunktionen. Zur Zeit kapseln alle Klassen, die `KernelFunction` implementieren lediglich den statischen Methodenaufruf und stellen die zusätzlichen Konstanten zur Verfügung. Damit wird einerseits eine Code-Duplizierung vermieden und andererseits werden alle Kernfunktionen zur direkten Verwendung ohne Objekt-Instanziierung bereitgestellt.

[47] vgl. aktuelle Versionen der Bibliothek sowie Abschnitt 2.5.

[48] vgl. [Sco92] Seite 131 ff.

[49] vgl. [DG01]

[50] vgl. Abschnitt 3.5.3

[51] Die *maximum likelihood CV* steht auch für mehrdimensionale Kernschätzer für einen Spezialfall zur Verfügung.

Die hybriden Verfahren stehen durch die Klassen `HybridKernelDensityEstimator` bzw. `HybridKernelCDF` zur Verfügung. Alle Kerndichte- und Kernselektivitätsschätzer stellen *Factory*-Methoden zur Verfügung, die mittels durch ein `Object[]` gegebene Parameter Instanzen des jeweiligen Typs liefern. Darüber hinaus besteht die Möglichkeit eine Bandbreite zur Nutzung vorzugeben oder diese nach jeder Änderung neu berechnen zu lassen.

Zur Verwendung des *Difference Kernel Estimators* werden transformierte Kernfunktionen benötigt. Die Klasse `TransformedKernelFunction` stellt diese transparent für alle beschränkten Kernfunktionen zur Verfügung. Zusätzlich implementieren Objekte dieses Typs sowohl die Schnittstelle `Differentiable` als auch `Integrable`, falls dies bei der *gewrappten* Kernfunktion bereits der Fall ist.

4.4 Probleme und Grenzen des Ansatzes

Das vorliegende Kapitel hat einen allgemeinen Ansatz zur Nutzung induktiver Statistiken über Datenströmen zu einem festen Speicherbedarf aufgezeigt. Damit können beliebige auf Stichproben basierende Verfahren der mathematischen Statistik mit beschränkter Komplexität zur Online Aggregation eingesetzt werden. Jedoch hat dieser Ansatz neben dem großen Vorteil der Generalität auch einige Nachteile.

So sind zum Zeitpunkt t alle Elemente des Datenstroms x_i mit $i = 1, \ldots, t$ bereits verarbeitet worden. Zum Aufbau und zur Auswertung der Statistik werden jedoch nur n Elemente mit in der Regel $n << t$ verwendet[52]. So wird es versäumt, die Informationen, die bereits zur Verfügung standen, vollständig zu nutzen.

Ein weiterer Nachteil der speicherbasierten Verfahren ist eine mangelnde Flexibilität auf veränderte Speicherplatzbedingungen in einem dynamischen Umfeld wie die eines Datenbanksystems. Eine Verkleinerung des genutzten Reservoirs zur Speicherung der aktuellen Stichprobe ist bei Kenntnis des aktuellen Zeitpunkts t zu realisieren. Dabei werden „zufällig“ so viele Elemente[53] ausgewählt, wie nach einer Verkleinerung des Speichers weniger im Speicher gehalten werden können. Auf zusätzlich zur Verfügung stehenden Speicherplatz kann jedoch nicht reagiert werden, da auf bereits verarbeitete Elemente nicht zugegriffen werden kann. Dies wäre jedoch notwendig, um die „leeren“ Plätze im Reservoir zu füllen.

Durch das zu Grunde liegende Prinzip des speicherbasierten Stichprobenziehens enthält das Reservoir immer eine *i.i.d.* Stichprobe des gesamten bereits verarbeiteten Datenstroms. Sind jedoch Teile der Daten verrauscht oder beinhalten Messfehler, so gelangen diese Werte mit gleicher Wahrscheinlichkeit in die Stichprobe wie qualitativ „bessere“ Werte. Ein Ausnutzen von Wissen über einzelne Daten oder ganze Bereiche von Daten ist so nicht möglich.

[52] D. h. n ist erheblich kleiner t

[53] Hier werden die Positionen im Reservoir zufällig gewählt. Damit die Wahrscheinlichkeit der Verdrängung für jedes Element gleich ist, muss der aktuelle Zeitpunkt und damit die Wahrscheinlichkeit eines Elements im Reservoir zu liegen bekannt sein.

Ein zusätzlicher Effekt zeigt sich auf Datenströmen über sehr großen oder unendlichen Grundgesamtheiten. Durch die feste Größe n des Reservoirs und der Eigenschaft, dass zu einem Zeitpunkt t die Wahrscheinlichkeit für ein Element x_{t+1} in die Stichprobe aufgenommen zu werden $\frac{n}{t+1}$ beträgt[54], haben es neue Datenwerte immer schwerer in die Stichprobe hinein zu kommen. Auf einem unendlichen Datenstrom mit $t \to \infty$ gilt sogar $\lim_{t\to\infty} \frac{n}{t} = 0$, d. h. die Stichprobe enthält mit sehr hoher Wahrscheinlichkeit nur sehr alte Elemente. Lokale und zeitliche Informationen gehen somit vollständig verloren. Im folgenden Kapitel werden diese Nachteile aufgegriffen und mit Hilfe eines neuen adaptiven Ansatzes gelöst.

[54] vgl. Satz 4.2

Kapitel 5

Online Aggregation Induktiver Statistiken

Der Verarbeitung von Datenströmen, die möglicherweise sogar unendlich sein können, wird in der schließenden Statistik keine Bedeutung beigemessen. Damit existieren auch keine allgemeinen Lösungsansätze für die Probleme, die das *one-pass*-Paradigma und das in der Regel beschränkte lokale Gedächtnis[1] mit sich bringen. In Kapitel 4 ist ein erster Schritt in diese Richtung aufgezeigt worden. Der Ansatz der speicherbasierten Verfahren hat jedoch einige entscheidende Nachteile. So ist es in der Regel nicht möglich, dynamisch auf veränderte Speicherbedürfnisse zu reagieren bzw. den verwendeten Speicher zur Laufzeit dynamisch anzupassen. Auch geht die implizite zeitliche Komponente der Daten, d. h. der Zeitpunkt der Verarbeitung, durch die zufällige Speicherung im Reservoir verloren. Damit ist auch keine angepasste Gewichtung von erst „kürzlich" verarbeiteten Daten möglich. In Abschnitt 4.4 sind diese und weitere Beschränkungen dieses Ansatzes bereits diskutiert worden.

Im folgenden Kapitel soll ein Ansatz entwickelt werden, der die oben genannten Probleme weitestgehend vermeidet. Dazu wird der Datenstrom logisch aufgebrochen und die entstandenen Segmente werden einzeln und unabhängig voneinander verarbeitet. Damit ist es leicht möglich, eine Gewichtung der einzelnen Segmente einzuführen und ein „lokales" Gedächtnis in dem Sinne zu etablieren, so dass eine zeitliche Nähe auch eine größere Gewichtung bedeutet. So ist zudem eine datenabhängige (*data-driven*) oder eine gleichmäßige Gewichtung leicht zu erreichen. Im folgenden Abschnitt wird die dazu notwendige Theorie erarbeitet und gezeigt, wie ein solcher Schätzer anzuwenden ist.

5.1 Allgemeiner Adaptiver Schätzer

Liefert ein Datenstrom seine Datenwerte in zufälliger oder annähernd zufälliger Reihenfolge bzw. kann eine der in Abschnitt 3.1 aufgezeigten Techniken verwendet werden, so enthält das Reservoir nach Abschluss der Initialisierungsphase bei Verwendung eines

[1] vgl. Abschnitt 2.3

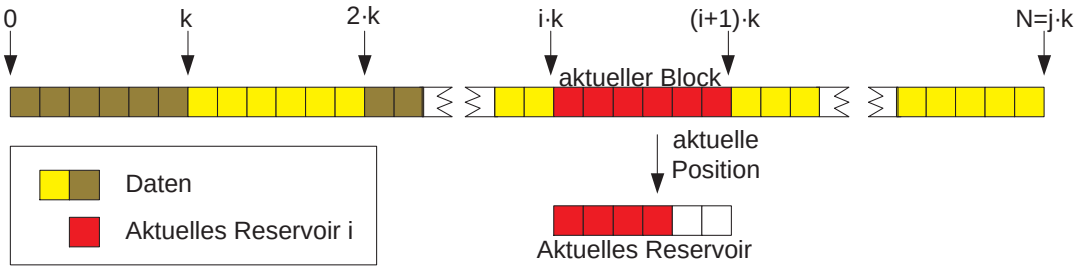

Abbildung 5.1: Segmentierung der Daten des Datenstroms in Blöcke der Größe k

speicherbasierten Stichprobenalgorithmus trivialerweise eine gültige *i.i.d.* Stichprobe[2]. Dies kann man nutzen, um einen *adaptiven* Schätzer zu etablieren, der alle Datenwerte, die verarbeitet werden, auch zur Berechnung des Schätzwertes bzw. der Schätzfunktion heranzieht.

Ist sichergestellt, dass die Daten in zufälliger oder bezüglich des zu aggregierenden Merkmals unkorrelierter Reihenfolge geliefert werden, kann der Datenstrom in Segmente der Größe k zerlegt werden, die alle eine *i.i.d.* Stichprobe der Grundgesamtheit enthalten. Abbildung 5.1 zeigt die Einteilung des *streams* in Abschnitte der Größe k. Setzt man voraus, dass die Größe der Grundgesamtheit N ein ganzzahlig Vielfaches der Reservoirgröße k ist, so ist die Einteilung eindeutig.

Definition 5.1 *Sei nun im folgenden $N \in \mathbb{N}$ die Größe der betrachteten Grundgesamtheit und gelte o. B. d. A. $N = j \cdot k$ mit $k \in \mathbb{N}_0$ der Größe der verwendeten Speicher und $j \in \mathbb{N}$ einem Laufindex. Sind nun $\hat{f}_i^{(k)}(x)$ mit $i = 1, 2, \ldots, j$ Schätzer für das i-te Segment der Größe k des Datenstroms, dann heißt*

$$\hat{g}_j(x) = \sum_{i=1}^{j} \omega_i \hat{f}_i^{(k)}(x) \tag{5.1}$$

adaptiver Schätzer *zum Zeitpunkt j zu den Gewichten ω_i falls*

$$\sum_{i=1}^{j} \omega_i = 1 \tag{5.2}$$

gilt.

Da alle Schätzer $f_i^{(k)}(x)$ nach Voraussetzung auf einer *i.i.d.* Stichprobe der Größe k der gleichen Grundgesamtheit arbeiten, erhält der adaptive Schätzer $\hat{g}_j(x)$ wichtige Eigenschaften der einzelnen Schätzer. So gilt folgender Satz.

Satz 5.2 *Sind die $f_i^{(k)}(x)$ erwartungstreu für alle $i = 1, 2, \ldots, j$ mit $E\left(f_i^{(k)}(x)\right) = \theta$, so ist auch $\hat{g}_j(x)$ erwartungstreu mit $E(\hat{g}_j(x)) = \theta$.*

[2] vgl. Abschnitt 4.2

Beweis:

$$\begin{aligned} E(\hat{g}_j(x)) &= E\left(\sum_{i=1}^{j} \omega_i \hat{f}_i^{(k)}(x)\right) \\ &= \sum_{i=1}^{j} \omega_i E\left(\hat{f}_i^{(k)}(x)\right) \\ &= \underbrace{\sum_{i=1}^{j} \omega_i}_{=1} \cdot \theta \\ &= \theta \end{aligned}$$

q.e.d.

Zwei der Eigenschaften, die ein Schätzer zur *Online Aggregation* erfüllen muss, sind die rekursive Berechenbarkeit der Schätzfunktion und die Gültigkeit des Schätzers zu jedem Zeitpunkt[3]. Damit ein *adaptiver Schätzer* gemäß Definition 5.1 *online* berechnet werden kann, muss somit folgender Satz gelten.

Satz 5.3 *Sei $\hat{g}_j(x)$ ein adaptiver Schätzer gemäß Formel (5.1). Dann existiert eine Darstellung der Form*

$$\hat{g}_i(x) = (1 - \tilde{\omega}_i)\hat{g}_{i-1}(x) + \tilde{\omega}_i \hat{f}_i(x) \tag{5.3}$$

für alle $i = 3, 4, 5, \ldots, j$ mit

$$\hat{g}_2(x) = (1 - \tilde{\omega}_2)\hat{f}_1(x) + \tilde{\omega}_2 \hat{f}_2(x)$$

für eine Folge von Gewichten $0 \leq \tilde{\omega}_i \leq 1$ für $i = 2, \ldots, j$. Zu jedem Zeitpunkt i ist $\hat{g}_i(x)$ ein adaptiver Schätzer für die ersten $(i \cdot k)$ Daten.

Beweis:

Es ist zu zeigen, dass für ein beliebiges aber festes j für alle $i = 2, \ldots, j$ eine rekursive Form für $\hat{g}_i(x) = \sum_{l=1}^{i} \omega_l^{(i)} \hat{f}_l(x)$ mit $\sum_{l=1}^{i} \omega_l^{(i)} = 1$ existiert. Der Beweis erfolgt mittels Induktion über i.

Induktionsanfang $(i = 2)$:
Es gilt nach Definition

$$\begin{aligned} \hat{g}_2(x) &= (1 - \tilde{\omega}_2)\hat{f}_1(x) + \tilde{\omega}_2 \hat{f}_2(x) \\ &= \sum_{l=1}^{2} \omega_l^{(2)} \hat{f}_l(x) \end{aligned}$$

mit $\omega_1^{(2)} := (1 - \tilde{\omega}_2)$ und $\omega_2^{(2)} := \tilde{\omega}_2$.

[3] vgl. Abschnitt 3.4

Gelte für i, zu zeigen für $i+1$ ($i \longrightarrow i+1$):

Es gilt

$$\begin{array}{lll} \hat{g}_{i+1}(x) & = & (1-\tilde{\omega}_{i+1})\hat{g}_i(x) + \tilde{\omega}_{i+1}\hat{f}_{i+1}(x) \\ & \overbrace{=}^{\text{nach Ind.-Vor.}} & (1-\tilde{\omega}_{i+1})\sum_{l=1}^{i}\omega_l^{(i)}\hat{f}_l(x) + \tilde{\omega}_{i+1}\hat{f}_{i+1}(x) \end{array}$$

mit den Gewichten $0 \leq \omega_l^{(i)} \leq 1$ und

$$\sum_{l=1}^{i}\omega_l^{(i)} = 1.$$

Damit bleibt zu zeigen, dass Gewichte $0 \leq \omega_l^{(i+1)} \leq 1$ existieren, so dass gilt:

$$\hat{g}_{i+1}(x) = \sum_{l=1}^{i+1}\omega_l^{(i+1)}\hat{f}_l(x)$$

Definiere folgende Gewichte

$$\begin{array}{lll} \omega_{i+1}^{(i+1)} & := & \tilde{\omega}_{i+1} \\ \omega_l^{(i+1)} & := & \omega_l^{(i)}(1-\tilde{\omega}_{i+1}) \text{ für } l = 1,\ldots,i \end{array}$$

Zu zeigen bleibt, dass

$$\sum_{l=1}^{i+1}\omega_l^{(i+1)} = 1$$

gilt. Also

$$\begin{array}{lll} \sum_{l=1}^{i+1}\omega_l^{(i+1)} & = & \sum_{l=1}^{i}\omega_l^{(i+1)} + \omega_{i+1}^{(i+1)} \\ & = & \sum_{l=1}^{i}\omega_l^{(i+1)} + \tilde{\omega}_{i+1} \\ & = & \sum_{l=1}^{i}(1-\tilde{\omega}_{i+1})\omega_l^{(i)} + \tilde{\omega}_{i+1} \\ & = & (1-\tilde{\omega}_{i+1})\underbrace{\sum_{l=1}^{i}\omega_l^{(i)}}_{=1 \text{ n.V.}} + \tilde{\omega}_{i+1} \\ & = & (1-\tilde{\omega}_{i+1}) + \tilde{\omega}_{i+1} \\ & = & 1 \end{array}$$

q.e.d.

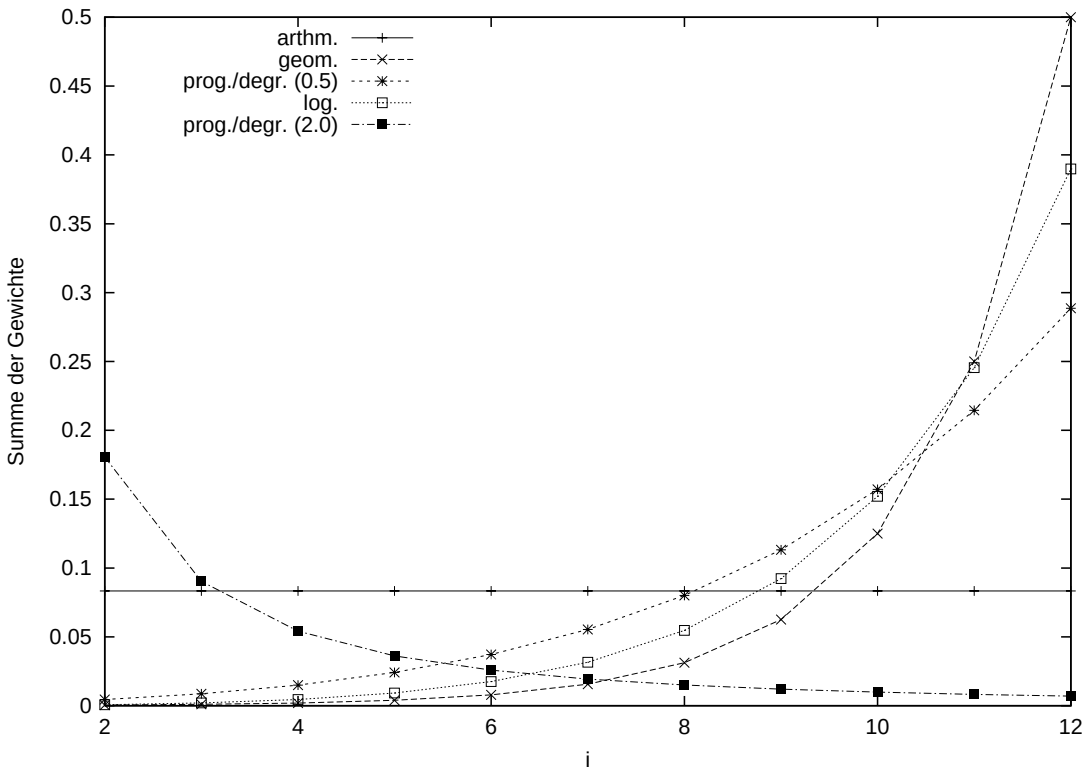

Abbildung 5.2: Laufende Gewichte zum Zeitpunkt $j = 12$

Der Beweis liefert zugleich eine Konstruktionsvorschrift für die Folge der Gewichte $\tilde{\omega}_l$, falls die geschlossene Darstellung als Summe bekannt ist. Damit lassen sich nun verschiedene Folgen von Gewichten $\tilde{\omega}_l$ angeben, die unterschiedliche Strategien den adaptiven Schätzer zu berechnen widerspiegeln.

Arithmetische Gewichtung: Wähle gleiche Gewichte für alle Block-Schätzer $\hat{f}_i(x)$, so dass

$$\hat{g}_j(x) = \frac{1}{j} \sum_{i=1}^{j} \hat{f}_i(x) \tag{5.4}$$

gilt. Damit gilt für ein Gewicht $\omega_i^{(j)}$ für den i-ten Schätzer zum Zeitpunkt j

$$\omega_i^{(j)} = \frac{1}{j} \text{ mit } i \geq 2 \tag{5.5}$$

und die Folge von Gewichten $\tilde{\omega}_i$ berechnet sich mittels

$$\tilde{\omega}_i = \frac{i-1}{i} \cdot \tilde{\omega}_{i-1}.$$

Für jedes $\hat{g}_i(x)$ mit $i = 3, \ldots, j$ gilt somit

$$\hat{g}_i(x) = \frac{i-1}{i} \hat{g}_{i-1}(x) + \frac{1}{i} \hat{f}_i(x)$$

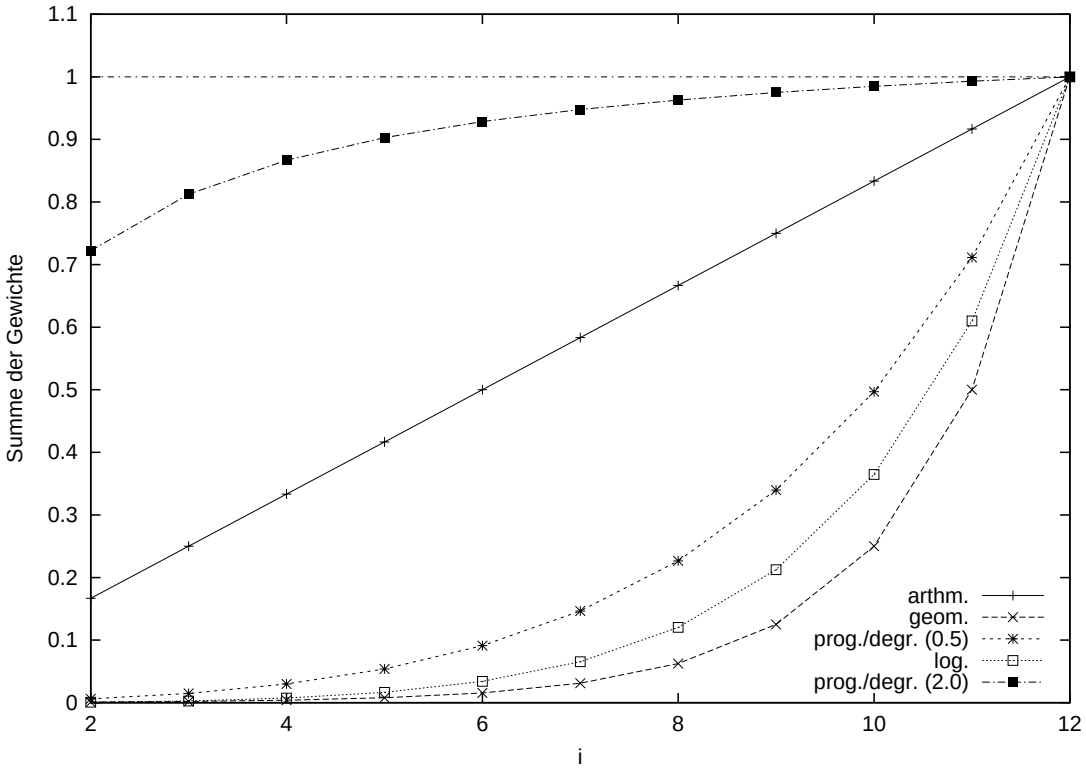

Abbildung 5.3: Summe der laufenden Gewichte zum Zeitpunkt $j = 12$

und

$$\hat{g}_2(x) = \frac{1}{2}\hat{f}_1(x) + \frac{1}{2}\hat{f}_2(x).$$

Geometrische Gewichtung: Wähle gleiche Gewichte für „neue“ Blockschätzer $\hat{f}_i(x)$, d. h. für ein beliebiges aber festes $c \in \mathbb{R}$ mit $c \geq 1$ gelte

$$\hat{g}_i(x) = \left(1 - \frac{1}{c}\right) \hat{g}_{i-1}(x) + \frac{1}{c}\hat{f}_i(x) \tag{5.6}$$

und

$$\hat{g}_2(x) = \left(1 - \frac{1}{c}\right) \hat{f}_1(x) + \frac{1}{c}\hat{f}_2(x).$$

Für das i-te Gewicht zum Zeitpunkt j gilt damit

$$\omega_i^{(j)} = \frac{1}{c}\left(1 - \frac{1}{c}\right)^{j-i} \text{ mit } i \geq 2.$$

Die Wahl solcher Gewichte liefert einen auch asymptotisch gültigen adaptiven Schätzer, der sich damit besonders für unendliche Grundgesamtheiten eignet.

Für den Schätzer gilt

$$\begin{aligned}
\hat{g}_j(x) &= \left(1-\frac{1}{c}\right)\hat{g}_{j-1}(x)+\frac{1}{c}\hat{f}_j(x)\\
&= \frac{1}{c}\hat{f}_j(x)+\left(\frac{c-1}{c}\right)\left(\frac{1}{c}\hat{f}_{j-1}(x)\right.\\
&\quad \left.+\left(\frac{c-1}{c}\right)\left(\ldots\left(\left(\frac{c-1}{c}\right)\hat{f}_1(x)+\frac{1}{c}\hat{f}_2(x)\right)\cdots\right)\right)\\
&= \frac{1}{c}\hat{f}_j(x)+\frac{1}{c}\left(\frac{c-1}{c}\right)\hat{f}_{j-1}(x)+\frac{1}{c}\left(\frac{c-1}{c}\right)^2\hat{f}_{j-2}(x)+\cdots\\
&\quad +\frac{1}{c}\left(\frac{c-1}{c}\right)^{j-2}\hat{f}_2(x)+\left(\frac{c-1}{c}\right)^{j-1}\hat{f}_1(x)\\
&= \sum_{l=0}^{j-2}\frac{1}{c}\left(\frac{c-1}{c}\right)^l\hat{f}_{j-l}+\left(\frac{c-1}{c}\right)^{j-1}\hat{f}_1.
\end{aligned}$$

Damit gilt für die Gewichte

$$\begin{aligned}
\sum_{l=0}^{j-2}\frac{1}{c}\left(\frac{c-1}{c}\right)^l+\left(\frac{c-1}{c}\right)^{j-1} &= \frac{1}{c}\underbrace{\sum_{l=0}^{j-2}\left(\frac{c-1}{c}\right)^l}_{\textit{endl. geometrische Reihe}}+\left(\frac{c-1}{c}\right)^{j-1}\\
&= \frac{1}{c}\cdot\left(\frac{1-\left(\frac{c-1}{c}\right)^{j-1}}{1-\left(\frac{c-1}{c}\right)}\right)+\left(\frac{c-1}{c}\right)^{j-1}\\
&= \frac{1}{c}\cdot\left(\frac{1-\left(\frac{c-1}{c}\right)^{j-1}}{\left(\frac{1}{c}\right)}\right)+\left(\frac{c-1}{c}\right)^{j-1}\\
&= 1-\left(\frac{c-1}{c}\right)^{j-1}+\left(\frac{c-1}{c}\right)^{j-1}=1.
\end{aligned}$$

Selbst für unendliche Datenströme mit $j\to\infty$ ist die Summe der Gewichte gleich eins[4]. Die Wahl einer solchen Folge von Gewichten entspricht einer abnehmenden Gewichtung von „weiter entfernt“ liegenden Daten, d. h. Daten, deren Verarbeitung länger zurück liegt, werden weniger stark gewichtet. Die Wahl $c=1$ führt zu einer Gewichtung eines „neuen“ Schätzers mit 1, und demzufolge eine Gewichtung des „alten“ Schätzers mit 0.

Progressive und Degressive Gewichtung: Durch eine Verallgemeinerung der Gewichte aus Formel (5.4) erhält man eine progressive bzw. degressive Gewichtung neuer Schätzer je nach Wahl der Folge der Parameter. Wählt man zum Zeipunkt j für das Gewicht

$$\tilde{\omega}_j=\omega_j^{(j)}=\frac{1}{j^{\alpha_j}}$$

[4]Mit der Konvergenz der geometrischen Reihe leicht nachzurechnen.

mit $\alpha_j > 0$, so gilt für die laufenden Gewichte

$$\omega_i^{(j)} = \frac{1}{i^{\alpha_i}} \prod_{r=i+1}^{j} \left(1 - \frac{1}{r^{\alpha_r}}\right) \text{ mit } i \geq 2$$

und

$$\omega_1^{(j)} = \left(1 - \frac{1}{2^{\alpha_2}}\right) \prod_{r=3}^{j} \left(1 - \frac{1}{r^{\alpha_r}}\right).$$

Gilt nun $\alpha_i < 1$ für alle i, so erhält man eine zuerst hohe aber mit j abnehmende Gewichtung neuer Schätzer. In den Abbildungen 5.2 und 5.3 ist dies für $j = 12$ und $\alpha_i \equiv \frac{1}{2}$ für alle $i = 1, 2, \ldots, j$ zu sehen. Eine degressive Gewichtung wird mit $\alpha_i > 1$ für alle i erreicht. In Abbildung 5.2 wird dies am Beispiel für $\alpha_i \equiv 2$ deutlich.

Logarithmische Gewichtung: Einen zur progressiven Gewichtung mit $\alpha < 1$ ähnlichen Verlauf der Folge der Gewichte kann durch die Wahl

$$\tilde{\omega}_j = \omega_j^{(j)} = \frac{1}{\ln(j+1)}$$

erreicht werden. Damit gilt für die Folge der Gewichte

$$\omega_i^{(j)} = \frac{1}{\ln(i+1)} \prod_{r=i+1}^{j} \left(1 - \frac{1}{\ln(r+1)}\right) \text{ mit } i \geq 2$$

und

$$\omega_1^{(j)} = \left(1 - \frac{1}{\ln(3)}\right) \prod_{r=3}^{j} \left(1 - \frac{1}{\ln(r+1)}\right).$$

Durch die Wahl des Ausdrucks $(i+1)$ für den Logarithmus im Nenner der Gewichte wird ein ungültiges Gewicht für $i = 2$ vermieden, da $\frac{1}{\ln 2} > 1$ gilt.

Über die hier aufgeführten Gewichte hinaus sind sicherlich noch weitere denkbar und sinnvoll. Datenabhängige Gewichte sind ebenso möglich, wie das Ausnutzen von *a priori* Informationen über die Daten.

Die laufenden Gewichte $w_i^{(j)}$ der oben vorgestellten Varianten, die Gewichte in Formel (5.1) zu wählen, werden in Abbildung 5.2 exemplarisch für den Zeitpunkt $j = 12$ dargestellt. Für die geometrische Folge von Gewichten ist dazu der Parameter $c = 2$ und für die progressive/degressive Folge $\alpha_i = \alpha = \frac{1}{2}$ beziehungsweise $\alpha_i = \alpha = 2$ gewählt worden. Deutlich ist der degressive bzw. progressive Charakter der Kurven bei der Wahl eines entsprechenden Parameters α zu erkennen. Jedoch wird auch bei laufendem Zeitpunkt j die geometrische und die logarithmische Folge die „letzten“ Schätzer immer stärker gewichten, egal welcher Parameter α gewählt worden ist. Abbildung 5.3 zeigt die laufenden Summen der Gewichte zum Zeitpunkt i. Hier werden die charakteristischen Eigenschaften der verschiedenen Folgen der Gewichte noch einmal deutlich.

Asymptotisch gilt für die Folge der geometrischen Gewichte

$$\lim_{j\to\infty} \tilde{\omega}_j = \lim_{j\to\infty} \omega_j^{(j)} = \lim_{j\to\infty} \frac{1}{c} = \frac{1}{c}.$$

Bei den anderen vorgestellten Verfahren verschwinden die Gewichte der „Schätzer“ asymptotisch, d. h. es gilt

$$\lim_{j\to\infty} \tilde{\omega}_j = \lim_{j\to\infty} \omega_j^{(j)} = 0,$$

da der Nenner in Abhängigkeit vom Zeitpunkt j eine streng monoton wachsende Folge bildet, und somit die Gewichte in j monoton abnehmen.[5]

5.2 Komprimierung reeller Funktionen mit Splines

Viele Methoden der mathematischen Statistik haben ein Problem gemein, welches ihren Nutzen im Rahmen eines Datenbanksystems stark einschränkt oder ihre Verwendung gar verhindert. Wie schon in Abschnitt 2.2 gesagt, existieren für die meisten Verfahren keine Betrachtungen zur Komplexität, zur Skalierbarkeit und zum Bedarf an Speicherplatz im Hauptspeicher oder gegebenenfalls sogar auf dem Sekundärspeicher. So wird bei fast allen Verfahren der induktiven Statistik implizit vorausgesetzt, dass die verwendete Stichprobe im Hauptspeicher vorliegt und dass ein wahlfreier Zugriff (*random access*) auf die Stichprobenwerte möglich ist. Ein Argument für diese Vorgehensweise ist sicherlich der rasante technische Fortschritt, vor allem im Bereich der Speichertechnik. So sind heute Hauptspeichergrößen in Desktopsystemen von 512 Megabyte bis zu ein Gigabyte keine Ausnahmen mehr und Serversysteme mit vier bis acht Prozessoren und zwei bis vier Gigabyte Hauptspeicher durchaus als üblich anzusehen. Jedoch ist man keinesfalls alleiniger Benutzer auf solchen Systemen und hat für seine Aufgaben alle Systemressourcen exklusiv zur Verfügung. Vielmehr stellt der Server für jeden Benutzer bzw. für jeden Prozess eine feste Speichergröße und eine beschränkte Prozessorzeit zur Verfügung. Die Größe eines solchen Leistungsfensters hängt von der aktuellen Auslastung des Systems ab und kann so stark variieren. Damit müssen auch statistische Verfahren, die in einem solchen Szenario Verwendung finden sollen, sich der Frage nach einem optimalen oder wenigstens effizienten Schätzer zu einer verfügbaren Speichergröße stellen. Eine Möglichkeit, statistische Methoden nachträglich einen festen und beschränkten Speicherplatzbedarf unterstützen zu lassen, ist der Einsatz von verlustfreien oder verlustbehafteten Komprimierungstechniken. Da im Rahmen dieser Arbeit ausschließlich solche Verfahren der nicht-parametrischen Statistik Verwendung finden, die das Prinzip der Glättung verwenden, bietet sich der Einsatz von Splines zur verlustbehafteten Komprimierung solcher Funktionen an.

Ziel ist es somit, zu einer vorgegebenen, festen und beschränkten Speichergröße den optimalen Schätzer zu einem gegebenen Problem zu finden. Hier wird als einfaches Modell für den verwendeten Speicher die Zahl der zu speichernden Werte angenommen.

[5] vgl. das Leibniz-Konvergenz-Kriterium

Ein nativer Kernschätzer, der über einer Stichprobe der Größe n mit einer Bandbreite h definiert ist, verbraucht somit einen Speicher der Größe $n + 1$.

In [KJJ99] haben Korn et al. erstmals eine Kombination von Kernschätzern und kubischen Splines im Bereich der Selektivitätsschätzung verwendet, um kernbasierte Verfahren mit Histogrammen unter gleichen Speicherbedingungen zu vergleichen. Im folgenden Abschnitt wird die dazu notwendige Theorie erarbeitet und einige Aussagen über den Speicherbedarf kubischer Splines gemacht.

5.2.1 Interpolation mit natürlichen kubischen Splines

Im folgenden Abschnitt wird kurz die Interpolation einer Funktion mit natürlichen kubischen Splines erläutert und eine Fehlerabschätzung angegeben, die für die Verwendung von Splinefunktionen zur verlustbehafteten Komprimierung benötigt wird. Zu den Beweisen der hier formulierten Sätze und zum tiefer gehenden Verständnis sei auf die einschlägige Literatur verwiesen. Eine gute Einführung erhält man u. a. in [Sti76] Seite 216 ff., [Sch86] Seite 125 ff. und [DH93] Seite 234 ff. Im Wesentlichen orientieren sich aber sowohl die Definitionen als auch die formulierten Sätze an [Sto89] und [DH93].

Formuliert man das Interpolationsproblem als Variationsaufgabe[6] mit der Forderung nach verschwindenden vierten Ableitungen der Interpolationsfunktion, so kann man das Interpolationsproblem mit folgenden Bedingungen charakterisieren:

$$s^{(4)}(x) = 0 \quad \text{für alle} \quad x \neq x_0, x_1, \cdots, x_n \tag{5.7}$$

$$s''(x_i + 0) = s''(x_i - 0) \quad \text{für} \quad i = 1, 2, \cdots, n-1 \tag{5.8}$$

$$s''(x_0) = 0 = s''(x_n) \tag{5.9}$$

mit $s(x)$ die gesuchte interpolierende Funktion über einem Gitter $x_0 \leq x_1 \leq \cdots \leq x_n$. Durch Bedingung (5.7) wird der Grad des Polynoms auf jedem Teilintervall (x_i, x_{i+1}) auf drei festgelegt. Bedingung (5.8) fordert die Stetigkeit der zweiten Ableitung der Interpolationsfunktion an den inneren Stützstellen und (5.9) sagt, dass an den beiden Endpunkten des Gitters die zweite Ableitung verschwindet. Bedingung (5.9) stellt für eine Interpolationsaufgabe eine „natürliche“[7] Forderung dar, woher der Name der natürlichen kubischen Splineinterpolation resultiert.

Durch diese drei Eigenschaften ist die interpolierende Splinefunktion zusammmen mit der Interpolationseigenschaft eindeutig bestimmt. Im Folgenden wird die Interpolation mit natürlichen kubischen Splines formalisiert und eine Berechnungsvorschrift angegeben.

Definition 5.4 *Sei $\Delta := a = x_0 < x_1 < x_n = b$ eine Unterteilung des Intervalls $[a, b]$. Unter einer zu Δ gehörigen **kubischen Splinefunktion** versteht man die reelle Funktion $S_\Delta : [a, b] \to \mathbb{R}$ mit den Eigenschaften*

1. *$S_\Delta \in C^2[a, b]$, d. h. S_Δ ist auf $[a, b]$ zweimal stetig differenzierbar.*

[6] vgl. [Sti76] Seite 217
[7] vgl. [Sti76] Seite 218

2. *Auf jedem Teilintervall* $[x_i, x_{i+1}]$, $i = 0, 1, \cdots, n-1$ *stimmt* S_Δ *mit einem Polynom dritten Grades überein.*

Somit besteht eine Splinefunktion aus n kubischen Polynomen, die so zusammengesetzt sind, dass S_Δ selbst und ihre beiden ersten Ableitungen an den Gitterpunkten keine Sprungstellen besitzen. Damit ist eine solche Funktion glatt über $[a, b]$ und eignet sich so besonders, um eine glatte bzw. geglättete Funktion wie einen Kernschätzer auf einem Intervall zu approximieren.

Bemerkung 5.5 *Definition 5.4 ist auf allgemeine Polynome vom Grad k erweiterbar. Kubische Splines haben jedoch sowohl Vorteile in der Anwendung, als auch gute theoretische Eigenschaften, so dass im Folgenden nur solche verwendet werden.*[8]

Ist $S_\Delta(x_i) = f(x_i) = y_i$, $i = 0, 1, \cdots, n$ mit f eine Funktion auf $[a, b]$ und $y_i \in \mathbb{R}$, so ist die Splinefunktion $S_\Delta := S_{\Delta,f}$ eine Approximation für die Punkte y_i bzw. für die Funktion f.

Satz 5.6 *Sei nun f eine zweimal total stetige Funktion*[9] *auf* $[a, b]$ *mit* $f(x_i) = y_i$, Δ *ein Gitter gemäß Definition 5.4 und* $S_{\Delta,f}(x)$ *eine Splinefunktion auf* $[a, b]$*, dann gilt*

$$||f - S_{\Delta,f}||^2 = ||f||^2 - ||S_{\Delta,f}||^2 \geq 0 \tag{5.10}$$

für jede Splinefunktion $S_{\Delta,f}$*, wenn zusätzlich eine der folgenden Eigenschaften erfüllt ist:*

1. $S''_{\Delta,f}(a) = S''_{\Delta,f}(b) = 0$
2. $f'(a) = S'_{\Delta,f}(a)$ *und* $f'(b) = S'_{\Delta,f}(b)$

In jedem dieser Fälle ist die Splinefunktion durch Δ *und f eindeutig bestimmt und existiert. Zum Beweis siehe u. a. [Sto89], Seite 84 f.*

Die Aussage von Satz (5.6) kann man auch so formulieren, dass $S_{\Delta,f}$ die Kurve mit der minimalen Gesamtkrümmung aller total stetigen Funktionen auf $[a, b]$ mit der Interpolationseigenschaft $f(x_i) = y_i$ ist. Damit ist $S_{\Delta,f}$ die „glatteste“ unter allen Interpolationsfunktionen und qualifiziert sich dadurch zusätzlich zur Komprimierung glatter Funktionen.

