Datenstromalgorithmen für Regression

Diplomarbeit
im Studiengang Angewandte Informatik

Fakultät für Informatik
Technische Universität Dortmund

vorgelegt von
Qingchui Zhu

28. März 2012

Gutachter:
Prof. Dr. Christian Sohler
Melanie Schmidt

Inhaltsverzeichnis

Abbildungsverzeichnis

Zusammenfassung

Das Ziel der Diplomarbeit ist die Implementierung der Datenstromalgorithmen für Regression. Es wird die Lösung von $Ax = B$ berechnet, wobei A eine $n \times m$-Matrix und B eine $n \times m'$-Matrix ist. Die empfangenen Daten A und B der Datenströme werden durch Time Series Modell oder Turnstile (Drehkreuz) Modell aktualisiert.

Im Kapitel 2 werden einige klassische Verfahren für Regression vorgestellt. Häufig können große Datenmengen von Datenströmen mit klassischen Algorithmen nicht behandelt werden, weil der zur Verfügung stehende Speicher oft beschränkt ist.

Im Kapitel 3 werden drei Datenstromalgorithmen für Regression erklärt. Zunächst wird über zwei Algorithmen im Time Series Modell diskutiert. Die erste Methode basiert auf Gauss'sche Normalgleichung, die von Q. Zhou im Jahr 2008 [43] verwendet wird. Das zweite Datenstromalgorithmus wird in der vorliegenden Arbeit entwickelt. Es arbeitet mit der QR-Zerlegung. Danach wird ein Datenstromalgorithmus im Turnstile Modell vorstellt. Dieses Algorithmus wird von K. L. Clarkson und D. P. Woodruff im Jahr 2009 in der Literatur [6] veröffentlicht.

Im Kapitel 4 werden einige Untersuchungen durchgeführt, um die Genauigkeit der Theorien zu überprüfen und die Algorithmen zu vergleichen.

Kapitel 1

Einleitung und Einordnung der Arbeit

1.1 Datenströme und Datenstromalgorithmen

Durch die Anwendung von Sensoren zur Messung und Kontrolle in der Industrie und die gestiegene Vernetzung von Rechnern insbesondere durch das Internet entsteht das Problem der Analyse riesiger Datenmengen, die in Form von Datenströmen auftreten [28]. Ein Datenstrom ist eine kontinuierliche Folge von digitalen Signalen, die gesendete Informationen darstellen [12]. Typischerweise werden Signale als Punkte, Tupel von Punkten, die durch Zahlen oder Zeichen dargestellt [12].

In den letzten Jahren sind immer mehr Anwendungen auf große Datenmengen, die in Form von Datenströmen auftreten, entstanden. Beispielsweise fallen sehr große Datenmengen an, wenn Netzwerkdatenverkehr durch Rechner überwacht, die empfangenen Sensordaten verarbeitet oder die Trends der Aktienkurse berechnen. Die großen Datenmengen passen häufig nicht in den Hauptspeicher eines Rechners und sind dadurch nicht komplett speicherbar [12]. Die Datenelemente werden meist in rascher Folge erzeugt, wodurch die Menge der Datensätze pro Zeiteinheit variiert [12]. Häufig können derartig große Datenmengen mit klassischen Algorithmen nicht mehr bearbeitet werden. Daher werden spezielle Algorithmen benötigt, die mit einem oder wenigen Durchlauf über die Daten und mit sehr wenig Speicher auskommen müssen [12]und somit oft nur eine Approximation der Lösung berechnen können [2]. Solche Algorithmen werden auch als Datenstromalgorithmen bezeichnet.

In dieser Arbeit wird die Methode der kleinsten Quadrate zur Analyse von Datenströmen untersucht. Beim Entwurf des Algorithmus spielen Randomisierung und Hashing eine wichtige Rolle [2].

Für die Anwendung von großen Datenmengen werden Datenstromalgorithmen erfordert, wie zum Beispiel bei der Erfassung von Routingdaten in Netzwerken, Aufzeichnungen von Telekommunikationsdaten, Banktransaktionen, Börsentickern usw. [24] .

1.2 Modelle der Datenströme

Der typische Verlauf eines Datenstromalgorithmus wird in Abbildung 1.2.1 veranschaulicht. Der Input (deutsch: Dateneingang) in Abbildung 1.2.1 ist in Form von

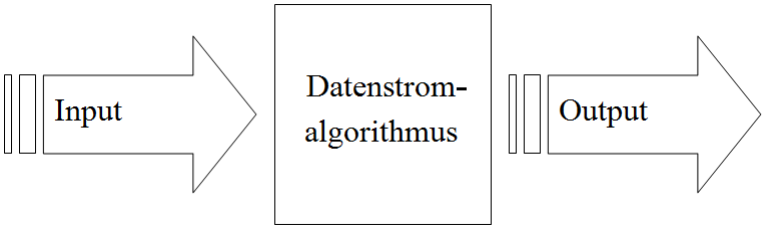

Abbildung 1.2.1: Der typische Verlauf eines Datenstromalgorithmus [43].

Strom dargestellt:

$$a_1, a_2, \cdots, a_i, \cdots .$$

Häufig wird der Datenstromalgorithmusverlauf von drei Modellen (Time Series Modell, Cash Register und Turnstile) beschrieben. Die drei Datenstrommodelle unterscheiden sich in dem, wie sie die empfangenen Daten A durch $a_i's$ aktualisieren:

- a) Time Series Modell: Das Signal kommt immer in der Reihenfolge des Index. Das aktuell zu empfangende Element hat immer die höchste Indexnummer. So werden die empfangenen Daten A nach $(A[i] = a_i)$ aktualisiert. Das Model ist am besten geeignet, wenn die Daten im festgelegten Zeitintervallen ankommen [12]. Der Windsensor misst zum Beispiel die Windgeschwindigkeit regelmäßig einmal pro Sekunde.

- b) Cash Register (Registrierkasse) Modell: Hier ist a_i ist ein Tupel: $a_i = (j, I_i)$, wobei $I_i \geq 0$. Mit: $A_i[j] = A_{i-1}[j] + I_i$ werden die empfangenen Daten A aktualisiert, wobei A_i der Signalzustand nachdem das i-te Element des Stromes gesehen wurde [12]. Zum Bsp.: Wie viele Bücher wurden verkauft? Die Bücher (Harry Potter) haben die Indexnummer j und wurden gerade Anzahl I_i verkauft. $A_i[j]$ ist die Anzahl der bis jetzt verkauften (Harry Potter) Bücher.

- c) Turnstile (Drehkreuz) Modell: a_i ist hier wie nach Cash Register Modell ein Tupel $a_i = (j, I)$, wobei $I \neq 0$. Die empfangenen Daten A werden aktualisiert nach $A[j] = A[j] + I$ [12]. Zum Bsp.: Wie viele Frauen und Männer sind im Wartezimmer. In diesem Beispiel ist die Indexnummer von Frauen $j = 1$ und die Indexnummer von Männern $j = 2$. Wenn ein Mann ins Wartezimmer kommt (d.h. $a_i = (j, I) = (2, 1)$), dann wird nach $A[2] = A[2] + 1$ berechnet.

Die empfangenen Daten A sind sehr groß und somit häufig nicht speicherbar. Während der Aktualisierung müssen sie sofort verarbeitet werden. Die Diplomarbeit dient zur Analyse der Datenstromalgorithmen für Regression. Die empfangenen Daten sind hier die Matrix A und der Vektor b in der Gleichung $Ax = b$.

1.3 Einordnung der vorliegenden Arbeit

Die vorliegende Diplomarbeit beschäftigt sich mit der Implementierung und Untersuchung der Datenstromalgorithmen für Regression.

Im ersten Teil der Arbeit (Kap. 2) werden die mathematischen Grundlagen und die generellen Algorithmen der Regression vorgestellt.

Die Theorien der Datenstromalgorithmen für Regression werden im zweiten Teil der Arbeit (Kap. 3) erläutert. Dabei werden zwei Algorithmen im Time Series Modell (Algorithmus von Zhou und QR- Datenstromalgorithmus) und ein Algorithmus im Turnstile Modell (Clarkson-Woodruff Algorithmus) erklärt, wobei das QR-Datenstromalgorithmus in der vorliegenden Arbeit entwickelt wird.

Der dritte Teil der Arbeit (Kap. 4) ist die experimentelle Untersuchung über die klassischen Methoden und Datenstromalgorithmen für Regression. In diesem Teil werden die Genauigkeit der Theorien und die Eigenschaften der Datenstromalgorithmen überprüft. Für das Clarkson-Woodruff Algorithmus werden desweiteren die Reihung der Hashfunktionen untersucht.

Mit der Zusammenfassung und dem Ausblick in Kapitel 5 wird die Arbeit abgeschlossen. Zusätzlich wird eine kurze Beschreibung der Implementierung im Anhang angefügt.

Kapitel 2

Klassische Algorithmen für Regression

Um ein physikalisches Problem zu lösen, muss dieses zuerst in eine mathematische Funktionen mit freien Parametern formuliert werden. Die mathematische Funktion wird als theoretische Funktionen bezeichnet. Die freien Parametern können durch eine Anpassung an die Messgröße empirisch bestimmt werden. Für die Anpassung wird die Methode der kleinsten Quadrate (MKQ) verwendet [13]. Die Bestimmung der Parameter erfolgt durch die möglichst nahen Anpassung der Funktion an die empirischen ermittelten Datenpunkte. D. h. es wird die Summe der quadratischen Abstände zwischen der Kurve der Funktion und der gegebenen Datenpunkten minimiert [36].

2.1 Methode der kleinsten Quadrate

2.1.1 Problemstellung der Regression

In einem Experiment ist das Ziel einer jeden Messung, den wahren Wert einer physikalischen Größe zu bestimmen. Wenn aber mehrere Messungen mit derselben oder einer gleichgearteten Messapparatur oder auch nach einem anderen Messverfahren durchgeführt werden, werden unterschiedliche Ergebnisse durch Messfehler auftreten [1]. Es erhebt sich nun die Frage, wie man relative gute Ergebnisse durch die Messungen berechnen kann.

2.1.2 Methode der kleinsten Quadrate

Zur Problemlösung können die Ergebnisse durch die "Methode der kleinsten Quadrat" berechnet werden. Die Methode der kleinsten Quadrate dient zur Minimierung der quadratischen Fehler. Ein konkreter Berechnungsfall: Es werden viele Wertepaare

$$(t_1, b_1), (t_2, b_2), \cdots, (t_n, b_n)$$

gemessen. Die Beziehung zwischen t und b können durch ein einfaches Polynom

$$b = f(t) = \sum_{k=1}^{m} x_k t^{k-1}$$

approximiert werden (z.B. $m = 2$: Gerade, $m = 3$: Parabel), um die "optimalen" Koeffizienten $x_1, x_2, \cdots, x_m$ zu suchen [5].

Mit der n Messungen (t_i, b_i) kann eine lineare Gleichung für die Unbekannten $x_1, x_2, \cdots, x_m$ beschreiben werden [5]:

$$x_1 + x_2 t_1^1 + \cdots + x_m t_1^{m-1} = b_1$$

$$\vdots$$

$$x_1 + x_2 t_n^1 + \cdots + x_m t_n^{m-1} = b_n.$$

Das lineare Gleichungssystem wird in Matrixschreibweise formuliert:

$$\underbrace{\begin{bmatrix} 1 & t_1 & \cdots & t_1^{m-1} \\ \vdots & & & \vdots \\ 1 & t_n & \cdots & t_n^{m-1} \end{bmatrix}}_{A \in \mathbb{R}^{n \times m}} \underbrace{\begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix}}_{x \in \mathbb{R}^m} = \underbrace{\begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}}_{b \in \mathbb{R}^n}. \tag{2.1.1}$$

Die Matrix A besitzt normalerweise sehr viel mehr Zeilen als Spalten ($n \gg m$). Die Gleichungen sind also sehr viel mehr als Unbekannte [5]. Deshalb wird im Allgemeinen kein x aufgefunden, um die Gleichung (2.1.1) genau zu erfüllen. Durch die Methode der kleinsten Quadrate wird ein x bestimmt, um die Gleichung möglichst gut zu erfüllen. Falls x die Lösung des Systems (2.1.1) ist, weist der Residuenvektor $r = Ax - b$ eine minimale Länge auf.

2.2 Lösen der MKQ durch Normalgleichung und Cholesky-Zerlegung

2.2.1 Normalgleichung

Das Quadrat des Residuenvektores kann durch die Funktion

$$f(x_1, x_2, \cdots, x_m) = \|r\|_2^2 \tag{2.2.1}$$

beschrieben werden. Das Minimum der Funktion wird gesucht durch [5]:

$$\begin{aligned} f(x_1, x_2, \cdots, x_m) &= \|r\|_2^2 \\ &= \langle Ax - b, Ax - b\rangle \\ &= \left\langle A^T Ax, x\right\rangle - 2\left\langle A^T b, x\right\rangle + \langle b, b\rangle \\ &= \langle Mx, x\rangle - 2\langle d, x\rangle + \|b\|_2^2, \end{aligned} \tag{2.2.2}$$

mit $M = A^T A \in \mathbb{R}^{m\times m}$ und $d = A^T b \in \mathbb{R}^n$. Die notwendige Bedingung zur Minimierung der Funktion $f(x_1, x_2, \cdots, x_m)$ ist [5]

$$0 = \nabla_x f = 2Mx - 2d, \tag{2.2.3}$$

d.h.

$$Mx = d \,\text{oder}\, A^T Ax = A^T b. \tag{2.2.4}$$

Die Gleichung 2.2.4 wird als Gauss'sche Normalgleichung bezeichnet. $x \in \mathbb{R}^m$ ist genau dann die Lösung der Minimierung der Funktion $f(x_1, x_2, \cdots, x_m)$ (das kleinste Quadrat des Residuenvektores), falls es die Normalgleichung 2.2.4 erfüllt. Somit besteht die Methode der kleinsten Quadrate darin, die Normalgleichung zu lösen.

2.2.2 Cholesky-Zerlegung

Um die Normalgleichung zu lösen, stehen die Verfahren der Cholesky-Zerlegung [19] im Prinzip zur Verfügung, da $M = A^T A \in \mathbb{R}^{m\times m}$ symmetrisch und positiv definiert ist, falls $\text{rang}(A) = m$ [29].

Die Matrix M kann eindeutig in der Form

$$M = LDL^T = GG^T \tag{2.2.5}$$

zerlegt werden, wobei $G = LD^{\frac{1}{2}}$, D eine positive Diagonalmatrix,

$$D^{\frac{1}{2}} = \text{diag}\left(\sqrt{d_{11}}, \sqrt{d_{22}}, \cdots, \sqrt{d_{mm}}\right)$$

und L eine untere Dreiecksmatrix, deren Diagonalelemente alle gleich 1 sind [35].

Zur Berechnung der Zerlegungsmatrix

$$G = \begin{bmatrix} g_{11} & & 0 \\ \vdots & \ddots & \\ g_{m1} & \cdots & g_{mm} \end{bmatrix} \tag{2.2.6}$$

wird direkt mit der Beziehung

$$M = GG^T \tag{2.2.7}$$

begonnen [22]. Durch das Multiplizieren von

$$\begin{bmatrix} g_{11} & & 0 \\ \vdots & \ddots & \\ g_{m1} & \cdots & g_{mm} \end{bmatrix} \begin{bmatrix} g_{11} & \cdots & g_{m1} \\ & \ddots & \vdots \\ 0 & & g_{mm} \end{bmatrix} = \begin{bmatrix} m_{11} & & m_{1m} \\ \vdots & \ddots & \\ m_{m1} & \cdots & m_{mm} \end{bmatrix} \tag{2.2.8}$$

wird das System (2.2.7) als $\frac{m(m+1)}{2}$ Gleichungen für die Größen g_{jk}, $k \leq j$ aufgefasst. Die erste Spalte von G ergibt sich:

$$m_{11} = g_{11}^2,\ m_{21} = g_{21}g_{11}, \cdots,\ m_{m1} = g_{m1}g_{11},$$

wobei sich

$$g_{11} = \sqrt{m_{11}},\ g_{j1} = \frac{m_{j1}}{g_{11}}\ j \in \{2, \cdots, m\}, \tag{2.2.9}$$

berechnet [22]. Aus

$$\begin{aligned} m_{ii} =& g_{i1}^2 + g_{i2}^2 + \cdots + g_{ii}^2,\ g_{ii} > 0 \\ m_{ji} =& g_{j1}g_{i1} + g_{j2}g_{i2} + \cdots + g_{ji}g_{ii},\ j > i \end{aligned}$$

werden die Elemente g_{ii} und g_{ji} berechnet durch [22, 35]:

$$g_{ji} = \begin{cases} 0 & \text{für } i > j \\ \sqrt{m_{ii} - \sum_{k=1}^{i-1} g_{ik}} & \text{für } i = j \\ g_{ii}^{-1}\left(m_{ji} - \sum_{k=1}^{i-1} g_{jk}g_{ik}\right) & \text{für } i < j. \end{cases} \tag{2.2.10}$$

Um g_{ji} zu berechnen, wird ein $O(m)$ Rechenaufwand benötigt. Der gesamte Rechenaufwand der Cholesky-Zerlegung ist somit $O(m^3)$. Der Pseudocode ist in der Literatur unter [35] zu finden.

2.2.3 Lösen der Normalgleichung mithilfe der Cholesky-Zerlegung

Die Normalgleichung $Mx = d$ lässt sich effizient durch die Cholesky-Zerlegung, Vorwärts- und Rückwärtseinsetzmethode lösen [35]. Die Berechnung teilt sich in folgende Schritte auf:

- Cholesky-Zerlegung: $M = GG^T$;
- Vorwärtseinsetzen Lösung des linearen Gleichungssystems: $Gy = d$;
- Rückwärtseinsetzen Lösung des linearen Gleichungssystems $G^T x = y$.

Vorwärtseinsetzen beim Eliminationsverfahren

Wird $G^T x = y$ in Gleichung $GG^T x = d$ eingesetzt, wird die folgende Gleichung berechnet:

$$\begin{bmatrix} g_{11} & & 0 \\ \vdots & \ddots & \\ g_{m1} & \cdots & g_{mm} \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} d_1 \\ \vdots \\ d_m \end{bmatrix}. \tag{2.2.11}$$

Aus Gaußsches Eliminationsverfahren ergeben sich die Variablen $[y_1, \cdots, y_m]$ rekursiv von oben nach unten:

$$\begin{aligned} y_1 &= \frac{d_1}{g_{11}} \\ y_2 &= \frac{d_2 - g_{21} y_1}{g_{22}} \\ &\vdots \\ y_m &= \frac{d_m - \sum_{i=1}^{m-1} g_{mi} y_i}{g_{mm}}. \end{aligned} \tag{2.2.12}$$

Das Verfahren 2.2.12 wird als Vorwärtseinsetzen bezeichnet. Die entsprechenden Pseudocode sind in der Literatur [19] aufgelistet. Die Laufzeit der Vorwärtseinsetzen ist $O(m^2)$.

Rückwärtseinsetzen beim Eliminationsverfahren

Die Gleichung

$$\begin{bmatrix} g_{11} & \cdots & g_{m1} \\ & \ddots & \vdots \\ 0 & & g_{mm} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix} \tag{2.2.13}$$

wird durch das Rückwärtseinsetzen der Eliminationsverfahren berechnet, wobei

$$\begin{aligned} x_m &= \frac{y_m}{g_{mm}} \\ x_{m-1} &= \frac{y_{m-1} - g_{m-1,m} y_m}{g_{m-1,m-1}} \\ &\vdots \\ x_1 &= \frac{y_1 - \sum_{i=2}^{m} g_{1i} y_i}{g_{11}}. \end{aligned} \tag{2.2.14}$$

Die Pseudocode des Rückwärtseinsetzen sind auch in der Literatur [19] zu finden. Der Rechenaufwand vom Rückwärtseinsetzen ist $O(m^2)$.

2.2.4 Fazit

$x \in \mathbb{R}^m$ ist genau dann die Lösung der Methode der kleinsten Quadrate, wenn die Normalgleichungen 2.2.4 erfüllt wird. Für das Problem werden mithilfe der Cholesky-Zerlegung die Operationen benötigt:

1. Berechnung von $M = A^T A$, $A \in \mathbb{R}^{n \times m}$: $O(nm^2)$;
2. Cholesky-Zerlegung: $O(m^3)$;
3. Vorwärts- und Rückwärtseinsetzen: $O(m^2)$.

Für $m \approx n$ werden $O(m^3)$ Operationen benötigt. Die Berechnung der Normalgleichung ist unter Numerikern schlecht, da die Kondition des linearen Gleichungssystems sich durch die Berechnung $A^T A$ verschlechtert, wobei die Kondition von A typischerweise schon schlecht ist [20, 29]. Somit werden hier die relativen Fehler der Lösung groß.

