Effiziente Algorithmen und Komplexität in der robusten Statistik

Dissertation

zur Erlangung des Grades eines

Doktors der Naturwissenschaften

der Universität Dortmund

am Fachbereich Informatik

von

Thorsten Bernholt

Dortmund

2006

Tag der mündlichen Prüfung:	16. Oktober 2006
Dekan:	Prof. Dr. Peter Buchholz
Gutachter:	PD Dr. Thomas Hofmeister, Prof. Dr. Ingo Wegener

Danksagung

Mein Dank geht an Roland Fried für die gute und erfolgreiche Zusammenarbeit und Andreas Christmann für die interessanten Gespräche über robuste Schätzer.

Für die Betreuung und anregende Gespräche möchte ich mich bedanken bei Paul Fischer, Thomas Hofmeister und Ingo Wegener und bei der Deutschen Forschungsgemeinschaft für die finanzielle Unterstützung meiner Arbeit.

Vielen Dank an die statistische Community für die interessanten Optimierungsprobleme!

Inhaltsverzeichnis

1 Einführung in die robuste Statistik

Ein Ausgangspunkt der robusten Statistik ist, dass der Least-Squares-Schätzer zwar einfach zu berechnen ist, aber Probleme mit Ausreißern hat. Es genügt, dass ein Punkt des Datensatzes von den anderen weit entfernt ist, um das Ergebnis stark zu verfälschen. Die robuste Statistik hat das Ziel, Schätzer zu finden, die wenig sensitiv gegenüber Ausreißern sind. Allerdings haben diese robusten Schätzer häufig den Nachteil, dass ad-hoc kein schneller Algorithmus für ihre Berechnung zur Verfügung steht. Was für die Informatik natürlich ein Vorteil ist, da hier interessante Probleme warten, für die Algorithmen entworfen werden müssen. In dieser Arbeit werden wir neue Algorithmen für einige robuste Schätzer vorstellen. Nachdem wir in den Kapiteln 1.1-1.4 eine Einführung in die robuste Statistik gegeben und in Kapitel 2 die verwendeten algorithmischen Methoden vorgestellt haben, behandeln wir die folgenden Schätzer und stellen jeweils neue Algorithmen vor:

- Kapitel 3: Algorithmen für den LQD-Schätzer (Least Quartile Difference) und das Multiresolutions-Kriterium, die beide auf Punktmengen in der Ebene arbeiten.
- Kapitel 4: Update-Algorithmen für den RM-Schätzer (Repeated Median) und den MAD-Schätzer (Median Absolute Deviation).
- Kapitel 5: Algorithmen für die Schätzer LMS (Least Median of Squares) und MCD (Minimum Covariance Determinant), die auf d-dimensionalen Punktmengen arbeiten.

Abschließend untersuchen wir in Kapitel 6 die Grenzen der Berechenbarkeit robuster Schätzer und beenden die Arbeit mit einem Ausblick in Kapitel 7.

1.1 Ausreißer

Eine der wichtigsten Wahrscheinlichkeitsverteilungen in der Statistik ist die Normalverteilung. Sie ist parametrisiert durch den Erwartungswert μ und die Standardabweichung σ. Liegen n Datenwerte $x_1, \ldots, x_n$ vor, die einer Normalverteilung entstammen, so lässt sich der Parameter μ aus den Daten mit $\frac{1}{n}\sum_i x_i$ schätzen. Das Problem ist nun: Wenn wir einen Punkt ins Unendliche verschieben, dann leistet der Punkt zu der Summe einen unendlich großen Beitrag. Die Schätzung des Erwartungswertes wird also durch einen einzigen Ausreißer beliebig stark verfälscht. Wir werden im nächsten Abschnitt ein Konzept darstellen, das sich dieses Problems annimmt. Barnett und Lewis (1994) definieren einen Ausreißer wie folgt:

Definition 1 (Ausreißer) *Ein Ausreißer ist ein Wert, der eine deutliche Abweichung zu den übrigen Datenpunkten aufweist.*

Barnett und Lewis (1994) geben folgende Quellen für Ausreißer an:

Inhärente Variabilität. Viele Verteilungen, wie z.B. die Normalverteilung oder Heavy-Tail-Verteilungen, liefern Werte aus $\mathbb{R}$ und sind nicht beschränkt. Die Wahrscheinlichkeit, dass ein Datensatz, der aus einer solchen Verteilung gezogen wurde, einige Datenwerte enthält, die bedeutend größer sind als andere, ist von Null verschieden. Diese natürliche Streuung kann eine Quelle für Ausreißer sein. Auch kann der datenerzeugende Mechanismus so gestaltet sein, dass mit kleiner Wahrscheinlichkeit die Werte aus einer anderen Verteilung stammen, die Ausreißer produziert. Zum Beispiel unterscheidet sich in der „Intrusion Detection" gewöhnlicher Datenverkehr von Einbruchsversuchen in Rechnernetzwerke, die typischerweise Ausreißer in den Daten darstellen, siehe auch Eskin, Arnold, Prerau, Portnoy und Stolfo (2002).

Fehler in den Daten. Bei der manuellen Datenerhebung gibt es zahlreiche Fehlerquellen: Zahlendreher, falsch eingegebene Werte und Vermischung unterschiedlicher Einheiten, wie z.B. beim Verlust der Raumsonde „Mars Climate Orbiter". Die Untersuchung ergab, dass eine Abteilung der NASA die Einheit „Fuß", eine andere Abteilung die Einheit „Meter" verwendet hatte. Dieser Fehler führte zu einer falschen Flugbahn und die Sonde verglühte in der Mars-Atmosphäre. [1]

Fehlerhaft gezogene Stichprobe. Da es häufig aufwendig ist, die Gesamtheit der Daten zu erheben, begnügt man sich damit, nur eine zufällige Stichprobe zu ziehen. Die Ergebnisse, die aus der Stichprobe gewonnen wurden, werden dann auf die Gesamtheit der Daten verallgemeinert. So werden z.B. vor Bundestagswahlen Meinungsumfragen durchgeführt, um das Wahlergebnis vorherzusagen. Hierbei kann es passieren, dass die Stichprobe verzerrt ist und Daten enthält, die nicht die Gesamtheit repräsentieren.

[1] http://mars.jpl.nasa.gov/msp98/news/mco991110.html

Um auf Daten, die Ausreißer enthalten, trotzdem den Erwartungswert z.B. einer Normalverteilung schätzen zu können, wird der Median der Daten verwendet. Wir vereinfachen die Notation, indem wir zunächst die geordnete Urliste definieren:

Definition 2 (geordnete Urliste) *Für eine Menge von Zahlen $x_1, \ldots, x_n$ bezeichnet $x_{(1)}, \ldots, x_{(n)}$ (mit in Klammern gesetzten Indizes) die sortierte Folge der Elemente, wobei das kleinste Element mit $x_{(1)}$ bezeichnet wird.*

Der Median ist definiert als ein Element aus dem Intervall $\left[x_{(\lfloor \frac{n+1}{2} \rfloor)}, x_{(\lceil \frac{n+1}{2} \rceil)}\right]$. Um einen eindeutigen Wert zu erhalten, verwendet die Statistik, siehe z.B. Fahrmeir, Künstler, Pigeot und Tutz (2002), den folgenden Median:

Definition 3 (Median) *Für eine Menge von Zahlen $x_1, \ldots, x_n$ ist der Median definiert als das mittlere Element in der sortierten Folge, für gerades n als der Durchschnitt aus den beiden mittleren Elementen:*

$$\underset{i=1,\ldots,n}{\text{med}}(x_i) = \begin{cases} x_{\left(\frac{n+1}{2}\right)} \ , & \text{für } n \text{ ungerade} \\ \frac{1}{2}\left(x_{(n/2)} + x_{(n/2+1)}\right) \ , & \text{für } n \text{ gerade} \ . \end{cases}$$

Wir werden in der Arbeit diese Variante verwenden, wohingegen in der Informatik im Normalfall die folgende Definition verwendet wird:

Definition 4 (Median linksseitige Variante) *Für eine Sequenz von Zahlen $x_1, \ldots, x_n$ ist der Median definiert als das Element*

$$\underset{i=1,\ldots,n}{\text{med}}^{Left}(x_i) = x_{(\lceil n/2 \rceil)} \ .$$

Der rechtsseitige Median med^{Right} ist analog definiert. Dor und Zwick (1999) geben einen deterministischen Algorithmus an, der mit weniger als $2.95n + o(n)$ Vergleichen den Median bestimmt. Der klassische Quickselect-Algorithmus, siehe Cormen, Leiserson, Rivest und Stein (2001, Kapitel 9.2), benötigt im Mittel weniger als $3.39n$ Vergleiche. Kiwiel (2004) verbessert die Strategie, mit der das Pivot-Element gewählt wird, und beschreibt einen Algorithmus, der im Mittel $1.5n + o(n)$ Vergleiche verwendet.

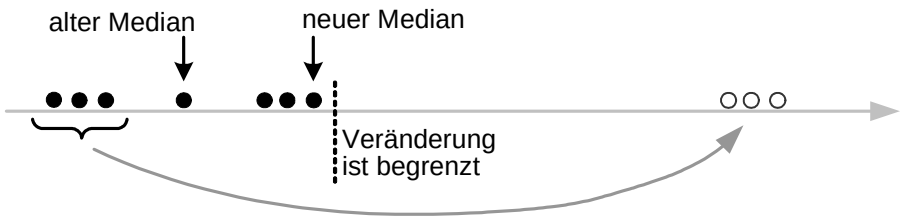
Aus med^{Left} und med^{Right} lässt sich zwar der *med* berechnen, aber es gibt Probleme, wie z.B. das $\text{RM}^{\mathcal{S}}$-Problem, das wir in Kapitel 2.2 beschreiben, bei denen nicht klar ist, wie eine Umrechnung effizient erfolgen soll.

1.2 Bruchpunkt

Donoho und Huber (1983) schlagen vor, die Robustheit eines Schätzers bezüglich Ausreißern wie folgt zu messen:

Definition 5 (Bruchpunkt) *Der* Finite-Sample-Ersetzungs-Bruchpunkt *eines Schätzers ist der kleinste Anteil von Datenpunkten, deren Ersetzung zu einer unbeschränkten Änderung der Schätzung führt.*

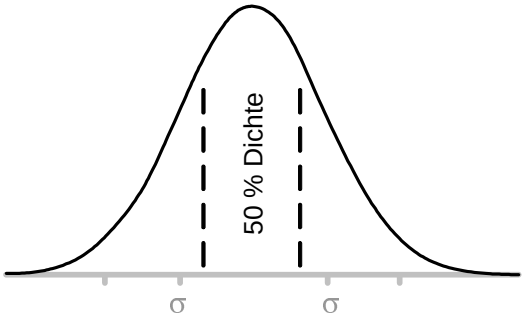
Dass der Median einen Bruchpunkt von 50% hat, lässt sich leicht einsehen: Wir betrachten n Datenwerte, n ungerade, und deren Median. Wir wählen $\lfloor n/2 \rfloor$ Datenwerte aus und verschieben sie ins Positiv-Unendliche. Da die nicht-verschobenen Datenwerte in der Mehrheit sind, entstammt der Median diesen Datenwerten. Der Median wird eventuell verändert, aber die Veränderung ist begrenzt, wie hier dargestellt:



Für die Varianz gibt es ebenfalls einen robusten Ersatz, den MAD-Schätzer (Median Absolute Deviation). Er ist wie folgt definiert:

Definition 6 (MAD-Schätzer) [2] *Für n Zahlen $x_1, \ldots, x_n$ mit $x_i \in \mathbb{R}$, und dem Median $\mu = \text{med}_{j=1,\ldots,n}(x_j)$ ist der MAD definiert als* $\text{MAD} = \text{med}_{i=1,\ldots,n} |x_i - \mu|$.

In der Abbildung rechts ist die Dichte der Normalverteilung dargestellt. Ziehen wir Punkte aus der Normalverteilung, so gibt der MAD-Schätzer für $n \to \infty$ die Breite des Intervalls $\mu \pm 0.67\sigma$ an, das genau 50% der Wahrscheinlichkeitsmasse enthält. Aus diesem Wert kann die Standardabweichung $\sigma = \frac{1}{0.67} \cdot \text{MAD}$ geschätzt werden. Für andere Verteilungen müssen andere Korrekturfaktoren verwendet werden.



[2] Ein Schätzer ist eine Funktion, die jeder Eingabe eine eindeutige Schätzung zuordnet.

1.3 Lineare Modelle

Ein Großteil der Schätzer, die in dieser Arbeit betrachtet werden, entstammt der parametrischen Statistik und es wird ein lineares Modell mit einer Hyperebene unterstellt. Die Parameter des Modells sind $\theta = (\theta_1, \ldots, \theta_d)$. Für ein gegebenes Modell wird ein Datenpunkt $P = (x_1, \ldots, x_{d-1}, y)$ generiert, indem zunächst die x_i bestimmt werden. Dies ist Aufgabe der so genannten Versuchsplanung und entweder erfolgt die Zuordnung deterministisch oder die x_i werden z.B. uniform zufällig ausgewürfelt. Die Koordinate y ergibt sich aus

$$y = \sum_{j=1,\ldots,d-1} \theta_j x_j + \theta_d + e_j \quad \text{mit } e_j \sim N(0, \sigma) \ ,$$

wobei die Normalverteilung $N(0, \sigma)$ mit Standardabweichung σ den Rauschanteil darstellt. Der vertikale Abstand eines Punktes zur Hyperebene (in Richtung der y-Achse) wird als Residuum $r(\theta)$ bezeichnet und ist gegeben durch

$$r(\theta) = y - \sum_{j=1,\ldots,d-1} \theta_j x_j - \theta_d \ .$$

Für n Punkte $P_1, \ldots, P_n$ bezeichnen wir die Residuen mit $r_1, \ldots, r_n$. Für eine Funktion f bezeichnet $f(r(\theta))_{(h)}$ das h-te Element in der sortierten Folge aller $f(r_i(\theta))$ und $f(r(\theta))_{(1)}$ ist das Residuum mit dem kleinsten Wert. Der bekannteste Regressions-Schätzer ist der LS-Schätzer (Least-Squares):

Definition 7 (LS-Schätzer) *Für n Punkte $P_1, \ldots, P_n$ ist der LS-Schätzer gegeben durch die Hyperebene θ, die $\frac{1}{n}\sum_{i=1}^{n} r_i(\theta)^2$ minimiert.*

Der LS-Schätzer hat einen Bruchpunkt von $1/n$. Das ist darin begründet, dass der Schätzer berechnet werden kann, indem ein lineares Gleichungssystem gelöst wird. Lassen wir den Wert einer Koordinate eines Punktes gegen unendlich laufen, so beeinflusst dieser Punkt das Gleichungssystem sehr stark und die Schätzung wird beliebig verfälscht. Die grundlegende Idee, Ausreißer einfach zu ignorieren, wird im LMS-Schätzer (Least-Median-of-Squares) verdeutlicht:

Definition 8 (LMS-Schätzer) *Sei $h = \lceil n/2 \rceil + \lceil (d+1)/2 \rceil$. Für n Punkte aus $\mathbb{R}^d$ ist der LMS-Schätzer gegeben durch die Hyperebene θ, die $(r(\theta)^2)_{(h)}$ minimiert.*

Der LMS-Schätzer ignoriert also Punkte mit einem Residuum größer als $(r^2)_{(h)}$, die somit keinen Einfluss auf das Ergebnis haben. Eine Veränderung dieser Punkte kann daher höchstens zu einer beschränkten Veränderung der Schätzung führen. Die Robustheit von LMS ist in Abbildung 1 zu erkennen:

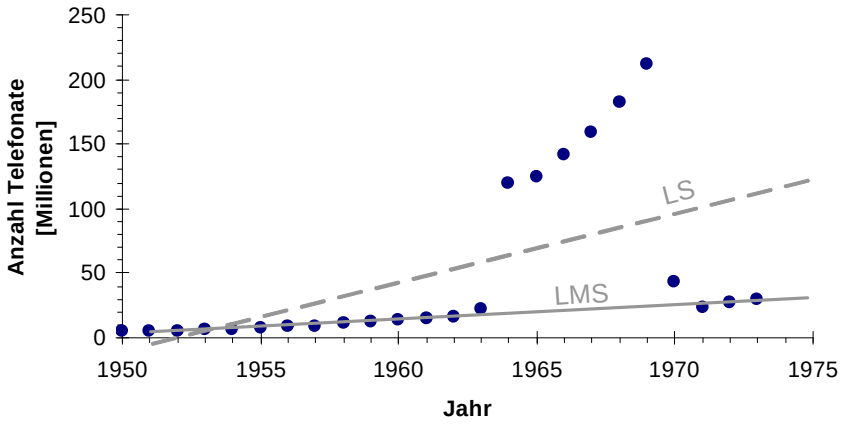


Abbildung 1: (Rousseeuw und Leroy (1987)) Dargestellt ist die Anzahl der internationalen Telefonate aus Belgien. In den Jahren 1964 bis 1969 wurden jedoch die Längen der Telefonate in Minuten gemessen, 1963 und 1970 jeweils eine Mischung aus beidem. Eingezeichnet sind die Regressionsgeraden des LS- und LMS-Schätzers. Der LMS-Schätzer wird von den Ausreißern nicht beeinflusst.

1.4 Exact-Fit-Property, Äquivarianz und weitere Eigenschaften

Schätzer, die immer eine konstante Schätzung ausgeben, sind zwar robust, aber sinnlos. Um sinnlose Schätzer mit hohem Bruchpunkt auszuschließen, wird Konsistenz gefordert: Ein Schätzer ist *konsistent*, falls für $n \to \infty$ der Schätzer gegen den wahren Wert eines unterstellten Modells konvergiert. Eine ähnliche Forderung, die wesentlich spezieller und eingeschränkter ist, geben Rousseeuw und Leroy (1987, Seite 120) an:

Definition 9 (Exact-Fit-Property) *Ein Regressions-Schätzer besitzt die Exact-Fit-Property mit Konstante c, wenn folgendes gilt: Falls von gegebenen n Punkten mindestens $\lceil c \cdot n \rceil$ Punkte exakt auf einer Hyperebene liegen, dann schätzt der Schätzer diese Hyperebene.*

Die Frage ist nicht nur, ob bestehende Schätzer schwierig zu berechnen sind, sondern auch, ob die Chance besteht, neue Schätzer zu finden, die die obigen Anforderungen erfüllen und die in einer Zeit polynomiell in der Dimension d berechnet werden können. In Kapitel 6 werden wir ein von der Exact-Fit-Property abgeleitetes Problem heranziehen, um die *NP*-Härte einiger robuster Schätzer zu zeigen.

Eine weitere Eigenschaft statistischer Schätzer ist die Äquivarianz: Sie besagt, dass bei einer Transformation der Datenpunkte $(x_1, y_1), \ldots, (x_n, y_n)$ mit $x_i \in \mathbb{R}^{d-1} \times \{1\}$ und $y_i \in \mathbb{R}$ die Schätzung in gleicher Weise transformiert wird. Einige sind hier aufgelistet:

Ein Schätzer T ist ...	mit $c \in \mathbb{R}$, $v \in \mathbb{R}^d$ und regulärer Matrix $A \in \mathbb{R}^{d \times d}$, wenn ...		
skalen-äquiv.	$T(\{\ldots, (x_i, cy_i), \ldots\})$	$=$	$c\ T(\{\ldots, (x_i, y_i), \ldots\})$
regressions-äquiv.	$T(\{\ldots, (x_i, y_i + x_i v), \ldots\})$	$=$	$v+\ T(\{\ldots, (x_i, y_i), \ldots\})$
affine-äquiv.	$T(\{\ldots, (x_i A, y_i), \ldots\})$	$=$	$A^{-1}\ T(\{\ldots, (x_i, y_i), \ldots\})$

Das folgende Theorem von Rousseeuw (1994) zeigt einen Zusammenhang zwischen der Exact-Fit-Property und der Regressions- bzw. Skalen-Äquivarianz:

Theorem 10 *Wenn ein Schätzer T regressions- und skalen-äquivariant ist und einen Bruchpunkt von $1 - c$ hat für Daten in allgemeiner Lage, dann erfüllt er die Exact-Fit-Property mit Konstante c.*

Lassen wir die Forderung nach Regressions- und Skalen-Äquivarianz fallen, so finden wir in der Praxis eingesetzte Schätzer, die häufig auf der Euklidischen Norm basieren. Beispiele aus dem Data Mining[3] sind „Nearest Neighbors", „Decision Trees" und „hierarchical Clustering". Diese lassen sich in polynomieller Zeit berechnen. Hingegen ist z.B. zwar das k-Means-Clustering-Problem *NP*-hart, es kann jedoch mit Güte $(1 + \varepsilon)$ approximiert werden, siehe Kumar, Sabharwal und Sen (2004).

Eine weitere, statistisch interessante Eigenschaft ist die *Gaußsche Effizienz:* Werden für ein Datenmodell mehrmals hintereinander Daten erzeugt, so wird aufgrund des Rauschens die Schätzung ebenfalls streuen. Die Varianz eines Schätzers ist ein Maß für diese Streuung, wobei der LS-Schätzer die kleinstmögliche Varianz für normalverteilte Fehler besitzt. Es ist gewünscht, dass robuste Schätzer ebenfalls eine geringe Varianz haben. Die Gaußsche Effizienz ist definiert als das Verhältnis aus Varianz des LS-Schätzers zu der Varianz eines anderen (robusten) Schätzers. Zum Beispiel hat der LMS-Schätzer eine Gaußsche Effizienz von 0% für $n \to \infty$. Dies motiviert die Entwicklung besserer Schätzer, wie dem LQD-Schätzer (siehe Kapitel 3.1), der eine Effizienz von 67% besitzt.

Eine weitere Eigenschaft, die in der statistischen Literatur zwar nicht genannt wird, aber doch implizit gefordert wird, ist die, dass ein Schätzer *zuverlässig* implementiert sein soll. Die Zuverlässigkeit spiegelt sich z.B. darin wieder, dass die exakte Berechnung robuster Schätzer gefordert wird.

[3] http://en.wikipedia.org/wiki/Data_mining

2 Algorithmische Methoden

Die algorithmischen Methoden, die in späteren Kapiteln Verwendung finden, stellen wir hier kurz vor. Dies sind einerseits die klassische Dualität und allgemeinere Problemtransformationen sowie die Nutzung der Entscheidungsvariante.

2.1 Klassische Dualität und Problemtransformationen

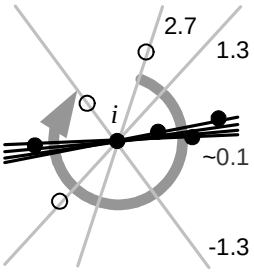
Wir werden die Dualität an dem RM-Schätzer (Repeated-Median) veranschaulichen. Das zugehörige Problem ist wie folgt definiert:

Definition 11 (RM-Problem (Repeated Median))

`Eingabe:`
n Punkte $(x_1, y_1), \ldots, (x_n, y_n)$ in der Ebene, $x_i \neq x_j$ für alle $1 \leq i < j \leq n$.
`Problem:`
Die Steigung der Geraden durch die Punkte (x_i, y_i) und (x_j, y_j) ist $a_{i,j} = \frac{y_i - y_j}{x_i - x_j}$.
Finde den Repeated-Median-Schätzer $(\beta_{\mathrm{RM}}, \alpha_{\mathrm{RM}})$, der gegeben ist durch

$$\beta_{\mathrm{RM}} = \operatorname*{med}_{i=1..n} \underbrace{\operatorname*{med}_{j=1..n, j\neq i} (a_{i,j})}_{\text{innere Mediane}} ,$$

$$\alpha_{\mathrm{RM}} = \operatorname*{med}_{i=1..n} (y_i - \beta_{\mathrm{RM}} x_i) .$$



Wir überzeugen uns zunächst, dass der RM-Schätzer eine Regressions-Gerade schätzen kann. Das „wahre" Modell ist eine Gerade mit Steigung 0.1. In nebenstehender Abbildung ist zu sehen, dass mehr als die Hälfte der Punkte nahe der Gerade mit Steigung 0.1 liegen. Wählen wir einen festen Punkt i und rotieren eine Gerade um i, wobei wir an jedem Punkt stoppen, der auf der Gerade zum liegen kommt, so erhalten wir mehrere

Steigungen. Die Mehrheit der Steigungen liegt nahe bei 0.1, so dass auch der innere Median $\text{med}_i(a_{i,j})$ ungefähr 0.1 ist. Da mehr als die Hälfte der inneren Mediane ungefähr 0.1 sind, wird insgesamt die wahre Steigung geschätzt.

Sich beim Algorithmen-Entwurf vorzustellen, dass eine Gerade um einen Punkt rotiert, ist nicht sehr hilfreich. Einen möglichen Ausweg bietet die duale Transformation, die wir nun beschreiben.

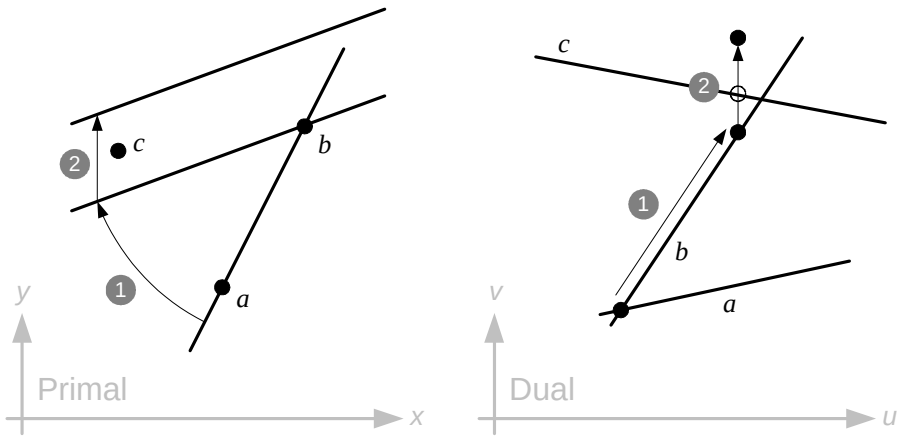


Abbildung 2: Klassische Dualität.

Im linken Teil der Abbildung 2 sind die Punkte a,b und c eingezeichnet. Die klassische Dualität bildet einen

$$\text{Punkt } (p_x, p_y) \text{ auf die Gerade } v = p_x u - p_y$$

ab, wobei u und v die Achsen des Dualraumes bezeichnen. Auf diese Weise erhalten wir die drei Geraden im rechten Teil der Abbildung 2. Betrachten wir zunächst im linken Teil die Gerade $y = \beta x + \alpha$, die durch die Punkte $a = (a_x, a_y)$ und $b = (b_x, b_y)$ verläuft. Die Steigung β und der y-Achsenabschnitt α der Geraden sind gegeben durch

$$\beta = \frac{a_y - b_y}{a_x - b_x} \quad \text{und} \quad \alpha = \frac{a_y b_x - a_x b_y}{b_x - a_x} \ .$$

Betrachten wir nun die dualen Geraden a und b, so schneiden sich diese in dem Punkt

$$\left(\frac{a_y - b_y}{a_x - b_x}, \frac{a_y b_x - a_x b_y}{b_x - a_x} \right) = (\beta, \alpha) \ ,$$

wie eine kurze Rechnung zeigt. Wir sehen, dass eine

$$\text{Gerade } y = \beta x + \alpha \text{ auf den Punkt } (\beta, \alpha)$$

abgebildet wird.

Ein Punkt (p_x, p_y) liegt *oberhalb* einer Geraden $v = p_x u + p_y$, falls ein $c > 0$ existiert, so dass der nach unten verschobene Punkt $(p_x, p_y - c)$ sich auf der Geraden befindet. Die Begriffe *unterhalb, links*, und *rechts* sind analog definiert. Um die Sprechweise zu vereinfachen, definieren wir, dass ein Punkt eine Gerade *dominiert*, falls der Punkt oberhalb oder auf der Geraden liegt. Eine weitere Eigenschaft der Dualität ist, dass räumliche Beziehungen erhalten bleiben: Falls ein Punkt sich oberhalb einer Geraden befindet, so ist die duale Gerade ebenfalls oberhalb des dualen Punktes, analoges gilt für oberhalb, links und rechts.

Um die Dualität detaillierter zu veranschaulichen, sind in Abbildung 2 zwei Bewegungen eingezeichnet. Wir betrachten auf der linken Seite die Gerade, die durch die Punkte a und b verläuft. In Bewegung 1 wird die Gerade um den Punkt b gedreht, um danach in Bewegung 2 vertikal verschoben zu werden, wobei der Punkt c passiert wird. Auf der rechten Seite der Abbildung ist der Dualraum dargestellt, und wie sich der duale Punkt bewegt. In Bewegung 1 bewegt er sich entlang einer Geraden, in Bewegung 2 wird er vertikal verschoben, wobei der Punkt die Gerade c passiert.

Wie bereits beschrieben, entspricht eine Rotation um den Punkt i dem Verschieben eines Punktes entlang der dualen Gerade i, wobei die anderen dualen Geraden bzw. Schnittpunkte passiert werden. Somit entsprechen die u-Koordinaten der Schnittpunkte den gesuchten Steigungen $a_{i,1}, \ldots, a_{i,n}$ des RM-Problems, die zudem sortiert auf der Geraden i angeordnet sind. Der innere Median für ein gewähltes i ist somit der mediane Schnittpunkt auf der Geraden i. Dies gilt jedoch nur, falls $n - 1$ ungerade ist. Wenn $n - 1$ gerade ist, dann ist nach der Definition des Medians der Mittelpunkt des medianen Segmentes zu wählen. Um die Beschreibung zu vereinfachen, werden wir im Weiteren vom inneren Median der Gerade i sprechen. Der zweite Schritt ist genau wie im Primalen. Aus allen n inneren Medianen wird der mediane Wert berechnet. Wir erhalten die folgende duale Formulierung des RM-Schätzers:

Definition 12 ($\mathrm{RM}^{\mathcal{T}}$-Problem)

`Eingabe:`

n Geraden $\ell_1 : v = x_1 u + y_1, \ldots, \ell_n : v = x_n u + y_n$ in der Ebene mit $x_i \neq x_j$.

`Problem:`

Sei $u_{i,j}$ die u-Koordinate des Schnittpunktes der beiden Geraden ℓ_i und ℓ_j. Berechne

$$\beta_{\mathrm{RM}} = \operatorname*{med}_{i=1,\ldots,n} \quad \operatorname*{med}_{j=1,\ldots,n, j\neq i} \quad (u_{i,j})$$
$$\alpha_{\mathrm{RM}} = \operatorname*{med}_{i=1,\ldots,n} \quad (y_i - \beta_{\mathrm{RM}} x_i) \ .$$

Da die Schnittpunkte für festes i alle in sortierter Folge auf der Geraden i aufgereiht sind, wie in Abbildung 3 zu sehen, bietet diese Sichtweise einen vereinfachten Zugang

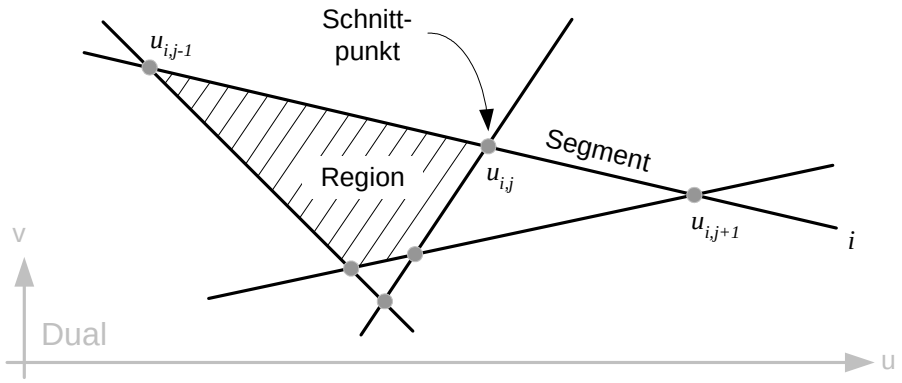


Abbildung 3: Die n Geraden im Dualen bilden ein *Arrangement von Geraden*, wie hier dargestellt. Das Arrangement unterteilt die Ebene in Regionen/Zellen und die Geraden in Segmente.

zum RM-Problem. Im Kapitel 4.2 werden wir eine Datenstruktur für Arrangements von Geraden beschreiben und in einem Algorithmus verwenden, der in linearer Zeit ein Update des RM-Schätzers berechnet.

