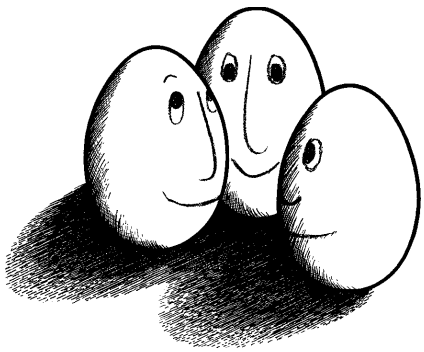
technische universität
dortmund

Bachelorarbeit

# Datenzusammenfassungen auf Datenströmen

Maik Schmidt
Juli 2017



Gutachter:
Prof. Dr. Katharina Morik
Sebastian Buschjäger (M.Sc.)

Technische Universität Dortmund
Fakultät für Informatik
Lehrstuhl für Künstliche Intelligenz (LS VIII)
https://www-ai.cs.tu-dortmund.de

# Inhaltsverzeichnis

# Kapitel 1

# Einleitung

## 1.1 Motivation

Der Begriff *Big Data* umfasst zu große oder zu komplexe Datenmengen, aber auch Daten, die zu schnell eintreffen, um sie per Hand zu analysieren. Maschinelle Hilfe ist für eine Echtzeitanalyse notwendig.
Eine Möglichkeit besteht in der Zusammenfassung der Daten. Das Ziel dabei ist die Extraktion einer repräsentativen Teilmenge der Daten, die den Großteil der Informationen beibehält, aber viel einfacher zu handhaben ist. Die Repräsentativität einer Zusammenfassung kann mit einer submodularen Funktion berechnet werden, die je nach Zielsetzung Eigenschaften wie Abdeckung, Vielfalt oder Informationsgehalt berücksichtigen kann. Die optimale Zusammenfassung maximiert diese Zielfunktion unter Beschränkung der Kardinalität.
Submodulare Funktionen sind Funktionen mit einer *diminishing returns* Eigenschaft und werden oft als diskretes Analog konvexer Funktionen beschrieben. Sie spielen eine fundamentale Rolle in kombinatorischer Optimierung, Wirtschaft, Spieltheorie und vielen weiteren Gebieten und wurden erst in den letzten Jahren ausgiebig auf dem Gebiet des Maschinellen Lernens untersucht. Zahlreiche Probleme wie die exemplarbasierende Clusterung und das aktive Lernen können als Optimierung einer submodularen Funktion modelliert werden.
Besondere Schwierigkeit liegt in der Berechnung der Zusammenfassung in einer Datenstromumgebung, die den Einsatz von Datenstromalgorithmen mit begrenzter Speichernutzung und Berechnungszeit pro eintreffendem Element erfordert. Ein typische Quelle für Datenströme sind Sensor- und Netzwerkdaten. Weil die Elemente eines Datenstroms kontinuierlich eintreffen, muss direkt über Verwerfung oder Aufnahme in die Zusammenfassung entschieden werden. Die Submodularität der Zielfunktion hilft bei dieser Entscheidung.