2.3 Lösen der MKQ mithilfe der QR-Zerlegung

Wird die Methode der kleinsten Quadrate direkt behandelt, ohne die Normalgleichung zu lösen, wird die Lösung verbessert, da die Kondition der Normalgleichungen sich durch die Berechnung A^TA verschlechtert. Die Grundidee ist, dass orthogonale Transformationen die euklidische Norm eines Vektors nicht verändern [42]. D. h. $\|r\|_2^2 =< Ax-b, Ax-b >=< Q(Ax-b), Q(Ax-b) >$, wobei Q eine beliebige orthogonale Matrix ist. Deshalb kann eine QR-Zerlegung [8, 9] verwendet werden, um die Lösung der Methode der kleinsten Quadrate zu berechnen. Dabei wird die Matrix A als Produkt von zwei Matrizen QR zerlegt, wobei $Q \in \mathbb{R}^{n\times n}$ orthogonal und $R \in \mathbb{R}^{n\times m}$ eine obere Rechtsdreiecksmatrix ist [29].

$$R = \begin{bmatrix} * & \cdots & * \\ & \ddots & \vdots \\ 0 & & * \\ 0 & \cdots & 0 \\ \vdots & & \vdots \\ 0 & \cdots & 0 \end{bmatrix} = \begin{bmatrix} R_1 \\ 0 \end{bmatrix},\ R_1 \in \mathbb{R}^{m\times m}$$

Der Residuenvektor $r = Ax - b$ darf mit Q^T multipliziert werden, ohne die Länge zu verändern, da Q^T orthogonal ist. D. h. [7]:

$$\begin{aligned} \|r\|_2^2 &= \left\|Q^T(Ax-b)\right\|_2^2 \\ &= \left\|Rx - Q^Tb\right\|_2^2 \\ &= \left\| \begin{bmatrix} R_1 \\ 0 \end{bmatrix} x - \begin{bmatrix} d_1 \\ d_2 \end{bmatrix} \right\|_2^2 \\ &= \|R_1x - d_1\|_2^2 + \|d_2\|_2^2, \end{aligned} \tag{2.3.1}$$

mit $d = Q^Tb = \begin{bmatrix} d_1 \\ d_2 \end{bmatrix}$. Also wird $\|r\|_2^2$ genau dann minimiert, wenn

$$R_1x - d_1 = 0 \Leftrightarrow R_1x = d_1.$$

Durch ein anschließendes Rückwärtseinsetzen (siehe Abschnitt 2.2.3) wird das lineare Gleichungssystem $Ax = b$ durch die Methode der kleinsten Quadrate gelöst.

2.3.1 Berechnung der QR-Zerlegung

In diesem Abschnitt wird die QR-Zerlegung der Matrix $A \in \mathbb{R}^{n\times m}$ beschrieben, wobei Q eine orthogonale Matrix und R eine rechte obere Dreiecksmatrix ist. Drei bekannte

Verfahren werden in diesem Fall verwenden. Zunächst wird in dieser Diplomarbeit die Zerlegung mit dem Gram-Schmidtschen Orthogonalisierungsverfahren und danach die Zerlegung mit Householder Transformationen und Givens-Rotationen erklärt.

Gram-Schmidtsches Orthogonalisierungsverfahren

Das linear unabhängige Elemente kann immer mit dem Gram-Schmidtschen Orthogonalisierungsverfahren in paarweise zu einander orthogonale Elemente, die in einem Vektorraum sind, umgerechnet werden. Für linear unabhängige Vektoren $a_1, a_2, \cdots, a_m$ können orthonormale Vektoren $q_1, q_2, \cdots, q_m$ nach folgendem Muster berechnet werden [19]:

$$\begin{aligned} &1.\ q_j = a_j \\ &2.\ q_j = q_j - \sum_{i=1}^{j-1} \frac{\langle q_j, q_i \rangle}{\langle q_i, q_i \rangle} q_i \\ &3.\ q_j = \frac{q_j}{\|q_j\|_2},\ j = 1, 2, \cdots, m. \end{aligned} \tag{2.3.2}$$

Wenn $a_i \in \mathbb{R}^n$ (notwendig ist $n \geq m$) als Spalten einer Matrix $A \in \mathbb{R}^{n \times m}$ und entsprechend $q_i \in \mathbb{R}^n$ als Spalten einer Matrix $Q \in \mathbb{R}^{n \times m}$ aufgefasst werden, ist das Ergebnis des Gram-Schmidtschen Verfahrens dann gleich wie bei der QR-Zerlegung $A = QR$, wobei $R \in \mathbb{R}^{m \times m}$ eine obere Rechtsdreiecksmatrix (mit Gleichung 2.3.3) ist [19].

$$r_{jj} = \|q_j\|,\ r_{jk} = \langle q_k, q_j \rangle,\ j < k < m \tag{2.3.3}$$

Das Gram-Schmidtsches Orthogonalisierungsverfahren ist für die numerische Berechnung durch einen Computer nicht geeignet, da durch die Rundungsfehler die berechneten Vektoren nicht mehr zwingend orthogonal sind [18]. Das Verfahren hat eine Laufzeit von $O(m^2 n)$.

Householder Transformation

Für einen Vektor $\nu \in \mathbb{R}^n$ wird

$$G_\nu = \nu \nu^T = \begin{bmatrix} \nu_1^2 & \cdots & \nu_1 \nu_n \\ \vdots & & \vdots \\ \nu_n \nu_1 & \cdots & \nu_n^2 \end{bmatrix} \in \mathbb{R}^{n \times n} \tag{2.3.4}$$

"dyadisches Produkt" genannt [22]. Wenn $\|\nu\|_2 = 1$, heißt die Matrix

$$U_\nu = I - 2G_\nu \tag{2.3.5}$$

"Householder Transformation" [22]. Sie beschreibt die Reflexionen an der Ebene, die auf ν senkrecht liegt [29]. Für einen gegebenen Vektor $x \in \mathbb{R}^n$ wird der Vektor $\nu \in \mathbb{R}^n$ mit $\|\nu\|_2 = 1$ so bestimmt, dass [19]

$$ce_1 = U_\nu x = \left(I - 2\nu\nu^T\right) x = x - 2\langle \nu, x\rangle \nu. \tag{2.3.6}$$

Daraus folgt

$$2\langle \nu, x\rangle \nu = x - ce_1. \tag{2.3.7}$$

Die Gleichung 2.3.7 bedeutet, dass der Vektor ν als ein Vielfaches vom Vektor $x - ce_1$ sein muss [19]. Es folgt die Lösung

$$\|x\|_2 = \|U_\nu x\|_2 = \|ce_1\|_2 = |c|\,, \tag{2.3.8}$$

weil U_ν Orthogonal ist [19]. Der Betrag von c lässt sich also durch 2.3.8 bestimmen. So folgt das Ergebnis [19]

$$c = \gamma \|x\|_2 \text{ mit } |\gamma| = 1. \tag{2.3.9}$$

Wird 2.3.9 in 2.3.7 eingesetzt, wird [19]

$$\underbrace{2\langle \nu, x\rangle \nu}_{\tilde{\nu}} = x - \gamma \|x\|_2 e_1 \text{ und } \nu = \frac{\tilde{\nu}}{\|\tilde{\nu}\|_2} \tag{2.3.10}$$

erzeugt. Um die Wahl von γ zu untersuchen, setzt man $\tilde{\nu}$ in Gleichung 2.3.7 ein. So erhält man die Gleichung $\overline{\gamma} x_1 = \gamma \overline{x_1}$, wobei der Querstrich den Übergang zur konjugiert komplexen Zahl andeutet [19]. Um Auslöschung bei der Bildung von $\tilde{\nu}$ zu vermeiden, wird die Lösung [19]

$$\gamma = \begin{cases} -1 & \text{falls } x_1 = 0 \\ -\frac{x_1}{|x_1|} & \text{sonst.} \end{cases} \tag{2.3.11}$$

gewählt. Wenn die Householder Transformation derart konstruiert wird, dass die Spalten $\left(A^{(1)}, A^{(2)}, \cdots, A^{(m)}\right)$ der Ausgangsmatrix $A \in \mathbb{R}^{n\times m}$ auf Vektoren, ihre Element $A^{(i)}_{(j)} = 0$ für $j > i$, abgebildet werden, lässt sich die QR-Zerlegung mithilfe von Householder Transformation berechnen [29], wobei $A^{(i)}_{(j)}$ die Element mit Index $(i; j)$ von A bezeichnet.

Die Matrix R der QR-Zerlegung von Matrix $A \in \mathbb{R}^{n\times m}$ lässt sich mithilfe von Householder Transformation in m Schritten

$$A = A_1 \to \cdots \to A_i \to \cdots \to A_m = R, \tag{2.3.12}$$

erzeugen, wobei A_{i-1} die Gestalt aufweist [22]:

$$A_{i-1} = \begin{array}{c} \left[\begin{array}{ccccc} * & & & & * \\ 0 & \ddots & & & \vdots \\ 0 & 0 & * & \cdots & * \\ 0 & 0 & \vdots & & \vdots \\ 0 & 0 & * & \cdots & * \end{array}\right] \\ \begin{array}{ccc} & & i \end{array} \end{array} \begin{array}{l} \\ \\ \\ \\ i \end{array} \tag{2.3.13}$$

Die Householder Transformation $U_i \in \mathbb{R}^{n \times n}$ im i-ten Schritten wird wie folgt bestimmt: [22]

$$U_i A_{i-1} = A_i \tag{2.3.14}$$

$$U_i = \begin{array}{c} \left[\begin{array}{ccc|ccc} & & & & & \\ & I & & & 0 & \\ & & & & & \\ \hline & & & & & \\ & 0 & & & I - 2\nu_i\nu_i^T & \\ & & & & & \end{array}\right] \\ \begin{array}{ccc} & & i \end{array} \end{array} \begin{array}{l} \\ \\ \\ \\ i \end{array} \tag{2.3.15}$$

Nach m Schritten wird die Matrix R erzeugt

$$R = \underbrace{U_m U_{m-1} \cdots U_1}_{Q^T} A = Q^T A,$$

wobei $Q \in \mathbb{R}^{n \times n}$ orthogonal und $R \in \mathbb{R}^{n \times m}$ eine obere Rechtsdreiecksmatrix ist [22].

Folglich ist ein Pseudocode formuliert, der die QR-Zerlegung mit Householder Transformation berechnet.

```
Zweck: Berechnung der QR–Zerlegung mit Householder
       Transformation
Parameter: n*m–Matrix A

for j:=1 to m do
begin
  x:= A[von j bis n][j];
  Bestimme Householder–Vektor v mit x;
  Berechne Householder–Matrix U mit v;
  Berechne A:= U*A;
end

Ergebnis: A ist eine rechte obere Dreiecksmatrix (R)
```

Die Laufzeit der QR-Zerlegung der Matrix $A \in \mathbb{R}^{n \times m}$ mit Householder Transformation ist $O\left(n^3 m + n^2 m^2\right)$.

Givens-Rotationen

Die Givens-Rotation ist benannt nach Wallace Givens. Sie ist in der linearen Algebra eine Drehung in einer Ebene, die durch zwei Koordinaten-Achsen aufgespannt wird [17]. Eine Givens-Rotation ist mit

$$U(i,j,\varphi) = \begin{bmatrix} 1 & & & & & & & & \\ & \ddots & & & & & & & \\ & & 1 & & & & & & \\ & & & \cos\varphi & & & \sin\varphi & & \leftarrow i \\ & & & & 1 & & & & \\ & & & & & \ddots & & & \\ & & & & & & 1 & & \\ & & & -\sin\varphi & & & \cos\varphi & & \leftarrow j \\ & & & & & & & 1 & \\ & & & & & & & & \ddots \\ & & & & & & & & & 1 \end{bmatrix} \tag{2.3.16}$$

bezeichnet[29]. Die Matrix $U(i,j,\varphi_{i,,j})$ beschreibt eine Drehung um den Winkel $\varphi_{i,j}$ in der (i,j)-Ebene. Eine geeignete $U(i,j,\varphi_{i,j})$ wird von links an eine Matrix

$A \in \mathbb{R}^{n\times m}$ multipliziert, so dass die Elemente unter der Diagonalen nacheinander zu 0 werden. Die Paare (i,j) von $U(i,j,\varphi)$ sind also z.B.
$(2,1),(3,1),\cdots,(n,1),\cdots,(n,m)$:

$$\begin{aligned} R &= U(n,n-1,\varphi_{n,n-1})\cdots U(n,1,\varphi_{n,1})\cdots U(3,1,\varphi_{3,1})\,U(2,1,\varphi_{2,1})\,A \\ \Rightarrow A &= \underbrace{\left(U(n,n-1,\varphi_{n,n-1})\cdots U(n,1,\varphi_{n,1})\cdots U(3,1,\varphi_{3,1})\,U(2,1,\varphi_{2,1})\right)^T}_{Q} R = QR. \end{aligned} \tag{2.3.17}$$

Eine Givens-Rotation $U(i,j,\varphi_{i,j})$ wird so berechnet, dass $(U(i,j,\varphi_{i,j})A)_{i,j} = 0$:

$$\begin{aligned} \sin\varphi &= \frac{a_{ij}}{\sqrt{a_{jj}^2 + a_{ij}^2}} \\ \cos\varphi &= \frac{a_{jj}}{\sqrt{a_{jj}^2 + a_{ij}^2}}. \end{aligned} \tag{2.3.18}$$

Es folgt der Pseudocode für die QR-Zerlegung mit Givens-Rotationen:

```
Zweck: Givens-Rotationen zur Berechnung der QR-Zerlegung
Parameter: n*m-Matrix A

for j:=1 to m do
begin
  for i:=j+1 to n
  begin
    Bestimme U(i,j), so dass (U(i,j)*A)[i,j]=0;
    Berechne A:=U(i,j)*A;
  end
end

Ergebnis: A ist eine rechte obere Dreiecksmatrix
```

Normalerweise sind Matrix-Matrix-Produkte sehr aufwendig zu berechnen. Bei der Betrachtung der Matrixprodukten von $U(i,j,\varphi_{i,j})Q$ und $U(i,j,\varphi_{i,j})A$ werden folgende Optimierungsansätze deutlich. Da der Unterschied zwischen $U(i,j,\varphi_{i,j})$

und Einheitsmatrix nur vier Elementen sind, kann der Rechenaufwand der Matrixprodukten $U(i,j,\varphi_{i,j})A$ durch eine spezielle Matrixmultiplikation von $O(n^2m)$ auf $O(m)$ reduziert werden. Damit wird der Rechenaufwand für die Berechnung der Matrix R der QR-Zerlegung der Matrix $A \in \mathbb{R}^{n\times m}$ mit der Givens-Rotationen auf $O(nm^2)$ verringert.

Diskussion

Das Gram-Schmidtsche Orthogonalisierungsverfahren zur praktischen Berechnung von Q und R einer QR-Zerlegung ist ungeeignet, da aufgrund von Rundungsfehlern die Orthogonalität der Spalten von Q schnell verloren geht. Die QR-Zerlegungen mit Hilfe von Householder Transformation und Givens-Rotationen sind sehr stabil im Vergleich zum Gram-Schmidtsche Verfahren [32]. Der Zeilenaustausch (d. h. Pivotisierung) ist ebenfalls nicht erforderlich [32]. Für das Algorithmus durch Householder Transformation ist $O(m)$ Matrixmultiplikationen $U * A$ durchzuführen, wobei $U \in \mathbb{R}^{n\times n}$ und $A \in \mathbb{R}^{n\times m}$. Der Rechenaufwand einer Matrixmultiplikation $U * A$ ist $O(n^2m)$. Das Algorithmus zur Berechnung QR-Zerlegung hat eine Laufzeit von insgesamt $O(n^3m)$. Trotzdem sind für das Algorithmus durch Givens-Rotationen $O(mn)$ Matrixmultiplikationen durchzuführen [32], wodurch die Laufzeit insgesamt aber wenig ist, da die spezielle Matrixmultiplikation nur $2m$ Werte pro Matrixmultiplikationen berechnet. Insgesamt beträgt dann der Aufwand für eine QR-Zerlegung einer vollbesetzten $n \times m$-Matrix A mit Hilfe von Givens-Rotationen $O(nm^2)$. Der Aufwand ist allerdings erheblich niedriger, wenn die Matrix A wenige Spalten besitzt [32].

2.3.2 Fazit

Im Abschnitt 2.3.1 werden drei Methoden zur QR-Zerlegung vorgestellt, wobei die optimalste Methode die Givens-Rotation ist. Somit werden die kleinsten Quadrate $(\mathrm{Min}\,\|Ax-b\|_2^2)$ mithilfe der QR-Zerlegung durch Givens-Rotationen bestimmt. Für die Methode sind die folgenden drei Schritte durchzuführen:

1. Bestimme Matrix R der QR-Zerlegung für Matrix $A \in \mathbb{R}^{n\times m}$ mithilfe der Givens-Rotationen durch:

   $$R = U\left(n, n-1, \varphi_{n,n-1}\right) \cdots U\left(n, 1, \varphi_{n,1}\right) \cdots U\left(3, 1, \varphi_{3,1}\right) U\left(2, 1, \varphi_{2,1}\right) A.$$

   die Laufzeit ist $O(nm^2)$

2. Berechne $d = Q^T b = \begin{bmatrix} d_1 \\ d_2 \end{bmatrix}$ durch:

   $$d = Q^T b = U(n, n-1, \varphi_{n,n-1}) \cdots U(n, 1, \varphi_{n,1}) \cdots U(3, 1, \varphi_{3,1})\, U(2, 1, \varphi_{2,1})\, b.$$

   die Laufzeit ist $O(nm)$

3. Löse $R_1 x = d_1$ durch Rückwärtseinsetzen, wobei $R = \begin{bmatrix} R_1 \\ 0 \end{bmatrix}$ und $R_1 \in \mathbb{R}^{m \times m}$. Die Laufzeit ist$O(m^2)$.

2.4 Lösen der MKQ mithilfe der Singulärwertzerlegung (SVD)

Jede $n \times m$-Matrix A besitzt eine Singulärwertzerlegung (SVD)

$$A = U \Sigma V^T, \tag{2.4.1}$$

wobei U eine $n \times n$ orthogonale Matrix, V eine $m \times m$ orthogonale Matrix und Σ eine $n \times m$ diagonale Matrix ist,

$$\Sigma = \begin{bmatrix} \Sigma_A \\ 0 \end{bmatrix},\ n < m \text{ bzw. } \Sigma = \begin{bmatrix} \Sigma_A & 0 \end{bmatrix},\ n > m$$

und

$$\Sigma_A = \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_k \end{bmatrix},\ k = \min(n, m),$$

wobei die Diagonalelemente $\sigma_1, \cdots, \sigma_k$ nicht negativ sind und sie in der Reihenfolge $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_k > 0$ geordnet werden [7]. Sie werden singuläre Werte von A genannt. Wird hier nur $n \gg m$ betrachtet, erfolgt

$$A = U \Sigma V^T$$
$$= \begin{bmatrix} u_{1,1} & \cdots & u_{1,m} & * & \cdots & * \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ u_{m,1} & \cdots & u_{m,m} & * & \cdots & * \\ u_{m+1,1} & \cdots & u_{m-1,m} & * & \cdots & * \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ u_{n,1} & \cdots & u_{n,m} & * & \cdots & * \end{bmatrix} \begin{bmatrix} \sigma_1 & & 0 \\ & \ddots & \\ 0 & & \sigma_m \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} v_{1,1} & \cdots & v_{m,1} \\ \vdots & \ddots & \vdots \\ v_{1,m} & \cdots & v_{m,m} \end{bmatrix}.$$

Es wird deutlich, dass beim Ausmultiplizieren die *-Werte mit den Nullen der Σ-Matrix verrechnet werden. Dadurch werden die Matrizen der Singulärwertzerlegung reduziert, indem die *-Werte in der Matrix A und die Nullen in der Matrix Σ entfernt werden. Als Ergebnis folgt die Singulärwertzerlegung

$$\begin{aligned} A &= U_A \Sigma_A V_A^T \\ &= \begin{bmatrix} u_{1,1} & \cdots & u_{1,m} \\ \vdots & \ddots & \vdots \\ u_{n,1} & \cdots & u_{n,m} \end{bmatrix} \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_m \end{bmatrix} \begin{bmatrix} v_{1,1} & \cdots & v_{m,1} \\ \vdots & \ddots & \vdots \\ v_{1,m} & \cdots & v_{m,m} \end{bmatrix}. \end{aligned}$$

Das entfernte Teil der Matrix U wird als $U_A^{\perp}$ bezeichnet. Schließlich wird für jede Matrix $Q \in \mathbb{R}^{n \times l}$, deren Spaltenvektoren paarweise orthonormal zueinander sind, $Q^{\perp} \in \mathbb{R}^{n \times n-l}$ als eine Matrix, deren Spaltenraum senkrecht zum Spaltenraum der Matrix Q ist, bezeichnet.