Eine andere Transformation findet sich bei dem LMS-Problem in Kapitel 5.1. In der so genannten geometrischen Sicht muss für das LMS-Problem ein Streifen gefunden werden, der h Punkte enthält und möglichst schmal ist. Die Bewegung 2 in Abbildung 2 können wir als Streifen auffassen und wir sehen, dass er im Dualen auf ein vertikales Intervall abgebildet wird. In Kapitel 5.1 werden wir eine Transformation verwenden, die die Breite des Streifens/Intervalls in einer zusätzlichen Achse kodiert, so dass die Lösung ein Punkt im dualen Raum ist.

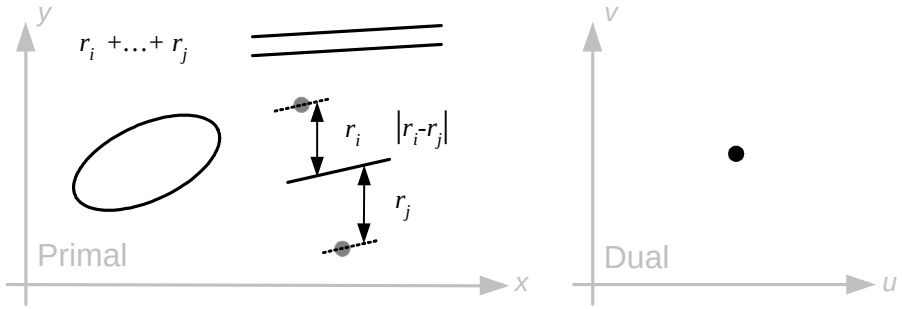
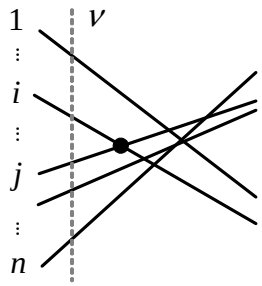


Abbildung 4: Alle Formen auf der linken Seite, die eine Lösung repräsentieren, werden auf einen Punkt im Dualen abgebildet.

Wie in Abbildung 4 angedeutet, werden neben Hyperstreifen auch Ellipsoide, Residuen-Differenzen und Summen als Lösungen auftauchen und wir werden jeweils geeignete Transformationen angeben, die die Lösung auf einen Punkt abbilden.

2.2 Entscheidungsvariante nutzen



Matoušek, Mount und Netanyahu (1998) verwenden eine weitere Charakterisierung des RM-Schätzers, wie folgt: Wir bilden die Punkte auf ihre dualen Geraden ab und betrachten das Geraden-Arrangement. Wir nehmen eine senkrechte Linie ν und platzieren sie links vom Arrangement, so dass alle Schnittpunkte sich rechts davon befinden und nummerieren die Geraden bezüglich des Schnittes mit ν von oben nach unten mit $1, \ldots, n$, wie in der Abbildung links zu sehen. Verschieben wir nun ν nach rechts, so treffen wir auf einen Schnittpunkt, z.B. der Geraden i und j mit $i < j$. Nachdem wir den Schnittpunkt überquert haben, haben die Schnitte der beiden Geraden auf ν ihre Position getauscht; oberhalb von i befindet sich ein Index, der größer ist, bzw. unterhalb von j befindet sich ein Index, der kleiner ist. Wir können also aus der Permutation der Geraden auf die Position der Schnittpunkte schließen. Zudem können wir die Anzahl der Schnittpunkte links von ν zählen, und erhalten somit die Information, ob innere Mediane sich links oder rechts von ν befinden. Somit können wir aus der Position aller inneren Mediane schließen, ob β_{RM} links oder rechts von ν ist. Es ergibt sich folgende Charakterisierung des RM-Schätzers:

Definition 13 (RM$^{\mathcal{S}}$-Problem (Suchvariante))

`Eingabe:`

n Geraden $\ell_1, \ldots, \ell_n$ in der Ebene, die nach wachsender Steigung sortiert sind.

`Problem:`

Für ein β sei $\nu(\ell_i)$ der Schnitt von ℓ_i mit einer vertikalen Linie an Position β. Wir definieren

$$\begin{aligned} \mathrm{inv}(\ell_i) = \quad & \#\{j \mid j < i \text{ und } \nu(\ell_i) > \nu(\ell_j)\} \\ & + \#\{j \mid j > i \text{ und } \nu(\ell_i) < \nu(\ell_j)\} \ , \\ L(\beta) = \ & \#\{i \mid \mathrm{inv}(\ell_i) < n/2\} \\ R(\beta) = \ & \#\{i \mid \mathrm{inv}(\ell_i) > n/2\} \ . \end{aligned}$$

Finde ein β_{RM}, so dass $|L(\beta_{\mathrm{RM}}) - R(\beta_{\mathrm{RM}})|$ minimal ist. Der Achsenabschnitt wird anschließend bestimmt mit $\alpha_{\mathrm{RM}} = \mathrm{med}_{i=1,\ldots,n}\ (y_i - \beta_{\mathrm{RM}} x_i)$.

Indem wir für verschiedene β anfragen, ob das gesuchte Optimum sich links oder rechts von β befindet, können wir mit einer binären Suche β_{RM} finden. Matoušek, Mount und Netanyahu (1998) geben einen Algorithmus an, der mit hoher Wahrscheinlichkeit den RM-Schätzer mit $\mathcal{O}(1)$ Anfragen berechnet.

Die Idee, die Entscheidungsvariante zu betrachten, findet sich auch beim altbekannten randomisierten Algorithmus, das Minimum der Zahlen $p_1, \ldots, p_n$ zu finden, siehe Chan (1999):

```
1. Wähle eine zufällige Permutation π
2. r* ← ∞
3. For i = 1, ..., n do
4.    if (p_π(i) < r*) then
5.       r* ← p_π(i)
6.  return r*
```

Es ist bekannt, dass die Zeile 5 im Mittel nur $\mathcal{O}(\log n)$-mal aufgerufen wird. Wenn wir uns nun vorstellen, dass jeder Eintrag i ein Teilproblem eines komplizierteren Optimierungsproblems ist, so stellt Zeile 4 die Entscheidungsvariante dar, die die Frage beantwortet, ob es eine bessere Lösung im Teilproblem i gibt. Zeile 5 hingegen ist die Optimierungsvariante, die den Wert p_i der besten Lösung des Teilproblems i bestimmt. Wenn die Entscheidungsvariante (Zeile 4) billiger in der Berechnung ist als die Optimierungsvariante (Zeile 5), kann hier Rechenzeit eingespart werden. Aus diesen Überlegungen ergibt sich das folgende Lemma:

Lemma 14 *Wir betrachten ein Problem* Π *und dessen Lösungsraum. Die folgenden Eigenschaften seien erfüllt:*

- *Der Lösungsraum kann in* n *Unterräume* $A_1, \ldots, A_n$ *unterteilt werden, wobei mindestens ein* A_i *eine optimale Lösung enthält.*
- *Es kann in Zeit* T_D *entschieden werden, ob eine Lösung im Unterraum* A_i *existiert, deren Lösungswert kleiner gleich einem gegebenen* r^* *ist.*
- *Es kann in Zeit* T_S *die optimale Lösung des Unterraumes* A_i *berechnet werden.*

Dann kann eine optimale Lösung von Π *in erwarteter Zeit* $\mathcal{O}(n \cdot T_D + T_S \cdot \log n)$ *berechnet werden.*

Wir werden das Lemma in Kapitel 5.1 für den Algorithmus zur Berechnung des LMS-Schätzers verwenden. Einen ähnlichen Ansatz werden wir in Kapitel 3.1 verfolgen und die Entscheidungsvariante mit der Geometrischen Suche kombinieren, um den LQD-Schätzer zu berechnen.

3 Schätzer in der Ebene

Wir beginnen das erste algorithmische Kapitel mit Schätzern, die auf Punktmengen in zwei Dimensionen arbeiten. Hier bietet die algorithmische Geometrie viele Konzepte an, die beim Algorithmen-Entwurf hilfreich sind und die bei vielen Schätzern zu effizienten Algorithmen führen.

3.1 LQD-Schätzer

Wie in der Einleitung angesprochen, bietet der LQD-Schätzer gegenüber anderen Schätzern, wie z.B. LMS, den Vorteil, dass die Gaußsche Effizienz besser ist. Regressions-Schätzer basieren üblicherweise auf Residuen $r_i(\ell) = y_i - \beta x_i - \alpha$ für eine Gerade $\ell : y = \beta x + \alpha$. Die grundlegende Idee beim LQD-Schätzer ist es, Residuen-Differenzen $r_{i,j}(\ell)$ zu betrachten:

$$r_{i,j}(\ell) = |r_i(\ell) - r_j(\ell)| = |(y_i - y_j) - \beta(x_i - x_j)| \ .$$

Bei dieser Rechnung fällt der y-Achsenabschnitt α heraus. Der LQD-Schätzer kann deswegen nur die Steigung β schätzen. Croux, Rousseeuw und Hössjer (1994) definieren ihn wie folgt:

Definition 15 (LQD-Problem (Least Quartile of Difference))

`Eingabe:`
n Punkte $(x_1, y_1), \ldots, (x_n, y_n)$ in der Ebene.

`Problem:`
Sei $h = \lfloor (n+3)/2 \rfloor$ und sei $r_{i,j}(\beta) = |(y_i - y_j) - \beta(x_i - x_j)|$. Finde eine Steigung β, so dass das $\binom{h}{2}$-te Element aus der sortierten Sequenz $(r_{i,j}(\beta) \mid 1 \leq i < j \leq n)$ minimal ist.

Croux, Rousseeuw und Hössjer (1994) stellen einen exakten Algorithmus mit Rechenzeit $\mathcal{O}(n^5 \log n)$ vor, der eine Transformation beinhaltet, die die Eingabe auf ein LMS-Problem abbildet. Tauschen wir den von Ihnen verwendeten LMS-Algorithmus gegen

den von Edelsbrunner und Souvaine (1990) aus, der eine Rechenzeit $\mathcal{O}(n^2)$ hat, so erhalten wir einen exakten Algorithmus mit Rechenzeit $\mathcal{O}(n^4)$. Agulló (2002) gibt heuristische Algorithmen an, analysiert allerdings keine Rechenzeiten.

Es gibt einige wenige Arbeiten, die den LQD-Schätzer verwenden. Dryden und Walker (1999) schlagen vor, LQD für Matchingprobleme in der Biologie zu verwenden. Wand, Shotts, Sekhon, Mebane Jr., Herron und Brady (2001) verwenden LQD, um Ausreißer in Stimmenauszählungen der Präsidentschaftswahlen zu finden.

Problemtransformation

Um das Problem besser handhaben zu können, verwenden wir eine Transformation, die eine Residuen-Differenz auf ein Geradenpaar abbildet, wobei die optimale Lösung wieder auf einen Punkt abgebildet wird. Sie ist in Abbildung 5 auf Seite 26 illustriert.

Definition 16 (LQD$^{\mathcal{T}}$-Problem)

`Eingabe:`
n Punkte $(x_1, y_1), \ldots, (x_n, y_n)$ in der Ebene.
Um die Beschreibung zu vereinfachen, nehmen wir an, dass die Punkte nach der x-Koordinate aufsteigend sortiert sind.

`Transformation:`
Jedes Punktepaar (x_i, y_i) und (x_j, y_j) mit $i < j$ wird auf die Geraden

$$\ell^+_{i,j} : v = +(x_i - x_j)u - (y_i - y_j)$$
$$\ell^-_{i,j} : v = -(x_i - x_j)u + (y_i - y_j)$$

abgebildet. Die Achsen des neuen Raumes sind u und v. Da die Punkte sortiert sind, haben alle Geraden $\ell^+_{i,j}$ eine negative Steigung und alle Geraden $\ell^-_{i,j}$ eine positive.

`Problem:`
Sei $h = \lfloor (n+3)/2 \rfloor$. Finde einen Punkt (β, r), so dass er $\binom{n}{2} + \binom{h}{2}$ Geraden dominiert[1] und r minimal ist.

Theorem 17 *Aus der minimalen Lösung (β, r) des LQD$^{\mathcal{T}}$-Problems erhalten wir eine minimale Lösung β des LQD-Problems, wobei das $\binom{h}{2}$-te Residuum den Wert r annimmt.*

[1] Ein Punkt *dominiert* eine Gerade, falls er oberhalb ist oder die Gerade schneidet. Wir verwenden den Begriff „schneiden", um die Sprechweise zu vereinfachen. Er entstammt der Anschauung, dass Punkt und Gerade als Mengen aufgefasst werden können und bezeichnet somit einen nichtleeren Schnitt zweier Mengen.

Beweis: Wir betrachten die beiden Geraden $\ell^+_{i,j}$ und $\ell^-_{i,j}$. Es gibt drei mögliche Konfigurationen:

1. Der Punkt (β, r) dominiert $\ell^+_{i,j}$ und $\ell^-_{i,j}$.
2. Eine der Geraden $\ell^+_{i,j}$ oder $\ell^-_{i,j}$ ist oberhalb von (β, r), die andere Gerade ist unterhalb oder schneidet.
3. Der Punkt (β, r) ist unterhalb von $\ell^+_{i,j}$ und $\ell^-_{i,j}$.

Ein Punkt (β, r) dominiert eine Gerade $v = au + b$, falls $a\beta + b \leq r$. Es genügt, Konfigurationen 1 zu betrachten:

$$\begin{aligned} & (\beta, r) \text{ dominiert } \ell^+_{i,j} \text{ und } \ell^-_{i,j} \\ \Leftrightarrow\ & (x_i - x_j)\beta - (y_i - y_j) \leq r \text{ und } -(x_i - x_j)\beta + (y_i - y_j) \leq r \\ \Leftrightarrow\ & |(x_i - x_j)\beta - (y_i - y_j)| \leq r \\ \Leftrightarrow\ & |r_{i,j}(\beta)| \leq r \ . \end{aligned} \tag{3.1}$$

Der Punkt (β, r) hat die Eigenschaft, mindestens $\binom{n}{2} + \binom{h}{2}$ Geraden zu dominieren. Aus Abzählargumenten folgt, dass es $\binom{h}{2}$ Paare (ℓ^+, ℓ^-) gibt, so dass beide Geraden dominiert werden. Somit folgt aus Gleichung 3.1, dass es $\binom{h}{2}$ Residuen-Differenzen gibt, die kleiner oder gleich r sind.

Um zu beweisen, dass $r \geq 0$ der Wert der kleinsten $\binom{h}{2}$-ten Residuen-Differenz ist, nehmen wir das Gegenteil an. Sei r' der Wert einer $\binom{h}{2}$-ten Residuen-Differenz mit $r' < r$. Dann können wir aus Gleichung 3.1 folgern, dass es einen Punkt (β', r') gibt, der $\binom{n}{2} + \binom{h}{2}$ Geraden dominiert. Da er eine bessere Lösung als das Optimum (β, r) ist, erhalten wir einen Widerspruch. Daher ist die Steigung β eine optimale Lösung für das LQD-Problem. □

Exakter Algorithmus für das LQD-Problem

Das LQD$^{\mathcal{T}}$-Problem kann modelliert werden als Lineare Programmierung mit k verletzten Nebenbedingungen. Die Geraden $\ell^+_{i,j}$ und $\ell^-_{i,j}$ bzw. die zugehörigen Halbräume lassen sich als Nebenbedingungen auffassen, von denen $\binom{n}{2} + \binom{h}{2}$ erfüllt sein sollen. Minimiert wird die v-Koordinate des Suchpunktes.

Für m Nebenbedingungen in zwei Dimensionen geben Cole, Sharir und Yap (1987) und Roos und Widmayer (1994) Algorithmen mit einer Rechenzeit von $\mathcal{O}\big(m \log^2 m\big)$ an, die

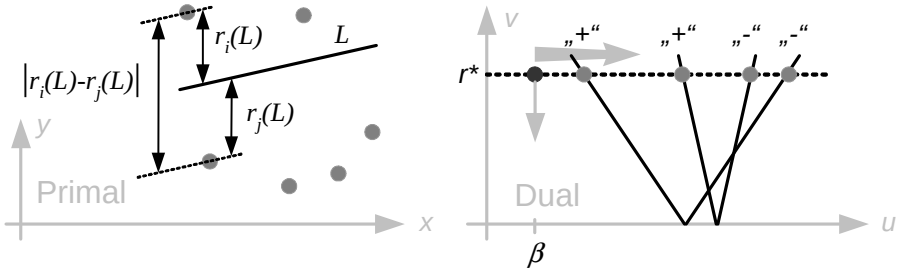


Abbildung 5: Links sind die Residuen-Differenzen im Primalraum dargestellt, rechts die Geraden, die sich aus der Transformation ergeben.

parametrische Suche verwenden (siehe Megiddo (1979)). Chan (1999) gibt einen randomisierten Algorithmus an, der so genannte Cuttings verwendet. Er hat eine erwartete Rechenzeit von $\mathcal{O}(m \log m)$. Da das LQD$^{\mathcal{T}}$-Problem $\mathcal{O}(n^2)$ Nebenbedingungen hat, ergibt sich hieraus die Rechenzeit $\mathcal{O}(n^2 \log n)$ und wir erhalten das folgende Theorem:

Theorem 18 *Das LQD-Problem kann in erwarteter Zeit $\mathcal{O}(n^2 \log n)$ exakt gelöst werden.*

Der Nachteil der obigen Ansätze ist, dass sie nur schwer zu implementieren sind. Cuttings wurden von Har-Peled (1998) implementiert, aber er hat Probleme mit Rundungsfehlern und verwendet daher eine Bibliothek, die das Rechnen mit rationalen Zahlen erlaubt, und entsprechend langsam ist. Auch parametrische Suche verwendet komplexe Techniken, die zu einer hohen Konstante in der Laufzeit führen, siehe van Oostrum und Veltkamp (2004).

Daher stellen wir als nächstes einen neuen Approximationsalgorithmus vor, der einfach implementierbar ist und keine Probleme mit Rundungsfehlern hat.

Entscheidungsvariante

Wir beschreiben zunächst die Prozedur DecideLQD(r^*), die für ein festes r^* entscheidet, ob es eine gültige Lösung (β, r) mit $r \leq r^*$ gibt. Da r^* auf der vertikalen Achse aufgetragen ist, bezeichnen wir diesen Wert auch als *Höhe* r^*. In der Höhe r^* ziehen wir eine horizontale Gerade H_{r^*} und berechnen die Schnittpunkte von H_{r^*} mit allen Geraden $\ell^+_{i,j}$ und $\ell^-_{i,j}$ für $i = 1, \ldots, n-1$ und $j = i+1, \ldots, n$. Wir versehen einen Schnittpunkt mit dem Label „+”, falls er von einer Gerade $\ell^+_{i,j}$ stammt. Anderenfalls versehen wir ihn mit dem Label „−”.

Der Algorithmus ist in Abbildung 5 veranschaulicht. Die Idee ist, einen Punkt auf H_{r^*} von links nach rechts zu schieben. Wir bestimmen am ersten Schnittpunkt die Anzahl C der Geraden, die sich unterhalb befinden oder schneiden. Nachdem wir alle Schnittpunkte nach ihrer u-Koordinate sortiert haben, durchlaufen wir sie von links nach rechts. Wenn wir einen Schnittpunkt mit Label „+" finden, wird C um Eins erhöht, da sich nun eine Gerade mehr unterhalb befindet. Ansonsten wird C um Eins vermindert. Falls wir einen Schnittpunkt mit $C \geq \binom{h}{2} + \binom{n}{2}$ finden, ist dieser eine gültige Lösung mit Wert r^*, die die Entscheidungsvariante zusätzlich ausgibt.

Die Prozedur DecideLQD hat die Rechenzeit $\mathcal{O}(n^2 \log n)$, da die Schnittpunkte sortiert werden müssen.

Approximations-Algorithmus für das LQD-Problem

Um den optimalen Lösungswert $r^* > 0$ einzugrenzen, verwendet der Algorithmus eine obere Schranke $r_{\max}$ und eine untere Schranke $r_{\min}$. Die Güte einer Lösung ist definiert als $r_{\max}/r^*$. Das Ziel ist es, Schranken zu bestimmen, so dass $r_{\max}/r_{\min} \leq 1 + \varepsilon$.

Um initiale Schranken zu erhalten, verfahren wir wie folgt. Zunächst rufen wir DecideLQD mit den Werten $\frac{1}{1+\varepsilon}, 1, 1 + \varepsilon$ auf und erfahren, ob die optimale Lösung in den Intervallen $[\frac{1}{1+\varepsilon}, 1]$ oder $[1, 1 + \varepsilon]$ liegt. Falls ja, haben wir eine Lösung mit Güte $1 + \varepsilon$ gefunden. Falls nicht, erhalten wir entweder eine obere Schranke $\frac{1}{1+\varepsilon}$ oder eine untere Schranke $1 + \varepsilon$. Wir beschreiben nun den Ablauf für die untere Schranke, der andere ist analog. Aus der unteren Schranke $1 + \varepsilon$ erhalten wir eine obere Schranke $r_{\max}$, indem wir $1 + \varepsilon$ solange quadrieren, bis DecideLQD JA ausgibt. Die Anzahl der benötigten Schritte k_1 hängt von dem optimalen Lösungswert r^* ab und ist der kleinste ganzzahlige Wert, der folgende Gleichung erfüllt:

$$\begin{aligned} & (1+\varepsilon)^{2^{k_1}} \geq r^* && \mid r^* > 1 + \varepsilon \\ \Leftrightarrow \quad & k_1 \geq \log\log r^* - \log\log(1+\varepsilon) \end{aligned}$$

Wir erhalten als obere Schranke $r_{\max} = (1 + \varepsilon)^{2^{k_1}}$ und als untere Schranke $r_{\min} = (1 + \varepsilon)^{2^{k_1-1}}$. Nun führen wir eine geometrische Suche durch und verwenden als Schachtelwert $r' = \sqrt{r_{\max} \cdot r_{\min}}$. Je nachdem, wie DecideLQD(r') entscheidet, wird $r_{\max}$ oder $r_{\min}$ auf den Wert von r' gesetzt. Für den Fall, dass $r_{\max}$ auf den Wert von r' gesetzt wird, erhalten wir, dass

$$\frac{r'}{r_{\min}} = \frac{\sqrt{r_{\max} \cdot r_{\min}}}{r_{\min}} = \sqrt{\frac{r_{\max}}{r_{\min}}} \ .$$

Die Güte wird also in jedem Schritt gewurzelt. Dies führen wir durch, bis die Güte $r_{\max}/r_{\min} \leq 1 + \varepsilon$ erreicht wird. Zu Beginn gilt, dass $r_{\max} = r_{\min}^2$ und es folgt, dass die

Güte $r_{\max}/r_{\min} = r_{\min} < r^*$. Die benötigten Schritte k_2, bis Güte $1+\varepsilon$ erreicht ist, lassen sich wie folgt abschätzen:

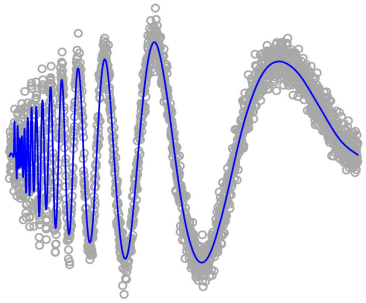
$$\begin{aligned} & (r^*)^{(1/2)^{k_2}} \leq 1+\varepsilon && \mid r^* > 1+\varepsilon \\ \Leftrightarrow \quad & k_2 \geq \log\log r^* - \log\log(1+\varepsilon) \ . \end{aligned}$$

Es genügen also $k_1 + k_2 = \mathcal{O}(\log\log r^* - \log\log(1+\varepsilon))$ Schritte, bis eine Lösung mit Güte $1+\varepsilon$ gefunden wird. Pro Schritt reicht ein Aufruf von DecideLQD:

Theorem 19 *Sei r^* der Wert der optimalen Lösung. Für n Punkte in der Ebene kann eine Lösung mit Güte $1+\varepsilon$ in folgender Zeit gefunden werden:*

$$\begin{cases} \mathcal{O}(n^2 \log n\,(\log\log r^* - \log\log(1+\varepsilon))) & , \textit{ wenn } r^* > 1+\varepsilon \\ \mathcal{O}\big(n^2 \log n\,(\log\log \frac{1}{r^*} - \log\log(1+\varepsilon))\big) & , \textit{ wenn } \frac{1}{r^*} > 1+\varepsilon \\ \mathcal{O}(n^2 \log n) & , \textit{ sonst } . \end{cases}$$

3.2 Multiresolutions-Kriterium



Im Gegensatz zum LQD-Schätzer, der robust eine Gerade anpasst, werden wir uns in diesem Kapitel mit einem nicht-robusten Ansatz befassen, der eine beliebige Kurve anpassen kann. Die Aufgabe ist, durch eine Punktmenge eine Kurve zu legen, die „gut aussieht". In nebenstehender Abbildung ist ein Beispiel zu sehen: Eine Kurve „sieht gut aus", falls die Residuen wie weißes Rauschen aussehen. Dies ist der übliche Ansatz in der Statistik, die Daten in Signal und Rauschen zu zerlegen. Davies und Kovac (2001) definieren das Multiresolutions-Kriterium, um mathematisch zu modellieren, wann Residuen wie weißes Rauschen aussehen. Für die nach der x-Koordinate sortierten Residuen $r_1, \ldots, r_n$ betrachten die Autoren alle kontinuierlichen Teilsequenzen $r_i, \ldots, r_j$ mit $i \leq j$ und testen, ob alle die Bedingung

$$\frac{(r_i + \cdots + r_j)^2}{j - i + 1} \leq \sigma \sqrt{\tau \log n}$$

erfüllen. Die Schranke besteht aus der geschätzten Varianz σ, einem τ, das für die Normalverteilung als 2.2 zu wählen ist, und dem Logarithmus aus der Anzahl der Residuen.

Um zu entscheiden, ob alle Bedingungen eingehalten werden, genügt es, die kontinuierliche Teilsequenz mit maximalem Funktionswert zu finden und mit der Schranke zu vergleichen. Wir definieren das zugehörige Problem wie folgt:

Definition 20 (Multiresolutions-Problem)

`Eingabe:`
n reelle Zahlen $r_1, \ldots, r_n$.
`Problem:`
Finde zwei Indizes i und j mit $i \leq j$, so dass $\dfrac{(r_i + \cdots + r_j)^2}{j - i + 1}$ maximal ist.

Problemtransformation

Für die Problemtransformation benötigen wir konvexe Hüllen, die wir zunächst definieren, siehe auch Klein (2004). Eine Teilmenge M des $\mathbb{R}^2$ ist *konvex*, falls für alle Punkte $p, q \in M$ gilt, dass das Liniensegment $\{\alpha p + (1 - \alpha)q \mid 0 \leq \alpha \leq 1\}$ in M enthalten ist. Die *konvexe Hülle* einer Menge M' ist die kleinste konvexe Menge, die M' enthält. Da wir hier endliche Punktmengen betrachten, kann der Rand der konvexen

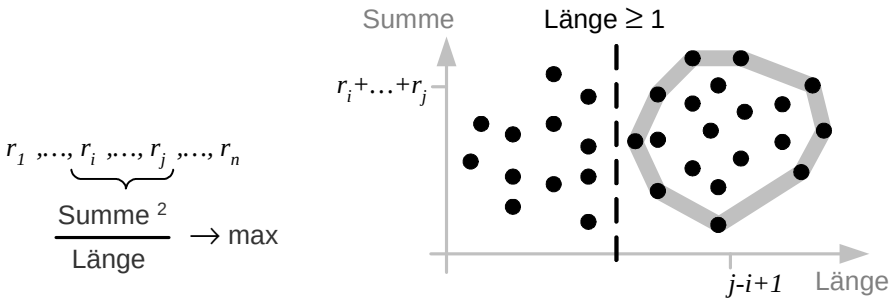


Abbildung 6: In der linken Hälfte ist das Multiresolutions-Problem dargestellt. Jede kontinuierliche Teilsequenz wird auf einen Punkt abgebildet. Nur der Rand der konvexen Hülle, der die quadratisch vielen Punkte der Minkowski-Summe umschließt, kann den maximalen Punkt enthalten. Auf dem Rand der konvexen Hülle liegen maximal $2n$ Punkte.

Hülle durch ein Polygon beschrieben werden, also eine geschlossene Abfolge von Liniensegmenten. Wir transformieren das Problem. Die Transformation ist in Abbildung 6 veranschaulicht und im Folgenden beschrieben:

Definition 21 (Multiresolutions$^{\mathcal{T}}$-Problem)

`Eingabe:`
n reelle Zahlen $r_1, \ldots, r_n$.

`Transformation:`
Jedes Präfix $r_1, \ldots, r_i$ wird auf die Punkte

$$p_i = (-i+1, -r_1 - \cdots - r_{i-1}) \text{ und } q_i = (i, r_1 + \cdots + r_i)$$

abgebildet. Wir erhalten die Punktmengen $\mathcal{P} = \{p_1, \ldots, p_n\}$ und $\mathcal{Q} = \{q_1, \ldots, q_n\}$. Die Minkowski-Summe der beiden Mengen ist

$$\mathcal{P} \oplus \mathcal{Q} = \{p + q \mid p \in \mathcal{P} \text{ und } q \in \mathcal{Q}\} \ .$$

Wie wir im Beweis von Lemma 24 zeigen werden, enthält die Minkowski-Summe eine Beschreibung aller kontinuierlichen Teilsequenzen, allerdings gibt es jetzt Sequenzen mit negativer Länge. Daher fügen wir eine passende Nebenbedingung hinzu und definieren die Menge

$$M = \{(\ell, s) \mid (\ell, s) \in \mathcal{P} \oplus \mathcal{Q} \text{ und } \ell \geq 1\} \ .$$

`Problem:`
Berechne die Extrempunkte $\mathcal{C}$ der konvexen Hülle von M.

Rücktransformation Gegeben ist die Funktion $f(\ell, s) = s^2/\ell$ und die konvexe Hülle $\mathcal{C}$, die Lösung des Multiresolutions$^{\mathcal{T}}$-Problems. Die Funktion f wird auf jeden Punkt der konvexen Hülle angewendet. Aus dem Punkt p_i+q_j mit maximalem Funktionswert ergeben sich die gesuchten Indizes i und j.