## 1.2 Verwandte Arbeiten

Die systematische Untersuchung von submodularen Funktionen begann 1970 von Edmonds, der Ähnlichkeiten zwischen Submodularität und Konvexität feststellte [13]. Aufbauend darauf formulierte Lovász 1983 eine konvexe Relaxation von submodularen Funktionen [39]. Die Verwandtheit mit der Konvexität ist algorithmisch ausnutzbar, sodass eine uneingeschränkte Minimierung von submodularen Funktionen in polynomieller Zeit durch die Ellipsoid Methode [22] und weitere kombinatorische Methoden [24, 46] möglich ist. Über die Minimierung mit Einschränkungen ist weniger bekannt. Für einige Einschränkungen existieren polynomielle Algorithmen [20, 48], für weitere sind nur Approximationsalgorithmen bekannt [19, 25].
Intensiver wurden approximative Methoden für die submodulare Maximierung untersucht, weil sie im Allgemeinen $NP$-schwer ist. Die Untersuchung begann 1978 mit der Arbeit von Nemhauser et al., in der sie einen einfachen *Greedy* Algorithmus vorstellten, der für monoton submodulare Funktionen unter Kardinalitätsbeschränkung eine $1 - \frac{1}{e} \approx 63\%$-Approximation liefert [45]. Feige zeigte 1998, dass dies die bestmögliche Approximationsgarantie für das Problem ist [14]. Für komplexere Einschränkungen folgten Varianten des *Greedy* Algorithmus [16, 30]. Die Maximierung von nicht-monotonen submodularen Funktionen ist schwieriger. Erst 2015 präsentierten Buchbinder et al. eine $\frac{1}{2}$-Approximation für die uneingeschränkte Maximierung [9]. Diese Schranke war bereits als bestmögliche Approximationsgarantie bekannt [15]. Für viele Einschränkungen existieren weitere Algorithmen mit kleineren Approximationsgarantien [33, 7].
Um mit dem exponentiellen Wachstum der generierten Daten mitzuhalten, ist die Entwicklung neuer Verfahren notwendig. Zum Einen gibt es Bemühungen zur Parallelisierung oder verteilten Verarbeitung der immer größer werdenden Datenmengen [43, 3]. Diese Verfahren beschränken die Speicherkosten an jedem Berechnungsknoten und die Kommunikation zwischen ihnen. Sie erstreben die Zusammenfassung riesiger Datenmengen mit vergleichbaren Approximationsgarantien zur (unberechenbaren) zentralisierten Berechnung.
Zum Anderen gehören dazu Ansätze, die eine Echtzeitverarbeitung von Datenströmen ermöglichen. Zu den Datenstromalgorithmen zählt der 2014 von Badanidiyuru et al. veröffentlichte *Sieve Streaming* Algorithmus [2]. Durch Verwaltung mehrerer Lösungen garantiert dieser bei konstantem Speicherverbrauch eine $\frac{1}{2} - \epsilon$-Approximation für die monoton submodulare Maximierung unter Kardinalitätsbeschränkung. Buchbinder et al. zeigen 2015, dass für dieses Problem keine bessere Approximationsgarantie als $\frac{1}{2}+\varepsilon$ mit $\varepsilon > 0$ möglich ist [8]. Auch für komplexere Einschränkungen [10, 11] wurden Datenstromalgorithmen entwickelt. Chekuri et al. entwarfen 2015 Datenstromalgorithmen mit neuen Approximationsgarantien für die nicht-monotone submodulare Maximierung [11].

## 1.3 Aufbau der Arbeit

In dieser Arbeit wird die Berechnung von Zusammenfassungen auf Datenströmen untersucht. Dieses Problem kann als Maximierung einer monoton submodularen Funktion unter Kardinalitätsbeschränkung modelliert werden. Dafür wird der *Sieve Streaming* Algorithmus verwendet, der auf dem aktuellen Forschungsstand mit $\frac{1}{2} - \epsilon$ die höchste Approximationsgarantie bietet.
Die Arbeit lässt sich folgendermaßen kapitelweise zusammenfassen:

- **Kapitel 2:** Die bereits erwähnten submodularen Funktionen und wichtige Eigenschaften dieser werden formal eingeführt. Insbesondere wird ihre Verwandtschaft mit konvexen Funktion und ihre Rolle im Maschinellen Lernen diskutiert. Außerdem wird auf zwei Beispiele für Zielfunktionen eingegangen, die in Datenstromumgebungen genutzt werden können.

- **Kapitel 3:** Nach der Klärung des Begriffs der Approximationsgarantie werden zwei Approximationsalgorithmen für die monoton submodulare Maximierung unter Kardinalitätsbeschränkung vorgestellt. Auf den populären *Greedy* Algorithmus mit seiner *state-of-the-art* Implementierung *Lazy Greedy* folgt der Datenstromalgorithmus *Sieve Streaming*. Außerdem wird auch *Reservoir Sampling* als Methode für eine zufällige Auswahl einer Zusammenfassung genannt, die in dieser Arbeit für verschiedene Zwecke verwendet wird.