2.4.1 Bestimmung der Singulärwertzerlegung

$A = U_A \Sigma_A V_A^T$ ist eine Singulärwertzerlegung von $A \in \mathbb{R}^{n \times m}$. Wegen

$$A^T A = (V_A \Sigma_A^T U_A^T)(U_A \Sigma_A V_A^T) = V_A \Sigma_A^T \Sigma_A V_A^T \tag{2.4.2}$$

ist der Übergang von $\Sigma_A^T \Sigma_A$ zu $A^T A$ eine Ähnlichkeitstransformation, da $V_A \in \mathbb{R}^{m \times m}$ orthogonale ist (d.h. $V_A^T = V_A^{-1}$) [40]. Die Singulärwerte der Matrix A werden bestimmt, indem die Quadratwurzeln aus den Eigenwerten von $A^T A$ berechnet werden. Danach können die Singulärwertzerlegung $A = U_A \Sigma_A V_A^T$ durch folgende Schritte bestimmt werden [10]:

1. Berechne$B = A^T A$ und $B \in \mathbb{R}^{m \times m}$.
2. Berechne die Eigenwerte von B, die kein negatives Vorzeichen aufweisen und in der Reihenfolge $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_m > 0$ geordnet werden. So wird die Matrix Σ_A mit $\sigma_i = \sqrt{\lambda_i}$ bestimmt.
3. Bilde die Spalten der Orthonormalbasis $V_A^{(1)}, ..., V_A^{(m)}$: Wenn $V_A^{(i)}$ der Eigenvektor für Eigenwert λ_i ist, dann ist $V_A = \left(V_A^{(1)}, V_A^{(2)} \cdots V_A^{(m)}\right)$ eine orthogonale Matrix ($V_A^T = V_A^{-1}$).
4. Berechne die orthogonale Matrix $U_A = \left(U_A^{(1)}, U_A^{(2)}, \cdots, U_A^{(m)}\right)$ mit den Spaltenvektoren $U_A^{(i)} = \frac{1}{\sigma_i} A V_A^{(i)}$.

2.4.2 Bestimmung der Eigenwerte mit dem QR-Algorithmus

Der QR-Algorithmus dient zur Bestimmung aller Eigenwerte einer Matrix $B \in \mathbb{R}^{m\times m}$ und wurde von Francis [8, 9] und Kublanovskaya [14] unabhängig voneinander im Jahr 1961 eingeführt [42]. Er ist gegeben durch

Algorithmus 2.1 Der QR-Algorithmus dient zur Bestimmung aller Eigenwerte einer Matrix $B \in \mathbb{R}^{m\times m}$ [42].

```
while (B ist nicht konvergent)
{
    Zerlege B=QR;
    B=RQ;
}
```

Der Übergang der Schritte i zu Schritte $i+1$ ist eine Ähnlichkeitstransformation [42]:

$$\begin{aligned} B_{i+1} &= R_i Q_i \\ &= \underbrace{Q_i^T Q_i}_{=I} R_i Q_i \\ &= Q_i^T \underbrace{Q_i R_i}_{B_i} Q_i = Q_i^T B_i Q_i. \end{aligned} \tag{2.4.3}$$

Somit besitzen die Matrizen $(B, B_1, \cdots, B_i, \cdots)$ dieselben Eigenwerte. Die Konvergenz des QR-Algorithmus wird beschrieben durch

$$\begin{aligned} \lim_{i\to\infty} b_{jk}^{(i)} &= 0 \ \text{ für } j > k \\ \lim_{i\to\infty} b_{kk}^{(i)} &= \lambda_k, \end{aligned} \tag{2.4.4}$$

wobei λ_k, $k = 1, \cdots, m$ die Eigenwerte von $B \in \mathbb{R}^{m\times m}$ sind[42].

2.4.3 Bestimmung der MKQ

Das Problem der Methode der kleinsten Quadrate kann auch mit der Singulärwertzerlegung $A = U\Sigma V^T$ direkt gelöst werden [7]. Der Residuenvektor $r = Ax - b$ darf

mit U^T multipliziert werden, ohne dabei die Länge zu verändern, d. h. [7]:

$$\begin{aligned}
\|r\|_2^2 &= \left\| \underbrace{U^T U}_{=I} \Sigma \underbrace{V^T x}_{y} - U^T b \right\|_2^2 \\
&= \left\| \Sigma y - U^T b \right\|_2^2 \\
&= \left\| \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_m \\ & 0 & \end{bmatrix} y - \begin{bmatrix} d_I \\ d_{II} \end{bmatrix} \right\|_2^2 \\
&= \left\| \underbrace{\begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_m \end{bmatrix}}_{\Sigma_A} y - d_I \right\|_2^2 + \left\| \begin{matrix} 0 \\ d_{II} \end{matrix} \right\|_2^2,
\end{aligned} \tag{2.4.5}$$

mit $d = U^T b = \begin{bmatrix} d_I \\ d_{II} \end{bmatrix}$, $d_I = (U_A)^T b$ und $d_{II} = \left(U_A^{\perp}\right)^T b$. Schließlich wird $\|r\|_2^2$ genau dann minimiert, wenn $\Sigma_A y - d_I = 0$, wobei y wird bestimmt durch:

$$y_i = \frac{d_{I,i}}{\sigma_i},\ i = 1, \cdots, m. \tag{2.4.6}$$

Es gilt:

$$x = Vy. \tag{2.4.7}$$

Dies liefert die Lösung der Methode der kleinsten Quadrate. Folglich wird der optimale Wert der kleinsten Quadrate berechnet:

$$\begin{aligned}
Z &= \min_{x \in \mathbb{R}^n} \|Ax - b\|_2 \\
&= \left\| \begin{bmatrix} 0 \\ d_{II} \end{bmatrix} \right\|_2 = \left\| U \begin{bmatrix} 0 \\ d_{II} \end{bmatrix} \right\|_2 \\
&= \left\| U \begin{bmatrix} 0 \\ d_{II} \end{bmatrix} \right\|_2 = U_A \cdot 0 + U_A^{\perp} d_{II} \\
&= \underbrace{U_A^{\perp} \left(U_A^{\perp}\right)^T b}_{=b^{\perp}}.
\end{aligned}$$

2.4.4 Diskussion

Um die Singulärwerte zu bestimmen, müssen die Eigenwerte der Matrix A^TA berechnen werden. Die Berechnung der Eigenwerte der Matrix A^TA hat unter Numerikern einen schlechten Ruf, da:

1. die Kondition des linearen Gleichungssystems sich durch die Berechnung A^TA verschlechtert,

2. die numerische Lösung des symmetrischen Eigenwertproblems zu A^TA sehr aufwändig und instabil ist, weil viele Matrix-Matrix-Produkte benötigt werden (siehe Algorithmus 2.1) [40].

2.5 Lösen der MKQ durch Pseudoinverse

Wenn $y = Bx$ bijektiv ist, ist die Matrix $B \in \mathbb{R}^{n\times n}$ regulär [31]. Die Inverse $B^{-1} \in \mathbb{R}^{n\times n}$ ist durch Umkehrabbildung: $x = B^{-1}y$ definiert [31]. Wenn die Matrix $A \in \mathbb{R}^{n\times m}$ für $Ax = b$ nicht regulär (z. B. rechteckig) ist, lässt sich die Inversion der Matrix A zur Beschreibung der inversen Funktion $b = A^{-1}x$ nicht durchführen, da die inverse Matrix A^{-1} nicht existiert [23]. Durch einige Umformungen mit Hilfe der linearen Algebra kann man auch die Lösung von $Ax = b$ ebenfalls in der Form $x = A^+b$ darstellen, wobei die Matrix A^+ Pseudoinverse auch Moore-Penrose-Inverse der Matrix A genannt wird. Sie ist eine Verallgemeinerung der inversen Matrix auf singuläre und nicht quadratische Matrizen, weshalb sie häufig auch als verallgemeinerte Inverse bezeichnet wird [38]. Das Produktes A^TA der Rechteckmatrix A ist quadratisch, damit ist dieses invertierbar, wenn dieses einen vollen Rang aufweist [23]. Dann kann die Gleichung $Ax = b$ mit der Transponierten A^T multiplizieret werden[23]:

$$A^TAx = A^Tb \quad . \tag{2.5.1}$$

Um die gesuchte Lösung x zu isolieren, muss die Gleichung 2.5.1 mit der Inversen von A^TA multiplizieret werden [23]:

$$x = \underbrace{\left(A^TA\right)^{-1}A^T}_{A^+}b = A^+b. \tag{2.5.2}$$

Die Gleichung 2.5.1 ist genau die notwendige Bedingung zur Methode der kleinsten Quadrate (siehe Abschnitt 2.2). Die Lösung der Methode der kleinsten Quadrate lässt sich dann mit der Pseudoinverse als $x = A^+b$ berechnen. Die Pseudoinverse

einer Matrix $A \in \mathbb{R}^{n\times m}$ ist die eindeutig bestimmte Matrix $A^+ \in \mathbb{R}^{m\times n}$, welche die folgenden Eigenschaften erfüllt [16, 38]:

$$\begin{aligned} AA^+A &= A \\ A^+AA^+ &= A^+ \\ AA^+ &= (AA^+)^T \\ A^+A &= (A^+A)^T \end{aligned} \tag{2.5.3}$$

2.5.1 Berechnung der Pseudoinverse durch Rang Dekomposition

Wenn die Matrix $A \in \mathbb{R}^{n\times m}$ den Rang k hat, kann A in das Produkt $A = BC$ zerlegt werden, wobei die Matrix $B \in \mathbb{R}^{n\times k}$ und die Matrix $C \in \mathbb{R}^{k\times m}$ [38]ist. Dann wird die Pseudoinverse A^+ der Matrix A berechnet durch [38]:

$$A^+ = C^T \left(CC^T\right)^{-1} \left(B^TB\right) B^T. \tag{2.5.4}$$

Wenn A den vollen Spaltenrang (d.h. $k = m$) hat, kann für C die Einheitsmatrix gewählt werden und die obige Formel reduziert sich zu [38]

$$A^+ = \left(A^TA\right)^{-1} A^T. \tag{2.5.5}$$

In ähnlicher Weise hat A den vollen Zeilenrang (d.h. $k = n$). Es gilt die Gleichung [38]

$$A^+ = A^T \left(AA^T\right)^{-1}. \tag{2.5.6}$$

Das Verfahren ist numerisch instabil, da das Produkt A^TA oder AA^T benötigt wird. Die Kondition des linearen Gleichungssystems wird durch die Berechnung des Produkts quadriert [38].

2.5.2 Berechnung der Pseudoinverse durch QR-Methode

Die Berechnung des Produkts A^TA oder AA^T und ihrer Inversen ist oft ein Grund von numerischen Rundungsfehlern und Rechenaufwand in der Praxis [37]. Ein anderer Ansatz mit der QR-Zerlegung von A kann wie folgt alternativ verwendet werden[37]:

Wenn die Matrix $A \in \mathbb{R}^{n\times m}\,(n \geq m)$ vollen Spaltenrang aufweist, kann A in das Produkt $A = QR$ zerlegt werden, wobei $Q \in \mathbb{R}^{n\times n}$ orthogonal und $R \in \mathbb{R}^{n\times m}$ eine

obere Rechtsdreiecksmatrix ist. Durch einfache Rechenumformung erhält man [39]:

$$\begin{aligned} A^+ &= (QR)^+ \\ \Rightarrow A^+ &= R^{-1}Q^T \\ \Rightarrow RA^+ &= Q^T. \end{aligned} \tag{2.5.7}$$

Um die Pseudoinverse der Matrix $A \in \mathbb{R}^{n\times m}$ mithilfe der QR-Zerlegung zu bestimmen, sind folgende drei Schritte durchzuführen:

1. Bestimme die Matrix R und Q der QR-Zerlegung für die Matrix $A \in \mathbb{R}^{n\times m}$; Laufzeit: $O(nm^2 + n^2m)$;
2. Berechne Q^T; Laufzeit: $O(n^2)$;
3. Löse $RA^+ = Q^T$ durch Rückwärtseinsetzen: $O(nm^2)$.

Insgesamt ist die Laufzeit der Pseudoinverse durch QR-Zerlegung $O(nm^2 + n^2m)$.

2.5.3 Berechnung der Pseudoinverse durch Singulärwertzerlegung

Wenn $A = U\Sigma V^T$ eine Singulärwertzerlegung von $A \in \mathbb{R}^{n\times m}$ ist (siehe Abschnitt 2.4), ist die Pseudoinverse A^+ gegeben durch [38]:

$$A^+ = (U\Sigma V^T)^+ = V\Sigma^+ U^T, \tag{2.5.8}$$

wobei

$$(\Sigma)^+_{ij} = \begin{cases} \frac{1}{\sigma_i,} & \text{falls } i = j \text{ und } \sigma_i \neq 0 \\ 0 & \text{sonst.} \end{cases} \tag{2.5.9}$$

2.5.4 Berechnung der Pseudoinverse durch Verfahren von Greville

Der Mathematiker T. N. E. Greville hatte im Jahr 1960 einen Algorithmus, der Pseudoinverse spaltenweise berechnen kann, [11], der dabei mit Hilfe einer iterativen Prozedur die Pseudoinverse A^+von $A \in \mathbb{R}^{n\times m}$ nach endlich vielen Schritten berechnet wird [26], vorgestellt. Die Matrix $A \in \mathbb{R}^{n\times m}$ kann in den Spalten $\left(A^{(1)}, A^{(2)}, \cdots, A^{(m)}\right)$ dargestellt werden [26]. $A_k = \left(A^{(1)}, A^{(2)}, \cdots, A^{(k)}\right)$ bezeichnet dann die Matrix, die aus den ersten k Spalten von A besteht [26]. Es gilt $A_k = (A_{k-1}, A^{(k)})$ [26]. Der Algorithmus ist gegeben durch [26]

1. $k = 1$:

$$A_1^+ = \left(A^{(1)}\right)^+ = \begin{cases} \frac{\left(A^{(1)}\right)^T}{\left(A^{(1)}\right)^T A^{(1)}} & \text{falls } A^{(1)} \neq 0 \\ 0^T & \text{falls } A^{(1)} = 0 \end{cases}$$

2. $j = k \geq 2$:

$$\begin{aligned} d_j^T &= \left(A^{(j)}\right)^T (A_{j-1}^+)^T A_{j-1}^+ \\ c_j &= (I - A_{j-1}A_{j-1}^+)A^{(j)} \\ b_j^T &= c_j^+ + \frac{1 - c_j^+ c_j}{1 + d_j^T A^{(j)}} d_j^T \\ A_j^+ &= \left(A_{j-1}, A^{(j)}\right)^+ \\ &= \begin{pmatrix} A_{j-1}^+ (I - A^{(j)} b_j^T) \\ b_j^T \end{pmatrix}. \end{aligned}$$

Die Laufzeit des Verfahrens von Greville ist $O\left(m \cdot (nm + n^2 m + n + n^2 m)\right) = O(n^2 m^2)$.

2.5.5 Diskussion

Zusammenfassend ist festzustellen, dass die QR-Methode zur Bestimmung der Pseudoinverse schnell und stabil verläuft. Die Lösung der Methode der kleinsten Quadrate mit Hilfe der Pseudoinverse ist nach den folgenden zwei Schritten durchzuführen:

1. Berechne A^+ durch QR-Methode ; Laufzeit: $O(nm^2 + n^2 m)$;
2. Berechne $x_{\text{opt}} = A^+ b$; Laufzeit: $O(nm)$.

2.6 Verallgemeinerung der MKQ

In diesem Abschnitt wird die Verallgemeinerung der Methode der kleinsten Quadrate dargestellt. Die Gleichung zur Verallgemeinerung der MKQ lautet 2.6.1

$$Z = \min_{X \in \mathbb{R}^{n \times m'}} \|AX - B\|, \tag{2.6.1}$$

wobei $A \in \mathbb{R}^{n \times m}$, $B \in \mathbb{R}^{n \times m'}$ und $\|.\|$ Frobeniusnorm ist. Die Gleichung2.6.1 kann auch durch die oben genannten Methoden gelöst werden. Z. B. durch die Pseudoinverse wird das Problem gelöst mit

$$X_{\text{opt}} = A^+ B.$$

Wie bereits erwähnt, ist die QR-Zerlegung mit Hilfe der Givens-Rotationen die beste Methode für MKQ. Um die Lösung der Gleichung2.6.1 schnell zu berechnen, wird der folgende Algorithmus verwendet:

Algorithmus 2.2

Input: $A \in \mathbb{R}^{n\times m}$, $B \in \mathbb{R}^{n\times m'}$ und $m' \geq 1$;
Output: $X_{\text{opt}} \in \mathbb{R}^{m\times m'}$;

1. Bestimme Matrix R der QR-Zerlegung für Matrix $A \in \mathbb{R}^{n\times m}$ Mithilfe der Givens-Rotationen durch:

$$R = U(n, n-1, \varphi_{n,n-1}) \cdots U(n, 1, \varphi_{n,1}) \cdots U(3, 1, \varphi_{3,1})\, U(2, 1, \varphi_{2,1})\, A.$$

2. Berechne $D = Q^T B$ durch:

$$D = Q^T b = U(n, n-1, \varphi_{n,n-1}) \cdots U(n, 1, \varphi_{n,1}) \cdots U(3, 1, \varphi_{3,1})\, U(2, 1, \varphi_{2,1})\, B.$$

3. Löse $RX_{\text{opt}} = B$ durch Rückwärtseinsetzen.

4. Return X_{opt};

Nachfolgend wird eine Pseudocode formuliert, um die Lösung durch die Methode der kleinsten Quadrate zu berechnen:

```
Zweck: Berechnen die Lösung der MKQ
Parameter: n*m-Matrix A und n*m'-Matrix B

for j:=1 to m do
begin
  for i:=j+1 to n
  begin
    Bestimme U(i,j), so dass (U(i,j)*A)[i,j]=0;
    Berechne A:=U(i,j)*A;
    Berechne B:=U(i,j)*B;
  end
end
Bestimme X durch Rückewätseinsetzen;

Ergebnis: X
```

Die Laufzeit der Berechnung durch die Methode der kleinsten Quadrate mit Hilfe der QR-Zerlegung ist insgesamt $O(nm^2 - nmm' + m'm^2)$. Für $n \geq m \geq m'$ ist die Laufzeit $O(nm^2)$.

Kapitel 3

Datenstromalgorithmen für Regression

Im Kapitel 2 werden einige klassische Algorithmen für Regression vorgestellt. Diese Algorithmen können aber nicht direkt für Datenströme angewendet werden. In der Regel sind die Datenströme bei z.B. Messungen sehr lang und können nicht vollständig gespeichert werden. Für solche Probleme werden Datenstrom-Algorithmen benötigt. In diesem Kapitel werden einige Datenstrom-Algorithmen für Regression vorgestellt.

3.1 Time Series Modell für die Methode der kleinsten Quadrate

Die verschiedenen Modelle der Datenströme unterscheiden sich durch ihre Aktualisierung (siehe Abschnitt 1.2). In diesem Abschnitt werden zwei Datenstromalgorithmen für Regression im Time Series Modell erklärt. Die Daten werden hierbei immer in der Reihenfolge des Indexes empfangen.

3.1.1 Das Algorithmus von Zhou

Q. Zhou hatte im Jahr 2008 [43] einen Datenstromalgorithmus für Regression, der auf Gauss'sche Normalgleichung basiert, entwickelt. Die Normalgleichung ist die notwendige Bedingung der Methode der kleinsten Quadrate. Für ein lineares Gleichungssystem $Ax = b$, wobei $A \in \mathbb{R}^{n \times m}$ und $b \in \mathbb{R}^n$, wird die Normalgleichung berechnet

durch:

$$A^T Ax = A^T b. \tag{3.1.1}$$

Es wird deutlich, dass die Größe der Matrix $M = A^T A \in \mathbb{R}^{m \times m}$ und des Spaltenvektors $d = A^T b$ nicht von n (Anzahl der Matrixzeilen) abhängig ist. Diese Eigenschaft wird im Folgenden für Datenstromalgorithmus verwendet.

Die Eingabe besteht aus einem Strom von Daten

$$\begin{array}{llll} A_{(1)}, & A_{(2)}, & \cdots, & A_{(i)} \quad \cdots \\ \text{und} & & & \\ b_{(1)}, & b_{(2)}, & \cdots & b_{(i)}, \quad \cdots \end{array}$$

wobei $A_{(i)}$ den i-ten Zeilenvektor von Matrix A ist und $b_{(i)}$ des i-te Element des Spaltenvektors b ist. Sie werden nacheinander als Eingabewerte gelesen. Nach der i-te Eingabe ($A_{(i)}$ und $b_{(i)}$) erhält man $A_i \in \mathbb{R}^{i \times m}$ und $b_i \in \mathbb{R}^i$. Die Matrix M_i und der Spaltenvektor d_i werden berechnet durch

$$\begin{array}{rcl} M_i & = & A_i^T A_i \\ \text{und} & & \\ d_i & = & A_i^T b_i \end{array} \tag{3.1.2}$$

Für die $(i+1)$-te Eingabe wird die Matrix M_{i+1} berechnet durch [43]

$$\begin{aligned} M_{i+1} &= A_{i+1}^T A_{i+1} \\ &= \begin{bmatrix} A_i \\ A_{(i+1)} \end{bmatrix}^T \begin{bmatrix} A_i \\ A_{(i+1)} \end{bmatrix} \\ &= A_i^T A_i + A_{(i+1)}^T A_{(i+1)} \\ &= M_i + A_{(i+1)}^T A_{(i+1)}. \end{aligned} \tag{3.1.3}$$

Nach der gleichen Rechenweise wird der Vektor $d_{i+1} = A_{i+1}^T b_{i+1}$ berechnet durch:

$$\begin{aligned} d_{i+1} &= A_{i+1}^T b_{i+1} \\ &= \begin{bmatrix} A_i \\ A_{(i+1)} \end{bmatrix}^T \begin{bmatrix} b_i \\ b_{(i+1)} \end{bmatrix} \\ &= A_i^T b_i + A_{(i+1)}^T b_{(i+1)} \\ &= d_i + A_{(i+1)}^T b_{(i+1)}. \end{aligned} \tag{3.1.4}$$

Hierbei werden sehr große Mengen von Daten unter Verwendung von sehr wenig Speicherplatz verarbeitet, weil n sehr sehr ($<<$) groß ist. Durch n Schritte werden die

Matrix $M = M_n$ und der Vektor $d = d_n$ der Normalgleichung berechnet. Schließlich wird nach den klassischen Algorithmus der Regression die Lösung x_{opt} des linearen Gleichungssystems $Mx = d$ ermittelt:

$$x_{opt} = M^{+}d, \tag{3.1.5}$$

wobei die Matrix M^{+} Pseudoinverse der Matrix M ist. Da die Matrix M symmetrisch und positiv definit ist, stehen die Verfahren der Cholesky-Zerlegung zur Verfügung (siehe Abschnitt 2.2).