Die Rücktransformation nutzt aus, dass die Funktion f quasikonvex ist und das gesuchte Optimum auf dem Rand der konvexen Hülle liegt. Quasikonvexität ist wie folgt definiert – für Definition und das folgende Theorem siehe auch Boyd und Vandenberghe (2004, Seite 95 und 98):

Definition 22 (quasikonvex) *Eine Funktion $f : A \to \mathbb{R}$ ist quasikonvex, wenn A konvex ist und für jedes $\alpha \in \mathbb{R}$ die Sublevel-Menge $S_\alpha = \{p \in A \mid f(p) \leq \alpha\}$ konvex ist.*

Theorem 23 (Jensen-Ungleichung für quasikonvexe Funktionen)
Eine Funktion $f : A \to \mathbb{R}$ ist genau dann quasikonvex, wenn

- *A konvex ist und*
- *$\forall p, q \in A$ und $0 \leq \alpha \leq 1 : f(\alpha p + (1-\alpha)q) \leq \max\{f(p), f(q)\}$.*

Lemma 24 *In Zeit $O(n)$ kann aus der Lösung des Multiresolutions$^{\mathcal{T}}$-Problems eine Lösung für das Multiresolutions-Problem berechnet werden. Die Funktion $f(\ell, s)$ wird $2n$-mal ausgewertet.*

Beweis: Zunächst weisen wir nach, dass M einen Punkt mit optimalem Funktionswert enthält. Sei $r_i, \ldots, r_j$ eine optimale Lösung für das Multiresolutions-Problem. Die Funktionsauswertung des Punktes $p_i + q_j$ ergibt

$$\begin{aligned} f(p_i + q_j) =& f(j - i + 1, r_1 + \cdots + r_j - r_1 - \cdots - r_{i-1}) \\ =& f(j - i + 1, r_i + \cdots + r_j) \\ =& \frac{(r_i + \cdots + r_j)^2}{j - i + 1} \quad , \end{aligned}$$

welches dem Funktionswert der kontinuierlichen Teilsequenz $r_i, \ldots, r_j$ entspricht. Des Weiteren enthält die Menge M keinen Punkt mit einem größeren Funktionswert als dem des Punktes $p_i + q_j$. Falls M einen Punkt mit größerem Funktionswert enthalten würde, dann erhielten wir aus obiger Rechnung eine bessere Lösung als $r_i, \ldots, r_j$. Dies ist ein Widerspruch zur Optimalität von $r_i, \ldots, r_j$.

Das transformierte Problem liefert eine konvexe Hülle $\mathcal{C}$. Die Nebenbedingung $\ell \geq 1$ stellt sicher, dass M nur Punkte enthält, die gültige Lösungen sind. Daher folgt aus Definition der Quasikonvexität, dass der optimale Punkt auf der konvexen Hülle liegt. Wir zeigen, dass die Funktion $f(\ell, s) = s^2/\ell$ quasikonvex ist, indem wir nachweisen,

dass $|s| \leq \alpha\sqrt{\ell}$ eine konvexe Menge ist. Für $\alpha < 0$ ist die Menge leer und für $\alpha = 0$ erhalten wir eine Gerade. Für $\alpha > 0$ ist die Menge die Vereinigung von $\{(l,s) \mid s \leq \alpha\sqrt{\ell}\} \cup \{(l,s) \mid -s \geq -\alpha\sqrt{\ell}\}$. Beide Mengen sind konvex, da $\sqrt{\ell}$ eine konkave Funktion ist. Somit ist f quasikonvex.

Da die ℓ-Koordinate der Punkte aus M nur die ganzzahligen Werte $1, \ldots, n$ annimmt, enthält die konvexe Hülle maximal $2n$ Extrempunkte. Die Funktion wird also maximal $2n$ mal ausgewertet und eine Auswertung kostet Zeit $O(1)$, woraus sich die Rechenzeit $\mathcal{O}(n)$ ergibt. □

Um zu demonstrieren, dass verschiedene Algorithmenansätze zu verschiedenen Laufzeiten führen, wird häufig das Maximum-Subsequence-Problem (Bentley (1984)) verwendet. Wir können ein allgemeineres Problem mit $2n$ Funktionsaufrufen von f lösen:

Definition 25 (Generalized-Maximum-Subsequence-Problem)

`Eingabe:`
n reelle Zahlen $r_1, \ldots, r_n$ und eine quasikonvexe Funktion $f(\ell, s)$.
`Problem:`
Finde zwei Indizes i und j mit $i \leq j$, so dass

$$f(j - i + 1, r_i + \cdots + r_j)$$

maximal ist.

Minkowski-Algorithmus

Bevor wir das Multiresolutions$^{\mathcal{T}}$-Problem lösen, befassen wir uns zunächst mit der Minkowski-Summe ohne Nebenbedingung. Die Minkowski-Summe zweier Punktmengen $\mathcal{P}$ und $\mathcal{Q}$ ist definiert als

$$\mathcal{P} \oplus \mathcal{Q} = \{p + q \mid p \in \mathcal{P} \text{ und } q \in \mathcal{Q}\} \ .$$

Sie wird in der Literatur, siehe de Berg, van Kreveld, Overmars und Schwarzkopf (2000), verwendet für Mengen, die von beliebigen Polygonzügen umschlossen sind. Wir beschränken uns hier auf konvexe Mengen und das zugehörige Polygon, den Rand der konvexen Hülle. Alle Punkte, die auf dem Rand der konvexen Hülle von $\mathcal{P} \oplus \mathcal{Q}$ liegen, bezeichnen wir als *Minkowski-Hülle* von $\mathcal{P} \oplus \mathcal{Q}$. Toussaint (1983) beschreibt einen Algorithmus, der die Minkowski-Hülle in Zeit $\mathcal{O}(|\mathcal{P}| + |\mathcal{Q}|)$ berechnet, auch zu finden in de Berg, van Kreveld, Overmars und Schwarzkopf (2000). Der Algorithmus

ist in Abbildung 7 beschrieben. Allerdings müssen zunächst die konvexen Hüllen von $\mathcal{P}$ und $\mathcal{Q}$ berechnet werden. Da es leicht möglich ist, bei der Transformation die Punkte in einer nach der ℓ-Koordinate sortierten Reihenfolge zu erzeugen, können in unserer Anwendung die konvexen Hüllen von $\mathcal{P}$ und $\mathcal{Q}$ in linearer Zeit berechnet werden, siehe Klein (2004, Theorem 4.7).

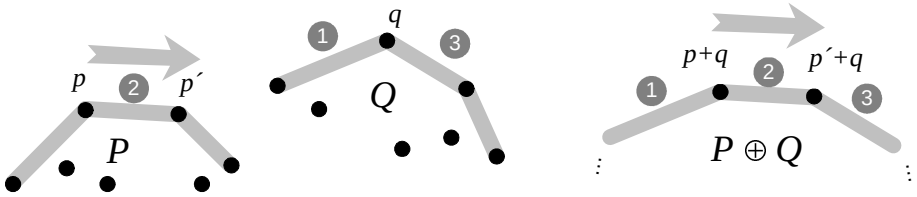


Abbildung 7: Minkowski-Algorithmus: Die Linien-Segmente der konvexen Hüllen von $\mathcal{P}$ und $\mathcal{Q}$ sind bezüglich ihrer Steigung sortiert. Im Wesentlichen mischt der Algorithmus die Segmente der beiden konvexen Hüllen. Ein Einzelschritt ist hier dargestellt. Der Algorithmus betrachtet die Punkte p und q und vergleicht die beiden nachfolgenden Segmente. Das Segment (p, p') hat die größere Steigung und daher wird die konvexe Hülle von $\mathcal{P} \oplus \mathcal{Q}$ um den Punkt $p' + q$ ergänzt.

Algorithmus für das Multiresolutions$^{\mathcal{T}}$-Problem

Das neue Resultat, das wir jetzt vorstellen, basiert darauf, die Mengen $\mathcal{P}$ und $\mathcal{Q}$ geeignet zu unterteilen. Wir betrachten den Punkt $p = (p_x, p_y) \in \mathcal{P}$ und den Punkt $q = (q_x, q_y) \in \mathcal{Q}$. Beide Punkte seien so gewählt, dass die Summe der beiden Punkte $p + q = (\ell, s) = (p_x + q_x, p_y + q_y)$ die Nebenbedingung $\ell \geq 1$ erfüllt; daher gilt: $p_x + q_x \geq 1$. Wir können nun ein γ wählen, z.B. $\gamma = (p_x - q_x)/2$, so dass gilt:

$$p_x \geq 1/2 + \gamma \text{ und } q_x \geq 1/2 - \gamma \ .$$

Das γ ist eine Verschiebung der Nebenbedingung. Jede Verschiebung greift andere Punkte p und q ab, von denen wir dann wissen, dass die Summe $p + q$ die Nebenbedingung erfüllt. Der Sachverhalt ist in Abbildung 8 dargestellt. Wir werden gleich einen Wert für γ festlegen. Zunächst definieren wir einige Mengen. Die Verschiebung γ unterteilt die Mengen $\mathcal{P}$ und $\mathcal{Q}$ in vier Teilmengen:

$$\begin{aligned}
\mathcal{P}_\gamma &= \{(p_x, p_y) \in \mathcal{P} \mid p_x \geq 1/2 + \gamma\} \\
\mathcal{Q}_\gamma &= \{(q_x, q_y) \in \mathcal{Q} \mid q_x \geq 1/2 - \gamma\} \\
\overline{\mathcal{P}_\gamma} &= \{(p_x, p_y) \in \mathcal{P} \mid p_x < 1/2 + \gamma\} \\
\overline{\mathcal{Q}_\gamma} &= \{(q_x, q_y) \in \mathcal{Q} \mid q_x < 1/2 - \gamma\}
\end{aligned}$$

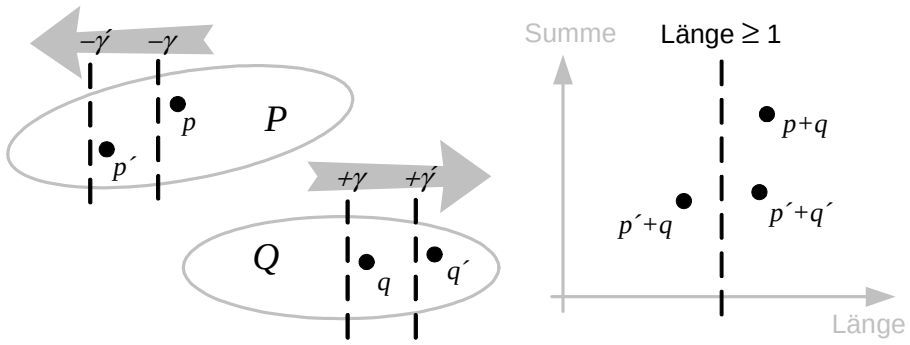


Abbildung 8: Auf der linken Seite sind die beiden Punktmengen $\mathcal{P}$ und $\mathcal{Q}$ abgebildet. Aus der Verschiebung γ ergeben sich zwei Nebenbedingungen, als gestrichelte Linien dargestellt. Die Punkte p und q erfüllen die um γ verschobene Nebenbedingung. Daher erfüllt auch deren Summe $p+q$ die Nebenbedingung. Wenn wir den Wert von γ auf γ' vergrößern, verschieben sich die beiden Nebenbedingungen in entgegengesetzte Richtungen. Der Punkt q verletzt nun die Nebenbedingung, der Punkt p' kommt neu hinzu. Daher verletzt $p' + q$ die Nebenbedingung, wohingegen $p' + q'$ die Nebenbedingung erfüllt.

Die Mengen $\mathcal{P}$ und $\mathcal{Q}$ enthalten jeweils n Punkte. Wenn γ von $-\infty$ bis ∞ läuft, so verkleinert sich $\mathcal{P}_\gamma$ und vergrößert sich $\mathcal{Q}_\gamma$. Wenn wir allgemeine Lage annehmen, können wir γ so wählen, dass

$$|\mathcal{P}_\gamma| + |\overline{\mathcal{Q}_\gamma}| = n \ , \quad \text{und} \quad |\overline{\mathcal{P}_\gamma}| + |\mathcal{Q}_\gamma| = n \ .$$

Den Fall, dass die Punkte nicht in allgemeiner Lage sind, werden wir später behandeln.

Um ein geeignetes γ zu finden, berechnen wir für die Punkte $p_1, \ldots, p_n$ die Werte $\gamma_1, \ldots, \gamma_n$, so dass $p_i = 1/2 + \gamma_i$ ist. Für $q_1, \ldots, q_n$ erhalten wir analog $\gamma'_1, \ldots, \gamma'_n$. Wir berechnen mit einem deterministischen Median-Algorithmus in linearer Zeit ein medianes γ aus $\gamma_1, \ldots, \gamma_n, \gamma'_1, \ldots, \gamma'_n$. Mit diesem γ erhalten wir Mengen $\mathcal{P}_\gamma, \overline{\mathcal{Q}_\gamma}, \overline{\mathcal{P}_\gamma}$ und $\mathcal{Q}_\gamma$, die obige Größenvorgaben erfüllen. Aus diesen Mengen ergeben sich vier Minkowski-Summen, die wir jetzt nacheinander betrachten werden.

Für die Punkte der Summe $\mathcal{P}_\gamma \oplus \mathcal{Q}_\gamma$ gilt, dass alle die Nebenbedingung erfüllen. Wir können daher den Minkowski-Algorithmus anwenden und erhalten in linearer Zeit eine konvexe Hülle $\mathcal{C}_1$. Für die Punkte der Summe $\overline{\mathcal{P}_\gamma} \oplus \overline{\mathcal{Q}_\gamma}$ gilt, dass alle Punkte die Nebenbedingung verletzen, daher kann diese Minkowski-Summe ignoriert werden. Die beiden übrigen Minkowski-Summen $\mathcal{P}_\gamma \oplus \overline{\mathcal{Q}_\gamma}$ und $\overline{\mathcal{P}_\gamma} \oplus \mathcal{Q}_\gamma$ enthalten im Allgemeinen sowohl Punkte $p+q$, die die Nebenbedingung erfüllen als auch Punkte, die sie verletzen. Es liegen also zwei Teilprobleme vor, die wir rekursiv lösen. Die rekursiven Aufrufe liefern zwei konvexe Hüllen $\mathcal{C}_2$ und $\mathcal{C}_3$.

Die drei konvexen Hüllen $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3$ fügen wir zusammen, indem wir die konvexen Hüllen simultan von links nach rechts durchlaufen und für jede der n möglichen x-Koordinaten die beiden extremen Punkte auswählen. Wir erhalten in linearer Zeit eine konvexe Hülle.

Wir schätzen nun die Rechenzeit der Rekursion ab. Die Mengen $\mathcal{P}_\gamma$ und $\mathcal{Q}_\gamma$ haben zusammen maximal $2n$ Punkte, so dass für die Berechnung der Minkowski-Hülle $\mathcal{O}(n)$ Zeit genügt. Wir führen 2 rekursive Aufrufe durch, die jeweils in der Summe maximal n Punkte haben. Wir bezeichnen mit $A(2n)$ die Rechenzeit des Algorithmus. Als Rekursionsgleichung erhalten wir

$$A(2n) = 2 \cdot A(n) + c \cdot n \ .$$

Offensichtlich ergibt sich die Rechenzeit $\mathcal{O}(n \log n)$.

Finden einer Aufteilung, falls keine allgemeine Lage vorliegt

Da in unserer Anwendung jeweils 2 Punkte die gleiche x-Koordinate haben, liegt keine allgemeine Lage vor. Das Problem ist, dass die folgenden Mengen mehr als einen Punkt enthalten können:

$$\mathcal{P}_\gamma^{\text{eq}} = \{(p_x, p_y) \in \mathcal{P} \mid p_x = 1/2 + \gamma\}$$
$$\mathcal{Q}_\gamma^{\text{eq}} = \{(q_x, q_y) \in \mathcal{Q} \mid q_x = 1/2 - \gamma\}$$

Die Idee ist, diese Punkte geeignet umzuverteilen. Wir inspizieren den Algorithmus, der Quickselect verwendet, um ein medianes, genauer ein n-tes Element γ zu bestimmen: Er findet auch im degenerierten Fall eine zulässige Aufteilung und wir erhalten aus dem Algorithmus folgende Mengen:

$$\mathcal{P}'_\gamma = \{p_i \in \mathcal{P} \mid \gamma_i \text{ rechts von } \gamma\}$$
$$\mathcal{Q}'_\gamma = \{q_i \in \mathcal{Q} \mid \gamma'_i \text{ links von oder gleich } \gamma\}$$
$$\overline{\mathcal{P}'_\gamma} = \{p_i \in \mathcal{P} \mid \gamma_i \text{ links von oder gleich } \gamma\}$$
$$\overline{\mathcal{Q}'_\gamma} = \{q_i \in \mathcal{Q} \mid \gamma'_i \text{ rechts von } \gamma\}$$

Zulässig heißt, dass die Größenvorgaben $|\mathcal{P}'_\gamma| + |\overline{\mathcal{Q}'_\gamma}| = n$ und $|\overline{\mathcal{P}'_\gamma}| + |\mathcal{Q}'_\gamma| = n$ eingehalten werden. Der Algorithmus wird so verändert, dass wir nun die Minkowski-Summen $(\mathcal{P}'_\gamma \cup \mathcal{P}_\gamma^{\text{eq}}) \oplus (\mathcal{Q}'_\gamma \cup \mathcal{Q}_\gamma^{\text{eq}})$ berechnen und die folgenden zwei rekursiven Probleme lösen: $\mathcal{P}'_\gamma \oplus \overline{\mathcal{Q}'_\gamma}$ und $\overline{\mathcal{P}'_\gamma} \oplus \mathcal{Q}'_\gamma$. Die restlichen Minkowski-Summen $\mathcal{P}_\gamma^{\text{eq}} \oplus \mathcal{Q}'_\gamma$, $\mathcal{P}'_\gamma \oplus \mathcal{Q}_\gamma^{\text{eq}}$ und $\overline{\mathcal{P}'_\gamma} \oplus \overline{\mathcal{Q}'_\gamma}$ brauchen nicht berücksichtigt zu werden, da alle enthaltenen Punkte die Nebenbedingung verletzen. Mit Lemma 24 von Seite 31 erhalten wir:

Theorem 26 *Das Multiresolutions-Problem kann in Zeit $\mathcal{O}(n \log n)$ gelöst werden.*

Verallgemeinerungen

Das Problem lässt sich in drei Punkten verallgemeinern:

- Die Funktion $f(\ell, s) = s^2/\ell$ kann durch eine beliebige quasikonvexe Funktion $f(x) : \mathbb{R}^2 \to \mathbb{R}$ ersetzt werden.
- Durch eine Drehung und Verschiebung der Punktmenge lässt sich jede Nebenbedingung $a \cdot x^\top \geq 1$ in die Bedingung $\ell \geq 1$ transformieren.
- Wir fordern nicht mehr, dass die ℓ-Koordinate nur die Werte $1, \ldots, n$ annimmt.

Das verallgemeinerte Problem sieht wie folgt aus:

Definition 27 (Minkowski-Summe mit einer Nebenbedingung)

`Eingabe:`
$2n$ Punkte $\mathcal{P} = \{p_1, \ldots, p_n\}$ und $\mathcal{Q} = \{q_1, \ldots, q_n\}$,
eine quasikonvexe Funktion $f(x) : \mathbb{R}^2 \to \mathbb{R}$ und
der Parametervektor $a \in \mathbb{R}^2$ sowie $b \in \mathbb{R}$ für die Nebenbedingung $a \cdot x^\top \geq b$.
`Problem:`
Für die *bedingte Minkowski-Summe*

$$(\mathcal{P} \oplus \mathcal{Q})_{a \cdot x^\top \geq b} = \{x \mid x \in \mathcal{P} \oplus \mathcal{Q} \text{ und } a \cdot x^\top \geq b\}$$

finde zwei Indizes i und j, so dass

$$(p_i + q_j) \in (\mathcal{P} \oplus \mathcal{Q})_{a \cdot x^\top \geq b} \text{ ist} \quad \text{und} \quad f(p_i + q_j) \text{ maximal ist.}$$

Der vorgestellte Algorithmus kann fast problemlos auf das verallgemeinerte Problem adaptiert werden. Für ein $\gamma \in \mathbb{R}$ sind die Mengen $\mathcal{P}_\gamma$,$\mathcal{Q}_\gamma$,$\overline{\mathcal{P}_\gamma}$ und $\overline{\mathcal{Q}_\gamma}$ jetzt definiert bezüglich der allgemeineren Nebenbedingung $a \cdot x^\top \geq b$. In der neuen Schreibweise wird die bedingte Minkowski-Summe vom Algorithmus aufgespalten in:

$$(\mathcal{P} \oplus \mathcal{Q})_{a \cdot x^\top \geq b} = \left(\overline{\mathcal{P}_\gamma} \oplus \mathcal{Q}_\gamma\right)_{a \cdot x^\top \geq b} \cup \left(\mathcal{P}_\gamma \oplus \overline{\mathcal{Q}_\gamma}\right)_{a \cdot x^\top \geq b} \cup \left(\mathcal{P}_\gamma \oplus \mathcal{Q}_\gamma\right)$$

Da die ℓ-Koordinaten nun beliebige Werte annehmen, könnte es passieren, dass zu viele Punkte auf der Minkowski-Hülle liegen und bei der Vereinigung der konvexen Hüllen extra Rechenzeit verursachen würden. Das folgende, neue Theorem beweist, dass dies kein Problem darstellt:

Theorem 28 *Die Minkowski-Hülle H^* von $(\mathcal{P} \oplus \mathcal{Q})_{a \cdot x^\top \geq b}$ enthält maximal* $\min\{2|\mathcal{P}| + |\mathcal{Q}|, |\mathcal{P}| + 2|\mathcal{Q}|\}$ *Punkte.*

Beweis: Der Beweis basiert auf der Idee, dass eine Minkowski-Hülle bezeugt, dass Paare $p_i + q_j$, die nicht vom Minkowski-Algorithmus gewählt wurden, innerhalb der Hülle H^* liegen. Wir wählen bestimmte Teilmengen von $\mathcal{P}$ und $\mathcal{Q}$ und betrachten die zugehörigen Minkowski-Hüllen. Ein Paar $p_i + q_j$ kann in einigen Minkowski-Hüllen auf dem Rand liegen oder im Innern. Falls es mindestens eine Minkowski-Hülle gibt, die das Paar $p_i + q_j$ im Innern enthält, ist das ein Zeuge dafür, dass $p_i + q_j$ nicht in H^* enthalten ist.

Wir sortieren die Punkte $p_1, \ldots, p_n$ bezüglich der Nebenbedingung, so dass $a \cdot p_1^\top \leq \cdots \leq a \cdot p_n^\top$. Wir betrachten die Mengen $\mathcal{P}_k = \{p_k, \ldots, p_n\}$ für $k = 1, \ldots, n$ und wählen die Menge

$$\mathcal{Q}_k = \{q \in \mathcal{Q} \mid p + q \text{ erfüllt die Nebenbedingung für alle } p \in \mathcal{P}_k\} \ .$$

Offensichtlich ist jedes Paar $p_i + q_j$, das die Nebenbedingung erfüllt, in mindestens einer der Minkowski-Summen $\mathcal{P}_1 \oplus \mathcal{Q}_1, \ldots, \mathcal{P}_n \oplus \mathcal{Q}_n$ berücksichtigt und die konvexe Hülle der Vereinigung aller Minkowski-Hüllen ergibt H^*. Wegen der Sortierung gilt, dass $\mathcal{P}_k \supseteq \mathcal{P}_{k+1}$.

O.B.d.A. nehmen wir an, dass der Minkowski-Algorithmus, angewendet auf $\mathcal{P}_k \oplus \mathcal{Q}_k$, den Punkt p_k mit den Punkten $q_1, \ldots, q_\tau$ kombiniert. Um die Paare zu zählen, weisen wir jedem Paar Kosten 1 zu. Die Kosten von $p_k + q_1$ und $p_k + q_\tau$ werden auf den Punkt p_k umgelegt, die Kosten von $p_k + q_2, \ldots, p_k + q_{\tau-1}$ werden jeweils auf $q_2, \ldots, q_{\tau-1}$ umgelegt. Da der Algorithmus die restlichen Paare $p_k + q_{\ldots}$ nicht gewählt hat, liegen diese im Innern und verursachen keine Kosten. Die Kostenzählung ist vollständig, da $\mathcal{P}_k \oplus \mathcal{Q}_k$ alle erlaubten Punkte $p_k + q_{\ldots}$ enthält. Die Umlegung der Kosten führen wir für alle Punkte $p_1, \ldots, p_n$ durch.

Jedem Punkt p_k werden maximal Kosten 2 zugewiesen. Für jeden Punkt aus $\mathcal{Q}$ wollen wir eine Kostenschranke von 1 zeigen, sie erhalten jedoch unter Umständen mehr als Kosten 1 zugewiesen. Wir betrachten ein $q^* \in \mathcal{Q}$, dem mindestens Kosten 2 zugewiesen wurden. Wir wählen von allen Paaren $p_{\ldots} + q^*$, deren Kosten auf q^* umgelegt wurden, das Paar $p_k + q^*$ mit dem kleinsten Index k. Da bei der Betrachtung der Menge $\mathcal{P}_k \oplus \mathcal{Q}_k$ die Kosten des Paares $p_k + q^*$ auf q^* umgelegt wurden, wissen wir, dass der Minkowski-Algorithmus, angewendet auf $\mathcal{P}_k \oplus \mathcal{Q}_k$, kein weiteres Paar $p_{\ldots} + q^*$ gewählt hat. Da $\mathcal{P}_k \supseteq \mathcal{P}_{k+1}$ ist, folgt, dass alle $p_{\ldots} + q^*$ außer $p_k + q^*$ im Innern der Minkowski-Hülle von $\mathcal{P}_k \oplus \mathcal{Q}_k$ liegen und diese Kosten daher verworfen werden können. Somit verursacht jedes $q^* \in \mathcal{Q}$ maximal Kosten 1 und es ergeben sich die Gesamtkosten $2|\mathcal{P}| + |\mathcal{Q}|$. Da wir die Rollen von $\mathcal{P}$ und $\mathcal{Q}$ vertauschen können, erhalten wir die gewünschte Aussage. □

Beliebig viele Nebenbedingungen

Wie arbeitet der Algorithmus zur Berechnung der bedingten Minkowski-Summe? Im Wesentlichen spaltet er eine bedingte Minkowski-Summe auf in zwei bedingte Minkowski-Summen der halben Größe und einer Minkowski-Summe ohne Bedingung. Anders ausgedrückt, er kann eine Bedingung entfernen. Wenn wir nun von vornherein mehrere Bedingungen haben, so können wir diese rekursiv bearbeiten. Wir definieren zunächst das allgemeinere Problem:

Definition 29 (Minkowski-Summe mit κ Nebenbedingungen)

`Eingabe:`
$2n$ Punkte $\mathcal{P} = \{p_1, \ldots, p_n\}$ und $\mathcal{Q} = \{q_1, \ldots, q_n\}$,
eine quasikonvexe Funktion $f(x)$ mit $x = (x_1, x_2)$ mit $x_1, x_2 \in \mathbb{R}$ und
eine $(\kappa \times 2)$-Matrix A sowie
ein Zeilenvektor $b \in \mathbb{R}^\kappa$ für die Nebenbedingung $A \cdot x^\top \geq b$.

`Problem:`
Für die *bedingte Minkowski-Summe*

$$(\mathcal{P} \oplus \mathcal{Q})_{A \cdot x^\top \geq b} = \{x \mid x \in \mathcal{P} \oplus \mathcal{Q} \text{ und } A \cdot x^\top \geq b\}$$

finde zwei Indizes i und j, so dass

$$(p_i + q_j) \in (\mathcal{P} \oplus \mathcal{Q})_{A \cdot x^\top \geq b} \text{ ist und } f(p_i + q_j) \text{ maximal ist.}$$

Wir lösen das Problem rekursiv. Sei MWSKI_1 der Algorithmus, der die Hülle der Minkowski-Summe mit einer Nebenbedingung berechnet. Wir nehmen an, dass wir einen Algorithmus $\text{MWSKI}_{\kappa-1}$ haben, der die Hülle der Minkowski-Summe mit $\kappa - 1$ Nebenbedingungen berechnet.

Sei $a \cdot x^\top \geq b$ eine der gegebenen Nebenbedingungen und seien $A' \cdot x^\top \geq b'$ die restlichen $\kappa - 1$ Nebenbedingungen. Die Mengen $\mathcal{P}_\gamma$, $\mathcal{Q}_\gamma$, $\overline{\mathcal{P}_\gamma}$ und $\overline{\mathcal{Q}_\gamma}$ sind definiert wie zuvor und wir wählen wieder ein γ, so dass $|\mathcal{P}_\gamma| + |\overline{\mathcal{Q}_\gamma}| = n$ und $|\overline{\mathcal{P}_\gamma}| + |\mathcal{Q}_\gamma| = n$. (Unter der Annahme der allgemeinen Lage). Wir stellen fest, dass alle Punkte aus $\left(\overline{\mathcal{P}_\gamma} \oplus \overline{\mathcal{Q}_\gamma}\right)_{A' \cdot x^\top \geq b'}$ die Bedingung verletzen, diese Minkowski-Summe kann also ignoriert werden. Andererseits erfüllen alle Punkte aus $(\mathcal{P}_\gamma \oplus \mathcal{Q}_\gamma)_{A' \cdot x^\top \geq b'}$ die Nebenbedingung $a \cdot x^\top \geq b$, aber nicht notwendigerweise die restlichen Nebenbedingungen $A' \cdot x^\top \geq b'$. Wir beobachten, dass folgende Gleichung gilt:

$$(\mathcal{P} \oplus \mathcal{Q})_{A \cdot x^\top \geq b} = \left(\overline{\mathcal{P}_\gamma} \oplus \mathcal{Q}_\gamma\right)_{A \cdot x^\top \geq b} \cup \left(\mathcal{P}_\gamma \oplus \overline{\mathcal{Q}_\gamma}\right)_{A \cdot x^\top \geq b} \cup (\mathcal{P}_\gamma \oplus \mathcal{Q}_\gamma)_{A' \cdot x^\top \geq b'}$$

Die Punktmengen der ersten beiden Summen sind nur halb so groß und wir berechnen die Minkowski-Hüllen rekursiv mit $\textsc{Mwski}_\kappa$, wohingegen wir die Minkowski-Hüllen des letzten Terms mit $\textsc{Mwski}_{\kappa-1}$ berechnen. Die Rekursionsenden sind zum einen, dass wir eine Minkowski-Summe ohne Bedingung erhalten, deren Hülle wir berechnen, und zum anderen, dass eine der Mengen $\mathcal{P}_\gamma$,$\mathcal{Q}_\gamma$,$\overline{\mathcal{P}_\gamma}$ oder $\overline{\mathcal{Q}_\gamma}$ leer ist. Von den Aufrufen erhalten wir konvexe Hüllen $\mathcal{C}_1$, $\mathcal{C}_2$ und $\mathcal{C}_3$, die wir, wie zuvor, mischen.

Sei $T_1(2n) = \mathcal{O}(n \log n)$ die Rechenzeit von $\textsc{Mwski}_1$. Wir nehmen an, dass $T_{\kappa-1}(2n) = \mathcal{O}\big(n \log^{\kappa-1} n\big)$ die Rechenzeit für $\textsc{Mwski}_{\kappa-1}$ ist. Es folgt, dass die Größen der konvexen Hüllen $\mathcal{C}_1$, $\mathcal{C}_2$ und $\mathcal{C}_3$ ebenfalls durch $\mathcal{O}\big(n \log^{\kappa-1} n\big)$ beschränkt sind, so dass wir die Rekursionsgleichung

$$\begin{aligned} T_\kappa(2n) &= 2T_\kappa(n) + T_{\kappa-1}(2n) \\ &= 2T_\kappa(n) + \mathcal{O}\big(n \log^{\kappa-1} n\big) \end{aligned}$$

erhalten. Es folgt:

Theorem 30 *Das Problem „Minkowski-Summe mit κ Nebenbedingungen" für konstantes κ kann in Zeit $\mathcal{O}(n \log^\kappa n)$ gelöst werden.*

Alle erlaubten Punkte aufzählen

Bisher waren wir an der konvexen Hülle bzw. an der Maximierung einer Funktion f interessiert. Stattdessen können wir den Standpunkt einnehmen, dass der Algorithmus die bedingte Minkowski-Summe

$$(\mathcal{P} \oplus \mathcal{Q})_{A \cdot x^\top \geq b} = (\mathcal{P}_{\gamma_1} \oplus \mathcal{Q}_{\gamma_1}) \ \cup \ (\mathcal{P}_{\gamma_2} \oplus \mathcal{Q}_{\gamma_2}) \ \cup \ (\mathcal{P}_{\gamma_3} \oplus \mathcal{Q}_{\gamma_3}) \ \cup \ \dots$$

in mehrere unbedingte Minkowski-Summen zerlegt für $\gamma_1, \gamma_2, \gamma_3, \dots$. Jetzt haben wir zusätzlich die Möglichkeit, in Zeit $\mathcal{O}(1)$ pro Punkt alle Punkte auszugeben, die die Nebenbedingung erfüllen (zuzüglich der Zeit, um die Zerlegung zu berechnen). Dies wird im nächsten Abschnitt nützlich sein, um den Algorithmus auf die letzten beiden Bioinformatikprobleme anzuwenden.