- **Kapitel 4:** Hier wird auf die Implementierung des *Sieve Streaming* Algorithmus unter Berücksichtigung von Laufzeit und Speicher eingegangen. Insbesondere ermöglicht die Abschätzung der möglichen Zielfunktionswerte eine Verkleinerung des Siebintervalls. Für zwei Zielfunktionen werden konkrete Implementierungshinweise zur Reduzierung des Berechnungsaufwandes gegeben. Außerdem wird kurz die Parallelisierbarkeit des Algorithmus untersucht.

- **Kapitel 5:** Es folgt eine Anwendung der vorgestellten Implementierung des *Sieve Streaming* Algorithmus auf reale Probleme. Im ersten Experiment wird die Performanz mit originalem und mit verkleinertem Siebintervall bei der Zusammenfassung von Sprachaufnahmen verglichen. Das zweite Experiment untersucht die Güte des *Sieve Streaming* Algorithmus bei der Erkennung von Anomalien. Durch die Maximierung der Vielfalt werden hier Teleskopdaten nach Ausreißer durchsucht.

- **Kapitel 6:** Abschließend wird ein Fazit aus der Arbeit gezogen und ein kurzer Ausblick gegeben.

# Kapitel 2

# Submodulare Funktionen

In diesem Kapitel werden zuerst wichtige Grundbegriffe der submodularen Funktionen definiert und erklärt, warum diese Funktionen wichtig sind. Dann werden Eigenschaften und Zusammenhänge mit der Konvexität diskutiert und mögliche Anwendungen erläutert. Am Ende des Kapitel wird auf zwei Beispiele für submodulare Funktionen eingegangen, die auf Datenströmen einsetzt werden können. Umfangreichere Übersichten über die submodulare Optimierung werden in [18, 28] gegeben.
Die Submodularität von Mengenfunktionen hat weitreichende theoretische und praktische Konsequenzen. Sie macht unlösbare diskrete Optimierungsprobleme einfach lösbar oder approximierbar. Submodulare Funktionen spielen eine große Rolle in kombinatorischer Optimierung, Wirtschaft, Spieltheorie und vielen weiteren Gebieten. Auch im Maschinellen Lernen können zahlreiche Probleme wie exemplarbasierende Clusterung oder aktives Lernen als Optimierung einer submodularen Funktion modelliert werden.
Die Datenzusammenfassung sucht eine kleine Teilmenge der Daten, die eine Gütefunktion maximieren. Dabei wird die Menge aller zulässigen Lösungen meistens eingeschränkt. Eine einfache und häufig benutzte Wahl der Einschränkung (*constraint*) ist die Kardinalitätsbeschränkung $|S| \leq k$ für eine natürliche Zahl $k$. Sei $V$ die Grundmenge der Daten und $f : 2^V \to \mathbb{R}$ eine Mengenfunktion mit $f(\emptyset) = 0$, welche die Güte von Teilmengen $S \subseteq V$ angibt. Dann kann dieses Problem als kombinatorisches Optimierungsproblem formuliert werden:

$$\max_{|S| \leq k} f(S)\,.$$

Ohne Annahmen über die Funktion $f$ ist dieses Problem $NP$-schwer. Gute Approximationen sind möglich, sobald $f$ als eine submodulare Funktion gewählt wird. Bessere Approximationen sind möglich, wenn die Funktion zusätzlich monoton ist. Diese Begriffe werden jetzt formal definiert.

## 2.1 Grundbegriffe

Submodulare Funktionen sind eine Klasse der diskreten Mengenfunktionen, das sind Funktionen $f : 2^V \to \mathbb{R}$, die jeder Teilmenge $S \subseteq V$ eine Wert zuweisen. Für die Definition der Submodularität wird zuerst der Begriff des Ertrags benötigt. Dabei geht es um den absoluten Gewinn im Funktionswert, der durch die Aufnahme eines Elements in eine Menge entsteht.