Satz 5.7 *Sei* $S_{\Delta,f}$ *die interpolierende Splinefunktion auf dem Gitter* $\Delta = a = x_0 < x_1 < x_n = b$ *zu* $f(x) \in C^4[a, b]$ *mit*

$$h := \max_{i=0}^{n-1} (x_{i+1} - x_i),$$

so gilt

$$||S^{(j)}_{\Delta,f} - f^{(j)}||_\infty \leq \frac{3}{8} h^{4-j} ||f^{(4)}||_\infty \text{ für } j = 0, 1, 2 \tag{5.11}$$

mit der Randbedingung $S''_{\Delta,f}(a) = f''(a)$ *und* $S''_{\Delta,f}(b) = f''(b)$*. Insbesonders gilt also*

$$||S_{\Delta,f} - f||_\infty \leq \frac{3}{8} h^4 ||f^{(4)}||_\infty. \tag{5.12}$$

[8] vgl. hierzu u. a. [Sto89]

[9] Zur Definition von total stetig vgl. [Sto89], Seite 82 ff.

Damit konvergiert eine Splinefunktion für eine glatte Funktion $f(x)$ gegen diese bei feiner werdendem Gitter. Weiter gilt sogar, dass man keinerlei Voraussetzungen für das Gitter Δ benötigt und dass auch die Ableitungen der Splinefunktion gegen die Ableitungen der Funktion f konvergieren.

Bemerkung 5.8 *Da der größte Gitterabstand h minimal für ein äquidistantes Gitter ist, so ist die Konvergenzgeschwindigkeit für ein äquidistantes Gitter Δ am größten. Zudem folgt direkt aus Formel (5.11), dass gilt:*

$$|S_{\Delta,f} - f| = O(h^4)$$

Im folgenden Abschnitt wird die Konstruktionsvorschrift für kubische Splines in der Bézier-Bernstein-Form vorgestellt.

5.2.2 Konstruktion und Berechnung von kubischen Splinefunktionen

Es existieren verschiedene Verfahren zur Berechnung einer Splinefunktion zur Interpolation mit unterschiedlichen Stärken und Schwächen. Zur Berechnung der benötigten Koeffizienten über einem äquidistanten Gitter bietet sich die lokale Bézier-Darstellung mit Hilfe der B-Splines und eine Lösung des entstehenden linearen Gleichungssystems (LGS) über die Gauss-Eliminierung an.[10]

Definition 5.9 *Die Darstellung einer interpolierenden Splinefunktion gemäß*

$$s(x) = s_i(x) = \sum_{j=0}^{3} b_{3i+j} \frac{1}{h^3} \binom{3}{j} x^j (h_i - x)^{3-j} \text{ für } x \in [x_i, x_{i+1}] \tag{5.13}$$

mit

$$h_i := x_{i+1} - x_i$$

heißt ***lokale Bézier-Darstellung****.*

Die b_i mit $i = 0, 1, \ldots, 3n$ heißen **Bézier-Koeffizienten** und bestimmen den Spline eindeutig. Diese Koeffizienten sollen nun bestimmt werden.

Aus der Stetigkeit und der Darstellung mittels Formel (5.13) folgt direkt, dass

$$b_{3i} = \frac{h_i}{h_{i-1} + h_i} b_{3i-1} + \frac{h_{i-1}}{h_{i-1} + h_i} b_{3i+1}.$$

Ferner gilt aufgrund der C^2-Stetigkeit, dass Punkte d_i für s existieren, so dass[11]

$$b_{3i+2} = -\frac{h_i}{h_{i-1}} d_i + \frac{h_{i-1} + h_i}{h_{i-1}} b_{3i+1}$$

$$b_{3i-2} = \frac{h_{i-1} + h_i}{h_i} b_{3i-1} - \frac{h_{i-1}}{h_i} d_i.$$

[10] vgl. hierzu [DH93] Seite 250

[11] vgl. [DH93] Seite 247

Zusammen folgt daraus

$$b_{3i+1} = \frac{h_i + h_{i+1}}{h_{i-1} + h_i + h_{i+1}} d_i + \frac{h_{i-1}}{h_{i-1} + h_i + h_{i+1}} d_{i+1}$$

$$b_{3i-1} = \frac{h_{i-2} + h_{i-1}}{h_{i-2} + h_{i-1} + h_i} d_i + \frac{h_i}{h_{i-2} + h_{i-1} + h_i} d_{i-1}.$$

Für den Rand gelte nun

$$\begin{aligned} h_{-1} := h_n &:= 0 \\ d_0 &:= b_1 \\ d_{-1} &:= b_0 \\ d_n &:= b_{3n-1} \\ d_{n+1} &:= b_{3n}. \end{aligned}$$

Damit sind die Bézier-Koeffizienten und so auch der durch sie definierte Spline eindeutig bestimmt. Wird jetzt die Interpolationsbedingung $f(x_i) = y_i = s(x_i)$ eingesetzt, so folgt sofort für den Rand

$$d_{-1} = y_0 \quad \text{und} \quad d_{n+1} = y_n.$$

Die verbleibenden d_i müssen somit folgendem linearen Gleichungssystem genügen:

$$\begin{pmatrix} 1 & & & & & \\ \alpha_1 & \beta_1 & \gamma_1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & \alpha_{n-1} & \beta_{n-1} & \gamma_{n-1} & \\ & & & & 1 & \end{pmatrix} \cdot \begin{pmatrix} d_0 \\ \vdots \\ \\ \vdots \\ d_n \end{pmatrix} = \begin{pmatrix} b_1 \\ (h_0 + h1)y_1 \\ \vdots \\ (h_{n-2} + h_{n-1})y_{n-1} \\ b_{3n-1} \end{pmatrix} \tag{5.14}$$

mit

$$\alpha_i := \frac{h_i^2}{h_{i-2} + h_{i-1} + h_i} \tag{5.15}$$

$$\beta_i := \frac{h_i(h_{i-2} + h_{i-1})}{h_{i-2} + h_{i-1} + h_i} + \frac{h_{i-1}(h_i + h_{i+1})}{h_{i-1} + h_i + h_{i+1}} \tag{5.16}$$

$$\gamma_i := \frac{h_{i-1}^2}{h_{i-1} + h_i + h_{i+1}}. \tag{5.17}$$

Die verbleibenden Unbekannten in diesem Lösungssystem sind die Bézier-Koeffizienten b_1 und b_{3n-1}. Diese kann man einfach aus der Randbedingung berechnen. Für die *vollständige* Splineinterpolation, also für

$$s'(a) = f'(a) \quad \text{und} \quad s'(b) = f'(b),$$

ergibt sich[12]

$$b_1 = \frac{h_0}{3} f'(a) + y_0 \quad \text{und} \quad b_{3n-1} = -\frac{h_{n-1}}{3} f'(b) + y_n, \tag{5.18}$$

[12] Zur Herleitung siehe [DH93] Seite 249

d. h. zur vollständigen Berechnung werden die Ableitungen der zu interpolierenden Funktion f in den Rändern des Gitters Δ benötigt.

Für die *natürliche* Splineinterpolation mit

$$s''(a) = s''(b) = 0$$

ergibt sich

$$b_1 = y_0 \quad \text{und} \quad b_{3n-1} = y_n. \tag{5.19}$$

Das zu lösende lineare Gleichungssystem ist regulär und gut konditioniert und durch seine Form einfach und effizient mit Hilfe des Gauß'schen Eliminierungsverfahrens zu berechnen.[13] Für den Speicherbedarf gilt folgender

Satz 5.10 *Der Speicherbedarf zur Berechnung einer vollständigen oder natürlichen Splineinterpolation auf einem Gitter $\Delta = x_0 < \cdots < x_n$ mit Hilfe des oben entwickelten Verfahrens ist $\Theta(n)$.*

Beweis:
Der Speicherbedarf ergibt sich direkt aus dem LGS (5.14) bzw. der Konstruktion. Die Polynome vom Grad drei sind durch jeweils drei Bézier-Koeffizienten im Intervall $[x_i, x_{i+1})$ plus dem Koeffizienten für x_{i+1}, welches auch zur Berechnung des Polynoms auf dem nachfolgenden Intervall benötigt wird, bestimmt. Für n Polynome müssen somit $3n$ Koeffizienten für $i = 0, \cdots, n-1$ gespeichert werden. Dazu kommt der Koeffizient für den „letzten" Gitterpunkt x_n. Zusätzlich muss das Gitter mit $(n+1)$ Werten gespeichert werden. Das ergibt zusammen einen Speicherbedarf von $4n-2$, also $\Theta(n)$. Auf einem äquidistanten Gitter reduziert sich der Speicherbedarf für das Gitter auf den Startwert $a = x_0$ und die Gitterbreite h. Damit müssen für ein solches Gitter nur $3n+3$ Werte gespeichert werden. **q.e.d.**

Der Aufwand zur Berechnung der Bézier-Koeffizienten ergibt sich durch das lineare Gleichungssystem aus (5.14), welches sich durch seine tridiagonale Form mit Hilfe des Gauß'schen Eliminierungsverfahrens in $3n$ Schritten berechnen lässt. Dazu kommt der Aufbau des Gleichungssystems und die Berechnung der Bézier-Koeffizienten aus dem Lösungsvektor. Mit diesen zusätzlich benötigten $2n$ Berechnungsschritten sind zusammen $5n$ Schritte notwendig. Der Aufwand zur Berechnung eines Wertes der Interpolationsfunktion $s(x)$ an der Stelle $x \in [a, b]$ setzt sich zusammen aus einer binären Suche auf dem Gitter zur Bestimmung des zu nutzenden Intervalls und der Berechnung des Polynoms dritten Grades in der lokalen Bézier-Form nach Definition 5.9. Somit ist der Aufwand gegeben durch $\log n + 4$. Alternativ kann das Intervall mit Hilfe einer Interpolationssuche bestimmt werden, welche einen erwarteten mittleren Aufwand von $\log\log n$ benötigt. Damit reduziert sich der Aufwand zur Berechnung von $s(x)$ an x im Mittel auf $\log\log n + 4$. Im Falle eines äquidistanten Gitters kann das Intervall direkt berechnet werden bzw. liefert die Interpolationssuche bereits nach dem ersten Schritt das richtige Intervall. Somit ist der Aufwand auf einem solchen Gitter konstant und damit $O(1)$.

[13] vgl. z. B. [Kau89]

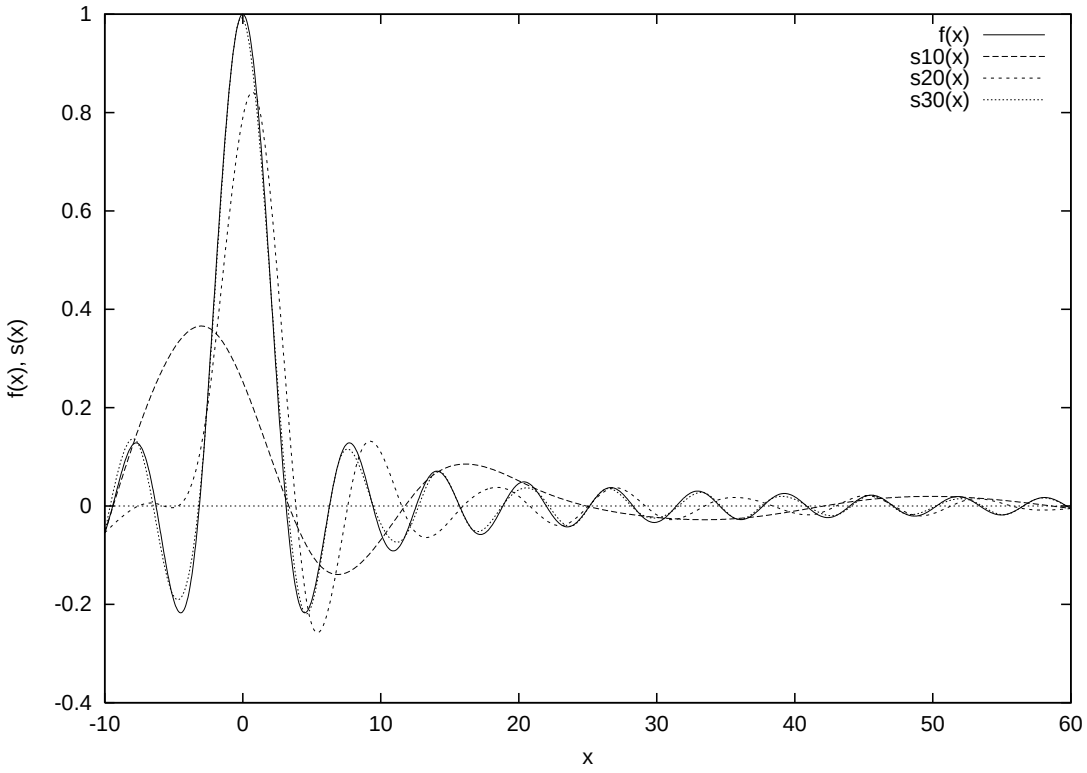

Abbildung 5.4: Approximation einer reellen Funktion mit Splinefunktionen unterschiedlicher Auflösung

5.2.3 Komprimierung mit Splines

In diesem Abschnitt soll gezeigt werden, wie die in den Abschnitten 5.2.1 und 5.2.2 vorgestellte Splineinterpolation zur verlustbehafteten Komprimierung von glatten Funktionen verwendet werden kann. Die Funktion

$$f(x) = \frac{\sin x}{x} \tag{5.20}$$

soll auf dem Intervall $[-10, 60]$ mit Hilfe einer natürlichen kubischen Splinefunktion approximiert werden. Diese Funktion eignet sich gut zur Veranschaulichung der Vorgehensweise aufgrund ihres glatten Verlaufs trotz des ozsillierenden Charakters der Funktion.

Die Abbildungen 5.4 und 5.5 zeigen die Leistungsfähigkeit der Splineinterpolation. Dazu ist die Funktion aus Formel (5.20) auf einem äquidistanten Gitter mit unterschiedlichen Auflösungen von 10, 20, 30 und 60 Stützstellen ausgewertet worden. Das Gitter und die korrespondierenden Funktionswerte dienen als Parameter für die zu konstruierende Splinefunktion.[14] Der Spline $s_{10}(x)$ ist über einem Gitter mit 10 Interpolationswerten aufgebaut, $s_{20}(x)$ mit 20, $s_{30}(x)$ mit 30 und $s_{60}(x)$ mit 60 Gitterpunkten auf $[-10, 60]$. Abbildung 5.4 zeigt die zu interpolierende Funktion und die Splinefunk-

[14] Hier bezeichnet s_n eine Splinefunktion mit n Stützstellen.

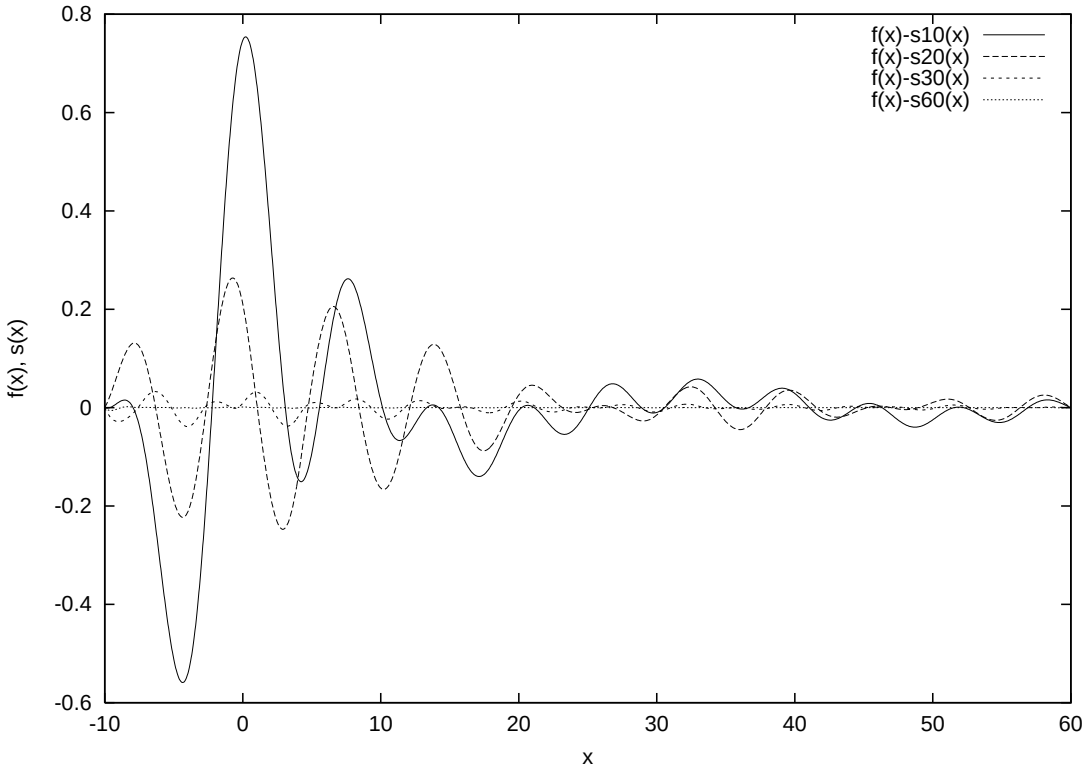

Abbildung 5.5: Fehler der Approximation einer reellen Funktion mit Splinefunktionen unterschiedlicher Auflösung

tionen eins bis drei. Splinefunktion vier fehlt in dieser Abbildung, da sie die Funktion bereits so gut approximiert, dass die Graphen der Funktionen nicht zu unterscheiden sind. In Abbildung 5.5 ist der Fehler der Interpolationsfunktionen eins bis vier, d. h. der Abstand $(f(x) - s_n(x))$ mit $n = 10, 20, 30, 60$, abgetragen. Hier zeigt sich noch einmal die Leistungsfähigkeit der Splineinterpolation anhand von Splinefunktion $s_{60}(x)$, die trotz des „extremen“ Verlaufs der zweiten Ableitung der Funktion $f(x)$ nahezu keinen Fehler aufweist.

5.2.4 Splines in XXL

Die Bibliothek XXL stellt sowohl die natürliche als auch die vollständige Splineinterpolation mit kubischen Splines in der Bézier-Darstellung zur Verfügung. Im Paket `xxl.statistics` findet sich die abstrakte Klasse `CubicBezierSpline`, die dem Konzept von XXL folgend von `xxl.functions.Function` erbt und zusätzlich aus Gründen der Geschwindigkeit `xxl.statistics.RealFunction` implementiert. Die Klassen `Rim1CubicBezierSpline` und `Rim2CubicBezierSpline` implementieren die abstrakte Methode `solveLGS()` aus `CubicBezierSpline`, die das benötigte Gleichungssystem aus den gegebenen Parametern konstruiert und löst, und stellen damit die natürliche bzw. vollständige Spline-Interpolation zur Verfügung. Die benötigten Parameter zur Instanziierung einer Spline-Funktion sind die Gitterpunkte und die dazu gehörenden

Funktionswerte der zu interpolierenden Funktion[15]. Im Falle der vollständigen Splineinterpolation müssen zusätzlich die Werte der ersten Ableitung an den Endpunkten des Gitters angegeben werden.

Steht die zu interpolierende Funktion als `xxl.functions.Function` oder als `xxl.statistics.RealFunction` zur Verfügung, kann die Funktion mit Hilfe der Methoden `equiGrid( double, double, int)` der Klasse `xxl.util.DoubleArrays` und `evalReal1DFunction( double[], Function)` der Klasse `xxl.util.Statistics` auf einem äquidistanten Gitter effizient ausgewertet werden. So kann für die Funktion `f` mit

```
double [] grid = xxl.util.DoubleArrays.equiGrid( -10.0, 60.0, 60);
Function spline = new Rim1CubicBezierSpline(
    grid,
    xxl.util.Statistics.evalReal1DFunction( grid, f)
);
```

ein Spline für die natürliche Splineinterpolation auf einem äquidistanten Gitter mit 60 Stützstellen erzeugt werden.

Die Berechnung der benötigten Bézier-Koeffizienten und die Lösung des tridiagonalen linearen Gleichungssystems erfolgt erst bei der ersten Auswertung der Interpolationsfunktion. Zudem ist die Suche nach den Gitterpunkten a und b mit $a \leq x \leq b$ für eine Auswertung der Interpolationsfunktion an der Stelle x mittels einer Interpolationssuche im Gitter effizient realisiert[16].

5.3 Vom speicherbasierten zum größenbeschränkten adaptiven Schätzer

In Kapitel 4 sind einige induktive Statistiken entwickelt worden, die mit Hilfe der Technik des speicherbasierten Stichprobenziehens direkt auf Datenströmen anwendbar sind. Die Vor- und Nachteile sind bereits im Abschnitt 4.4 ausführlich diskutiert worden. Der Hauptvorteil liegt im begrenzten und von der tatsächlichen Anzahl der Datenwerte unabhängigen Speicherplatzbedarf. Ein entscheidender Nachteil ist das „Vergessen" bereits verarbeiteter Datenwerte.

Der im Abschnitt 5.1 entwickelte allgemeine adaptive Schätzer über einem Datenstrom beseitigt den wichtigsten Nachteil der speicherbasierten Techniken, gibt aber gleichzeitig den Hauptvorteil des beschränkten und unabhängigen Speicherplatzbedarfs auf. Damit ein adaptives Verfahren trotzdem zum Einsatz kommen kann, gibt es prinzipiell zwei unterschiedliche Vorgehensweisen. Entweder muss die Datengrundlage, d. h. die Stichprobe und alle weiteren benötigten Werte des Schätzers ausreichend komprimiert werden, oder der Schätzer selbst muss einem vorgegebenen beschränkten Speicherplatzbedarf genügen.

[15] Die Gitterpunkte müssen sortiert in aufsteigender Reihenfolge gegeben sein.

[16] Der Aufwand einer Interpolationssuche in einem sortierten Array ist im Mittel $\Theta(\log\log n)$. Auf einem äquidistanten Gitter ist der Aufwand konstant, d. h. $\Theta(1)$.

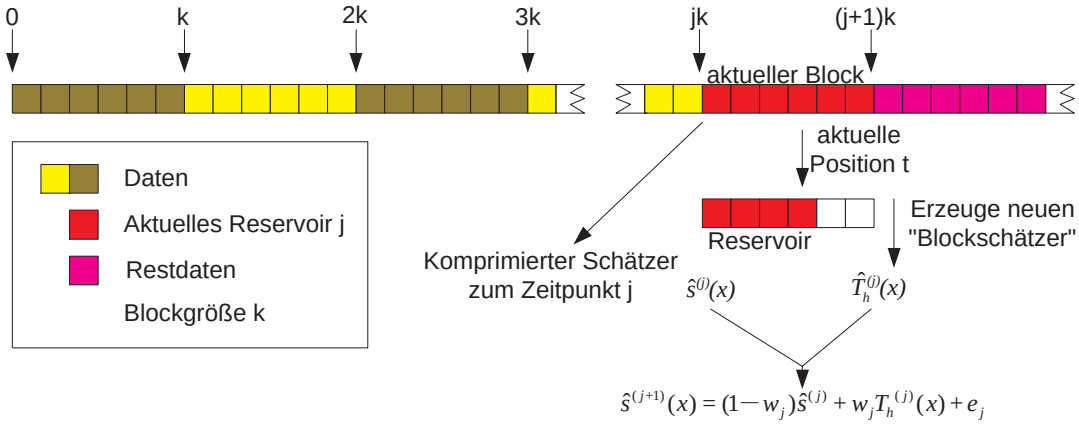

Abbildung 5.6: Adaptive größenbeschränkte Statistik zum Zeitpunkt t über einem Datenstrom

Verfolgt man den Ansatz, die Datengrundlage der Statistik zu komprimieren und legt dabei das Hauptaugenmerk auf kernbasierte Verfahren der nichtparametrischen Statistik, so ist ein intuitiver Ansatz sicherlich das „Verschmelzen“ zweier unabhängiger Stichproben.

Seien $X^{(1)}$ und $X^{(2)}$ unabhängige *i.i.d.* Stichproben der Größe n der gleichen Grundgesamtheit und $\hat{T}_n^{(1)}(x; X^{(1)})$ bzw. $\hat{T}_n^{(2)}(x; X^{(2)})$ zwei kernbasierte Statistiken über jeweils einer der beiden Stichproben. Das Prinzip der lokalen Gewichtung, d. h. des Verteilens der Masse eines Stichprobenwertes X_i in seine Umgebung, macht es möglich, zwei Stichprobenwerte mit ähnlicher Verteilung ihrer Masse mit einem geringen Fehler zu verschmelzen. Eine Kernfunktion wird dann nicht mehr über einem Stichprobenwert X_i plaziert, sondern über einem Repräsentanten mehrerer verschmolzener Stichprobenwerte. Damit gilt für den „neuen“ Schätzer

$$\hat{T}_n^{(1,2)}(x; X^{(1,2)}) = \omega_1 \hat{T}_n^{(1)}(x; X^{(1)}) + \omega_2 \hat{T}_n^{(2)}(x; X^{(2)}) + \epsilon(X^{(1)}, X^{(2)}, X^{(1,2)}),$$

mit geeigneten Gewichten ω_1 und ω_2 und einem Fehler ϵ, der direkt von den beiden Stichproben und deren Verschmelzung zur Repräsentantenmenge $X^{(1,2)}$ abhängt. In einem adaptiven kernbasierten Verfahren[17] kann über eine Anpassung der Bandbreite in einem solchen Bereich der entstehende Fehler weiter verringert werden. Adpative Verfahren benötigen jedoch in der Regel mindestens den doppelten Speicherplatzbedarf wie klassische Verfahren.

In dieser Arbeit wird der Ansatz verfolgt, den Schätzer selbst zu komprimieren. Dazu wird die Schätzfunktion $T(x; \tilde{X})$ der Statistik $T(X)$ mit einer Realisation $\tilde{X}$ der Stichprobe X mit Hilfe einer Splineinterpolation approximiert. Im Folgenden wird der Einfachheit halber nicht mehr zwischen der Statistik $T(X)$, die eine Zufallsvariable darstellt, und der Schätzfunktion $T(x; \tilde{X})$ auf Basis einer Realisation der Stichprobe unterschieden. Es wird implizit angenommen, dass die Schätzfunktion vorliegt.

[17] Adaptive Kernschätzer verwenden eine Bandbreite $h_{(i,x)}$, die entweder vom Stichprobenwert X_i oder Auswertungspunkt $x \in \mathbb{R}$ abhängt.

Abbildung 5.6 zeigt die generelle Vorgehensweise, wie mit Hilfe einer Komprimierungstechnik ein bezüglich des Speicherplatzbedarfs beschränkter adaptiver Schätzer realisiert werden kann. Dazu wird der Datenstrom logisch in Blöcke der Größe k aufgebrochen. Jeder Block $j+1$ wird isoliert und unabhängig von den bisher verarbeiteten Blöcken $i = 0, \ldots, j$ zum Aufbau eines Schätzers T_j verwendet, der dann geeignet gewichtet[18] mit dem aktuell gültigen komprimierten Schätzer $\hat{s}_j$ kombiniert wird. Dabei wird zwischen den Zeitpunkten jk und $(j+1)k$ das aktuelle Reservoir gefüllt. Die Größe des Reservoirs hängt dabei vom temporär zur Verfügung stehenden Speicher ab und kann zur Laufzeit angepasst werden. Werden verschiedene Blockgrößen verwendet, muss unter Umständen das Gewicht des Blockschätzers angepasst werden. Im Folgenden werden jedoch zur Vereinfachung lediglich gleich große Blöcke angenommen.

5.3.1 Splinekomprimierte adaptive Schätzer

Werden zur Komprimierung kubische Splines verwendet und sind die Blockschätzer kernbasierte Statistiken, kommen im wesentlichen zwei Möglichkeiten, die n Stützstellen zu wählen, in Frage. Zum einen kann ein äquidistantes Gitter verwendet werden. In diesem Fall muss eine Splineinterpolationsfunktion in Bézier-Darstellung $3n+3$ Werte speichern[19]. Werden hingegen die Stützstellen so gewählt, dass sie auf einen Stichprobenwert X_i fallen[20], also zwischen den Gitterpunkten keine gleichen Abstände bestehen, benötigt die Bézier-Darstellung $4n+2$ Werte. Das letztere Vorgehen hat zusätzlich den Nachteil, dass entweder in jedem Schritt neue Stützstellen gewählt werden müssen oder dass das erste Reservoir die Lage der Stützstellen bestimmt. Ist der erste Block bereits repräsentativ für den Datenstrom[21], ist diese Wahl sicherlich leicht zu motivieren. Ist der Datenstrom hingegen sortiert, muss diese Herangehensweise zu erheblichen Fehlern führen.

Steht also ein Speicherplatz der Größe M zur Verfügung, können etwa $\frac{1}{3}M$ bzw. $\frac{1}{4}M$ Stützstellen verwendet werden. Im Gegensatz zu Korn et al. , die in [KJJ99] von demjenigen Speicherplatzbedarf ausgehen, der benötigt wird eine kubische Splineinterpolationsfunktion aufzubauen[22], wird hier von dem Speicher ausgegangen, der zur Auswertung der Funktion in einem Punkt $x \in \mathbb{R}$ tatsächlich benötigt wird. Dies wird durch die Annahme motiviert, dass die Interpolationsfunktion mehrmals hintereinander an verschiedenen Stellen x ausgewertet wird. Damit darf nicht vom minimalen Speicherplatzbedarf ausgegangen werden[23], obwohl die Berechnung der Bézier-Koeffizienten lediglich eine Komplexität von $\Theta(n)$ hat und damit auch zum Auswertungszeitpunkt denkbar wäre. Es muss der Speicherbedarf in der Auswertungsphase zugrunde gelegt

[18] vgl. Abschnitt 5.1

[19] vgl. Abschnitt 5.2.2

[20] Enthält das Reservoir zuviele Werte, um alle als Stützstelle zu verwenden, kann z. B. jeder zweite oder dritte Wert verwendet werden.

[21] D. h. der erste Block kann als *i.i.d.* Stichprobe angesehen werden.

[22] Für n Stützstellen bedeutet dies einen Speicherplatzbedarf von $n+2$ im Falle äquidistanter Stützstellen und $2n+1$ im allgemeinen Fall.

[23] D. h. von den Gitterpunkten und zugehörigen Funktionswerten über dem Gitter

werden.

Für eine kernbasierte Statistik f gilt damit für den Fehler der Splineinterpolation gemäß Formel (5.12) auf einem äquidistanten Gitter über $[a, b]$

$$\begin{aligned} ||\hat{s}_f - f||_{[a,b]} &\leq \frac{3}{8}\left(3\frac{b-a}{M}\right)^4 ||f^{(4)}||_{[a,b]} \\ &= \frac{243}{8}\left(\frac{b-a}{M}\right)^4 ||f^{(4)}||_{[a,b]}. \end{aligned} \tag{5.21}$$

Für einen realisierten Schätzer ist der obige Ausdruck bis auf die Speichergröße M konstant. Damit gilt für den Fehler der Splineinterpolation eines kernbasierten Verfahrens in Abhängigkeit des verwendeten Speichers $O(\frac{1}{M^4})$.

Ist f ein Schätzer für die Dichte gemäß Formel (4.4), kann Formel (5.21) weiter vereinfacht werden zu

$$\begin{aligned} ||\hat{s}_f - f||_{[a,b]} &= \frac{243}{8}\left(\frac{b-a}{M}\right)^4 \left|\left|\frac{1}{nh^4}\sum_{i=1}^{n} K^{(4)}\left(\frac{x-X_i}{h}\right)\right|\right|_{[a,b]} \\ &= \frac{243}{8}\left(\frac{b-a}{M}\right)^4 \frac{1}{h^4}\left|\left|K^{(4)}\left(\frac{x-X_i}{h}\right)\right|\right|_{[a,b]}. \end{aligned} \tag{5.22}$$

Wird zusätzlich der Gauss-Kern verwendet mit

$$K^{(4)}(x) = \frac{1}{\sqrt{2\pi}}(x^4 - 6x^2 + 3)e^{-\frac{x^2}{2}}$$

und

$$||K^{(4)}(x)||_\infty = ||K^{(4)}(0)||_\infty \approx 7.52 \leq 8,$$

nimmt der Fehler die Form

$$||\hat{s}_f - f||_{[a,b]} \leq \frac{243}{h^4}\left(\frac{b-a}{M}\right)^4$$

an. Damit ist der maximale Fehler unabhängig von der verwendeten Realisation der Stichprobe und hängt lediglich von der verwendeten Bandbreite h und der Speichergröße M ab.

Wird ein allgemeiner adaptiver Schätzer der Form (5.3) in jedem Schritt mit Hilfe der Splineinterpolation komprimiert, so gilt mit der Folge

$$\begin{aligned} \hat{s}_1(x) &:= \hat{f}_1(x) + \epsilon_1 \\ \hat{s}_2(x) &:= (1-\omega_2)\hat{s}_1(x) + \omega_2\hat{f}_2(x) + \epsilon_2 \\ \hat{s}_3(x) &:= (1-\omega_3)\hat{s}_2(x) + \omega_3\hat{f}_3(x) + \epsilon_3 \\ &\vdots \quad\quad \vdots \\ \hat{s}_j(x) &:= (1-\omega_j)\hat{s}_{j-1}(x) + \omega_j\hat{f}_j(x) + \epsilon_j \end{aligned}$$

wegen

$$\begin{aligned}\hat{s}_j(x) &= (1-\omega_j)\hat{s}_{j-1}(x) + \omega_j \hat{f}_j(x) + \epsilon_j \\ &= (1-\omega_j)((1-\omega_{j-1})\hat{s}_{j-2}(x) + \omega_{j-1}\hat{f}_{j-1} + \epsilon_{j-1}) + \omega_j \hat{f}_j(x) + \epsilon_j \\ &= (1-\omega_j)(\ldots((1-\omega_2)(\hat{f}_1+\epsilon_1) + \omega_2\hat{f}_2 + \epsilon_2)\ldots) + \epsilon_j\end{aligned}$$

für den summierten Fehler $\epsilon^{(j)}$ zum Zeitpunkt $j \geq 1$ des j-ten komprimierten Schätzers $\hat{s}_j$, dass

$$\epsilon^{(j)} = \sum_{i=1}^{j} \epsilon_i \prod_{l=i+1}^{j} (1-\omega_l).$$

Mit Formel (5.21) und

$$\epsilon := c_\epsilon \left(\frac{b-a}{M}\right)^4$$

gilt nun für eine geometrische Gewichtung mit Parameter c gemäß Formel (5.6)

$$\epsilon^{(j)} \leq \sum_{i=1}^{j} \epsilon \prod_{l=i+1}^{j} \left(1-\frac{1}{c}\right) = c\epsilon\left(1-\left(\frac{c-1}{c}\right)^j\right) \leq c\epsilon.$$

Im Falle einer geometrischen Gewichtung ist damit der maximale zusätzliche Fehler, der durch die Komprimierung verursacht wird, beschränkt und unabhängig vom Zeitpunkt j des adaptiven Schätzers. Für eine arithmetische Gewichtung gemäß Formel (5.4) gilt

$$\begin{aligned}\epsilon^{(j)} &\leq \sum_{i=1}^{j} \epsilon \prod_{l=i+1}^{j} \left(1-\frac{1}{l}\right) \\ &= \epsilon \sum_{i=1}^{j} \prod_{l=i+1}^{j} \left(\frac{l-1}{l}\right) \\ &= \epsilon \sum_{i=1}^{j} \frac{i}{j} \\ &= \frac{1}{j}\epsilon(1+2+3+\ldots+j) = \frac{j-1}{2}\epsilon \\ &< \frac{1}{2}j\epsilon.\end{aligned}$$

Damit hängt der maximale Fehler bei arithmetischer Gewichtung linear von der Anzahl der Komprimierungsschritte ab. Für die weiteren in Abschnitt 5.1 genannten Folgen von Gewichten gelten ähnliche Aussagen.

5.3.2 Adaptive Schätzer in XXL

Das allgemeine Prinzip eines adaptiven Schätzers steht in XXL transparent für allgemeine Funktionen des Typs `xxl.functions.Function`, die Objek-

te des Typs `java.lang.Number`[24] konsumieren, als auch für Objekte des Typs `xxl.statistics.RealFunction` zur Verfügung. Alle Klassen, die zur Verwendung eines adaptiven Schätzers benötigt werden, finden sich im Paket `xxl.statistics`. Die Klasse `AdaptiveWeightFunctions` stellt verschiedene Strategien[25] für die Wahl der Gewichte in Abhängigkeit des aktuellen Verarbeitungsschritts zur Verfügung. Die Klasse `AdaptiveAggregationFunction` implementiert das allgemeine adaptive Prinzip.

Zum Verständnis der Funktionsweise sei hier noch einmal kurz auf das Prinzip der Aggregationsfunktionen und des Aggregators eingegangen. Eine Aggregationsfunktion ist eine binäre Funktion, dessen erstes Argument der Aggregationswert des vorangegangenen Schrittes und dessen zweites Argument das nächste Datum im Eingabestrom ist. Der Aggregator leistet transparent diesen Aufruf und gibt seinerseits den Rückgabewert eines solchen Aufrufs nach außen[26]. Damit eine adaptive Verarbeitung zwischengeschaltet werden kann, muss die verwendete Aggregationsfunktion zwei Voraussetzungen erfüllen. Zum einen müssen die von der Funktion gelieferten Objekte vom Typ `Function` oder `RealFunction` sein. Zum anderen muss jeweils nach Verarbeitung von k Objekten des Eingabestroms eine **neue** Funktion erzeugt und zurückgegeben werden.

Die Klasse `AdaptiveAggregationFunction` selbst ist eine binäre Funktion, die dem zugrunde liegendem Prinzip einer Aggregatsfunktion gehorcht. Damit reicht eine Instanz der Klasse alle Objekte des Eingabestroms an die interne Aggregationsfunktion weiter und verarbeitet die zurückgegebenen „Funktions"-Objekte. Ein Aufruf der Form

```
Aggregator agg = new Aggregator(
    input,
    new AdaptiveAggregationFunction(
        new MyFunction(),
        new AdaptiveWeightFunctions.GeometricWeights( 2.0),
        false
    )
);
```

erzeugt einen Cursor, der den Eingabedatenstrom `input` mit Hilfe einer Aggregationsfunktion vom Typ `MyFunction` verarbeitet und intern die Aggregationsresultate mit einer geometrischen Gewichtung neu kombiniert. Der Wert vom Typ `boolean` zeigt an, dass die interne Aggregationsfunktion Objekte vom Typ `Function` zurückliefert. Andernfalls werden Objekte vom Typ `RealFunction` erwartet.

Die Klasse `KernelBasedBlockEstimatorAggregationFunction` im Paket `xxl.statistics.kernels` stellt einen Rahmen für allgemeine Aggregationsfunktionen zur Verfügung, die blockweise kernbasierte Schätzer als Aggregationswerte liefern. Dazu muss zum einen der Eingabedatenstrom zur Laufzeit vorverarbeitet und zum anderen eine *Factory* zur Erzeugung der kernbasierten Statistiken übergeben

[24] Alle in Java verfügbaren Objekttypen derjenigen Basistypen, die Zahlen repräsentieren, erben direkt von `Number`. Damit besitzen alle Objekte, die Zahlen repräsentieren, die Methode `double doubleValue()` und können somit auf einen gemeinsamen Basistyp zurückgeführt werden.

[25] Alle in Abschnitt 5.1 entwickelten Gewichte sind implementiert.

[26] Eine ausführliche Erläuterung des Aggregators und der zugehörigen Aggregationsfunktionen findet sich in Abschnitt 2.5.2.