Anwendungsbeispiele aus der Bioinformatik

In diesem Abschnitt beschreiben wir einige Probleme, die sich als Minkowski-Summe mit mehreren Nebenbedingungen modellieren lassen.

- Goldwasser, Kao und Lu (2005) befassen sich mit dem Maximum-Density-Segment-Problem. Gegeben sind Paare (r_i, w_i) mit $i = 1, \ldots, n$ und $w_i > 0$ sowie zwei Grenzen L und U. Finde ein i und j, so dass $(r_i + \cdots + r_j)/(w_i + \cdots + w_j)$ maximal ist unter den zwei Nebenbedingungen $L \leq w_i + \cdots + w_j \leq U$. Dieses Problem können wir mit dem Minkowski-Problem wie folgt modellieren:

$$\begin{aligned} p_i &= (-r_1 - \cdots - r_{i-1}, -w_1 - \cdots - w_{i-1}) \\ q_j &= (r_1 + \cdots + r_j, w_1 + \cdots + w_j) \\ f(x) &= x_1/x_2 \\ (0,1) \cdot x^\top &\geq \max\{L, \min\{w_i\}\} \\ (0,1) \cdot x^\top &\leq U \end{aligned}$$

- Fan, Lee, Lu, Tsou, Wang und Yao (2003) behandeln das Maximum-Sum-Segment-Problem. Gegeben sind reelle Zahlen $r_1, \ldots, r_n$ und zwei Grenzen L und U mit $L \geq 1$. Finde ein i und j, so dass $r_i + \cdots + r_j$ maximal ist unter den Nebenbedingungen $L \leq j - i + 1 \leq U$. Dieses Problem können wir mit dem Minkowski-Problem wie folgt modellieren:

$$\begin{aligned} p_i &= (-r_1 - \cdots - r_{i-1}, -i + 1) \\ q_j &= (r_1 + \cdots + r_j, j) \\ f(x) &= x_1 \\ (0,1) \cdot x^\top &\geq L \\ (0,1) \cdot x^\top &\leq U \end{aligned}$$

- Allison (2003) befasst sich mit dem Non-Negative-Sum-Intervals-Problem. Gegeben sind reelle Zahlen $r_1, \ldots, r_n$. Finde i und j, so dass $j - i + 1$ maximal ist unter der Nebenbedingung $r_i + \cdots + r_j \geq 0$. Dieses Problem können wir mit dem Minkowski-Problem wie folgt modellieren:

$$\begin{aligned} p_i &= (-r_1 - \cdots - r_{i-1}, -i + 1) \\ q_j &= (r_1 + \cdots + r_j, j) \\ f(x) &= x_2 \\ (1,0) \cdot x^\top &\geq 0 \end{aligned}$$

- Lin, Jiang und Chao (2002) behandeln einerseits das folgende Problem: Gegeben sind reelle Zahlen $r_1, \ldots, r_n$ und eine obere Grenze $U \geq 1$. Finde i und j, so dass $r_i + \cdots + r_j$ maximal ist und die Länge $j - i + 1 \leq U$. Dieses Problem können wir mit dem Minkowski-Problem wie folgt modellieren:

$$\begin{aligned} p_i &= (-r_1 - \cdots - r_{i-1}, -i + 1) \\ q_j &= (r_1 + \cdots + r_j, j) \\ f(x) &= x_1 \\ (0,1) \cdot x^\top &\geq 1 \\ (0,1) \cdot x^\top &\leq U \end{aligned}$$

- Des Weiteren behandeln Lin, Jiang und Chao (2002) das folgende Problem: Gegeben sind reelle Zahlen $r_1, \ldots, r_n$ und eine untere Grenze L mit $L \geq 1$. Finde i und j, so dass $(r_i + \cdots + r_j)/(j - i + 1)$ maximal ist und die Länge $j - i + 1 \geq L$. Dieses Problem können wir mit dem Minkowski-Problem wie folgt modellieren:

$$\begin{aligned} p_i &= (-r_1 - \cdots - r_{i-1}, -i + 1) \\ q_j &= (r_1 + \cdots + r_j, j) \\ f(x) &= x_1 / x_2 \\ (0,1) \cdot x^\top &\geq L \end{aligned}$$

- Chen, Jaffe und Church (2001) befassen sich mit dem Subsequence-Sum-Problem: Gegeben sind zwei Sequenzen $a_1, \ldots, a_n$ und $b_1, \ldots, b_n$ und eine Zahl M. Zähle alle möglichen Teilsequenzen $a_i, \ldots, a_j$ und $b_k, \ldots, b_\ell$ auf, für die gilt, dass

$$\textstyle\sum_{s=i,\ldots,j} a_s + \sum_{t=k,\ldots,\ell} b_t = M \ .$$

  Dieses Problem können wir mit dem Minkowski-Problem wie folgt modellieren:

$$\begin{aligned} p_{i,j} &= (\textstyle\sum_{s=i,\ldots,j} a_s, 0) \\ q_{k,\ell} &= (\textstyle\sum_{t=k,\ldots,\ell} b_t, 0) \\ (1,0) \cdot x^\top &\geq M \end{aligned}$$

  Wir modifizieren den Algorithmus so, dass er keine konvexen Hüllen berechnet, sondern alle Punkte der unbedingten Minkowski-Summen ausgibt, die die Gleichung $(1,0) \cdot x^\top = M$ erfüllen. Wir erreichen die gleiche Rechenzeit $\mathcal{O}(n^2 \log n + \#\text{Lösungen})$ wie die Autoren.

- Bengtsson und Chen (2004) behandeln das folgende Problem: Gegeben sind reelle Zahlen $r_1, \ldots, r_n$ und eine ganze Zahl k mit $1 \leq k \leq \frac{1}{2}n(n-1)$. Zähle die k größten Summen $\sum_{\ell=i\ldots j} r_\ell$ mit $1 \leq i \leq j \leq n$ auf. Die Entscheidungsvariante, alle Summen größer α aufzuzählen, können wir mit dem Minkowski-Problem wie folgt modellieren:

$$\begin{aligned} p_i &= (-r_1 - \cdots - r_{i-1}, -i+1) \\ q_j &= (r_1 + \cdots + r_j, j) \\ (0,1) \cdot x^\top &\geq 1 \\ (1,0) \cdot x^\top &\geq \alpha \end{aligned}$$

  Wir modifizieren den Algorithmus, so dass er keine konvexe Hüllen berechnet, sondern alle Punkte der unbedingten Minkowski-Summen ausgibt. Es werden also alle kontinuierlichen Teilsequenzen $r_i, \ldots, r_j$ aufgezählt, deren Summe größer als α ist. Durch eine randomisierte binäre Suche, die einen $\log(n)$-Faktor Rechenzeit kostet, können wir einen Wert α bestimmen, so dass genau k Lösungen ausgegeben werden. Wir benötigen im Mittel Rechenzeit $\mathcal{O}\big(k + n \log^3 n\big)$.

4 Zeitreihen und Update robuster Schätzer

Ein Teilgebiet der Statistik beschäftigt sich mit der Analyse von Zeitreihen. Eine Charakteristik von Zeitreihen ist, dass zeitlich aufeinander folgende Datenpunkte in den meisten Fällen voneinander abhängig sind. Zeitreihen treten z.B. in der Intensivmedizin auf, wo Herzfrequenz, Blutdruck, Atemfrequenz und andere Vital-Parameter gemessen werden oder in den Wirtschaftswissenschaften, wo Aktienkurse oder das Wirtschaftswachstum betrachtet werden.

Die Daten sollen in systematische Komponenten und in irreguläre Komponenten zerlegt werden. Systematische Komponenten sind Saison, Konjunktur, Trend oder Sprünge. Auf Problematiken wie die Saisonbereinigung gehen wir nicht weiter ein, da bei medizinischen Daten nur Trends und Sprünge relevant sind. In Abbildung 9 ist ein Beispiel zu sehen, das mit dem LS-Schätzer geglättet wurde.

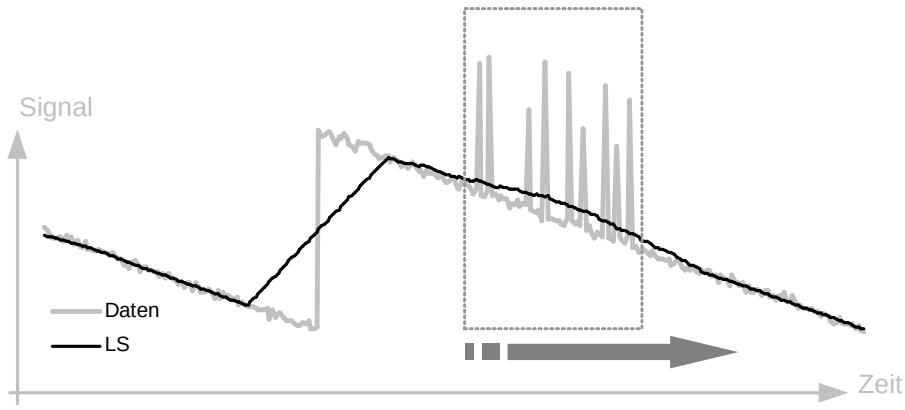


Abbildung 9: Die künstlichen Daten, grau dargestellt, enthalten einen Sprung und eine Stelle mit Ausreißern. Die Least-Squares-Schätzung, schwarz dargestellt, kann den Sprung nicht nachbilden und wird durch die Ausreißer nach oben verzogen.

Von Davies, Fried und Gather (2003) wird der Ansatz gewählt, die Daten mit lokaler Regression zu analysieren. Lokal bedeutet, dass ein Fenster einer bestimmten Breite

über die Daten geschoben wird. Für die Datenpunkte, die sich in dem Fenster befinden, wird eine Regressionsgerade geschätzt und der Schnittpunkt aus Regressionsgerade und der Mitte des Fensters ist der geschätzte *Level* des Fensters. Die Level aller Fenster bilden nun eine geglättete Zeitreihe.

Die Problematik, die sich auch hier wieder stellt, ist die, dass in den Daten Ausreißer vorhanden sind, die die Glättung beeinflussen, aber medizinisch irrelevant sind. Daher wurde der Schätzer durch robuste Schätzer ersetzt; zwei der Schätzer wollen wir hier nennen. Zum einen den MTM-Schätzer (Modified Trimmed Median), der auf dem MAD-Schätzer basiert und den RM-Schätzer (Repeated Median). Beide haben einen Bruchpunkt von 50% und die Ergebnisse sind in Abbildung 10 dargestellt.

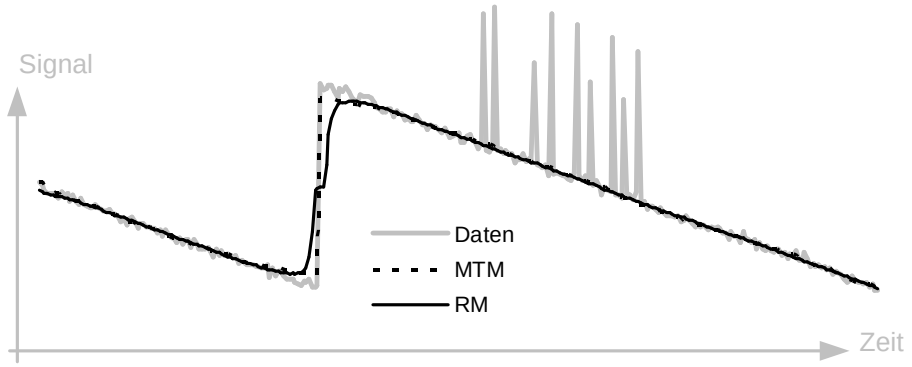


Abbildung 10: Der Repeated Median (schwarz) glättet den Sprung leicht und wird durch die Ausreißer nicht beeinflusst. Der Modified Trimmed Median (gestrichelt) kann den Sprung sehr gut nachbilden, schätzt aber keine Regressionsgerade, die für eine Vorhersage verwendet werden könnte.

Der Adhoc-Ansatz besteht natürlich darin, für jedes Datenfenster die Schätzer neu zu berechnen. Da zwei benachbarte Fenster sich aber nur um zwei Punkte unterscheiden, stellt sich hier natürlich die Aufgabe, Update-Algorithmen und Datenstrukturen zu entwerfen, die weniger Rechenzeit benötigen.

Wir stellen in Kapitel 4.1 einen Update-Algorithmus für den MAD-Schätzer vor mit Rechenzeit $\mathcal{O}(\log n)$ und skizzieren in Kapitel 4.1.1 den Update des MTM-Schätzers. Schließlich stellen wir in Kapitel 4.2 einen Update-Algorithmus für den RM-Schätzer vor mit Rechenzeit $\mathcal{O}(n)$.

4.1 Update des MAD-Schätzers

Der MTM-Schätzer, den wir auf Seite 52 genauer betrachten, benötigt einen Varianzschätzer. Wie in der Einleitung bereits erläutert, ist der robuste Schätzer für die Varianz der MAD-Schätzer. Das Problem ist wie folgt definiert:

Definition 31 (MAD-Problem (Median Absolute Deviation))

`Eingabe:`
n Zahlen $r_1, \ldots, r_n$ mit $r_i \in \mathbb{R}$.
`Problem:`
Berechne $\text{MAD} = \text{med}_i\, |r_i - \mu|$ mit $\mu = \text{med}_j(r_j)$.

Problemtransformation

Um einen Algorithmus zu erhalten, der updatefähig ist, transformieren wir zunächst das ursprüngliche Problem. Wir vereinfachen die Beschreibung, indem wir vorübergehend annehmen, dass alle r_i verschieden sind und der MAD ein ausgezeichnetes Element ist. Wir teilen die Werte $|r_i - \mu|$ auf zwei Arrays auf. Werte mit $r_i \leq \mu$ werden in dem Array Val^- platziert, Werte mit $r_i \geq \mu$ in Val^+. Bis auf μ wird jeder Wert genau einmal gespeichert. Für n ungerade hingegen wird der Median dupliziert, so dass beide Arrays eine gleiche Anzahl von Elementen enthalten. Die Werte in den beiden Arrays werden sortiert. Das kleinste Element befindet sich an der linken Position.

Eine Möglichkeit, den MAD zu berechnen, wäre, die beiden Arrays zu mergen und das Element mit dem Rang $\lceil n/2 \rceil + 1$ zu finden. (Die +1 resultiert aus dem duplizierten Median.) Den Rang eines Elementes x_i in dem gemergten Array bezeichnen wir als Merge-Rang und verwenden den Bezeichner $\text{rang}_M(x_i)$. Mit $\text{rang}(x_i)$ hingegen bezeichnen wir die Position des Elementes x_i in dem nichtgemergten Array, der x_i enthält, wobei das kleinste Element den Rang 1 erhält.

Statt zu mergen, halten wir die beiden Arrays jedoch getrennt. Die beiden Arrays sind in Abbildung 11 abgebildet. Lassen wir nun in der Abbildung die gestrichelte Linie nach links und rechts fallen, so trifft sie auf x_i und x_{i+1}. Wir schauen uns den Rang der Elemente näher an. Es gilt: $\text{rang}_M(\text{MAD}) = \lceil n/2 \rceil + 1$. Die Elemente, die mit $\text{rang}_M(\text{MAD})$ gezählt werden, verteilen sich auf beide Arrays und wir erhalten, dass

$$\text{rang}_M(\text{MAD}) = \text{rang}(x_i) + \text{rang}(\text{MAD}) = \lceil n/2 \rceil + 1 \ . \qquad \text{(Rang-Bedingung)}$$

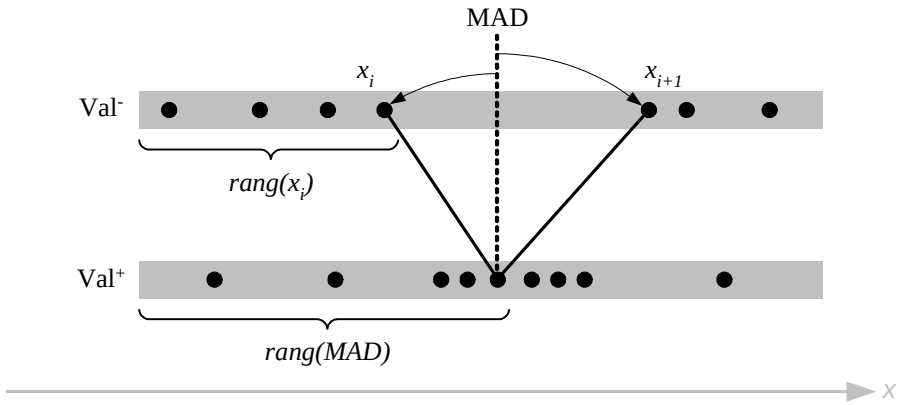


Abbildung 11: Die Eingabe wird in kleine Daten (Val^-) und große Daten (Val^+) aufgeteilt. Aus den Elementen x_i und x_{i+1} kann der MAD bestimmt werden.

Um den MAD zu bestimmen, genügt es also, ein x_i zu finden, das sich links vom MAD befindet und dessen Nachfolger x_{i+1} rechts vom MAD. Wir werden später eine Regel angeben, die ohne den MAD zu kennen, diese Eigenschaft feststellt. Das Element x_i nennen wir *co-MAD*. Die neue Sichtweise scheint zunächst das Problem zu komplizieren, bietet beim späteren Update aber entscheidende Vorteile. Wir fassen die neue Sichtweise in folgender Definition zusammen:

Definition 32 (co-MAD-Problem)

`Eingabe:`

n Zahlen $r_1, \ldots, r_n$ mit $r_i \in \mathbb{R}$ und $k = \lceil n/2 \rceil + 1$.

`Transformation:`

Jede Zahl r_i wird abgebildet auf $r'_i = |r_i - \mu|$, wobei $\mu = \text{med}_j\,(r_j)$. Die Werte werden in zwei Multimengen V^- und V^+ aufgeteilt,

$$V^- = \{r'_i \mid r_i \leq \mu\} \cup \{\infty\} \text{ und } V^+ = \{r'_i \mid r_i \geq \mu\} \cup \{\infty\} \ ,$$

so dass $|V^-| = k$ und $|V^+| = k$. Wir benennen die Elemente um und sortieren sie nach ihrem Wert aufsteigend, wobei der Vergleich die Indizes mit einbezieht, so dass die Sortierung eindeutig ist. Wir erhalten als transformierte Eingabe zwei sortierte Folgen

$$\text{Val}^- = (x_1, \ldots, x_k) \text{ und } \text{Val}^+ = (x_{k+1}, \ldots, x_{2k}) \ .$$

`Problem:`

Finde ein $x_i \in \text{Val}^- \cup \text{Val}^+$, $x_i \neq \infty$, so dass $\text{rang}_M(x_i) < k$ und $\text{rang}_M(x_{i+1}) > k$. Finde ein $x^* \in \text{Val}^- \cup \text{Val}^+$, so dass $\text{rang}(x^*) + \text{rang}(x_i) = k$ und x^* und x_i in verschiedenen Folgen liegen.

Lemma 33 *Das co-MAD-Problem ist wohldefiniert und es gilt:* $\text{rang}_M(x^*) = k$.

Beweis: Zuerst zeigen wir, dass immer ein x_i und x_{i+1} mit den geforderten Eigenschaften existieren. Das größte Element ist jeweils ∞, welches wir zusätzlich eingefügt haben und daher befindet sich mindestens ein Element rechts von x^*, so dass x_{i+1} existiert. Die beiden Elemente, aus denen der Median μ berechnet wird, werden für n gerade auf beide Mengen aufgeteilt und stellen die kleinsten Elemente dar. Für n ungerade wird μ dupliziert und stellt in beiden Folgen das kleinste Element dar. Beide Elemente können nicht der MAD sein und daher gibt es mindestens ein Element, das sich links von x^* befindet. Daher können wir immer ein x_i und x_{i+1} mit den geforderten Eigenschaften finden und das co-MAD-Problem ist wohldefiniert.

Als Nächstes zeigen wir, dass $\text{rang}_M(x^*) = k$. O.B.d.A. sei $x_i \in \text{Val}^+$. Wir können uns leicht überlegen, dass aus der Eigenschaft $\text{rang}(x^*) + \text{rang}(x_i) = k$ und den Eigenschaften von x_i und x_{i+1} folgt, dass $x_i < x^* < x_{i+1}$. Da die Elemente x_i und x_{i+1} benachbart sind und x^* und x_i in verschiedenen Folgen enthalten sind, folgt, dass $\text{rang}(x_i)$ die Elemente in Val^+ und $\text{rang}(x^*)$ die Elemente in Val^- zählt, die kleiner gleich x^* sind. Also hat x^* einen Merge-Rang von k. □

Theorem 34 *Aus x^* kann eine Lösung für das MAD-Problem berechnet werden.*

Beweis: Für n ungerade ist x^* das gesuchte MAD-Element. Für n gerade benötigen wir das Element x' mit einem Merge-Rang von $k-1$. Der MAD ist dann der Mittelwert aus x' und x^*. Als Kandidaten für x' kommen x_i und der linke Nachbar von x^* in Frage. Das Maximum aus beiden Elementen ist x'. □

Berechnung des co-MAD

Wir lassen die Update-Problematik zunächst außer Acht und stellen uns vor, dass die beiden sortierten Folgen Val^- und Val^+ in zwei Arrays gespeichert sind. Der Algorithmus arbeitet mit Intervallschachtelung auf den beiden Arrays. Die geschlossenen Intervalle $[L^-, R^-]$ und $[L^+, R^+]$ werden mit $[-\infty, +\infty]$ initialisiert. Die linken Grenzen L^- und L^+ sind zugleich Kandidaten für das gesuchte x_i. Aus beiden offenen Intervallen wählen wir mittlere Elemente $p \in]L^-, R^-[$ und $q \in]L^+, R^+[$. Später werden wir genauer erläutern, wie die Elemente gewählt werden.

Um zu bestimmen, wie die Intervalle verkleinert werden, vergleichen wir $\text{rang}(p) + \text{rang}(q)$ mit k. Der Fall, dass $\text{rang}(p) + \text{rang}(q) \leq k$ und $p < q$ gilt, ist in der linken Hälfte von Abbildung 12 dargestellt.

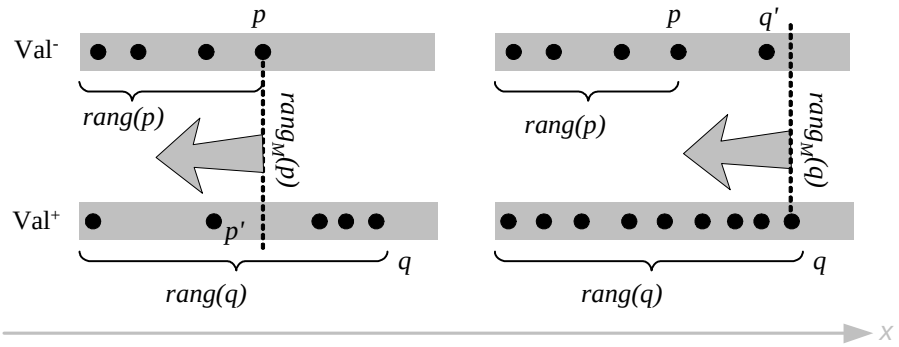


Abbildung 12: Sei $p < q$. Im linken Teil der Abbildung ist der Fall $\text{rang}(p)+\text{rang}(q) \leq k$ dargestellt. Daraus lässt sich eine obere Schranke für $\text{rang}_M(p)$ berechnen, d.h., die Anzahl der Elemente links der gestrichelten Linie ist kleiner als k. Im rechten Teil der Abbildung ist der Fall $\text{rang}(p) + \text{rang}(q) > k$ dargestellt. Es ergibt sich, dass sich mehr als k Elemente links der gestrichelten Linie befinden.

Der Merge-Rang ist die Anzahl der Elemente in beiden Arrays, die kleiner gleich p sind. Sei p' das größte Element aus dem anderen Array, das kleiner als p ist. Der Merge-Rang von p ergibt sich aus folgender Rechnung:

$$\begin{aligned} \text{rang}_M(p) &= \text{rang}(p) + \text{rang}(p') \\ &< \text{rang}(p) + \text{rang}(q) \qquad \mid\ p \neq q, \text{ da } p' < p < q \\ &\leq k\ . \end{aligned}$$

Somit ist $\text{rang}_M(p) < k$ und das Element p befindet sich in der gemergten Folge links vom MAD-Element. Alle Elemente links von p in Val^- und p selbst können ausgeschlossen werden. Daher setzen wir $L^- = p$. Für den Fall $p > q$ vertauschen die beiden Arrays ihre Rollen und wir setzen $L^+ = q$.

Der Fall, dass $\text{rang}(p) + \text{rang}(q) > k$ und $p < q$ gilt, ist in der rechten Hälfte von Abbildung 12 dargestellt. Der Merge-Rang von q ergibt sich aus folgender Rechnung:

$$\begin{aligned} \text{rang}_M(q) &= \text{rang}(q') + \text{rang}(q) \\ &\geq \text{rang}(p) + \text{rang}(q) \\ &> k\ . \end{aligned}$$

Somit ist $\text{rang}_M(q) > k$ und das Element q befindet sich in der gemergten Folge rechts vom MAD-Element. Alle Elemente rechts von q in Val^+ und q selbst können ausgeschlossen werden. Daher setzen wir $R^+ = q$. Für den Fall $p > q$ setzen wir $R^- = p$.

In jedem Schritt wird ein Intervall verkürzt, und zwar solange, bis eines der beiden offenen Intervalle $]L^-, R^-[$ oder $]L^+, R^+[$ keine Elemente mehr enthält. Das Element

mit Merge-Rang k wird niemals ausgeschlossen, daher ist die linke Grenze des leeren Intervalls das gesuchte co-MAD-Element x_i.

Update

Um den Algorithmus updatefähig zu machen, benötigen wir eine dynamische Datenstruktur für Val^- und Val^+. Wir verwenden zwei AVL-Bäume T^- und T^+. Da der Median μ sich im Updateschritt ändern könnte, speichern wir nicht $|r_1 - \mu|, \ldots, |r_n - \mu|$, sondern die Eingabedaten $r_1, \ldots, r_n$. Erst beim Zugriff auf ein Element im Baum wird der Median subtrahiert und der Betrag genommen. Zusätzlich wird in jedem Teilbaum die Anzahl der gespeicherten Elemente verwaltet, womit der Rang eines Elementes leicht berechnet werden kann, siehe auch Cormen, Leiserson, Rivest und Stein (2001, Kapitel 15.1).

Wir speichern die Elemente aus der Folge Val^- in T^-, die der anderen in T^+. Durch das Verschieben des Zeitfensters muss ein alter Wert gelöscht und ein neuer eingefügt werden. Falls die Operationen in verschiedenen Bäumen stattfinden, stimmt der Inhalt der Bäume nicht mehr mit der Definition von Val^- und Val^+ überein. Es genügt, maximal ein Element vom einen in den anderen Baum zu verschieben, um die Übereinstimmung wieder herzustellen. Hierbei ist zu berücksichtigen, dass für ungerades n das mediane Element in beiden Bäumen gespeichert wird. Der Median μ ergibt sich sowohl für ungerades n als auch gerades n als der Mittelwert aus dem größten Element aus T^- und dem kleinsten aus T^+.

Die Intervallschachtelung entspricht im Baum dem Durchlaufen eines Pfades von der Wurzel bis zu einem Blatt. Wird z.B. das Intervall $]L^-, R^-[$ zu $]p, R^-[$ verkürzt, so entspricht dies im Baum dem Verzweigen nach rechts am Knoten, der p enthält. Beim Verzweigen nach rechts werden alle Elemente des linken Teilbaumes und der aktuelle Knoten ausgeschlossen. Das offene Intervall ist leer, sobald ein Null-Zeiger gefunden wird.

Da AVL-Bäume eine maximale Tiefe von $\mathcal{O}(\log n)$ garantieren, ist die Suche nach dem co-MAD x_i in Zeit $\mathcal{O}(\log n)$ abgeschlossen. Aus x_i lässt sich der Rang des Elementes x^* berechnen und sowohl x_i als auch x' lassen sich mit einer Suche im Baum einfach finden. Die AVL-Bäume benötigen linearen Platz. Wir erhalten:

Theorem 35 *In einem Datenfenster der Breite n kann ein Update der MAD-Schätzung in Zeit $\mathcal{O}(\log n)$ und Platz $\mathcal{O}(n)$ berechnet werden.*

4.1.1 Update des MTM-Schätzer

Statt wie beim so genannten Mean-Filter alle Residuen zu addieren, um eine Zeitreihe zu erhalten, wird beim MTM-Schätzer(Modified Trimmed Mean) nur über die kleinen Residuen summiert. Als Kriterium wird der Median und der MAD wie folgt verwendet:

Definition 36 (MTM-Problem (Modified Trimmed Mean))

`Eingabe:`
n Zahlen $r_1, \ldots, r_n$ mit $r_i \in \mathbb{R}$ und eine Konstante c.
`Problem:`
Berechne $MTM(r_1, \ldots, r_n) = \sum_{|r_i - \mu| \leq \sigma} r_i$ mit $\mu = \operatorname{med}_i(r_i)$
und $\sigma = c \cdot MAD(r_1, \ldots, r_n)$.

Es ist leicht zu sehen, dass aus gegebenen μ und σ der MTM-Schätzer in Zeit $\mathcal{O}(\log n)$ aktualisiert werden kann. Da der Update des Medians μ in Zeit $\mathcal{O}(\log n)$ möglich ist, siehe Cormen, Leiserson, Rivest und Stein (2001), und der MAD-Schätzer ebenfalls in Zeit $\mathcal{O}(\log n)$ aktualisiert werden kann, wie in Kapitel 4.1 beschrieben, kann der MTM-Schätzer ebenfalls in Zeit $\mathcal{O}(\log n)$ aktualisiert werden, indem geeignete Suchanfragen an die AVL-Bäume gestellt werden.

4.2 Update des RM-Schätzers

Der zweite Schätzer, den wir im Zeitreihenkontext betrachten, ist der RM-Schätzer (Repeated Median). Der RM-Schätzer wurde von Siegel (1982) vorgeschlagen. Er hat einen Bruchpunkt von 50% und für Punktmengen in der Ebene ist er wie folgt definiert:

Definition 37 (RM-Problem (Repeated Median))

`Eingabe:`
n Punkte $(x_1, y_1), \ldots, (x_n, y_n)$ in der Ebene, $x_i \neq x_j$ für alle $1 \leq i < j \leq n$.
`Problem:`
Die Steigung der Geraden durch die Punkte (x_i, y_i) und (x_j, y_j) ist $a_{i,j} = \frac{y_i - y_j}{x_i - x_j}$.
Finde den Repeated-Median-Schätzer $(\beta_{\mathrm{RM}}, \alpha_{\mathrm{RM}})$, der gegeben ist durch

$$\beta_{\mathrm{RM}} = \underset{i=1,\ldots,n}{\mathrm{med}} \underbrace{\underset{j=1,\ldots,n, j\neq i}{\mathrm{med}} (a_{i,j})}_{\text{innere Mediane}} ,$$

$$\alpha_{\mathrm{RM}} = \underset{i=1,\ldots,n}{\mathrm{med}} (y_i - \beta_{\mathrm{RM}} x_i) .$$

Da es möglich ist, einen Median in linearer Zeit zu berechnen, kann der Repeated Median mit zwei verschachtelten Median-Operationen in Zeit $O(n^2)$ berechnet werden, im späteren auch Brute-Force-Ansatz genannt.