**2.1.1 Definition ([28], Ertrag).** Für eine Mengenfunktion $f : 2^V \to \mathbb{R}$ ist der Ertrag (*discrete derivative, marginal gain*) eines Elements $e \in V$, dass zu einer Menge $S \subseteq V$ hinzugefügt wird,

$$\Delta_f(e|S) := f(S \cup \{e\}) - f(S)\,.$$

Eine angenehme Eigenschaft ist die Monotonie von Funktionen, die Optimierungsprobleme erheblich vereinfacht. Diese Eigenschaft lässt sich auch auf Mengenfunktionen definieren.

**2.1.2 Definition ([28], Monotonie).** Eine Mengenfunktion $f : 2^V \to \mathbb{R}$ ist monoton, wenn für alle Mengen $S \subseteq S' \subseteq V$ gilt:

$$f(S) \leq f(S')\,.$$

Mithilfe des Ertrags lässt sich eine äquivalente Definition formulieren: Die Funktion $f$ ist genau dann monoton, wenn für alle $S \subseteq V$ und $e \in V$ gilt:

$$\Delta_f(e|S) \geq 0\,.$$

Der Funktionswert monotoner Zielfunktionen kann durch die Aufnahme von Elementen nicht sinken. Bei monotonen Funktionen ist die Formulierung einer Einschränkung wie der Kardinalitätsbeschränkung notwendig, da das Optimierungsproblem sonst trivial ist. Die Monotonie ist unter Kardinalitätsbeschränkung bei vielen Anwendungen eine vernünftige Annahme und ermöglicht höhere Approximationsgarantien. Mit dem Ertrag kann jetzt die Submodularität definiert werden.

**2.1.3 Definition ([28], Submodularität).** Eine Mengenfunktion $f : 2^V \to \mathbb{R}$ ist submodular, wenn für alle $S \subseteq S' \subseteq V$ und $e \in V \setminus S'$ gilt:

$$\Delta_f(e|S) \geq \Delta_f(e|S')\,.$$

Ist die Funktion $f$ zusätzlich monoton, dann ist die Ungleichung sogar für alle $e \in V$ erfüllt.

Die Submodularität charakterisiert also Funktionen, deren Erträge nach Hinzufügen von neuen Elementen nicht steigen dürfen. Diese Eigenschaft wird häufig mit sinkenden Erträgen (*diminishing returns*) beschrieben. Abbildung 2.1 illustriert diesen Effekt für eine

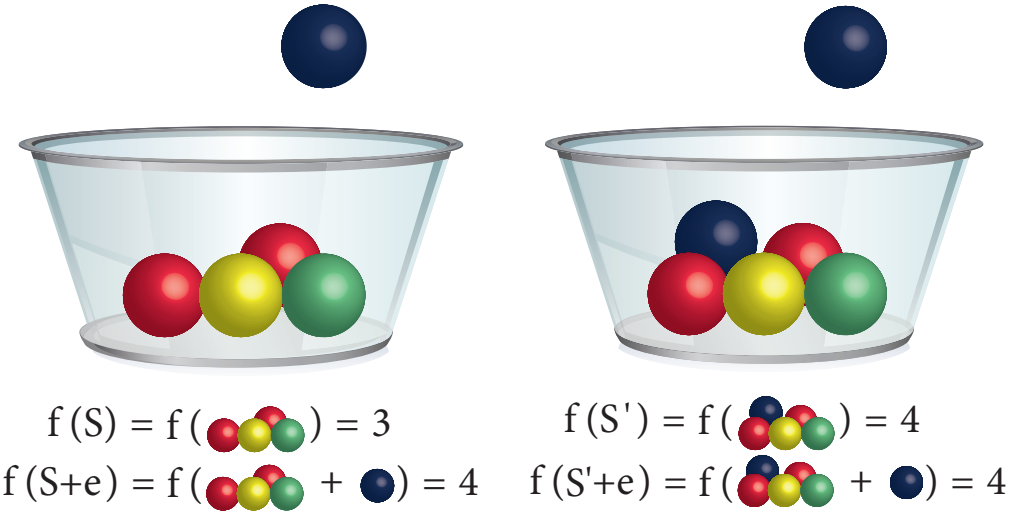



**Abbildung 2.1:** Illustration der submodularen Eigenschaft von Mengenfunktionen [35]. Der Ertrag des blauen Balls ist für den linken Behälter größer als für den rechten Behälter.