Für diesen Algorithmus wird $O(m^2)$ Speicherplatz benötigt. Außerdem ist die Laufzeit in einem Schritt $O(m^2)$. Zu beachten ist, dass ein Zeilenvektor (m Matrixelemente) pro Schritt aktualisiert wird. Die Aktualisierungszeit pro Matrixelement ist $O(m)$. Der Pseudocode des Algorithmus sieht wie folgt aus:

```
Zweck: Berechnen die Lösung durch den Algorithmus von Zhou

Initialisiere Matrix M := 0;
Initialisiere Vektor d := 0;

repeat
  Ai := lese nächsten Zeilenvektor von Matrix A;
  bi := lese nächstes Element des Vektors b;
  Berechne AiT := Transponieren des Vektors Ai;
  Berechne M   := M + AiT * Ai;
  Berechne d   := d + Ait * b;
until DatenStrom beendet;

Bestimme X durch Cholesky-Zerlegung;

Ergebnis: X
```

Beim Algorithmus tritt das Problem der relativen Fehler auf, weil hier die Normalgleichung verwendet wird (siehe Abschnitt 2.2.1). Die Kondition des linearen Gleichungssystems verschlechtert sich durch die Berechnung $A^T A$. Die Diagonalelemente der Matrix $A_{(i)}^T A_{(i)}$ sind quadratische Zahlen. Somit ist z.B. das k-te Diagonalelement

$M_{(k)}^{(k)}$ der Matrix M:

$$M_{(k)}^{(k)} = \sum_{i=1}^{n} \left(A_{(i)}^{(k)} \right)^2 . \tag{3.1.6}$$

Da n sehr groß ist, tritt das Risiko auf, dass die Diagonalelemente der Matrix M überlaufen werden könnten.

3.1.2 QR-Datenstromalgorithmus

Die Methode der kleinsten Quadrate von Datenströmen kann direkt, ohne die Normalgleichung zu lösen, behandelt werden. Dafür kann die QR-Zerlegung verwendet werden. Es ist ein lineares Gleichungssystem $Ax = b$, wobei $A \in \mathbb{R}^{n \times m}$ und $b \in \mathbb{R}^n$ und die Eingabedaten

$$\begin{array}{llll} A_{(1)}, & A_{(2)}, & \cdots, & A_{(i)} \quad \cdots \\ \text{und} & & & \\ b_{(1)}, & b_{(2)}, & \cdots & b_{(i)}, \quad \cdots \end{array}$$

nach einem bestimmten Reihenfolge gelesen werden. Nach der$i$-te Eingabe erhaltet man $A_i \in \mathbb{R}^{i \times m}$ und $b_i \in \mathbb{R}^i$. Nun ist es möglich, die Matrix A_i als Produkt von zwei Matrizen Q_i und R_i zu zerlegen, wobei $Q_i \in \mathbb{R}^{i \times i}$ orthogonal und $R_i \in \mathbb{R}^{i \times m}$ eine obere Rechtsdreiecksmatrix ist [29].

$$R_i = \begin{bmatrix} * & \cdots & * \\ & \ddots & \vdots \\ 0 & & * \\ 0 & \cdots & 0 \\ \vdots & & \vdots \\ 0 & \cdots & 0 \end{bmatrix} = \begin{bmatrix} R_{i1} \\ 0 \end{bmatrix}, \; R_{i1} \in \mathbb{R}^{m \times m}$$

Es erhebt sich nun die Frage, ob man den 0-Anteil der Matrix R_i löschen darf, um den Speicherverbrauch zu reduzieren. Um die Frage zu beantworten, wird die Theorie der MKQ mit der QR-Zerlegung nochmal analysiert.

Die Grundidee der MKQ mit der QR-Zerlegung ist, dass der Residuenvektor $r_i = A_i x - b_i$ mit Q_i^T multipliziert werden darf, ohne die Länge zu verändern, d. h.

[7]:

$$\begin{aligned} \|r_i\|^2 &= \left\|Q_i^T\left(A_i x - b_i\right)\right\|^2 \\ &= \left\|R_i x - Q_i^T b_i\right\|^2 \\ &= \left\| \begin{bmatrix} R_{i1} \\ 0 \end{bmatrix} x - \begin{bmatrix} d_{i1} \\ d_{i2} \end{bmatrix} \right\|^2 \\ &= \|R_{i1}x - d_{i1}\|^2 + \|d_{i2}\|^2, \end{aligned} \tag{3.1.7}$$

mit $d_i = Q_i^T b_i = \begin{bmatrix} d_{i1} \\ d_{i2} \end{bmatrix}$. Man erhält (siehe Abschnitt 2.3):

$$R_{i1}x = d_{i1} \Leftrightarrow A_i x = b_i.$$

Es werden nur R_{i1} und d_{i1} gespeichert. Nach dem Einlesen von $A_{(i+1)}$ und $b_{(i+1)}$ wird überprüft, ob nur mit R_{i1} und d_{i1} weiter gerechnet werden kann.

Die Länge des Residuenvektores $r_{i+1} = A_{i+1}x - b_{i+1}$ wird berechnet durch:

$$\begin{aligned} \|r_{i+1}\|^2 &= \left\| \begin{bmatrix} 1 & 0 \\ 0 & Q_i \end{bmatrix}^T \left(A_{i+1}x - b_{i+1}\right) \right\|^2 \\ &= \left\| \begin{bmatrix} 1 & 0 \\ 0 & Q_i^T \end{bmatrix} \left(\begin{bmatrix} A_{(i+1)} \\ A_i \end{bmatrix} x - \begin{bmatrix} b_{(i+1)} \\ b_i \end{bmatrix} \right) \right\|^2 \\ &= \left\| \begin{bmatrix} A_{(i+1)} \\ Q_i^T A_i \end{bmatrix} x - \begin{bmatrix} b_{(i+1)} \\ Q_i^T b_i \end{bmatrix} \right\|^2 \\ &= \left\| \begin{bmatrix} A_{(i+1)} \\ R_{i1} \\ 0 \end{bmatrix} x - \begin{bmatrix} b_{(i+1)} \\ d_{i1} \\ d_{i2} \end{bmatrix} \right\|^2 \\ &= \left\| \begin{bmatrix} A_{(i+1)} \\ R_{i1} \end{bmatrix} x - \begin{bmatrix} b_{(i+1)} \\ d_{i1} \end{bmatrix} \right\|^2 + \|d_{i2}\|^2. \end{aligned} \tag{3.1.8}$$

Also wird $\|r_{i+1}\|^2$ genau dann minimiert, wenn

$$\left\| \begin{bmatrix} A_{(i+1)} \\ R_{i1} \end{bmatrix} x - \begin{bmatrix} b_{(i+1)} \\ d_{i1} \end{bmatrix} \right\|^2$$

minimiert wird. Das Gleichungssystem $A_{i+1}x = b_{i+1}$ wird in

$$\begin{bmatrix} A_{(i+1)} \\ R_{i1} \end{bmatrix} x = \begin{bmatrix} b_{(i+1)} \\ d_{i1} \end{bmatrix} \tag{3.1.9}$$

umgewandelt, wobei

$$\begin{bmatrix} A_{(i+1)} \\ R_{i1} \end{bmatrix} = \underbrace{\begin{bmatrix} A^{(1)}_{(i+1)} & A^{(2)}_{(i+1)} & \cdots & \cdots & A^{(m)}_{(i+1)} \\ * & * & * & * & * \\ 0 & * & * & * & * \\ \vdots & 0 & * & * & * \\ \vdots & 0 & 0 & * & * \\ 0 & \cdots & \cdots & 0 & * \end{bmatrix}}_{\in \mathbb{R}^{(m+1)\times m}} . \tag{3.1.10}$$

Es wird deutlich, dass nun m Einträge, die ungleich 0 sind, unterhalb der Hauptdiagonale der Matrix $\begin{bmatrix} A_{(i+1)} \\ R_{i1} \end{bmatrix}$ liegen. Durch die Givens-Rotationen wird die Matrix $\begin{bmatrix} A_{(i+1)} \\ R_{i1} \end{bmatrix}$ nach der Zeit $O(m^2)$ in eine obere Rechtsdreiecksmatrix umgerechnet (siehe Abschnitt 2.3.1).

Der Pseudocode des Algorithmus ist im folgenden Fenster dargestellt:

```
Zweck: Berechnen der Lösung durch den
       QR-Datenstromalgorithmus

Initialisiere (m+1)*m Matrix A := 0;
Initialisiere m+1 Elementen Vektor b := 0;

repeat
  Ai := lese nächste Zeilenvektor von Matrix A;
  bi := lese nächste Element des Vektors b;
  Verschiebe Elemente der Matrix A nach unten um 1 Zeile;
  Verschiebe Elemente des Vektors b nach unten um 1 Zeile;
  A  := A + (Ai als erste Zeile);
  b  := b + (bi als erste Element);
  for j:=1 to m
  begin
    Bestimme U(j+1,j), so dass (U(j+1,j)*A)[j+1,j]=0;
    Berechne A:=U(j+1,j)*A;
    Berechne b:=U(j+1,j)*b;
  end
until DatenStrom beendet;

Bestimme x durch Rückewätseinsetzen;

Ergebnis: x
```

Der obige Pseudocode zeigt, dass für den Algorithmus nur $O(m^2)$ Speicherplatz benötigt wird. Wenn eine Zeile der Matrix A gelesen wird, werden maximal m Givens-Rotationen benötigt, um die Elemente unterhalb der Hauptdiagonale zu 0 umzurechnen. Der Berechnungsaufwand einer Givens-Rotation ist $O(m)$. Insgesamt ist die Aktualisierungszeit $O(m^2)$ pro Zeile und $O(m)$ pro Element.

3.2 Turnstile Modell für Methode der kleinsten Quadrate

Während die Datenströme beim Time Series Modell in der Reihenfolge des Indexes verarbeitet werden müssen, spielt ihre Reihenfolge beim Turnstile Modell keine Rolle.

In diesem Abschnitt wird ein Datenstrom-Algorithmus mit dem Turnstile Modell vorgestellt. Der Algorithmus wurde von K. L. Clarkson und D. P. Woodruff in der Literatur [6] veröffentlicht.

3.2.1 Clarkson-Woodruff Algorithmus

Der Algorithmus von K. L. Clarkson und D. P. Woodruff berechnet das gleiche Datenstrom-Problem für ein lineares Gleichungssystem $Ax = b$, wobei $A \in \mathbb{R}^{n \times m}$ und $b \in \mathbb{R}^n$ ist. Die Eingabedaten

$$\begin{array}{llll} A_{(1)}, & A_{(2)}, & \cdots, & A_{(i)} \quad \cdots \\ \text{und} & & & \\ b_{(1)}, & b_{(2)}, & \cdots & b_{(i)}, \quad \cdots \end{array}$$

können nicht nacheinander gelesen werden. Die Matrix A und der Vektor b werden durch das Turnstile Modell aktualisiert. Wenn ein Tupel (i, j, v, A) eintrifft, wird das Element $A^{(j)}_{(i)}$ aktualisiert durch:

$$A^{(j)}_{(i)} \leftarrow A^{(j)}_{(i)} + v. \tag{3.2.1}$$

Die Elemente des Vektors b werden ähnlich aktualisiert. D. h. dass das Element b_i aktualisiert wird durch:

$$b_i \leftarrow b_i + v, \tag{3.2.2}$$

wenn ein Tupel (i, v, b) eintrifft.

Formulierung

Um die Datenströme mit dem eingeschränkten Speicher zu verarbeiten, formulieren Clarkson und Woodruff das lineare Gleichungssystem $Ax = b$ in $S^T Ax = S^T b$ um, wobei die Matrix S eine randomisierte $n \times d$ Sign-Matrix ist:

$$S \in \mathbb{R}^{n \times d} \text{ mit } s_{i,j} = \begin{cases} +1 & \text{mit Wahrscheinlichkeit } \frac{1}{2} \\ -1 & \text{mit Wahrscheinlichkeit } \frac{1}{2}. \end{cases} \tag{3.2.3}$$

Durch die Umformulierung erhalten man eine Matrix $A' = S^T A \in \mathbb{R}^{d \times m}$ und einen Spaltenvektor $b' = S^T b \in \mathbb{R}^d$. Es wird deutlich, dass die Größe der Matrix A' und des Spaltenvektors b' nicht von n abhängig sind.

Aktualisierung und Berechnung

Für die Aktualisierung werden zwei Fragen aufgestellt:

1. Wie kann man die Matrix A' und den Vektor b' berechnen, ohne die Matrix A und den Vektor b zu speichern?
2. Wie kann man die Sign-Matrix S berechnen?

Die Matrix $S \in \mathbb{R}^{n \times d}$ ist sehr groß und kann nicht gespeichert werden. Prof. C. Sohler schlägt vor, dass eine Hashfunktion

$$h(x) : \{1, \cdots, n\} \times \{1, \cdots, d\} \rightarrow \{-1, +1\} \tag{3.2.4}$$

als Sign-Matrix S verwendet werden soll. Wenn ein Tupel $a_{i,j} \leftarrow (i, j, v, A)$ gelesen wird, wird die Matrix A' direkt aktualisiert durch:

$$\underbrace{\begin{bmatrix} * & * & s_{i,1} & * & * \\ * & * & . & * & * \\ * & * & . & * & * \\ * & * & . & * & * \\ * & * & s_{i,d} & * & * \end{bmatrix}}_{S^T \in \mathbb{R}^{d \times n}} \cdot \underbrace{\begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ * & * & a_{ij} \leftarrow a_{ij} + v & * & * \\ * & * & * & * & * \\ * & * & * & * & * \end{bmatrix}}_{A \in \mathbb{R}^{n \times m}} \leftarrow i \; = \underbrace{\begin{bmatrix} * & * & a'_{1,j} & * & * \\ * & * & . & * & * \\ * & * & . & * & * \\ * & * & . & * & * \\ * & * & a'_{d,j} & * & * \end{bmatrix}}_{A' \in \mathbb{R}^{d \times m}} \tag{3.2.5}$$

wobei

$$a'_{l,j} = a'_{l,j} + s_{i,l} \cdot v \text{ für all } l = \{1, \cdots, d\}. \tag{3.2.6}$$

Die Elemente des Vektors b' werden ähnlich aktualisiert, wenn ein Tupel $b_i \leftarrow (i, v, b)$ gelesen wird:

$$\underbrace{\begin{bmatrix} * & * & s_{i,1} & * & * \\ * & * & . & * & * \\ * & * & . & * & * \\ * & * & . & * & * \\ * & * & s_{i,d} & * & * \end{bmatrix}}_{S^T \in \mathbb{R}^{d \times n}} \cdot \underbrace{\begin{bmatrix} * \\ * \\ b_i \leftarrow b_i + v \\ * \\ * \end{bmatrix}}_{b \in \mathbb{R}^n} \leftarrow i \; = \underbrace{\begin{bmatrix} b'_1 \\ . \\ . \\ . \\ b'_d \end{bmatrix}}_{b' \in \mathbb{R}^d} \tag{3.2.7}$$

wobei

$$b'_l = b'_l + s_{i,l} \cdot v \text{ für all } l = \{1, \cdots, d\}. \tag{3.2.8}$$

Die obige Aktualisierung zeigt, dass die Aktualisierungszeit für ein Tupel $O(d)$ nur von d abhängig ist.

Qualität des Clarkson-Woodruff Algorithmus

Beim Ende des Datenstroms werden die Matrix A' und der Vektor b' vollständig berechnet. Danach kann mit dem klassischen Algorithmus der Regression die Lösung x^*_{opt} des linearen Gleichungssystems

$$\underbrace{S^T A}_{A'} x = \underbrace{S^T b}_{b'} \tag{3.2.9}$$

ermittelt werden:

$$x^*_{\text{opt}} = (A')^+ d'. \tag{3.2.10}$$

Die letzte offene Frage des Clarkson-Woodruff Algorithmus ist, wie groß die Abweichung zwischen $r_{\min} = \|Ax_{\text{opt}} - b\|$ und $r^*_{\min} = \left\|Ax^*_{\text{opt}} - b\right\|$ ist, wobei x_{opt} die Lösung des originalen linearen Gleichungssystems $Ax = b$.

Mit Hilfe der Singulärwertzerlegung $A = U\Sigma V = U_A \Sigma_A V_A^T$ erhält man die minimale Länge des Residuenvektors $r_{\min}$:

$$r_{\min} = \min_{x \in \mathbb{R}^n} \|Ax - b\| = \left\| U_A^{\perp} \left(U_A^{\perp}\right)^T b \right\| = b^{\perp},$$

wobei $U_A \in \mathbb{R}^k$, $U_A^{\perp} \in \mathbb{R}^{n-k}$ und k der Rang der Matrix A ist (siehe Abschnitt 2.4.3). Führt man nun eine kleine Gleichungsumformung von $\left\|Ax^*_{\text{opt}} - b\right\|^2$ durch, erhält man [6]:

$$\begin{aligned} \|r^*_{\min}\|^2 &= \left\|Ax^*_{\text{opt}} - b\right\|^2 \\ &= \left\|b - Ax_{\text{opt}} + Ax_{\text{opt}} - Ax^*_{\text{opt}}\right\|^2 \\ &= \left\| \underbrace{(b - Ax_{\text{opt}})}_{=b^{\perp}=r_{\min}} + A(x_{\text{opt}} - x^*_{\text{opt}}) \right\|^2 \end{aligned} \tag{3.2.11}$$

Da $b^{\perp}$ senkrecht zu $A(x_{\text{opt}} - x^*_{\text{opt}})$ ist, könnte hier das Theorem 1.5 (Satz des Pythagoras) in der Literatur [6] verwendet werden. So erhält man:

$$\|r^*_{\min}\|^2 = \|r_{\min}\|^2 + \left\|A(x_{\text{opt}} - x^*_{\text{opt}})\right\|^2. \tag{3.2.12}$$

a. Umschreibung der MKQ [3]

$$\begin{aligned}
\min_{x\in\mathbb{R}^m} \left\|S^Tb - S^TAx\right\|^2 &= \min_{y\in\mathbb{R}^m} \left\| S^T \underbrace{\left(Ax_{\text{opt}} + b^\perp\right)}_{=b} - S^TA\underbrace{\left(x_{\text{opt}} + y\right)}_{=x} \right\|^2 \\
&= \min_{y\in\mathbb{R}^m} \left\|S^Tb^\perp - S^TAy\right\|^2 \\
&= \min_{y\in\mathbb{R}^m} \left\| S^Tb^\perp - S^TU_A\underbrace{\Sigma_AV_A^Ty}_{=z} \right\|^2 \\
&= \min_{z\in\mathbb{R}^m} \left\|S^Tb^\perp - S^TU_Az\right\|^2 \\
&= \left\|S^Tb^\perp - S^TU_Az_{\text{opt}}\right\|^2.
\end{aligned} \tag{3.2.13}$$

b. weitere Umschreibung der MKQ [3]

$$\begin{aligned}
\min_{x\in\mathbb{R}^m} \left\|S^Tb - S^TAx\right\|^2 &= \left\|S^Tb - S^TAx^*_{\text{opt}}\right\|^2 \\
&= \left\|S^T\left(Ax^*_{opt} + b^\perp\right) - S^TA\left(x_{\text{opt}} + x^*_{\text{opt}} - x_{\text{opt}}\right)\right\|^2 \\
&= \left\|S^Tb^\perp - S^TA\left(x^*_{opt} - x_{\text{opt}}\right)\right\|^2.
\end{aligned} \tag{3.2.14}$$

Beim Vergleich der Gleichungen 3.2.13 und 3.2.14 stellt man fest, dass

$$U_Az_{\text{opt}} = A\left(x^*_{\text{opt}} - x_{\text{opt}}\right) \tag{3.2.15}$$

ist. Die Gleichung 3.2.13 wird in der Gauss'sche Normalgleichung umschreiben (siehe Abschnitt 2.2.1) [3]

$$\begin{aligned}
\left(S^TU_A\right)^T\left(S^TU_A\right)z_{\text{opt}} &= \left(S^TU_A\right)^TS^Tb^\perp \\
\Rightarrow\ U_A^TSS^TU_Az_{\text{opt}} &= U_A^TSS^Tb^\perp.
\end{aligned} \tag{3.2.16}$$