Stein und Werman (1992) stellen einen deterministischen Algorithmus vor, der die duale Transformation verwendet. Für eine gegebene Steigung können sie in Zeit $\mathcal{O}(n \log n)$ entscheiden, ob der RM eine kleinere oder größere Steigung hat. Sie verknüpfen diese Entscheidungsvarianten mit einem Sortiernetzwerk und erzielen damit eine Rechenzeit von $\mathcal{O}\big(n \log^2 n\big)$. Mit Randomisierung verbessern Matoušek, Mount und Netanyahu (1998) die Rechenzeit auf $O(n \log n)$. Die Verbesserung beruht darauf, dass sie eine Suchstrategie angeben, die mit Subsampling ein Intervall berechnet, das mit hoher Wahrscheinlichkeit den RM enthält. Damit erreichen sie, dass die Entscheidungsvariante im Mittel nur konstant oft aufgerufen wird. Allerdings ist eine Implementierung sehr aufwändig, daher beschreiben sie zusätzlich einen einfachen Algorithmus mit Rechenzeit $\mathcal{O}\big(n \log^2 n\big)$.

Problemtransformation

Wir verwenden die duale Transformation, welche in der Einleitung beschrieben wurde, und erhalten die folgende Problemformulierung:

Definition 38 (RM$^{\mathcal{T}}$-Problem)

`Eingabe:`
n Punkte $(x_1, y_1), ..., (x_n, y_n)$ in der Ebene mit $x_i \neq x_j$ für alle $1 \leq i < j \leq n$.
`Transformation:`
Jeder Punkt (x_i, y_i) wird auf die Gerade $\ell_i : v = x_i u - y_i$ abgebildet. Die Geraden ℓ_i und ℓ_j schneiden sich in einem Punkt, dessen u-Koordinate wir mit $u_{i,j}$ bezeichnen. Auf jeder Geraden ℓ_i gibt es einen Punkt, so dass sich gleich viele Punkte links und rechts davon befinden. Seine u-Koordinate ist gegeben durch $\mathrm{med}_{j \neq i}(u_{i,j})$.
`Problem:`
Finde ein u^*, so dass $u^* = \mathrm{med}_i \, \mathrm{med}_{j \neq i}(u_{i,j})$.

Update des Repeated Median

Die Geraden ℓ_i und ℓ_j schneiden sich in einem Schnittpunkt mit der u-Koordinate $u_{i,j}$. Eine einfache Rechnung zeigt, dass

$$u_{i,j} = \frac{y_i - y_j}{x_i - x_j} = a_{i,j} \ .$$

Somit ist $u_{i,j}$ gleich der Steigung $a_{i,j}$, wie sie in der RM-Definition verwendet wurde. Um die Sprechweise zu vereinfachen, wird im Folgenden $u_{i,j}$ auch als Schnittpunkt bezeichnet, da die v-Koordinate nicht weiter wichtig ist. In der Datenstruktur *Hammock-Graph*, die im nächsten Abschnitt ausführlicher beschrieben wird, werden alle Schnittpunkte explizit gespeichert. Eine weitere Eigenschaft der Datenstruktur ist, dass alle Schnittpunkte einer Geraden in sortierter Reihenfolge gehalten werden. Sie unterteilen die Geraden in Segmente. Nach der Transformation sind die inneren Mediane $\mathrm{med}_j(x_{i,j})$ Punkte auf den Geraden, die nicht notwendigerweise auf Schnittpunkten liegen.

Falls $n-1$ gerade ist, liegt der Punkt $\mathrm{med}_j(x_{i,j})$ auf einem Segment von ℓ_i, anderenfalls ist er der rechte Schnittpunkt eines Segments. Dieses Segment bezeichnen wir mit m_i. Der Update-Algorithmus basiert darauf, sich alle medianen Segmente m_i zu merken und im Verlauf der Rechnung aktuell zu halten.

Bei einem Updateschritt wird ein alter Punkt entfernt und ein neuer eingefügt. Im Dualraum entspricht dies dem Entfernen und Einfügen einer Geraden. Die Gerade ℓ_i hat daher genau einen Schnittpunkt mit der alten und einen mit der neuen Geraden. Je nachdem, wo diese beiden Schnittpunkte bezüglich des medianen Segments m_i liegen, wird m_i sich maximal um eine Position in der sortierten Folge verschieben.

Daher ist es im Hammock-Graphen möglich, die Mediane $m_1, \ldots, m_n$ in Zeit $\mathcal{O}(n)$ zu aktualisieren, siehe Lemma 42, Seite 61. Ebenfalls in linearer Zeit lässt sich aus $m_1, \ldots, m_n$ der Median $\text{med}_i(m_i) = \beta_{\text{RM}}$ berechnen, siehe auch Dor und Zwick (1999) für einen deterministischen Algorithmus und Cormen, Leiserson, Rivest und Stein (2001) für den randomisierten Algorithmus Quickselect.

Die Lemmata 40 und 41 auf Seite 59 sagen aus, dass das Entfernen und Einfügen einer Geraden in den Hammock-Graphen ebenfalls in linearer Zeit möglich ist; wir werden die beiden Operationen ab Seite 58 detaillierter beschreiben. Ein degeneriertes Geraden-Arrangement liegt dann vor, wenn sich mehr als zwei Geraden in dem gleichen Punkt schneiden. In Lemma 43 auf Seite 61 zeigen wir, dass auch dieser Fall korrekt behandelt wird. Somit ergibt sich das folgende Theorem:

Theorem 39 *Für n gegebene Punkte in der Ebene gibt es eine Datenstruktur, die Platz $\mathcal{O}(n^2)$ verwendet und die Repeated-Median-Schätzung bei einem Update, d.h. Entfernen und Einfügen eines Punktes, in Zeit $\mathcal{O}(n)$ aktuell hält.*

Die praktische Relevanz des Algorithmus ist in Abbildung 13 dargestellt.

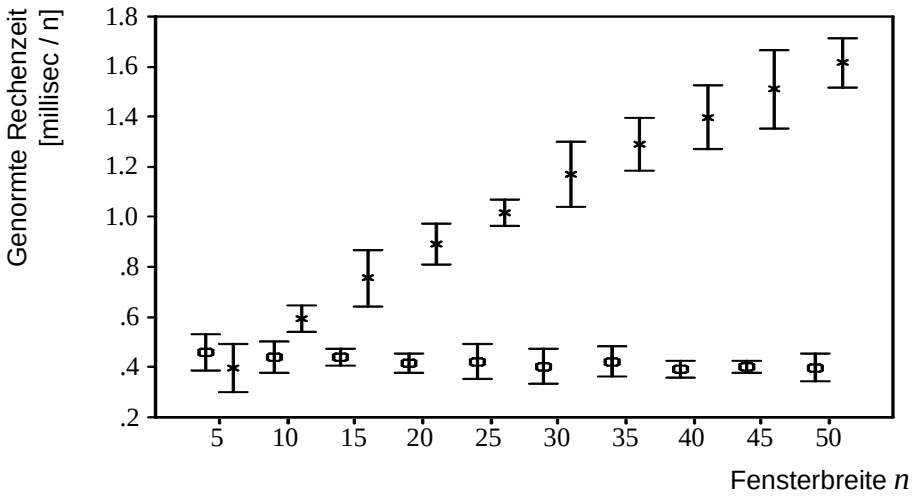


Abbildung 13: Für 100 Update-Schritte sind die mittlere normierte Rechenzeit und die Standardabweichung dargestellt. Die Fensterbreite n variiert von 5 bis 50, wobei die Kreise leicht nach links und die Kreuze leicht nach rechts verschoben sind. Der Update-Algorithmus(Kreis) wird verglichen mit dem Brute-Force-Algorithmus (Kreuz) und er ist für $n \geq 10$ überlegen. Die Rechenzeiten wurden gemessen auf einem Pentium III 800 Mhz mit 512 MB. Matoušek, Mount und Netanyahu (1998) geben an, dass sie ab $n > 155$ schneller als der Brute-Force-Algorithmus sind.

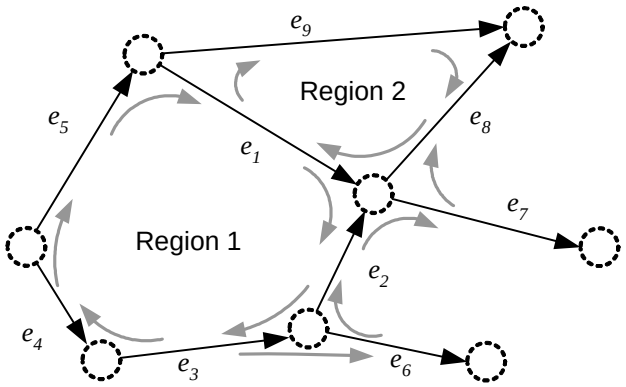


Abbildung 14: Die Segmente im Hammock-Graphen werden im Uhrzeigersinn verbunden. Die Zeiger werden als graue, geschwungene Pfeile gezeichnet. Die impliziten Knoten werden als gestrichelte Kreise dargestellt.

Hammock-Graph

Chazelle (1986) stellt die Datenstruktur Hammock-Graph vor, die n Geraden speichert und die Operationen „Einfügen", „Entfernen" und „Zugriff auf einzelne Segmente" zur Verfügung stellt. Die n Geraden schneiden sich in $\binom{n}{2}$ Schnittpunkten, wobei Schnittpunkte verschiedener Geraden mit gleichen Koordinaten mehrfach gezählt und separat gespeichert werden. Eine Gerade wird hierdurch in $n + 2$ *Segmente* unterteilt, wobei Geraden gleicher Steigung nicht zulässig sind. Der Raum selbst wird in Regionen unterteilt. Eine Region ist ein Teil der Ebene, der teilweise oder vollständig von Segmenten umgeben ist. Die Segmente einer Region werden im Uhrzeigersinn angeordnet. Ausgehend von einem Segment bezeichnen wir das nächste Segment in der Ordnung als *Kreis-Nachfolger*. Die Segmente einer Geraden sind bezüglich der u-Koordinaten von links nach rechts angeordnet. Das nächste Segment bezeichnen wir (in beiden Richtungen) als *Geraden-Nachfolger*. Bei degenerierten Arrangements kann es mehrere aufeinanderfolgende Segmente mit Länge Null geben, so dass dann die Einfügereihenfolge maßgebend ist. Der Hammock-Graph unterstützt folgende Operationen:

- Bestimmung des Kreis-Nachfolgers eines Segments in Zeit $\mathcal{O}(1)$, wobei die Segmente einer Region im Uhrzeigersinn geordnet sind.
- Bestimmung des Geraden-Nachfolgers eines Segments in Zeit $\mathcal{O}(1)$.
- Einfügen einer Geraden in Zeit $\mathcal{O}(n)$.
- Entfernen einer Geraden in Zeit $\mathcal{O}(n)$.

Für jedes Segment werden folgende Informationen gespeichert: Jedes Segment grenzt an zwei implizite Knoten, deren u-Koordinaten u_{Links} und u_{Rechts} an dem Segment gespeichert werden. Ein Segment besitzt eine Richtung, die in Abbildung 14 mit einer Pfeilspitze gekennzeichnet ist, die immer nach rechts weist. Da jedes Segment an zwei Regionen angrenzt, benötigen wir zwei Zeiger auf die jeweiligen Kreis-Nachfolger. Zeiger sind in den Abbildungen als graue, geschwungene Pfeile dargestellt.

Jeder Zeiger hat ein Flag, das angibt, wie ein Segment und sein Kreis-Nachfolger zueinander orientiert sind. Falls an dem gemeinsamen impliziten Knoten genau eine Pfeilspitze und ein Pfeilende angrenzen, wird in dem Flag ein *false* gespeichert. Dies ist in Abbildung 14 z.B. an den Segmenten e_5 und e_1 der Fall. Wenn im anderen Fall zwei Pfeilspitzen oder Pfeilenden aneinander stoßen, wird ein *true* gespeichert, wie z.B. bei den Segmenten e_1 und e_2. Das Flag ist notwendig, wenn wir z.B. die Region 1 im Uhrzeigersinn durchlaufen wollen. Beim Durchlaufen müssen wir entscheiden, ob der Zeiger an der Pfeilspitze oder am Pfeilende auf ein Segment der Region 1 zeigt. Das Flag gibt nun an, ob der Kreis-Nachfolger eine andere Orientierung hat und somit zum Zeiger am anderen Segmentende gewechselt werden muss.

An jeder Geraden ℓ_i wird ein medianes Segment m_i gespeichert, aus dem sich die inneren Mediane ergeben, und eine Zählvariable $z(\ell_i)$, die zählt, wie viele Schnittpunkte der Geraden ℓ_i sich links von m_i befinden. Zusätzlich wird in jedem Segment die Gerade gespeichert, aus der es hervorgegangen ist, um Zugriff auf $z(\ell_i)$ zu haben. Wahlweise ermöglicht es dieser Eintrag, die u-Koordinaten der Schnittpunkte auszurechnen, so dass die Einträge u_{Links} und u_{Rechts} nicht unbedingt abgespeichert werden müssen.

Die Datenstruktur weicht von der sonst verwendeten „doubly connected edge list", beschrieben in de Berg, van Kreveld, Overmars und Schwarzkopf (2000), dadurch ab, dass jedes Segment nur einmal gespeichert und nur eine einfache Verkettung verwendet wird, wodurch die Hälfte an Speicherplatz eingespart werden kann.

Grundoperationen

Wie aus der Verkettung zu erkennen ist, lassen sich die Segmente, die an eine Region angrenzen, in Zeit $\mathcal{O}(1)$ pro Segment besuchen. Hierbei ist zu beachten, dass beim Durchlaufen der oberen Hälfte der Zeiger an der Pfeilspitze verwendet wird, in der unteren Hälfte der andere Zeiger. Der Übergang zwischen oberer und unterer Hälfte liegt dort vor, wo das Flag auf true gesetzt ist, da an diesen beiden Stellen zwei Pfeilspitzen oder Pfeilenden aufeinander treffen. Um den Geraden-Nachfolger bzw. -Vorgänger zu finden, müssen zwei Zeiger verfolgt werden, wie in Abbildung 14 leicht zu sehen. Die Einfüge-Operation stellt sicher, dass sich auch in degenerierten Arrangements in jedem Schnittpunkt genau zwei Geraden schneiden.

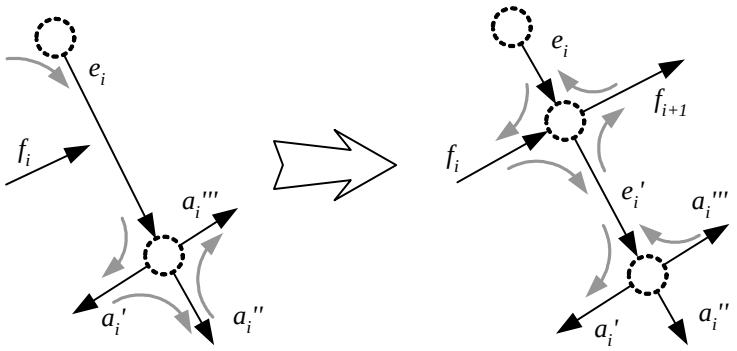


Abbildung 15: Beim Einfügen einer Geraden wird das vorhandene Segment e_i zerteilt und die neuen Segmente f_i und f_{i+1} eingefügt. In der linken Bildhälfte ist die Situation vorher dargestellt. In der rechten Hälfte sind die Zeiger zu sehen, die neu gesetzt werden müssen.

Einfügen einer Geraden

Um die Implementierung einfacher zu gestalten, wird das Arrangement links und rechts im Unendlichen von zwei vertikalen Geraden L und R begrenzt. Daher besteht der leere Hammock-Graph aus zwei Segmenten L und R. Um das Ergebnis für das erste Zeitfenster zu erhalten, müssen die ersten n Geraden eingefügt werden. Da eine Gerade $n-1$ Schnitte mit anderen Geraden hat und zwei weitere mit L und R, besteht jede Gerade aus $n+2$ Segmenten. Die Segmente, die sich jenseits von L und R befinden, dienen dem Zweck, beim Löschen einer Geraden Fallunterscheidungen zu vermeiden. Dies wird im nächsten Abschnitt genauer erläutert.

Das Einfügen einer neuen Geraden ℓ_k besteht aus zwei Teilschritten, die in der Implementierung allerdings simultan ausgeführt werden. Als Erstes werden die Segmente $e_1, \ldots, e_{n+1}$ bestimmt, die von der Geraden ℓ_k geschnitten werden. Dazu starten wir am untersten Segment der Gerade L und durchlaufen die Kreis-Segmente der untersten Region, bis ein Segment gefunden wird, das von der neuen Geraden geschnitten wird. Dazu wird die u-Koordinate u^* des Schnittpunktes von Segment und eingefügter Gerade berechnet und mit denen der Koordinaten des Segments verglichen. Falls $u_{\text{Links}} \leq u^* \leq u_{\text{Rechts}}$ gilt, ist ein Schnitt gefunden, und die Suche wird in der benachbarten Region fortgeführt. Bei der Suche werden insgesamt maximal $\mathcal{O}(n)$ Segmente besucht, wie sich aus dem so genannten Zone-Theorem ergibt, siehe Chazelle (1986); de Berg, van Kreveld, Overmars und Schwarzkopf (2000). Die Suche ist beendet, falls ein Schnitt mit einem Segment von R gefunden wird.

Im zweiten Teilschritt werden die Segmente $f_1, \ldots, f_{n+2}$ der neuen Geraden in die Datenstruktur eingefügt. Die Segmente werden miteinander verknüpft, wie in Abbil-

dung 15 dargestellt. Zu beachten ist, dass die Richtung des neuen Segments e_i' mit e_i übereinstimmt. Somit erhalten wir:

Lemma 40 *Eine Gerade kann in einen Hammock-Graphen, der n Geraden beinhaltet, in Zeit $\mathcal{O}(n)$ eingefügt werden.*

Entfernen einer Geraden

Beim Entfernen werden alle Segmente $f_1, \ldots, f_{n+2}$, die zur alten Geraden gehören, aus dem Hammock-Graphen entfernt, wie in Abbildung 16 dargestellt, und alle Segmente e_i und e_i' zu einem Segment verschmolzen. Alle abgebildeten Segmente, insbesondere a_i', a_i'', a_i''', existieren, da im schlimmsten Fall a_i' und a_i''' Segmente der vertikalen Gerade R sind, und jede Gerade noch jenseits von R über ein Segment verfügt. Falls ein medianes Segment involviert ist, wird es vorher „zur Seite geschoben", um Komplikationen zu vermeiden. Somit erhalten wir:

Lemma 41 *Eine Gerade kann aus einem Hammock-Graphen, der n Geraden beinhaltet, in Zeit $\mathcal{O}(n)$ entfernt werden.*

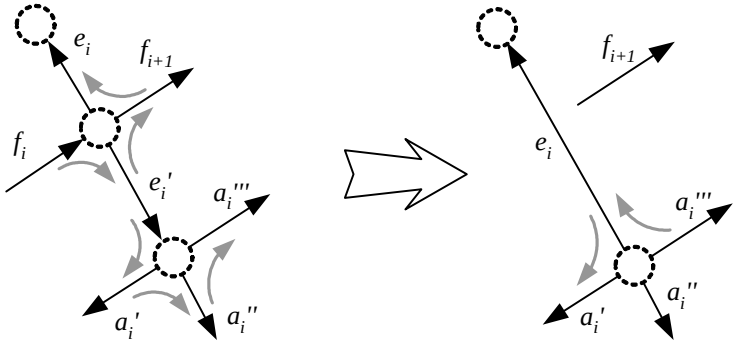


Abbildung 16: Beim Entfernen einer Geraden werden die Segmente f_i und f_{i+1} entfernt, und die Segmente e_i und e_i' zu einem Segment vereinigt. In der linken Bildhälfte die Situation vorher. Die neu zu setzenden Zeiger sind in der rechten Bildhälfte eingezeichnet.

Update der Mediane

Wie in der Einleitung bereits angesprochen, wird auf jeder Geraden ℓ_i ein medianes Segment m_i gespeichert, aus dessen Endpunkten sich nach Definition 3 auf Seite 9 die inneren Mediane berechnen lassen. Beim Einfügen oder Entfernen einer Gerade entsteht auf jeder Geraden ℓ_i ein neuer Schnittpunkt, oder es wird einer entfernt. Die

	c unterhalb von ν^*	c oberhalb von ν^*	
Steigung(ℓ_q) > Steigung(ℓ_i)	links	rechts	
Steigung(ℓ_q) < Steigung(ℓ_i)	rechts	links	

Abbildung 17: Die Einträge geben an, ob der Schnittpunkt ν^* sich links oder rechts vom medianen Segment m_i befindet.

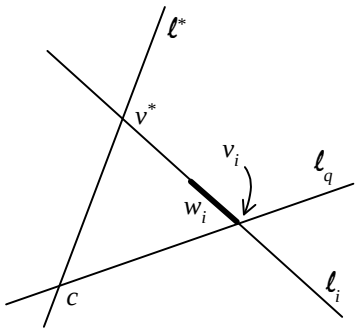


Abbildung 18: Dargestellt ist der linke obere Fall aus nebenstehender Tabelle.

Zählvariable $z(\ell_i)$ wird angepasst, falls der eingefügte/entfernte Schnittpunkt links von m_i liegt. Falls $z(\ell_i) \neq \lfloor (n-1)/2 \rfloor$ ist, so muss das mediane Segment m_i entsprechend verschoben werden.

Im Folgenden beschreiben wir, wie bestimmt wird, ob der Schnittpunkt ν^* aus ℓ_i und ℓ^* sich links von m_i befindet. Ein einfacher Vergleich der Koordinaten der Schnittpunkte führt nicht zum Ziel, da das Arrangement degeneriert sein könnte und dann mehrere Schnittpunkte die gleichen Koordinaten hätten. Wie in Abbildung 18 dargestellt, bezeichnen wir den rechten Endpunkt von m_i mit ν_i . Sei ℓ_q die Gerade, die in dem Punkt ν_i schneidet. Die Gerade ℓ_q und die zuletzt eingefügte Gerade ℓ^* schneiden sich im Punkt c. Hierbei sind vier Fälle zu unterscheiden, und zwar einmal, ob die Steigung von ℓ_q größer ist als die von l_i und ob der Punkt c unterhalb oder oberhalb von ℓ_i liegt. Die vier Fälle sind in Tabelle 17 aufgeführt und aus dem Vorliegen eines Falles kann gefolgert werden, ob der eingefügte/entfernte Schnittpunkt links von m_i liegt.

Zunächst zur Korrektheit der Herangehensweise; später beschreiben wir, wie die vier Fälle algorithmisch unterschieden werden. Die erste Annahme ist die, dass die eingefügte Gerade die größte Steigung und die entfernte Gerade die kleinste Steigung besitzt. (Wir können uns auch eine senkrechte Gerade vorstellen.) Dies resultiert daraus, dass die Steigung im Primalraum der Zeitkoordinate entspricht, und jeweils der jüngste bzw. älteste Punkt eingefügt bzw. gelöscht wird. Somit folgt, dass beide Schnittpunkte ν^* und c sich entweder beide links oder beide rechts von ν_i befinden.

Man kann leicht einsehen, dass unter der Annahme ν^* ist links von ν_i, gilt: c ist unterhalb von ν^*, genau dann, wenn die Steigung von ℓ_q größer ist als die von ℓ_i. Falls ν^* rechts von ν_i ist, verhält es sich genau umgekehrt.

Um zu ermitteln, ob c unterhalb von ν^* ist, markieren wir beim Einfügen von ℓ^* die bereits abgearbeiteten Geraden. Wenn c unterhalb von ν^* liegt, wird die zugehörige Gerade früher markiert. Da bei einer Update-Operation auf einer Geraden genau ein Schnittpunkt entsteht und einer entfernt wird, ändert sich die Anzahl $z(\ell_i)$ maximal um zwei. Wir erhalten:

Lemma 42 *Das Update der n medianen Segmente $m_1, \ldots, m_n$ kann in Zeit $\mathcal{O}(n)$ durchgeführt werden.*

Degenerierte Arrangements

In der Statistik wird angenommen, dass die Daten zufällig erzeugt werden und sich in allgemeiner Lage befinden. Es kann jedoch passieren, dass drei oder mehr Punkte auf einer Linie liegen, wobei die Wahrscheinlichkeit dafür in den meisten Fällen gering ist. Aber auch in diesen Fällen soll der Algorithmus korrekt funktionieren. Im dualen Raum entspricht dies dem Problem, dass drei oder mehr Geraden sich in einem Punkt schneiden. So ein Mehrfach-Schnittpunkt wird nicht direkt im Hammock-Graph gespeichert, sondern für jedes Paar der beteiligten Geraden wird ein separater Schnittpunkt verwendet. Daher enthält der Hammock-Graph Segmente der Länge 0 mit $u_{\text{Links}} = u_{\text{Rechts}}$, die wir als *Null-Segmente* bezeichnen. Die Operationen „Entfernen einer Geraden" und „Update der Mediane" verwenden keine Koordinaten, daher werden wir uns nur die Operation „Einfügen einer Geraden" genauer ansehen. Wir betrachten die Gerade $\ell_n : v = x_n u + y_n$, die eingefügt wird und mindestens einen Mehrfach-Schnittpunkt durchläuft. Wir vergleichen diese mit der Geraden $\ell_n^\varepsilon : v = x_n u + y_n + \varepsilon$, die um $\varepsilon > 0$ verschoben wurde, wobei das ε so klein gewählt ist, dass lediglich der Mehrfach-Schnittpunkt verlassen wird und kein anderer Punkt zwischen ℓ_n und ℓ_n^ε liegt. Wir zeigen:

Lemma 43 *Beim Einfügen der Geraden ℓ_n oder ℓ_n^ε werden dieselben Segmente zerteilt und abgesehen von den u-Koordinaten ist die Struktur der Hammock-Graphen identisch.*

Beweis: Wir betrachten ein Segment e_i, das von der Geraden ℓ_n^ε zerteilt wird. Da kein Schnittpunkt zwischen den beiden Geraden liegt, ist der einzige kritische Fall der, dass der rechte Schnittpunkt von e_i ein Mehrfach-Schnittpunkt ist und ℓ_i diesen schneidet. Da beim Einfügen die Segmente der Region im Uhrzeigersinn durchlaufen werden, wissen wir, dass zuerst e_i bearbeitet wird und danach die Null-Segmente, die zu M gehören. Da die Schnittbedingung $u_{\text{Links}} \leq u^* \leq u_{\text{Rechts}}$, die beim Einfügen verwendet wird, mit dem Punkt u_{Rechts} auf Gleichheit testet, folgt, dass auch ℓ_n das Segment e_i zerteilt. □

5 Schätzer in d Dimensionen

5.1 LMS-Schätzer

Einer der ersten robusten Schätzer, der das „Subset-Paradigma" verwendet, ist der LMS-Schätzer (Least Median of Squares). Anwendungen des LMS-Schätzers finden sich in zahlreichen Veröffentlichungen. Der LMS wird z.B. von Plets und Vynckier (1999) verwendet, die einen 3-dimensionalen astronomischen Datensatz analysieren. Pizarro und Singh (2003) verwenden den robusten Schätzer, um Unterwasser-Bilder zu einem Mosaik zusammenzusetzen. Li, Xu, Morrison, Nightingale und Morphett (2003) generieren aus einem MPEG-Video ein Panorama-Bild.

Rousseeuw (1984) und Rousseeuw und Leroy (1987) definieren den LMS-Schätzer (siehe Definition 8 auf Seite 11) als diejenige Hyperebene, die das h-te Residuum minimiert. In der geometrischen Sicht ist es einfacher, die Hyperebene nach oben und unten um den gleichen Betrag zu verschieben, so dass wir einen Hyperstreifen erhalten. Wir wählen die Breite des Hyperstreifens so, dass alle Punkte, deren Residuen kleiner gleich dem h-ten Residuum sind, sich in diesem Hyperstreifen befinden, wobei der Rand zum Hyperstreifen gezählt wird. Aus dieser Sicht ergibt sich die folgende Definition:

Definition 44 (LMS-Problem (Least Median of Squares))

`Eingabe:`
n Punkte $P_1, \ldots, P_n$ mit $P_i \in \mathbb{R}^d$, und eine ganze Zahl h, mit$\lceil n/2 \rceil \leq h \leq n$.
`Problem:`
Finde einen Hyperstreifen (L^+, L^-) mit

$$L^+ : y = a_1 x_1 + \cdots + a_{d-1} x_{d-1} + a_d + r$$
$$L^- : y = a_1 x_1 + \cdots + a_{d-1} x_{d-1} + a_d - r \ ,$$

so dass r minimal ist und der Hyperstreifen h Punkte enthält.

Der LMS-Schätzer ist in der Statistik nur für $h = n/2$ bzw. $h = \lceil n/2 \rceil + \lceil (d+1)/2 \rceil$, je nach Literaturstelle, definiert.(Die ursprüngliche Idee war, das mediane Residuum

zu minimieren, daher $h = n/2$. Die zweite Wahl von h optimiert statistische Eigenschaften des Schätzers.) In der Informatik hat es sich eingebürgert, ein h zwischen $n/2$ und n zuzulassen. Dieser Schätzer wird in der Statistik als Least-Quantile-of-Squares-Schätzer bezeichnet. Wir werden aber weiterhin den Term LMS verwenden.

In der Statistik wird im Kontext von LMS oft Stromberg (1993) zitiert. Er beschreibt einen Algorithmus für LMS mit einer Rechenzeit von $\mathcal{O}\big(n^{d+2}\log n\big)$. Er testet alle $\binom{n}{d+1}$ Punktteilmengen, die einen Hyperstreifen definieren. Um einen Hyperstreifen eindeutig festzulegen, werden $d+1$ linear unabhängige Punkte benötigt. Allerdings gibt es 2^{d+1} mögliche Zuordnungen der $d+1$ Punkte zu L^+ oder L^-. Er verwendet so genannte Chebyshev-Fits und zitiert Theoreme, die aussagen, dass nur eine der Zuordnungen eine minimale Lösung ist. Falls h Punkte im Hyperstreifen liegen, werden die Residuen sortiert und das h-te gewählt. Die Rechenzeit von Stromberg (1993) lässt sich auf $\mathcal{O}\big(n^{d+2}\big)$ verbessern, indem nicht sortiert, sondern eine Median-Berechnung vorgenommen wird. Von Erickson, Har-Peled und Mount (2004) wurde die Rechenzeit auf $\mathcal{O}\big(n^d \log n\big)$ verbessert. In zwei Dimensionen geben Edelsbrunner und Souvaine (1990) einen Algorithmus mit einer Rechenzeit $\mathcal{O}(n^2)$ und linearem Platz an.

Güte $1-\varepsilon$ erreichen Mount et al. (2000) und benötigen eine Rechenzeit von $\mathcal{O}\big(n\log n + (1/\varepsilon)^{\mathcal{O}(d)}\big)$. Mount et al. (1997) approximieren LMS in zwei Dimensionen mit einem Branch & Bound-Ansatz. Der Nachteil der Approximation ist, dass der Schätzer für $n \to \infty$ nicht konvergiert, eine wichtige Eigenschaft des Schätzers somit verloren geht.