Funktion, welche die Anzahl der Farben von Bällen in einem Behälter zählt. Der linke Behälter enthält die Menge $S$ mit drei verschiedenen Farben. Durch Hinzufügen des blauen Balls, steigt die Anzahl der Farben auf vier. Der Ertrag des neuen Elements liegt also bei eins. Der rechte Behälter enthält die Obermenge $S' \supseteq S$ mit vier verschiedenen Farben. Durch Hinzufügen des blauen Balls zu dieser Menge bleibt die Anzahl der Farben identisch, der Ertrag liegt bei null. Die Farben zählende Funktion ist (monoton) submodular.
Weitere Beispiele für submodulare Funktionen tauchen beim *facility-location*-Problem, beim *set-covering*-Problem, sowie als Shannon-Entropie und Transinformation von Zufallsvariablen auf. Mit diesen Funktionen kann eine große Anzahl an realen Problem modelliert werden.

## 2.2 Eigenschaften submodular Funktionen

Submodulare Funktionen sind eine Verallgemeinerung modularer Funktionen. Diese erfüllen die submodulare Eigenschaft mit Gleichheit, sodass ein konstanter Ertrag folgt: $\Delta(e|S) = \Delta(e|S')$ für $e \notin S \cup S'$. Sie werden als diskretes Analog linearer Funktionen angesehen. Ein Beispiel für eine modulare Funktionen ist die Kardinalität: $f(S) = |S|$, welche allen Elementen den Ertrag eins zuweist. Auf der anderen Seite der submodularen Funktion stehen die supermodularen Funktionen. Diese teilen die Eigenschaft von steigenden Erträgen: $\Delta(e|S) \leq \Delta(e|S')$ für $S \subseteq S'$. Ihr Vorkommen beschränkt sich hauptsächlich auf die Spieltheorie. Eine Funktion $f$ ist genau dann submodular, wenn $-f$ supermodular ist. Wenn sie gleichzeitig sub- und supermodular ist, dann ist sie auch modular.
Die Konstruktion von mächtigen submodularen Funktionen aus modularen und einfachen

submodularen Funktionen wird durch die Abgeschlossenheit der Submodularität unter vielen Operationen ermöglicht. Einige der Operationen werden im Folgenden genannt:

- Nicht-negative Linearkombination: $\sum_i a_i f_i(S)$ für $a_i \geq 0$
- Komplement: $f(V \setminus S)$
- Trunkierung: $\min\{f(S), a\}$

Andersherum lassen sich submodulare Funktionen oft in einfachere Bestandteile zerlegen, die wieder submodular sind. Diese Methode kann zum Beispiel bei der Approximation von submodularen Funktionen helfen.

## 2.3 Submodulare Funktionen im Maschinellen Lernen

Submodulare Funktionen werden oft mit konvexen Funktionen assoziiert. Die Lovász-Erweiterung [39] formalisiert die Verwandtschaft zwischen der Submodularität und der Konvexität, indem sie submodulare Funktionen auf konvexe, stückweise lineare Funktionen erweitert. Weitere Gemeinsamkeiten zeigte Fujishige durch die Gültigkeit konvexer Dualitätstheoreme für submodulare Funktionen [17, 18]. Diese Erkenntnisse rechtfertigen die Betrachtung der Submodularität als diskretes Analog der Konvexität.
Weil viele Optimierungsprobleme im Maschinellen Lernen konvex sind, haben konvexe Funktionen hier eine enorme Anwendbarkeit. Die Rolle der submodularen Funktionen im Maschinellen Lernen ist bisher wesentlich unklarer, obwohl dort unzählige kombinatorische Probleme auftreten. Bei der Clusterung [44] und der Zusammenfassung von Dokumenten [36] und Bildern [56] sind Repräsentanten der Daten von Interesse. Bei der Merkmalsselektion [38] und beim aktiven Lernen [60] werden Teilmenge unter Maximierung des Informationsgehalts gesucht. Die Platzierung von Sensoren [32] erfordert die Suche nach geeigneten Positionen. Die Bildsegmentierung [5] beschäftigt sich mit der Suche nach Vordergründen in Bildern. Die Auswahl einer Teilmenge kann in jedem dieser Probleme als submodulares Optimierungsproblem modelliert werden. Dabei können Eigenschaften wie Abdeckung, Diversität oder Informationsgehalt berücksichtigt werden. Für diese herausfordernden Probleme des Maschinellen Lernens hat die Untersuchung von submodulare Funktionen ein hohes Potenzial.