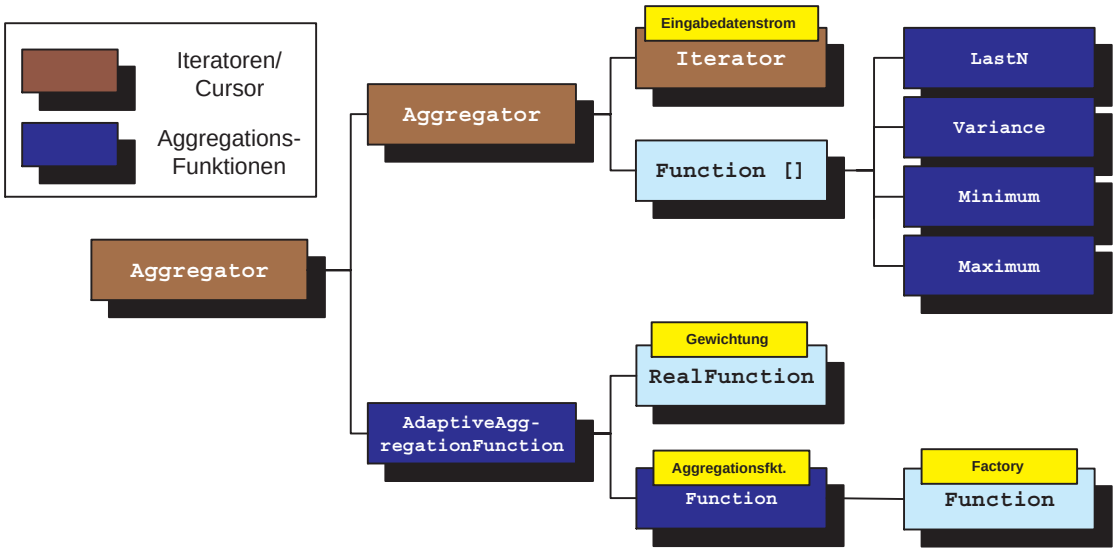

Abbildung 5.7: Hierarchie der benötigten Instanzen einer adaptiven Aggregation über kernbasierten Verfahren

werden. Die Vorverarbeitung des Datenstroms stellt logisch das Aufbrechen der Daten in Blöcke einer vorgegebenen Größe und das Sammeln weiterer für die meisten Schätzer benötigter Informationen dar. Die Methode `Cursor inputCursor( Iterator, int)` verpackt den Datenstrom in einen Aggregator, der die Vorverarbeitung transparent ausführt. Viele Klassen, die Schätzer implementieren, stellen bereits *Factory*-Methoden zur Verfügung. Um zum Beispiel eine blockweise Dichteschätzung mit Randbehandlung auf einem Datenstrom `input` mit einer arithmetischen Gewichtung durchzuführen, reicht ein Aufruf der Form:

```
Aggregator agg = new Aggregator(
    KernelBasedBlockEstimatorAggregationFunction.inputCursor(
        input,
        k
    ),
    new AdaptiveAggregationFunction(
        new KernelBasedBlockEstimatorAggregationFunction(
            ReflectionKernelDensityEstimator.FACTORY,
            new EpanechnikowKernel,
            KernelBandwidths.THUMB_RULE_1D
        ),
        new AdaptiveWeightFunctions.ArithmeticWeights()
    )
);
```

Damit wird nach jeweils `k` Objekten im Eingabedatenstrom `input` ein neuer Dichteschätzer erzeugt, der mit Hilfe des Epanechnikow-Kerns und der *normal-scale-rule* über dem Minimum bzw. Maximum der bisher verarbeiteten Daten aufgebaut wird.

Dieser Schätzer wird mit dem vorangegangenen Schätzer mittels einer arithmetischen Gewichtung kombiniert.

Damit ist der Aggregationswert zum Zeitpunkt t nach der Verarbeitung von j Blöcken gemäß Formel (5.3) definiert. Der dazu benötigte Speicherplatz ist linear in der Anzahl der bisher verarbeiteten Daten, da der Dichteschätzer über allen Datenpunkten eines Blocks aufgebaut wird und keinerlei Kompression stattfindet.

Da der Aufruf komplex erscheint, zeigt Abbildung 5.7 die Hierarchie der beteiligten Instanzen. Der äußere Aggregator arbeitet auf dem Cursor, der die Daten des Eingabestroms mit Hilfe der n-stelligen Aggregation vorverarbeitet und blockweise zusammenfasst. Diese Objekte werden dann vom äußeren Aggregator auf die adaptive Aggregationsfunktion angewendet. Diese leitet die Daten unverändert an die eigentliche Aggregationsfunktion weiter, die pro Block mit Hilfe der *Factory*[27] einen neuen Schätzer erzeugt und diesen an die adaptive Funktion zurückgibt. Unterscheidet sich diese Funktion von der vorhergehenden, d. h. hat die interne Aggregationsfunktion einen neuen Schätzer etabliert, wird diese neue Funktion mit ihrem Vorgänger mit Hilfe der Gewichtsfunktion kombiniert und nach außen gegeben. Die braunen Objekte symbolisieren dabei die benötigten Iteratoren und die blauen die verwendeten binären Aggregationsfunktionen. Die hellen Objekte stehen für die weiteren benötigten Funktionen. Die mit gelb markierten Objekte müssen vom Anwender vorgegeben werden und bestimmen im wesentlichen die Funktionalität der Aggregation.

5.3.3 Komprimierte adaptive Schätzer in XXL

Wie eine beliebige glatte Funktion mit kubischen Splinen komprimiert werden kann, ist in Abschnitt 5.2.3 bereits gezeigt worden. Damit eine online Komprimierung eines Schätzers über einem Datenstrom zur Laufzeit und automatisch möglich ist, muss die Komprimierung ebenfalls mittels einer Aggregationsfunktion realisiert sein. Die Klasse `SplineCompressedFunctionAggregateFunction` aus dem Paket `xxl.statistics` stellt die Komprimierung beliebiger glatter reellwertiger Funktionen zur Verfügung. Da Objekte dieses Typs selbst wieder Aggregationsfunktionen sind, können sie einfach zwischen adaptiver Aggregationsfunktion und Aggregator geschaltet werden. Um die adaptiven Schätzer des Beispiels in Abschnitt 5.3.2 bezüglich eines Gitters `grid` zu komprimieren, genügt ein Aufruf der Form:

```
Aggregator agg = new Aggregator(
    KernelBasedBlockEstimatorAggregationFunction.inputCursor(
        input,
        k
    ),
    new SplineCompressedFunctionAggregateFunction(
        new AdaptiveAggregationFunction(
            new KernelBasedBlockEstimatorAggregationFunction(
                ReflectionKernelDensityEstimator.FACTORY,
```

[27] Für *Factories* verschiedener kernbasierter Verfahren siehe Abschnitt 4.3.6.

```
                new EpanechnikowKernel,
                KernelBandwidths.THUMB_RULE_1D
            ),
            new AdaptiveWeightFunctions.ArithmeticWeights()
        ),
        new MyPredicate(),
        grid
    )
);
```

Dabei reicht das Komprimierungsverfahren alle Objekte des Eingabedatenstroms unverändert an die innere Aggregationsfunktion durch. Lediglich die zurückgegebenen Funktionen werden mit Hilfe des übergebenen Prädikats überprüft und gegebenenfalls komprimiert. Durch dieses Prinzip ist es leicht möglich, die Komprimierung zu modifizieren oder durch ein anderes Verfahren zu ersetzen. Auch ein Austausch der intern genutzten Blockschätzer auf Basis von Kernfunktionen ist leicht möglich. Um zum Beispiel einen adaptiven nativen Kernselektivitätsschätzer oder einen adaptiven *Difference Kernel Estimator* zu erzeugen, muss im obigen Aufruf lediglich die *Factory* ausgetauscht werden.

Sollen eigene Verfahren mit Hilfe des adaptiven Prinzips realisiert werden, muss die interne Aggregationsfunktion ausgetauscht werden. Sie ist für die Erzeugung neuer Blockschätzer zuständig, die dann durch die Instanz des Typs `AdaptiveAggregationFunction` gemäß der Gewichte kombiniert und mittels der Splines komprimiert werden.

Einen einfachen Blockschätzer auf Basis der empirischen Stichprobenfunktion kann mittels folgender Aggregationsfunktion einfach realisiert werden.

```
new Function(){
    int c = 0;
    public Object invoke( Object old, Object next){
        c++;
        if ( next == null) return null;
        if( c % blocksize == 0)
            return EmpiricalCDF.FACTORY.invoke( ((Object[]) next)[0]);
        else return old;
    }
}
```

Benötigt diese interne Aggregationsfunktion zusätzliche Informationen über die Elemente des Datenstroms, wie zum Beispiel die letzten n der bisher verarbeiteten Daten, so müssen diese durch ein *wrappen* des Datenstroms nach innen „durchgereicht“ werden. Für einen Blockschätzer, der neben den letzten n Elementen zusätzlich das Mimimum und das Maximum der bisher verarbeiteten Daten zur Erzeugung eines neuen Schätzers benötigt, kann dies mit folgendem Aufruf geschehen:

```
Iterator wrappedInput= new Aggregator( input,
    new Function[]{
```

```
        new LastN( blocksize),
        new Minimum(),
        new Maximum()
    }
);
```

5.4 Evaluation der Verfahren

Alle in dieser Arbeit vorgestellten Verfahren bieten die Möglichkeit, zu einem festen vorgegebenen Speicherplatz auf einem Datenstrom genutzt zu werden. Somit ist ein wichtiges Kriterium für die Qualität eines Verfahrens der Fehler zu einem gegebenen festen und beschränkten Speicherplatz. Dabei kommt ein einfaches Speichermodell zum Einsatz, dass für jedes zu speichernde Datum genau eine Speichereinheit zählt. Ein Kerndichteschätzer gemäß Formel (4.27) über einer Stichprobe der Größe $n = 2000$ benötigt somit einen Speicher von $M = 2000$ Einheiten. Wird ein solcher Schätzer mit Hilfe eines Splines über einem äquidistantem Gitter mit $k = 99$ Stützstellen komprimiert, so benötigt der Spline einen Speicher von $3k + 2 \approx 300$ Einheiten. Im Folgenden wird zur Vereinfachung und soweit der Sprachgebrauch eindeutig ist, nur noch von benötigtem Speicher bzw. Speicherplatz gesprochen.

Bedingt durch die Vielzahl unterschiedlicher Verfahren und deren Parameter ist eine vollständige Evaluation aller Verfahren mit allen möglichen Kombinationen an Verfahrens- und Testparametern kaum möglich. Aus diesem Grund wird im folgenden Abschnitt 5.5 der Einfluss verschiedener wichtiger Verfahrens- und Testparameter anhand einiger Beispiele untersucht und diskutiert. Dies geschieht vor allem mit Hilfe der Dichteschätzer und erzeugten Datensätze.

In Abschnitt 5.6 werden die angewandten Schätzer zur Selektivitätsschätzung mit Hilfe verschiedener Sätze von Testanfragen untersucht und mit einem Referenzverfahren der Literatur verglichen.

5.4.1 Testkriterien und Modellannahmen

Alle Auswertungen in den folgenden Abschnitten setzen folgende Modellparameter voraus:

1. Alle Daten werden als Datenstrom verarbeitet, wobei die Position eines Elements im Datenstrom als diskreter Zeitpunkt t verstanden wird (vgl. Abschnitt 2.3).

2. Auf den Grad der Sortiertheit bzw. die Reihenfolge der Elemente in einem Strom wird unter Umständen explizit eingegangen.

3. Benötigte a priori Information wie die Größe der Grundgesamtheit oder Extremwerte werden explizit genannt, d. h. in der Regel laufen alle Verfahren vollständig unüberwacht ab.

4. Alle Verfahren verwenden einen vorgegebenen festen Speicherplatz M und unter Umständen einen zusätzlichen temporär genutzten Speicherplatz M_t gemäß des oben erläuterten Speichermodells.

Alle Tests können einfach mittels spezieller *Factory*-Methoden der Klasse `Aggregators` im *Package* `xxl.statistics` nachvollzogen werden. Steht für einen Test keine spezielle Methode zur Verfügung, wird der allgemeine Aufruf zur Konstruktion des vewendeten Schätzers angegeben.

5.4.2 Fehler und Fehlermaße

Die Bestimmung eines quantitativen Fehlers eines Dichteschätzers ist oft schwierig. Dies kann zum einen daran liegen, dass die tatsächliche Dichte nicht bekannt ist oder in analytisch geschlossenener Form zur Verfügung steht. Zum anderen existiert im Falle eines nichtparametrischen Verfahrens in der Regel keine analytisch geschlossene Form des Schätzers. In einem solchen Fall kann der Fehler eines Dichteschätzers nur in einem Punkt $x \in \mathbb{R}$, nicht jedoch für die gesamte Kurve bestimmt werden.

Wird die Verteilungsfunktion einer Grundgesamtheit geschätzt, ist die Bestimmung eines Fehlers einfacher. Hier kann mit Hilfe der empirischen Verteilungsfunktion der Grundgesamtheit eine Referenzfunktion angegeben und der Fehler zu dieser Funktion bestimmt werden. Wird der Schätzer einer Verteilungsfunktion auf Basis des in Abschnitt 4.3.3 eingeführten Modells zur Selektivitätsschätzung genutzt, kann mit Hilfe der empirischen Verteilungsfunktion die tatsächliche Instanzselektivität[28] einer Anfrage und der Fehler des Schätzers bezüglich der Instanzselektivität dieser Anfrage bestimmt werden.

Die Dichte einer Verteilung hat jedoch im Gegensatz zur Verteilungsfunktion den Vorteil, spezifische Eigenschaften und Charakteristika der Verteilung besser darzustellen. Daher werden in dieser Arbeit ausschließlich Dichteschätzer zur Erklärung und Beschreibung verschiedener Effekte verwendet. Zur Evaluation der Selektivitätsschätzer wird der mittlere Fehler bezüglich einer Menge von 1000 Testanfragen bestimmt. Die Eigenschaften dieser Anfragen werden in Abschnitt 5.4.4 erläutert.

Somit ist der **mittlere absolute Fehler** eines Selektivitätsschätzers $\hat{\sigma}$ zu einer Menge von 1000 Anfragen $\mathrm{Q}_q^{\mathrm{GG}} = \{q_1, q_2, \ldots, q_{1000}\}$ mit den wahren relativen Selektivitäten $\{\sigma_1, \ldots, \sigma_{1000}\}$ definiert durch

$$\mathrm{MAF}(\mathrm{Q}_q^{\mathrm{GG}}) = \frac{1}{1000} \sum_{i=1}^{1000} |\sigma_i - \hat{\sigma}(q_i)| \,.$$

Entsprechend des absoluten Fehlers ist der **mittlere relative Fehler** definiert durch

$$\mathrm{MRF}(\mathrm{Q}_q^{\mathrm{GG}}) = \frac{1}{1000} \sum_{i=1}^{1000} \left| \frac{\sigma_i - \hat{\sigma}(q_i)}{\sigma_i} \right| .$$

[28] vgl. [BKS99]

5.4.3 Testdaten

Zur Evaluation der verschiedenen Verfahren werden Datensätze mit verschiedenen Charakteristika verwendet. Zum einen dienen erzeugte Grundgesamtheiten mit wohlbekannten Dichte- und Verteilungsfunktionen zur Untersuchung des Einflusses verschiedener Verfahrensparameter. Des Weiteren kommt ein realer Testdatensatz zum Einsatz, um die Leistungsfähigkeit der angewandten Schätzer zur Selektivitätsschätzung in einem praxisnahen Versuch zu ermitteln.

Alle wichtigen Eigenschaften der in dieser Arbeit genutzen Daten finden sich in Anhang A. In den folgenden Abschnitten wird erläutert, wie künstliche Daten zu vorgegebenen Verteilungen im Allgemeinen und speziell in XXL erzeugt werden können. Zusätzlich werden Maße für den Grad der Sortiertheit eines Datensatzes angegeben. Wie ein Datensatz modifiziert werden kann, so dass er einer vorgegebenen Sortierung gehorcht, wird ebenfalls im Anhang in Abschnitt A.4 erläutert.

5.4.3.1 Erzeugung von Daten beliebiger Verteilung

Erzeugte Daten, die einer vorgegebenen oft glatten Verteilung folgen, zeigen besonders gut die Leistungsfähigkeit von Schätzern, die auf dem Prinzip der lokalen Glättung beruhen[29]. So genügt oft bereits eine kleine Stichprobe, um mit Hilfe eines klassischen Kerndichteschätzers die Dichte einer standardnormalverteilten Zufallsvariablen X sehr gut zu approximieren. Daten der realen Welt sind hingegen oft nicht glatt bzw. besitzen Sprünge und bieten damit eine bessere Möglichkeit, glättende Schätzer auf ihre Leistungsfähigkeit hin zu testen.

Um jedoch die Leistungsfähigkeit hybrider Schätzer[30] und damit auch die Qualität von Schätzern zur Erkennung von Sprungstellen testen zu können, eignen sich solche realen Daten nicht, da in der Regel die Verteilung bzw. die Dichte solcher Daten weder bekannt ist noch in analytisch geschlossener Form angegeben werden kann. Damit werden nun Testdaten benötigt, die einer Verteilung folgen, deren Dichte bekannt ist und die Sprungstellen besitzen, deren Lage und Höhe zur Verfügung stehen.

Zur Erzeugung solcher Daten muß zunächst die Dichte f in geschlossener Form bekannt sein. Damit ist es nun mit Hilfe gleichverteilter Zufallszahlen möglich, Daten zu generieren, die gemäß $F(y) = \int_{-\infty}^{y} f(x)\mathrm{dx}$ verteilt sind. Existiert die Inverse $F^{-1}(y)$ zur Verteilungsfunktion F, so kann man sich die Beziehung der Gleichverteilung zu beliebigen anderen Verteilungen zunutze machen, denn mit Hilfe der Eigenschaft[31]

$$F^{-1}(U) \sim F \tag{5.23}$$

für eine stetige Verteilung F und einer gleichverteilten Zufallsvariablen U ist es leicht möglich, gemäß F verteilte Daten zu erzeugen. Dieses Verfahren wird Inversionsprinzip

[29] vgl. Abschnitt 4.3.1

[30] vgl. Abschnitte 4.3.2 und 4.3.4

[31] Allgemein muss sogar nur die Inverse der stetigen Verteilung F bekannt sein, jedoch keine Dichte existieren.

(*inversion method*) genannt[32].

Leider jedoch existiert oft keine analytisch geschlossene Form der Inversen F^{-1} der Verteilungsfunktion. Damit ist nur eine numerische Lösung möglich, deren Berechnung jedoch in der Regel sehr zeitintensiv ist und die direkt von der benötigten Genauigkeit abhängt. Hier schafft das Prinzip der Ablehnung/Annahme (*rejection/acception method*, oft auch nur *rejection method*) Abhilfe[33]. Möchte man Zufallszahlen erzeugen, die gemäß F verteilt sind und deren Dichte f ist, so braucht man zunächst eine weitere Dichte $g(x)$ und eine Konstante c mit

$$f(x) \leq cg(x) \text{ für alle x.} \tag{5.24}$$

Dann können mit folgendem Algorithmus Zufallszahlen erzeugt werden, die gemäß F verteilt sind:

Algorithmus 5.11 *Erzeuge* `randU` *gleichverteilte Zufallszahlen und* `randG` *Zufallszahlen verteilt gemäß G. Gegeben seien $g(x)$ und c gemäß Formel (5.24) und f die Dichtefunktion der Verteilung F. Dann erzeugt der folgende Algorithmus eine Zufallszahl, die gemäß F verteilt ist.*

```
1    DO
2        LET x = randG.getNext()
3        LET r = c * g(x) / f(x)
4        LET u = randU.getNext()
5    WHILE u*r >= 1
6    RETURN x
```

Wählt man die benötigte Vergleichsdichte $g(x)$ so, dass die Inverse $G^{-1}(y)$ der Verteilungsfunktion G existiert und effizient auszuwerten ist, können mit einer Kombination beider Verfahren Daten beliebiger Verteilungen erzeugt werden.

Bemerkung 5.12 *Für die Anzahl der Iterationen N in Algorithmus 5.11 gilt*

$$E(N) = c. \tag{5.25}$$

Daraus folgt für die Effizienz des Verfahrens, dass die Konstante c möglichst klein zu wählen ist.

In XXL stehen beiden Verfahren zur Erzeugung beliebig verteilter Zufallszahlen im *Package* `xxl.util` zur Verfügung. Die Klassen `InversionDistributionBasedPRNG` und `RejectionDistributionBasedPRNG` implementieren das Inversionsprinzip und Ablehnungsprinzip. Eine Folge von stetigen Zufallszahlen liefert die Klasse `xxl.cursors.ContinuousRandomNumber` mit Hilfe eines Zufallszahlengenerators vom Typ `xxl.util.ContinuousRandomWrapper`[34]. Ein Aufruf der Form

[32] vgl. [Dev86] Seite 27 ff.

[33] vgl. [Dev86] Seite 40 ff.

[34] Klassen, die diese Schnittstelle implementieren, bieten die Möglichkeit Zufallszahlengeneratoren beliebiger Java-Bibliotheken einfach in XXL zu integrieren. Zwei einfache Beispiele zeigen die Klassen `JavaContinuousRandomWrapper` und `connectivity.colt.ColtContinuousRandomWrapper` für den in Java integrierten Zufallszahlengenerator und Objekte von Typ `RandomElement` der Java-Bibliothek

```
Cursor randomNumbers = new ContinuousRandomNumber( myPRNG);
```

liefert einen unendlichen Datenstrom gemäß `myPRNG` verteilter Zufallszahlen. Um einen Zufallszahlengenerator auf Basis des Inversionsprinzips zu erzeugen, ist lediglich die Umkehrfunktion der Verteilungsfunktion der gewünschten Verteilung in Form einer `RealFunction` und ein Zufallszahlengenerator für gleichverteilte Zufallszahlen zu übergeben.

```
ContinuousRandomWrapper crwInversion = new InversionDistributionBasedPRNG(
    myInverse,                          // Inverse der Verteilung G
    new JavaContinuousRandomWrapper()   // gleichverteilte Zufallszahlen
);
```

Steht keine analytisch geschlossene Form der Inversen zur Verfügung, kann die Ablehnungsmethode verwendet werden. Dazu ist jedoch eine Konstante `c` und eine bezüglich der Konstanten „dominierende“ Dichte `g` in Form einer `RealFunction` nötig, deren Umkehrfunktion der Verteilungsfunktion bekannt ist. Dies ist notwendig, so dass mit Hilfe der Inversionsmethode gemäß G verteilte Zufallszahlen erzeugt werden können. Auch hier wird zusätzlich ein Zufallszahlengenerator für gleichverteilte Zufallszahlen gebraucht. Auf Aufruf der Form

```
ContinuousRandomWrapper crwRejection = new RejectionDistributionBasedPRNG(
    new JavaContinuousRandomWrapper(),  // gleichverteilte Zufallszahlen
    f,                                  // Dichte der Verteilung
    g,                                  // Vergleichsdichte
    c,                                  // Konstante
    crwInversion                        // gemäß G verteilte Zufallszahlen
);
```

erzeugt somit eine Instanz des gewünschten Zufallszahlengenerators zur vorgegebenen Dichte `f`.

5.4.3.2 Teilsortierte Daten

Zur Messung des Grades der Sortierung eines Datensatzes existieren verschiedene Ansätze und eine Vielzahl von Maßen[35]. Eine Einführung in das Gebiet des adaptiven Sortierens findet sich bei Knuth in [Knu98] ab Seite 15. Einen guten Überblick geben Estivill-Castro and Wood in [ECW92].

Ein wichtiges Maß für den Grad der Sortiertheit bzw. den Grad der Unsortiertheit eines Datensatzes G der Größe N ist die Anzahl der Inversionen im Datensatz. Eine Inversion ist ein Tupel (x_i, x_j) mit $1 \leq i < j \leq N$ und $x_i > x_j$, also

$$\mathrm{inv}(G) := |\{(x_i, x_j) : 1 \leq i < j \leq N \text{ und } x_i > x_j\}|.$$

Colt (vgl. [Col]). Diese Schnittstelle bietet darüberhinaus die Möglichkeit, eigene Verfahren zur Erzeugung von Zufallszahlen in XXL zu integrieren. Entsprechende Klassen existieren auch für diskrete Verteilungen.

[35] In [ECW92] finden sich elf verschiedene Maße mit unterschiedlichen Vor- und Nachteilen, den Grad der Ordnung in einem Datensatz zu bestimmen.

Damit gilt für einen Datensatz der Größe N

$$0 \leq \text{inv}(G) \leq \frac{N(N-1)}{2},$$

mit inv $= 0$ für eine sortierte und inv $= \frac{N(N-1)}{2}$ für eine invers sortierte Folge. Die Zahl der Inversionen gibt jedoch nur den Grad einer globalen Sortiertheit an, berücksichtigt jedoch keine lokalen Informationen. So hat die Folge $\{6, 7, 8, 9, 10, 1, 2, 3, 4, 5\}$ eine Inversionszahl von inv $= 25$, intuitiv jedoch ist die Folge „fast" sortiert. Ein Maß, welches im besonderen lokale Sortiertheit berücksichtigt, ist die Zahl der *Runs* in einem Datensatz

$$\text{runs}(G) := |\{x_i : 1 \leq i \leq N-1 \text{ und } x_i > x_{i+1}\}| + 1,$$

mit $1 \leq \text{runs}(G) \leq N$. Dieses Maß gibt damit die Zahl der sortierten Folgen in einem Datensatz an. Für das obige Beispiel ist runs $= 2$, da lediglich die 10 bzw. das Tupel $(10, 1)$ zu zählen ist.

Im Folgenden wird soweit nicht anders vermerkt, der Grad der Sortiertheit eines Datensatzes mittels der Zahl der normierten Inversionen angegeben, d. h.

$$\text{inv}_N(G) := 2\frac{\text{inv}(G)}{N(N-1)}. \tag{5.26}$$

Oft ist zusätzlich in Klammern die Zahl der (normierten) *Runs* angegeben, damit auch der lokale Grad der Sortiertheit abgeschätzt werden kann.

5.4.4 Fensteranfragen

Zur Evaluation der angewandten Schätzer zur Selektivitätsschätzung von Fensteranfragen sind aus jedem Testdatensatz 1000 Fensteranfragen zufällig erzeugt und deren wahre relativen Selektivitäten bestimmt worden. Um den Einfluss der Anfragengröße auf den Schätzfehler beurteilen zu können, kommen fünf unterschiedliche Fenstergrößen zum Einsatz. Die Größen der Intervalle sind relativ zu den minimalen und maximalen Datenpunkten einer Verteilung mit 1%, 2%, 5%, 10% und 20% gewählt worden. Die Verteilung der Mittelpunkte der Fensteranfragen folgt dem jeweiligen Datensatz. Intervalle, die über die Ränder[36] der Verteilung hinausgehen, sind verworfen worden, so dass alle Anfragen einer Größenklasse tatsächlich identische Breiten haben. Die jeweiligen Anfragenmengen werden mit Q_q^{GG} für eine Grundgesamtheit GG und eine Fenstergröße q bezeichnet. Somit steht zum Beispiel $\text{Q}_{0.01}^{\text{CP1}}$ für Anfragen des Datensatzes CP1 mit einer Fenstergröße von 1%.

5.4.5 Die empirische Verteilungsfunktion

Definition 5.13 *Zu einer reellen Zufallsvariablen X ist die Verteilungsfunktion definiert als*

$$F(x) = P(X \leq x).$$

[36] Da der Fehler eines Selektivitätsschätzers zur Instanzselektivität (vgl. [BKS99]) berechnet werden soll, werden die Extremwerte des jeweiligen Datensatzes als Rand angenommen.

Einen statistischen Schätzer zur Verteilungsfunktion basierend auf einer Stichprobe erhält man durch

$$F_n(x) = \frac{1}{n} \sum_{i=1}^{n} I(X_i \leq x), \tag{5.27}$$

mit der Indikatorfunktion $I(\cdot)$. $F_n(x)$ heißt **empirische Verteilungsfunktion** *oder auch* **Stichprobenverteilung**. *Anders ausgedrückt gibt Formel (5.27) für alle reellen x die relative Häufigkeit des Ereignisses an, dessen Verteilungsfunktion durch $F(x)$ gegeben ist.*

Mit Hilfe der empirischen Verteilungsfunktion und einer *i.i.d.* Stichprobe ist es leicht möglich die Wahrscheinlichkeit zu schätzen, dass für ein $x \in \mathbb{R}$ $a \leq x \leq b$ gilt. Diese einfache Möglichkeit zur Selektivitätsschätzung wird seit ihrer ersten Verwendung in Datenbanksystemen mit *Sampling* bezeichnet. Diese Wahl des Namens ist meines Erachtens nach unglücklich und irreführend, basiert aber wohl auf der Tatsache, dass die empirische Stichprobenfunktion durch einfaches Zählen der Stichprobenwerte in einem Intervall $[a, b]$ ausgewertet werden kann. Die empirische Stichprobenfunktion hat einige sehr schöne Eigenschaften, so gilt zum Beispiel folgender Satz:

Satz 5.14 [37] *Für ein festes $x \in \mathbb{R}$ gilt*

$$lim_{n\to\infty} F_n(x) = F(x) \text{ } p\text{-fast sicher}$$

Betrachtet man $F_n(x)$ in Abhängigkeit der Stichprobe X als Zufallsfunktion, so kann die Statistik

$$D_n = sup_{x\in\mathbb{R}} |F_n(x) - F(x)|$$

definiert werden. Die entsprechenden einseitigen Statistiken snd

$$D_n^+ = sup_{x\in\mathbb{R}} (F_n(x) - F(x))$$

und

$$D_n^- = sup_{x\in\mathbb{R}} (F(x) - F_n(x)).$$

Eine wichtige Eigenschaft dieser Statistiken ist

$$lim_{n\to\infty} D_n = 0 \text{ p-fast sicher.}$$

Zudem gilt

$$lim_{n\to\infty} P\left(n^{\frac{1}{2}} D_n \leq y\right) = \sum_{k=-\infty}^{\infty} (-1)^k e^{-2k^2y^2} \text{ für } y > 0$$

und weiter für $y > 0$

$$lim_{n\to\infty} P\left(D_n \leq \frac{y}{\sqrt{n}}\right) = 1 - 2 \cdot \sum_{k=1}^{\infty} (-1)^{k-1} e^{-2k^2y^2} := Q(y)$$

[37] Zum Beweis vgl. u. a. [BB98] und [Ras95]

und

$$lim_{n\to\infty} P\left(D_n^+ \leq \frac{y}{\sqrt{n}}\right) = 1 - e^{-2y^2} := Q^+(y).$$

Für $n > 40$ sind[38]

$$P\left(D_n \leq \frac{y}{n}\right) \approx Q(y) \quad \text{und} \quad P\left(D_n^+ \leq \frac{y}{n}\right) \approx Q^+(y)$$

bereits recht gute Näherungen. Damit sind die oben angegebenen Statistiken verteilungsfrei[39].

Doch trotz der oben genannten Eigenschaften wird die empirische Verteilungsfunktion als Schätzer für eine unbekannte Verteilung in der statistischen Praxis im Gegensatz zur Datenbankwelt selten verwendet, da für signifikante Aussagen im Allgemeinen eine zu große Stichprobe benötigt wird[40]. Jedoch muss sich ein Verfahren zur Selektitätsschätzung dem Vergleich mit einfachem *Sampling* stellen, da dieses Verfahren nicht zuletzt wegen der einfachen Handhabung in einigen kommerziellen Datenbanksystemen Verwendung findet. So wird eine Selektivitätsschätzung mit Hilfe der empirischen Verteilungsfunktion auch in dieser Arbeit als Referenzverfahren verwendet.

In XXL kann die empirische Verteilungsfunktion gemäß Formel (5.27) direkt durch die Klasse `EmpiricalCDF` im *Package* `xxl.statistics` sowohl zur Selektivitätsschätzung einer Fensteranfrage als auch zur Schätzung einer Verteilungsfunktion verwendet werden. Ein Aufruf der Form

```
Function sampling = new EmpiricalCDF( sample);
```

erzeugt eine neue Instanz über einer Stichprobe `sample`, die in Form eines `Object[]` übergeben wird. Die Klasse `EmpiricalCDFAggregateFunction` stellt das Verfahren zur Verwendung mit einem Aggregator-Cursor als Aggregationsfunktion[41] zur Verfügung.

5.5 Ergebnisse der Tests

Im folgenden Abschnitt werden die in dieser Arbeit vorgestellten Verfahren evaluiert und ihre Leistungsfähigkeit in praxisnahen Anwendungen unter Beweis gestellt. Zunächst werden die wichtigsten Parameter der unterschiedlichen Verfahren und deren Einfluss auf die Schätzer aufgezeigt. Dazu kommen fast ausschließlich Dichteschätzer zum Einsatz. Dies liegt zum einen daran, dass anhand der Dichte einer Verteilung die Verdeutlichung einiger Effekte leichter fällt, zum anderen ist im Falle der erzeugten Datensätze die wahre Dichte bekannt und kann so als Referenzkurve dienen.

In einem zweiten Teil werden mit Hilfe der Verteilungsschätzer die Selektivitäten einer vorgegebenen Menge von Fensteranfragen mit bekannten Selektivitäten geschätzt. Mit Hilfe dieser praxisnahen Anwendung kann der Fehler bezüglich der wahren Selektivitäten als Maß für die Leistungsfähigkeit der Verfahren dienen.

[38] vgl. [Bos86], Seite 250 ff.

[39] Diese Statistiken sind verteilungsfrei und nicht nichtparametrisch, wie oft fälscherlicherweise angegebnen wird. Vgl. dazu auch Abschnitt 2.2.4

[40] vgl. [BB98]

[41] vgl. Abschnitt 2.5.2

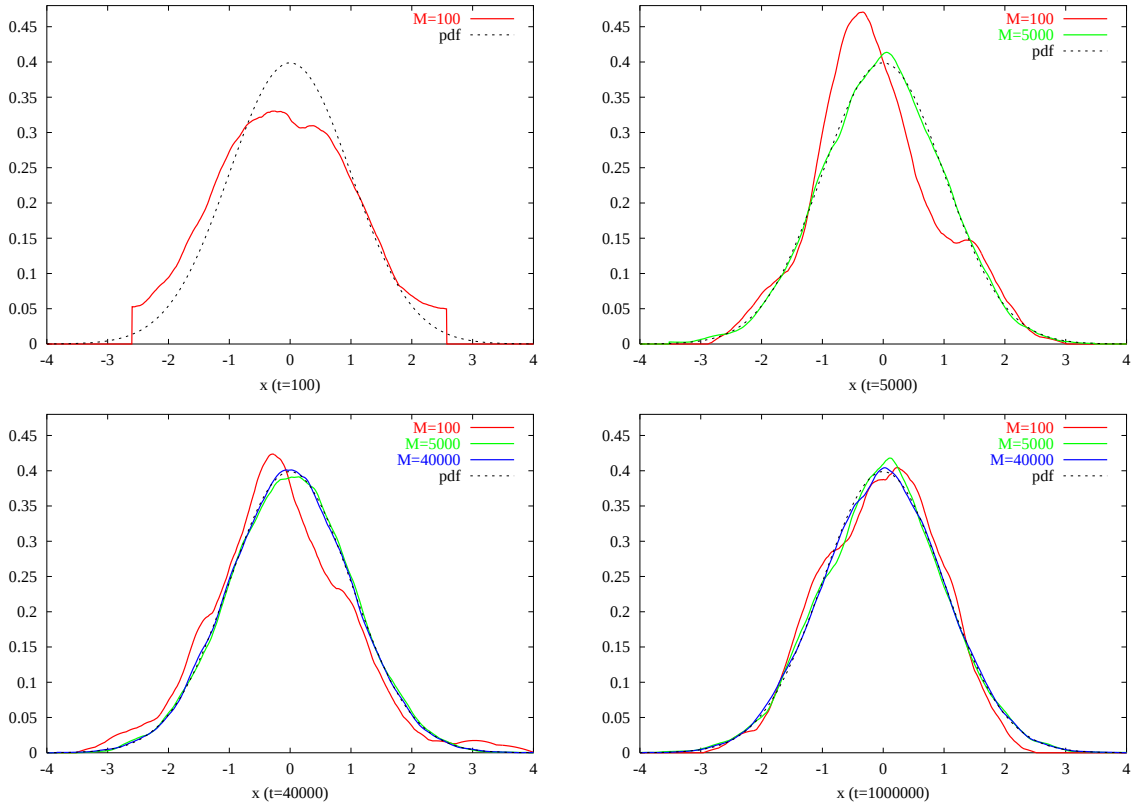

Abbildung 5.8: Gespiegelte speicherbasierte Dichteschätzer über normalverteilten Daten mit unterschiedlichen Speichergrößen

5.5.1 Einfluss der Speichergröße auf speicherbasierte Dichteschätzer

Die in Kapitel 4 entwickelten speicherbasierten Verfahren bieten die Möglichkeit, einfache Verfahren der schließenden Statistik zu einem vorgegebenen festen zur Verfügung stehenden Speicherplatz über Datenströmen anzuwenden. Damit stellt sich die Frage des Einflusses des zur Verfügung stehenden Speicherplatzes. Anhand einer speicherbasierten Dichteschätzung mit stark unterschiedlichen Größen der verwendeten Speicher werden zwei gegenläufige Effekte sichtbar. Abbildung 5.8 zeigt drei speicherbasierte Kerndichteschätzer über einem normalverteilten Datensatz (N(0,1)) und die wahre Dichte (*pdf*) als Referenzkurve. Alle drei Schätzer verwenden die Technik der Spiegelung zur Randbehandlung, eine Epanechnikow-Kernfunktion und die Normalskalierungsregel zur Bestimmung der Bandbreite. Einen Aggregator, der die Schätzer über der Eingabemenge `input` zur gegebenen Speichergröße `reservoirSize` zurück liefert, kann durch den Aufruf der *Factory*-Methode `getRKDEAggregator(...)` der Klasse `xxl.statistics.Aggregators` erzeugt werden.

```
Aggregator aggregator = Aggregators.getRKDEAggregator(
    input,
    new EpanechnikowKernel(),
    reservoirSize,
    ReservoirSampler.XTYPE
    KernelBandwidths.THUMB_RULE_1D
);
```

Damit unterscheiden sich die drei Schätzer nur in dem zur Verfügung stehenden Speicher M. Die rote Kurve zeigt einen Schätzer zu einem Speicherplatz vom $M = 100$. Dieser Schätzer liefert eine erste recht ungenaue Schätzung der wahren Dichte bereits zum Zeitpunkt $t = 100$. Ein zweiter Schätzer zu einem Speicherplatz $M = 5\,000$ (grüne Kurve) zeigt bereits ein besseres Bild, liefert eine erste Schätzung jedoch erst zu einem erheblich späterer Zeitpunkt $t = 5000$. Der dritte Schätzer (blaue Kurve) mit $M = 40\,000$ zeigt augenscheinlich die besten Ergebnisse, liefert jedoch erst zum Zeitpunkt $t = 40\,000$ eine erste Schätzung.

Somit zeigt sich folgendes Bild. Die Qualität der Schätzung steigt offensichtlich mit zunehmendem Speicher. Jedoch hängt der Zeitpunkt der ersten zur Verfügung stehenden Schätzung direkt vom verwendeten Speicher ab.

5.5.2 Einfluss des Grads der Sortiertheit auf speicherbasierte Dichteschätzer

Neben denjenigen Parametern, die die Verfahren selbst zur Verfügung stellen, beeinflussen auch die Daten die Schätzer und den Verlauf der Schätzungen über die Zeit hinweg. Alle in dieser Arbeit vorgestellten Verfahren der mathematischen Statistik basieren auf der Annahme, dass die zur Verfügung stehenden Teilmengen der Daten einfache Zufallstichproben der gesamten Daten darstellen. Liefert der Datenstrom die Datenwerte jedoch in einer sortierten oder teilweise sortierten Reihenfolge, hat dies direkte Auswirkungen auf die Qualität der Schätzer.