Mit Hilfe der Ungleichung $\|C + D\| \leq \|C\| + \|D\|$ kann $\|z_{\text{opt}}\|$ berechnet werden:

$$\begin{aligned}
\|z_{\text{opt}}\| &= \left\| \frac{U_A^TSS^TU_Az_{\text{opt}}}{d} - z_{\text{opt}} - \frac{U_A^TSS^TU_Az_{\text{opt}}}{d} \right\| \\
&\leq \left\| \frac{U_A^TSS^TU_Az_{\text{opt}}}{d} - z_{\text{opt}} \right\| + \left\| \frac{U_A^TSS^TU_Az_{\text{opt}}}{d} \right\|.
\end{aligned} \tag{3.2.17}$$

Mit der Ungleichung $\|CD\| \leq \|C\|_2 \|D\|$ und der Gleichung 3.2.16 kann $\|z_{\text{opt}}\|$ weiter berechnet werden [6]:

$$\begin{aligned}
\|z_{\text{opt}}\| &\leq \left\| \frac{U_A^T S S^T U_A}{d} - I \right\|_2 \|z_{\text{opt}}\| + \left\| \frac{U_A^T S S^T b^\perp}{d} \right\| \\
&= \left\| \frac{U_A^T S S^T U_A}{d} - I \right\|_2 \|z_{\text{opt}}\| + \left\| \frac{U_A^T S S^T b^\perp}{d} - \underbrace{U_A^T \overbrace{U_A^\perp \left(U_A^\perp\right)^T b}^{=b^\perp}}_{=0} \right\| \\
&= \left\| \frac{U_A^T S S^T U_A}{d} - I \right\|_2 \|z_{\text{opt}}\| + \left\| \frac{U_A^T S S^T b^\perp}{d} - U_A^T b^\perp \right\|,
\end{aligned} \tag{3.2.18}$$

da $U_A^\perp$ senkrecht zu U_A ist. Mit dem Theorem 2.2 und dem Theorem 3.4 in der Literatur [6] und dem entsprechenden d

$$\begin{aligned}
d &= O\left(2k \log\left(1/\delta\right)/\epsilon\right) \\
&= O(\log\left(1/\delta\right) / \left(\sqrt{\epsilon/2k}\right)^2) \\
&= O(\log\left(1/\delta\right)/\epsilon_0^2)
\end{aligned} \tag{3.2.19}$$

erfüllt die Ungleichung

$$\begin{aligned}
\|z_{\text{opt}}\| &\leq \frac{\epsilon}{2} \|z_{\text{opt}}\| + \epsilon_0 \|U_A\| \cdot \left\|b^\perp\right\| \\
&= \frac{\epsilon}{2} \|z_{\text{opt}}\| + \sqrt{\epsilon/2k} \underbrace{\|U_A\|}_{=\sqrt{k}} \cdot \underbrace{\left\|b^\perp\right\|}_{=r_{\min}} \\
\Rightarrow (1 - \frac{\epsilon}{2}) \|z_{\text{opt}}\| &\leq \sqrt{\epsilon/2} r_{\min} \\
\Rightarrow \|z_{\text{opt}}\| &\leq \frac{\sqrt{\epsilon/2}}{1 - \epsilon/2} r_{\min} \\
&= \frac{\sqrt{2\epsilon}}{2 - \epsilon} r_{\min} \\
&\leq \sqrt{2\epsilon} r_{\min}
\end{aligned} \tag{3.2.20}$$

mit einer Wahrscheinlichkeit von mindestens $1 - \delta$, wobei $\epsilon \in (0, 1)$, $\delta \in (0, 1)$ und k

den Rang der Matrix $A \in \mathbb{R}^{n\times m}$ ist. Aus 3.2.12, 3.2.15 und 3.2.20 folgt:

$$\begin{aligned}\|r^*_{\min}\|^2 &= \|r_{\min}\|^2 + \underbrace{\left\|A(x_{\text{opt}} - x^*_{\text{opt}})\right\|^2}_{=\|U_A z_{\text{opt}}\|} \\ &= \|r_{\min}\|^2 + \|Z_{\text{opt}}\|^2 \\ &\leq \|r_{\min}\|^2 + 2\epsilon \|r_{\min}\|^2 \\ &\leq (1+2\epsilon)\|r_{\min}\|^2 .\end{aligned} \tag{3.2.21}$$

Mit Hilfe der Ungleichung $\sqrt{1+2\epsilon} \leq 1+\epsilon$ erhält man schließlich:

$$\begin{aligned}\|r^*_{\min}\| &\leq \sqrt{1+2\epsilon}\,\|r_{\min}\| \\ &\leq (1+\epsilon)\|r_{\min}\| .\end{aligned} \tag{3.2.22}$$

Die Ungleichung 3.2.22 ist mit einer Wahrscheinlichkeit von mindestens $1-\delta$ erfüllt.

3.2.2 Hashfunktion

Für Clarkson-Woodruff Algorithmus wird eine zusätzliche Sign-Matrix $S \in \mathbb{R}^{n\times d}$ benötigt, wobei $d = O\left(k\log\left(1/\delta\right)/\epsilon\right)$, $\epsilon \in (0,1)$, $\delta \in (0,1)$ und k den Rang der Matrix $A \in \mathbb{R}^{n\times m}$ ist. Die Größe der Sign-Matrix S hängt von n ab und ist daher groß. Es mag sein, dass die Sign-Matrix S größer als die Matrix A wird, wenn man ϵ und δ entsprechend wählt. Wegen ihrer Größe kann die Sign-Matrix kaum oder gar nicht speichert werden. Eine Lösung zur Speicherung wäre, eine Hashfunktion

$$h(x) : \{1,\cdots,n\} \times \{1,\cdots,d\} \to \{-1,+1\} \tag{3.2.23}$$

für die Sign-Matrix S zu verwenden. In diesem Abschnitt werden einige Hashfunktionen für die Sing-Matrix S dargestellt.

Divisions-Rest-Methode

Mit der Divisions-Rest-Methode kann der Hashwert von Schlüssel k sehr schnell berechnet werden, indem der Rest des k durch m ausgegeben wird [4]. Das heißt, dass die Hashfunktion

$$h_d\left(k\right) = k \bmod m \tag{3.2.24}$$

lautet. Die Hashfunktion 3.2.24 kann aber nicht direkt in dem Clarkson-Woodruff Algorithmus verwendet werden, weil in der Sign-Matrix S nur die Werte -1 oder/und $+1$ vorhanden sein dürfen. Um die Divisions-Rest-Methode in dem Clarkson-Woodruff

Algorithmus zu verwenden, muss die Hashfunktion 3.2.24 weiter durch 2 dividiert werden:

$$h_d(k) = (k \bmod m) \bmod 2. \tag{3.2.25}$$

Der Rest der Hashfunktion 3.2.25 ist $+1$ oder 0. Wenn man den Rest der Hashfunktion 3.2.25 0 in -1 umrechnet, erhält man eine Hashfunktion, genau wie die in Abbildung 3.2.23 zu sehen. Eine gute Hashfunktion für Clarkson-Woodruff Algorithmus erfüllt die Annahme des zufälligen Hashings: Jeder Schlüssel wird mit der gleichen Wahrscheinlichkeit $\frac{1}{2}$ auf die Werte $\{-1, +1\}$ abgebildet. Für die Qualität der Hashfunktion 3.2.25 spielt der Divisor m eine groß Rolle. Wenn m eine Potenz von 2 ist (z. B. $m = 2^p$), erhält man durch die Hashfunktion genau die p niederwertigsten Bits von k [4]. Potenzwerte mit Basis 2 sollten für m vermieden werden. Wie in der Literatur [4] beschrieben, kann hier $m = 701$ verwendet werden. Der Pseudocode sieht bei der Hashfunktion der Divisions-Rest-Methode wie folgt aus:

```
Zweck: Berechnen ein Element
       der n*d Sign-Matrix S von Index (i,j)

berechne den Schlüssel k := i*d + j;
berechne h := k mod 701;
berechne h := h mode 2;
if h==0 then h := -1;

Ergebnis: h
```

Multiplikative Methode

Die Multiplikative Methode zur Erstellung von Hashfunktionen arbeitet in zwei Schritten [4]. Zunächst wird der Schlüssel k mit einer Zahl A im Bereich $0 \leq A \leq 1$ multipliziert und den Bruchteil von kA extrahiert, so dass der Schlüssel k in das Intervall $[0, 1]$ abgebildet wird [4]. Dann multipliziert man das Ergebnis mit der Anzahl der Hashtabellenadressen m und runden das Produkt nach unten ab [4, 33]. Die beiden Schritte werden in Formelschreibweise geschrieben als [4]:

$$\begin{aligned} h_m(k) &= \lfloor m(kA \bmod 1) \rfloor \\ &= \lfloor m(kA - \lfloor kA \rfloor) \rfloor . \end{aligned} \tag{3.2.26}$$

Die Anzahl der Hashtabellenadressen m ist 2, weil die Hashfunktion in $\{-1,+1\}$ abgebildet werden soll. Die Qualität der Hashfunktion der Multiplikative Methode ist nur von der Konstanten A abhängig. Donald E. Knuth schlägt vor, dass mit

$$A \approx \frac{\sqrt{5}-1}{2} \approx 0,6180339887 \tag{3.2.27}$$

die Hashfunktion wahrscheinlich recht gut arbeitet [4]. Um den Vorschlag anzupassen, wird

$$A = 2654435769/2^{32} \tag{3.2.28}$$

gewählt [4]. Der Pseudocode der Hashfunktion der Multiplikative Methode ist, siehe unten, abgebildet:

```
Zweck: Berechnen ein Element
        der n*d Sign-Matrix S von Index (i,j)

berechne den Schlüssel k := i*d + j;
berechne A := 2654435769/4294967296;
berechne h := k*A;
berechne h1:= runden h nach unten ab;
berechne h := h - h1;
berechne h := h*2;
berechne h := runden h nach unten ab;
if h==0 then h := -1;

Ergebnis: h
```

Gemischte Hashfunktion

Die multiplikative Methode und die Divisions-Rest-Methode kann in einer Hashfunktion verwendet werden. Der Hashwert des Schlüssels k wird durch zwei Schritten berechnet. Zunächst wird ein Hashwert des Schlüssels k durch die Multiplikative Methode berechnet:

$$h_m(k) = \lfloor m(kA - \lfloor kA \rfloor) \rfloor . \tag{3.2.29}$$

Hier wird m als Potenz von 2 gewählt, da der Hashwert der Multiplikationsmethode von m nicht kritisch ist [4]. Beispielsweise könnte man $m = 2^{14} = 16384$ nehmen [4].

Danach wird der Wert weiter durch die Divisions-Rest-Methode berechnet:

$$h_g(k) = (h_m(k) \bmod 701) \bmod 2. \tag{3.2.30}$$

Zufallsmatrix Hashfunktion

Für diese Hashfunktion wird zunächst eine Zufallsmatrix berechnet. Z. B. wird eine Zufallsmatrix $K \in \mathbb{Z}^{701\times 701}$ durch die Zufallsfunktion berechnet, wobei nur -1 und $+1$ in der Matrix K vorhanden sind. Dann kann man den Hashwert des Schlüssel k durch die Hashfunktionen (z. B. Divisions-Rest-Methode und multiplikative Methode) in der Zufallsmatrix K auslesen. Z. B. berechnet man den Wert für Index (i, j) der Matrix S mit dem folgenden Pseudocode:

```
Zweck: Berechnen ein Element
       der n*d Sign-Matrix S von Index (i,j)

berechne den Schlüssel k := i*d + j;
berechne km := der Hashwert des Schlüssel k
       durch multiplikative Methode;
abhole Hashwert h := K[k mod 701][km mod 701];

Ergebnis: h
```

SHA1.

Kryptologische Hashfunktionen können auch für die Sign-Matrix $S \in \mathbb{R}^{n\times d}$ verwendet werden. SHA1 und MD5 sind die bekanntesten Verfahren für Kryptologische Hashfunktionen. In diesem Abschnitt wird die SHA1 Hashfunktion vorgestellt. Die Abbildung 3.2.1 zeigt die schematische Darstellung der SHA1 Hashfunktion. Der Eingabewert ist hier eine Zeichenkette, die in n 512-Bit-Blöcken zerlegt wird. SHA1 verarbeitet die 512-Bit-Blöcken nacheinander. Zunächst wird jeder Block bei Eingabe in 80 Wörter expandiert [21, 27]. Nach der Eingabe der Zeichenkette werden die 80 Wörter nacheinander durch eine Schleife verarbeitet [21, 25]. Schließlich wird ein Hashwert ausgegeben. Bei Index (i, j) der Sign-Matrix $S \in \mathbb{R}^{n\times d}$ wird ein Schlüssel $k := i \cdot d + j$ berechnet. Man kann die Schlüssel k als Zeichenkette in der SHA1 Hashfunktion eingeben. Schließlich wird einen Hashwert berechnet. Den Hashwert kann

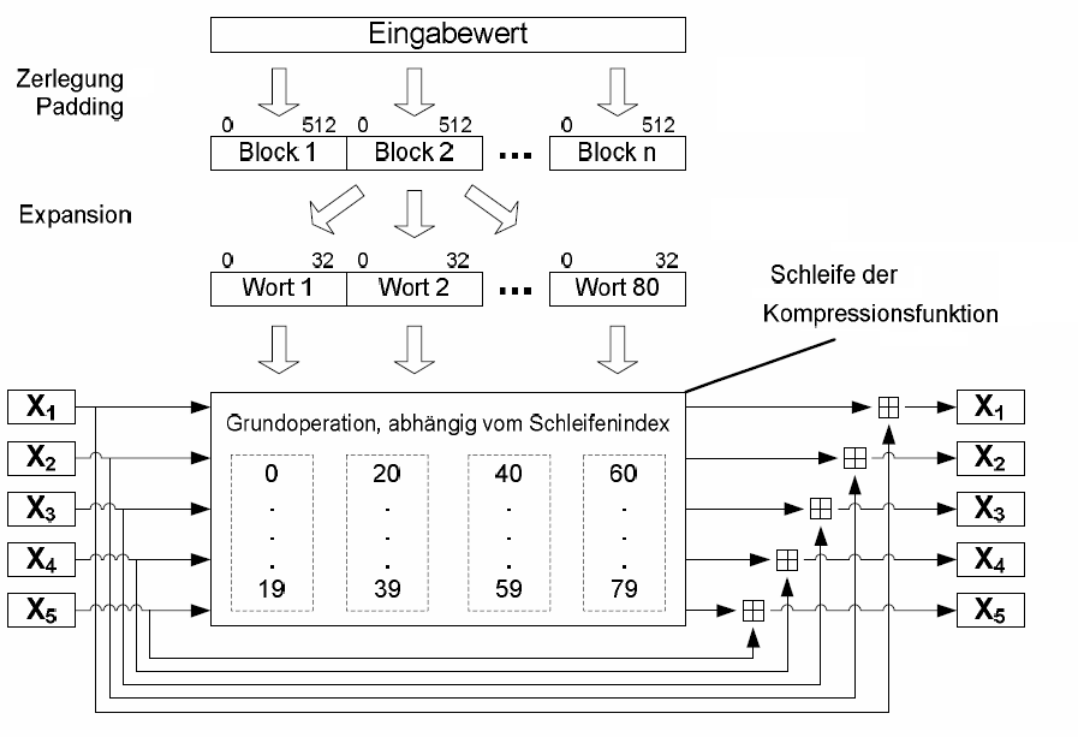

Abbildung 3.2.1: Die schematische Darstellung der SHA1 Hashfunktion. [21, 25].

durch die Divisions-Rest-Methode in $\{-1, +1\}$ berechnet werden. Den Pseudocode der SHA1 ist in [34] zu finden. Der Java-Code wird in der Literatur [21] aufgelistet.

3.2.3 Implementierung des Clarkson-Woodruff Algorithmus

Im Folgenden wird die Implementierung des Clarkson-Woodruff Algorithmus erläutert. Die wichtigsten Anteile des Clarkson-Woodruff Algorithmus sind die Aktualisierung im Abschnitt 3.2.1 und die Hashfunktion im Abschnitt 3.2.2. Der Pseudocode des Algorithmus zeigt das folgende Fenster:

```
Zweck: Berechnung der Lösung durch den Algorithmus
        von Clarkson-Woodruff

Initialisiere d*m Matrix A' := 0;
Initialisiere d Elementen Vektor b' := 0;

repeat
  lese nächste Tupel (i,j,v,A) oder (i,v,b);
  for l=1 to d
  begin
    berechne das Element sil der Sign-Matrix S
          mit Index (i,l) durch Hashfunktion;
    if (ist Tupel (i,j,v,A)) then
      aktualisiere das Element alj der Matrix A'
        mit Index (l,j) durch alj := alj + sil*v;
    else
      aktualisiere das Element bl der Matrix b'
        mit Index l durch bl := bl + sil*v;
    end
  end
until DatenStrom beendet;

Bestimme x durch klassische MKQ

Ergebnis: x
```

3.3 Diskussion

Der Pseudocode im Abschnitt 3.2.3 hat gezeigt, dass $O(md)$ Speicherplätze für das Clarkson-Woodruff Algorithmus benötigt werden. Wenn ein Tupel gelesen wird, sind d Elemente aktualisiert. Der Berechnungsaufwand ist für jedes Tupel $O(d)$. Man weiß:

$$d = O\left(k \log\left(1/\delta\right)/\epsilon\right), \tag{3.3.1}$$

wobei $\epsilon \in (0,1)$, $\delta \in (0,1)$ und k den Rang der Matrix $A \in \mathbb{R}^{n \times m}$ ist. Also ist $k \leq m$. Dann erhält man

$$\begin{aligned} d &= O(k\log(1/\delta)/\epsilon) \\ &= O(k) \\ &= O(m). \end{aligned} \tag{3.3.2}$$

Der Speicheraufwand des Clarkson-Woodruff Algorithmus hat die gleiche Einschränkung $O(m^2)$ wie die beiden Algorithmen der Time Series Modelle. Durch den Pseudocode ist ersichtlich, dass die i-te Zeile der Sign-Matrix $S \in \mathbb{R}^{n \times d}$ nur bei der Aktualisierung i-te Zeile der Matrix $A \in \mathbb{R}^{n \times m}$ und i-te Element des Vektors $b \in \mathbb{R}^n$ verwendet wird. Wenn die Daten zeilenweise verarbeitet werden, wird die Hashfunktion für das Clarkson-Woodruff Algorithmus nicht mehr benötigt. Der Berechnungsaufwand der Aktualisierung für eine Zeile ist $O(m^2)$. Er ist gleich so groß wie der Berechnungsaufwand der beide Algorithmen der Time Series Modelle. Aber die exakte Lösung wird durch die beide Algorithmen der Time Series Modelle ermittelt, während des Clarkson-Woodruff Algorithmus nur eine Näherung berechnen kann, trotzdem ist die Abweichung klein.
Im Abschnitt 3.1.1 wird vorgestellt, dass beim Algorithmus von Zhou nicht nur die Kondition des linearen Gleichungssystems sich verschlechtert. Ein weiteres Risiko ist, dass die Diagonalelemente der Matrix M überlaufen werden könnten.

Kapitel 4

Experimente

In diesem Kapitel werden die klassische MKQ und die Datenstromalgorithmen der MKQ, die in Kapiteln 2 und 3 vorgestellt werden, bei der Verarbeitung von Testdaten eingesetzt, um die Genauigkeit der Theorien und die Eigenschaften der Datenstromalgorithmen zu überprüfen.

4.1 Die experimentelle Umgebung

Alle Programme wurden in der Programmiersprache C ohne Verwendung zusätzlicher Bibliotheken verfasst. Sie wurden mit dem Compiler (Dev-C++ Version 4.9.9.2) kompiliert. Der verwendeten Rechner ist ein Windows 7 PC mit dem Prozessor (Pentium Dual-Core E5400).