Problemtransformation

Definition 45 (LMS$^{\mathcal{T}}$-Problem)

`Eingabe:`
n Punkte $P_1, \ldots, P_n$ mit $P_i \in \mathbb{R}^d$ und eine ganze Zahl h mit $\lceil n/2 \rceil \leq h \leq n$.
`Transformation:`
Jeder Punkt $P_i = (p_{i,1}, \ldots, p_{i,d})$ wird auf die Hyperebenen

$$H_i^+ : v = +p_{i,1}u_1 + \cdots + p_{i,d-1}u_{d-1} + u_d - p_{i,d}$$
$$H_i^- : v = -p_{i,1}u_1 - \cdots - p_{i,d-1}u_{d-1} - u_d + p_{i,d}$$

abgebildet. Die Raumachsen sind $u_1, \ldots, u_d, v$.
`Problem:`
Finde einen Punkt $L = (a_1, \ldots, a_d, r)$, so dass seine Höhe r minimal ist und er $n+h$ Hyperebenen dominiert.

Lemma 46 *Aus einer minimalen Lösung L des LMS$^{\mathcal{T}}$-Problems ergibt sich eine minimale Lösung (L^+, L^-) des LMS-Problems.*

Beweis: Die beiden Lösungen sind über ihre Parameter $a_1, \ldots, a_d, r$ miteinander verknüpft. Wir schauen uns zunächst die Lage der beiden Lösungen zueinander an:

$$
\begin{aligned}
& L \text{ dominiert } H_i^+ \\
\Leftrightarrow \quad & L \text{ ist oberhalb oder schneidet } H_i^+ \\
\Leftrightarrow \quad & \exists c \geq 0 : (a_1, \ldots, a_d, r - c) \text{ schneidet } H_i^+ \\
\Leftrightarrow \quad & \exists c \geq 0 : r - c = p_{i,1}a_1 + \cdots + p_{i,d-1}a_{d-1} + a_d - p_{i,d} \\
\Leftrightarrow \quad & \exists c \geq 0 : p_{i,d} - c = a_1 p_{i,1} + \cdots + a_{d-1}p_{i,d-1} + a_d - r \\
\Leftrightarrow \quad & \exists c \geq 0 : (p_{i,1}, \ldots, p_{i,d-1}, p_{i,d} - c) \text{ schneidet } L^- \\
\Leftrightarrow \quad & P_i \text{ ist oberhalb oder schneidet } L^-
\end{aligned}
$$

Eine analoge Rechnung zeigt, dass

$$L \text{ dominiert } H_i^- \Leftrightarrow \quad P_i \text{ ist unterhalb oder schneidet } L^+ \ .$$

Daraus folgt: Der Punkt L dominiert genau dann $n + h$ Hyperebenen, wenn der Hyperstreifen (L^+, L^-) h Punkte enthält. Die Behauptung, dass L eine minimale Lösung ist, folgt, indem wir die Negation nachweisen:

$$(L^+, L^-)' \text{ ist nicht minimal} \Leftarrow L \text{ ist nicht minimal} \ .$$

Wenn eine Lösung $(L^+, L^-)'$ nicht minimal ist, gibt es einen schmaleren Hyperstreifen der Breite $r' < r$. Aus obiger Rechnung folgt, dass es einen Punkt mit Höhe r' gibt, der eine Lösung ist. Also ist die Lösung L' mit Höhe $r > r'$ nicht minimal. Aus dem Widerspruch folgt, dass (L^+, L^-) eine minimale Lösung für das LMS-Problem ist. □

Algorithmus

Die transformierte Eingabe besteht aus $2n$ Hyperebenen in $d + 1$ Dimensionen, die sich in $\mathcal{O}\big(n^{d+1}\big)$ vielen Schnittpunkten schneiden. Eine Idee dieses neuen Algorithmus ist es, die Eigenschaft auszunutzen, dass die v-Achse den Wert der Lösung misst. Eine Hyperebene H_{r^*}, die die v-Achse in einer Höhe von r^* schneidet und senkrecht zur Achse liegt, enthält alle Punkte, die mögliche Lösungen mit dem Wert r^* darstellen. H_{r^*} enthält nur $\mathcal{O}\big(n^d\big)$ Schnittpunkte. Im nächsten Abschnitt werden wir zeigen, dass ein Aufzählen aller Schnittpunkte das Entscheidungsproblem in Zeit $\mathcal{O}\big(n^d\big)$ löst.

Um zu einer Optimierungsvariante zu gelangen, wenden wir die Technik von Chan an, die in der Einleitung bereits beschrieben wurde. Dazu unterteilen wir den Lösungsraum des LMS-Problems in $2n$ Unterräume $A_1, \ldots, A_{2n}$. Jedes A_i wird wiederum in

$A_{i,1}, \ldots, A_{i,n-1}$ unterteilt. Der Unterraum A_i enthält alle Schnittpunkte, die auf einer fixierten Hyperebene liegen. Der Unterraum $A_{i,j}$ enthält alle Schnittpunkte, die sich als Schnitt aus zwei H^+- oder aus zwei H^--Hyperebenen ergeben. Die anderen beiden Kombinationen müssen nicht berücksichtigt werden, wie wir später zeigen werden. Die Teilprobleme werden kanonisch gewählt, so dass jeder Schnittpunkt des Lösungsraumes in mindestens einem A_i und einem $A_{i,j}$ enthalten ist.

Wir werden später zwei Prozeduren vorstellen. Die Prozedur „DecideLMS" entscheidet, ob A_i bzw. $A_{i,j}$ eine Lösung enthält, die kleiner oder gleich r^* ist. Die Prozedur „SolveLMS" berechnet eine minimale Lösung aus dem Lösungsraum $A_{i,j}$.

Der Algorithmus hat zwei Rekursionsebenen. Zu Beginn setzen wir $r^* = \infty$. Die Variable enthält zu jedem Zeitpunkt die bisher beste gefundene Lösung. Auf der ersten Ebene durchlaufen wir die Unterräume $A_1, \ldots, A_{2n}$ in einer zufälligen Reihenfolge. Für jedes A_i wird zuerst mit „DecideLMS" getestet, ob eine bessere Lösung als r^* in A_i existiert. Falls nicht, schreiten wir fort und betrachten den nächsten Unterraum in der zufälligen Folge. Falls eine bessere Lösung existiert, verzweigen wir zur zweiten Rekursionsebene, die uns eine minimale Lösung für A_i liefert, die wir in r^* speichern. In der zweiten Rekursionsebene durchlaufen wir die Unterräume $A_{i,1}, \ldots, A_{i,2n}$ ebenfalls in einer zufälligen Reihenfolge. Wie zuvor prüfen wir mit „DecideLMS", ob eine bessere Lösung als r^* existiert. Falls eine Lösung existiert, verwenden wir „SolveLMS", um die beste Lösung im Unterraum zu finden und r^* zu aktualisieren. Nachdem alle Unterräume $A_{i,1}, \ldots, A_{i,2n}$ betrachtet wurden, erhalten wir eine minimale Lösung für A_i .

Die Rechenzeit beider Prozeduren ist proportional zur Anzahl der Schnittpunkte des betrachteten Unterraumes. Die Rechenzeit der Prozedur „SolveLMS", angewendet auf den gesamten Lösungsraum, ist um den Faktor n größer als die von „DecideLMS", da hier der Schnitt mit H_{r^*} eine Dimension einspart. Da aber „SolveLMS" auf Räumen $A_{i,j}$ arbeitet, deren Dimension um 2 reduziert wurde, verringert sich die Rechenzeit von $\mathcal{O}\big(n^{d+1}\big)$ auf $\mathcal{O}\big(n^{d-1}\big)$. Wie wir in Lemma 14 auf Seite 20 gezeigt haben, bewirkt die obige Konstruktion, dass „SolveLMS" nur $\mathcal{O}(\log n)$-mal aufgerufen wird. Die Rechenzeit wird daher von „DecideLMS" dominiert, die in der ersten Rekursionsebene $2n$ mal aufgerufen wird. Insgesamt genügt somit eine Rechenzeit von $\mathcal{O}\big(n^d\big)$. Da der Raum $A_{i,j}$ eine Dimension von $d-1$ hat und SolveLMS auf mindestens zweidimensionalen Unterräumen arbeitet, funktioniert dieser Ansatz nur für $d \geq 3$. Für $d = 2$ kann der Algorithmus von Edelsbrunner und Souvaine (1990) verwendet werden. Aus Lemma 48 und Lemma 49 auf Seite 69 folgt die Korrektheit und wir erhalten:

Theorem 47 *Eine Lösung für das LMS-Problem kann in Zeit $\mathcal{O}\big(n^d\big)$ und Platz $\mathcal{O}(n)$ berechnet werden.*

SolveLMS: Finden einer minimalen Lösung in $A_{i,j}$.

Die Prozedur basiert darauf, alle Schnittpunkte in $A_{i,j}$ aufzuzählen, und für jeden die Anzahl der Hyperebenen zu bestimmen, die dominiert werden, also sich unterhalb befinden oder schneiden. Pro Schnittpunkt wollen wir mit Rechenzeit $\mathcal{O}(1)$ auskommen.

Wir nehmen eine weitere Unterteilung des Unterraumes $A_{i,j}$ vor, so dass wir lineare Unterräume der Dimension zwei erhalten. Diese können wir wie folgt durchsuchen: Ein Unterraum enthält maximal $\binom{2n}{2}$ Schnittpunkte, die auf maximal $2n$ Geraden angeordnet sind. Wir bestimmen diese Geraden und verwenden den topologischen Sweepline-Algorithmus von Rafalin, Souvaine und Streinu (2002). Er zählt alle Schnittpunkte in Zeit $\mathcal{O}(n^2)$ auf, wobei die Schnittpunkte, die auf einer Geraden liegen, in sortierter Folge aufgezählt werden. In jedem Punkt schneidet genau eine Hyperebene und somit ändert sich die Anzahl D der dominierten Hyperebenen genau um Eins. Dies ermöglicht es uns, nachdem wir für den ersten Punkt jeder Geraden die Anzahl D in Zeit $\mathcal{O}(n)$ bestimmt haben, D für die nachfolgenden Punkte in Zeit $\mathcal{O}(1)$ zu aktualisieren.

Auf diese Weise wird für jeden Schnittpunkt in $A_{i,j}$ die Anzahl dominierter Hyperebenen bestimmt. Aus allen Punkten wird der Punkt mit minimaler v-Koordinate, der $n+h$ Hyperebenen dominiert, gewählt und ist die minimale Lösung in $A_{i,j}$. Kumuliert beträgt die Rechenzeit pro Schnittpunkt $\mathcal{O}(1)$, also benötigt SolveLMS Zeit $\mathcal{O}\big(n^{d-1}\big)$.

DecideLMS: Entscheiden, ob eine Lösung kleiner gleich r^* in A_i bzw. $A_{i,j}$ existiert.

Wie bereits zuvor erwähnt, haben alle Punkte auf der Hyperebene H_{r^*} den gleichen Lösungswert r^*. Wir betrachten alle Schnittpunkte, die im Schnitt von H_{r^*} und A_i bzw. $A_{i,j}$ liegen. Wir verwenden den topologischen Sweepline-Algorithmus und erhalten wie im vorherigen Abschnitt für jeden Punkt die Anzahl der dominierten Hyperebenen. Falls es einen Punkt gibt, der mindestens $n+h$ Hyperebenen dominiert, wird JA ausgegeben, ansonsten NEIN. Die Rechenzeit für A_i bzw. $A_{i,j}$ ist $\mathcal{O}\big(n^{d-1}\big)$ bzw. $\mathcal{O}\big(n^{d-2}\big)$, da wie zuvor pro Schnittpunkt eine Rechenzeit von $\mathcal{O}(1)$ genügt. Die einzige Ausnahme ist der Fall, dass DecideLMS für $A_{i,j}$ mit $d=3$ aufgerufen wird. Dann ist $A_{i,j}$ ein eindimensionaler linearer Unterraum, also eine Gerade. Da wir die Punkte auf der Geraden sortieren müssen, hat SolveLMS die Rechenzeit $\mathcal{O}(n \log n)$ für $d=3$.

Lemma 48 *Wenn A_i und $A_{i,j}$ eine optimale Lösung enthalten, gibt DecideLMS auf diesen Unterräumen JA aus.*

Beweis: Sei L eine optimale Lösung aus $A_{i,j}$ und r^* der Wert der bisher besten Lösung vor dem Aufruf von DecideLMS. Wir wählen eine Menge M, die $d+1$ Hy-

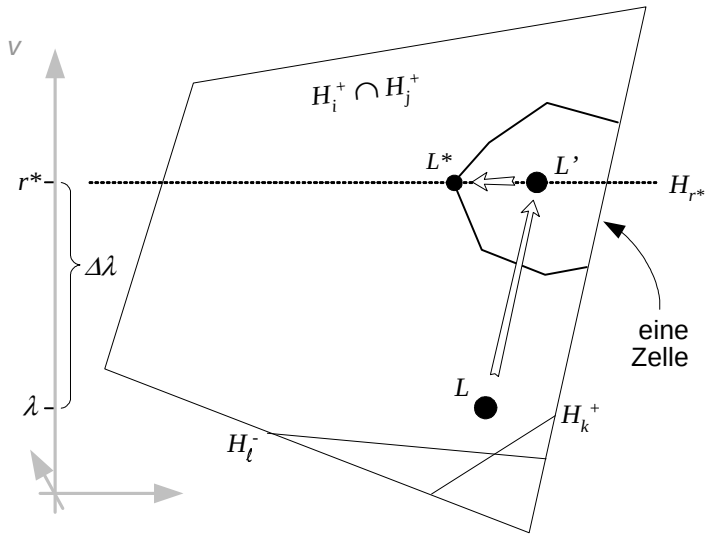


Abbildung 19: Wenn der Punkt L zum Punkt L' geschoben wird, bleiben die Hyperebenen H_k^+ und H_ℓ^- unterhalb. Der Punkt wird weiter verschoben zur Position L^*, wo er jetzt $d+1$ Hyperebenen schneidet.

perebenen enthält, die sich im Punkt L schneiden. Die Menge M ist Teilmenge von $\{H_1^+, \ldots, H_n^+, H_1^-, \ldots, H_n^-\}$. Für $d \geq 3$ enthält M zwei H^+ oder zwei H^--Hyperebenen. O.B.d.A. betrachten wir den Fall, dass M die Hyperebenen H_i^+ und H_j^+ enthält, wie in Abbildung 19 dargestellt. Der Lösungspunkt $L = (a_1, \ldots, a_d, \lambda)$ ist im Schnitt von H_i^+ und H_j^+ enthalten. Setzen wir L in die Gleichungen der Hyperebenen ein, so erhalten wir

$$\begin{aligned} \lambda &= p_{i,1}a_1 + \cdots + p_{i,d-1}a_{d-1} + a_d - p_{i,d} \\ \lambda &= p_{j,1}a_1 + \cdots + p_{j,d-1}a_{d-1} + a_d - p_{j,d} \ . \end{aligned}$$

Damit es übersichtlicher wird, fassen wir einige Terme zusammen:

$$\begin{aligned} \lambda &= Q_i + a_d \\ \lambda &= Q_j + a_d \ . \end{aligned}$$

Wenn wir nun λ und a_d um $\Delta\lambda$ vergrößern, so folgt aus den Gleichungen, dass der Punkt $L = (a_1, \ldots, a_d + \Delta\lambda, \lambda + \Delta\lambda)$ immer noch im Schnitt von H_i^+ und H_j^+ liegt. Wir wählen $\Delta\lambda$, so dass $\lambda + \Delta\lambda = r^*$ und erhalten den Punkt L'. Eventuell ist L' kein Schnittpunkt und befindet sich somit in einer Zelle des Arrangements. Wir wählen einen Punkt L^*, der ein Schnittpunkt von H_{r^*} mit einer Seitenkante dieser Zelle ist. Wir zeigen nun, dass L^* mindestens $n + h$ Hyperebenen dominiert. Wir betrachten die Hyperebenen H_k^+ und H_ℓ^-, die von L dominiert werden, also unterhalb liegen oder den Punkt schneiden. Daher sind folgende Ungleichungen erfüllt

$$\begin{aligned} \lambda &\geq Q_k + a_d \\ \lambda &\geq -Q_\ell - a_d \ . \end{aligned}$$

Der Punkt L' erfüllt ebenfalls diese Ungleichungen, da λ und a_d höchstens größer sind. Da die Verbindungslinie von L' und L^* keine anderen Hyperebenen als H_k^+ und H_ℓ^- schneidet, gilt für beide Punkte, dass H_k^+ und H_ℓ^- sich unterhalb befinden oder zumindest schneiden. Da L mindestens $n+h$ Hyperebenen dominiert, folgt, dass L^* ebenfalls mindestens $n+h$ Hyperebenen dominiert. Darüber hinaus liegt L^* in dem Unterraum $H_{r^*} \cap A_{i,j}$. Er wird also von DecideLMS aufgezählt, und da er $n+h$ Hyperebenen dominiert, gibt DecideLMS JA aus. Die gleichen Argumente gelten auch für den Unterraum A_i. □

Mit einer analogen Argumentation folgt:

Lemma 49 *Wenn A_i und $A_{i,j}$ keine Lösung $\leq r^*$ enthalten, gibt DecideLMS(r^*) auf diesen Unterräumen NEIN aus.*

5.2 MCD-Schätzer

Neben dem linearen Modell mit einer Hyperebene und anderen offenen Kurven bzw. Flächen gibt es das elliptische Modell, das davon ausgeht, dass dem Datenmodell ein Ellipsoid zugrunde liegt. Die Dichtefunktion der zugrunde gelegten Verteilung ist derart gestaltet, dass es ein Ellipsoid gibt, so dass jeder Punkt auf dem Rand des Ellipsoids die gleiche Dichte hat. Das gängige Beispiel hierfür ist die multivariate Normalverteilung. Variieren wir den Radius des Ellipsoids, so erhalten wir eine Dichte in Abhängigkeit vom Radius, in diesem Beispiel die Dichte der Normalverteilung.

Um robust für einen gegebenen Datensatz ein elliptisches Modell zu schätzen, gibt es z.B. den MCD-Schätzer (Minimum Covariance Determinant) und MVE-Schätzer (Minimum Volume Ellipsoid). Ein Beispiel ist in Abbildung 20 dargestellt:

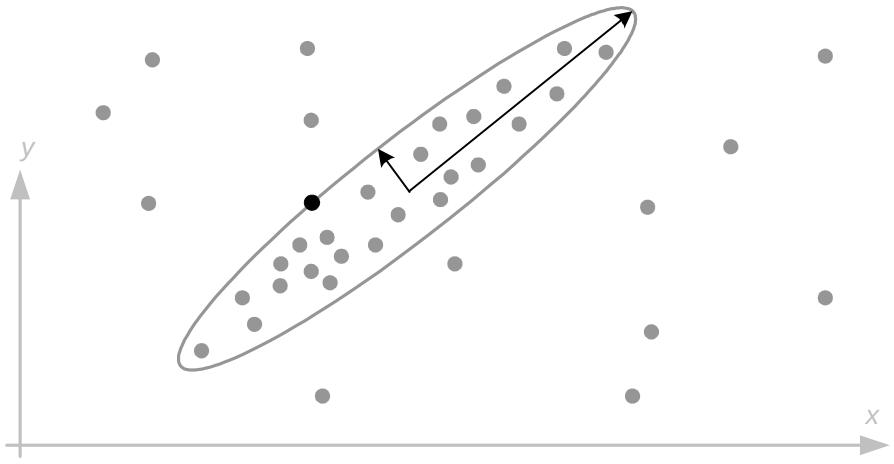


Abbildung 20: Eine Kovarianzellipse, die robust mit dem MCD-Schätzer an die Daten angepasst wurde. Es liegt genau ein Punkt auf dem Rand und Entfernungen (Mahalanobis-Distanzen) werden in dem Koordinatensystem gemessen, das sich aus den beiden Achsen der Ellipse ergeben, die identisch zu den Eigenvektoren der Kovarianzmatrix sind.

Bei dem MVE-Schätzer wird ein Ellipsoid gesucht, das h Punkte enthält und ein minimales Volumen hat. In einer Vergleichsstudie von Becker und Gather (2001) schneidet der MVE-Schätzer am schlechtesten ab. Ein weiterer Nachteil des MVE-Schätzers ist, dass die Form des Ellipsoids stark von den Randpunkten abhängt und daher die Schätzung schnell verfälscht wird. Bei dem MCD-Schätzer hingegen ergibt sich das Ellipsoid aus allen Punkten, die innerhalb des Ellipsoids liegen.

Die Punktmenge $S = \{P_1, \ldots, P_h\}$ hat den Schwerpunkt

$$\text{mean}(S) = \frac{1}{h} \sum_{i=1,\ldots,h} P_i \ .$$

Einen Punkt $P_i \in \mathbb{R}^d$ schreiben wir als d-Tupel bzw. Zeilenvektor $P_i = (p_{i,1}, \ldots, p_{i,d})$. Durch Transponieren erhalten wir einen Spaltenvektor $P_i^\top = (p_{i,1}, \ldots, p_{i,d})^\top$. Die Kovarianzmatrix der Punktmenge S bezeichnen wir mit $\text{cov}(S)$ und sie ist gegeben durch

$$\begin{aligned} \text{cov}(S) &= \frac{1}{h} \sum_{i=1,\ldots,h} (P_i - \text{mean}(S))^\top \cdot (P_i - \text{mean}(S)) \\ &= \frac{1}{h} \left(\sum_{i=1,\ldots,h} P_i^\top \cdot P_i \right) - \text{mean}(S)^\top \cdot \text{mean}(S) \ . \end{aligned}$$

Die Determinante der Kovarianzmatrix $\det(\text{cov}(S))$ schreiben wir kurz als $\det(S)$.

Definition 50 (MCD-Problem (Minimum Covariance Determinant))

`Eingabe:`
Eine Menge von n Punkten $\mathcal{P} = \{P_1, \ldots, P_n\}$ mit $P_k \in \mathbb{R}^d$ und eine ganze Zahl h mit $\lceil n/2 \rceil \leq h \leq n$.

`Problem:`
Finde eine Teilmenge $S \subseteq \mathcal{P}$ der Größe $|S| = h$, so dass die Determinante der Kovarianzmatrix $\det(S)$ minimal ist.

Anwendungen des MCD-Schätzers finden sich z.B. in Filzmoser, Garrett und Reimann (2005), Hubert und Engelen (2004) und Zhao, Yue und Fang (2004).

In zahlreichen Arbeiten werden Suchheuristiken vorgestellt, die den MCD-Schätzer berechnen; die derzeit schnellste ist der Fast-MCD von Rousseeuw und van Driessen (1999). Weitere Arbeiten sind Rocke und Woodruff (1993), Hawkins und Olive (1999b) und Rocke und Woodruff (1999).

Ein Schätzer ist jedoch nur robust, wenn er exakt berechnet wird. Daher sind Suchheuristiken nicht geeignet, den MCD-Schätzer zuverlässig zu berechnen. Pesch (2000) verwendet verschiedene Methoden, u.a. Branch & Bound, um nicht alle $\binom{n}{h}$ Teilmengen betrachten zu müssen, erlangt jedoch keine theoretischen Aussagen über Laufzeiten. Wir werden eine neue obere Schranke von ungefähr $n^{\mathcal{O}(d^2)}$ beweisen.

Für spätere Beweise benötigen wir die Mahalanobis-Distanz. Sie ist normalerweise in Bezug zu einer Matrix und einem Schwerpunktvektor definiert. Um die Notation zu vereinfachen, definieren wir sie in Relation zu einer Punktmenge, aus der wir dann die Matrix und den Schwerpunktvektor erhalten:

Definition 51 (Mahalanobis-Distanz) *Sei S eine Punktmenge mit Punkten aus $\mathbb{R}^d$. Die Mahalanobis-Distanz* $\mathrm{md}(P_i, S)$ *eines Punktes P_i bezüglich S ist gegeben durch*

$$\mathrm{md}(P_i, S) = \sqrt{(P_i - \mathrm{mean}(S)) \cdot \mathrm{cov}(S)^{-1} \cdot (P_i - \mathrm{mean}(S))^\top} \ .$$

Ein Ellipsoid mit Radius r ist definiert als die Menge

$$\mathrm{ell}(S, r) = \{x \mid (x - \mathrm{mean}(S)) \cdot \mathrm{cov}(S)^{-1} \cdot (x - \mathrm{mean}(S))^\top \leq r\} \ .$$

Wir erhalten den Rand des Ellipsoids, wenn wir in obiger Formel $= r$ verwenden und somit haben alle Punkte auf dem Rand die gleiche Mahalanobis-Distanz. Als Kovarianz-Ellipsoid $\mathrm{ell}(S)$ bezeichnen wir das Ellipsoid $\mathrm{ell}(S, r)$, das einen minimalen Radius r hat, so dass alle Punkte der Menge S innerhalb liegen. Die Determinante der Kovarianzmatrix $\det(S)$ ist ein Maß für das Volumen von $\mathrm{ell}(S, 1)$. Rousseeuw und van Driessen (1999, Korollar 1) beweisen das folgende Lemma:

Lemma 52 *Sei $\mathcal{P}$ eine Punktmenge. Für die optimale Lösung S des MCD-Problems gibt es ein Ellipsoid, das S von $\mathcal{P} \backslash S$ trennt.*

Da das Kovarianz-Ellipsoid einen minimalen Radius hat, befindet sich mindestens ein Punkt aus S auf dem Rand. Rousseeuw und van Driessen (1999) merken jedoch an, dass auf dem Rand eines trennenden Ellipsoids unter Umständen Punkte aus beiden Mengen liegen können. Für minimale Kovarianz-Ellipsoide ist das jedoch nicht der Fall, wie wir jetzt zeigen:

Lemma 53 *Sei S mit $|S| = h$ eine Teilmenge einer Punktmenge $\mathcal{P}$, so dass* $\det(S)$ *minimal ist. Dann enthält der Rand des Kovarianz-Ellipsoids* $\mathrm{ell}(S)$ *keine Punkte aus $\mathcal{P} \backslash S$.*

Beweis: Wir nehmen das Gegenteil an und betrachten eine Teilmenge $S \subset \mathcal{P}$ mit $|S| = h$, die so gewählt ist, dass ein Punkt $P \in S$ und ein Punkt $Q \in \mathcal{P} \backslash S$ auf dem Rand von $\mathrm{ell}(S)$ liegen. Wir werden zeigen, dass für die Teilmenge $S' = (S \backslash \{P\}) \cup \{Q\}$ gilt: $\det(S') < \det(S)$. Da das Kovarianz-Ellipsoid äquivariant gegenüber Rotation, Skalierung und Translation ist, transformieren wir die Punktmenge $\mathcal{P}$ derart, dass die Kovarianzmatrix die Identität ist, d.h. $\det(S) = 1$ und der Schwerpunkt $\mathrm{mean}(S) = 0$.

Da Kovarianzmatrizen positiv definit sind (siehe Horn und Johnson (1985, Seite 392)), ist nach der Hadamard-Ungleichung $\det(S)$ kleiner als das Produkt der Diagonalelemente (siehe Horn und Johnson (1985, Seite 477)). Mit p_i bzw. q_i bezeichnen wir die i-te Koordinate der Punkte P bzw. Q und mit t_i die i-te Koordinate von $mean(S')$. Da alle Punkte paarweise verschieden sind, ist $mean(S') \neq 0$. Das Produkt der Diagonalelemente von $\mathrm{cov}(S')$ ist

$$\frac{1}{h} \prod_{i=1\ldots d} \left(h - p_i^2 + q_i^2 - \frac{t_i^2}{h} \right) < \frac{1}{h} \prod_{i=1\ldots d} \left(h - p_i^2 + q_i^2 \right) = \prod_{i=1\ldots d} (1 + z_i) \quad , \tag{5.1}$$

mit $z_i = \frac{1}{h}(-p_i^2+q_i^2)$. Da beide Punkte auf dem Rand liegen, gilt $\mathrm{md}(P, S') = \mathrm{md}(Q, S')$ und wir erhalten $\sum_i p_i^2 = \sum_i q_i^2$, also $\sum_i z_i = 0$.

Aus der Definition der Kovarianzmatrix folgt, dass $1 + z_i \geq 0$ ist. Da $\mathrm{md}(P, S) = \mathrm{md}(Q, S) = 1$, folgt $p_i^2 \leq 1$ und $q_i^2 \leq 1$. Da $h \geq d + 1$ ist, folgt $z_i > -1$ und wir können das Produkt schreiben als $\prod_i (1+z_i) = \sum_i \log(1+z_i)$. Aus der Konkavität des Logarithmus folgt, dass unter der Nebenbedingung $\sum_i z_i = 0$ das Produkt $\prod_i (1 + z_i)$ maximal ist, falls alle z_i gleich sind, also $z_i = 0$ und wir erhalten, dass $\prod_i (1 + z_i) \leq 1$. Mit Gleichung 5.1 folgt, dass $\det(S') < \det(S)$.

Falls es also einen Punkt aus $\mathcal{P} \backslash S$ auf dem Rand gibt, folgt, dass durch einen Punktaustausch die Determinante verkleinert werden kann und dies steht im Widerspruch zur Minimalität von $\det(S)$. □

Problemtransformation

Wir transformieren das Problem, um es einfacher handhaben zu können:

Definition 54 (MCD$^{\mathcal{T}}$-Problem)

`Eingabe:`
Eine Menge von n Punkten $\mathcal{P} = \{P_1, \ldots, P_n\}$ mit $P_k \in \mathbb{R}^d$ und eine ganze Zahl h mit $\lceil n/2 \rceil \leq h \leq n$.

`Transformation:`
Wir nehmen an, dass die Koordinaten aller Punkte ungleich Null sind. (Anderenfalls kann dies durch eine Verschiebung der Punktmenge leicht herbeigeführt werden.) Jeder Punkt $P_k = (p_{k,1}, \ldots, p_{k,d})$ wird auf die Hyperebene

$$H_k : v = \sum_i p_{k,i} \cdot u_i + \sum_{i \leq j} p_{k,i}\, p_{k,j} \cdot u_{i,j}$$

abgebildet. Die Raumachsen sind $u_1, \ldots, u_d, u_{1,1}, u_{1,2}, \ldots, u_{d,d}$ und v.

`Problem:`
Finde einen Punkt $L' = (a_1, \ldots, a_d, a_{1,1}, a_{1,2}, \ldots, a_{d,d}, r')$, so dass L' mindestens h Hyperebenen $H_{\pi(1)}, \ldots, H_{\pi(h)}$ dominiert und $\det(S)$ mit $S = \{P_{\pi(1)}, \ldots, P_{\pi(h)}\}$ minimal ist.