Abbildung 5.9 zeigt drei einfache speicherbasierte Kerndichteschätzer an vier Zeitpunkten über unterschiedlich stark sortierten Datenströmen einer normalverteilten Grundgesamtheit der Größe $N = 1\,000\,000$. Die entsprechenden Aggregatoren können wie oben mittels einer *Factory*-Methode der Klasse `xxl.statistics.Aggregators` erzeugt werden.

```
Aggregator aggregator = Aggregators.getNKDEAggregator(
    input,
    new EpanechnikowKernel(),
    2000,
    ReservoirSampler.XTYPE
    KernelBandwidths.THUMB_RULE_1D
);
```

Dabei liefert der Eingabedatenstrom `input` normalverteilte Daten in verschiedenen Sortierungen. Die blaue Kurve zeigt den Schätzer über fast vollständig sortierten Daten

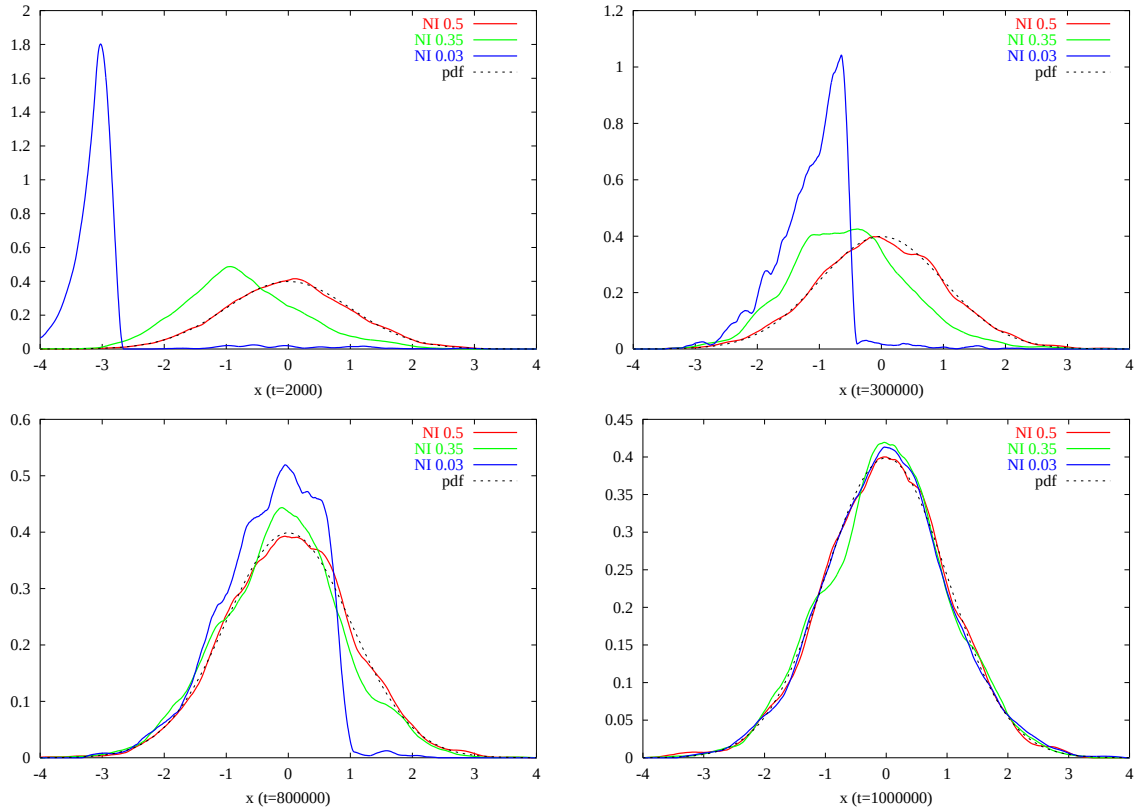

Abbildung 5.9: Einfacher speicherbasierter Dichteschätzer an vier Zeitpunkten über verschieden sortierten Datensätzen

$N(0,1)_{0.03}$ mit 14 999 985 006 Inversionen[42] und 44 417 *Runs*. Deutlich zeigt sich hier die Annäherung an die Referenzkurve der Dichte *pdf* über die Zeit hinweg. Während zum Zeitpunkt der Initialisierung nur ein kleiner Bereich vom Schätzer berücksichtigt wird, kann zum Zeitpunkt $t = 30\,000$ bereits eine bessere Aussage über die Dichte gemacht werden. Deutlich ist das „wandern“ des Schätzers hin zur wahren Dichte zu erkennen. Ein ähnliches nicht so extremes Bild zeigt die grüne Kurve. Dieser Schätzer arbeitet auf einem Datenstrom $N(0,1)_{0.35}$ mit 174 999 825 004 Inversionen[43] und 499 945 *Runs*. Kommen die Daten jedoch in annähernd zufälliger Reihenfolge, so zeigt der Schätzer bereits zum Zeitpunkt der Initialisierung ein klares Bild der wahren Dichte. Die rote Kurve verdeutlicht dies anhand eines Schätzers über einem randomisierten Datenstrom $N(0,1)$ mit 249 713 078 234 Inversionen[44] und 500 225 *Runs*.

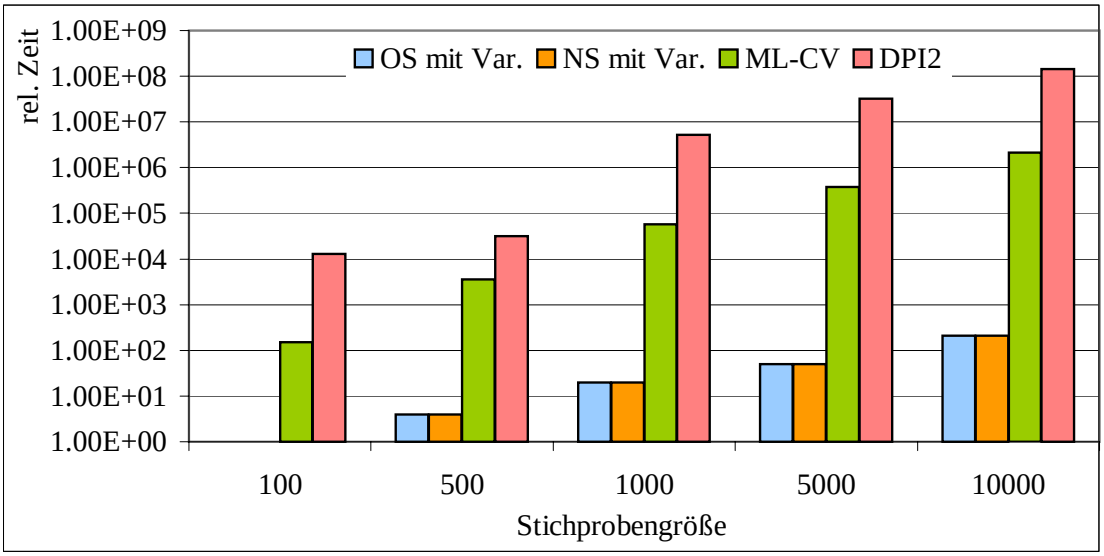

Abbildung 5.10: Relative Berechnungszeiten verschiedener Bandbreiten auf logarithmischer Skala

5.5.3 Einfluss der Verfahrensparameter auf speicherbasierte Schätzer

Der Einfluss der Größe des zur Verfügung stehenden Speichers ist bereits in Abschnitt 5.5.1 diskutiert worden. Bei den verbleibenden Parametern spielen sowohl die verwendete Kernfunktion als auch die Strategie des Ziehens der Stichprobe eine untergeordnete Rolle[45].

Lediglich das Verfahren zur Bestimmung der Bandbreite hat größeren Einfluss auf den Schätzer. Welche Bandbreite für welche Daten günstig ist, hat Korus in [Kor99] bereits untersucht. In dem hier verwendeten Modell der Berechnung über einem Datenstrom ist ein anderer Aspekt besonders wichtig. Die Bandbreite eines Schätzers muss unüberwacht und in beschränkter Zeit zu berechnen sein. Benötigt die Berechnung einer Bandbreite zu viel Zeit, kann unter Umständen die Forderung nach konstantem Aufwand pro Datenwert[46] nicht erfüllt werden.

Die theoretischen Komplexitäten der Algorithmen zur Berechnung der verschiedenen Bandbreiten sind einfach an den Berechnungsvorschriften abzulesen. So benötigen die einfache Normalskalierungs- und Überglättungsregel bei bekannter Varianz $\Theta(1)$. Muss die Varianz selbst mit Hilfe der Stichprobe berechnet werden, so benötigen beide Verfahren für eine Stichprobe der Größe n $\Theta(n)$. Der Aufwand eines auf Kreuzvali-

[42] Dieser Wert entspricht einer normierten Inversionszahl von 0.03.
[43] Dieser Wert entspricht einer normierten Inversionszahl von 0.35.
[44] Dieser Wert entspricht ungefähr einer normierten Inversionszahl von 0.5.
[45] Der Einfluss der Kernfunktion ist in [Blo98] und [Kor99] untersucht worden. Den Einfluss der Strategie zum Ziehen einer speicherbasierten Stichprobe zeigt Abschnitt 4.2.4 auf.
[46] vgl. Abschnitt 2.3 und Abschnitt 3.4

Bandbreite/Stichprobengröße	100	500	2000	5000	10000
Überglättungregel (Streuung bekannt)	*n.m.*	*n.m.*	*n.m.*	*n.m.*	1
Überglättungregel (Streuung unbekannt)	*n.m.*	4	20	50	210
Normalskalierung (Streuung bekannt)	*n.m.*	*n.m.*	*n.m.*	*n.m.*	1
Normalskalierung (Streuung unbekannt)	*n.m.*	4	20	50	210
Maximum-Likelihood-CV	160	3600	58 Tsd.	378 Tsd.	2.12 Mio.
Zweistufige Einsetzungsregel	13000	320 Tsd.	5.15 Mio.	32.4 Mio.	141.5 Mio.

Tabelle 5.1: Relative Berechnungszeiten verschiedener Bandbreiten über verschieden großen Stichproben

dierung beruhenden Verfahrens beträgt immer mindestens $\Theta(n^2)$ aufgrund des *Leave-One-Out*-Prinzips, bei dem alle Stichprobenwerte paarweise kombiniert werden. Dies gilt somit auch für die in XXL zur Verfügung stehende *Maximum-Likelihood-CV*. Die in der statistischen Praxis sehr wichtigen Einsetzungsregeln arbeiten in mehreren Stufen. Die zweistufige Einsetzungsregel muss in jeder Berechnungsstufe die *Roughness* der unbekannten Dichte schätzen. Dies geschieht in der Regel durch eine Kreuzvalidierung der Stichprobe. Damit ist der Aufwand zur Berechnung einer solchen Bandbreite ebenfalls $\Theta(n^2)$.

Die Auswirkungen der theoretischen Komplexitäten zeigen sich, wenn die benötigte Zeit relativ zum schnellsten Verfahren zur Berechnung über verschieden großen Stichproben gemessen wird. Tabelle 5.1 zeigt die gemessenen Zeiten relativ zueinander für verschiedene Stichprobengrößen. Dabei bezeichnet *n.m.* eine zum Messen zu geringe Berechnungszeit. Deutlich zeigt sich das Ansteigen der Berechnungzeiten für die Verfahren mit quadratischem Aufwand. Die aufgrund der theoretischen Komplexitäten unerwartet großen Unterschiede zwischen dem Verfahren der *Maximum-Likelihood-CV* und der zweistufigen Einsetzungsregel ergeben sich durch die Tatsache, dass der tatsächliche Aufwand für die Einsetzungsregel durch das mehrstufige Prinzip und den notwendigen Berechnungen höherer Ableitungen von Kernfunktionen um ein Vielfaches größer ist. Abbildung 5.10 zeigt die benötigte relative Zeit zur Berechnung derjenigen Bandbreiten, deren Komplexität mindestens linear ist, auf einer logarithmischen Skala. Deutlich ist das quadratische Anwachsen der Zeiten für die Einsetzungs- und Kreuzvalidierungsregel zu erkennen.

Da die Überglättungsregel nicht zur direkten Verwendung als Bandbreite eines Dichteschätzers konzipiert ist und die *Maximum-Likelihood-CV* nicht unüberwacht[47] verwendet werden kann, bleibt somit lediglich die Normalskalierungsregel zur uneingeschränkten Verwendung über Datenströmen.

5.5.4 Einfluss der Gewichte auf Adaptive Dichteschätzer

Ähnlich den speicherbasierten Verfahren bestimmt auch eine Vielzahl unterschiedlicher Parameter die Leistungsfähigkeit und das Verhalten der adaptiven Schätzer. In diesem Abschnitt wird auf den Einfluss der Strategie zur Wahl der Gewichte unter

[47] vgl. [WJ95]

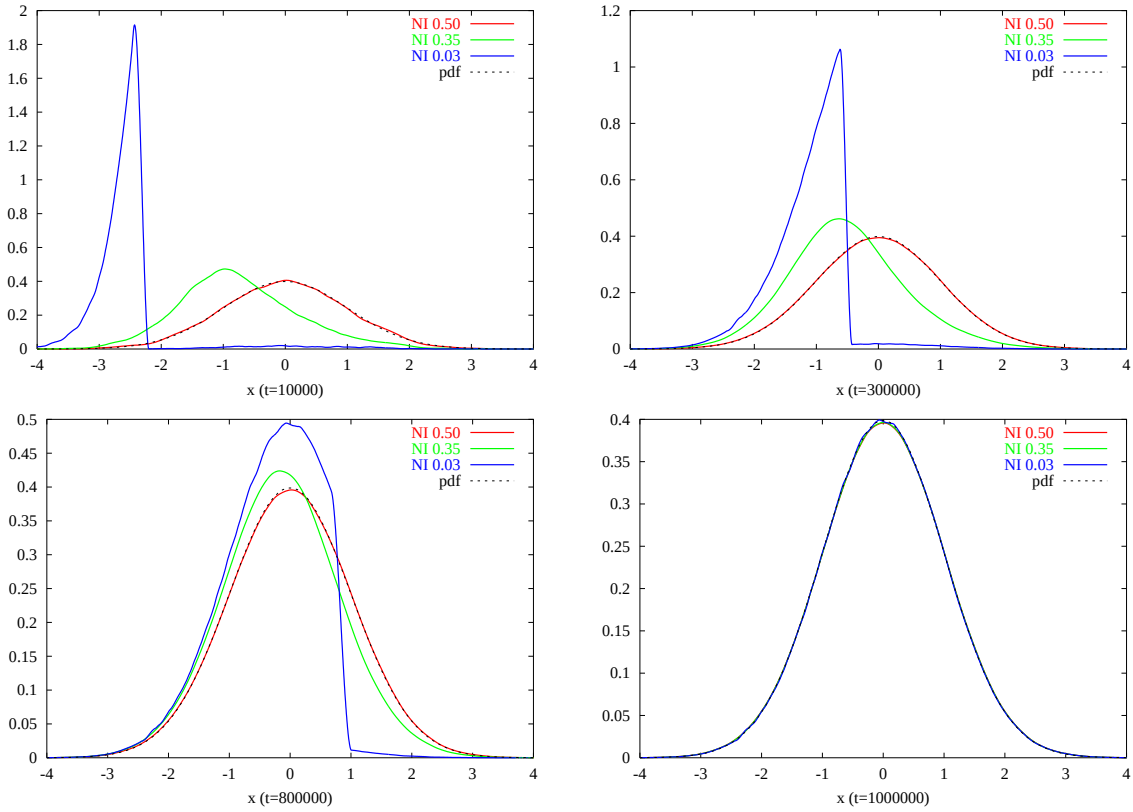

Abbildung 5.11: Gespiegelter adaptiver und komprimierter Dichteschätzer an vier Zeitpunkten über verschieden sortierten Datensätzen mit arithmetischer Gewichtung

verschiedenen Bedingungen eingegangen.

5.5.4.1 Arithmetische Gewichte und teilsortierte Daten

Soll ein Dichteschätzer über einem randomisierten Datenstrom erzeugt werden, ist die arithmetische Gewichtung sicherlich eine intuitive Wahl. Bereits zum Zeitpunkt der Initialisierung liefert das adaptive Verfahren einen Schätzer mit einer dem äquivalenten speicherbasierten Verfahren entsprechenden Qualität. Da die Elemente des Datenstroms in zufälliger Reihenfolge verarbeitet werden, erfüllt bereits der erste Block die Eigenschaften einer *i.i.d.* Stichprobe. Mit zunehmender Verarbeitung steigt die Qualität des Schätzers bis hin zu einem kaum erkennbaren Fehler nach einem Konsumieren aller Elemente. Abbildung 5.11 zeigt dies anhand der roten Kurve zu vier unterschiedlichen Zeitpunkten über $\mathrm{N}(0,1)_{0.5}$ Daten.

Ist der Datenstrom jedoch sortiert oder teilsortiert, bildet der erste Block keine *i.i.d.* Stichprobe. Somit kann zu diesem Zeitpunkt keine Aussage über den tatsächlichen Verlauf der Dichte gemacht werden. Wegen der arithmetischen Gewichtung nähert sich der Schätzer jedoch mit zunehmender Verarbeitung der Daten immer mehr der tatsächlichen Funktion an. In Abbildung 5.11 kann dies mittels der grünen und blauen Kurve verifiziert werden. Dabei zeigt die grüne Kurve den Verlauf der Schätzfunktion

über einem teilsortierten Datenstrom gemäß $N(0,1)_{0.35}$. Der extreme Verlauf der blauen Kurve erklärt sich durch den verwendeten Datensatz $N(0,1)_{0.03}$.
Die in Abbildung 5.11 dargestellten Schätzer werden durch folgenden Aufruf erzeugt:

```
Aggregator aggregator =
    Aggregators.getSplineCompressedAdaptiveKernelBasedAggregator(
        ReflectionCDF.FACTORY,                      // Schätzer
        input,
        new EpanechnikowKernel(),                   // Kernfunktion
        KernelBandwidths.THUMB_RULE_1D,             // Bandbreite
        10000                                       // Blockgröße
        new AdaptiveWeightFunctions.ArithmeticWeights(),
        666,                                        // Anzahl Stützstellen
        true,                                       // real mode
        false                                       // no cdf mode
);
```

Dabei bezeichnet `input` den gegebenenfalls teilsortierten Datenstrom. Die Klasse `AdaptiveWeightFunctions` im *Package* `xxl.statistics` stellt dabei verschiedene Strategien für die Wahl der Gewichte zur Verfügung.

5.5.4.2 Geometrische Gewichte und teilsortierte Daten

Eine geometrische Gewichtung über teilsortierten Daten zeigt einen anderen Effekt. Da geometrische Gewichte je nach Wahl des Parameters ein stärkeres oder schwächeres zeitliches Gedächtnis bewirken, „wandert“ die Schätzfunktion mit der Sortierung über die Daten. Abbildung 5.12 zeigt diesen Effekt am Beispiel teilsortierter N(0,1) Daten. Ein Gewichtsparameter $c = 2$ verursacht eine Gleichgewichtung des aktuellen Blocks und des bisherigen Schätzers und damit aller bisheriger Blöcke. So zeigt die grüne Kurve in Abbildung 5.12, welchen Effekt dies auf teilsortierte Daten gemäß $N(0,1)_{0.35}$ hat. Noch stärker ist dieser Effekt bei fast vollständig sortierten Daten gemäß $N(0,1)_{0.03}$ zu beobachten. Hier kann anhand des Schätzers der Verlauf der Sortierung bzw. des Datenstroms nachvollzogen werden. Dieser Effekt ist umso stärker, je größer der Grad der Sortiertheit eines Datenstroms ist. Somit ist eine geometrische Strategie optimal, falls neueren Daten eine größere Bedeutung als älteren beigemessen wird, diese jedoch nicht völlig ignoriert werden können oder sollen.

Die rote Kurve zeigt zum Vergleich eine geometrische Gewichtung über randomisierten Daten gemäß $N(0,1)_{0,5}$. Die Konstruktion der Schätzer aus Abbildung 5.12 erfolgt mit folgendem Aufruf:

```
Aggregator aggregator =
    Aggregators.getSplineCompressedAdaptiveKernelBasedAggregator(
        ReflectionCDF.FACTORY,                      // Schätzer
        input,
        new EpanechnikowKernel(),                   // Kernfunktion
        KernelBandwidths.THUMB_RULE_1D,             // Bandbreite
```

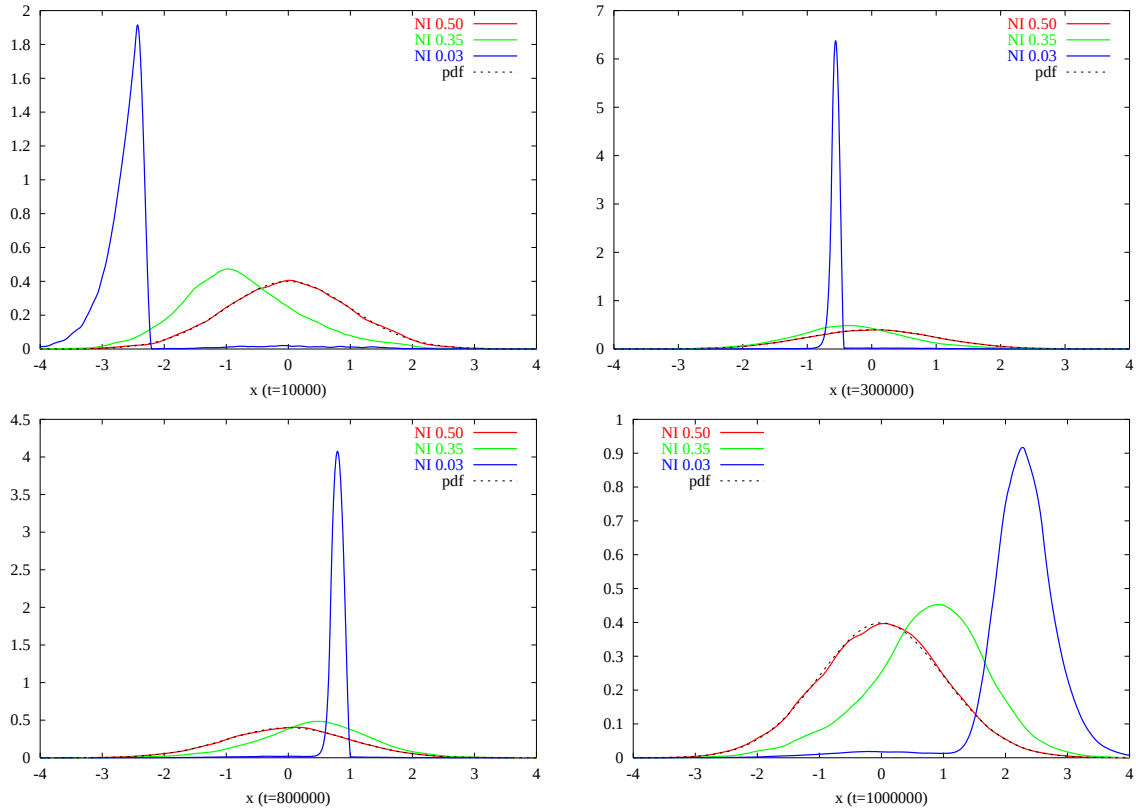

Abbildung 5.12: Gespiegelter adaptiver und komprimierter Dichteschätzer an vier Zeitpunkten über verschieden sortierten Datensätzen mit geometrischer Gewichtung

```
        10000                                          // Blockgröße
        new AdaptiveWeightFunctions.GeometricWeights( 2.0),
        666,                                           // Anzahl Stützstellen
        true,                                          // real mode
        false                                          // no cdf mode
);
```

Auch hier bezeichnet `input` die unterschiedlich sortierten Eingabedatenströme.

5.5.4.3 Teilsortierte Daten und unterschiedliche Gewichtsfunktionen

In XXL stehen bereits diverse Strategien zur Wahl der Gewichte eines adaptiven Schätzers zur Verfügung. Neben möglichen benutzerdefinierten datenabhängigen Gewichten können datenunabhängige Gewichte unterschiedliche Präferenzen eines Nutzers widerspiegeln. Bereits in den vorangegangenen Abschnitten ist der Einfluss einer arithmetischen oder geometrischen Gewichtung auf teilsortierten Datenströmen untersucht worden. Abbildung 5.13 zeigt drei verschiedene Gewichtungsstrategien über teilsortierten Daten des Typs $N(0,1)_{0.35}$. Die entsprechenden Gewichtsfunktionen sind jeweils über einen der drei folgenden Aufrufe erzeugt worden.

```
RealFunction weights = new AdaptiveWeightFunctions.ArithmeticWeights();
```

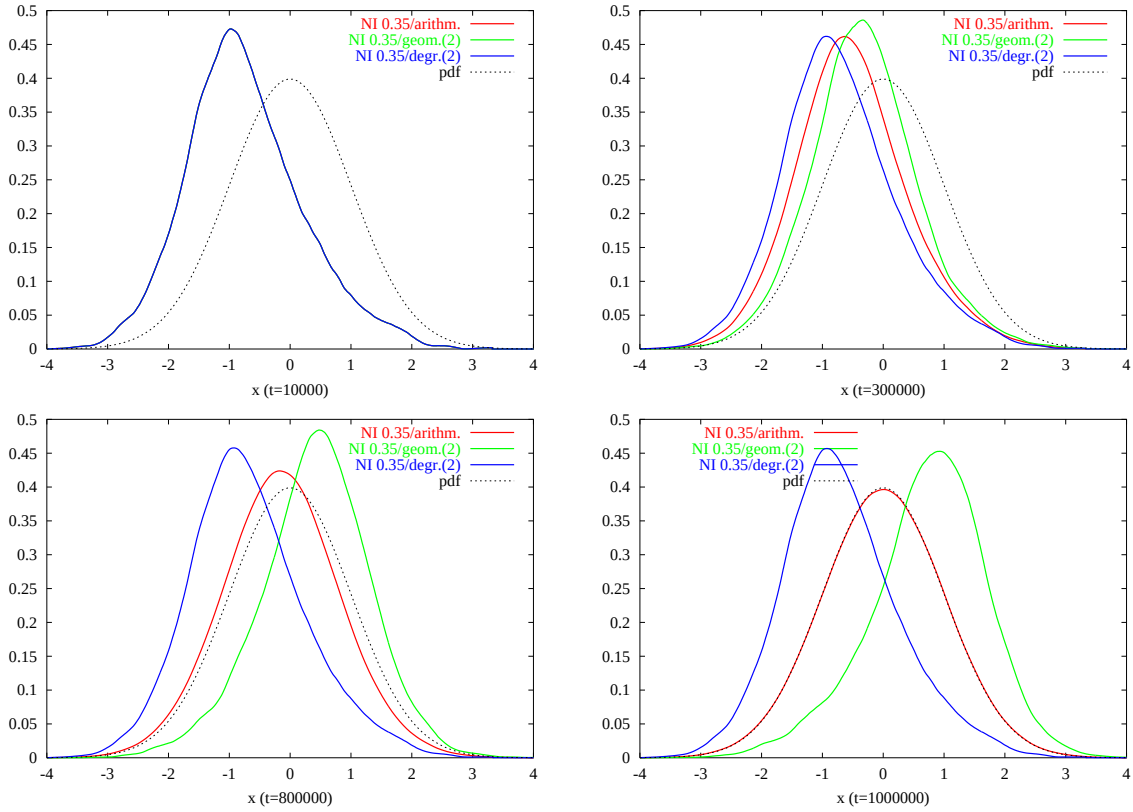

Abbildung 5.13: Gespiegelter adaptiver und komprimierter Dichteschätzer an vier Zeitpunkten über teilsortierten Daten mit unterschiedlicher Gewichtung

```
RealFunction weights =
    new AdaptiveWeightFunctions.GeometricWeights( 2.0);
RealFunction weights =
    new AdaptiveWeightFunctions.ProgressiveDegressiveWeights( 2.0);
```

Der direkte Vergleich verschiedener Gewichte über teilsortierten Daten lässt die Charakteristika dieser Gewichte gut erkennen. So zeigt in Abbildung 5.13 die blaue Kurve eine degressive Gewichtung mit $\alpha = 2$, die grüne Kurve eine geometrische Gewichtung mit $c = 2$ und die rote Kurve eine arithmetische Gewichtung.

Die rote Kurve nähert sich erwartungsgemäß der wahren Dichte *pdf* über die Zeit an. Deutlich ist jedoch der Unterschied zwischen der blauen und der grünen Kurve zu erkennen. Während die blaue Kurve durch die stärkere Gewichtung älterer Daten fast in der Form zum Zeitpunkt der Initialisierung ($t = 10\,000$) verbleibt, passt sich die grüne Kurve immer direkt mit einer mittels des Parameters c steuerbaren Verzögerung der Sortierung der Daten an. Die Aggregatoren aus Abbildung 5.13 sind mit einem Iterator `input` über dem Datensatz vom Typ $\mathrm{N}(0,1)_{0.35}$ und entsprechender Gewichtsfunktion `weights` erzeugt worden.

```
Aggregator aggregator =
    Aggregators.getSplineCompressedAdaptiveKernelBasedAggregator(
        ReflectionKernelDensityEstimator.FACTORY,
```

```
        input,                                      // Datenstrom
        new EpanechnikowKernel(),                   // Kernfunktion
        KernelBandwidths.THUMB_RULE_1D,             // Bandbreite
        10000                                       // Blockgröße
        weights                          // verschiedene Gewichtsfunktionen
        666,                                        // Anzahl Stützstellen
        true,                                       // real mode
        false                                       // no cdf mode
);
```

Die Charakteristika der oben untersuchten Gewichtsfunktionen können mit Hilfe eines anderen Experiments noch deutlicher gezeigt werden. Abbildung 5.14 zeigt den Verlauf eines adaptiven Dichteschätzers über die Zeit mit einer Änderung der Verteilung zum Zeitpunkt $t = 500\,000$.

Zum Einsatz kommt dabei ein modifizierter Datenstrom über dem N(0,1) Datensatz. Nach der Verarbeitung der Hälfte[48] der Daten des randomisierten Datenstroms werden alle Elemente um den Wert $\delta = 3$ nach rechts verschoben. Somit sind die Daten zwischen $t = 500\,001$ und $t = 1\,000\,000$ nicht mehr länger $N(0, 1)$-verteilt, sondern folgen der $N(3, 1)$-Verteilung.

Deutlich ist der Einfluss dieser Änderung auf den arithmetisch gewichteten, adaptiven Schätzer (rote Kurve) zu erkennen. So ist bereits zum Zeitpunkt $t = 600\,000$ eine Veränderung der Kurve zu erkennen. Bei $t = 700\,000$ zeigt der Schätzer bereits einen bimodalen Charakter, der sich bis zur vollständigen Verarbeitung der Daten verstärkt. Der Modus der blauen Kurve bleibt jedoch stabil über $x = 0$ durch die schwächere Gewichtung neuerer Daten. Der Schätzer mit der geometrischen Gewichtung (grüne Kurve) hat dagegen bereits zum Zeitpunkt $t = 600\,000$ das Aussehen der neuen Verteilung angenommen.

So zeigen zum Zeitpunkt der Verteilungsänderung alle Strategien ein einheitliches Bild, reagieren jedoch unterschiedlich auf die veränderte Situation. Eine degressive Gewichtung kann somit zum Einsatz kommen, wenn die Qualität früherer Daten im Gegensatz zu später verarbeiteten Daten gesichert ist.

5.5.4.4 Speicherbasierter und adaptiver Difference Kernel Estimator

Abbildung 5.15 zeigt deutlich die Unterschiede zwischen den speicherbasierten und den adaptiven *Difference Kernel Estimators* (DKE). Während eine adaptive Vorgehensweise zu einer im gesamten glatten Kurve führt, oszilliert der speicherbasiert aufgebaute DKE sehr stark. Diese Schwankungen sind jedoch nicht stark genug, um einen signifikanten Unterschied bei der Verwendung der verschiedenen DKE-Funktionen mit einem *Difference Apart Kernel Estimator* (DAKE) zu bewirken. Jedoch liefert der DAKE bei Verwendung eines adaptiven DKE immer die gleiche Anzahl an Kandidaten für Sprungstellen mit annähernd den gleichen Positionen. Werden speicherbasierte DKE-Funktionen genutzt, variieren sowohl die Anzahl der gefundenen Sprünge als auch deren

[48] Dies bedeutet zum Zeitpunkt $t = 500\,000$.

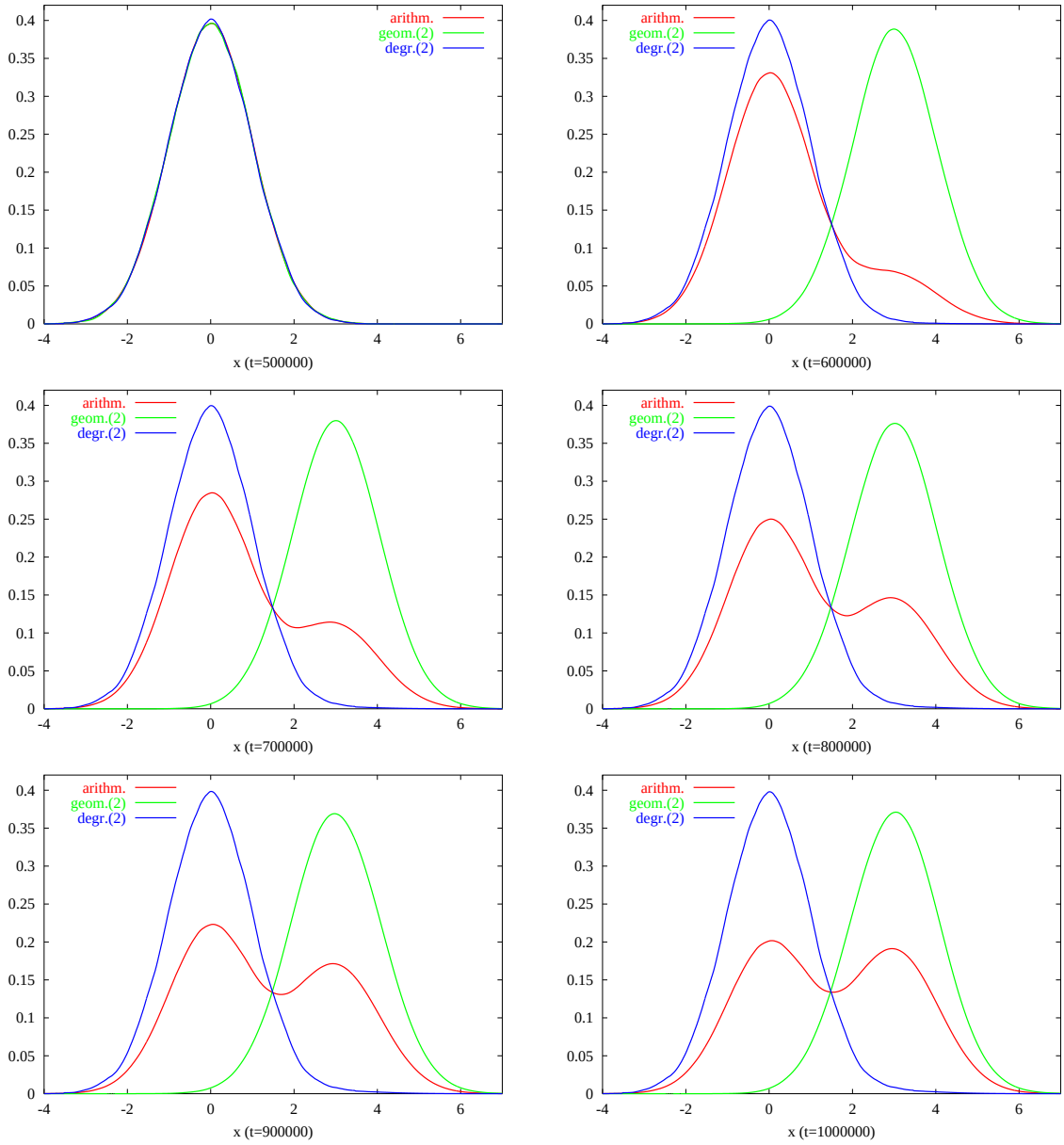

Abbildung 5.14: Gespiegelte adaptive und komprimierte Dichteschätzer an sechs Zeitpunkten über modifizierten normalverteilten Daten mit unterschiedlicher Gewichtung

Positionen gering.

So findet der DAKE bei einer Vorgabe des Schwellenwertes $b_n = 0.4$ bei Daten des Typs CP2 mit Hilfe eines adaptiven DKEs Sprünge an den Stellen $x = 0.301$, $x = 0.5$ und $x = 0.601$[49]. Damit hat der DAKE die beiden Sprungstellen in den Daten als solche idendifiziert. Der zusätzliche Kandidat für eine Sprungstelle $x = 0.5$ lässt sich leicht

[49]Die x-Werte sind auf drei Nachkommastellen gerundet. Dazu kam ein Parameter $m = 50$ zum Einsatz und der benötigte Speicherplatz war $M = 5000$.

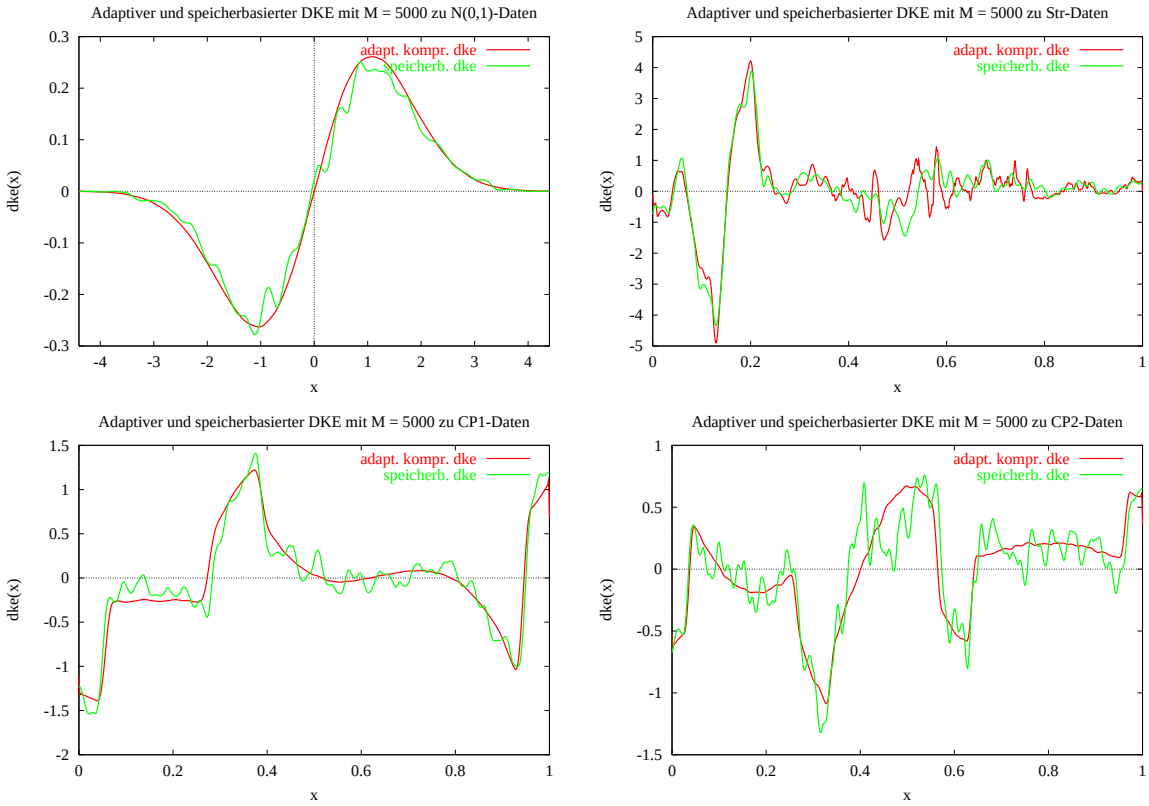

Abbildung 5.15: Speicherbasierte und adaptive *Difference Kernel Estimator* über verschiedenen Daten zu festem Speicher M=5000

bei Betrachtung des Verlaufs der Dichtefunktion der Daten erklären[50]. So besitzt die Dichtefunktion an der Stelle $x = 0.5$ den höchsten Ableitungswert und damit die größte Steigung der gesamten Kurve. Diesen Bereich erkennt der DKE und damit der DAKE als Gebiet mit starken lokalen Unterschieden in der Dichte und somit als Kandidat für eine Sprungstelle.

Im Falle der Daten vom Typ CP1 findet der DAKE zu einem $b_n = 0.9$ die Kandidaten $x = 0.05$ und $x = 0.35$. Während der erste Wert auf Randeffekte zurückzuführen ist und dies durch seine Nähe zum Minimum der Daten auch erkannt werden kann, liegt der zweite Wert dicht an dem tatsächlichen Sprung in $x = 0.3333$.

Bei den Daten des Typs STR ist weder bekannt, ob die Dichte dieser Verteilung Sprünge aufweist, noch wo gegebenenfalls die Sprungstellen liegen. Eine Anwendung des DAKE liefert Kandidaten zu den Punkten $x = 0.116$, $x = 0.1875$, $x = 0.473$ und $x = 0.5806$. Dabei kommt die automatisch berechnete Schranke $b_n = 1.15$ zum Einsatz. Wird diese Schranke erhöht, so fällt als erster Wert $x = 0.5806$ weg. Die verbleibenden drei Werte sind auch in den späteren Tests mit speicherbasierten und adaptiven hybriden Kernschätzern auf den Daten vom Typ STR verwendet worden.

[50] Siehe im Anhang Abbildung A.2.