## 2.4 Zielfunktionen für Datenströme

Essenziell für die Qualität der berechneten Zusammenfassung ist die Wahl der Zielfunktion. In Datenstromumgebungen ist die Auswahl deutlich eingeschränkt, weil sie unabhängig von der Grundmenge $V$ der Daten auswertbar sein muss. Badanidiyuru et al. [2] schlagen dafür zwei Zielfunktion vor.

### 2.4.1 Clusterungsbasierend

Für eine Zusammenfassung mit hoher Abdeckung eignen sich die Clusterzentren einer exemplarbasierenden Clusterung. Das K-Medoid-Problem [26] erstrebt eine Selektion $S$ aus $k$ Elementen, sodass der Abstand zwischen den Daten und den nächstgelegenen selektierten Elementen minimal ist. Dieses Ziel wird durch die Minimierung der Verlustfunktion

$$L(S) = \frac{1}{|V|} \sum_{e \in V} \min_{v \in S} d(e, v)$$

beabsichtigt. Die Distanz zwischen zwei Elementen wird durch einen beliebige nicht-negative Distanzfunktion $d$ angegeben. Aus der Verlustfunktion und einem Vergleichselement $e_0$ kann die Zielfunktion

$$f(S) = L(\{e_0\}) - L(S \cup \{e_0\})$$

konstruiert werden, deren Maximierung äquivalent zur Minimierung der Verlustfunktion $L$ ist. Sie ist außerdem monoton submodular [29].
In dieser Form ist die Zielfunktion abhängig von der Grundmenge $V$ der Daten. Für eine exakte Auswertung ist der Zugriff auf den gesamten Datensatz notwendig. Für eine approximative Auswertung hilft die additive Zerlegbarkeit von $f$ in submodulare Funktionen $f_e(S)$, die unabhängig von $V$ sind:

$$\begin{aligned} f(S) &= L(\{e_0\}) - L(S \cup \{e_0\}) \\ &= \frac{1}{|V|} \sum_{e \in V} \min_{v \in \{e_0\}} d(e, v) - \frac{1}{|V|} \sum_{e \in V} \min_{v \in S \cup \{e_0\}} d(e, v) \\ &= \frac{1}{|V|} \sum_{e \in V} \underbrace{\left( d(e, e_0) - \min_{v \in S \cup \{e_0\}} d(e, v) \right)}_{=: f_e(S)} . \end{aligned}$$

Eine Approximation der Funktion $f$ ist möglich, indem die Auswertung auf ein Reservoir $W \subset V$ beschränkt wird:

$$f_W(S) = \frac{1}{|W|} \sum_{e \in W} f_e(S) .$$

Bei zufälliger Auswahl des Reservoirs kann eine Garantie für die Güte der Approximation in Abhängigkeit von der Größe des Reservoirs angegeben werden [2].
Für das Sammeln des Reservoir $W$ ist allerdings ein zusätzlicher Lauf über die Daten notwendig, deshalb ist auch die Approximation nicht in Datenstromumgebungen einsetzbar, in denen Elemente kontinuierlich ankommen. Dafür kann eine heuristische Variante mit einem einzelnen Lauf über die Daten verwendet werden [2]. Diese nutzt die ersten $|W|$ Elemente des Datenstroms als Reservoir. Während der Berechnung der Zusammenfassung wird das Reservoir kontinuierlich aktualisiert. In Kapitel 3 wird diese Aktualisierung als *Reservoir Sampling* eingeführt.