4.2 Testdaten Generator

Für das Experiment werden Testdaten benötigt. Wenn die Testdaten nicht vorhanden sind, müssen zuerst Testdaten erzeugt werden. Möglicherweise können Testdaten durch bekannte physikalischen Größen generiert werden. So werden die Testdaten im Folgenden mithilfe der physikalischen Gleichung "Van-der-Vaals-Gleichung":

$$p = \frac{RT}{V - c_2} - \frac{c_1}{V^2}, \tag{4.2.1}$$

wobei p: Druck, T : Temperatur, V: Volumen und $R = 8,3145 \frac{\mathrm{J}}{\mathrm{mol \cdot K}}$: Allgemeine Gaskonstante [15], generiert. Man nimmt an, dass 10% relative Fehler bei der Messung der physikalischen Größen P und V auftreten. Bei $T = 5\mathrm{K}$ werden die Werte von p

für $0,35\,\mathrm{dm}^3 \leq V \leq 1,35\,\mathrm{dm}^3$ berechnet durch:

$$p = \left(\frac{RT}{V - c_2} - \frac{c_1}{V^2}\right) \cdot \left(1 + \frac{Z}{10}\right) \quad , \tag{4.2.2}$$

wobei Z eine Zufallsvariable ist, die über dem Intervall $[-1, +1]$ gleich verteilt ist. D. h.

$$Z \sim U(-1, +1).$$

Mittels der berechneten Daten von p und V können die physikalischen Größen c_1 und c_2 bestimmt werden. Durch eine Umformulierung der Gleichung 4.2.1 erhält man

$$\frac{c_1}{V} - pc_2 - \frac{c_1 c_2}{V^2} = RT - pV.$$

Danach wird die lineare Gleichungen für die Unbekannten $(c_1, c_2, c_1 c_2)$ beschrieben durch:

$$\frac{c_1}{V_1} - p_1 c_2 - \frac{c_1 c_2}{V_1^2} = \underbrace{RT - p_1 V_1}_{b_1}$$

$$\vdots$$

$$\frac{c_1}{V_n} - p_n c_2 - \frac{c_1 c_2}{V_n^2} = \underbrace{RT - p_n V_n}_{b_n}.$$

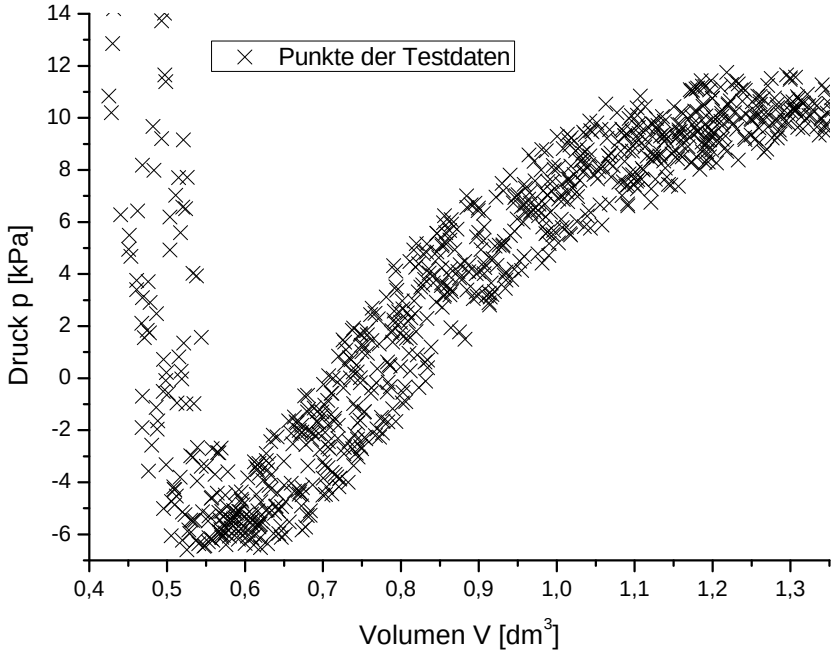

Abbildung 4.2.1: 1000 generierte Testdaten durch das C-Programm (Testdaten Generator)

Das lineare Gleichungssystem wird in Matrixschreibweise formuliert:

$$\underbrace{\begin{bmatrix} \frac{1}{V_1} & -p_1 & -\frac{1}{V_1^2} \\ \vdots & \vdots & \vdots \\ \frac{1}{V_n} & -p_n & -\frac{1}{V_n^2} \end{bmatrix}}_{A\in\mathbb{R}^{n\times 3}} \underbrace{\begin{bmatrix} c_1 \\ c_2 \\ c_1c_2 \end{bmatrix}}_{x\in\mathbb{R}^3} = \underbrace{\begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}}_{b\in\mathbb{R}^n}. \tag{4.2.3}$$

Für Benzol (C_6H_6) $c_1 = 52,74\frac{\mathrm{kPa\cdot dm^6}}{\mathrm{mol^2}}$ $c_2 = 0,3043\frac{\mathrm{dm^3}}{\mathrm{mol}}$ [41] werden z. B. 1000 Daten von (p, V) nach der Formel 4.2.2 berechnet. Die generierten Daten (p, V) werden in Abbildung 4.2.1 veranschaulicht. Schließlich werden die Testdaten (die Matrix A und das Vektor b) durch die Formel 4.2.3 und die Daten (p, V) ermittelt.

Die Matrix A hat jetzt nur drei Spalten. Für weitere Spalten kann man z. B. einfach Zufallszahlen in $[-10, +10]$ verwenden. D. h.

$$a_{i,j} \in [-10, +10] \ \forall j \geq 4.$$

Die erwartete Lösung x ist

$$\begin{aligned} x &= (x_1, x_2, x_3, x_4, x_5, \cdots) \\ &= (c_1, c_2, c_1 c_2, 0, 0, \cdots) \\ & \\ & (x_j = 0 \ \ \forall j \geq 4), \end{aligned}$$

wenn man die Lineare Gleichung $Ax = b$ löst.

4.3 Prüfung der zwei Algorithmen der Time Series Modell

4.3.1 Die Vergleichung der Residuen $\|Ax - b\|$

Wenn die beiden Algorithmen der Time Series Modell zur Verarbeitung der Testdaten eingesetzt werden, erhält man genau die Ergebnisse, die durch die klassische MKQ ermittelt werden. Die Lösungen der beiden Algorithmen unterscheiden sich nur

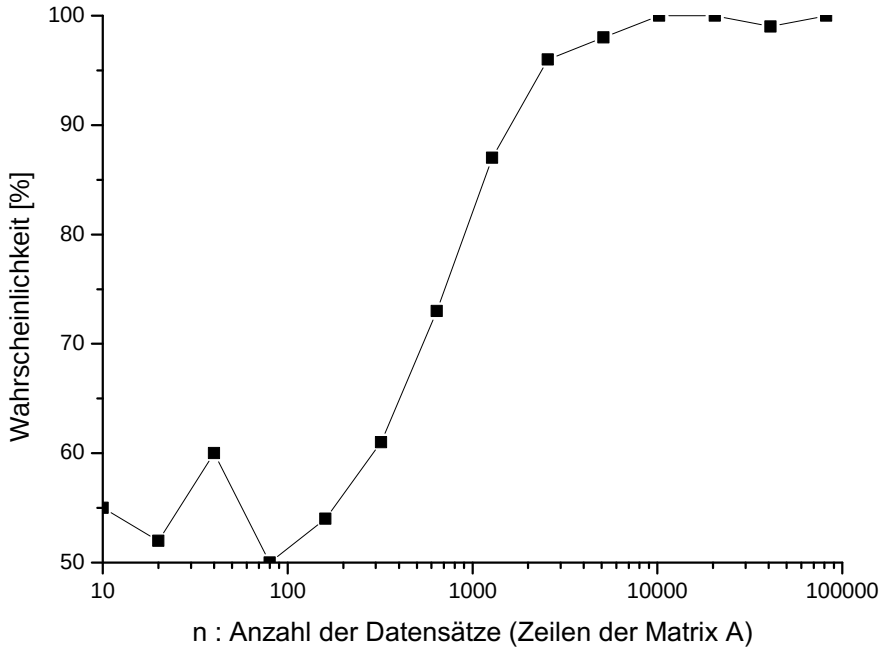

Abbildung 4.3.1: Die Wahrscheinlichkeit, dass die Ungleichung $\left\|Ax_{\text{opt}}^{\text{QR}} - b\right\| < \left\|Ax_{\text{opt}}^{\text{Zhou}} - b\right\|$ erfüllt, gegen die Anzahl der Datensätze aufgetragen, wobei $x_{\text{opt}}^{\text{QR}}$ und $x_{\text{opt}}^{\text{Zhou}}$ die Lösung des QR-Datenstromalgorithmus und die Lösung des Algorithmus von Zhou sind.

durch die letzte Ziffer, die durch (Ab-)Aufrunden erhalten wurde. Theoretisch soll der relative Fehler der Ergebnisse, die durch das Algorithmus von Zhou berechnet, größer sein. Mit Residuen wird die Güte einer Regressionsfunktion beurteilt. Die

Qualität der beiden Algorithmen können durch ihre Residuen beurteilt werden. In Abbildung 4.3.1 werden die Wahrscheinlichkeit, dass die Ungleichung $\left\|Ax_{\text{opt}}^{\text{QR}} - b\right\| < \left\|Ax_{\text{opt}}^{\text{Zhou}} - b\right\|$ erfüllt, gegen die Anzahl der Datensätze aufgetragen, wobei $x_{\text{opt}}^{\text{QR}}$ und $x_{\text{opt}}^{\text{Zhou}}$ die Lösung des QR-Datenstromalgorithmus und die Lösung des Algorithmus von Zhou sind. Die Abbildung 4.3.1 zeigt, dass die Wahrscheinlichkeit (der QR-Datenstromalgorithmus liefert bessere Ergebnisse als der Algorithmus von Zhou) mit zunehmenden Datensätze ansteigt.

4.3.2 Die Dimension Abhängigkeit der Residuen $\|Ax - b\|$

Mit der Abbildung 4.3.1 vermutet man, dass je größer die Datenmenge, desto besse-

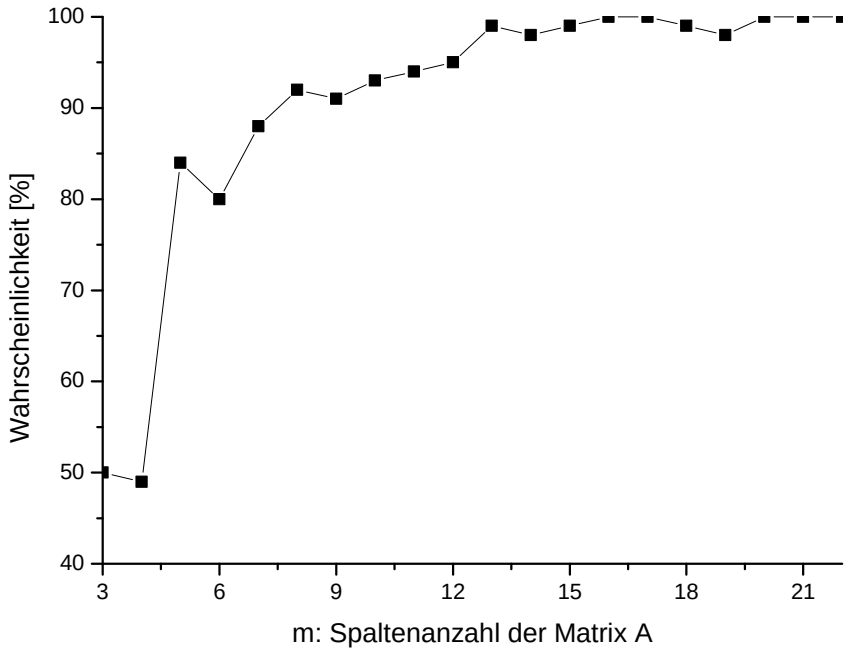

Abbildung 4.3.2: Die Wahrscheinlichkeit, dass die Ungleichung $\left\|Ax_{\text{opt}}^{\text{QR}} - b\right\| < \left\|Ax_{\text{opt}}^{\text{Zhou}} - b\right\|$ erfüllt, gegen die Anzahl der Spaltenanzahl der Matrix A aufgetragen, wobei $x_{\text{opt}}^{\text{QR}}$ und $x_{\text{opt}}^{\text{Zhou}}$ die Lösung des QR-Datenstromalgorithmus und die Lösung des Algorithmus von Zhou sind.

re Ergebnisse der QR-Datenstromalgorithmus als der Algorithmus von Zhou liefern kann. Um diese zu überprüfen, werden die Wahrscheinlichkeit, dass die Ungleichung $\left\|Ax_{\text{opt}}^{\text{QR}} - b\right\| < \left\|Ax_{\text{opt}}^{\text{Zhou}} - b\right\|$ erfüllt, nochmal gegen die Spaltenanzahl der Matrix A aufgetragen (siehe Abbildung 4.3.2). Durch die zwei Experimente wird es gezeigt, dass der QR-Datenstromalgorithmus als Stromalgorithmus besser als der Algorithmus von Zhou ist.

4.4 Experimente des Clarkson-Woodruff Algorithmus

4.4.1 Vergleich der Hashfunktionen

Die Qualität des Clarkson-Woodruff Algorithmus ist von den gewählten Hashfunktion abhängig. In Abschnitt 3.2.2 werden fünf Hashfunktionen vorgestellt. Die Auswahl der Hashfunktion spielt für das Clarkson-Woodruff Algorithmus eine große Rolle. In diesem Abschnitt werden einige Experimenten dargestellt, um zu ermitteln, welche Hashfunktionen relativ gut für das Clarkson-Woodruff Algorithmus geeignet sind.

Wegen der Laufzeit werden nur kleine Instanzen der Matrix A und des Vektors b für die Experimente verwendet, die durch einen Testdaten Generator (siehe Abschnitt 4.2) erzeugt sind. Zu Beginn wird die exakte Lösung x_{opt} mit einer Instanz (A und b) durch das klassische Verfahren oder die Algorithmen der Time Series Modell berechnet. Danach wird die Lösung x_{opt}^* mit der gleichen Instanz durch das Clarkson-Woodruff Algorithmus ermittelt. Dann wird überprüft, ob $\left\|Ax_{\text{opt}}^* - b\right\| \leq (1+\epsilon)\left\|Ax_{\text{opt}} - b\right\|$ ist. Nach mehrmaliger Berechnungen (Z. B. 100 mal) erhält man die Wahrscheinlichkeit w der Ungleichung $\left\|Ax_{\text{opt}}^* - b\right\| \leq (1+\epsilon)\left\|Ax_{\text{opt}} - b\right\|$. In der Abbildung 4.4.1 wird die Wahrscheinlichkeit w gegen die Anzahl der Zeilen der Instanzen (A und b) eingetragen. Jeder Punkt in der Abbildung 4.4.1 wird 30 mal berechnet. In der Abbildung wird dargestellt, dass die Zufallsmatrix der Hashfunktion und SHA1 Hashfunktion viel besser als die Divisions-Rest-Methode, Multiplikative Methode und Gemische Hashfunktion sind. Bei diesen Experimenten sind die Parameter $\epsilon = 0.1$, $\delta = 0.1$ und $c = 1$. Man benötigt hier den Parameter c, um d (die

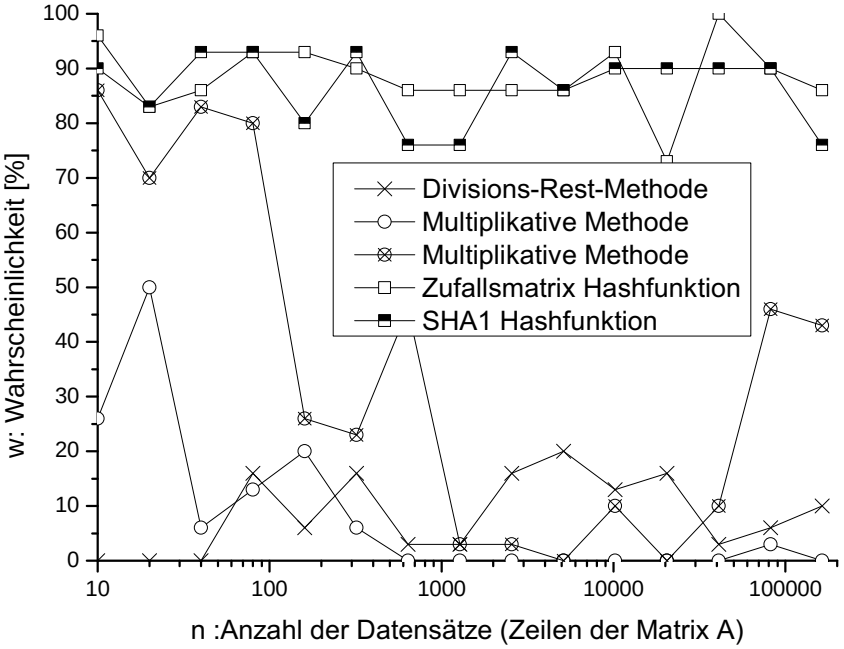

Abbildung 4.4.1: Die Qualität des Clarkson-Woodruff Algorithmus mit entsprechenden Hashfunktionen gegen die Anzahl der Datensätze (die Anzahl der Zeilen der Matrix $A \in \mathbb{R}^{n\times m}$, wobei $m = 3$). Die Parameter: $\epsilon = 0.1$, $\delta = 0.1$ und $c = 1$.

Anzahl der Spalten der Sign-Matrix $S \in \mathbb{R}^{n\times d}$) zu berechnen. D. h.

$$\begin{aligned} d &= O(k\log(1/\delta)/\epsilon) \\ &= c \cdot m \cdot \log(1/\delta)/\epsilon \\ &= 1 \cdot 3 \cdot \log(1/0.1)/0.1 \\ &= 30, \end{aligned} \tag{4.4.1}$$

wobei m die Anzahl der Spalten der Matrix $A \in \mathbb{R}^{n\times m}$ ist und k den Rang der Matrix $A \in \mathbb{R}^{n\times m}$ ist. Die Qualität des Clarkson-Woodruff Algorithmus soll auch von dem Parameter c abhängig sein. Um diese zu überprüfen, wird das zweite Experiment durchgeführt (siehe Abbildung 4.4.2). In der Abbildung4.4.2 wird dargestellt, dass die Wahrscheinlichkeit w bei SHA1 und Zufallsmatrix Hashfunktion mit zunehmender c schnell gegen 100% verlaufen. Jedoch sind die Divisions-Rest-Methode,

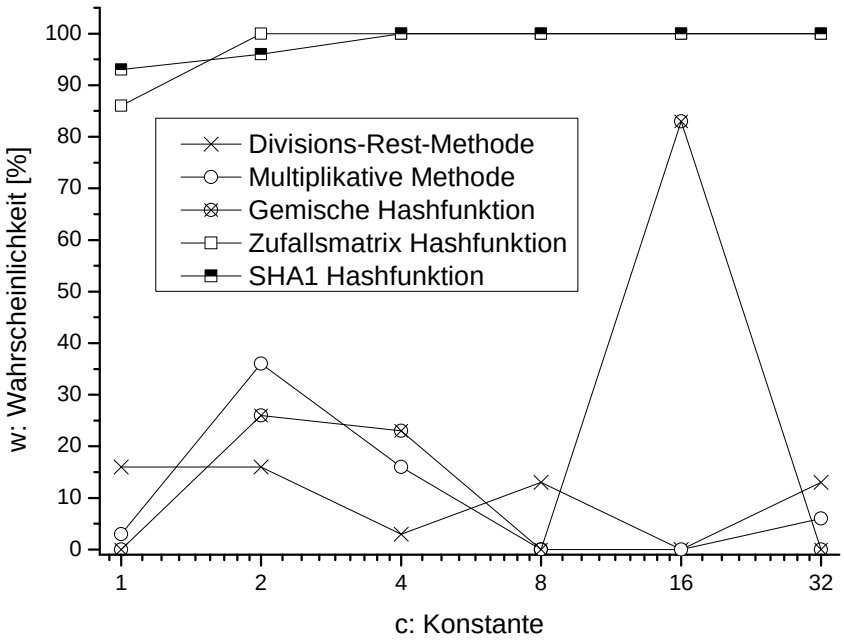

Abbildung 4.4.2: Die Qualität des Clarkson-Woodruff Algorithmus mit entsprechenden Hashfunktionen gegen den Parameter c. Hier ist die Anzahl der Datensätze $n = 10^5$ (die Anzahl der Zeilen der Matrix $A \in \mathbb{R}^{n \times m}$, wobei $m = 3$). Die Parameter: $\epsilon = 0.1$ und $\delta = 0.1$.

Multiplikative Methode und gemischte Hashfunktion nicht verbessert wurden. Die beiden Experimente zeigen, dass die Qualität des Clarkson-Woodruff Algorithmus bei SHA1 und die Zufallsmatrix Hashfunktion gut sind.

Gute Algorithmen sollten nicht nur gute Ergebnisse liefern, sondern auch die Ergebnisse in kurzer Zeit berechnen können. Als weiterer Schritt der Experimentsdurchführung wird die Laufzeit der SHA1 und der Zufallsmatrix Hashfunktion durch den Computer simuliert (siehe Abbildung 4.4.3). Das dritte Experiment zeigt, dass die Zufallsmatrix Hashfunktion ca. 200 mal so schnell wie SHA1 ist.