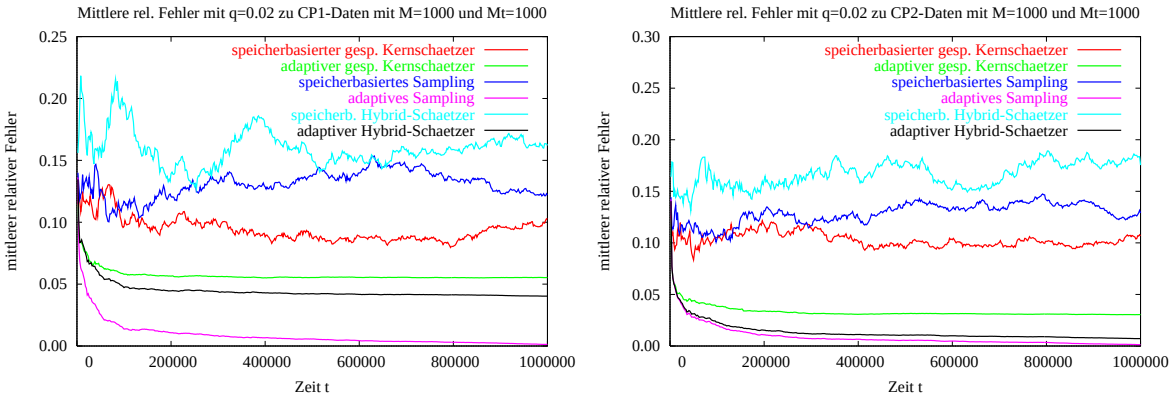

Abbildung 5.16: Vergleich der angewandten Schätzer über Daten mit Sprungstellen

5.6 Evaluation der angewandten Schätzer

Im folgenden Abschnitt werden die in Kapitel 4 vorgestellten Verfahren zur Selektivitätsschätzung mit Hilfe verschiedener Anfragemengen evaluiert. Dazu wird der mittlere Fehler bezüglich jeweils 1000 Fensteranfragen einer festen Größe ermittelt[51].

5.6.1 Vergleich der Verfahren

Ein direkter Vergleich aller vorgestellter Verfahren zur Selektivitätssschätzung zeigt ein in dieser Form nicht erwartetes Ergebnis. So zeigt Abbildung 5.16 den Verlauf des mittleren relativen Fehlers bezüglich Anfragen einer Größe von 2% des Datenraumes[52] über randomisierten Datenströmen der CP1 und CP2 Daten. Alle Verfahren hatten den gleichen Speicher der Größe 2000 zur Verfügung. Die adaptiven Verfahren nutzen jeweils die Hälfte des Speichers zum Aufbau neuer Blockschätzer und zur Speicherung des komprimierten Schätzers selbst. Die weiteren Parameter werden identisch gewählt. So kommt zur Bestimmung der Bandbreite die *normal scale rule* und als Kernfunktion der Biweight-Kern zum Einsatz. Zudem benötigen die Verfahren keinerlei a priori Informationen über die Daten. Lediglich die hybriden Verfahren bekommen die bekannten bzw. bei den Daten des Typs STR_die mit Hilfe des DAKE ermittelten Sprungstellen übergeben.

Wie erwartet sind die mittleren Fehler aller adaptiven Verfahren erheblich geringer als die ihrer speicherbasierten Gegenüber, da sie alle verarbeiteten Daten zum Aufbau der Schätzfunktion heranziehen. Auch der monoton fallende Verlauf der Fehlerkurven der adaptiven Schätzer lässt sich damit erklären. Der oszillierende Verlauf der Fehlerkurven der speicherbasierten Schätzer um ein stabiles Zentrum erklärt sich aus der Funktionsweise der Schätzer. Sie arbeiten zu jedem Zeitpunkt über einer gültigen *i.i.d.* Stichprobe der gesamten Daten[53], die sich jedoch in jedem Zeitpunkt t ändern kann.

[51] vgl. Abschnitte 5.4.4 und 5.4.2
[52] Anfragen vom Typ $Q^{CP1}_{0.02}$ und $Q^{CP2}_{0.02}$
[53] Dies gilt nur, da die Datenreihenfolge zufällig ist.

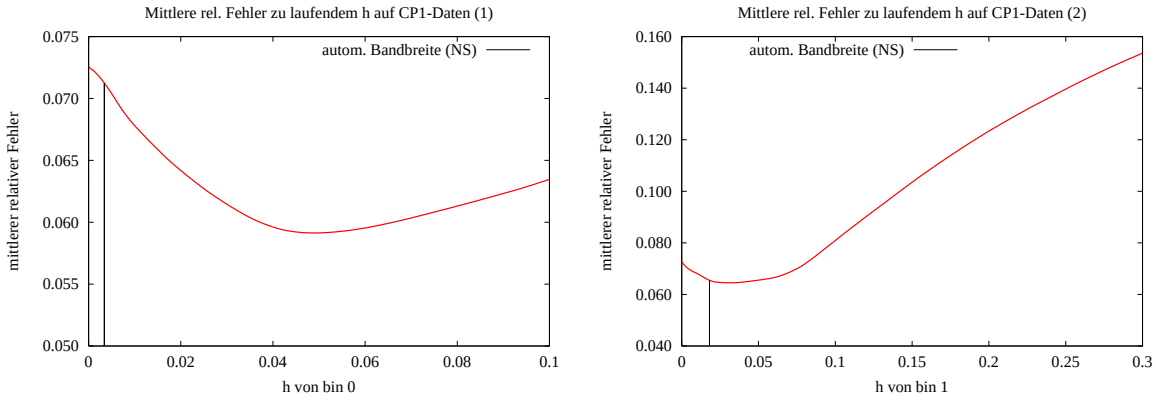

Abbildung 5.17: Relativer Fehler eines hybriden speicherbasierten Kernselektivitätsschätzers

Doch zeigt der speicherbasierte hybride Kernschätzer einen unerwartet hohen Fehler im Vergleich zum einfachen speicherbasierten Kernschätzer. Dies ist umso erstaunlicher, als dass der adaptive hybride Schätzer einen deutlich geringeren Fehler aufweist als der adaptive Kernschätzer. Ein Erklärung für dieses Phänomen liefert eine Untersuchung des Fehlers in Abhängigkeit der verwendeten Bandbreiten für die verschiedenen *Bins* eines hybriden Kernschätzers. Abbildung 5.17 zeigt die mittleren relativen Fehler nach Verarbeitung aller Daten eines speicherbasierten hybriden Kernschätzers zu Anfragen der Größe 2% der CP1 Daten in Abhängigkeit der genutzten Bandbreite h. Auch hier kommt die Normalskalierungsregel und der Biweight-Kern zum Einsatz. Der verwendete Speicher beträgt $M = 5000$. Die automatisch gewählten Bandbreiten für die beiden *Bins* des Schätzers sind jeweils durch die senkrechte schwarze Linie eingezeichnet. Die linke Grafik zeigt den Verlauf des Fehlers in Abhängigkeit zur verwendeten Bandbreite für den linken *Bin* des Schätzers bei gleichzeitigem Festhalten der Bandbreite des rechten *Bins*. Entsprechend zeigt die rechte Grafik den Verlauf des Fehlers bei Variation der Bandbreite des rechten bei konstanter Bandbreite des linken *Bins*.

Deutlich ist der Einfluss der Bandbreiten auf den gesamten Fehler zu sehen. Eine „gute“ Wahl der Bandbreiten ist damit ausschlaggebend für die Qualität eines hybriden Schätzers. Der trotz dieser Problematik geringe Fehler des adaptiven hybriden Schätzers erklärt sich durch die zusätzliche Glättung der Schätzfunktion durch die Komprimierung mit kubischen Splines. Durch zu klein gewählte Bandbreiten verursachte Oszillationen in der Schätzfunktion werden auf diese Weise wieder eliminiert. Die richtige Wahl der Bandbreiten für die verschiedenen *Bins* eines hybriden Schätzers muss damit Ziel zukünftiger Untersuchungen und Experimente sein.

Neben dem unerwartet hohen Fehler des speicherbasierten hybriden Schätzers ist der sehr niedrige Fehler des adaptiven *Samplings* besonders bemerkenswert. Bereits nach Verarbeitung von lediglich 25% der Daten liegt der mittlere Fehler unter 1% und ist damit geringer als die Fehler aller anderen Verfahren nach Verarbeitung aller Daten. Speicherbasiertes *Sampling* liefert dagegen das erwartet schlechte Bild und dominiert lediglich das problembehaftete speicherbasierte hybride Verfahren.

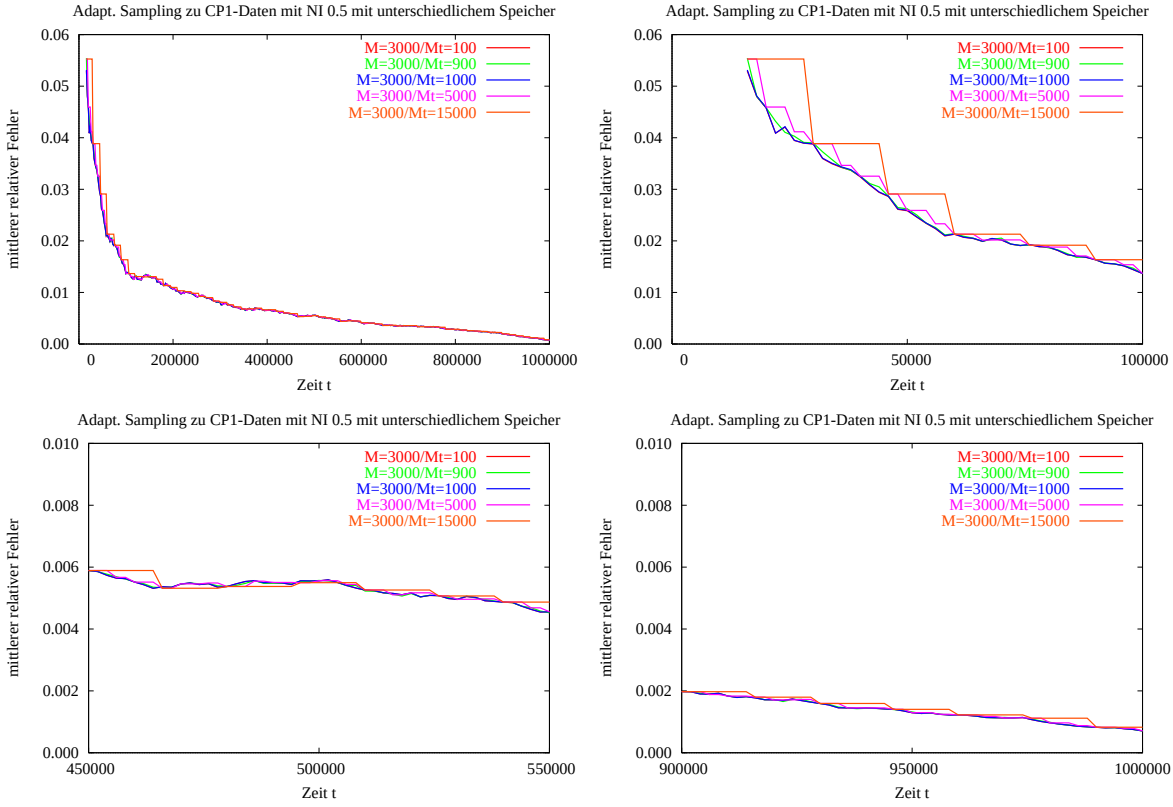

Abbildung 5.18: Relativer Fehler des adaptiven Sampling-Schätzers zu unterschiedlichen Speichergrößen über CP1 und Anfragen $Q_{0.02}^{CP1}$

Die Verwendung der empirischen Stichprobenfunktion als Schätzer für eine unbekannte Verteilung hat in dem hier betrachteten Modell einen großen Vorteil gegenüber anderen Verfahren. Der einzige zu wählende Parameter ist die zur Verfügung stehende Speichergröße zum Aufbau des Schätzers. Damit stellt sich die Frage nach dem Einfluss des Speicherplatzes auf den Fehler eines adaptiven Schätzers auf Basis der empirischen Stichprobenfunktion.

5.6.2 Einfluss der Speichergröße auf adaptives Sampling

Zum Vergleich der Qualität verschiedener adaptiver Schätzer auf Basis der empirischen Stichprobenfunktion mit unterschiedlichen Speicherplatzstrategien und -größen sind folgende Testparameter gewählt worden. Alle Schätzer arbeiten auf randomisierten CP1-Daten und verwenden einen permanenten Speicherplatz von $M = 3000$ zur komprimierten Speicherung des Schätzers. Lediglich der temporär zum Aufbau der Blockschätzer verwendete Speicherplatz wird variiert. Abbildung 5.18 zeigt die mittleren relativen Fehler der Schätzer zu Anfragen der Größe 2%.

Die Grafik links oben in Abbildung 5.18 zeigt den Verlauf aller Schätzer über die Zeit hinweg bis zur Verarbeitung des gesamten Datenstroms. Die Fehlerkurven der Schätzer zeigen lediglich am Beginn des Datenstroms signifikante Unterschiede und

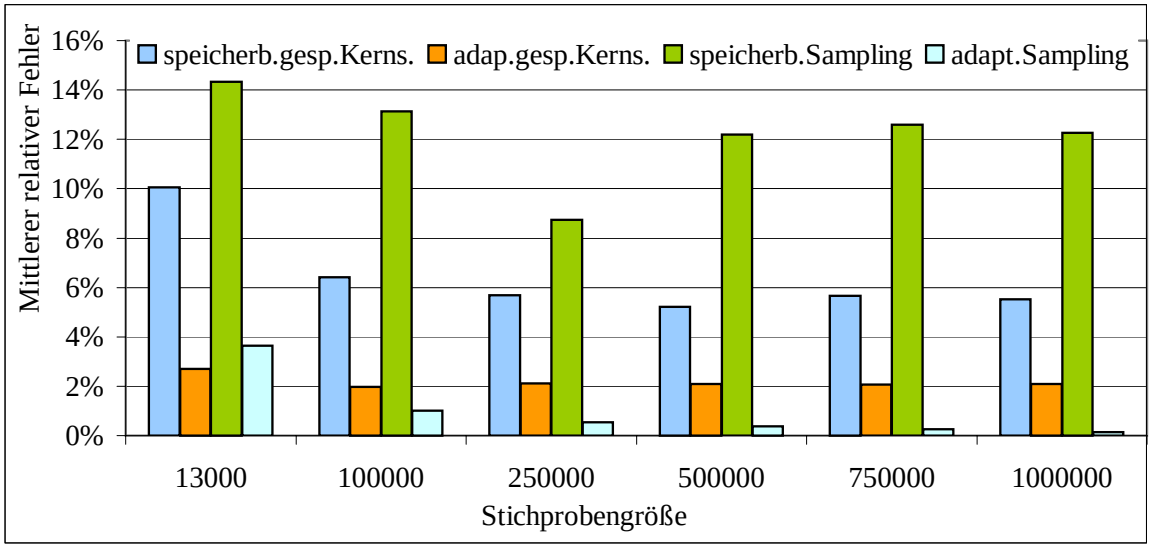

Abbildung 5.19: Mittlere relative Fehler über $N(0,1)_{0.5}$-Daten zu 2% Anfragen an sechs Zeitpunkten

sind im späteren Verlauf kaum noch zu unterscheiden. Die Grafik rechts oben zeigt die Schätzer zu Beginn des Datenstroms in einem Ausschnitt. Hier ist deutlich der Effekt der unterschiedlichen Speichergrößen zu erkennen. Durch die blockweise Verarbeitung der Daten entstehen Treppeneffekte entsprechend der Größe der Blöcke in den Fehlern. Die Grafiken links und rechts unten zeigen jeweils Ausschnitte der Fehlerkurve zur Mitte bzw. zum Ende des Datenstroms.

Bemerkenswert ist, dass der verwendete Speicherplatz nur einen zeitlich begrenzten Effekt auf die Qualität des Schätzers hat, der durch die Monotonieeigenschaft des Fehlers bei weiterer Verarbeitung der Daten an Bedeutung verliert. Die Dichtefunktionen der hier untersuchten Daten sind bis auf die Sprünge glatte Kurven, die sich mit Hilfe der Splines gut approximieren lassen. Daher reicht es bei diesen Daten lediglich die Größe der jeweiligen Blöcke zu variieren und nicht den zur Komprimierung verfügbaren Speicherplatz.

Ein etwas anderes Bild zeigen jedoch die Experimente mit Daten des Typs STR. Auf die Besonderheiten, die sich aus dem nicht glatten Verlauf dieser nicht erzeugten Testdaten ergeben, wird am Ende des Abschnitts 5.6.3 näher eingegangen.

5.6.3 Quantitative und qualitative Evaluation der Verfahren

Im Anhang C sind zu jedem Testdatensatz die bestimmten mittleren relativen Fehler zu jeweils einem Grad an Sortiertheit angegeben. Die jeweiligen Schätzer sind mit Hilfe der *Factory*-Methoden der Klasse `Aggregators` erzeugt und zu festen Zeitpunkten mit Hilfe der Testanfragen ausgewertet worden.

Abbildung 5.19 zeigt am Beispiel des $N(0,1)_{0.5}$-Datensatzes und der $Q_{0.02}^{N(0,1)}$-Anfragen die bestimmten mittleren relativen Fehler an sechs Zeitpunkten der Ver-

arbeitung. Deutlich ist die Konstanz der Fehler der speicherbasierten Verfahren zu erkennen. Die Fehler der adaptiven Verfahren fallen jedoch monoton in der Zeit. Bereits nach Verarbeitung der Hälfte der Daten des Datenstroms zeigt das Verfahren des adaptiven *Samplings* einen unerheblichen Fehler im Promille-Bereich.

Eine qualitative Auswertung der Tabellen in Anhang C zeigt einige Trends in den Ergebnissen der Experimente:

1. Deutlich sind die adaptiven den speicherbasierten Verfahren bei gleichem gegebenem Speicherplatz überlegen. Gerade bei geringen Speichergrößen treten zum Teil erhebliche Unterschiede in den mittleren Fehlern nach Verarbeitung des gesamten Datenstroms und einer arithmetischen Gewichtung auf. So zeigt zum Beispiel ein auf adaptivem Sampling beruhender Schätzer mit den Parametern $M = 100$ und $M_t = 100$ einen mittleren relativen Fehler zum Zeitpunkt $t = 1\,000\,000$ von 0.5425% auf $\mathrm{N}(0,1)_{0.5}$-Daten. Sein speicherbasiertes Gegenüber jedoch einen Fehler von ca. 38% bei Verwendung der gleichen $\mathrm{Q}_{0.01}^{\mathrm{N}(0,1)}$ Anfragen. Ein ähnliches Ergebnis zeigt sich bei $\mathrm{CP1}_{0.03}$-Daten zum gleichen Speicherplatz und gleicher Anfragengröße. Hier beträgt der Unterschied 1.12% zu ca. 60%. Das gleiche Bild zeigt sich auch bei den hybriden Methoden und den einfachen Kernschätzern. Jedoch sind die Unterschiede in der Regel bei einfachen Kernschätzern nicht so dramatisch wie bei den beiden anderen Verfahren.

2. Der Einfluss der verschiedenen Anfragengrößen zeigt sich bei den Experimenten wie erwartet. Je größer der Bereich der Fensteranfrage, desto geringer der Fehler. Dies gilt zu jedem Zeitpunkt, für jeden Datensatz und für jeden Schätzer. Lediglich am Anfang eines „fast“ sortierten Datenstroms können Abweichungen auftreten. Dies liegt jedoch darin begründet, dass eine Schätzung zu einem solchen Zeitpunkt für Anfragen, die den gesamten Datenbereich abdecken, unvorhersehbare Fehler produziert.

3. Auch der Grad der Sortiertheit beeinflusst die Schätzer weitestgehend wie erwartet. Auf teilsortierten oder fast sortierten Datenströmen können die Fehler sehr hoch sein, wenn der Schätzer nicht bereits „genug“ Daten verarbeitet hat. So sind die Fehler bei fast sortierten Daten des Typs $\mathrm{CP2}_{0.03}$ zum Zeitpunkt $t = 750\,000$ in der Regel jenseits der 50%, fallen aber dann schnell gegen Werte zum Teil unter 1%. Bei den Experimenten mit $\mathrm{CP1}_{0.35}$-Daten zeigen die Schätzer bereits nach Verarbeitung erheblich weniger Daten (oft ca. $t = 250\,000$) die gleiche Leistung mit Fehlern um ca. 50%. Dies zeigt sehr deutlich, wie wichtig eine Unabhängigkeit der Reihenfolge der Elemente in einem Datenstrom von der genutzten Schätzfunktion ist.

4. Doch die Reihenfolge der Elemente in einem Datenstrom hat keine signifikante Auswirkung auf den Fehler nach Verarbeitung aller Daten.

5. Deutlich ist die Abhängigkeit der Qualität des speicherbasierten *Samplings* vom verwendeten Speicher zu erkennen. Bei geringen Speichergrößen[54] verursacht das

[54] Zum Beispiel $M = 100$ und $M_t = 100$

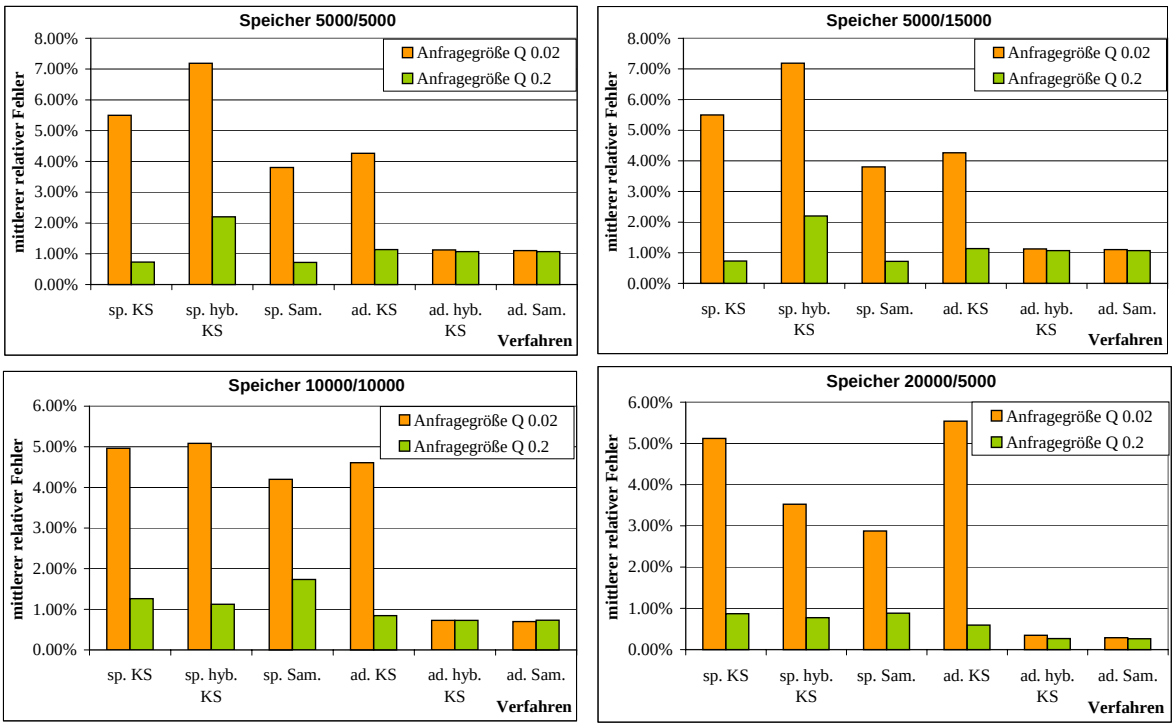

Abbildung 5.20: Mittlere relative Fehler der Schätzer auf $STR_{0.51}$-Daten zum Zeitpunkt $t = 1\,888\,012$ zu verschiedenen Speichergrößen

einfache *Sampling* erhebliche höhere Fehler wie einfache speicherbasierte Kernschätzer. Mit zunehmendem Speicher jedoch kann das Verfahren die Lücke schließen und bei CP1-Daten in Kombination mit großen Anfragen einfache Kernschätzer sogar überholen.

6. Die Leistungsfähigkeit des adaptiven hybriden Kernschätzers und des adaptiven *Samplings* sind in etwa gleich hoch. Beide Verfahren verursachen signifikant niedrigere Fehler im Vergleich zu allen anderen Verfahren. Dabei zeigt sich sogar eine leichte Überlegenheit des Verfahrens auf Basis der empirischen Stichprobenfunktion. Nicht zuletzt wegen der wenigeren kritischen Parameter im Vergleich zu den hybriden Verfahren ist es bestens geeignet, unüberwacht verwendet zu werden. Andererseits bieten die adaptiven hybriden Verfahren noch einiges Potenzial bei richtiger und effizienter Wahl der Parameter. Vor allem die Wahl der Bandbreiten für die verschiedenen Klassen eines hybriden Schätzers hat sich als problematisch erwiesen. Hier könnten Verbesserungen als erstes ansetzen.

7. Die Fehler der adaptiven Verfahren hängen offensichtlich stärker vom Grad der Komprimierung, d. h. vom verwendeten Speicherplatz zum Aufbau der Splines, ab, als von der genutzten Blockgröße.

Die Experimente mit den realen Daten des Typs STR zeigen in einigen Punkten deutliche Unterschiede zu den Ergebnissen mit den erzeugten Daten. Abbildung 5.20

zeigt exemplarisch die mittleren Fehler zu zwei Anfragengrößen ($Q_{0.02}^{STR}$ und $Q_{0.2}^{STR}$) der verschiedenen Schätzer nach Verarbeitung aller Daten ($t = 1\,888\,012$) des $STR_{0.51}$ Datensatzes. Dabei bezeichnet „sp. KS“ einen einfachen speicherbasierten gespiegelten Kernselektivitätsschätzer, „sp. hyb. KS“ einen speicherbasierten hybriden Kernselektivitätsschätzer und „sp. Sam.“ speicherbasiertes *Sampling*. Ein vorangestelltes „ad.“ markiert die adaptiven Gegenüber.

1. Einfache Kernschätzer zeigen erhebliche Schwächen bei einer Anwendung auf reale Daten. So verursacht ein adaptives Vorgehen in diesem Fall einen ähnlich hohen Fehler wie ein speicherbasiertes. Doch sowohl die hybriden als auch die auf *Sampling* basierenden Schätzer zeigen einen erheblichen Leistungszuwachs bei Verwendung des adaptiven Prinzips. Das Prinzip der Glättung ohne weitere Berücksichtigung besonderen Strukturen im Verlauf der Daten eignet sich somit prizipiell nicht für nicht glatte Datenverteilungen.

2. Im Gegensatz zu allen anderen Verfahren reagieren die beiden adaptiven Verfahren des hybriden Kernschätzers und des *Samplings* kaum auf eine Variation der Anfragengröße. Dies legt die Vermutung nahe, dass diese beiden Schätzer gleichmäßig auf dem gesamten Datenbereich gut arbeiten, und damit einzelne Anfragen über kleinen Bereichen mit großen relativen Fehlern kaum auftreten. Dies ist um so bemerkenswerter, als dass ein solcher Effekt bei Experimenten mit glatten Verteilungen nicht erkennbar war. Hier müssen weitere Untersuchungen mehr Klarheit schaffen.

3. Der Einfluss der Speichergröße auf speicherbasiertes *Sampling* entspricht dem bei Verwendung erzeugter Daten. Jedoch zeigt sich ein unerwartet hoher Unterschied im mittleren relativen Fehler bei Variation der Anfragengröße. Während für kleine Anfragen der Fehler im erwarteten Bereich liegt, ist die Qualität bei großen Anfragen besser als erwartet.

4. Die speicherbasierten Verfahren zeigen im Gesamten jedoch ein recht uneinheitliches Bild. So variieren die Fehler bei gleicher Speichergröße ($M = 5\,000/M_t = 15\,000$ und $M = 10\,000/M_t = 10\,000$) erheblich, was Abbildung 5.20 sehr deutlich zeigt.

5. Die mittleren Fehler der beiden dominanten adaptiven Verfahren auf Daten des Typs STR liegen deutlich über den mittleren Fehlern bei Anwendung auf Daten mit glatter Dichte. Trotzdem sind die mittleren relativen Fehler, die in einem Bereich um 1% liegen, unerwartet niedrig. Diese Verfahren sind damit, wie auch schon im Fall der Daten mit glatter Verteilung, den anderen in jedem Fall vorzuziehen. Doch auch hier gilt das oben bereits gesagte. Das adaptive hybride Verfahren hat mehr kritische Parameter die datenabhängig gewählt werden müssen im Vergleich zum adaptiven *Sampling*. Doch auch hier kann die Leistungsfähigkeit des adaptiven hybriden Kernselektivitätsschätzer bei „richtiger“ Wahl der Parameter sicherlich noch einmal gesteigert werden.

Eine Tabelle mit den Ergebnissen der Experimente über $STR_{0.51}$-Daten findet sich in Anhang C.4.

5.7 Diskussion des allgemeinen adaptiven Schätzers

Der Ansatz des allgemeinen adaptiven Schätzers im Kombination mit einem geeigneten Komprmierungsverfahren kann in dem hier betrachteten Modell sehr überzeugen. So sind nicht nur die Fehler der vorgestellten Verfahren erheblich geringer als die der speicherbasierten Gegenüber, sondern sie erlauben zusätzlich eine dynamische Anpassung an veränderte Speicherbedingungen. Im Falle einer arithmetischen Gewichtung können Einfüge- und Löschoperationen auf den Daten blöckeweise einfach unterstützt werden. Haben sich eine Zahl an Löschoperationen auf den Daten in Größe eines Blockes angesammelt, wird über diesen Werten ein Blockschätzer wie im Falle der adaptiven Verarbeitung aufgebaut. Diese Funktion wird nun mit Hilfe des aktuellen Gewichts von der aktuell gültigen Funktion abgezogen.

Auch im Falle nicht glatter Daten einer realen Verteilung produzieren die adaptiven Verfahren für viele Anwendungen Fehler in vernachlässigbarer Höhe. So erreicht ein vollständig aufgebauter adaptiver Schätzer auf Basis der empirischen Stichprobenfunktion (*Sampling*) einen mittleren Fehler von in der Regel unter einem Prozent bei beliebiger Anfragegröße. Das Handicap der langsamen Konvergenz des mit guten theoretischen Eigenschaften versehenen Schätzers kann somit weitestgehend eliminiert werden.

Doch wie erwartet ist auch bei einem adaptiven Vorgehen die Reihenfolge bzw. der Grad der Sortiertheit der Elemente in einem Datenstrom kritisch. So reicht ein Verarbeiten von 75 Prozent eines Datenstroms bei einem Grad der Sortiertheit, der einer normierten Inversionszahl von 0.35 entspricht, nicht aus, eine aussagekräftige Schätzfunktion aufzubauen.

Zerfällt der Datenstrom bereits in zusammengehörige Bereiche bzw. können die Abschnitte so gewählt werden, dass die Varianz innerhalb eines Blocks deutlich geringer im Vergleich zur Varianz der Grundgesamtheit ist, kann dies genutzt werden, um den Schätzer zu verbessern. Wie beim Ziehen einer Stichprobe aus einer geschichteten Grundgesamtheit (*stratified sampling*) führt das Kombinieren der Schätzer einzelner Bereiche mit kleiner[55] Varianz zu einer geringeren Varianz des Fehlers des Schätzers. Eine ausführliche Übersicht dazu findet sich in [Bar91].

[55] Relativ zur Varianz der Grundgesamtheit

Kapitel 6

Diskussion und Ausblick

Neue Verfahren im Bereich des *Data Mining* müssen sich neuen Anforderungen sowohl im Bezug auf Komplexität und Ressourcenabhängigkeit, als auch bezüglich ihrer verwendeten statistischen Modelle stellen. So reicht es schon lange nicht mehr bekannte Verfahren der mathematischen Statistik in Datenbanksystemen anzuwenden, ohne einen Blick auf ihre statistischen Modelle und die Interpretierbarkeit der Ergebnisse zu werfen. Vielmehr sind Anpassungen an die besonderen Gegebenheiten immer dann zwingend notwendig, wenn die Verarbeitung großer Datenmengen im Vordergrund steht.

Ein stimmiges und leicht zu interpretierendes mathematisches Modell und ein sehr leistungsfähiger Schätzer mit guten theoretischen Eigenschaften reichen in einem solchen Fall nicht aus. Vielmehr rücken andere begrenzende und bestimmende Faktoren in den Vordergrund. Gerade in Bereichen, in denen Daten *just in time*, d. h. direkt bei ihrer Entstehung, weiterverarbeitet werden, spielen der Faktor Zeit, der Faktor Skalierbarkeit und der Faktor Metainformationen eine mindestens ebenso wichtige Rolle. So darf ein Verfahren nicht länger zur Verarbeitung eines Datums benötigen wie die Zeit, die zwischen der „Ankunft“ zweier Elemente des Stroms liegt. Auch muss die Verarbeitung, und damit sowohl die Komplexität als auch der Speicherplatzbedarf, vollständig unabhängig von der Gesamtzahl der zu verarbeitenden Daten sein. Zu guter Letzt liegen oft keinerlei Metainformationen über den Datenstrom selbst vor. Das bedeutet, oftmals notwendige Größen wie der Grad der Sortiertheit, Extremalwerte oder die Anzahl der Elemente im Datenstrom sind nicht bekannt.

Ein ähnliches Problem stellt eine nicht triviale Analyse bereits gespeicherter Daten dar. Für extreme Datenmengen liefert das klassische *Black Box*-Prinzip unbefriedigende Ergebnisse. Lange Wartezeiten für falsch formulierte Anfragen oder nichts sagende Antworten sind ebenso wenig akzeptabel wie eine intensive Einarbeitung in verschiedenste Analysewerkzeuge zur Erzeugung trivialer Ergebnisse. Eine Ursache für die oftmals sehr langen Laufzeiten analysierender Anfragen ist die Tatsache, dass vor der eigentlichen Analysephase alle vorgeschalteten Operatoren ausgewertet werden und die Daten dann zwischengespeichert vorliegen. Auch hier muss der Ansatz weg von der Vorgabe gehen, dass zu untersuchende Daten bereits in geeigneter Form persistent vorliegen.

Damit stellt sich die Frage nach statistischen Verfahren, die über Datenströmen effizient arbeiten können.

Erste Ansätze in diese Richtung, jedoch ohne eine genaue Formulierung der Zielkriterien, finden sich schon recht früh in der Literatur. Doch erst Hellerstein [Hel96] und Hellerstein et al. [HHW97] formulieren Ziele und Probleme solcher Verfahren und zeigen erste Lösungen für klassische statistische Kennzahlen. Dieser Ansatz ist jedoch auf die wohlbekannnten Aggregate (Minimum, Maximum, Mittelwert, Anzahl und Varianz) beschränkt. Eine Erweiterung des Begriffs der *Online Aggregation* auf Verfahren der schließenden Statistik ist jedoch möglich. Aus einer solch veränderten Betrachtungsweise der Verarbeitung von Aggregaten in Datenbanksystemen resultieren zum einen neue Chancen und Möglichkeiten, zum anderen aber auch neue Restriktionen und Probleme. So werden in dieser Arbeit die notwendigen Kriterien für Verfahren zur *Online Aggregation* formuliert und zusätzliche wünschenswerte Eigenschaften der Verfahren angegeben.

Doch bevor Verfahren der schließenden Statistik integraler Bestandteil heutiger Datenbanksysteme werden können, müssen die zwei am meisten beteiligten Disziplinen zu einer einheitlichen Begriffsbildung finden. So sei hier nur das Beispiel der für statistische Modelle essentiellen Messniveaus der Daten genannt. Für viele Verfahren der schließenden Statistik muss die Messskala, der die zu untersuchenden Daten angehören, bekannt sein bzw. sind die Ergebnisse nur bei Kenntnis der Skala aussagekräftig. So unterscheiden sich verschiedene Verfahren auf Basis der in der Datenbankwelt sehr populären Histogramme trotz gleicher Vorgehensweise deutlich in der Interpretation der Ergebnisse, wenn die Daten unterschiedlichen Skalen zuzuordnen sind. Diese Unterscheidung wird jedoch im Kontext von Datenbanksystemen kaum getroffen. Durch die Ansiedelung der vorliegenden Arbeit im Grenzgebiet der Statistik und der Datenbanklehre stellt sich auch hier die Frage nach einer gemeinsamen Begriffsbildung. Ein Teil dieser Arbeit widmet sich diesem Problem.

Zu den wichtigsten Eigenschaften der zur *Online Aggregation* geeigneten Verfahren gehören die schrittweise Berechenbarkeit über einem Datenstrom und ein datenunabhängiger Speicherplatzbedarf. Diese Anforderungen bringt kaum ein Verfahren der Statistik ohne spezielle Anpassungen mit sich. Auch zu den bekannten und wichtigen kernbasierten Verfahren der nichtparametrischen Statistik existieren bisher keine Betrachtungen zu diesem Problem. Kernschätzer haben sich zwar in letzter Zeit durchaus in Bereichen wie *Clustering* oder Selektivitätsschätzung in Datenbanksystemen etabliert, ohne jedoch das Problem stark beschränkter Ressourcen zu fokussieren. So werden in dieser Arbeit zwei allgemeine Vorgehensweisen entwickelt wie beliebige glättende Schätzfunktionen zu gegebenem Speicherplatzbedarf auf Datenströmen zur *Online Aggregation* eingesetzt werden können. Dabei kommt zum einen ein spezielles Verfahren zum Ziehen einer *i.i.d.* Stichprobe über Grundgesamtheiten mit unbekannter Größe und zum anderen ein adaptives Vorgehen in Kombination mit verlustbehafteter Komprimierung zum Einsatz.

Der bekannte speicherbasierte Ansatz zum Ziehen einer Stichprobe über Datenmengen unbekannter Größe ist direkt auf des Problem des Ziehens einer Sichprobe über ei-

nem Datenstrom anwendbar. So kann die Position eines Elements eines Datenstroms in der Reihenfolge des Auftretens der Elemente als diskreter Zeitpunkt betrachtet werden. Zu einem beliebigen aber festen Zeitpunkt t kann nun mit Hilfe des speicherbasierten Ansatzes eine Stichprobe der vor diesem Zeitpunkt gelieferten Daten ohne Kenntnis des Zeitpunktes t gezogen werden. Für einen Datenstrom bedeutet dies, dass der Speicher zu jedem Zeitpunkt eine gültige *i.i.d.* Stichprobe enthält.

Der größte Vorteil dieses speicherbasierten Ansatzes ist vor allem die universelle Kombinierbarkeit mit beliebigen induktiven Statistiken, deren Komplexität genügend klein ist[1]. Dieser Ansatz ist in der vorliegenden Arbeit anhand verschiedener kernbasierter Verfahren validiert worden. Neben einem einfachen Kerndichteschätzer und einer modifizierten Variante zur Verringerung der oft auftretenden Randeffekte, ist der Ansatz auch mit einem hybriden Verfahren, welches die Erkennung von Sprungstellen mit Histogrammen und Kernschätzern kombiniert, experimentell ausgewertet worden. Zudem wurde in praxisnahen Experimenten mit Hilfe eines einfachen und eines hybriden Kernselektivitätsschätzers die Leistungsfähigkeit des speicherbasierten Ansatzes gezeigt. Dieses universell einsetzbare Prinzip birgt jedoch einige Nachteile. So wird jedes Element eines Datenstroms „betrachtet“, jedoch nur eine kleine Teilmenge zur Berechnung des Schätzers herangezogen. Zudem kann nur ungenügend auf veränderte Speicherplatzbedingungen eingegangen werden.

Das allgemeine adaptive Prinzip, in Kombination mit einer geeigneten Komprimierung, stellt eine Lösung zum Problem des Aufbaus einer statistischen Schätzfunktion über Daten auf einer Intervallskala, die mittels eines Datenstroms geliefert werden, zu vorgegebenen beschränkten Ressourcen zur Verfügung. Zudem hat sich die verlustbehaftete Komprimierung mit Hilfe kubischer Splines für die in dieser Arbeit betrachteten Verfahren als sehr effizient gezeigt. Auch hier hat eine Evaluation mit verschiedenen kernbasierten Verfahren die praktische Verwendungsfähigkeit gezeigt. Im Vergleich zu einem Referenzverfahren zur Selektivitätsschätzung in Datenbanken kann das adaptive Prinzip bei der Verwendung über einem Datenstrom absolut überzeugen.

So zeigen die betrachteten Verfahren sowohl gute theoretische Eigenschaften als auch geringe Fehler in praxisnahen Experimenten. Doch ist die Aussagefähigkeit solcher unter Laborbedingungen durchgeführter Tests oft stark eingeschränkt. Dies liegt häufig an den kaum nachvollziehbaren Testparametern und den selten zur Verfügung stehenden Referenzimplementierungen.

Aus diesem Grund stehen alle in der vorliegenden Arbeit entwickelten und zu Tests herangezogenen Vefahren vollständig dokumentiert in der plattformunabhängigen und frei verfügbaren[2] Java-Bibliothek XXL zur Verfügung. Zudem sind die Testparameter und Testaufrufe vollständig und ausführlich beschrieben, so dass alle Experimente transparent nachvollzogen werden können. Auch die Verfahren und Implementierungen zur Erzeugung der verwendeten Testdatensätze sind beschrieben und dokumentiert und stehen in persistenter Form frei zur Verfügung.