### 2.4.2 Entropiebasierend

Eine weitere Zielfunktion maximiert die Vielfalt der Zusammenfassung, indem die Aufnahme von ähnlichen Elementen vermieden wird. Die Ähnlichkeit zwischen zwei Elementen wird dabei durch eine sogenannte Kernfunktion bestimmt. Eine vereinfachte Aussage von Mercer's Theorem [40] definiert die Klasse der positiv definiten Kernfunktionen.

**2.4.1 Definition (Positiv definite Kernfunktion[1]).** Für eine nicht-leere Menge $\mathcal{X}$ ist die Funktion $\mathcal{K} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ eine positiv definite Kernfunktion, wenn sie symmetrisch ist, das heißt $\mathcal{K}(x, y) = \mathcal{K}(y, x)$, und wenn für beliebige Mengen $\{x_1, \ldots, x_n\} \in \mathcal{X}^n$ und alle $n \in \mathbb{N}, c_1, \ldots, c_n \in \mathbb{R}$ gilt:

$$\sum_{i}^{n} \sum_{j}^{n} c_i c_j \mathcal{K}(x_i, x_j) \geq 0\,.$$

Eine äquivalente Definition lautet: Die Kernfunktion $\mathcal{K}$ ist positiv definit, wenn ihre Kernmatrix

$$\Sigma_S = \begin{pmatrix} \mathcal{K}(x_1, x_1) & \cdots & \mathcal{K}(x_1, x_k) \\ \vdots & \ddots & \vdots \\ \mathcal{K}(x_k, x_1) & \cdots & \mathcal{K}(x_k, x_k) \end{pmatrix}$$

über beliebige Mengen $S = \{x_1, \ldots, x_n\} \in \mathcal{X}^n$ positiv semi-definit ist.

Die Kernmatrix über einer Menge gibt also die paarweisen Ähnlichkeiten ihrer Elemente an. Wenn die Kernmatrix positiv semi-definit ist, folgen vorteilhafte Eigenschaften. Zum Beispiel ist dann ihre Determinante nicht-negativ. Diese Eigenschaft kann für die Monotonie der Zielfunktion ausgenutzt werden. Die Klasse der positiv semi-definiten Matrizen und eine weitere Klasse, die positiv definiten Matrizen, tauchen in vielen Anwendungen auf und werden jetzt definiert:

**2.4.2 Definition (Positiv (semi-)definite Matrix).** Eine symmetrische, reelle $n \times n$ Matrix $A$ ist positiv semi-definit, wenn $x^\intercal A x \geq 0$, und positiv definit, wenn $x^\intercal A x > 0$ für alle Vektoren $x \neq \vec{0}$ der Länge $n$ gilt.

Die positiv definiten Matrizen sind eine Unterklasse der positiv semi-definiten Matrizen. Beide Klassen sind unter Addition abgeschlossen. Das folgende Korollar ermöglicht die Konstruktion einer positiv definiten Matrix aus einer positiv semi-definiten Matrix.

**2.4.3 Korollar.** *Für eine positiv definite $n \times n$ Matrix $A$ und eine positiv semi-definite $n \times n$ Matrix $B$ ist die Matrix $A + B$ positiv definit.*

[1] Analog zu Matrizen wird gelegentlich zwischen positiv definiten und positiv semi-definiten Kernfunktionen unterschieden. Bei einer Unterscheidung wären hier die positiv semi-definiten Kernfunktionen gemeint.

*Beweis.* Seien $x \neq \vec{0}$ ein beliebiger Vektor der Länge $n$. Dann gilt:

$$x^\intercal(A+B)x = \underbrace{x^\intercal Ax}_{>0} + \underbrace{x^\intercal Bx}_{\geq 0} > 0\,. \qquad \square$$