Wir sagen, dass ein Ellipsoid eine Punktmenge *selektiert*, falls die Punkte im Innern oder auf dem Rand des Ellipsoids liegen. Wir zeigen nun, dass die Transformation korrekt ist:

Lemma 55 *Die Punktteilmenge S, die von der Lösung L' des MCD$^{\mathcal{T}}$-Problems selektiert wird, ist eine Teilmenge mit minimaler Determinante und S ist eine optimale Lösung des MCD-Problems.*

Beweis: Wir betrachten zunächst eine Teilmenge S, die eine optimale Lösung des MCD-Problems ist. Aus Lemma 52 und Lemma 53 folgt, dass es ein Kovarianz-Ellipsoid L gibt, das die Punkte aus S selektiert.

$$
\begin{aligned}
& L \text{ selektiert } S \\
\Leftrightarrow \quad & \forall P_k \in S : \mathrm{md}(P_k, S) \leq r^2 \\
\Leftrightarrow \quad & \forall P_k \in S : (P_k - \mathrm{mean}(S)) \cdot \mathrm{cov}(S)^{-1} \cdot (P_k - \mathrm{mean}(S))^\top \leq r^2
\end{aligned}
$$

Wir multiplizieren obigen Term aus und ersetzen im Ausdruck $P_k \,\mathrm{cov}(S)^{-1} P_k^\top$ den Term $\mathrm{cov}(S)^{-1}$ durch die symmetrische Matrix

$$
\begin{pmatrix}
a_{1,1} & \frac{1}{2}a_{1,2} & \dots & \dots & \frac{1}{2}a_{1,d} \\
\frac{1}{2}a_{1,2} & a_{2,2} & & & \vdots \\
\vdots & & \ddots & & \vdots \\
\vdots & & & \ddots & \frac{1}{2}a_{d-1,d} \\
\frac{1}{2}a_{1,d} & \dots & \dots & \frac{1}{2}a_{d,d-1} & a_{d,d}
\end{pmatrix} .
$$

Den Teilausdruck $-2\,\mathrm{mean}(S)\,\mathrm{cov}(S)^{-1}$ ersetzen wir durch $(a_1, \dots, a_n)$ und den Teilausdruck $\mathrm{mean}(S)\,\mathrm{cov}(S)^{-1}\,\mathrm{mean}(S)^\top$ durch a_0 und erhalten:

$$
\begin{aligned}
\Leftrightarrow \quad & \forall P_k \in S : + \sum_{i \leq j} p_{k,i}\, p_{k,j}\, a_{i,j} + \sum_i p_{k,i}\, a_i - a_0 \leq r^2 \\
\Leftrightarrow \quad & \forall P_k \in S : \ \exists c \geq 0 : (r^2 + a_0) - c = + \sum_i p_{k,i}\, a_i + \sum_{i \leq j} p_{k,i}\, p_{k,j}\, a_{i,j} \\
\Leftrightarrow \quad & \forall P_k \in S : \ \exists c \geq 0 : (a_1, \dots, a_d, a_{1,1}, a_{1,2}, \dots, a_{d,d}, \underbrace{(r^2 + a_0)}_{r'} - c) \text{ schneidet } H_k \\
\Leftrightarrow \quad & \forall P_k \in S : L' \text{ ist oberhalb oder schneidet } H_k \\
\Leftrightarrow \quad & \forall P_k \in S : L' \text{ dominiert } H_k \ .
\end{aligned}
$$

Daraus folgt: Der Punkt L' dominiert h Hyperebenen genau dann, wenn das Ellipsoid h Punkte enthält. Die Behauptung, dass L' eine minimale Lösung ist, folgt ebenfalls aus der obigen Rechnung. □

Algorithmus

In der Transformation werden die Datenpunkte auf ein Arrangement von Hyperebenen abgebildet und der Dualraum hat eine Dimension von $\delta = d(d+3)/2$. Ein Kovarianz-Ellipsoid entspricht jetzt einem Punkt im Dualraum. Alle Punkte, die sich innerhalb einer Zelle des Hyperebenen-Arrangements befinden, dominieren dieselben Hyperebenen $H_{\pi(1)}, \ldots, H_{\pi(h)}$ für eine geeignete Permutation π.

Um das Problem algorithmisch zu lösen, genügt es, alle Zellen des Hyperebenen-Arrangements aufzuzählen und für jede die dominierten Hyperebenen zu bestimmen.

Wir betrachten einen optimalen Punkt p^*, der nach Lemma 53 in genau einer Seitenfläche einer Zelle liegt. Er dominiert die Menge der Hyperebenen $S = \{H_{\pi^*(1)}, \ldots, H_{\pi^*(h)}\}$. Wir verschieben das Optimum p^* in einen Eckpunkt der Zelle, der ein Schnittpunkt aus δ Hyperebenen ist, die jeweils eine $\delta-1$-dimensionale Seitenfläche der Zelle enthalten. Diese Hyperebenen und die Information, ob sie oberhalb oder unterhalb liegen, charakterisieren die Zelle eindeutig. (Falls das Arrangement degeneriert ist, können sich in einem Eckpunkt mehr als δ Hyperebenen schneiden. Diese enthalten aber keine Seitenflächen der Zelle.)

Der Algorithmus zählt nun alle $\binom{n}{\delta}$ Schnittpunkte der Hyperebenen auf. Wir erhalten δ charakterisierende Hyperebenen für jeden Schnittpunkt. Eine charakterisierende Hyperebene kann oberhalb oder unterhalb der gesuchten Zelle liegen. Wir zählen daher alle 2^δ Möglichkeiten auf, die Hyperebenen entsprechend zu platzieren und erhalten bis zu 2^δ Zellen. Es läßt sich leicht in $\mathcal{O}(n)$ Zeit für eine Zelle bestimmen, welche Hyperebenen unterhalb der Zelle liegen.

Falls eine Zelle genau h Hyperebenen dominiert, ist sie ein Kandidat für eine optimale Lösung. Wir berechnen die Determinante der Kovarianzmatrix der zugehörigen Punktmenge in Zeit $\mathcal{O}(d^3)$. Das Aufzählen aller Zellen benötigt Rechenzeit $\mathcal{O}\big(d^3 \cdot 2^\delta \cdot n^{\delta+1}\big)$. Die Punktmenge mit kleinster Determinante ist das gesuchte Optimum. Wir erhalten:

Theorem 56 *Das MCD-Problem für n Punkte in d Dimensionen kann in Zeit $\mathcal{O}\big(d^3 \cdot 2^\delta \cdot n^{\delta+1}\big)$ mit $\delta = d(d+3)/2$ und Platz $\mathcal{O}(n)$ berechnet werden.*

6 Untere Schranken für die Berechnung robuster Schätzer

In der robusten Statistik sind einige untere Schranken für die Berechnung von robusten Schätzern bekannt. Gajentaan und Overmars (1995) zeigen, dass in zwei Dimensionen für $h = 3$ das LMS-Problem 3SUM-hart ist. Ein Problem Π ist 3SUM-hart, falls eine Reduktion in subquadratischer Zeit zwischen dem 3SUM-Problem und Π möglich ist. Das 3SUM-Problem besteht darin, für drei Mengen A, B, C mit $|A| + |B| + |C| = n$ zu entscheiden, ob es drei Zahlen $a \in A, b \in B, c \in C$ gibt, so dass $a + b + c = 0$ ist. Es wird vermutet, dass algebraische Entscheidungsbäume für dieses Problem quadratische Tiefe haben. Mittels einer Reduktion überträgt sich diese Vermutung auf das Problem, zu entscheiden, ob 3 aus n Punkten auf einer Geraden liegen, welches identisch zur Entscheidungsvariante des LMS-Problems für $h = 3$ ist.

Allerdings macht diese Schranke keine Aussage über andere Werte von h, und insbesondere bei LMS ist $h = \lceil 0.5n \rceil$ gebräuchlich. D.h., eigentlich sind wir an dem Problem interessiert, zu entscheiden, ob $\lceil 0.5n \rceil$ von n Punkten auf einer Geraden liegen. Ein einfacher randomisierter Algorithmus löst das Problem in linearer Zeit: Rate 2 Punkte, berechne die Gerade, die durch beide verläuft und prüfe, ob auf der Geraden $\lceil 0.5n \rceil$ Punkte liegen. Der Algorithmus hat eine Erfolgswahrscheinlichkeit von 1/4, die sich mit Multistarts beliebig steigern lässt. Das LMS-Problem ist jedoch allgemeiner, es soll ein Streifen gefunden werden, der h Punkte enthält und möglichst schmal ist. Der obige randomisierte Algorithmus findet nur Lösungen mit Streifen der Breite Null. Im allgemeinen Fall zeigen Chien und Steiger (1995) für LMS mit $h = \lceil 0.5n \rceil$ eine untere Schranke von $\Omega(n \log n)$ für algebraische Entscheidungsbäume. Des Weiteren besteht noch eine Lücke zum besten bekannten deterministischen Algorithmus mit Rechenzeit $\mathcal{O}(n^2)$ von Edelsbrunner und Souvaine (1990).

In diesem Kapitel werden wir den Fokus auf den Fall beliebiger Dimension d legen und die *NP*-Härte einiger Schätzprobleme zeigen. Ein Resultat aus der Literatur ist von Amenta, Bern, Eppstein und Teng (2000), die zeigen, dass das so genannte Regression-Depth-Problem *NP*-hart ist. Des Weiteren zeigen sie, dass die Berechnung der Entscheidungsvariante von Halfspace-Depth, auch Tukey-Median genannt, *co-NP*-vollständig ist.

All diese Probleme sind verwandt mit dem Max-FLS-Problem, das darin besteht, für ein System von n Gleichungen über $d < n$ Variablen eine Lösung zu finden, die möglichst viele Gleichungen erfüllt. Amaldi und Kann (1995) zeigen, dass unter der Annahme $P \neq NP$ Max-FLS nicht für alle $\varepsilon > 0$ auf den Faktor n^{ε} approximiert werden kann. Dieses Resultat lässt sich z.B. auf das LMS-Problem mittels einer dualen Transformation übertragen; die Gleichungen entsprechen Punkten, eine Lösung entspricht einer Hyperebene. Allerdings macht die Approximation des LMS-Problems keinen Sinn, da die Konsistenz-Eigenschaft verloren geht und die Schätzung für $n \to \infty$ nicht mehr konvergiert.

Erickson, Har-Peled und Mount (2004) machen die Annahme, dass Zeit $\Omega(n^{d+1})$ notwendig ist, um zu entscheiden, ob eine Menge von n Punkten aus $\mathbb{Z}^d$ eine Teilmenge von $d+1$ Punkten enthält, die auf einer gemeinsamen Hyperebene liegen. Dies übertragen sie direkt auf LMS, wobei dieses Resultat nur eine Aussage für $h = d+1$ macht. Diese Aussage hat wieder den Nachteil, dass die Statistik an $h = \lceil 0.5n \rceil$ interessiert ist, d.h., Statistiker machen die Annahme, dass ein konstanter Anteil der n Punkte aus Ausreißern besteht. Wir werden im nächsten Kapitel die *NP*-Härte für $h = cn$ mit $0.5 \leq c < 1$ zeigen und dieses neue Resultat auf weitere robuste Schätzer verallgemeinern.

6.1 Degenerierte Punktteilmengen

Die Exact-Fit-Property eines Schätzers, siehe Rousseeuw (1994), fordert, dass der Schätzer eine degenerierte Punktmenge findet. Dies motiviert das c-DPS-Problem, das wir verwenden, um die *NP*-Härte einiger robuster Schätzer zu zeigen:

Definition 57 (c-DPS-Problem (Degenerate Point Subset))

`Eingabe:`
n Punkte in d Dimensionen.
`Parameter:`
Ein fixierter Wert c mit $0 < c \leq 0.5$.
`Problem:`
Gibt es Parameter $\beta_1, \ldots, \beta_d$, so dass sich $h = \lceil (1-c) \cdot n \rceil$ Punkte auf der Hyperebene $y = \sum_{i=1..d-1} \beta_i x_i + \beta_d$ befinden?

Der Beweis für c-DPS selbst basiert auf dem *NP*-vollständigen Problem Vertex Cover, siehe (Wegener, 2003; Garey und Johnson, 1979).

Definition 58 (VC-Problem (Vertex Cover))

`Eingabe:`
Ein Graph $G = (V, E)$ und eine ganze Zahl $k \in \mathbb{N}$.
`Problem:`
Gibt es eine Teilmenge $V' \subseteq V$ der Größe $|V'| = k$, so dass von jeder Kante $e \in E$ mindestens ein inzidenter Knoten in V' enthalten ist?

Theorem 59 *Das c-DPS-Problem ist NP-hart für alle fixierten c mit* $0 < c \leq 0.5$.

Beweis: Wir führen eine polynomielle Reduktion VC $\leq_p$ c-DPS durch und transformieren den Graph in eine Punktmenge. Die Information, welcher Knoten in der Teilmenge V' ist, ist in den Parametern $\beta_1, \ldots, \beta_d$ der Hyperebene kodiert. Für $i = 1, \ldots, |V|$ bedeutet ein Wert $\beta_i = 2$, dass $v_i \in V'$ ist. Falls $\beta_i = 1$, so ist $v_i \notin V'$. Zunächst geben wir eine informelle Beschreibung der verwendeten Punkte:

- v_i und $\overline{v_i}$:
  Für jeden Knoten i konstruieren wir zwei Punkte v_i und $\overline{v_i}$. Wir verwenden sie später in der Konstruktion, um zu erzwingen, dass β_i nur die Werte 1 oder 2 annimmt.

- e'_{ij} und e''_{ij}:
  Für jede Kante (i, j) konstruieren wir diese beiden Punkte. Sie „prüfen", ob jede Kante des Vertex Covers abgedeckt ist.

- K^* und $\overline{K^*}$:
  Diese beiden Punkte „prüfen", ob das Vertex Cover aus genau k Knoten besteht.

- p^* und $\overline{p^*}$:
  Diese beiden Punkte erzwingen, dass $\beta_d = 0$ ist.

- $p_1, \ldots, p_\zeta$:
  Ohne diese Punkte würde die Konstruktion nur für $c = 0.5$ funktionieren. Mit genügend vielen Punkten kann der Beweis für jeden Wert von $0 < c \leq 0.5$ durchgeführt werden.

Die Koordinaten der genannten Punkte sind in der folgenden Tabelle aufgelistet:

$$
\begin{array}{rcccl}
 & & y & , & x_1,\ldots,x_i,\ldots\ldots,x_j,\ldots,x_{|V|}, \quad x_{|V|+1},\ldots,x_{|V|+\ell},\ldots,x_{|V|+\zeta} \\
\overline{v_i} & = (& 1 & , & 0,\ldots,0,1,0,\ldots\ldots\ldots\ldots,0, \quad 0,\ldots\ldots\ldots\ldots\ldots\ldots,0) \\
v_i & = (& 2 & , & 0,\ldots,0,1,0,\ldots\ldots\ldots\ldots,0, \quad 0,\ldots\ldots\ldots\ldots\ldots\ldots,0) \\
e'_{ij} & = (& 3 & , & 0,\ldots,0,1,0,\ldots,0,1,0,\ldots,0, \quad 0,\ldots\ldots\ldots\ldots\ldots\ldots,0) \\
e''_{ij} & = (& 4 & , & 0,\ldots,0,1,0,\ldots,0,1,0,\ldots,0, \quad 0,\ldots\ldots\ldots\ldots\ldots\ldots,0) \\
K^* & = (& |V|+k & , & 1,\ldots\ldots\ldots\ldots\ldots\ldots,1, \quad 0,\ldots\ldots\ldots\ldots\ldots\ldots,0) \\
\overline{K^*} & = (& |V|+k-1, & & 1,\ldots\ldots\ldots\ldots\ldots\ldots,1, \quad 0,\ldots\ldots\ldots\ldots\ldots\ldots,0) \\
p^* & = (& 0 & , & 0,\ldots\ldots\ldots\ldots\ldots\ldots,0, \quad 0,\ldots\ldots\ldots\ldots\ldots\ldots,0) \\
\overline{p^*} & = (& 1 & , & 0,\ldots\ldots\ldots\ldots\ldots\ldots,0, \quad 0,\ldots\ldots\ldots\ldots\ldots\ldots,0) \\
p_\ell & = (& 1 & , & 0,\ldots\ldots\ldots\ldots\ldots\ldots,0, \quad 0,\ldots\ldots\ldots,0,1,0,\ldots\ldots\ldots,0)
\end{array}
$$

Jeder Punkt besteht aus 3 Blöcken, der zweite Block hat eine Länge von $|V|$, der dritte eine Länge von ζ. Der Wert von ζ und die anderen Parameter sind hier aufgeführt:

$$
\begin{array}{rcl}
\zeta & = & 2\left\lceil \frac{1-2c}{2c} \cdot (|V|+|E|+2) \right\rceil \\
d & = & 1+|V|+\zeta \\
n & = & 2\cdot(|V|+|E|+2)+\zeta \\
h & = & 1\cdot(|V|+|E|+2)+\zeta
\end{array}
$$

Die Reduktion arbeitet nun wie folgt: Für jeden Knoten i des Graphen erzeugen wir die Punkte v_i und $\overline{v_i}$. Für jede Kante (i,j) erzeugen wir die Punkte e'_{ij} und e''_{ij}. Zusätzlich erzeugen wir die Punkte K^*, $\overline{K^*}$, p^* und $\overline{p^*}$. Aus dem Wert c erhalten wir ein ζ (siehe obige Tabelle). Wir erzeugen ζ Punkte $p_1,\ldots,p_\zeta$.

Behauptung 60 *Wenn der Graph G ein Vertex Cover der Größe k enthält, dann gibt es eine Hyperebene, die h Punkte enthält.*

Beweis: Wir betrachten ein Vertex Cover V' der Größe k. Wir wählen die Parameter der Hyperebene wie folgt: Wenn $i \in V'$, dann setzen wir $\beta_i = 2$, ansonsten $\beta_i = 1$. Die anderen Parameter wählen wir als $\beta_{|V|+1},\ldots,\beta_{d-1} = 1$ und $\beta_d = 0$.

Wir zählen nun die Punkte, die sich auf der Hyperebene befinden. Für jedes $i = 1,\ldots,|V|$ befindet sich entweder v_i oder $\overline{v_i}$ auf der Hyperebene. Dies sind $|V|$ Punkte. Da von jeder Kante (i,j) mindestens ein Knoten in V' ist, gilt, dass für jedes Punktepaar e'_{ij} und e''_{ij} entweder $\beta_i = 2$ oder $\beta_j = 2$ oder beides. Daher befindet sich genau einer der beiden Punkte auf der Hyperebene. Akkumuliert sind dies $|V|+|E|$ Punkte. Es ist leicht nachzuprüfen, dass K^*, p^* und $p_1,\ldots,p_\zeta$ sich auf der Hyperebene befinden. Insgesamt ergeben sich $|V|+|E|+2+\zeta$ Punkte. Daher enthält die Hyperebene h Punkte. □

Behauptung 61 *Wenn alle Vertex Cover größer als k sind, dann befinden sich auf jeder Hyperebene weniger als h Punkte.*

Beweis: Die Knoten v_i und $\overline{v_i}$ können sich nicht auf der gleichen Hyperebene befinden, da vertikale Hyperebenen mit der verwendeten Form nicht darstellbar sind. Analoges gilt für die Tupel (e'_{ij}, e''_{ij}), $(K^*, \overline{K^*})$ und $(p^*, \overline{p^*})$. Eine Ausnahme bilden $p_1, \ldots, p_\zeta$, die alle zugleich auf einer Hyperebene sein können. Es folgt, dass maximal $|V|+|E|+2+\zeta = h$ Punkte sich auf einer gemeinsamen Hyperebene befinden können. Falls ein Punkt fehlt, kann die erforderliche Zahl von h Punkten nicht erreicht werden. Wir werden im folgenden für jede Hyperebene zeigen, dass diese weniger als h Punkte enthält.

1.Fall $\beta_i \neq 1$ und $\beta_i \neq 2$ für ein $i = 1, \ldots, |V|$
Diese Hyperebenen enthalten keinen der Punkte v_i und $\overline{v_i}$.

2.Fall $\beta_d \neq 0$
Diese Hyperebenen enthalten nicht den Punkt p^*. Für $\beta_d = 1$ liegt der Punkt $\overline{p^*}$ auf der Hyperebene, allerdings ist es dann nicht möglich, dass ein Punkt eines jeden Paares $(v_i, \overline{v_i})$ und der Punkt K^* auf der Hyperebene liegen.

3.Fall $\beta_{|V|+1} \neq 1$ oder $\ldots$ oder $\beta_{|V|+\zeta} \neq 1$
Falls der Parameter $\beta_{|V|+\ell}$ ungleich Eins ist, liegt p_ℓ nicht auf der Hyperebene.

4.Fall: $\beta_i = 1$ oder $\beta_i = 2$ für alle $i = 1, \ldots, |V|$, $\beta_{|V|+1}, \ldots, \beta_{d-1} = 1$ und $\beta_d = 0$.
Wir definieren $V' = \{i \mid \beta_i = 2\}$ und betrachten zwei Fälle:

Falls $|V'| \leq k$, so wissen wir aus der Voraussetzung, dass V' kein Vertex Cover ist und daher gibt es eine Kante (i, j), die nicht abgedeckt ist. Daher sind $\beta_i = 1$ und $\beta_j = 1$ und weder e'_{ij} noch e''_{ij} befinden sich auf der Hyperebene.

Falls $|V'| > k$, so ist die Summe $\sum_{i=1..|V|} \beta_i > |V| + k$ und weder K^* noch $\overline{K^*}$ befinden sich auf der Hyperebene. □

Behauptung 62 *Die Parameter n, h und ζ erfüllen $h = \lceil (1-c) \cdot n \rceil$.*

Beweis: Wir ersetzen in der Gleichung n und h wie in der Parametertabelle spezifiziert, ersetzen $|V|+|E|+2$ durch X, und erhalten:

$$
\begin{aligned}
& & X + \zeta &= \lceil (1-c) \cdot (2 \cdot X + \zeta) \rceil \\
\Leftrightarrow & & 0 &= \lceil (1-c) \cdot (2 \cdot X + \zeta) - X - \zeta \rceil \\
\Leftrightarrow & & 0 &= \lceil X - 2c \cdot X - c \cdot \zeta \rceil \\
& \text{Einsetzen von } \zeta\text{:} & 0 &= \left\lceil X - 2c \cdot X - 2c \cdot \left\lceil \frac{1-2c}{2c} X \right\rceil \right\rceil \\
\Leftrightarrow & & 0 &= \left\lceil X - 2c \cdot X - 2c \cdot \left\lceil \frac{1}{2c} X - X \right\rceil \right\rceil
\end{aligned}
$$

Wir können die inneren Gaußklammern durch ein $\Delta \in [0, 1[$ ersetzen und erhalten:

$$\Leftrightarrow \quad 0 = \lceil -2c\Delta \rceil \quad .$$

Die Gleichung ist wahr, da wir $0 < c \leq 0.5$ gefordert haben, woraus $-2c\Delta \in \ ]-1, 0]$ folgt. □

Aus den drei Behauptungen folgt die *NP*-Härte von c-DPS. □

6.2 Subset-Schätzer

Wie wir gleich sehen werden, besteht eine grundlegende Idee der Robustheit darin, nur eine Teilmenge aller Punkte zu betrachten und die übrigen als Ausreißer zu bezeichnen. Um zu entscheiden, welche Teilmenge die richtige ist, wird jede Teilmenge mit einer Funktion bewertet. Eine Verallgemeinerung vieler Schätzer ist das folgende Problem:

Definition 63 (ε-Subset-Estimator-Problem)

`Eingabe:`
Eine Menge $\mathcal{P}$ von n Punkten in d Dimensionen.

`Parameter:`
Ein Parameter ε mit $0 < \varepsilon \leq 0.5$ und eine fixierte Funktion $f_{\text{subset}} : \mathbb{P}(\mathcal{P}) \to \mathbb{R}^+$ mit der Eigenschaft

$$f_{\text{subset}}(S) \begin{cases} = 0 & \text{falls } S \text{ auf einer Hyperebene liegt} \\ > 0 & \text{sonst} \quad . \end{cases}$$

`Problem:`
Sei $h = \lceil (1 - \varepsilon) \cdot n \rceil$. Finde eine Teilmenge $S \subseteq \mathcal{P}$ der Größe $|S| = h$, die $f_{\text{subset}}(S)$ minimiert.

Theorem 64 *Für jedes feste ε und jede Funktion f_{subset} mit den geforderten Eigenschaften ist das Subset-Estimator-Problem NP-hart.*

Beweis: Das c-DPS-Problem ist eine Entscheidungsvariante des Subset-Estimator-Problems. □

Die LXX-Probleme, die wir gleich definieren, finden sich z.B. in folgenden Arbeiten: LMS in Rousseeuw (1984), LQS in Rousseeuw und Leroy (1987), LTS in Rousseeuw und Van Driessen (2002) und LTA in Hawkins und Olive (1999a):

Definition 65 (ε-LXX-Probleme)

`Eingabe:`
Eine Menge $\mathcal{P}$ von n Punkten in d Dimensionen.
`Parameter:`
Ein ε mit $0 < \varepsilon \leq 0.5$.
`Problem:`
Sei $h = \lceil (1-\varepsilon) \cdot n \rceil$. Finde eine Hyperebene mit Parametervektor β, so dass

für	LMS:	f_{LMS}	$=$	$(r(\beta)^2)_{(\lceil n/2 \rceil + \lceil (d+1)/2 \rceil)}$
für	ε-LQS:	$f_{\varepsilon\text{-LQS}}$	$=$	$(r(\beta)^2)_{(h)}$
für	ε-LTS:	$f_{\varepsilon\text{-LTS}}$	$=$	$\frac{1}{h}\sum_{k=1}^{h}(r(\beta)^2)_{(k)}$
für	ε-LTA:	$f_{\varepsilon\text{-LTA}}$	$=$	$\frac{1}{h}\sum_{k=1}^{h}\lvert r(\beta)\rvert_{(k)}$

minimal ist.

Der Funktionswert aller Probleme basiert auf den Residuen. Die h kleinsten Residuen bilden die Menge $S = \{r_i \mid r_i \leq r_{(h)}\}$. Das Residuum eines Punktes, der auf der Hyperebene β liegt, ist gleich Null. Daraus ergibt sich, dass die LXX-Probleme die Definition des Subset-Estimator-Problems erfüllen und wir erhalten:

Theorem 66 *Für jedes feste ε sind die LXX-Probleme NP-hart.*

Definition 67 (ε-MCD und ε-MVE-Problem)

`Eingabe:`
Eine Menge $\mathcal{P}$ von n Punkten in d Dimensionen.
`Parameter:`
Ein ε mit $0 < \varepsilon \leq 0.5$.
`Problem:`
Sei $h = \lceil (1-\varepsilon) \cdot n \rceil$. Finde eine Teilmenge aus h Punkten, so dass

- für ε-MCD: die Determinante der Kovarianz-Matrix
- für ε-MVE: das Volumen des Ellipsoids, das die Punkte umschließt,

minimal ist.

Theorem 68 *Für jedes feste ε ist das MCD- und das MVE-Problem NP-hart.*

Beweis: Beide Probleme haben den Funktionswert Null, falls h Punkte auf einer Hyperebene liegen und sind Spezialfälle des Subset-Estimator-Problems:

- Bei dem MCD-Problem ist die Determinante gegeben durch das Produkt der Eigenwerte der Kovarianzmatrix. Liegen nun alle h Punkte auf einer Hyperebene, so ist ein Eigenvektor gleich Null, woraus sich eine Determinante von Null ergibt. Falls keine h Punkte auf einer Hyperebene liegen, hat die zugehörige Kovarianzmatrix vollen Rang und somit eine Determinante von größer Null.
- Bei dem MVE-Problem ist ein Ellipsoid, das auf einer Hyperebene liegt, flach und hat somit ein Volumen von Null. Ein Ellipsoid, das eine lineare unabhängige Punktmenge umschließt, hat offensichtlich Volumen größer Null.

□

Auch für den CM-Schätzer (Edlund und Ekblom (2005)) und den Stahel-Donoho-Schätzer (Maronna und Yohai (1995)) lässt sich auf ähnliche Weise die *NP*-Härte zeigen, siehe Bernholt (2005).

Exact-Fit-Property

Wir greifen hier die Eigenschaft auf, die wir auf Seite 12 eingeführt haben.

Theorem 69 (Rousseeuw (1994)) *Wenn ein Schätzer regressions- und skalenäquivariant ist und einen Bruchpunkt von $1-c$ hat für Daten in* ***allgemeiner*** *Lage[1], dann erfüllt er die Exact-Fit-Property mit Konstante c.*

Da das Degenerate-Point-Subset-Problem sehr ähnlich zur Exact-Fit-Property ist, legt das Theorem 69 die folgende Gedankenkette nahe:

$$\text{Regressions- und Skalen-Äquivarianz} \Rightarrow \text{Exact-Fit-Property} \Rightarrow \textit{NP}\text{-Härte}$$

Allerdings ist das Problem, dass allgemeine Lage gefordert wird, aber in dem *NP*-Härte-Beweis die Punkte linear abhängig sind. Die Gedankenkette gilt daher leider nicht. In der Statistik ist die allgemeine Lage gegeben, da z.B. in Modellannahmen gefordert wird, dass die $x_1, \ldots, x_{d-1}$-Koordinaten uniform gleichverteilt gewählt werden. Diese Annahmen scheinen nicht mit der Konstruktion eines *NP*-Härte-Beweises vereinbar. In Kapitel 7 werden wir eine neue Suchheuristik skizzieren, die diese Annahme ausnutzt.

[1] bezüglich der $x_1, \ldots, x_{d-1}$-Koordinaten

6.3 Experimente mit Fast-LTS auf schwierigen Datensätzen

Um die praktisch relevanten Auswirkungen der NP-Härte zu beleuchten, beschreiben wir in diesem Abschnitt ein Experiment mit dem LTS-Schätzer, genauer mit der Suchheuristik „Fast-LTS", siehe Rousseeuw und Van Driessen (2002). Wir konstruieren einen Datensatz, so dass der Suchraum ein globales und ein lokales Optimum enthält. Optima, die durch Rauschen verursacht sind, zählen wir nicht. Der Datensatz ist in Abbildung 21 dargestellt und kann auf d Dimensionen erweitert werden.

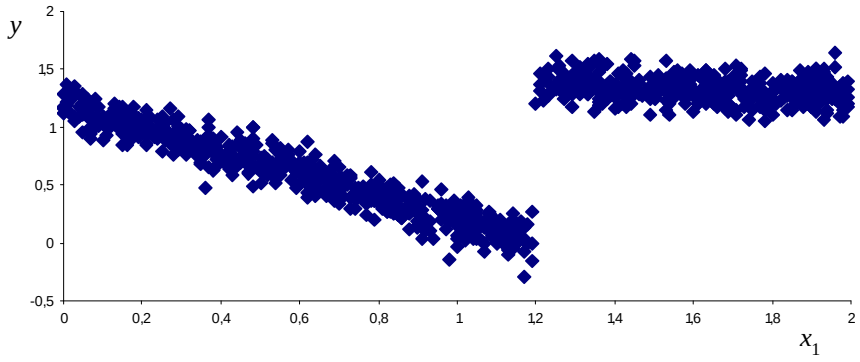


Abbildung 21: Dargestellt ist der schwierige Datensatz. 550 Punkte auf der linken Seite and 450 Punkte auf der rechten. Es ist eine Standard-Abweichung von 0.1 dargestellt, im Experiment wurde jedoch der Wert 0.001 verwendet.

Die Parameter des Datenmodells sind wie folgt:

$$\begin{array}{lll} 550 \text{ Punkte:} & y = \varepsilon_{\text{noise}} - 1 \cdot x_1 + 1.2 & \text{mit } x_1 \sim \text{uniform}(0, 1.2) \\ 450 \text{ Punkte:} & y = \varepsilon_{\text{noise}} - \frac{1}{10} \cdot x_1 + 1.5 & \text{mit } x_1 \sim \text{uniform}(1.2, 2) \end{array}$$

Die Variable $\varepsilon_{\text{noise}} \sim N(0, 0.001^2)$ und die anderen Variablen sind $x_2 \sim \cdots \sim x_d \sim \text{uniform}(-1, 1)$.