Neben den technischen und mathematischen Aspekten der untersuchten Verfahren

[1] Subquadratische Laufzeiten sind in der Regel notwendig.

[2] XXL ist unter der GLPL (*GNU lesser public license*) verfügbar (Siehe [XXL]).

hat diese Arbeit vor allem eines gezeigt. Untersuchungen im Gebiet des *Data Mining* und die Entwicklung neuer Verfahren sind ohne Überblick über verschiedene verwandte und angrenzende Gebiete wie die der Datenbanklehre, der mathematischen Statistik oder der KI (Künstliche Intelligenz) kaum zu leisten. Durch die extrem hohe Zahl an begrenzenden Faktoren und Parametern und die oft ungenügend formulierten oder formulierbaren Zielkriterien kann nur ein Zusammenspiel verschiedener Sichten und Ansätze und damit eine Zusammenarbeit zwischen den Disziplinen zum Erfolg führen. Eine Fokussierung auf eine einzige Disziplin kann in diesem Kontext nur kontraproduktiv sein.

So sind auch viele Fragen, die diese Arbeit offen lassen muss, durch unterschiedliche Gebiete motiviert. Die aus der Bild- und Videoverarbeitung bekannten Wavelet- und Kosinustransformationen werden in letzter Zeit zunehmend zur Datenkompression und Verteilungsapproximation auf dem Gebiet der Datenbanklehre eingesetzt. Hier stellt sich die Frage, ob diese Techniken auch zur Komprimierung induktiver Statistiken auf Datenströmen eingesetzt werden können und die Voraussetzungen zur *Online Aggregation* erfüllen. Hier ist besonders die Eignung dieser Techniken im bi- oder multivaraten Fall interessant, der noch immer große Herausforderungen an Verfahren und Systeme stellt. Auch eine mehrdimensionale Splineinterpolation wäre vor diesem Hintergrund zu betrachten. Dabei muss immer das im hochdimensionalen Fall noch kritischere Problem des beschränkten Speicherplatzes im Vordergrund stehen.

So bedarf auch die Wahl datenabhängiger Gewichtungsstrategien zur Erzielung benutzergesteuerter Effekte im Falle der allgemeinen adaptiven Schätzer einer näheren Untersuchung. Sowohl zeit- als auch verteilungsabhängige Stategien sind auf diese Weise realisierbar. Unter diesen Aspekt fällt auch eine Modifikation der speicherbasierten Verfahren, die eine Unterstützung zum Ziehen gewichteter bzw. dichteabhängiger Stichproben bietet. Einige Veröffentlichungen zu diesem Thema sind für verschiedene Konferenzen des Jahres angekündigt.

Auch eine direkte Komprimierung der Stichprobe bzw. der adaptiv verarbeiteten Blöcke wäre als Alternative zur Komprimierung der Schätzfunktion denkbar. Im Falle kernbasierter Verfahren können zum Beispiel Stichprobenwerte mit ähnlicher oder identischer Masse unter genauer Angabe des Fehlers zu einem Repräsentanten zusammengefasst und damit komprimiert werden. Eine Kombination beider Techniken könnte einen zusätzlichen Effekt bewirken.

Eine interessante Frage stellt sich bei genauerer Betrachtung der Ergebnisse der praxisnahen Experimente mit hybriden Selektivitätsschätzern auf Basis der auch in Randnähe kosistenten gespiegelten Kernschätzern. Offensichtlich reicht eine einfache Bandbreitenbestimmung aufgrund bekannter Regeln und Verfahren nicht aus, die besonderen Erfordernisse die der Einsatz hybrider Methoden mit sich bringt zu erfüllen. So wird der Fehler dieser Verfahren im Vergleich zu einfachen Kernschätzern noch stärker durch die Wahl der Bandbreiten für die einzelnen Klassen bestimmt. Einsetzungregeln, die bei einfachen Kerndichte- und Kernselektivitätsschätzern in praktischen Tests[3] die besten Resultate erzielt haben, sind auf Grund ihrer hohen Komplexität in

[3] vgl. [Kor99]

dem hier betrachteten Modell nicht einsetzbar. Auch alle Regeln zur Bandbreitenbestimmung auf Basis einer Kreuvalidierung sind in dieser Hinsicht problematisch.

Die neben den Dichte- und Selektivitätsschätzern gerade im Rahmen der hybriden Methoden verwendeten Verfahren zur Erkennung von Sprungstellen in der Dichte einer Zufallsvariablen, bedürfen einer genaueren Betrachtung und ausführlicherer Tests über realen Daten. Die Ergebnisse der Experimente über erzeugten Daten mit Sprungstellen in der Wahrscheinlichkeitsdichte lassen jedoch das hohe Potenzial dieser Verfahren erkennen.

Danksagung

Mein Dank gilt vor allem meiner Freundin und langjährigen Partnerin Frau Katrin Eigel, die mich bei der Erstellung dieser Arbeit nicht nur durch ihre frischen Ideen immer wieder daran erinnert hat, dass es neben der Forschung und der Universität auch andere wichtige Dinge im Leben gibt. Des Weiteren gilt mein Dank selbstverständlich meinem Doktorvater Herrrn Prof. Dr. Bernhard Seeger, der in den Jahren meiner Tätigkeit als Assistent an der Universität mehr ein Freund denn ein Vorgesetzter war. Auch Herrn Martin Schneider und besonders Herrn Jochen van den Bercken möchte ich für die schönen und lehrreichen Stunden („Hallo Jochen, ich hab' da mal eine Frage ...“) danken. Und natürlich danke ich all jenen, die mich in irgendeiner erdenklichen Form unterstützt haben und mir vor allem in den letzten Wochen der Fertigstellung dieser Arbeit Nachsicht haben zukommen lassen.

Anhang A

Testdaten

Damit eine qualitative Aussage der Güte der in dieser Arbeit verwendeten Methoden möglich ist, sind zum einen Daten der realen Welt und zum anderen erzeugte Daten für Tests verwendet worden. Über die realen Daten sind lediglich statistische Kennzahlen bekannt. Die Verteilung wird jedoch als stetig angenommen. Damit kann der zur Verfügung stehende Datensatz als Repräsentation einer Stichprobe der tatsächlichen unbekannten Verteilung angesehen werden. Die künstlichen Datensätze sind jedoch so erzeugt worden, dass neben den üblichen Kennzahlen wie Mittelwert und Varianz auch die Dichtefunktion der Verteilung der Daten bekannt ist.

A.1 Erzeugte Datensätze

Bedingt durch die unterschiedlichen Zielsetzungen der in dieser Arbeit vorgestellten Verfahren kommen zwei verschiedene Arten von erzeugten Testdaten zum Einsatz. Zum einen werden Daten verwendet, deren Dichte einer hinlänglich bekannten Familie von Verteilungsfunktionen angehört. Diese Daten eignen sich besonders, um die theoretische Leistungsfähigkeit von glättenden Schätzern zu untermauern. Für die Verfahren zur Sprungstellenerkennung werden jedoch Verteilungen benötigt, deren Dichten Sprünge an bekannten Stellen in bekannter Höhe aufweisen.

A.1.1 Klassische Verteilungen

Um den Einfluss unterschiedlicher Verfahrens- und Modellparameter auf die Leistungsfähigkeit der Schätzer untersuchen zu können, muss die Verteilung der Testdatensätze wohlbekannt sein. Die wohl am Besten untersuchte Verteilungsfamile in der mathematischen Statistik ist die der Normalverteilungen. Daher kommt vor allem zur Untersuchung der auftretenden Effekte bei Variation unterschiedlicher Parameter die Standardnormalverteilung zum Einsatz[1].

Der in dieser Arbeit benötigte standardnormalverteilte Testdatensatz ist mit Hilfe

[1] vgl. Abschnitt 5.5

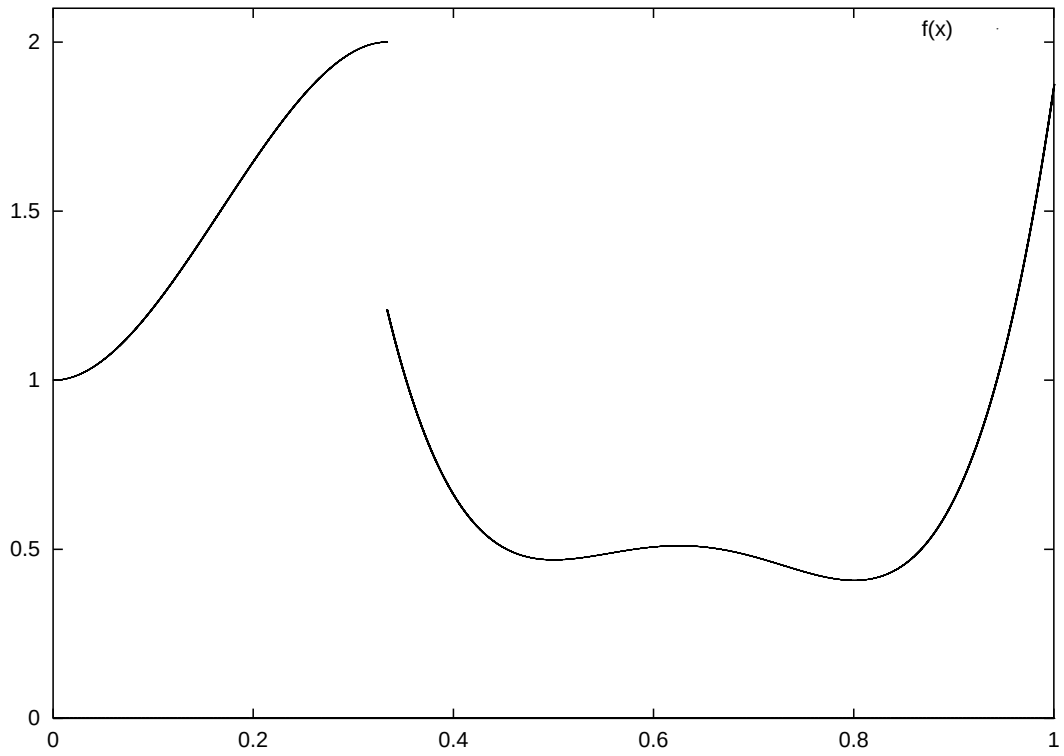

Abbildung A.1: Dichte der Daten vom Typ CP1

der Java-Bibliothek *Colt*[2] erzeugt worden. Eine *Fatory*-Methode zur Erzeugung eines *Cursors*, der solche Zufallszahlen als Datenstrom zur Verfügung stellt, findet sich in der Klasse `RandomNumbers` im *Package* `connectivity.colt` in den Zusatzpaketen der Bibliothek XXL .

A.1.2 Daten mit Sprungstellen

Die zwei erzeugten Datensätze, deren Dichtefunktionen eine bzw. zwei Sprungstellen besitzen, sind mit Hilfe der Ablehnungsmethode[3] erzeugt worden. Dabei kam in beiden Fällen die Vergleichsdichte

$$g(x) = \begin{cases} \frac{1}{2} + x & \text{für } 0 \leq x \leq 1 \\ 0 & \text{sonst} \end{cases} \tag{A.1}$$

zum Einsatz. Für die Funktion $g(x)$ gilt $g(x) \geq 0$ für alle $x \in \mathbb{R}$ und $\int_{-\infty}^{\infty} g(x)\mathrm{dx} = 1$. Damit ist $g(x)$ eine Dichte. Für die zugehörige Verteilungsfunktion

$$G(y) = \int_{-\infty}^{y} g(x)\mathrm{dx} = \int_{0}^{y} g(x)\mathrm{dx} = \frac{1}{2}(y + y^2) \tag{A.2}$$

[2] vgl. [Col]
[3] vgl. Abschnitt 5.4.3.1

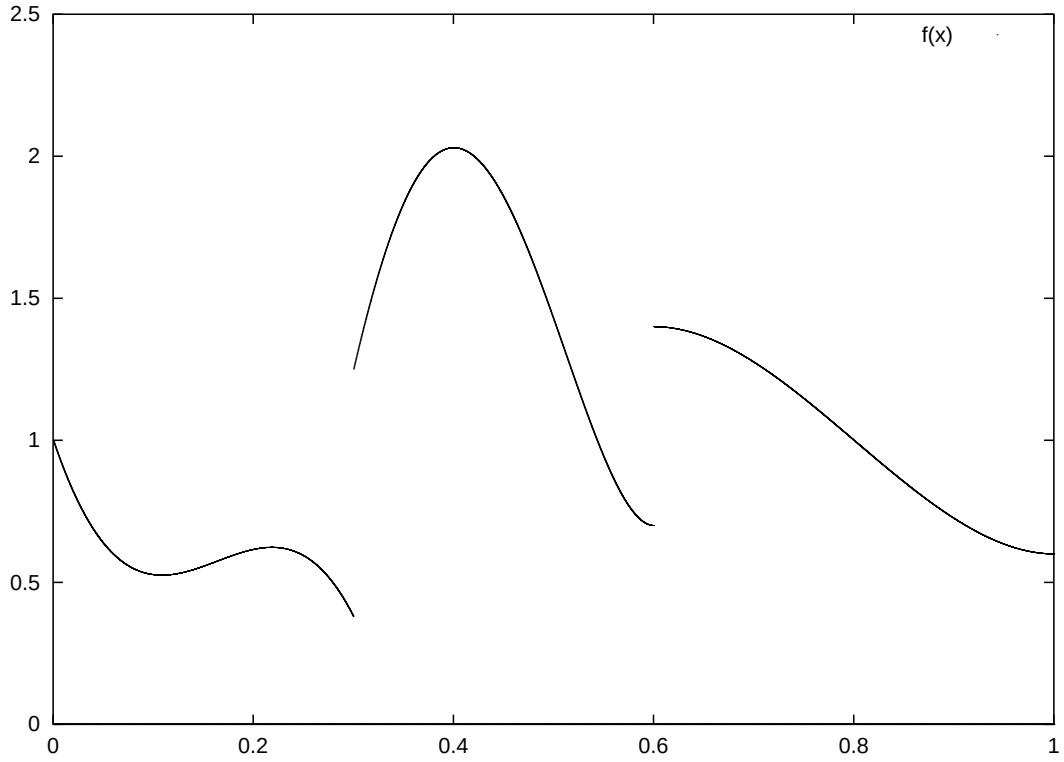

Abbildung A.2: Dichte der Daten vom Typ CP2

existiert auf $[0,1]$ die eindeutige positive Inverse

$$G^{-1}(y) = \frac{1}{2}(\sqrt{8y+1} - 1).$$

Mit Hilfe der Inversionsmethode und einem Zufallszahlengenerator, der auf $[0,1]$ stetig gleichverteilte Zufallszahlen liefert, können somit gemäß Formel (A.2) verteilte Zahlen erzeugt werden.

Ein Datensatz folgt einer Verteilung, deren Dichte genau einen Sprung in $x = \frac{1}{3}$ mit der Höhe $\Delta = \frac{2}{3}$ besitzt. Die Verteilung dieses mit CP1 bezeichneten Datensatzes besitzt die Dichte

$$f(x) = \begin{cases} 1 + 27x^2 - 54x^3 & \text{für } 0 \le x \le \frac{1}{3} \\ \frac{169}{8} - 135x + \frac{2619}{8}x^2 - \frac{693}{2}x^3 + 135x^4 & \text{für } \frac{1}{3} < x \le 1. \end{cases}$$

Für eine Konstante $c = 2.4$ erfüllt $g(x)$ aus Formel (A.1) die zur Anwendung der Ablehnungsmethode geforderte Ungleichung (5.24).

Die Dichte des zweiten Datensatzes CP2 besitzt zwei Sprünge an den Stellen $x = 0.3$ und $x = 0.6$ mit den Höhen $\Delta_{0.3} = \frac{1}{2}$ und $\Delta_{0.6} = \frac{7}{10}$. Für die Dichte gilt

$$f(x) = \begin{cases} 1 - \frac{35}{3}x + \frac{725}{9}x^2 - \frac{4000}{27}x^3 & \text{für } 0 \le x \le \frac{3}{10} \\ -\frac{546979}{40399150} - \frac{653372272}{12119745}x + \frac{2949790460}{7271847}x^2 - \frac{6469685560}{7271847}x^3 + \frac{5500}{9}x^4 & \text{für } \frac{3}{10} < x < \frac{6}{10} \\ -\frac{47}{5} + 45x - 60x^2 + 25x^3 & \text{für } \frac{6}{10} \le x \le 1. \end{cases}$$

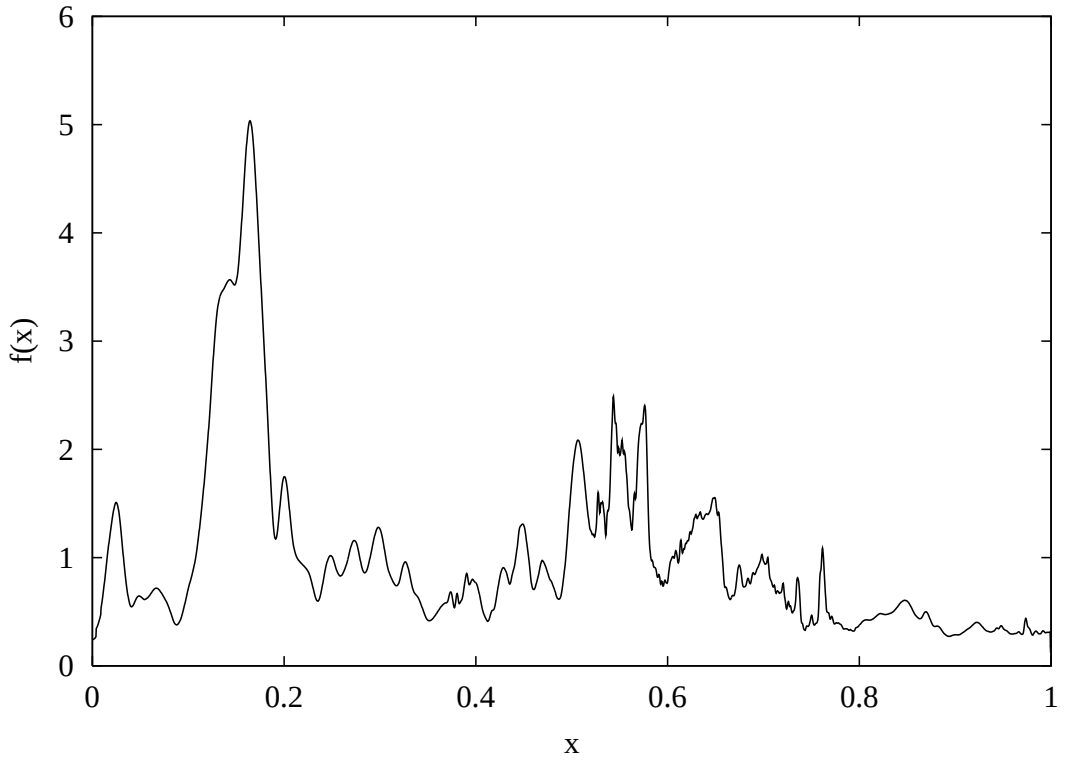

Abbildung A.3: Geschätzte Dichte der Daten vom Typ STR

Zur Anwendung der Ablehnungsmethode mit der Vergleichsdichte aus Formel (A.1) ist die Konstante $c = \frac{7}{3}$ gewählt worden.

A.2 Reale Datensätze

Für die praxisnahen Experimente ist ein Datensatz der *TIGER/LINE95-Files* verwendet worden. Die Originaldaten stellen die Koordinaten der minimal umgebenden Rechtecke (MUR) aller Straßen des Bundesstaates Kalifornien der Vereinigten Staaten von Amerika dar. Von diesen Koordinatenpaaren ist die linke untere (y1) Koordinate ausgewählt und mittels einer einfachen linearen Transformation

$$x' = \frac{x - min}{max - min}$$

auf das Intervall $[0, 1]$ transformiert und normiert worden. Daten dieses Typs werden in dieser Arbeit mit STR abgekürzt.

Im Gegensatz zu den erzeugten Daten sind die Experimente mit diesem Testdatensatz mit vier unterschiedlichen Graden an Sortiertheit der Datenstöme durchgeführt worden. Dabei bezeichnen $\mathrm{STR}_{0.03}$, $\mathrm{STR}_{0.35}$ und $\mathrm{STR}_{0.5}$ die Datenströme mit den entsprechenden normierten Inversionen. Der Datensatz STR ist jedoch in der Reihenfolge

Datensatz	Anzahl	Minimum	Maximum	Mittelwert	Varianz
N(0,1)	1 Mio.	≈ -4.662	≈ 4.715	≈ 0.00174	≈ 1.0
CP1	1 Mio.	$\approx 5.2E-7$	≈ 0.9999	≈ 0.4251	≈ 0.09138
CP2	1 Mio.	$\approx 5.2E-7$	≈ 0.9999	≈ 0.5069	≈ 0.06035
Str	1888012	0.0	1.0	≈ 0.3970	≈ 0.06439

Tabelle A.1: Statistische Eigenschaften der Testdatensätze

der Elemente nicht bearbeitet worden und entspricht damit der ursprünglichen „natürlichen“ Reihenfolge der Daten.

Abbildung A.3 zeigt eine adaptiv berechnete Dichtefunktion der STR Daten mittels eines gespiegelten Kerndichteschätzers[4].

A.3 Statistische Eigenschaften der Testdaten

In Tabelle A.1 sind alle für Tests verwendeten Datensätze mit ihren wichtigsten statistischen Eigenschaften dargestellt. Die Werte für das Minimum, das Maximum, den Mittelwert, die Varianz und die Anzahl der Daten sind mit Hilfe der entsprechenden Aggregatsfunktionen[5] und einem Aggregator in einem einzigen Datendurchlauf berechnet worden. Das Bestimmen der Anzahl der Inversionen[6] hat hingegen quadratischen Aufwand. Die in Tabelle A.1 angegebenen Werte gelten selbstverständlich sowohl im Falle der erzeugten als auch der realen Daten für alle teilsortierten Datensätze des jeweiligen Typs.

A.4 Teilsortierte Datensätze

In Tabelle A.2 sind die Datensätze und der Grad ihrer jeweiligen aufsteigenden Sortierung angegeben. Die verwendeten Maße sind zum einen die Zahl der Inversionen im Datensatz als ein globales Maß der Sortiertheit, zum anderen die Zahl der aufsteigenden sortierten Teilfolgen als ein Maß der lokalen Sortiertheit[7]. Des Weiteren ist die normierte Zahl der Inversionen angegeben, die auch zur Kennzeichnung der Datenströme mit unterschiedlicher Sortierung verwendet wird. Die Ausgangsdatensätze sind der Vollständigkeit halber noch einmal mitangegeben.

Für einen Datensatz der Größe n mit einer Inversionszahl I_c kann eine Permutation mit einer Inversionszahl I_t erzeugt werden mit Hilfe des folgenden Algorithmus:

Algorithmus A.1 *Erzeuge* `randU` *gleichverteilte diskrete Zufallszahlen* u *mit* $1 \leq u \leq n$. *Gegeben seien* I_c *die aktuelle Zahl der Inversionen in einem Datensatz*

[4] Dabei ist der Biweightkern, die Normalskalierungsregel, und ein Speicher der Größe 94400 verwendet worden.

[5] vgl. Abschnitt 2.5.2

[6] vgl. Abschnitt 5.4.3.2 und Tabelle A.2

[7] vglÄbschnitt 5.4.3.2

Datensatz	Normierte Inversionen	Inversionen	Runs
$N(0,1)_{0.5}/N(0,1)$	≈ 0.5	249 713 078 234	500 225
$N(0,1)_{0.35}$	≈ 0.35	174 999 825 004	499 945
$N(0,1)_{0.03}$	≈ 0.03	14 999 985 006	44 417
$CP1_{0.5}/CP1$	≈ 0.5	249 949 796 063	500 215
$CP1_{0.35}$	≈ 0.35	174 999 825 018	499 669
$CP1_{0.03}$	≈ 0.03	14 999 985 012	44 416
$CP2_{0.5}/CP2$	≈ 0.5	249 943 098 543	500 135
$CP2_{0.35}$	≈ 0.35	174 999 825 009	499 987
$CP2_{0.03}$	≈ 0.03	14 999 985 014	44 416
$Str_{0.51}/Str$	≈ 0.5140	916 147 768 964	1 027 526
$Str_{0.5}$	≈ 0.5	891 146 856 017	1 017 911
$Str_{0.35}$	≈ 0.35	623 802 799 207	957 913
$Str_{0.03}$	≈ 0.03	53 468 811 347	83 897

Tabelle A.2: Maße für den Grad der Teilsortiertheit der Testdaten

$X = \{x_1, \ldots, x_n\}$ und I_t die gewünschte Anzahl von Inversionen. Dann liefert der folgende Algorithmus eine Permutation des Datensatzes mit der gegebenen Zahl von Inversionen.

```
 1     WHILE( Ic != It)
 2         LET u1 = randU.getNext()
 3         LET u2 = randU.getNext()
 4         u1 = min( u1, u2) AND u2 = max( u1, u2)
 5         LET swapit = FALSE
 6         IF ( Ic < It)
 7             IF ( x[u1] < x[u2])
 8                 swapit = TRUE
 9         ELSE
10             IF ( x[u1] > x[u2])
11                 swapit = TRUE
12         IF ( swapit )
13             LET delta_inv = COUNT_INV( u1, u2)
14             IF ( ABS( Ic - It ) > ABS( Ic + delta_inv - It )
15                 Ic = Ic + delta_inv
16                 SWAP( u1, u2)
```

Das Unterprogram `COUNT_INV( u1, u2)` *folgt dabei*

```
 1     LET t = 0
 2     IF ( x[u1] < x[u2]) t++
 3     ELSE t--
 4     FOR i = u1+1 to u2-1
 5         IF x[i] > x[u1] t++
 6         IF x[i] < x[u1] t--
 7         IF x[i] > x[u2] t--
```

```
8          IF x[i] < x[u1] t++
9      RETURN t
```

das für die Elemente zwischen den beiden zu tauschenden Werten die Zahl der Inversionen anpasst.

Die Bedingung in Zeile `14` stellt sicher, dass im Falle eines Tausches die aktuelle Zahl der Inversionen auch tatsächlich näher am Ziel als die bisherige Zahl der Inversionen liegt, damit unter Umständen nicht zu weit „über das Ziel hinausgeschossen wird".

Dieses Verfahren hat den Nachteil, dass durch die genaue Vorgabe der gewünschten Zahl der Inversionen und der zufälligen Auswahl der zu tauschenden Positionen die Laufzeit unter Umständen beliebig groß werden kann. Abhilfe schafft hier eine Abweichung der Größe $\epsilon > 0$ um das Ziel I_t zuzulassen. Damit ändert sich die Abbruchbedingung in Zeile `1` auf

```
1      WHILE( ! (Ic IN [It-epsilon , It+epsilon]))
```

mit einem vernachlässigbaren Einfluss auf das Maß der normierten Inversionen.

Anhang B

Algorithmen

Im Folgenden werden die in dieser Arbeit verwendeten Algorithmen zur Berechnung der induktiven Statistiken aufgeführt.

B.1 Dichteschätzer

Ein Kerndichteschätzer nach Formel (4.4) kann einfach mittels folgendem Algorithmus implementiert werden.

Algorithmus B.1 *Seien K eine Kernfunktion, $X = X_1, \ldots, X_n$ eine nicht notwendigerweise sortierte i.i.d. Stichprobe und $h > 0$ eine Bandbreite. Dann kann mit folgendem Algorithmus die Dichte f der unbekannten Verteilung F in $x \in \mathbb{R}$ geschätzt werden.*

```
1    LET r = 0
2    FOR EACH X[i] IN X
3        r = r + K( ( x - X[i] ) / h )
4    r = r / ( h * n)
5    RETURN r
```

Um eine induktive Statistik als speicherbasierten Schätzer verwenden zu können, darf die Stichprobe bzw. die Reihenfolge der Stichprobenelemente nicht verändert werden. Der obige Algorithmus B.1 erfüllt diese Forderung, „bezahlt“ dies aber mit einer Komplexität $\Theta(n)$.

Lässt man diese Forderung fallen[1], so kann die Komplexität des Algorithmus mit einfachen Mitteln auf $\Theta(\log n)$ reduziert werden.

Algorithmus B.2 *Seien K eine auf $[-1, 1]$ beschränkte Kernfunktion[2], X eine sortierte i.i.d. Stichprobe mit $X1 \leq X_2 \leq \ldots \leq X_n$ und $h > 0$ eine Bandbreite. Dann kann mit folgendem Algorithmus die Dichte f der unbekannten Verteilung F in $x \in \mathbb{R}$ geschätzt werden.*

[1] Dies kann z. B. durch ein Duplizieren der Stichprobe erreicht werden.

[2] Die verwendete Kernfunktion muss nicht notwendigerweise auf [-1,1] beschränkt sein. Es muss lediglich gefordert werden, dass der positive Träger der Funktion beschränkt ist.

```
1    LET r = 0
2    LET i1 = INSERTION_INDEX( x - h , X)
3    LET i2 = INSERTION_INDEX( x + h , X)
4    FOR EACH X[i] WITH i1 <= i <= i2
5        r = r + K( ( x - X[i] ) / h )
6    r = r / ( h * n )
7    RETURN r
```

Das Unterprogramm `INSERTION_INDEX( p , X)` gibt den Index des Arrays `X` zurück, an dem der Wert `p` eingefügt würde. Dies kann einfach und effizient mittels binärer Suche auf dem sortierten Array geschehen. Die Komplexität einer binären Suche auf einem sortierten Array ist $\Theta(\log n)$. Ist die Einfügeposition des Wertes im Array nicht eindeutig[3], so kann eine beliebige gültige Position zurückgegeben werden. Die Sortierung des Arrays hat den einmaligen Aufwand $\Theta(n \log n)$. Damit lohnt sich diese Modifikation bereits beim zweiten Aufruf des Algorithmus.

Da diese Optimierung für jedes der folgenden Verfahren verwendet werden kann, werden nur noch die auf Sortierung basierenden Verfahren angegeben. Eine Vereinfachung auf den nativen Ansatz ist leicht möglich.

Der auf Spiegelung basierende Kerndichteschätzer nach Formel (4.20) kann auf folgende Weise implementiert werden.

Algorithmus B.3 *Seien K eine auf $[-1, 1]$ beschränkte Kernfunktion, X eine sortierte i.i.d. Stichprobe mit $X1 \leq X_2 \leq \ldots \leq X_n$ und $h > 0$ eine Bandbreite. Dann kann mit folgendem Algorithmus die Dichte f der unbekannten auf $[a, b]$ beschränkten Verteilung F in $x \in \mathbb{R}$ geschätzt werden.*

```
1    IF ( x < a) OR ( x > b) RETURN 0
2    ELSE
3        LET r = 0
4        LET i1 = INSERTION_INDEX( x - h , X)
5        LET i2 = INSERTION_INDEX( x + h , X)
6        FOR EACH X[i] WITH i1 <= i <= i2
7            r = r + K( ( x - X[i] ) /h )
8            IF ( Xi < a + h )
9                LET Xa = 2 * a - X[i]
10               r = r + K( ( x - Xa ) /h )
11           IF ( Xi > b - h )
12               LET Xb = 2 * b - X[i]
13               r = r + K( ( x - Xb ) /h )
14       r = r / ( h * n )
15       RETURN r
```

Das Verfahren des hybriden Kerndichteschätzers setzt eine Einteilung des Datenraums in $k + 1$ Klassen mit den Grenzen $(c_i, c_{i+1}]$ und die Zuordnung der Stichprobenwerte X_i zu den Klassen voraus. Dies kann mit einem Aufwand von $\Theta(n \log n)$

[3] Dies ist dann möglich, wenn Duplikate im Array vorkommen.

geschehen[4]. Nun können $k+1$ Dichteschätzer auf Basis der einer Klasse zugeordneten Stichprobenwerte etabliert und kombiniert werden.

Algorithmus B.4 *Seien* `c[i]` *mit* `c[0]`$= a$ *und* `c[k+1]`$= b$ *ein aufsteigend sortiertes Array mit den Klassengrenzen,* `f[i]` *für* $i = 0, \ldots, k$ *die zu den Klassen gehörenden Dichteschätzer und* `n[i]` *die Zahl der der Klasse* i *zugeordneten Stichprobenwerte. Dann kann ein hybrider Dichteschätzer mittels folgendem Algorithmus implementiert werden.*

```
1    IF ( x < c[0]) OR ( x > c[k+1]) RETURN 0
2    ELSE
3        LET r = 0
4        LET i1 = INSERTION_INDEX( x , c) -1
6        RETURN n[i] * f[i1](x) / n
```

Soll der hybride Dichteschätzer in seinen Rändern unbeschränkt sein, so kann einfach $a = -\infty$ *und* $b = +\infty$ *gesetzt werden.*

B.2 Selektivitätsschätzer

Zur Implementierung eines einfachen kernbasierten Selektivitätsschätzers wird die Stammfunktion des verwendeten Kerns benötigt. Der folgende Algorithmus verwendet einen auf $[-1, 1]$ beschränkten Kern und setzt eine sortierte Stichprobe `X[i]` voraus.

Algorithmus B.5 *Sei* `K.primitive(x)` *die Stammfunktion eines auf* $-1, 1]$ *beschränkten Kerns,* `X[1]`$\leq \ldots \leq$`X[n]` *eine aufsteigend sortierte i.i.d. Stichprobe und* $h > 0$ *eine Bandbreite. Dann liefert der folgende Algorithmus eine geschätzte Selektivität zu einer Bereichsanfrage* $Q([d, e])$.

```
1    LET r = 0
2    LET i1 = INSERTION_INDEX( d - h , X)
3    LET i2 = INSERTION_INDEX( e + h , X)
4    FOR EACH X[i] WITH i1 <= i <= i2
5        IF ( d + h < X[i] < e - h )
6            r = r + 1
7        ELSE
8            LET q1 = MAX( d - X[i] / h, -1)
9            LET q2 = MIN( e - X[i] / h,  1)
10           r = r + K.primitive( q2 ) - K.primitive( q1 )
11   RETURN r / n
```

Die Komplexität dieses Algorithmus ist entsprechend des Algorithmus zur einfachen Dichteschätzung B.2 $\Theta(\log n)$. Auch im Fall der Selektivitätsschätzung können Randeffekte auftreten. Der folgende Algorithmus nutzt die Technik der Spiegelung am Falle der Selektivitätsschätzung für eine auf $[a, b]$ beschränkte Verteilung.

[4] Zum Beispiel durch ein Sortieren der Stichprobenwerte und einem anschließenden linearen Durchlauf

Algorithmus B.6 *Sei* `K.primitive(x)` *die Stammfunktion eines auf* $[-1,1]$ *beschränkten Kerns,* `X[1]`$\leq \ldots \leq$`X[n]` *eine aufsteigend sortierte i.i.d. Stichprobe und* $h > 0$ *eine Bandbreite. Dann liefert der folgende Algorithmus eine geschätzte Selektivität zu einer Bereichsanfrage* $Q([d,e])$ *für eine auf* $[a,b]$ *beschränkte Verteilung.*

```
 1    LET r = 0
 2    LET i1 = INSERTION_INDEX( d - h , X)
 3    LET i2 = INSERTION_INDEX( e + h , X)
 4    FOR EACH X[i] WITH i1 <= i <= i2
 5        IF ( d + h < X[i] < e - h )
 6            r = r + 1
 7        ELSE
 8            LET q1 = MAX( d - X[i] / h, -1)
 9            LET q2 = MIN( e - X[i] / h,  1)
10            r = r + K.primitive( q2 ) - K.primitive( q1 )
11            IF ( X[i] < a + h )
12                LET Xa = 2 * a - X[i]
13                q1 = MAX( d - Xa / h, -1)
14                q2 = MIN( e - Xa / h,  1)
15                r = r + K.primitive( q2 ) - K.primitive( q1 )
11            IF ( X[i] > b - h )
12                LET Xb = 2 * b - X[i]
13                q1 = MAX( d - Xb / h, -1)
14                q2 = MIN( e - Xb / h,  1)
15                r = r + K.primitive( q2 ) - K.primitive( q1 )
16    RETURN r / n
```

Das Verfahren des hybriden Kernselektivitätsschätzers setzt, wie bereits der hybride Ansatz zur Dichteschätzung, eine Einteilung des Datenraums in $k+1$ Klassen mit den Grenzen $(c_i, c_{i+1}]$ und die Zuordnung der Stichprobenwerte X_i zu den Klassen voraus. Dies kann auch hier mit einem Aufwand von $\Theta(n \log n)$ erreicht werden. Nun können $k+1$ Selektivitätsschätzer auf Basis der einer Klasse zugeordneten Stichprobenwerte etabliert und kombiniert werden.

Algorithmus B.7 *Seien* `c[i]` *mit* `c[0]`$= a$ *und* `c[k+1]`$= b$ *ein aufsteigend sortiertes Array mit den Klassengrenzen,* `sel[i]` *für* $i = 0, \ldots, k$ *die zu den Klassen gehörenden Selektivitätsschätzer und* `n[i]` *die Zahl der der Klasse* i *zugeordneten Stichprobenwerte. Dann kann ein hybrider Selektivitätsschätzer zur Unterstützung von Bereichsanfragen der Form* $Q([d,e])$ *einer auf* $[a,b]$ *beschränkten Verteilung mittels folgendem Algorithmus implementiert werden.*

```
1    LET r = 0
2    LET i1 = INSERTION_INDEX( d , c) - 1
3    LET i2 = INSERTION_INDEX( e , c) - 1
4    FOR EACH c[i] WITH i1 + 1 <= i <= i2 - 1
5        r = r + n[i]
6    r = r + n[i1] * sel[i1] ( d, c[i1 + 1] )
```

```
7     r = r + n[i2] * sel[i2] ( c[i2], e )
8     RETURN r / n
```

Die Zeilen 4 und 5 summieren die Selektivität der vollständig durch die Anfrage überdeckten Klassen auf. Eine genaue Berechnung ist lediglich für die beiden Klassen notwendig, die geschnitten, aber nicht vollständig überdeckt werden. Für die Grenzen der Klassen gilt das bereits für den hybriden Dichteschätzer Gesagte ebenso für den hybriden Selektivitätsschätzer. Auch hier sollten zur Verringerung der Randeffekte nur Selektivitätsschätzer für die einzelnen Klassen verwendet werden, die Randfehler weitestgehend vermeiden.

Anhang C

Ergebnisse der Tests

Die folgenden Abschnitte zeigen exemplarisch für jeden in dieser Arbeit verwendeten Datentyp zu jeweils einem Grad an Sortiertheit die mittleren Fehler zu verschieden großen Anfragen über die Zeit. Dabei sind die verschiedenen Schätzfunktionen an sechs Zeitpunkten mittels der Testanfragen ausgewertet worden.

Die Spalte „Query“ bezeichnet den verwendeten Satz von Testanfragen zum jeweiligen Datensatz. Die Spalten „M“ und „M_t“ bezeichnen den verwendeten permanenten und temporären Speicher zum Aufbau der adaptiven Schätzer. Für die aufgeführten speicherbasierten Schätzer ist die Summe beider Größen als zur Verfügung stehenden Speicherplatz verwendet worden. Spalte „t“ bezeichnet den diskreten Zeitpunkt der Auswertung über dem Datenstrom.

Alle Schätzer der folgenden Tabellen sind bis auf die gegebenen Sprungstellen vollständig unüberwacht und ohne a priori Informationen angewendet worden. Die Bandbreiten aller Verfahren werden mit Hilfe der Normalskalierungsregel berechnet und als Kernfunktion kommt der Biweight Kern zum Einsatz. Für die adaptiven Verfahren wird eine arithmetische Gewichtung gewählt, da hier vor allem die Fehler nach Verarbeitung aller Daten von Interesse sind.

Aus Gründen der Übersichtlichkeit ist in dieser Arbeit zu jedem Datentyp und jedem Grad an Sortiertheit jweils nur eine Tabelle mit den mittleren Fehler enthalten. Damit sich der Leser trotzdem ein Bild der Einflüsse verschiedener Parameter machen kann, ist zu jedem Datensatz jeweils ein anderer Grad der Sortierung gewählt worden. So sind die Tests zu N(0,1)-Daten für den Datensatz mit der normierten Inversionszahl 0.5 angegeben. Für die CP1-Daten sind die Ergebnisse zum Datensatz mit einer normierten Inversionszahl von 0.03 und für die CP2-Daten von 0.35 angegeben.