Für den Vergleich zwischen SHA1 und Zufallsmatrix Hashfunktion soll der Speicherbedarf nicht untersucht werden, da der Speicherbedarf $O(1)$ ist, der weder von n (Anzahl der Matrixzeilen) noch von m (Anzahl der Matrixspalten) abhängig ist. Mittels der obigen drei Experimente kann die Entscheidung getroffen werden, dass

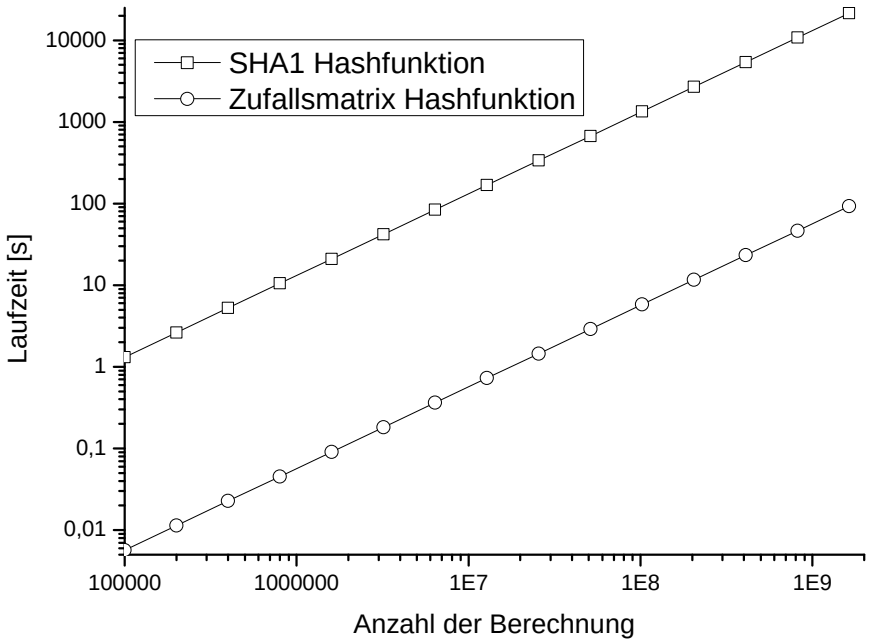

Abbildung 4.4.3: Die Berechnungszeit der Hashfunktionen gegen die Anzahl der Berechnung.

die Zufallsmatrix Hashfunktion für das Clarkson-Woodruff Algorithmus ausgewählt werden soll.

4.4.2 Prüfungen des Clarkson-Woodruff Algorithmus

Das Clarkson-Woodruff Algorithmus kann eine Lösung x^*_{opt} liefern, die sehr gut ist und nur geringfügig von einer exakten Lösung x_{opt} abweicht. In diesem Abschnitt werden einige Experimente dargestellt, um das Clarkson-Woodruff Algorithmus auf seine Qualität zu überprüfen.

4.4.2.1 Die Dimension Abhängigkeit

In diesem Abschnitt wird überprüft, ob die Qualität des Clarkson-Woodruff Algorithmus von der Dimension der Daten abhängig ist. In Abbildung 4.4.4 und 4.4.5 werden

die Wahrscheinlichkeit, dass die Ungleichung $\left\|Ax^*_{\text{opt}} - b\right\| \leq (1+\epsilon)\left\|Ax^{\text{QR}}_{\text{opt}} - b\right\|$ erfüllt, gegen die Anzahl der Datensätze aufgetragen, wobei x^*_{opt} und $x^{\text{QR}}_{\text{opt}}$ die Lösung des Clarkson-Woodruff Algorithmus und die Lösung des QR-Datenstromalgorithmus sind. Die Testdaten werden durch den Generator (siehe Abschnitt 4.2) generiert.

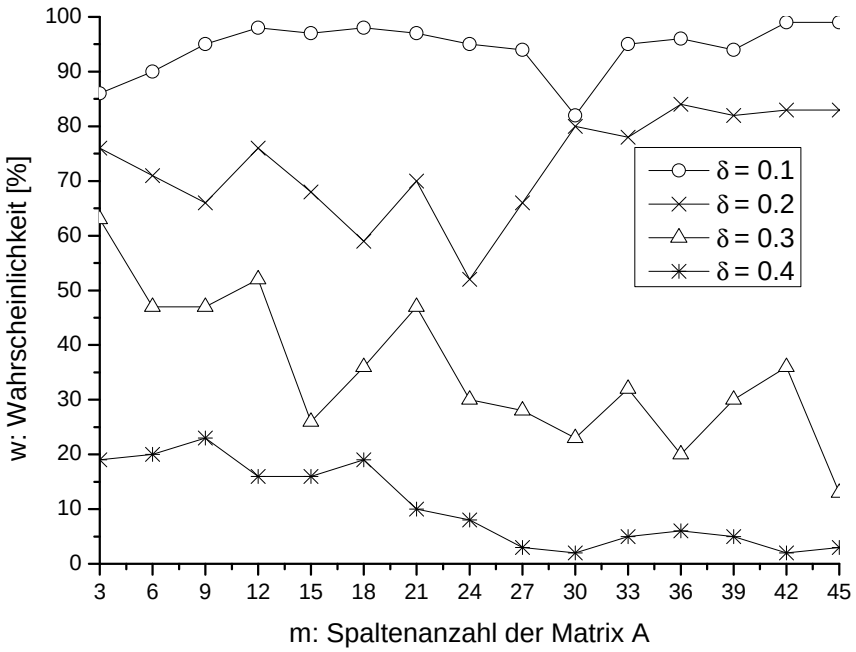

Abbildung 4.4.4: Die Qualität des Clarkson-Woodruff Algorithmus gegen die Spaltanzahl der Matrix A. Die Testdaten werden durch den Generator generiert, wobei die Anzahl der Datensätze $n = 10^5$ ist (die Anzahl der Zeilen der Matrix A). Die Parameter: $\epsilon = 0.1$ und $c = 1$.

Wegen der Laufzeit werden nur 10^5 Datensätze verwendet. Jeder Punkt wird 100 mal berechnet. Nach der Theorie des Clarkson-Woodruff sollen die Ungleichung 3.2.22 mit einer Wahrscheinlichkeit w von mindestens $1 - \delta$ erfüllt sein. Wenn der Parameter $c = 1$ eingestellt wird, ist die Bedingung nicht erfüllt, und nimmt die Qualität des Clarkson-Woodruff Algorithmus für $\delta \geq 0.3$ mit zunehmenden Dimensionen ab (siehe Abbildung 4.4.4). Ist $c = 2$, nimmt die die Qualität des Clarkson-Woodruff Algorithmus für $\delta \leq 0.4$ mit zunehmenden Dimensionen nicht mehr ab. Für die Aufwertung

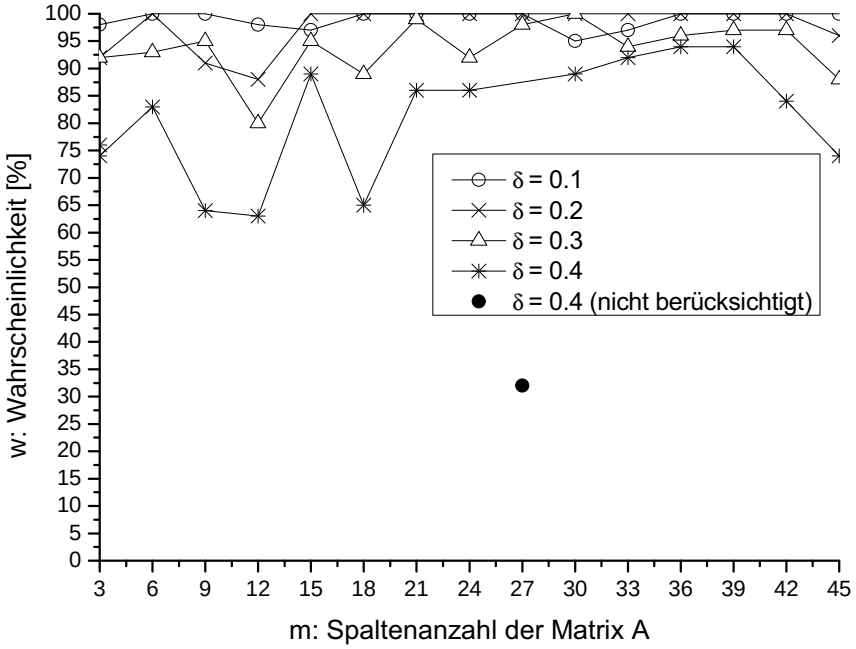

Abbildung 4.4.5: Die Qualität des Clarkson-Woodruff Algorithmus gegen die Spaltanzahl der Matrix A. Die Testdaten werden durch den Generator generiert, wobei die Anzahl der Datensätze $n = 10^5$ ist (die Anzahl der Zeilen der Matrix A). Die Parameter: $\epsilon = 0.1$ und $c = 1$.

wird einer Punkt in Abbildung 4.4.5 nicht berücksichtigt, weil er etwas weiter als andere abweicht. Dieser kann verursacht durch systematische Fehler und statistische Fehler sein.

Die gleiche Untersuchungen auf eine andere Testdaten durchgeführt werden, werden die gleiche Entscheidungen getroffen (siehe Abbildung 4.4.6 und 4.4.7). Die Testdaten, die 386 Spalten und 53500 Datensätze umfasst, werden hier aus dem Internet [30] heruntergeladen. Für das Experiment werden nur die ersten 1000 Datensätze verwendet. Es werden zufällig m Spalten aus die Daten als Matrix A ausgewählt. Außerdem wird angenommen, dass die letzte Spalte der Daten der Vektor b ist. Das bedeutet, dass immer die Spalte 386 als Vektor b verwendet wird. Somit erhält man die lineare Gleichung

$$Ax = b. \tag{4.4.2}$$

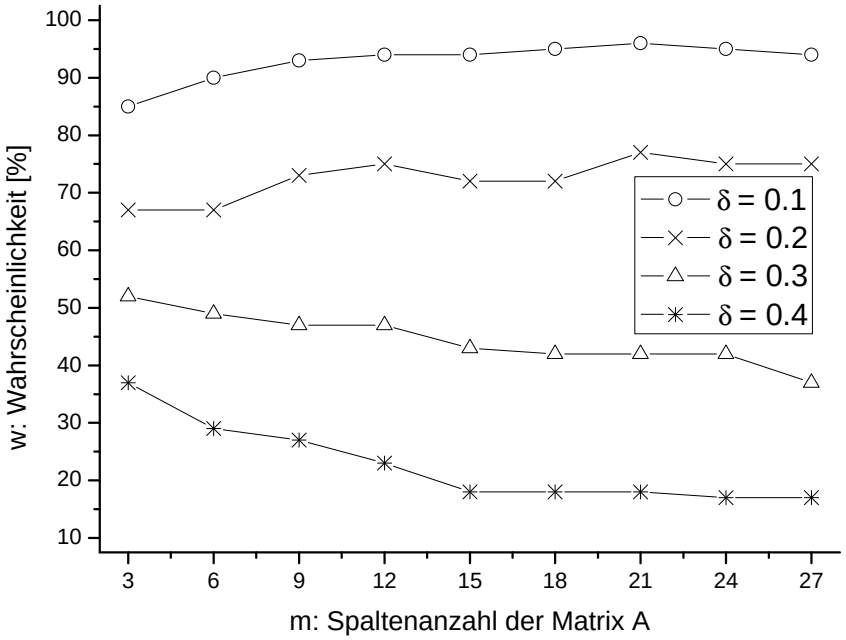

Abbildung 4.4.6: Die Qualität des Clarkson-Woodruff Algorithmus gegen die Spaltanzahl der Matrix A. Die Testdaten werden aus dem Internet [30] heruntergeladen, wobei nur die ersten 1000 Datensätze verwendet (die Anzahl der Zeilen der Matrix A ist 1000). Die Parameter: $\epsilon = 0.1$ und $c = 1$.

Durch das Experiment macht deutlich, dass der Parameter $c \geq 2$ sein muss und die Qualität des Clarkson-Woodruff Algorithmus von der Dimensionen fast unabhängig ist. Es kann sein, dass die Qualität des Clarkson-Woodruff Algorithmus mit zunehmenden Dimensionen leicht ansteigt, wenn $c \geq 2$ ist.

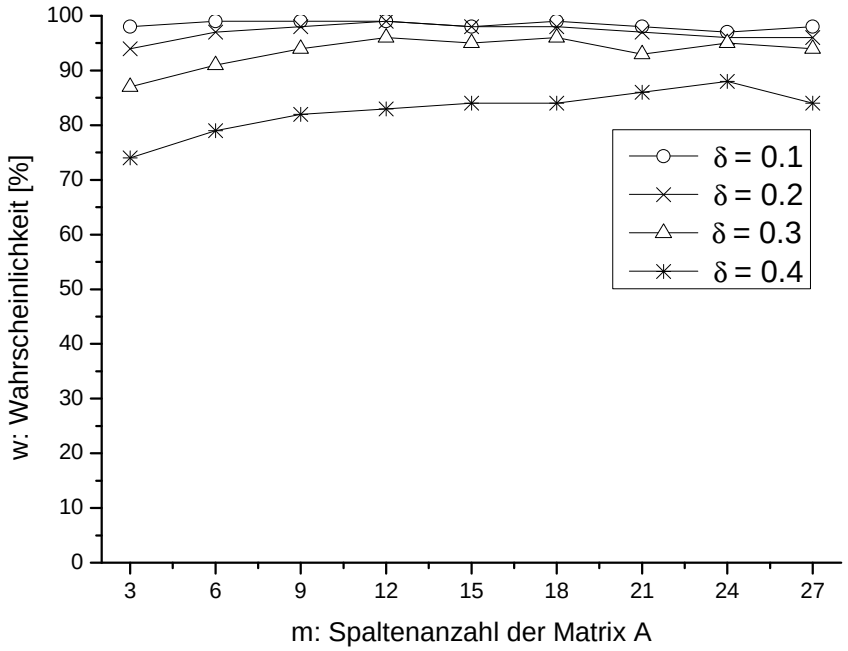

Abbildung 4.4.7: Die Qualität des Clarkson-Woodruff Algorithmus gegen die Spaltanzahl der Matrix A. Die Testdaten werden aus dem Internet [30] heruntergeladen, wobei nur die ersten 1000 Datensätze verwendet (die Anzahl der Zeilen der Matrix A ist 1000). Die Parameter: $\epsilon = 0.1$ und $c = 2$.

4.4.2.2 Das Residuum

Mit Residuen wird die Güte einer Regressionsfunktion beurteilt. Man bewertet die Güte eines Datenstromalgorithmus für Regression, indem man die gefundene Lösung mit der klassischen optimalen Lösung vergleicht und daraus das kompetitive Verhältnis $\frac{\|Ax^*_{\text{opt}}-b\|}{\|Ax_{\text{opt}}-b\|}$ bestimmt, wobei x_{opt} und x^*_{opt} die klassische optimale Lösung und die Lösung des Datenstromalgorithmus sind. Die zwei Datenstromalgorithmen der Time Series Modell ermitteln genau die Ergebnisse, die durch die klassische Regression ermittelt werden (siehe Abschnitt 4.3). D. h. ist das Verhältnis $\frac{\|Ax^*_{\text{opt}}-b\|}{\|Ax_{\text{opt}}-b\|} = 1$ für die zwei Datenstromalgorithmen der Time Series Modell. Beim Clarkson-Woodruff

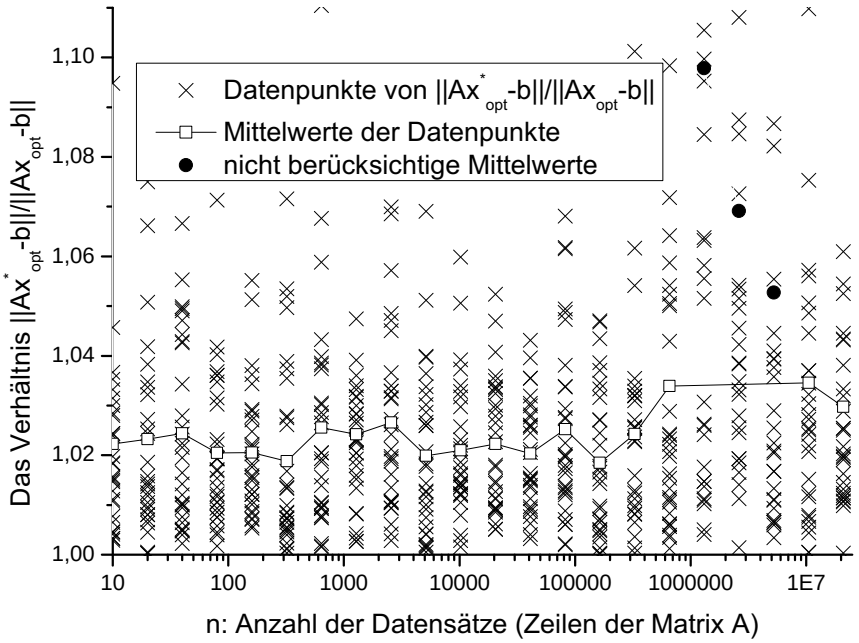

Abbildung 4.4.8: Das Verhältnis $\frac{\left\|Ax^*_{\text{opt}}-b\right\|}{\|Ax_{\text{opt}}-b\|}$ des Clarkson-Woodruff Algorithmus gegen die Anzahl der Datensätze (die Anzahl der Zeilen der Matrix $A \in \mathbb{R}^{\text{n}\times\ \text{m}}$, wobei $m = 3$ verwendet wird). Die Parameter: $\epsilon = 0.1$, $\delta = 0.1$ und $c = 2$.

Algorithmus ist die Ungleichung

$$\left\|Ax^*_{\text{opt}} - b\right\| \leq (1+\epsilon)\left\|Ax_{\text{opt}} - b\right\|$$

mit einer Wahrscheinlichkeit von mindestens $1-\delta$ erfüllt. Das Verhältnis $\frac{\left\|Ax^*_{\text{opt}}-b\right\|}{\|Ax_{\text{opt}}-b\|} \leq (1+\epsilon)$ ist also auch mit einer Wahrscheinlichkeit von mindestens $1-\delta$ erfüllt. In Abbildung 4.4.8 und Abbildung 4.4.9 werden die Ergebnisse der Experimente für das Verhältnis $\frac{\left\|Ax^*_{\text{opt}}-b\right\|}{\|Ax_{\text{opt}}-b\|}$ des Clarkson-Woodruff Algorithmus aufgezeigt, wobei die verwendete Daten durch den Generator generiert werden. Durch die beiden Abbildung wird gezeigt, dass der Mittelwert der Verhältnisse $\frac{\left\|Ax^*_{\text{opt}}-b\right\|}{\|Ax_{\text{opt}}-b\|}$ des Clarkson-Woodruff Algorithmus fast von der Zeilen- und Spaltenanzahl der Matrix $A \in \mathbb{R}^{n\times m}$ unabhängig ist. In Abbildung 4.4.8 werden drei Werte für die Aufwertung nicht berücksichtigt,

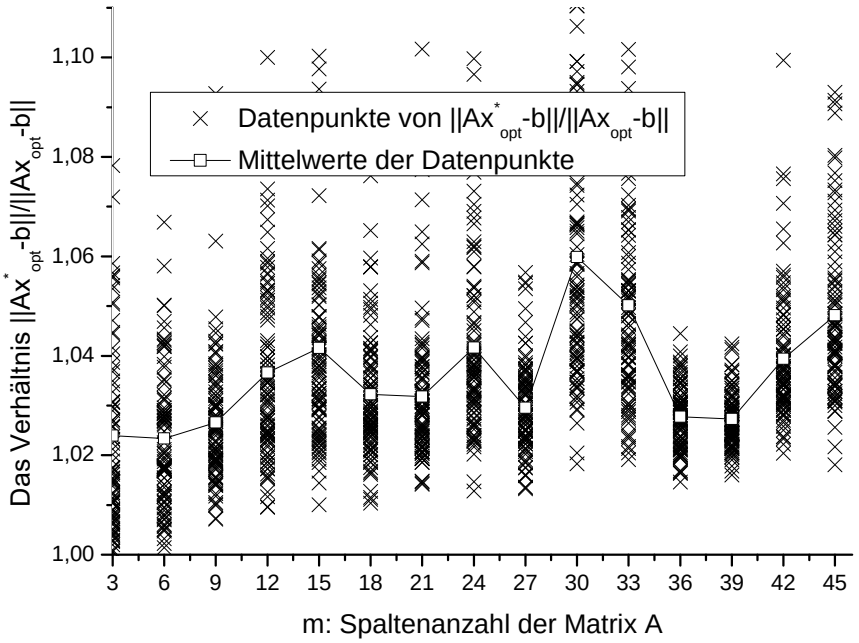

Abbildung 4.4.9: Das Verhältnis $\frac{\|Ax^*_{\text{opt}}-b\|}{\|Ax_{\text{opt}}-b\|}$ des Clarkson-Woodruff Algorithmus gegen die Spaltanzahl der Matrix A. Hier ist die Anzahl der Datensätze $n = 1000$ (die Anzahl der Zeilen der Matrix A). Die Parameter: $\epsilon = 0.1$, $\delta = 0.1$ und $c = 2$.

weil sie etwas weiter als andere abweichen. Dieser kann verursacht durch systematische Fehler und statistische Fehler sein.