Wir erwarten, dass der LTS-Schätzer eine Hyperebene mit den Parametern $\beta_1 = -1$ und $\beta_2 = 0, \ldots, \beta_d = 0, \beta_{d+1} = 1.2$ ausgibt, da die Mehrheit der Punkte dicht an dieser Hyperebene liegen. Der Trick besteht darin, dass wir ein Modell mit zwei Hyperebenen zugrundelegen, die Suchheuristik aber nach einem Modell mit einer Hyperebene suchen lassen. Wir verwenden den Algorithmus `ltsReg()` des `rrcov`-Paketes[2] der R-Software, die eine Implementierung des „Fast-LTS" von Rousseeuw und Van Driessen (2002) ist.

[2]R-Project Website: „http://cran.r-project.org/doc/packages/rrcov.pdf"

Wir setzen den Parameter „alpha" = 0.51, um einen hohen Bruchpunkt zu erhalten. Der Wert „alpha" entspricht dem ε des ε-LTS-Problems aus Definition 6.2 auf Seite 83.

Wir lassen ltsReg 10000 mal laufen und variieren die Dimension des Datensatzes von 5 bis 15, mit einem zusätzlichen Lauf von 18. Um den Fehlerbalken klein zu halten, wurde in 18 Dimensionen die Anzahl der Läufe auf 200000 erhöht. Die gefundenen Lösungen sind in Abbildung 22 dargestellt, wobei exemplarisch der 12-dimensionale Datensatz verwendet wurde.

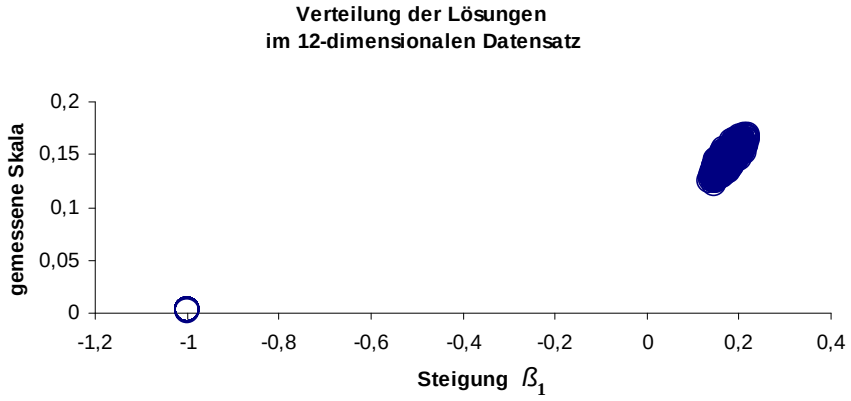


Abbildung 22: Es sind die gefundenen Lösungen des 12-dimensionalen Datensatzes dargestellt. Die meisten Lösungen liegen in der rechten Punktwolke, die ein lokales Optimum ist. Nur einige wenige Lösungen befinden sich in der linken Punktwolke, die das globale Optimum ist.

Wie erwartet haben die Lösungen mit $\beta_1 \approx -1, \beta_2 \approx 0, \ldots, \beta_d \approx 0, \beta_{d+1} \approx 1.2$ eine kleine Skala von ≈ 0.0025. In diesen Fällen wurde das globale Optimum gefunden. Das zweite, lokale Optimum enthält Lösungen mit einer Skala von ≈ 0.15 mit den Werten $\beta_1 \approx 1.05$ und $\beta_{d+1} \approx 0.17$.

Um zu messen, wie oft der Algorithmus das globale Optimum findet, zählen wir die Anzahl der Läufe, die eine Skala kleiner 0.01 gefunden haben. Die Anzahl wird durch 10000 geteilt und wir erhalten die Erfolgswahrscheinlichkeit des Algorithmus. Die Wahrscheinlichkeiten sind in Abbildung 23 dargestellt.

Die Fehlerbalken geben das 1% Quantile an, d.h., wenn die wahre Wahrscheinlichkeit am Ende des Fehlerbalkens liegen würde, so würde diese Messung nur mit einer Wahrscheinlichkeit von weniger als 1% eintreten. Die Fehlerbalken wurden mit Chernoff-Schranken berechnet.

Auf dem 18-dimensionalen Datensatz findet der Algorithmus das globale Optimum nur mit einer Wahrscheinlichkeit von 0,00006. In Abbildung 23 ist die Erfolgswahrscheinlichkeit logarithmisch auf der vertikalen Achse aufgetragen. Man gewinnt den Eindruck, dass sie exponentiell mit der Dimension fällt. Daher ist der Algorithmus für höhere Dimensionen nicht praktikabel.

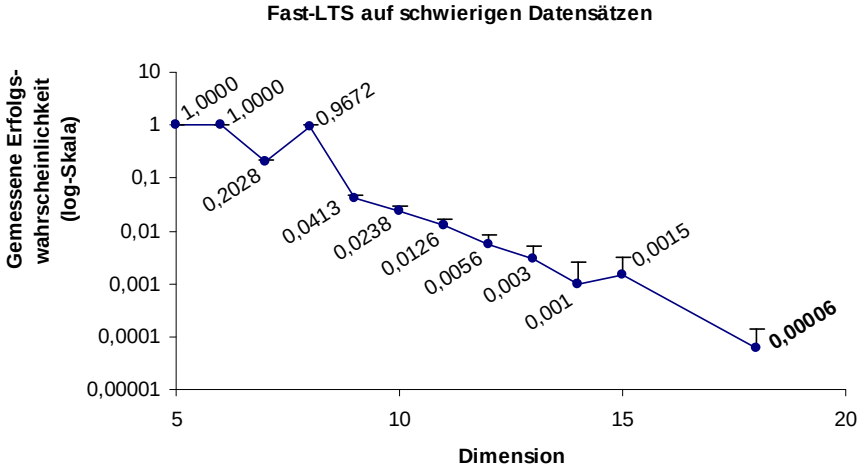


Abbildung 23: Fast-LTS ist eine Suchheuristik, die nur mit einer bestimmten Wahrscheinlichkeit eine Lösung findet. Der Logarithmus der Erfolgswahrscheinlichkeit ist auf der vertikalen Achse aufgetragen. Die Versuche wurden auf Datensätzen der Dimensionen 5 bis 18 durchgeführt.

7 Schlussbemerkungen

Komplexität des RM-Schätzers

Was die Komplexität robuster Schätzer anbelangt, so konnte in dieser Arbeit für viele Schätzer gezeigt werden, dass sie für d-dimensionale Punktmengen *NP*-hart sind. Der Repeated-Median-Schätzer bildet eine Ausnahme und bleibt mysteriös. Für Punkte $p_1, \ldots, p_n$ mit $p_i \in \mathbb{R}^d$ ist der RM-Schätzer für allgemeines d definiert als

$$\beta_j = \operatorname*{med}_{i_1} \ldots \operatorname*{med}_{i_d \neq i_1, \ldots, i_{d-1}} \eta_j(i_1, \ldots, i_d) \ ,$$

wobei $\eta_j(i_1, \ldots, i_d)$ der j-te Steigungsparameter derjenigen Hyperebene ist, die durch die Punkte $p_{i_1}, \ldots, p_{i_d}$ verläuft.

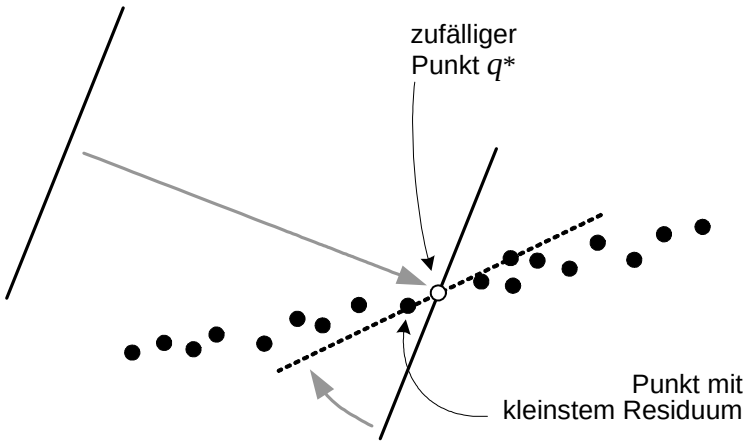
Es ist bereits unklar, ob die Entscheidungsvariante des RM-Schätzers in *NP* enthalten ist, da die naive Rate-Verifikations-Turingmaschine exponentiell viele η raten müsste. Der Median verhält sich wie ein $\exists\forall$-Quantorpaar, was zu der naheliegenden Einordnung in die Komplexitätsklasse Σ_k mit $k = 2d$ der polynomiellen Hierarchie führt. Es scheint schwierig, den beschriebenen *NP*-Härte-Beweis auf den RM-Schätzer anzuwenden. Ein Problem ist, dass die verwendeten Punkte linear abhängig sind, und η für viele Hyperebenen unbestimmt ist. Zudem ist es bei *NP*-harten Problemen natürlich, dass es mehrere Optima geben kann, bei dem RM-Schätzer hingegen gibt es genau ein Optimum. Die komplexitätstheoretische Einordnung des RM-Schätzers ist noch offen.

Neuer Suchoperator

Ein Ansatz, evolutionäre Algorithmen auf den ε-LTS-Schätzer anzuwenden, modelliert ein Individuum wie folgt: Wir wählen d Stützpunkte $q_1, \ldots, q_d$, die den n Punkten der Eingabe entstammen. Aus den Stützpunkten ergibt sich eine Hyperebene β, die genau durch diese Punkte verläuft (lineare Unabhängigkeit vorausgesetzt). Aus der Hyperebene β berechnen sich die Residuen $r_1, \ldots, r_n$ und der Fitnesswert ergibt sich aus der Funktion $f_{\varepsilon\text{-LTS}} = \frac{1}{h} \sum_{k=1}^{h} r_{(k)}(\beta)^2$. Der Mutationsoperator der evolutionären Suche arbeitet wie folgt: Ein zufälliger Stützpunkt wird gegen einen zufälligen Punkt aus der Eingabe getauscht. Falls diese Mutation eine Lösung mit einem besseren oder gleich guten Funktionswert erzielt, wird die neue Hyperebene beibehalten, ansonsten

verworfen. Mit einer Multistart-Strategie, die initial d Punkte zufällig auswählt, lässt sich ein vergleichbares Verhalten zur Fast-LTS-Heuristik erzielen. Bei der schwierigen Eingabe aus Kapitel 6.3 besteht wieder das Problem, dass die Wahrscheinlichkeit, eine gute Startlösung (alle Stützpunkte sind aus optimaler h-Teilmenge) zu treffen, lediglich 2^{-d} beträgt und die Mutation nur eine lokale Suche durchführt.

Bei dem *NP*-Härte-Beweis gibt es die Diskrepanz zur statistischen Anwendung, dass die Punkte des *NP*-Härte-Beweises linear abhängig sind, in der statistischen Anwendung aber allgemeine Lage vorausgesetzt wird. Eine weitere Annahme ist, dass eine h-Teilmenge der Punkte in der Nähe einer gemeinsamen Hyperebene liegt und die Punkte bezüglich der x-Koordinaten z.B. uniform verteilt sind. Dies motiviert den Move-Operator, der wie folgt arbeitet: Er wählt einen zufälligen Punkt q^* der Eingabe und verschiebt die aktuelle Hyperebene parallel, so dass q^* in der Hyperebene liegt. Neben q^* werden als neue Stützpunkte die $d-1$ Punkte mit kleinstem Residuum gewählt. Der Move-Operator ist in folgender Abbildung veranschaulicht:



Die Idee ist, dass mit hoher Wahrscheinlichkeit die $d-1$ Punkte mit kleinstem Residuum der h-Teilmenge entstammen und wir somit eine Hyperebene erhalten, die einen guten Fitnesswert ergibt. Wir nutzen somit die Annahme der Statistik aus, dass die Punkte der h-Teilmenge gleichverteilt auf einer Hyperebene liegen. Eine genauere Untersuchung der Eigenschaften des Move-Operators steht noch aus, aber erste Experimente zeigen, dass damit eine wesentlich bessere Laufzeit erzielt werden kann.

Ideen für neue robuste Schätzer

Auch wenn die Gedankenkette formal nicht gilt, so scheinen hoher Bruchpunkt und Äquivarianz der Grund dafür zu sein, dass die Schätzer in hoher Dimension schwierig zu berechnen sind. Die Äquivarianz-Anforderung fallen zu lassen, haben wir bereits in der Einleitung diskutiert und führt zu praktikablen Analyse-Tools. Eine andere

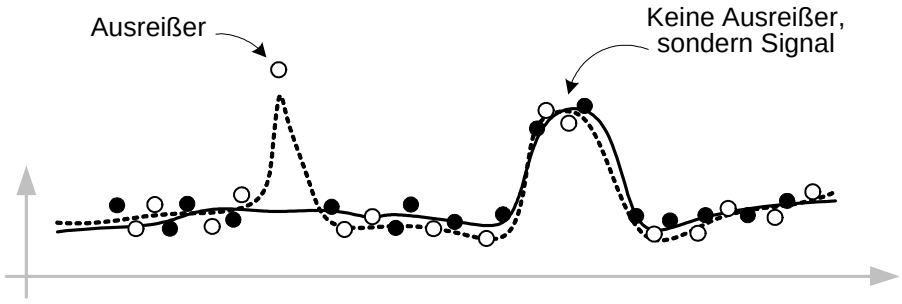
Möglichkeit ist, den hohen Bruchpunkt aufzuweichen. Dies werden wir im Folgenden diskutieren.

Zu Beginn waren Ausreißer als Punkte definiert worden, die eine große Abweichung zu den anderen Datenpunkten aufweisen. Dem Subset-Schätzer hingegen liegt der Gedanke zu Grunde, dass alle Punkte, die nicht dem angenommenen Modell gehorchen, als Ausreißer anzusehen sind, wie das Daten-Beispiel in Kapitel 6.3 veranschaulicht.

Der Gedanke, das Modell einzuschränken, ist meiner Meinung nach ein Irrweg. Lassen wir beliebig flexible Flächen zu, z.B. Splines oder RBF-Kernels, so haben wir es mit Modellwahl zu tun, die nicht auf den Trick aus Kapitel 6.3 hereinfällt. Dort hatten wir Daten aus einem Modell mit zwei Hyperebenen verwendet, die Suchheuristik allerdings nach einem Modell mit einer Hyperebene suchen lassen.

Wir konstruieren den Schätzer derart, dass die Fläche mittels Least-Squares angepasst wird. Dabei stellt sich die Frage, wie viele frei-platzierbare Stützpunkte für die Fläche nötig sind, um eine gute Anpassung an die Daten zu erhalten. Die richtige Anzahl Stützpunkte bestimmen wir, indem wir messen, ob Overfitting vorliegt und zwar wie folgt: Wir ziehen mehrmals zufällige Teilmengen der Größe $n/2$ und messen die Varianz des Schätzers. Bei guter Anpassung an die Daten ist die Varianz des Schätzers wesentlich kleiner als die Varianz der Punkte. Wenn wir die Anzahl der Stützpunkte stark erhöhen, fängt die Fläche an, jeden Punkt einzeln anzupassen und es liegt Overfitting vor. In diesem Fall ist die Varianz des Schätzers größer als die Varianz der Punkte, da jede zufällige Teilmenge die Fläche an sich zieht. Diese Idee wollen wir nutzen, um Ausreißer zu erkennen. Aus meiner Perspektive können Ausreißer an folgendem Verhalten erkannt werden:

Definition 70 *Ausreißer sind Punkte, die lokal Overfitting verursachen.*



Die Idee ist in der obigen Abbildung veranschaulicht. Es sind zwei zufällige Teilmengen, weiße bzw. schwarze Punkte, dargestellt. An den weißen Punkten wurde die gestrichelte Kurve angepasst, und der Ausreißer zieht die gestrichelte Kurve nach oben.

Die schwarze Kurve wird nicht beeinflusst, da sie nur an die schwarzen Punkte angepasst wurde, und die schwarze Kurve hat an dieser Stelle eine große Abweichung zur weißen Kurve. Auf diese Weise kann der Punkt als Ausreißer detektiert werden und der Ansatz könnte es ermöglichen, Ausreißer in polynomieller Zeit zu erkennen.

Fazit

Die Ideen der robusten Statistik funktionieren gut auf Punktmengen in der Ebene. Hier gibt es zahlreiche Arbeiten, die effiziente Algorithmen für robuste Schätzer in der Ebene vorstellen. In dieser Arbeit wurden einige neue Resultate erzielt, wie effiziente Algorithmen für den LQD-Schätzer und das Multiresolutions-Kriterium. Des Weiteren wurden neue Update-Algorithmen für den RM-Schätzer und den MAD-Schätzer vorgestellt.

Für Punktmengen mit Dimension $d > 2$ wurden der LMS- und MCD-Schätzer behandelt und Algorithmen mit besserer Laufzeit im Vergleich zu früheren Arbeiten angegeben. Für variable Dimension d zeigten wir in dieser Arbeit die *NP*-Härte. Dies verdeutlicht die Unvereinbarkeit der Anforderungen „hoher Bruchpunkt", „Äquivarianz", „hohe Dimension" und „schnelle Berechnung". Hier muss ein Mittelweg gefunden werden, der die Anforderungen geeignet aufweicht, um für die Praxis geeignete Schätzer zu erhalten.

Überblick über die Veröffentlichungen und Eigenanteil

1. Thorsten Bernholt, Friedrich Eisenbrand und Thomas Hofmeister (2006)
   A Geometric Framework for Solving Subsequence Problems in Computational Biology Efficiently
   Eingereicht. (siehe Kapitel 3.2)

2. Thorsten Bernholt und Thomas Hofmeister (2006)
   An Algorithm for a Generalized Maximum Subsequence Problem
   Proceedings of the 7^{th} Latin American Symposium (LATIN 2006),
   Seiten 178 - 189.
   (siehe Kapitel 3.2)

3. Roland Fried, Thorsten Bernholt, Ursula Gather und Ingo Wegener (2006)
   Modified Repeated Median Filters
   Statistics and Computing 16, Seiten 177-192, Springer.
   (siehe Kapitel 4.1)

4. Thorsten Bernholt (2005)
   Robust Estimators are Hard to Compute
   Technical Report 52/2005, Sonderforschungsbereich 475, Universität Dortmund
   (siehe Kapitel 6)

5. Thorsten Bernholt, Robin Nunkesser und Karen Schettlinger (2005)
   Computing the Least Quartile Difference Estimator in the Plane
   unter Vorbehalt angenommen bei Computational Statistics & Data Analysis.
   (siehe Kapitel 3.1)

6. Thorsten Bernholt (2005)
   Computing the Least Median of Squares Estimator in Time $O(n^d)$
   Proceedings of the International Conference on Computational Science and Its Applications (ICCSA 2005), Seiten 697-706.
   (siehe Kapitel 5.1)

7. Roland Fried, Thorsten Bernholt und Ursula Gather (2004)
   Repeated Median and Hybrid Filters
   Computational Statistics & Data Analysis 50 (9), Seiten 2313-2338, Elsevier.

8. Thorsten Bernholt und Paul Fischer (2004)
   The Complexity of Computing the MCD-Estimator
   Theoretical Computer Science 326, Seiten 383-398, Elsevier.
   Auch: Computing Science and Statistics 33: Proceedings of the 33rd Symposium on the Interface 2001.
   (siehe Kapitel 5.2)

9. Thorsten Bernholt und Roland Fried (2003)
   Computing the Update of the Repeated Median Regression Line in Linear Time
   Information Processing Letters 88 (3), Seiten 111-117, Elsevier.
   (siehe Kapitel 4.2)

Eigenanteil des Doktoranden

Veröffentlichung	Eigenanteil
1.	33%
2.	50%
3.	25%
4.	100%
5.	45%
6.	100%
7.	33%
8.	50%
9.	95%

Literaturverzeichnis

Agulló, J. (2002): An exchange algorithm for computing the least quartile difference estimator, *Metrika*, **55**, Seiten 3–16.

Allison, L. (2003): Longest biased interval and longest non-negative sum interval, *Bioinformatics Application Note*, **19**(10), Seiten 1294–1295.

Amaldi, E. und Kann, V. (1995): The complexity and approximability of finding maximum feasible subsystems of linear relations, *Theoretical Computer Science*, **147**, Seiten 181–210.

Amenta, N., Bern, M., Eppstein, D. und Teng, S.-H. (2000): Regression depth and center points, *Discrete & Computational Geometry*, **23**(3), Seiten 305–323.

Barnett, V. und Lewis, T. (1994): *Outliers in Statistical Data (Third Edition)*, Wiley.

Becker, C. und Gather, U. (2001): The largest nonidentifiable outlier: A comparison of multivariate simultaneous outlier identification rules, *Computational Statistics and Data Analysis*, **36**, Seiten 119–127.

Bengtsson, F. und Chen, J. (2004): Efficient algorithms for k maximum sums, in: *Proceedings of the* 15^{th} *International Symposium on Algorithms and Computation (ISAAC 2004)*, Seiten 137–148.

Bentley, J. (1984): *Programming Pearls*, Addison-Wesley.

Bernholt, T. (2005): Robust estimators are hard to compute, *Technical Report 52/2005, Sonderforschungsbereich 475, Universität Dortmund.*

Boyd, S. und Vandenberghe, L. (2004): *Convex Optimization*, Cambridge University Press.

Chan, T. M. (1999): Geometric applications of a randomized optimization technique, *Discrete & Computational Geometry*, **22**(4), Seiten 547–567.

Chazelle, B. (1986): Reporting and counting segment intersections, *Journal of Computer and Systems Science*, **32**, Seiten 156–182.

Chen, T., Jaffe, J. D. und Church, G. M. (2001): Algorithms for identifying protein cross-links via tandem mass spectrometry, *Journal of Computational Biology*, **8**(6), Seiten 571–583.

Chien, H. und Steiger, W. (1995): Some geometric lower bounds, in: *Proceedings of the* 6^{th} *International Symposium on Algorithms and Computation(ISAAC '95)*, Seiten 72–81.

Cole, R., Sharir, M. und Yap, C. K. (1987): On k-hulls and related problems, *SIAM Journal on Computing*, **16**(1), Seiten 61–77.

Cormen, T. H., Leiserson, C. E., Rivest, R. L. und Stein, C. (2001): *Introduction to Algorithms (Second Edition)*, MIT Press & McGraw-Hill.

Croux, C., Rousseeuw, P. J. und Hössjer, O. (1994): Generalized S-estimators, *Journal of the American Statistical Association*, **89**, Seiten 1271–1281.

Davies, P. L., Fried, R. und Gather, U. (2003): Robust signal extraction for on-line monitoring data, *Journal of Statistical Planning and Inference*, **122**, Seiten 65–78.

Davies, P. L. und Kovac, A. (2001): Local extremes, runs, strings and multiresolution (with discussion), *Annals of Statistics*, **29**(1), Seiten 1–65.

de Berg, M., van Kreveld, M., Overmars, M. und Schwarzkopf, O. (2000): *Computational Geometry: Algorithms and Applications*, Springer.

Donoho, D. L. und Huber, P. J. (1983): The notion of breakdown point, in: Bickel, P. J., Doksum, K. A. und Hodges jr., J. L. (Hrsg.): *A Festschrift for Erich L. Lehmann*, Seiten 157–184, Wadsworth, Belmont, CA.

Dor, D. und Zwick, U. (1999): Selecting the median, *SIAM Journal on Computing*, **28**(5), Seiten 1722–1758.

Dryden, I. L. und Walker, G. (1999): Highly resistant regression and object matching, *Biometrics*, **55**(3), Seiten 820–825.

Edelsbrunner, H. und Souvaine, D. L. (1990): Computing least median of squares regression line and guided topological sweep, *Journal of the American Statistical Association*, **85**, Seiten 115–119.

Edlund, O. und Ekblom, H. (2005): Computing the constrained M-estimates for regression, *Computational Statistics and Data Analysis*, **49**, Seiten 19–32.

Erickson, J., Har-Peled, S. und Mount, D. M. (2004): On the least median square problem, in: *Proceedings of the* 20^{th} *ACM Symposium on Computational Geometry (SoCG 2004)*, Seiten 273–297.

Eskin, E., Arnold, A., Prerau, M., Portnoy, L. und Stolfo, S. (2002): A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data, in: Barbara, D. und Jajodia, S. (Hrsg.): *Applications of Data Mining in Computer Security*, Kluwer.

Fahrmeir, L., Künstler, R., Pigeot, I. und Tutz, G. (2002): *Statistik - Der Weg zur Datenanalyse (4. Auflage)*, Springer.

Fan, T.-H., Lee, S., Lu, H.-I., Tsou, T.-S., Wang, T. C. und Yao, A. (2003): An optimal algorithm for maximum-sum segment and its application in bioinformatics, in: *Proceedings of the* 8^{th} *International Conference on Implementation and Application of Automata (CIAA 2003)*, Seiten 251–257.

Filzmoser, P., Garrett, R. G. und Reimann, C. (2005): Multivariate outlier detection in exploration geochemistry, *Computer & Geoscience*, **31**, Seiten 579–587.

Gajentaan, A. und Overmars, M. H. (1995): On a class of $O(n^2)$ problems in computational geometry, *Computational Geometry: Theory and Applications*, **5**, Seiten 165–185.

Garey, M. R. und Johnson, D. S. (1979): *Computers and Intractability — A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, New York.

Goldwasser, M. H., Kao, M.-Y. und Lu, H.-I. (2005): Linear-time algorithms for computing maximum-density sequence segments with bioinformatics applications, *Journal of Computer and System Science*, **70**(2), Seiten 128–144.

Har-Peled, S. (1998): Constructing cuttings in theory and practice, in: *Proceedings of the* 14^{th} *ACM symposium on computational geometry (SoCG '98)*, Seiten 327–336.

Hawkins, D. M. und Olive, D. J. (1999a): Applications and algorithms for least trimmed sum of absolute deviations regression, *Computational Statistics and Data Analysis*, **32**, Seiten 119–134.

Hawkins, D. M. und Olive, D. J. (1999b): Improved feasible solution algorithms for high breakdown estimation, *Computational Statistics and Data Analysis*, **30**, Seiten 1–11.

Horn, R. A. und Johnson, C. R. (1985): *Matrix Analysis*, Cambridge University Press.

Hubert, M. und Engelen, S. (2004): Robust PCA and classification in biosciences, *Bioinformatics*, **20**, Seiten 1728–1736.

Kiwiel, K. C. (2004): Improved randomized selection, *Computing Research Repository (CoRR)*, **cs.DS/0402005**.

Klein, R. (2004): *Algorithmische Geometrie (2. Auflage)*, Springer.

Kumar, A., Sabharwal, Y. und Sen, S. (2004): A simple linear time $(1+\varepsilon)$-approximation algorithm for k-means clustering in any dimensions, in: *Proceedings of the* 45^{th} *IEEE Symposium on Foundations of Computer Science (FOCS 2004)*, Seiten 454–462.

Li, Y., Xu, L.-Q., Morrison, G., Nightingale, C. und Morphett, J. (2003): Robust panorama from MPEG video, in: *Proceedings of the 2003 IEEE International Conference on Multimedia and Expo (ICME 2003)*, Seiten 81–84.

Lin, Y.-L., Jiang, T. und Chao, K.-M. (2002): Efficient algorithms for locating the length-constrained heaviest segments with applications to biomolecular sequence analysis, *Journal of Computer and System Science*, **65**(3), Seiten 570–586.

Maronna, R. A. und Yohai, V. J. (1995): The behavior of the stahel-donoho robust multivariate estimator, *Journal of the American Statistical Association*, **90**(429), Seiten 330–341.

Matoušek, J., Mount, D. M. und Netanyahu, N. (1998): Efficient randomized algorithms for the repeated median line estimator, *Algorithmica*, **20**(2), Seiten 136–150.

Megiddo, N. (1979): Combinatorial optimization with rational objective functions, *Mathematics of Operations Research*, **4**(4), Seiten 414–424.

Mount, D. M., Netanyahu, N. S., Piatko, C. D., Silverman, R. und Wu, A. Y. (2000): Quantile approximation for robust statistical estimation and k-enclosing problems, *International Journal of Computational Geometry and Applications*, **10**, Seiten 593–608.

Mount, D. M., Netanyahu, N. S., Romanik, K., Silverman, R. und Wu, A. Y. (1997): A practical approximation algorithm for the LMS line estimator, in: *Proceedings of the* 8^{th} *ACM-SIAM Symposium on Discrete Algorithms (SODA '97)*, Seiten 473–482.

Pesch, C. (2000): *Eigenschaften des gegenüber Ausreißern robusten MCD-Schätzers und Algorithmen zu seiner Berechnung*, dissertation.de-Verlag im Internet GmbH, Berlin.

Pizarro, O. und Singh, H. (2003): Toward large-area mosaicing for underwater scientific applications, *IEEE Journal of Oceanic Engineering*, **28**(4), Seiten 651–672.

Plets, H. und Vynckier, C. (1999): An analysis of the incidence of the VEGA phenomenon among main-sequence and POST main-sequence stars, *Astronomy and Astrophysics*, **343**, Seiten 496–506.

Rafalin, E., Souvaine, D. und Streinu, I. (2002): Topological sweep in degenerate cases, in: *Proceedings of the* 4^{th} *International Workshop on Algorithm Engineering and Experiments (ALENEX 2002)*, Seiten 155–165.

Rocke, D. M. und Woodruff, D. L. (1993): Computation of robust estimates of multivariate location and shape, *Statistica Neerlandica*, **47**(1), Seiten 27–42.

Rocke, D. M. und Woodruff, D. L. (1999): A synthesis of outlier and cluster identification, *unveröffentlicht*.

Roos, T. und Widmayer, P. (1994): k-violation linear programming, *Information Processing Letters*, **52**(2), Seiten 109–114.

Rousseeuw, P. J. (1984): Least median of squares regression, *Journal of the American Statistical Association*, **79**, Seiten 871–880.

Rousseeuw, P. J. (1994): Unconventional features of positive-breakdown estimators, *Statistics & Probability Letters*, **19**, Seiten 417–431.

Rousseeuw, P. J. und Leroy, A. M. (1987): *Robust Regression and Outlier Detection*, Wiley.

Rousseeuw, P. J. und van Driessen, K. (1999): A fast algorithm for the minimum covariance determinant estimator, *Technometrics*, **41**, Seiten 212–223.

Rousseeuw, P. J. und Van Driessen, K. (2002): Computing LTS regression for large data sets, *Estadistica*, **54**, Seiten 163–190.

Siegel, A. F. (1982): Robust regression using repeated medians, *Biometrika*, **69**(1), Seiten 242–244.

Stein, A. und Werman, M. (1992): Finding the repeated median regression line, in: *Proceedings of the* 3^{rd} *ACM-SIAM Symposium on Discrete Algorithms (SODA '92)*, Seiten 409–413.

Stromberg, A. J. (1993): Computing the exact least median of squares estimate and stability diagnostics in multiple linear regression, *SIAM Journal on Scientific Computing*, **14**(6), Seiten 1289–1299.

Toussaint, G. T. (1983): Solving geometric problems with rotating calipers, in: *Proceedings of the* 2^{nd} *IEEE Mediterranean Electrotechnical Conference (MELECON '83)*, Seiten A10.02/1–4.

van Oostrum, R. und Veltkamp, R. C. (2004): Parametric search made practical, *Computational Geometry: Theory and Applications*, **28**, Seiten 75–88.

Wand, J. N., Shotts, K. W., Sekhon, J. S., Mebane Jr., W. R., Herron, M. C. und Brady, H. E. (2001): The butterfly did it: The aberrant vote for Buchanan in Palm Beach County, Florida, *American Political Science Review*, **95**(4), Seiten 793–810.

Wegener, I. (2003): *Komplexitätstheorie: Grenzen der Effizienz von Algorithmen*, Springer.

Zhao, H.-Y., Yue, P. Y. K. und Fang, K.-T. (2004): Identification of differentially expressed genes with multivariate outlier analysis, *Journal of Biopharmaceutical Statistics*, **14**(3), Seiten 629–646.