C.1 N(0,1)-Daten in zufälliger Reihenfolge

Die Daten des Typs $\mathrm{N}(0,1)_{0.5}$ wurden mit vier verschiedenen Schätzern ausgewertet. Dabei bezeichnet „sp. KS“ einen speicherbasierten gespiegelten Kernselektivitätsschätzer und „adapt. KS“ sein adaptives Gegenüber. Die Spalten „sp. Sampl.“ und „ad. Sampl.“

enthalten die Fehler zu einfachem speicherbasiertem und adaptivem Sampling.

Query	M	M_t	t	sp. KS	adapt. KS	sp. Sampl.	ad. Sampl.
$Q_{0.01}$	100	100	13000	10.9162%	3.8153%	40.5381%	2.5850%
			100000	14.4907%	3.4884%	39.1097%	0.8256%
			250000	11.6319%	3.6200%	41.1081%	0.6745%
			500000	11.3595%	3.5924%	40.1514%	0.5830%
			750000	9.1930%	3.5615%	34.6437%	0.5493%
			1000000	8.8942%	3.5819%	38.4006%	0.5425%
	500	500	13000	9.3985%	2.5621%	18.2530%	4.6539%
			100000	6.2745%	1.9142%	16.9980%	1.4552%
			250000	5.4952%	2.0136%	14.5378%	0.7764%
			500000	5.0485%	1.9611%	16.3969%	0.5073%
			750000	5.5096%	1.9297%	17.2628%	0.3874%
			1000000	4.9615%	1.9537%	16.7394%	0.2719%
	500	1500	13000	6.0646%	1.9542%	11.2471%	4.7520%
			100000	4.2945%	1.3144%	12.3436%	1.4477%
			250000	3.9950%	1.3747%	13.8196%	0.7749%
			500000	6.0262%	1.3318%	10.8147%	0.4981%
			750000	4.7943%	1.2957%	11.9728%	0.3789%
			1000000	4.5850%	1.3175%	12.0415%	0.2719%
	1000	4000	13000	3.1399%	1.9617%	7.3869%	5.1213%
			100000	3.1664%	0.9978%	8.0147%	1.3870%
			250000	3.1817%	1.0039%	7.1349%	0.7902%
			500000	4.1178%	0.9799%	8.0722%	0.5485%
			750000	4.1423%	0.9485%	7.8544%	0.3489%
			1000000	3.8062%	0.9691%	7.2039%	0.1762%
	3000	10000	13000	2.1838%	2.3082%	4.7755%	5.4597%
			100000	1.9889%	0.8498%	4.6410%	1.4576%
			250000	2.5906%	0.8361%	4.4109%	0.7721%
			500000	2.9435%	0.7810%	4.9911%	0.5369%
			750000	2.4824%	0.7494%	5.5314%	0.3194%
			1000000	2.3195%	0.7605%	4.9217%	0.1047%
$Q_{0.02}$	100	100	13000	11.8155%	4.0820%	28.1744%	2.5989%
			100000	15.9168%	3.6916%	29.3560%	0.6858%
			250000	13.3093%	3.8783%	31.8867%	0.5261%
			500000	13.2343%	3.8831%	34.5844%	0.4241%
			750000	9.8749%	3.8475%	25.4927%	0.3477%
			1000000	9.9705%	3.8690%	27.4933%	0.3427%
	500	500	13000	10.0611%	2.7050%	14.3327%	3.6531%
			100000	6.4238%	1.9851%	13.1253%	1.0150%
			250000	5.6905%	2.1160%	8.7515%	0.5288%
			500000	5.2182%	2.1006%	12.1976%	0.3648%
			750000	5.6708%	2.0597%	12.5969%	0.2653%
			1000000	5.5128%	2.0820%	12.2748%	0.1406%
	500	1500	13000	6.1356%	2.0582%	8.1145%	3.6434%
			100000	4.2066%	1.3150%	7.8879%	1.0519%
			250000	4.3326%	1.4018%	10.4780%	0.5246%
			500000	6.2770%	1.3936%	8.1315%	0.3574%
			750000	4.6558%	1.3427%	8.8558%	0.2599%
			1000000	4.7262%	1.3647%	8.5750%	0.1399%
	1000	4000	13000	3.2840%	2.0449%	5.0206%	3.7048%
			100000	3.4637%	0.9167%	5.4250%	1.0167%

Query	M	M_t	t	sp. KS	adapt. KS	sp. Sampl.	ad. Sampl.
			250000	3.5643%	0.9711%	5.4916%	0.5753%
			500000	4.5600%	0.9686%	6.4175%	0.3807%
			750000	4.3600%	0.9151%	5.3218%	0.2386%
			1000000	4.0399%	0.9417%	6.0543%	0.0990%
	3000	10000	13000	2.1412%	2.4267%	3.7325%	4.1159%
			100000	1.9709%	0.7133%	3.0737%	1.0706%
			250000	2.4912%	0.7488%	3.2783%	0.5506%
			500000	2.9270%	0.7062%	3.4084%	0.3713%
			750000	2.4398%	0.6488%	4.5524%	0.2244%
			1000000	2.4319%	0.6721%	4.0038%	0.0547%
$Q_{0.05}$	100	100	13000	11.0453%	3.7553%	17.1588%	1.8339%
			100000	14.7240%	3.4479%	22.4777%	0.4713%
			250000	11.8114%	3.6304%	20.1475%	0.3067%
			500000	12.0405%	3.6089%	20.6196%	0.2462%
			750000	9.2431%	3.5877%	14.0679%	0.1545%
			1000000	8.3274%	3.6081%	18.3905%	0.1457%
	500	500	13000	9.2000%	2.3030%	8.4721%	1.8213%
			100000	4.9614%	1.8039%	6.6072%	0.5204%
			250000	3.9097%	1.9684%	5.1730%	0.3071%
			500000	4.5721%	1.9291%	6.8365%	0.2381%
			750000	5.0357%	1.9043%	7.3258%	0.1229%
			1000000	5.0414%	1.9280%	6.1273%	0.0499%
	500	1500	13000	5.0919%	1.6726%	4.6097%	1.8516%
			100000	3.4015%	1.1758%	3.5934%	0.5321%
			250000	3.4484%	1.2923%	6.0331%	0.3032%
			500000	5.3872%	1.2636%	4.3322%	0.2340%
			750000	4.0762%	1.2281%	5.6629%	0.1210%
			1000000	4.4773%	1.2510%	5.2310%	0.0509%
	1000	4000	13000	2.1634%	1.5280%	2.3968%	1.9062%
			100000	2.5710%	0.7870%	3.3578%	0.5386%
			250000	2.6066%	0.8821%	3.7786%	0.3206%
			500000	3.5471%	0.8677%	3.3216%	0.2381%
			750000	3.2637%	0.8240%	2.4764%	0.1154%
			1000000	3.0113%	0.8518%	3.7788%	0.0345%
	3000	10000	13000	1.6107%	1.6104%	1.8835%	2.0865%
			100000	1.7125%	0.5666%	1.6727%	0.5413%
			250000	2.2192%	0.6660%	1.7077%	0.3094%
			500000	2.4023%	0.6249%	2.1639%	0.2466%
			750000	1.6146%	0.5682%	3.2129%	0.1139%
			1000000	1.8218%	0.5940%	2.4129%	0.0218%
$Q_{0.1}$	100	100	13000	9.1578%	3.4471%	13.7702%	1.1872%
			100000	12.5969%	3.1727%	16.9599%	0.2985%
			250000	10.1388%	3.3247%	10.1097%	0.2037%
			500000	10.2176%	3.2991%	9.7489%	0.1682%
			750000	6.3863%	3.2730%	7.2742%	0.0830%
			1000000	7.2114%	3.2914%	10.9839%	0.0707%
	500	500	13000	8.0521%	2.1382%	4.4699%	1.0996%
			100000	3.3922%	1.6682%	4.0371%	0.3445%
			250000	2.5986%	1.8010%	3.5494%	0.2059%
			500000	3.8190%	1.7626%	3.8757%	0.1819%
			750000	3.8260%	1.7312%	4.3147%	0.0793%

Query	M	M_t	t	sp. KS	adapt. KS	sp. Sampl.	ad. Sampl.
			1000000	3.2622%	1.7511%	3.0089%	0.0263%
	500	1500	13000	3.6387%	1.4255%	3.2088%	0.9152%
			100000	2.5854%	1.0734%	2.1582%	0.3293%
			250000	2.4566%	1.1816%	3.1693%	0.2000%
			500000	3.6243%	1.1556%	2.6137%	0.1779%
			750000	2.6901%	1.1154%	3.4370%	0.0785%
			1000000	3.7745%	1.1330%	3.4563%	0.0274%
	1000	4000	13000	1.5481%	1.1738%	1.1072%	0.9305%
			100000	2.0395%	0.7117%	2.4914%	0.3414%
			250000	2.1580%	0.8037%	2.5841%	0.2051%
			500000	2.3604%	0.7899%	1.8530%	0.1785%
			750000	1.9395%	0.7500%	1.6198%	0.0780%
			1000000	2.1390%	0.7691%	2.9217%	0.0186%
	3000	10000	13000	1.2394%	0.9020%	1.1096%	0.9525%
			100000	1.4148%	0.4880%	1.2384%	0.3465%
			250000	1.8027%	0.6012%	0.9725%	0.2035%
			500000	2.0028%	0.5623%	1.1125%	0.1806%
			750000	1.1412%	0.5175%	1.7375%	0.0737%
			1000000	1.2563%	0.5346%	1.2868%	0.0113%
$Q_{0.2}$	100	100	13000	5.9453%	2.5291%	9.0847%	0.7955%
			100000	9.3308%	2.3999%	14.3567%	0.2474%
			250000	5.7563%	2.5404%	5.4369%	0.1503%
			500000	7.8680%	2.5253%	5.2888%	0.1268%
			750000	4.7259%	2.5069%	3.4663%	0.0445%
			1000000	5.9882%	2.5170%	5.8419%	0.0333%
	500	500	13000	6.1353%	1.5039%	3.3376%	0.8506%
			100000	1.8257%	1.2425%	2.7586%	0.2673%
			250000	1.5760%	1.3764%	2.1250%	0.1298%
			500000	2.8272%	1.3486%	1.8604%	0.1149%
			750000	2.7174%	1.3271%	3.1833%	0.0367%
			1000000	1.6785%	1.3382%	1.8475%	0.0139%
	500	1500	13000	2.5656%	0.9066%	2.2298%	0.6482%
			100000	1.5955%	0.7894%	1.1596%	0.2393%
			250000	1.7175%	0.9059%	1.4513%	0.1224%
			500000	2.9572%	0.8837%	0.9551%	0.1178%
			750000	1.3497%	0.8555%	1.5225%	0.0363%
			1000000	2.9384%	0.8660%	1.5720%	0.0135%
	1000	4000	13000	0.9021%	0.7431%	0.6200%	0.6553%
			100000	1.2550%	0.5292%	1.5653%	0.2722%
			250000	1.5142%	0.6220%	1.3830%	0.1180%
			500000	1.6600%	0.6049%	1.2944%	0.1186%
			750000	1.1577%	0.5761%	0.8774%	0.0368%
			1000000	1.3319%	0.5871%	2.0651%	0.0097%
	3000	10000	13000	0.9162%	0.5273%	0.8416%	0.5449%
			100000	0.6126%	0.3793%	1.0270%	0.2728%
			250000	1.2728%	0.4664%	0.5190%	0.1328%
			500000	1.3092%	0.4300%	0.4908%	0.1178%
			750000	0.6920%	0.3963%	0.7844%	0.0361%
			1000000	0.5659%	0.4079%	0.6982%	0.0056%

Tabelle C.1: Mittlere relative Fehler zu verschiedenen Zeitpunkten mit verschiedenen Selektivitätsschätzern über $N(0,1)_{0.5}$-Daten

C.2 CP2-Daten in teilsortierter Reihenfolge

Die Daten des Typs $CP2_{0.35}$ wurden mit sechs verschiedenen Schätzern ausgewertet. Die Spaltenbezeichnungen stimmen mit denen in Abschnitt C.1 überein. Zusätzlich bezeichnen die Spalten „hyb. sp. KS“ und „hyb. ad. KS“ die Ergebnisse des speicherbasierten bzw. adaptiven hybriden Kernschätzers.

Query	M	M_t	t	sp. KS	adapt. KS	sp. Sampl.	ad. Sampl.	hyb.sp.KS	hyb.ad.KS
$Q_{0.01}$	100	100	13000	66.4200%	59.4308%	89.7792%	65.7921%	108.1346%	65.5757%
			100000	52.7212%	54.2373%	80.0258%	60.7382%	82.7768%	60.5284%
			250000	52.5703%	48.2352%	72.9956%	50.9111%	80.2633%	50.8486%
			500000	46.1460%	31.6476%	60.5611%	34.3181%	72.0816%	34.2343%
			750000	28.9095%	17.6877%	56.8312%	17.5090%	66.6432%	17.4243%
			1000000	24.7880%	4.2088%	53.0521%	1.7634%	61.2591%	1.7496%
	500	500	13000	64.1999%	63.2072%	67.8707%	66.3225%	62.2637%	66.2607%
			100000	60.5176%	58.3305%	64.7870%	60.8650%	63.0826%	60.8249%
			250000	50.1587%	50.5342%	52.4793%	51.0084%	58.6913%	50.9944%
			500000	36.0706%	34.7416%	36.1518%	34.2811%	45.2429%	34.2523%
			750000	21.1872%	18.7189%	25.5271%	16.9621%	33.3888%	16.9395%
			1000000	18.3802%	3.5113%	25.3011%	0.4778%	28.3427%	0.4209%
	500	1500	13000	61.1736%	63.5643%	67.5395%	66.2518%	72.3038%	66.2338%
			100000	60.1376%	58.7624%	64.5120%	60.9557%	66.0102%	60.9255%
			250000	50.3044%	50.4555%	51.4619%	51.0717%	53.2793%	51.0595%
			500000	35.1096%	34.5387%	36.7765%	34.3180%	49.5288%	34.2949%
			750000	20.1410%	18.3185%	22.4279%	16.9624%	36.7886%	16.9419%
			1000000	12.2081%	2.9458%	17.2563%	0.4843%	31.7552%	0.4156%
	1000	4000	13000	64.5410%	63.2362%	66.9640%	66.3002%	65.3876%	66.2565%
			100000	59.5594%	58.8299%	60.6023%	60.9168%	62.7800%	60.8872%
			250000	49.0514%	50.2204%	52.6890%	51.1366%	56.5335%	51.1450%
			500000	33.0299%	34.4713%	36.7353%	34.2790%	39.6855%	34.2716%
			750000	16.9118%	18.3091%	19.1152%	17.0922%	26.6825%	17.0910%
			1000000	7.4839%	2.5885%	11.0182%	0.2811%	19.6565%	0.2654%
	3000	10000	13000	64.0049%	64.4386%	66.2216%	66.9278%	67.5628%	66.9322%
			100000	60.1077%	59.2214%	60.4406%	60.9118%	60.5855%	60.8922%
			250000	50.9101%	50.2825%	51.3656%	51.0347%	52.9306%	51.0082%
			500000	35.0060%	34.4948%	35.2006%	34.2821%	36.3831%	34.2669%
			750000	18.4091%	17.9818%	18.7508%	16.9664%	20.0164%	16.9742%
			1000000	5.0628%	2.1606%	7.6805%	0.1515%	10.2074%	0.2357%
$Q_{0.02}$	100	100	13000	66.4823%	59.9610%	77.2252%	64.8234%	100.1698%	64.6839%
			100000	50.5036%	54.9925%	70.0907%	60.2450%	70.7016%	60.0821%
			250000	51.5514%	48.7521%	58.0622%	50.8285%	66.0256%	50.7975%
			500000	45.2798%	32.3323%	48.8884%	34.3345%	59.9656%	34.2856%
			750000	29.2541%	17.6443%	40.3783%	17.3739%	51.8223%	17.3168%
			1000000	24.2507%	3.3152%	40.2388%	1.1266%	48.2438%	1.1024%
	500	500	13000	63.1823%	62.1804%	65.5983%	64.8410%	58.5734%	64.7203%
			100000	59.0841%	58.1080%	61.8301%	60.1328%	61.2515%	60.0965%
			250000	50.6752%	49.8990%	50.6540%	50.6662%	56.4695%	50.6710%
			500000	34.9599%	34.6237%	33.2622%	34.2384%	42.9824%	34.2249%
			750000	20.3212%	18.5029%	19.3585%	17.0042%	27.5152%	16.9913%
			1000000	16.8536%	2.7234%	16.8658%	0.2302%	21.1121%	0.2160%
	500	1500	13000	60.0820%	61.7873%	66.4039%	64.8620%	70.1752%	64.7346%
			100000	60.0980%	57.9429%	62.2642%	60.2234%	62.8682%	60.1884%
			250000	49.7306%	49.5012%	50.3519%	50.7284%	52.4042%	50.7312%
			500000	34.5986%	34.2256%	34.4595%	34.2749%	47.9847%	34.2629%
			750000	19.8492%	17.9485%	18.2370%	17.0042%	31.5489%	16.9930%
			1000000	10.6067%	2.0429%	12.6379%	0.2459%	24.4371%	0.2237%
	1000	4000	13000	63.5428%	62.0242%	66.2429%	64.7859%	62.1505%	64.7543%
			100000	58.9396%	57.9879%	60.1413%	60.1252%	59.7204%	60.1029%
			250000	49.1611%	49.5170%	52.0409%	50.7948%	53.1387%	50.7935%
			500000	32.9377%	33.9931%	36.3499%	34.2168%	36.3453%	34.2144%
			750000	16.5542%	17.7114%	17.7551%	17.1134%	22.0543%	17.1121%
			1000000	6.3095%	1.5343%	7.6726%	0.1327%	13.5765%	0.1216%
	3000	10000	13000	62.6897%	63.0288%	64.7774%	65.3655%	65.0525%	65.2792%
			100000	59.1501%	58.2876%	59.5373%	60.1150%	59.7316%	60.1050%

Query	M	M_t	t	sp. KS	adapt. KS	sp. Sampl.	ad. Sampl.	hyb.sp.KS	hyb.ad.KS
			250000	50.0187%	49.5355%	51.2314%	50.6590%	52.6778%	50.6584%
			500000	34.3307%	33.8928%	35.2769%	34.2113%	35.3805%	34.2160%
			750000	17.6968%	17.3171%	18.5419%	16.9835%	18.5135%	16.9857%
			1000000	3.8747%	1.1609%	5.1962%	0.0749%	7.4077%	0.1051%
$Q_{0.05}$	100	100	13000	64.6258%	59.6051%	65.9886%	62.5381%	87.1847%	62.5028%
			100000	49.0511%	55.0893%	62.4543%	58.2153%	60.3540%	58.1535%
			250000	47.2705%	47.6888%	52.2173%	49.2275%	51.6633%	49.2291%
			500000	38.6808%	31.8875%	36.6139%	33.2809%	43.1019%	33.2764%
			750000	24.1111%	16.4331%	27.3427%	16.3518%	34.3251%	16.3342%
			1000000	20.9640%	1.9909%	26.3620%	0.4791%	32.0064%	0.4501%
	500	500	13000	61.5796%	60.2292%	63.7345%	62.4850%	55.2551%	62.4574%
			100000	57.4379%	56.1892%	59.2827%	58.1347%	60.5878%	58.1318%
			250000	48.8977%	47.7217%	48.0397%	49.1382%	53.6524%	49.1337%
			500000	32.8075%	32.4593%	30.7725%	33.2424%	39.0230%	33.2333%
			750000	17.2525%	16.4022%	16.6639%	16.2960%	22.5177%	16.2877%
			1000000	12.9517%	1.2447%	12.1444%	0.0967%	14.1194%	0.0827%
	500	1500	13000	59.5312%	60.9948%	64.3250%	62.4239%	68.2452%	62.4059%
			100000	59.0326%	57.0110%	60.9730%	58.2179%	60.5255%	58.2180%
			250000	48.0162%	48.3107%	48.1266%	49.1944%	51.1857%	49.1903%
			500000	32.5283%	32.6535%	32.6115%	33.2774%	44.1864%	33.2700%
			750000	16.4932%	16.2223%	15.3203%	16.2959%	24.4646%	16.2886%
			1000000	6.5616%	0.8006%	7.6161%	0.1227%	16.5220%	0.1050%
	1000	4000	13000	62.9191%	61.5260%	63.6091%	62.4279%	59.5762%	62.4144%
			100000	58.3472%	57.3692%	57.4866%	58.1366%	57.5609%	58.1254%
			250000	48.1944%	48.7239%	50.7366%	49.2554%	50.3874%	49.2595%
			500000	31.6253%	32.8261%	35.4674%	33.2316%	34.9597%	33.2328%
			750000	14.7300%	16.2519%	16.2891%	16.4142%	19.8042%	16.4129%
			1000000	3.7155%	0.5411%	3.8811%	0.0539%	9.9556%	0.0497%
	3000	10000	13000	61.9919%	62.3345%	62.4406%	62.8540%	63.0353%	62.8564%
			100000	58.2133%	57.6579%	57.5664%	58.1241%	57.7377%	58.1259%
			250000	48.9480%	48.7932%	49.9089%	49.1328%	50.9772%	49.1336%
			500000	33.3970%	32.9763%	34.0412%	33.2322%	34.3319%	33.2346%
			750000	16.5656%	16.1566%	17.5452%	16.2846%	17.0360%	16.2885%
			1000000	2.2033%	0.3726%	3.0786%	0.0293%	4.8414%	0.0439%
$Q_{0.1}$	100	100	13000	59.4416%	56.2615%	58.0203%	57.8516%	81.7877%	57.8170%
			100000	48.1496%	52.6243%	55.9579%	54.2685%	55.0443%	54.2254%
			250000	43.0440%	45.6904%	47.0676%	46.5744%	44.0677%	46.5792%
			500000	34.8647%	31.2182%	29.3630%	31.9975%	31.7129%	31.9990%
			750000	16.9915%	15.5256%	22.3778%	15.7857%	22.4385%	15.7783%
			1000000	15.6240%	1.1205%	17.0158%	0.2723%	22.0887%	0.2651%
	500	500	13000	57.4602%	56.8684%	59.1213%	57.8182%	50.6960%	57.7898%
			100000	54.5683%	53.3252%	56.7749%	54.1863%	56.2215%	54.1820%
			250000	46.9476%	45.9169%	45.9646%	46.5562%	51.1086%	46.5534%
			500000	32.2187%	31.4898%	29.3948%	31.9973%	37.4124%	31.9931%
			750000	16.4161%	15.7307%	14.4751%	15.7729%	19.0823%	15.7711%
			1000000	8.4076%	0.6713%	10.1651%	0.0441%	10.9013%	0.0432%
	500	1500	13000	55.0299%	56.9200%	58.8504%	57.6631%	61.6562%	57.6417%
			100000	54.5171%	53.6103%	56.2808%	54.2538%	53.5571%	54.2513%
			250000	45.1126%	46.1428%	45.7812%	46.6078%	48.6688%	46.6054%
			500000	31.2297%	31.6634%	31.4867%	32.0311%	41.1531%	32.0279%
			750000	15.5436%	15.6553%	14.4452%	15.7730%	20.7318%	15.7712%
			1000000	3.6474%	0.4711%	5.8375%	0.0777%	10.7406%	0.0753%
	1000	4000	13000	58.0446%	57.0828%	58.5972%	57.6757%	54.8405%	57.6458%
			100000	54.9256%	53.7029%	53.7130%	54.1819%	52.4138%	54.1773%
			250000	45.7871%	46.3090%	47.5148%	46.6578%	47.1497%	46.6600%
			500000	30.3895%	31.7193%	33.9676%	31.9889%	32.7587%	31.9891%
			750000	14.3606%	15.7601%	15.5472%	15.8939%	17.7191%	15.8950%
			1000000	2.4258%	0.3354%	2.0086%	0.0267%	7.9616%	0.0263%
	3000	10000	13000	57.4189%	57.7756%	57.8021%	58.2097%	58.5731%	58.2031%
			100000	54.3457%	53.8352%	53.5284%	54.1807%	53.6868%	54.1778%
			250000	46.6350%	46.3023%	47.4079%	46.5527%	47.4248%	46.5498%
			500000	32.0301%	31.7980%	32.9284%	31.9921%	32.5719%	31.9885%
			750000	15.9036%	15.6647%	16.8140%	15.7750%	16.2276%	15.7765%
			1000000	1.5867%	0.2414%	1.9941%	0.0141%	3.0649%	0.0227%
$Q_{0.2}$	100	100	13000	51.4331%	49.4768%	50.1983%	50.6197%	73.7617%	50.5930%

Query	M	M_t	t	sp. KS	adapt. KS	sp. Sampl.	ad. Sampl.	hyb.sp.KS	hyb.ad.KS
			100000	41.9270%	46.5004%	50.1416%	47.7574%	49.3093%	47.7266%
			250000	37.5548%	41.0353%	40.0448%	41.7306%	37.3115%	41.7248%
			500000	31.6082%	29.0230%	23.8546%	29.5797%	22.7321%	29.5757%
			750000	14.0429%	14.5715%	17.5121%	14.9898%	13.1295%	14.9806%
			1000000	11.2519%	0.7071%	10.9447%	0.1471%	15.3902%	0.1401%
	500	500	13000	49.8874%	49.7561%	51.7269%	50.5557%	43.6559%	50.5409%
			100000	48.0221%	46.9553%	50.0269%	47.7273%	50.2119%	47.7193%
			250000	42.3329%	41.1497%	41.0762%	41.7087%	46.0194%	41.7053%
			500000	30.1978%	29.1768%	27.8197%	29.5851%	34.1036%	29.5805%
			750000	15.3523%	14.6887%	12.9108%	14.9901%	16.9800%	14.9857%
			1000000	6.4261%	0.4506%	8.1794%	0.0243%	7.2601%	0.0228%
	500	1500	13000	48.4652%	49.7908%	51.3115%	50.3098%	53.1240%	50.3096%
			100000	47.6596%	47.2602%	49.3203%	47.7766%	46.9289%	47.7713%
			250000	39.6539%	41.3927%	41.0657%	41.7524%	43.0858%	41.7502%
			500000	28.4129%	29.3510%	29.1187%	29.6148%	38.0721%	29.6111%
			750000	14.3126%	14.7934%	13.6620%	14.9901%	18.1999%	14.9866%
			1000000	2.9003%	0.2996%	3.4010%	0.0631%	5.5364%	0.0597%
	1000	4000	13000	50.6395%	49.9302%	51.1258%	50.3186%	48.1698%	50.3073%
			100000	48.3553%	47.3630%	47.3782%	47.7229%	47.0886%	47.7202%
			250000	41.0961%	41.5235%	42.5659%	41.7913%	42.2300%	41.7926%
			500000	27.7610%	29.3931%	31.2579%	29.5839%	29.3547%	29.5839%
			750000	13.6980%	14.9600%	14.5932%	15.1029%	15.7013%	15.1027%
			1000000	1.0584%	0.2134%	1.0735%	0.0132%	5.5128%	0.0127%
	3000	10000	13000	50.2808%	50.5054%	50.5374%	50.7830%	51.6923%	50.7770%
			100000	47.9354%	47.4624%	47.1352%	47.7216%	47.5704%	47.7208%
			250000	41.7251%	41.5024%	42.6815%	41.7072%	42.5134%	41.7064%
			500000	29.5478%	29.4416%	30.7089%	29.5837%	30.1919%	29.5842%
			750000	14.9379%	14.8833%	15.9751%	14.9891%	15.5986%	14.9901%
			1000000	1.2174%	0.1576%	1.3374%	0.0069%	2.7325%	0.0104%

Tabelle C.2: Mittlere relative Fehler zu verschiedenen Zeitpunkten mit verschiedenen Selektivitätsschätzern über $CP2_{0.35}$-Daten

C.3 CP1-Daten in fast sortierter Reihenfolge

Die Daten des Typs $CP1_{0.03}$ wurden entsprechend den Tests des vorherigen Abschnitts mit sechs verschiedenen Schätzern ausgewertet. Die Spaltenbezeichnungen stimmen mit denen in Abschnitt C.2 überein.

Query	M	M_t	t	sp. KS	adapt. KS	sp. Sampl.	ad. Sampl.	hyb.sp.KS	hyb.ad.KS
$Q_{0.01}$	100	100	13000	107.3404%	98.5780%	122.8153%	155.0907%	122.4912%	154.2617%
			100000	157.6061%	151.0066%	155.4733%	157.5247%	159.8297%	157.4478%
			250000	139.9492%	138.6349%	138.4790%	141.2296%	148.4408%	141.1902%
			500000	90.5427%	91.1470%	93.7362%	92.4255%	104.6243%	92.4216%
			750000	52.2538%	47.6636%	66.6705%	48.2675%	84.5880%	48.2451%
			1000000	23.8324%	2.8928%	60.6445%	1.1244%	57.0725%	1.1410%
	500	500	13000	113.4577%	110.1846%	124.0967%	124.7710%	124.0754%	124.8208%
			100000	153.6542%	156.8343%	159.5223%	159.4101%	157.2469%	159.4021%
			250000	141.0902%	140.2232%	139.3962%	141.5760%	139.0476%	141.5766%
			500000	94.3207%	94.0483%	94.9168%	94.6186%	94.9299%	94.6141%
			750000	50.0689%	48.5940%	54.3917%	48.8258%	56.3171%	48.7971%
			1000000	13.0896%	1.1947%	26.7176%	0.4007%	22.7337%	0.5729%
	500	1500	13000	114.4167%	112.7035%	123.3746%	123.4114%	122.6527%	123.2997%
			100000	157.9127%	157.8743%	159.2339%	160.1042%	159.5878%	160.1047%
			250000	141.2134%	140.6837%	141.8522%	141.5334%	143.4988%	141.5340%
			500000	94.3627%	94.3157%	94.9596%	94.7079%	96.7843%	94.7026%
			750000	48.9852%	48.5837%	50.1751%	48.8258%	55.3872%	48.8171%
			1000000	12.0647%	0.9745%	15.2413%	0.4359%	23.1555%	0.4640%
	1000	4000	13000	117.8435%	113.5344%	124.0199%	122.7797%	123.2775%	122.7539%
			100000	158.9942%	157.9394%	158.4814%	159.2637%	158.1573%	159.2776%
			250000	141.6899%	140.8226%	140.7099%	141.4326%	140.7137%	141.4319%
			500000	94.7261%	94.3662%	94.0163%	94.7534%	93.6115%	94.7532%
			750000	49.1057%	48.9576%	48.7570%	49.1819%	50.9362%	49.1765%

Query	M	M_t	t	sp. KS	adapt. KS	sp. Sampl.	ad. Sampl.	hyb.sp.KS	hyb.ad.KS
			1000000	7.2923%	0.6783%	9.7840%	0.2538%	18.3306%	0.2547%
	3000	10000	13000	119.4691%	113.4528%	123.9554%	120.4380%	123.7784%	120.4676%
			100000	158.8123%	158.4250%	159.2297%	159.3776%	160.1325%	159.3929%
			250000	141.0783%	140.8781%	140.4133%	141.2843%	142.3417%	141.2870%
			500000	94.4319%	94.4974%	94.4288%	94.7490%	95.4130%	94.7491%
			750000	48.6959%	48.6329%	48.5925%	48.8409%	49.5612%	48.8284%
			1000000	5.6903%	0.5612%	6.9914%	0.1359%	11.2821%	0.1492%
$Q_{0.02}$	100	100	13000	107.1623%	98.6272%	113.3894%	157.4457%	112.5484%	156.5956%
			100000	164.1798%	157.1946%	165.6815%	163.5585%	162.1547%	163.4514%
			250000	142.0563%	140.6610%	145.7105%	143.2145%	150.6115%	143.1493%
			500000	89.5544%	90.3520%	93.4672%	91.6056%	98.6428%	91.5813%
			750000	51.8218%	46.2835%	57.5555%	46.9284%	72.0694%	46.9151%
			1000000	24.3246%	2.3690%	42.4226%	0.8422%	49.0358%	0.8642%
	500	500	13000	111.1778%	109.1438%	114.3021%	114.3830%	113.5289%	114.5524%
			100000	160.0835%	162.5057%	161.7778%	163.4640%	163.4654%	163.4325%
			250000	144.1558%	142.9362%	139.3702%	143.6013%	141.2914%	143.5739%
			500000	93.4247%	93.4615%	92.2219%	93.7349%	93.2314%	93.7071%
			750000	47.4636%	46.9989%	47.8799%	47.2299%	51.8416%	47.2367%
			1000000	12.4055%	0.7344%	18.6458%	0.2081%	20.3989%	0.2900%
	500	1500	13000	111.6119%	110.5144%	113.5407%	112.5159%	112.6855%	112.4457%
			100000	163.2153%	162.8132%	162.8173%	163.5502%	164.7602%	163.5256%
			250000	143.4301%	143.4844%	143.1031%	143.7832%	145.5200%	143.7615%
			500000	93.7651%	93.7101%	92.9826%	93.8231%	94.9404%	93.7968%
			750000	46.8935%	47.0892%	47.4365%	47.2301%	51.9805%	47.2285%
			1000000	11.4027%	0.5674%	11.2124%	0.2245%	20.2825%	0.2367%
	1000	4000	13000	112.8709%	109.5904%	114.0612%	111.3777%	112.2045%	111.3454%
			100000	163.9951%	163.1800%	162.9568%	163.5089%	161.2268%	163.4975%
			250000	143.5474%	143.8887%	143.0561%	144.1022%	142.5903%	144.0881%
			500000	93.5729%	93.6412%	93.1262%	93.7624%	90.9522%	93.7508%
			750000	46.4624%	47.4117%	47.0465%	47.5127%	46.5157%	47.5130%
			1000000	7.2166%	0.3783%	5.9729%	0.1255%	13.6753%	0.1304%
	3000	10000	13000	112.9354%	105.5948%	113.8965%	106.2113%	113.2461%	106.1645%
			100000	163.7246%	163.3450%	163.2939%	163.5859%	163.1846%	163.5707%
			250000	143.3909%	143.3095%	142.0666%	143.4296%	143.7701%	143.4183%
			500000	93.7120%	93.6855%	93.4726%	93.7549%	94.2199%	93.7460%
			750000	47.2700%	47.1668%	47.1719%	47.2653%	47.9569%	47.2640%
			1000000	5.4643%	0.2918%	4.7151%	0.0703%	10.0958%	0.0839%
$Q_{0.05}$	100	100	13000	98.9291%	94.8760%	103.1906%	114.8340%	102.3850%	114.5470%
			100000	145.2430%	140.6479%	144.7082%	142.8815%	149.0499%	142.8165%
			250000	135.2489%	137.0639%	137.2004%	137.9883%	147.7370%	137.9561%
			500000	86.1023%	89.5358%	88.2005%	89.9764%	96.6220%	89.9748%
			750000	43.5486%	44.3458%	46.5432%	44.6626%	56.4818%	44.6493%
			1000000	21.0823%	1.1991%	28.0446%	0.3695%	32.9248%	0.3978%
	500	500	13000	100.3862%	99.3860%	103.4944%	103.5853%	102.5778%	103.6466%
			100000	141.6732%	143.8157%	141.4834%	144.5421%	143.1814%	144.5138%
			250000	138.8906%	137.3319%	133.0144%	137.6302%	135.7941%	137.6113%
			500000	90.7080%	90.4522%	89.0668%	90.5757%	90.9764%	90.5678%
			750000	45.1995%	44.8037%	44.3606%	44.9542%	46.4362%	44.9432%
			1000000	9.0649%	0.4543%	10.4341%	0.0776%	13.6699%	0.1154%
	500	1500	13000	101.2694%	100.5332%	102.9427%	102.8679%	102.2968%	102.8458%
			100000	144.9198%	143.9550%	143.0295%	144.5532%	144.3288%	144.5401%
			250000	138.4098%	137.6202%	136.5356%	137.7824%	140.1200%	137.7720%
			500000	91.2935%	90.5862%	89.8420%	90.6516%	93.6272%	90.6485%
			750000	45.3307%	44.8618%	45.0450%	44.9543%	48.8819%	44.9440%
			1000000	8.3676%	0.3612%	7.4509%	0.1216%	17.2562%	0.1285%
	1000	4000	13000	102.1611%	100.7018%	103.2901%	102.5452%	102.6566%	102.5411%
			100000	144.3663%	144.1609%	143.8664%	144.4207%	142.9037%	144.4173%
			250000	137.4189%	137.8384%	137.1443%	137.9426%	137.3834%	137.9365%
			500000	90.1159%	90.5329%	90.2985%	90.5933%	88.7849%	90.5973%
			750000	44.4245%	45.2452%	45.0754%	45.3119%	43.0202%	45.3042%
			1000000	5.2625%	0.2286%	3.1166%	0.0498%	10.9780%	0.0538%
	3000	10000	13000	102.2963%	99.8224%	103.2224%	101.2200%	103.0897%	101.2167%
			100000	144.4409%	144.2708%	144.3049%	144.4490%	144.7218%	144.4434%
			250000	137.5973%	137.5067%	136.5018%	137.5711%	138.2028%	137.5665%
			500000	90.5955%	90.5572%	90.3274%	90.5937%	91.3861%	90.5948%

Query	M	M_t	t	sp. KS	adapt. KS	sp. Sampl.	ad. Sampl.	hyb.sp.KS	hyb.ad.KS
			750000	45.1955%	44.9129%	44.6941%	44.9567%	44.8566%	44.9510%
			1000000	3.5515%	0.1725%	2.8355%	0.0291%	6.3373%	0.0322%
$Q_{0.1}$	100	100	13000	96.3893%	94.0102%	97.9777%	103.1295%	96.5005%	102.9384%
			100000	118.0022%	115.2044%	117.0644%	116.4726%	120.4118%	116.4349%
			250000	119.5841%	121.0400%	120.4823%	121.5684%	129.8792%	121.5450%
			500000	82.8333%	86.1921%	83.0327%	86.4379%	89.3838%	86.4261%
			750000	41.7854%	42.7190%	39.6062%	42.8972%	45.7612%	42.8834%
			1000000	16.6216%	0.8019%	20.5220%	0.2025%	23.2752%	0.2239%
	500	500	13000	95.9217%	96.0367%	97.9892%	97.9428%	97.3298%	97.9521%
			100000	114.7031%	116.4421%	114.7105%	116.6707%	115.8768%	116.6435%
			250000	122.4435%	121.2884%	117.4269%	121.4251%	120.0822%	121.4052%
			500000	87.1713%	86.6700%	86.1124%	86.7421%	86.2480%	86.7276%
			750000	42.1360%	42.9205%	42.6783%	42.9858%	43.1228%	42.9743%
			1000000	6.2232%	0.3096%	7.0367%	0.0451%	7.0496%	0.0591%
	500	1500	13000	97.0410%	96.5487%	97.5993%	97.5826%	97.3777%	97.5570%
			100000	117.0075%	116.2866%	115.5180%	116.4174%	117.4662%	116.4050%
			250000	122.3090%	121.3974%	120.2571%	121.4678%	123.3011%	121.4597%
			500000	87.2257%	86.7603%	85.8641%	86.7968%	90.9155%	86.7884%
			750000	43.2313%	42.9468%	42.4105%	42.9858%	47.2587%	42.9792%
			1000000	4.7689%	0.2690%	5.5643%	0.1003%	13.0966%	0.1049%
	1000	4000	13000	97.4527%	96.6740%	98.0300%	97.4828%	97.4350%	97.4684%
			100000	116.6484%	116.5573%	116.3073%	116.6547%	115.7380%	116.6483%
			250000	121.1249%	121.4732%	121.0291%	121.5241%	121.1361%	121.5189%
			500000	86.5709%	86.7141%	86.2291%	86.7462%	84.5889%	86.7434%
			750000	42.2360%	43.2981%	42.7825%	43.3266%	41.1866%	43.3213%
			1000000	3.9882%	0.1512%	1.8716%	0.0254%	9.5572%	0.0280%
	3000	10000	13000	97.4307%	96.3016%	97.9110%	96.9796%	97.9337%	96.9689%
			100000	116.7706%	116.5981%	116.4915%	116.6553%	116.8366%	116.6514%
			250000	121.3609%	121.3683%	120.3990%	121.3961%	122.0217%	121.3929%
			500000	86.7806%	86.7266%	86.2664%	86.7463%	86.8416%	86.7439%
			750000	43.3183%	42.9662%	42.6049%	42.9868%	43.0292%	42.9838%
			1000000	2.6274%	0.1124%	2.0721%	0.0151%	4.7073%	0.0179%
$Q_{0.2}$	100	100	13000	95.7104%	94.7152%	96.1245%	95.5262%	93.8886%	95.4347%
			100000	97.2905%	95.0136%	96.5091%	95.7706%	99.7509%	95.7355%
			250000	93.3732%	94.1520%	93.7458%	94.4677%	100.2649%	94.4575%
			500000	74.7651%	76.3331%	72.2292%	76.4830%	77.6477%	76.4681%
			750000	35.0596%	37.6170%	34.3294%	37.7158%	40.2090%	37.7033%
			1000000	11.9598%	0.4170%	16.0212%	0.1116%	13.1423%	0.1219%
	500	500	13000	94.1515%	94.5563%	95.7560%	95.6550%	95.0384%	95.6599%
			100000	94.7467%	96.1709%	94.6903%	96.3068%	95.2372%	96.2979%
			250000	95.4295%	94.4002%	91.8276%	94.4655%	92.7303%	94.4566%
			500000	77.3026%	76.6781%	77.5224%	76.7155%	75.1600%	76.7035%
			750000	36.8752%	37.7305%	38.7670%	37.7605%	37.2899%	37.7540%
			1000000	4.1466%	0.1548%	4.7551%	0.0218%	4.4710%	0.0311%
	500	1500	13000	95.2509%	94.8596%	95.0255%	95.4702%	95.2931%	95.4611%
			100000	96.1961%	96.2080%	95.6664%	96.2962%	97.8500%	96.2935%
			250000	95.1130%	94.4541%	93.7919%	94.4859%	95.5499%	94.4823%
			500000	76.8722%	76.7248%	75.7145%	76.7438%	80.8197%	76.7369%
			750000	37.9076%	37.7431%	37.0135%	37.7605%	43.2420%	37.7571%
			1000000	2.8605%	0.1762%	4.2854%	0.0964%	9.9996%	0.0978%
	1000	4000	13000	95.4287%	95.0302%	95.6873%	95.4634%	95.4726%	95.4608%
			100000	96.4122%	96.2477%	95.8464%	96.2967%	95.1327%	96.2969%
			250000	94.2435%	94.4916%	94.2160%	94.5142%	94.2725%	94.5126%
			500000	76.8796%	76.7021%	75.9374%	76.7179%	74.6183%	76.7157%
			750000	37.0820%	38.1551%	37.3332%	38.1647%	36.2944%	38.1628%
			1000000	2.3576%	0.0750%	1.2596%	0.0130%	8.3166%	0.0144%
	3000	10000	13000	95.3607%	95.0484%	95.5628%	95.3695%	95.7306%	95.3670%
			100000	96.3989%	96.2680%	96.1702%	96.2975%	96.2045%	96.2973%
			250000	94.3987%	94.4406%	93.8171%	94.4526%	94.8598%	94.4516%
			500000	76.6726%	76.7082%	76.1367%	76.7194%	76.0145%	76.7177%
			750000	37.9095%	37.7523%	37.4242%	37.7605%	37.7668%	37.7587%
			1000000	1.9434%	0.0558%	1.7446%	0.0076%	2.8740%	0.0098%

Tabelle C.3: Mittlere relative Fehler zu verschiedenen Zeitpunkten mit verschiedenen Selektivitätsschätzern über $CP1_{0.03}$-Daten

C.4 STR-Daten in nicht veränderter Reihenfolge

Die Daten des Typs $STR_{0.51}$ wurden entsprechend den Tests der beiden vorherigen Abschnitte mit sechs verschiedenen Schätzern ausgewertet. Die Reihenfolge der Elemente im Datenstrom ist bei diesen Tests nicht verändert worden. Die Zahl der Inversionen legt es nahe, diesen Datenstrom als nicht sortiert bzw. fast randomisiert anzusehen. Ein Blick auf die Zahl der *Runs* zeigt jedoch eine hohe lokale Sortiertheit in den Daten. Die damit einhergehenden Fehler zu Zeitpunkten vor Verarbeitung aller Datenelemente sind in Tabelle C.4 deutlich zu sehen. Die Spaltenbezeichnungen stimmen mit denen in Abschnitt C.2 überein.