4.5 Laufzeit der Algorithmen für Regression

Theoretisch ist die Laufzeit der klassischen Regression gleich $O(nm^2)$ (siehe Abschnitt 2.6). Die Aktualisierungszeiten der drei Datenstromalgorithmen pro Element sind $O(m)$ (siehe Abschnitt 3.1.1, 3.1.2 und 3.3). Es gibt nm Elemente in der Matrix $A \in \mathbb{R}^{n\times m}$. Insgesamt sind die Aktualisierungszeiten der drei Datenstromalgorithmen gleich $O(nm^2)$. Sie sind genau so groß wie die Laufzeit der klassischen Regression. Jedoch ist die O-Notation nur eine Abschätzung der Laufzeit. Konstante Faktoren und Konstante Summanden werden vernachlässigt. Das Verhältnis der konstante Fak-

toren der Laufzeit der vier Algorithmen können durch Computer simuliert werden. Die Ergebnisse der Simulation sind in Abbildung 4.5.1 dargestellt. Das Verhältnis

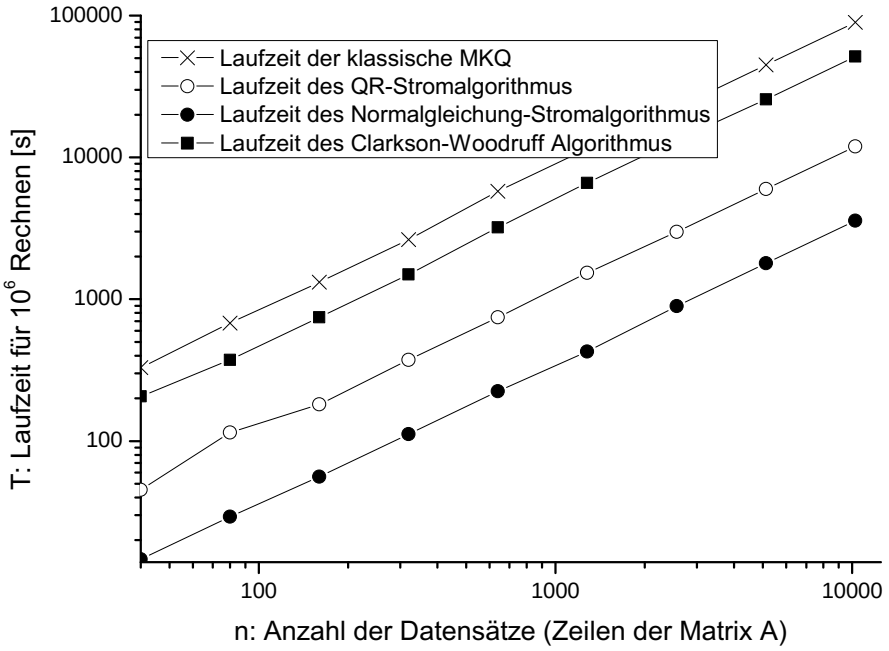

Abbildung 4.5.1: Die Laufzeit der vier Algorithmen gegen die Spaltanzahl der Matrix A. Hier ist die Anzahl der Datensätze $n = 1000$ (die Anzahl der Zeilen der Matrix A). Die Parameter des Clarkson-Woodruff Algorithmus sind $\epsilon = 0.1$, $\delta = 0.1$ und $c = 2$.

der konstante Faktoren der Laufzeit der vier Algorithmen ist:

$$T_{\text{Zhou}}/T_{\text{QR}}/T_{\text{Clarkson-Woodruff}}/T_{\text{klassisch}} \approx 1 : 3 : 14 : 25.$$

Die Laufzeit des Clarkson-Woodruff Algorithmus ist zusätzlich von den Parametern ϵ, δ und c abhängig. Bei den großen ϵ, δ und kleinen c -Parametern ist die Laufzeit kleiner.

Kapitel 5

Zusammenfassung und Ausblick

5.1 Zusammenfassung der Ergebnisse

Die vorliegende Diplomarbeit behandelt die Implementierung und Untersuchung der Datenstromalgorithmen für Regression.

Die zwei Algorithmen im Time Series Modell ermitteln die Ergebnisse, die genau mit den klassischen Lösungen übereinstimmen. Da sich die Kondition des linearen Gleichungssystems durch die Berechnung A^TA verschlechtert, ist der relative Fehler des Algorithmus von Zhou groß. Die Wahrscheinlichkeit, dass die Ungleichung $\left\|Ax_{\text{opt}}^{\text{QR}} - b\right\| < \left\|Ax_{\text{opt}}^{\text{Zhou}} - b\right\|$ erfüllt wird, steigt an, wenn die Datenmenge steigt, wobei $x_{\text{opt}}^{\text{QR}}$ und $x_{\text{opt}}^{\text{Zhou}}$ die Lösung des QR-Datenstromalgorithmus und die Lösung des Algorithmus von Zhou sind. D. h. dass die Wahrscheinlichkeit (der QR-Datenstromalgorithmus liefert bessere Ergebnisse als der Algorithmus von Zhou) mit zunehmenden Datensätze ansteigt.

Im Clarkson-Woodruff Algorithmus liegt der Vorteil, dass die Reihenfolge der Aktualisierung keine Rolle spielt. Die Lösung des Clarkson-Woodruff Algorithmus ist ebenfalls gut, dann sie weicht nur geringfügig von einer exakten Lösung (klassische Lösung) ab. Die Qualität des Clarkson-Woodruff Algorithmus ist fast unabhängig von der Datenmenge. Es kann sein, dass sie mit zunehmenden Dimensionen leicht ansteigt

Durch das Experiment der Laufzeituntersuchung wird deutlich, dass das Verhältnis der konstanten Faktoren der Laufzeit der Algorithmen ist:

$$T_{\text{Zhou}}/T_{\text{QR}}/T_{\text{Clarkson-Woodruff}}/T_{\text{klassisch}} \approx 1:3:14:25.$$

5.2 Ausblick

Die in der Diplomarbeit analysierten Algorithmen sind Datenstromalgorithmen für große Datenmengen. Wegen der Laufzeit sind die Datenmengen, die durch Experimente simuliert werden, nicht groß genug, weil die Experimente mittels eines PC durchgeführt werden. In zukünftigen Untersuchungen sollten mehr und größere Datensätze verwendet werden. Noch besser wäre es, wenn man einen Algorithmus im Turnstile Modell findet, der genau die klassische Lösung liefern kann.

Literaturverzeichnis

[1] Fehlerrechnung bei der Auswertung physikalischer Messungen. [Online http://praktikum.physik.uni-dortmund.de/AP-Anleitungen/Fehlerrechnung/Fehlerrechnung_neu.pdf].

[2] Bachelor-Seminar zu aktuellen Themen der Theoretischen Informatik., 2004. [Online http://ls2-www.cs.uni-dortmund.de/lehre/winter200405/dstr/ankuend.html.

[3] P. Drineas at. al. Faster least squares approximation., 2010. [Online http://arxiv.org/PS_cache/arxiv/pdf/0710/0710.1435v4.pdf.

[4] Thomas H. Cormen at. al. *Introduction to Algorithms*. 2001.

[5] B. Burgeth. Vorlesungsskript — Mathematik für Informatiker iii, 2007. [Online http://www.mia.uni-saarland.de/Teaching/MFI0708/mfi_3_skript.pdf.

[6] K. Clarkson and D. Woodruff. Numerical Linear Algebra in the Streaming Model. In *In Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC)*, 2009.

[7] T. Damm. Das lineare Ausgleichsproblem., 2012. [Online http://www.mathematik.uni-kl.de/ damm/PRAMA/ausgleich.pdf; Stand 22. Januar 2012].

[8] J. G. F. Francis. *The QR Transformation: A Unitary Analogue to the LR Transformation-Part 1*. The Computer Journal, 1961.

[9] J. G. F. Francis. *The QR Transformation-Part 2*. The Computer Journal, 1962.

[10] Peter Furlan. Singulärwertzerlegung und Pseudoinverse., 2007. [Online http://www.das-gelbe-rechenbuch.de/download/Swz.pdf; Stand 11. Juli 2007].

[11] T. N. E. Greville. *Some applications of the pseudo inverse of a matrix.* SIAM Rev., No. 2, 1960.

[12] R. Guilbourd. Übersicht über streaming (Datenstrom-) Algorithmen., 2004. [Online http://www.inf.fu-berlin.de/lehre/SS07/AlgorithmenSeminar/4-Streaming%20Algorithmen.pdf.

[13] H.Pruys. *Vorlesungsskript Datenanalyse - Sommersemester.* 2004.

[14] V.N. Kaublanowskaya. *On some algorithms for the solution of the complete eigenvalue problem.* USSR Comp. Math. Phys. 3, 1961.

[15] H. Keiter. *Thermodynamik und Statistik (Vorlesungsskript).* 2005.

[16] Max Köcher. *Lineare Algebra and analytische Geometrie.* Springer-Verlag Berlin, 1997.

[17] Wurzelzieher Mathepedia. Givens-Rotation, 2012. [Online http://www.mathepedia.de/Givens-Rotation.aspx; Stand 22. Januar 2012].

[18] Wurzelzieher Mathepedia. Gram-Schmidtsches Orthogonalisierungsverfahren, 2012. [Online http://www.mathepedia.de/Gram-Schmidtsches_Orthogonalisierungsverfahren.aspx; Stand 22. Januar 2012].

[19] G. Opfer. *Numerische Mathematik für Anfänger. Eine Einführung für Mathematiker, Ingenieure und Informatiker.* vieweg, 2001.

[20] P.Arbenz. *Kurze Einführung in Matlab.* 2007.

[21] D. Pfeiffer. *Kryptografische Hashfunktionen, (Seminar Ausarbeitung).*

[22] R. Rannacher. Einführung in die Numerische Mathematik, 2005. [Online http://numerik.uni-hd.de/ lehre/notes/.

[23] Karl H. Ruhm. Pseudoinverse., 2005. [Online http://www.mmm.ethz.ch/dok01/d0000245.pdf; Stand 30. Juni 2005].

[24] M. Saürhoff. Datenstromalgorithmen, Vorlesungsankündigung., 2004. [Online http://www.tks.informatik.uni-frankfurt.de/teaching/ss11/aktuelle-themen-bachelor.

[25] Klaus Schmeh. *Kryptografie, Verfahren, Protokolle, Infrastrukturen.* 2007.

[26] K. Schmidt and G.Trenkler. *Einführung in die Moderne Matrix-Algebra, Mit Anwendungen in der Statistik.* Springer Berlin Heidelberg, 2006.

[27] Bruce Schneier. *Angewandte Kryptographie, Protokolle, Algorithmen und Sourcecode in C, Addison-Wesley.* 1996.

[28] C. Sohler. Datenstromalgorithmen., 2008. [Online http://web.informatik.uni-bonn.de/ICS/sohler/Webseiten/Teaching/summer08.htm.

[29] U. Trottenberg. Gausssche Ausgleichsrechnung, 2006. [Online http://www.scai.fraunhofer.de/fileadmin/ArbeitsgruppeTrottenberg/SS06_duis/kap6.pdf.

[30] UCI. Relative location of CT slices on axial axis Data set, 2011. [Online http://archive.ics.uci.edu/ml/datasets/Relative+location+of+CT+slices+on+axial+axis; Stand 05. Februar 2012].

[31] R. Walter. *Höhere Mathematik I (Physiker, Elektrotechniker, Ingenieur-Informatiker).* 2002.

[32] Wikipedia. Givens-Rotation — Wikipedia, Die freie Enzyklopädie, 2011. [Online http://de.wikipedia.org/wiki/Givens-Rotation; Stand 22. Januar 2012].

[33] Wikipedia. Multiplikative Methode — Wikipedia, die freie Enzyklopädie, 2011. [Online http://de.wikipedia.org/wiki/Multiplikative_Methode; Stand 22. Januar 2012].

[34] Wikipedia. Secure Hash Algorithm — Wikipedia, Die freie Enzyklopädie, 2011. [Online http://de.wikipedia.org/wiki/Secure_Hash_Algorithm; Stand 22. Januar 2012].

[35] Wikipedia. Cholesky-Zerlegung — Wikipedia, Die freie Enzyklopädie, 2012. [Online http://de.wikipedia.org/wiki/Cholesky-Zerlegung; Stand 22. Januar 2012].

[36] Wikipedia. Methode der kleinsten Quadrate — Wikipedia, Die freie Enzyklopädie, 2012. [Online http://de.wikipedia.org/wiki/Methode_der_kleinsten_Quadrate; Stand 22. Januar 2012].

[37] Wikipedia. Moore-Penrose pseudoinverse — Wikipedia, The Free Encyclopedia, 2012. [Online http://en.wikipedia.org/wiki/Moore%E2%80%93Penrose_pseudoinverse; accessed 22 January 2012].

[38] Wikipedia. Pseudoinverse — Wikipedia, Die freie Enzyklopädie, 2012. [Online http://de.wikipedia.org/wiki/Pseudoinverse; Stand 22. Januar 2012].

[39] Wikipedia. Qr-Zerlegung — Wikipedia, Die freie Enzyklopädie, 2012. [Online http://de.wikipedia.org/wiki/QR-Zerlegung; Stand 22. Januar 2012].

[40] Wikipedia. Singulärwertzerlegung — Wikipedia, Die freie Enzyklopädie, 2012. [Online http://de.wikipedia.org/wiki/Singul%C3%A4rwertzerlegung; Stand 22. Januar 2012].

[41] Wikipedia. Van-der-Waals-Gleichung — Wikipedia, Die freie Enzyklopädie, 2012. [Online http://de.wikipedia.org/wiki/Van-der-Waals-Gleichung; Stand 22. Januar 2012].

[42] Jens-Peter M. Zemke. *Numerische Verfahren (für Studierende des Studienganges Bauingierurwesen und Umwelttechnik).* 2005.

[43] Q. Zhou. *High Performance Data Stream Pattern Discovery Algorithms and Their Applications.* 2008.

Anhang

Notation

Für jede Matrix $A \in \mathbb{R}^{n \times m}$ bezeichnet

- $A_{(i)}$ die i-te Zeile von A als Zeilenvektor;
- $A^{(j)}$ die i-te Spalte von A als Spaltenvektor;
- $a_{i,j}$ oder $A^{(j)}_{(i)}$ die Element mit Index (i, j) von A;
- $\|A\|_F^2 = \sum_{i=1}^{n} \sum_{j=1}^{m} a_{i,j}^2$ das Quadrat der Frobeniusnorm von A;
- $\|A\|_2 = \max_{x \neq 0} \frac{\|Ax\|_2}{\|x\|_2}$ die Spektralnorm von A.

Kurze Beschreibung der Implementierung

Der Algorithmus von Zhou

Der Algorithmus berechnet zuerst die Matrix $M = A^T A$ und den Vektor $d = A^T b$. Danach wird die Matrix M mit dem Verfahren der Cholesky-Zerlegung in ein Produkt $M = GG^T$ zerlegt, wobei G eine untere Dreiecksmatrix ist. Schließlich wird der Lösungsvektor x durch Vorwärtseinsetzen und Rückwärtseinsetzen ermittelt. Der Pseudocode des Algorithmus wird im Abschnitt 3.1.1 aufgelistet. Der C-Code des Algorithmus sieht wie folgt aus:

```
void Time_Series_Modell_normalgleichung(double *A,
        double *B,double *X,int n, int m,int mb)
{
        double Ms2[m][m];
        double Ds2[m][mb];
        updateNull(&Ms2[0][0],m,m);
        updateNull(&Ds2[0][0],m,mb);
        double ai[1][m];
        double ait[m][1];
        double bi[1][mb];
        double MM[m][m];
        double DD[m][mb];
        for (int i=0;i<n;i++)
        {
                for (int j=0;j<m;j++)
                {
                        ai[0][j]=*(A+i*m+j);
                        ait[j][0]=ai[0][j];
                }
                for (int j=0;j<mb;j++)
                        bi[0][j]=*(B+i*mb+j);
                Matrix_Multiplikation(&ait[0][0],&ai[0][0],
                        &MM[0][0],m,1,m);
                Matrix_Multiplikation(&ait[0][0],&bi[0][0],
                        &DD[0][0],m,1,mb);
                MatrixAdd(&MM[0][0],&Ms2[0][0],m,m);
                MatrixAdd(&DD[0][0],&Ds2[0][0],m,mb);
        }
        Choleskyzerlegen(&Ms2[0][0],m);
        double Lt[m][m];
        matrix_transponieren(&Ms2[0][0],&Lt[0][0],m,m);
        double Y[m][mb];
        Vorwaetseinsetzen(&Ms2[0][0],&Ds2[0][0],&Y[0][0],
                m,m,mb);
        Rueckwaetseinsetzen(&Lt[0][0],&Y[0][0],X,m,m,mb);
}
```

Der QR-Datenstromalgorithmus

Die Kernfunktion des QR-Datenstromalgorithmus ist die spezielle QR-Zerlegung. Sie dient dazu, die Matrix $\begin{bmatrix} A_{(i+1)} \\ R_{i1} \end{bmatrix}$ in eine obere Rechtsdreiecksmatrix R_{i+1} umzurechnen (siehe Abschnitt 3.1.2). Durch die spezielle QR-Zerlegung wird die Matrix A schrittweise in obere Rechtsdreiecksmatrix R umgerechnet. Danach wird der Lösungsvektor x durch Rückwärtseinsetzen ermittelt. Der C-Code des Algorithmus sieht wie folgt aus:

```
void Time_Series_Modell_QR(double *A,double *B,
        double *X,int n, int m,int mb)
{
        double Ms[m+1][m];
        double Ds[m+1][mb];
        updateNull(&Ms[0][0],m+1,m);
        updateNull(&Ds[0][0],m+1,mb);
        for (int i=0;i<n;i++)
        {
                verschieben(&Ms[0][0],m+1,m);
                verschieben(&Ds[0][0],m+1,mb);
                for (int j=0;j<m;j++)
                        Ms[0][j]=*(A+i*m+j);
                for (int j=0;j<mb;j++)
                        Ds[0][j]=*(B+i*mb+j);
                QR_zerlegen_Givens_RD_Spezill(&Ms[0][0],
                        m+1,m,&Ds[0][0],mb);
        }
        Rueckwaetseinsetzen(&Ms[0][0],&Ds[0][0],X,m+1,m,mb);
}
```

Der Clarkson-Woodruff Algorithmus

Der wichtigste Bauteil des Clarkson-Woodruff Algorithmus ist die Funktion für die Aktualisierung. Sie dient dazu, die Matrix $A' \in \mathbb{R}^{d \times m}$ und $b' \in \mathbb{R}^{d}$ zu aktualisieren (siehe Abschnitt 3.2.1). Durch den klassischen Algorithmus für Regression (MKQ) wird dann der Lösungsvektor x ermittelt. Der C-Code des Algorithmus sieht wie folgt aus:

```
void Woodruff_zufallsmatrix_hash(int d,struct stu *U,
        double *X,int n,int m,int mb)
{
        double M[d][m];
        double D[d][mb];
        updateNull(&M[0][0],d,m);
        updateNull(&D[0][0],d,mb);
        struct stu u;
        for (int i=0;i<n;i++)
        for( int j=0;j<m+mb;j++)
        {
                u=*(U+i*(m+mb)+j);
                if (u.col>=m)
                        Update_zufallsmatrix_Hash(&D[0][0]
                        d,mb,u.row,u.col % m,u.value);
                else
                        Update_zufallsmatrix_Hash(&M[0][0]
                        d,m,u.row,u.col,u.value);
        }
        MKQ(&M[0][0],&D[0][0],X,d,m,mb);
}
```

In der C-Code wird die Aktualisierungsfunktion mit Zufallsmatrix Hashfunktion verwendet. Die Theorie der Aktualisierung findet man im Abschnitt 3.2.1. Der C-Code der Funktion sieht wie folgt aus:

```
void Update_zufallsmatrix_Hash(double *M, int d,
        int m, int i,int j, double v)
{
        int sil =0;
        for (int l=0;l<d;l++)
        {
                sil = i*d+l;
                sil = zufallsmatrix_Hash(sil);
                *(M+l*m+j) = *(M+l*m+j)+ sil*v;
        }
}
```

Die wichtige Funktionen für die Experimente werden folgt aufgelistet:

- test_Time_Series_gross(): Vergleichung der zwei Algorithmen der Time Series Modell mit zunehmende Datensätze;
- test_Time_Series_dimension_gross(): Vergleichung der zwei Algorithmen Time Series Modell mit zunehmende; Dimensionen;
- test_woodruff_hash(): Überprüfung der Reihung der Hashfunktionen mit zunehmende Datensätze;
- test_Woodruff_hash_cons(): Überprüfung der Reihung der Hashfunktionen mit zunehmendem Parameter c;
- Test_Hash_laufzeit(): Simulieren der Laufzeit für die Hashfunktionen;
- test_Woodruff_dimension_generator(); Dimension Abhängigkeit des des Clarkson-Woodruff Algorithmus mit generierte Daten;
- test_Woodruff_dimension_File(); Dimension Abhängigkeit des des Clarkson-Woodruff Algorithmus mit Testdaten aus Internet;
- test_Woodruff_gut_gross(); Überprüfung des Verhältnis $\frac{\left\|Ax^*_{\text{opt}}-b\right\|}{\|Ax_{\text{opt}}-b\|}$ gegen Anzahl der Datensätze;
- test_Woodruff_dimension_generator_gut(); Überprüfung des Verhältnis $\frac{\left\|Ax^*_{\text{opt}}-b\right\|}{\|Ax_{\text{opt}}-b\|}$ gegen Spaltenanzahl der Matrix A.

Danksagung

Ich möchte mich an dieser Stelle bei Frau M. Schmidt und Herrn Prof. Dr. C. Sohler herzlich bedanken, die direkt zum Gelingen dieser Arbeit beigetragen haben. Mein besonderer Dank geht an Frau M. Schmidt, die mit bei vielen Formulierungen bei der Korrektur der Diplomarbeit sehr hilfreich zur Seite standen.