Query	M	M_t	t	sp. KS	adapt. KS	sp. Sampl.	ad. Sampl.	hyb.sp.KS	hyb.ad.KS
$Q_{0.01}$	5000	5000	25000	175.1544%	174.8928%	174.9187%	174.8947%	174.8031%	174.8975%
$Q_{0.01}$	5000	5000	300000	94.5622%	96.2174%	97.2749%	96.5356%	95.9894%	96.3882%
$Q_{0.01}$	5000	5000	900000	36.7297%	36.5174%	39.8992%	38.1721%	40.0844%	38.1419%
$Q_{0.01}$	5000	5000	1400000	25.9216%	24.7039%	28.5804%	24.9292%	26.4572%	24.9378%
$Q_{0.01}$	5000	5000	1888012	12.2509%	10.3384%	7.9622%	0.4277%	10.9238%	0.5907%
$Q_{0.01}$	5000	15000	25000	174.9097%	174.2642%	174.8940%	174.2600%	174.6221%	174.2642%
$Q_{0.01}$	5000	15000	300000	94.0550%	96.1173%	96.9601%	96.5401%	95.7284%	96.3695%
$Q_{0.01}$	5000	15000	900000	35.6950%	37.0201%	39.1300%	38.1682%	40.9364%	38.0785%
$Q_{0.01}$	5000	15000	1400000	23.5939%	23.9798%	26.6344%	24.6043%	27.0770%	24.6028%
$Q_{0.01}$	5000	15000	1888012	10.2662%	8.4363%	5.5568%	1.2534%	10.0775%	1.3547%
$Q_{0.01}$	10000	10000	25000	174.9253%	172.3912%	174.9517%	172.3919%	174.9773%	172.3964%
$Q_{0.01}$	10000	10000	300000	94.0119%	96.1424%	96.8956%	96.5243%	95.4924%	96.4224%
$Q_{0.01}$	10000	10000	900000	36.8343%	36.8402%	38.4121%	38.1605%	38.7805%	38.0745%
$Q_{0.01}$	10000	10000	1400000	24.8391%	24.4061%	24.8544%	24.9281%	24.4551%	24.9374%
$Q_{0.01}$	10000	10000	1888012	9.9051%	9.0457%	6.0448%	0.8054%	7.3960%	0.9583%
$Q_{0.01}$	20000	5000	25000	174.8974%	174.8926%	174.9174%	174.9003%	174.8282%	174.8977%
$Q_{0.01}$	20000	5000	300000	94.5337%	96.2161%	96.9489%	96.5261%	96.3629%	96.3851%
$Q_{0.01}$	20000	5000	900000	37.2775%	36.5164%	37.8062%	38.1600%	37.9321%	38.1327%
$Q_{0.01}$	20000	5000	1400000	24.8470%	24.7191%	24.4513%	24.9352%	26.1417%	24.9446%
$Q_{0.01}$	20000	5000	1888012	9.7641%	10.3467%	4.2072%	0.3390%	5.1208%	0.5247%
$Q_{0.02}$	5000	5000	25000	178.4551%	178.3034%	178.3658%	178.3006%	178.2382%	178.2994%
$Q_{0.02}$	5000	5000	300000	100.1914%	101.4771%	102.1127%	101.6715%	101.0634%	101.5136%
$Q_{0.02}$	5000	5000	900000	33.8931%	34.1699%	37.0530%	35.8966%	37.4281%	35.8658%
$Q_{0.02}$	5000	5000	1400000	23.3432%	23.3002%	27.4732%	24.3421%	25.8232%	24.3532%
$Q_{0.02}$	5000	5000	1888012	6.7567%	5.5383%	5.9319%	0.3056%	8.2496%	0.3655%
$Q_{0.02}$	5000	15000	25000	178.3277%	171.8689%	178.1997%	171.8671%	178.2905%	171.8689%
$Q_{0.02}$	5000	15000	300000	99.9024%	101.4664%	101.6274%	101.6728%	100.8740%	101.5396%
$Q_{0.02}$	5000	15000	900000	33.1649%	34.6868%	36.7975%	35.8975%	38.3187%	35.8865%
$Q_{0.02}$	5000	15000	1400000	21.9098%	23.2274%	25.4017%	24.1810%	25.2221%	24.1781%
$Q_{0.02}$	5000	15000	1888012	5.4991%	4.2607%	3.8010%	1.1017%	7.1893%	1.1225%
$Q_{0.02}$	10000	10000	25000	178.3646%	175.4098%	178.3213%	175.4176%	178.2786%	175.4120%
$Q_{0.02}$	10000	10000	300000	99.8390%	101.4629%	101.9822%	101.6733%	101.1592%	101.5085%
$Q_{0.02}$	10000	10000	900000	34.4493%	34.5102%	36.4544%	35.8886%	36.3454%	35.8380%
$Q_{0.02}$	10000	10000	1400000	23.0192%	23.3259%	24.0637%	24.3446%	23.7606%	24.3483%
$Q_{0.02}$	10000	10000	1888012	4.9618%	4.6096%	4.1988%	0.6983%	5.0859%	0.7286%
$Q_{0.02}$	20000	5000	25000	178.3051%	178.3038%	178.2886%	178.3064%	178.2926%	178.3047%
$Q_{0.02}$	20000	5000	300000	100.2198%	101.4780%	101.9092%	101.6778%	101.5141%	101.5117%
$Q_{0.02}$	20000	5000	900000	34.7982%	34.1709%	35.5802%	35.8918%	36.2785%	35.8645%
$Q_{0.02}$	20000	5000	1400000	22.7505%	23.2991%	23.5652%	24.3429%	25.3940%	24.3558%
$Q_{0.02}$	20000	5000	1888012	5.1209%	5.5377%	2.8777%	0.2879%	3.5236%	0.3482%
$Q_{0.05}$	5000	5000	25000	169.1483%	169.1557%	169.4151%	169.1549%	169.0151%	169.1554%
$Q_{0.05}$	5000	5000	300000	87.2900%	88.1237%	88.6585%	88.3320%	88.2928%	88.3350%
$Q_{0.05}$	5000	5000	900000	28.5668%	28.2527%	29.9709%	29.3791%	30.0187%	29.3845%
$Q_{0.05}$	5000	5000	1400000	22.0633%	21.1975%	23.5291%	22.1065%	22.9717%	22.1081%
$Q_{0.05}$	5000	5000	1888012	3.7306%	2.4091%	4.5707%	0.3342%	5.3190%	0.3468%
$Q_{0.05}$	5000	15000	25000	169.1739%	167.0843%	168.9918%	167.0829%	169.1370%	167.0843%
$Q_{0.05}$	5000	15000	300000	87.5014%	88.1633%	88.7992%	88.3310%	88.4551%	88.3043%
$Q_{0.05}$	5000	15000	900000	27.5993%	28.6294%	30.1082%	29.3791%	30.6923%	29.4068%
$Q_{0.05}$	5000	15000	1400000	20.2559%	21.2521%	22.6654%	21.8679%	22.8290%	21.8689%

Query	M	M_t	t	sp. KS	adapt. KS	sp. Sampl.	ad. Sampl.	hyb.sp.KS	hyb.ad.KS
$Q_{0.05}$	5000	15000	1888012	2.4437%	2.2607%	2.2919%	1.2813%	4.2840%	1.2835%
$Q_{0.05}$	10000	10000	25000	169.2679%	169.0189%	169.1903%	169.0199%	169.2844%	169.0198%
$Q_{0.05}$	10000	10000	300000	87.4130%	88.1534%	88.4459%	88.3286%	88.0474%	88.3118%
$Q_{0.05}$	10000	10000	900000	28.4273%	28.5106%	29.8120%	29.3822%	29.3895%	29.3780%
$Q_{0.05}$	10000	10000	1400000	21.2524%	21.3693%	21.9861%	22.1052%	21.3730%	22.1085%
$Q_{0.05}$	10000	10000	1888012	2.5357%	2.2113%	2.8267%	0.8611%	2.9396%	0.8614%
$Q_{0.05}$	20000	5000	25000	169.1551%	169.1556%	169.1464%	169.1582%	169.1516%	169.1576%
$Q_{0.05}$	20000	5000	300000	87.5605%	88.1233%	88.4701%	88.3272%	88.4050%	88.3300%
$Q_{0.05}$	20000	5000	900000	29.0043%	28.2526%	29.3255%	29.3830%	29.6329%	29.3865%
$Q_{0.05}$	20000	5000	1400000	21.0950%	21.1962%	21.5885%	22.1041%	23.2910%	22.1048%
$Q_{0.05}$	20000	5000	1888012	2.6231%	2.4102%	2.0148%	0.3314%	2.3875%	0.3448%
$Q_{0.1}$	5000	5000	25000	164.7674%	164.7573%	164.8039%	164.7570%	164.6527%	164.7571%
$Q_{0.1}$	5000	5000	300000	88.2126%	88.4670%	88.1623%	88.4736%	89.2653%	88.4780%
$Q_{0.1}$	5000	5000	900000	16.3437%	16.1052%	16.8548%	16.3818%	18.6565%	16.3755%
$Q_{0.1}$	5000	5000	1400000	19.4260%	18.8797%	20.7860%	19.2790%	19.7281%	19.2763%
$Q_{0.1}$	5000	5000	1888012	2.3917%	1.1114%	3.2497%	0.2981%	4.8130%	0.3018%
$Q_{0.1}$	5000	15000	25000	164.7609%	162.0543%	164.7156%	162.0541%	164.7547%	162.0543%
$Q_{0.1}$	5000	15000	300000	88.4801%	88.4641%	88.8065%	88.4742%	87.5736%	88.4751%
$Q_{0.1}$	5000	15000	900000	15.5201%	16.1756%	16.5366%	16.3821%	16.6553%	16.3808%
$Q_{0.1}$	5000	15000	1400000	18.1323%	19.0141%	19.3761%	19.2638%	19.8889%	19.2597%
$Q_{0.1}$	5000	15000	1888012	1.3851%	1.2779%	1.4755%	1.2158%	2.8018%	1.2133%
$Q_{0.1}$	10000	10000	25000	164.7918%	163.5706%	164.7657%	163.5718%	164.7994%	163.5709%
$Q_{0.1}$	10000	10000	300000	88.1341%	88.4650%	88.6919%	88.4752%	88.4050%	88.4801%
$Q_{0.1}$	10000	10000	900000	16.1954%	16.1521%	16.4132%	16.3814%	15.9451%	16.3714%
$Q_{0.1}$	10000	10000	1400000	18.7660%	18.9823%	19.6447%	19.2772%	19.3132%	19.2744%
$Q_{0.1}$	10000	10000	1888012	1.6177%	1.1528%	2.1034%	0.8189%	2.0938%	0.8187%
$Q_{0.1}$	20000	5000	25000	164.7576%	164.7572%	164.7552%	164.7578%	164.7697%	164.7576%
$Q_{0.1}$	20000	5000	300000	88.4048%	88.4670%	88.3058%	88.4742%	88.3552%	88.4778%
$Q_{0.1}$	20000	5000	900000	16.5902%	16.1050%	16.5505%	16.3814%	16.4604%	16.3750%
$Q_{0.1}$	20000	5000	1400000	18.8837%	18.8794%	18.7602%	19.2773%	20.0363%	19.2752%
$Q_{0.1}$	20000	5000	1888012	1.6673%	1.1115%	1.3081%	0.2987%	1.3128%	0.3021%
$Q_{0.2}$	5000	5000	25000	151.0528%	151.0506%	151.1141%	151.0507%	151.0067%	151.0507%
$Q_{0.2}$	5000	5000	300000	81.5943%	81.9949%	81.7483%	81.9911%	82.5481%	81.9919%
$Q_{0.2}$	5000	5000	900000	11.8177%	11.8730%	10.8298%	11.8775%	13.9879%	11.8784%
$Q_{0.2}$	5000	5000	1400000	17.1023%	16.7977%	18.4874%	16.9489%	17.0040%	16.9480%
$Q_{0.2}$	5000	5000	1888012	1.4822%	0.5949%	2.3443%	0.2639%	3.4746%	0.2654%
$Q_{0.2}$	5000	15000	25000	151.0504%	149.8838%	151.0143%	149.8840%	151.0548%	149.8838%
$Q_{0.2}$	5000	15000	300000	82.1213%	81.9946%	82.3190%	81.9916%	81.2254%	81.9954%
$Q_{0.2}$	5000	15000	900000	11.2645%	11.8758%	11.5896%	11.8775%	12.3132%	11.8815%
$Q_{0.2}$	5000	15000	1400000	16.4171%	16.7462%	17.0710%	16.8397%	17.8662%	16.8388%
$Q_{0.2}$	5000	15000	1888012	0.7310%	1.1341%	0.7165%	1.0690%	2.2018%	1.0682%
$Q_{0.2}$	10000	10000	25000	151.0783%	150.7785%	151.0600%	150.7783%	151.0673%	150.7785%
$Q_{0.2}$	10000	10000	300000	81.6088%	81.9948%	82.0529%	81.9932%	82.0689%	81.9956%
$Q_{0.2}$	10000	10000	900000	11.8903%	11.8749%	11.6198%	11.8773%	11.5749%	11.8765%
$Q_{0.2}$	10000	10000	1400000	16.2135%	16.8306%	17.4980%	16.9484%	16.8009%	16.9481%
$Q_{0.2}$	10000	10000	1888012	1.2612%	0.8442%	1.7328%	0.7302%	1.1255%	0.7283%
$Q_{0.2}$	20000	5000	25000	151.0504%	151.0505%	151.0499%	151.0505%	151.0549%	151.0504%
$Q_{0.2}$	20000	5000	300000	82.0613%	81.9951%	81.8311%	81.9927%	81.8972%	81.9931%
$Q_{0.2}$	20000	5000	900000	12.3340%	11.8731%	12.3807%	11.8776%	11.6005%	11.8793%
$Q_{0.2}$	20000	5000	1400000	17.1470%	16.7980%	16.5383%	16.9484%	16.9353%	16.9474%
$Q_{0.2}$	20000	5000	1888012	0.8704%	0.5946%	0.8841%	0.2651%	0.7745%	0.2662%

Tabelle C.4: Mittlere relative Fehler zu verschiedenen Zeitpunkten mit verschiedenen Selektivitätsschätzern über $STR_{0.51}$-Daten

Abbildungsverzeichnis

2.1 Operatorbaum über drei Datenquellen mit Aggregation 7

3.1 Mögliche GUI zur Unterstützung von Online Aggregation. Nachbildung der *POSTGRES95* Oberfläche aus [HHW97] 22

4.1 Mittlere Laufzeit der *Reservoir Sampling* Techniken vom Typ R, X und Z 45
4.2 Mittlerer relativer Fehler zum Mittelwert der *Reservoir Sampling* Techniken vom Typ R, X und Z 46
4.3 Speicherbasierte induktive Statistik auf einem Datenstrom zum Zeitpunkt k 47
4.4 Epanechnikow Kern 51
4.5 Biweight Kern 51
4.6 Triweight Kern 51
4.7 Kosinus Kern 51
4.8 Epanechnikow Kern mit transformierten Kernen 61
4.9 Biweight Kern mit transformierten Kernen 61
4.10 *Difference Kernel Estimator* der Dichte einer Stichprobe der Größe $n = 2000$ einer gemäß$N(0,1)$ verteilten Grundgesamtheit mit wahrer Dichte f 62

5.1 Segmentierung der Daten des Datenstroms in Blöcke der Größe k . . . 68
5.2 Laufende Gewichte zum Zeitpunkt $j = 12$ 71
5.3 Summe der laufenden Gewichte zum Zeitpunkt $j = 12$ 72
5.4 Approximation einer reellen Funktion mit Splinefunktionen unterschiedlicher Auflösung 81
5.5 Fehler der Approximation einer reellen Funktion mit Splinefunktionen unterschiedlicher Auflösung 82
5.6 Adaptive größenbeschränkte Statistik zum Zeitpunkt t über einem Datenstrom 84
5.7 Hierarchie der benötigten Instanzen einer adaptiven Aggregation über kernbasierten Verfahren 89

5.8 Gespiegelte speicherbasierte Dichteschätzer über normalverteilten Daten mit unterschiedlichen Speichergrößen 100
5.9 Einfacher speicherbasierter Dichteschätzer an vier Zeitpunkten über verschieden sortierten Datensätzen 102
5.10 Relative Berechnungszeiten verschiedener Bandbreiten auf logarithmischer Skala 103
5.11 Gespiegelter adaptiver und komprimierter Dichteschätzer an vier Zeitpunkten über verschieden sortierten Datensätzen mit arithmetischer Gewichtung 105
5.12 Gespiegelter adaptiver und komprimierter Dichteschätzer an vier Zeitpunkten über verschieden sortierten Datensätzen mit geometrischer Gewichtung 107
5.13 Gespiegelter adaptiver und komprimierter Dichteschätzer an vier Zeitpunkten über teilsortierten Daten mit unterschiedlicher Gewichtung . . 108
5.14 Gespiegelte adaptive und komprimierte Dichteschätzer an sechs Zeitpunkten über modifizierten normalverteilten Daten mit unterschiedlicher Gewichtung 110
5.15 Speicherbasierte und adaptive *Difference Kernel Estimator* über verschiedenen Daten zu festem Speicher M=5000 111
5.16 Vergleich der angewandten Schätzer über Daten mit Sprungstellen . . . 112
5.17 Relativer Fehler eines hybriden speicherbasierten Kernselektivitätsschätzers 113
5.18 Relativer Fehler des adaptiven Sampling-Schätzers zu unterschiedlichen Speichergrößen über CP1 und Anfragen $Q_{0.02}^{CP1}$ 114
5.19 Mittlere relative Fehler über $N(0,1)_{0.5}$-Daten zu 2% Anfragen an sechs Zeitpunkten 115
5.20 Mittlere relative Fehler der Schätzer auf $STR_{0.51}$-Daten zum Zeitpunkt $t = 1\,888\,012$ zu verschiedenen Speichergrößen 117

A.1 Dichte der Daten vom Typ CP1 128
A.2 Dichte der Daten vom Typ CP2 129
A.3 Geschätzte Dichte der Daten vom Typ STR 130

Tabellenverzeichnis

3.1 Konfidenzintervalle zu den wichtigsten klassischen Aggregaten 32
3.2 Klassische Aggregate mit ihren laufenden und a priori benötigten Größen 33

4.1 Performanz der Verfahren R, X, Y und Z 44
4.2 In XXL verfügbare eindimensionale Kernfunktionen 50
4.3 Vergleich der Konvergenzgeschwindigkeiten zwischen Kernschätzern und Histogrammen (Quelle: [Här91] Seite 62) 53

5.1 Relative Berechnungszeiten verschiedener Bandbreiten über verschieden großen Stichproben 104

A.1 Statistische Eigenschaften der Testdatensätze 131
A.2 Maße für den Grad der Teilsortiertheit der Testdaten 132

C.1 Mittlere relative Fehler zu verschiedenen Zeitpunkten mit verschiedenen Selektivitätsschätzern über $N(0,1)_{0.5}$-Daten 144
C.2 Mittlere relative Fehler zu verschiedenen Zeitpunkten mit verschiedenen Selektivitätsschätzern über $CP2_{0.35}$-Daten 147
C.3 Mittlere relative Fehler zu verschiedenen Zeitpunkten mit verschiedenen Selektivitätsschätzern über $CP1_{0.03}$-Daten 149
C.4 Mittlere relative Fehler zu verschiedenen Zeitpunkten mit verschiedenen Selektivitätsschätzern über $STR_{0.51}$-Daten 151

Literaturverzeichnis

[Bar91] Vic Barnett: *Sample Survey Principles and Methods.* Edward Arnold, London Melbourne Auckland, 1991.

[BB98] Frank Baur und Günter Bamberg: *Statistik.* Oldenbourg, 10. Auflage, 1998.

[BBD+01] Jochen van den Bercken, Björn Blohsfeld, Jens-Peter Dittrich, Jürgen Krämer, Tobias Schäfer, Martin Schneider und Bernhard Seeger: *XXL - A Library Approach to Supporting Efficient Implementations of Advanced Database Queries.* In: Peter M. G. Apers, Paolo Atzeni, Stefano Ceri, Stefano Paraboschi, Kotagiri Ramamohanarao und Richard T. Snodgrass (Herausgeber): *VLDB 2001, Proceedings of 27th International Conference on Very Large Data Bases, September 11-14, 2001, Roma, Italy*, 2001.

[BKS99] Björn Blohsfeld, Dieter Korus und Bernhard Seeger: *A Comparison of Selectivity Estimators for Range Queries on Metric Attributes.* In: Alex Delis, Christos Faloutsos und Shahram Ghandeharizadeh (Herausgeber): *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadephia, Pennsylvania, USA*, Seiten 239–250. ACM Press, 1999.

[Blo98] Björn Blohsfeld: *Selektivitätsschätzung von mehrdimensionalen Datenbankanfragen mit Kernschätzern.* Diplomarbeit, University of Marburg, Marburg, 1998.

[Bos86] Karl Bosch: *Angewandte Statistik.* Vieweg, Braunschweig, 1986.

[BT94] Herbert Büning und Götz Trenkler: *Nichtparametrische statistische Methoden.* Walter de Gruyter, Berlin, New York, 2. Auflage, 1994.

[BW01] Shivnath Babu und Jennifer Widom: *Continuous Queries over Data Streams.* SIGMOD Record, 30(3):109–120, 2001.

[CGL83] T.F. Chan, G.H. Golub und R.J. LeVeque: *Algorithms for Computing the Sample Variance: Analysis and Recommendations.* The American Statistician, 37:242–247, 1983.

[Col] *The Colt Distribution - Open Source Libraries for High Performance Scientific and Technical Computing in Java URL: http://tilde-hoschek.home.cern.ch/%7Ehoschek/colt/index.htm.* Online im Internet [Stand: März 2002], 2002. Version 1.0.2.

[DD93] C. J. Date und H. Darwen: *A Guide to The SQL Standard.* Addison-Wesley, 3. Auflage, 1993.

[Dev86] Luc Devroye: *Non-Uniform Random Variate Generation.* Springer Verlag, New York, 1986.

[DG01] C. Domeniconi und D. Gunopulos: *An Efficent Approach for Approximation Multi-dimensional Range Queries and Nearest Neighbor Classification in Large Datasets*, 2001.

[DH93] P. Deuflhard und A. Hohmann: *Numerische Mathematik I.* Walter de Gruyter, Berlin, New York, 2. Auflage, 1993.

[DH01] P. Domingos und G. Hulten: *Catching Up with the Data: Research Issues in Mining Data Streams*, 2001.

[ECW92] Vladimir Estivill-Castro und Derick Wood: *A Survey of Adaptive Sorting Algorithms.* ACM Computing Surveys, 24(4):441–476, 1992.

[GKTD00] Dimitrios Gunopulos, George Kollios, Vassilis J. Tsotras und Carlotta Domeniconi: *Approximating Multi-Dimensional Aggregate Range Queries over Real Attributes.* In: Weidong Chen, Jeffrey F. Naughton und Philip A. Bernstein (Herausgeber): *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA*, Band 29, Seiten 463–474. ACM, 2000.

[Haa97] Peter J. Haas: *Large-Sample and Deterministic Confidence Intervals for Online Aggregation.* In: Yannis E. Ioannidis und David M. Hansen (Herausgeber): *Ninth International Conference on Scientific and Statistical Database Management, Proceedings, August 11-13, 1997, Olympia, Washington, USA*, Seiten 51–63. IEEE Computer Society, 1997.

[Han98] David J. Hand: *Data Mining: Statistics and More?* The American Statistician, 52(2):112–118, 1998.

[Han99] David J. Hand: *Statistics and Data Mining: Intersecting Disciplines.* SIGKDD: SIGKDD Explorations: Newsletter of the Special Interest Group (SIG) on Knowledge Discovery & Data Mining, ACM, 1, 1999.

[HAR00] Joseph M. Hellerstein, Ron Avnur und Vijayshankar Raman: *Informix under CONTROL: Online Query Processing.* Data Mining and Knowledge Discovery, 4(4):281–314, 2000.

[Hel96] Joseph M. Hellerstein: *The Case for Online Aggregation: New Challenges in User Interfaces, Performance Goals, and DBMS Design*, 1996.

[Hel97] Joseph M. Hellerstein: *Data Engineering Bulletin*, 1997.

[HHW97] Joseph M. Hellerstein, Peter J. Haas und Helen Wang: *Online Aggregation.* In: Joan Peckham (Herausgeber): *SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data, May 13-15, 1997, Tucson, Arizona, USA*, Seiten 171–182. ACM Press, 1997.

[HK98] Alexander Hinneburg und Daniel A. Keim: *An Efficient Approach to Clustering in Large Multimedia Databases with Noise.* In: *Knowledge Discovery and Data Mining*, Seiten 58–65, 1998.

[Hoe63] W. Hoeffding: *Probability inequalities for sums of bounded variables*, 1963.

[HÖT88] Wen-Chi Hou, Gultekin Özsoyoglu und Baldeo K. Taneja: *Statistical Estimators for Relational Algebra Expressions.* In: *Proceedings of the Seventh ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, March 21-23, 1988, Austin, Texas*, Seiten 276–287. ACM Press, 1988.

[HÖT89] Wen-Chi Hou, Gultekin Özsoyoglu und Baldeo K. Taneja: *Processing Aggregate Relational Queries with Hard Time Constraints.* In: James Clifford, Bruce G. Lindsay und David Maier (Herausgeber): *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data, Portland, Oregon, May 31 - June 2, 1989*, Seiten 68–77. ACM Press, 1989.

[Här91] W. Härdle: *Smoothing Techniques with Implementations in S.* Springer, New York, 1991.

[HSJM91] P. Hall, S. Sheather, M. Jones und J. Marron: *An optimal data-based bandwidth selection in kernel density estimation*, 1991.

[Kau89] Ernst Kausen: *Numerische Mathematik mit Turbo-Pascal.* Dr. Alfred Hüthig Verlag GmbH, Heidelberg, 1989.

[KJJ99] Flip Korn, Theodore Johnson und H. V. Jagadish: *Range Selectivity Estimation for Continuous Attributes.* In: *Statistical and Scientific Database Management*, Seiten 244–253, 1999.

[Knu98] Donald E. Knuth: *The Art of Computer Programming, Vol. 3*, 1998.

[Kor99] Dieter Korus: *Selektivitätsschätzung von Bereichsanfragen auf metrischen Attributen mit nichtparametrischen Verfahren.* Doktorarbeit, Philipps-Universität Marburg, Marburg, 1999.

[LNS90] Richard J. Lipton, Jeffrey F. Naughton und Donovan A. Schneider: *Practical selectivity estimation through adaptive sampling.* Seiten 1–11, 1990.

[LNSS93] Richard J. Lipton, Jeffrey F. Naughton, Donovan A. Schneider und S. Seshadri: *Efficient sampling strategies for relational database operations.* Theoretical Computer Science, 116(1):195–226, 1993.

[Man97] Heikki Mannila: *Methods and Problems in Data Mining.* In: *ICDT*, Seiten 41–55, 1997.

[Mül92] H.G. Müller: *Change-points in nonparametric regression analysis.* Annals of Statistics, 20:737–761, 1992.

[Olk93] Frank Olken: *Random Sampling from Databases.* Technischer Bericht LBL-32883, Mailstop 50B-3238, 1 Cyclotron Road, Berkeley, California 94720, U.S.A., 1993.

[OMM+02] Liadan O'Callaghan, Nina Mishra, Adam Meyerson, Sudipto Guha und Rajeev Motwani: *Streaming-Data Algorithms For High-Quality Clustering.* In: *ICDE*, 2002.

[OR95] F. Olken und D. Rotem: *Random Sampling from Databases - A Survey*, 1995.

[Poo97] Viswanath Poosala: *Histogram-Based Estimation Techniques in Database Systems.* Doktorarbeit, Madison, Wisconsin, U.S.A., 1997.

[QAL91] P. Qiu, C. Asano und X. Li: *Estimation of jump regression function.* Bulletin of Informatics and Cybernetics, 24:197–212, 1991.

[Qiu94] P. Qiu: *Estimation of the number of jumps of the jump regression functions.* 1994.

[Ras95] D. Rasch: *Mathematische Statistik.* Johann Ambrosius Barth, Heidelberg, Leipzig, 1. Auflage, 1995.

[Sch86] H. G. Schwarz: *Numerische Mathematik I.* B. G. Teubner, Stuttgart, 1. Auflage, 1986.

[Sco92] D. W. Scott: *Multivariate Density Estimation.* John Wiley & Sons, Inc. , New York, 1992.

[Sti76] E. Stiefel: *Einführung in die numerische Mathematik.* B. G. Teubner, Stuttgart, 5. Auflage, 1976.

[Sto89] Josef Stoer: *Numerische Mathematik I.* Springer, 3. Auflage, 1989.

[Tur93] Berwin A. Turlach: *Bandwidth Selection in Kernel Density Estimation: A Review*, 1993.

[Vit85] Jeffrey Scott Vitter: *Random Sampling with a Reservoir.* ACM Transactions on Mathematical Software, 11(1):37–57, 1985.

[WJ95] M. P. Wand und M. C. Jones: *Kernel Smoothing.* Monographs on Statistics and Applied Probability. Chapman & Hall, London, 1995.

[XXL] Philipps University of Marburg: The Database Research Group: *XXL- eXtensible and fleXible Library. URL: http://www.mathematik.uni-marburg.de/DBS/xxl/.* Online im Internet [Stand: März 2002], 2002.

Index

Adaptiver Schätzer, 67, 85
    Fehler, 86
    in XXL, 87
    komprimierter, 85
Aggregat, 57
Aggregationsfunktion, 17, 99
    `EmpiricalCDFAggregateFunction`, 99
    `LastNthAverage`, 17
    `Minimum`, 17
    `ReservoirSample`, 43
Arbeitsphase, *siehe* Stichprobe
Arithmetische Gewichtung, *siehe* Gewichtsfunktion
Assoziationsregel, 11
Asymptotisches Konfidenzintervall, 54

Bandbreite, *siehe* Kernschätzer
    Einsetzungsregel, 52
    in XXL, 64
    Kreuzvalidierung, 52
    Normalskalierungsregel, 51, 103
    Überglättungsregel, 51, 103, 104
Biweightkern, *siehe* Kernfunktion

Data Mining, 8, 12, 13, 121
Datenquelle
    aktive, 6
    passive, 7
Datenstrom, 12, 45, 67, 109
Degressive Gewichtung, *siehe* Gewichtsfunktion
Despriptive Statistik, 10
Dichte, 60
Difference Apart Kernel Estimator, 62, 109
Difference Kernel Estimator, *siehe* Sprungstellenschätzer
diskret, 9

Einsetzungsregel, *siehe* Bandbreite
Empirische Verteilungsfunktion, 97
Epanechnikowkern, *siehe* Kernfunktion

Fensteranfrage, 57

Gausskern, *siehe* Kernfunktion
Geometrische Gewichtung, *siehe* Gewichtsfunktion
Gespiegelter Kernschätzer, 54
    Randkern, 54
Gespiegelter Kernselektivitätsschätzer, *siehe* Kernselektivitätsschätzer
Gewichtsfunktion
    arithmetische, 71
    degressive, 73
    geometrische, 72
    laufende, 74
    logarithmische, 74
    progressive, 73
Grundgesamtheit, 13, 57

Hermite-Polynom, 53
Histogramm
    Max-Diff, 36
Histogramme, 55
Hybrider Kerndichteschätzer, *siehe* Kerndichteschätzer
Hybrider Kernselektivitätsschätzer, *siehe* Kernselektivitätsschätzer

Indikatorfunktion, 58
Induktive Statistik, 10
Initialisierungsphase, *siehe* Stichprobe

Java, 14

Kerndichteschätzer, 47

Hybrider Kerndichteschätzer, 55
in XXL, 64
Kernfunktion, 49, 60
Biweightkern, 50
Epanechnikowkern, 50
Gausskern, 50, 58
in XXL, 63
Kosinuskern, 50
roughness, 50, 103
transformierte, 60
Triweightkern, 50
Kernschätzer, 47
Bandbreite, 49
Bias, 48
speicherbasierte, 65
Varianz, 48
Kernselektivitätsschätzer, 57
gespiegelter, 58
hybrider, 59
in XXL, 64
nativer, 57
Konfidenzintervall, 53
Konfidenzlevel, 55
Konsistenz, 54, 61
Konvergenzgeschwindigkeit, 55
Konvergenzgeschwindigkeiten, 53
Kosinuskern, *siehe* Kernfunktion
Kreuzvalidierung, *siehe* Bandbreite

Laufende Gewichtung, *siehe* Gewichtsfunktion
Logarithmische Gewichtung, *siehe* Gewichtsfunktion

Meßniveau, 8

Nativer Kernselektivitätsschätzer, *siehe* Kernselektivitätsschätzer
Nichtparametrische Statistik, 11
Normalskalierungsregel, *siehe* Bandbreite

Online Aggregation, 22
Klassische Aggregate in XXL, 33
Mittelwert, 27
Standardabweichung, 32
Summe, 29
Varianz, 30
Voraussetzungen zur, 26
Zugriffsverfahren zur, 23
Operatorbaum, 7, 57

packages, 14
Parametrische Statistik, 11
Progressive Gewichtung, *siehe* Gewichtsfunktion

Randkern, *siehe* Gespiegelter Kernschätzer
Randproblem, 53
Randspiegelung, 53
ReservoirSampler, *siehe* XXL
roughness, *siehe* Kernfunktion

Selektivität, 57
Skala, 8
Speicherbasierte Kernschätzer, *siehe* Kernschätzer
Speicherbedarf, *siehe* Speicherplatz
Speicherplatz, 92
Splinegitter, 78
Splineinterpolation, 76
natürliche, 80
vollständige, 80
Splinekomprimierung, 75, 81
Splines, 75
Bézier-Bernstein-Form, 78
Bézier-Koeffizienten, 79
in XXL, 82
kubische, 77
Sprungstelle, 56, 59
Sprungstellenschätzer, 59
Difference Kernel Estimator, 59, 61, 91, 109
SQL, 25
Standardnormalverteilung, 54
Quantil, 54
stetig, 9
Stichprobe, 36, 99
Arbeitsphase, 37
Initialisierungsphase, 37

speicherbasierte, 37, 43
Ziehen einer, 38–42
Ziehen einer Stichprobe mit XXL , 43

Transformierte Kernfunktion, *siehe* Kernfunktion
Triweightkern, *siehe* Kernfunktion

Überglättungsregel, *siehe* Bandbreite

verteilungsfrei, 99
Verteilungsfunktion
empirische, *siehe* Empirische Verteilungsfunktion

XXL, 14, 43, 63
`ReservoirSampler`, 44
`cursors`, 16
`functions`, 15
`predicates`, 15
`statistics.kernels`, 19
`statistics`, 18
Aggregator-Cursor, 17, 43
Version, 15

Zwischenergebnisse, 7

Lebenslauf

Björn Blohsfeld
Lange Meile 10a
61352 Bad Homburg v.d.H.

Geburtsdatum: 28. April 1971
Familienstand: ledig
Staatsangehörigkeit: deutsch

August 1977 - Juni 1987:	Besuch der Grundschule Ulmtal, Greifenstein Holzhausen/Allendorf, der Förderstufenschule Biskirchen, Leun/Biskirchen und der Gesamtschule Solms, Solms/Burgsolms
Juli 1987 - Juni 1990:	Besuch der Gymnasialen Oberstufenschule Goetheschule Wetzlar, Wetzlar/Lahn
Juni 1990:	Abschluss: Allgemeine Hochschulreife (Abitur)
Juli 1990 - Juni 1991:	Ableistung des Grundwehrdienstes
Juli 1991 - September 1991:	Feste Anstellung auf Zeit im Sägewerk Rink, Holzhausen
Oktober 1991 - Mai 1998:	Studium der Wirtschaftsmathematik an der Philipps-Universität Marburg, Marburg/Lahn
Mai 1998:	Abschluss: Diplom-Wirtschaftsmathematiker
September 1998 - August 2001:	Wissenschaftlicher Mitarbeiter an der Philipps-Universität Marburg, Marburg/Lahn am Fachbereich Mathematik und Informatik im Fachgebiet Datenbanken, Mitarbeit am Forschungsprojekt: *Massenverarbeitung von Anfragen in Geo- und Zeitdatenbanken*
September 2001 - Februar 2002:	Wissenschaftlicher Mitarbeiter an der Philipps-Universität Marburg, Marburg/Lahn am Fachbereich Mathematik und Informatik in einem Sonderprojekt zur Unterstützung der Lehre