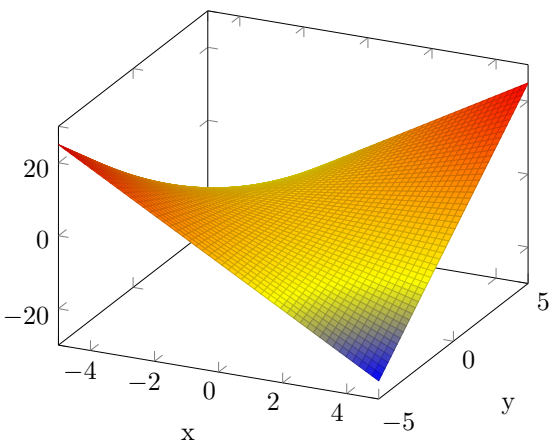
Die *Informative Vector Machine* (IVM) [23] ist verwandt mit der submodularen Maximierung. Sie erstrebt die Auswahl einer möglichst vielfältigen Teilmenge der Daten. Basierend auf der Entropie schlagen die Autoren dafür die Maximierung der Funktion

$$f(S) = \frac{1}{2}\log\det(\mathcal{I} + \sigma^{-2}\Sigma_S)$$

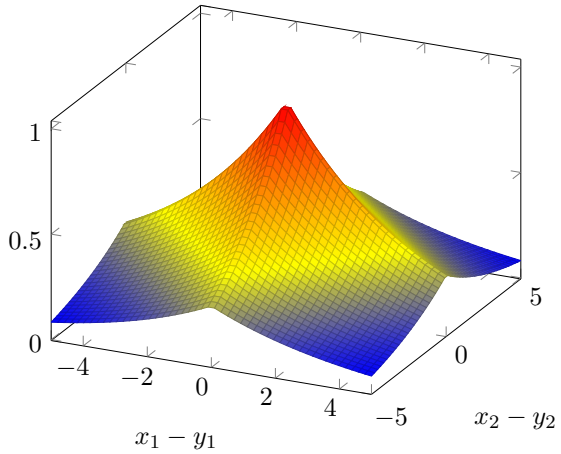
vor. Dabei bezeichnet $\mathcal{I}$ die $k \times k$ Einheitsmatrix und $\Sigma_S$ die Kernmatrix einer positiv definiten Kernfunktion über der Menge $S$. In [52] wird die monotone Submodularität der Funktion bewiesen. Die berechnete Zusammenfassung ist durch die Wahl der Kernfunktion steuerbar, dadurch ist die Anpassung an verschiedene Zielsetzungen möglich. Tabelle 2.1 enthält Beispiele für positiv definite Kernfunktionen. Die einfachste Kernfunktion ist die Lineare Kernfunktion, welche die Ähnlichkeit von zwei Elementen als Skalarprodukt angibt. Sie ist ein Spezialfall der Polynomiellen Kernfunktion, welche das Skalarprodukt zusätzlich potenziert. Üblicher in der Datenzusammenfassung ist die Verwendung von stationären Kernfunktionen (Abbildung 2.2). Sie sind translationsinvariant: $\mathcal{K}(x,y) = \mathcal{K}(x+a, y+a) = \mathcal{K}(x-y, 0)$, also nur abhängig von der Differenz der Elemente. Die Laplace-, Exponentielle und Gauß-Kernfunktion unterscheiden sich durch die Verwendung verschiedener Normen. Durch die endliche Maschinengenauigkeit hat der Parameter $h$ dabei einen problematisch hohen Einfluss und muss sorgfältig angepasst werden. Hierfür kann Kenntnis über die Daten helfen.
Um komplexere Kernfunktionen zu erzeugen, können verschiedene Kernfunktionen miteinander addiert oder multipliziert werden. Die Resultate sind wieder positiv definit, wenn die verwendeten Kernfunktionen positiv definit sind [12]. Das folgende Korollar erlaubt dabei auch die Multiplikation mit nicht-negatives Skalaren.

| **Kernel** | $\mathcal{K}$**(x, y)** | **p.d.** | **stationär** |
|---|---|---|---|
| Polynomiell | $\left(x^T y\right)^d$ | ✓ | |
| Laplace | $\exp\left(-\frac{\|\|x-y\|\|_1}{2h^2}\right)$ | ✓ | ✓ |
| Exponentiell | $\exp\left(-\frac{\|\|x-y\|\|_2}{2h^2}\right)$ | ✓ | ✓ |
| Gauß | $\exp\left(-\frac{\|\|x-y\|\|_2^2}{2h^2}\right)$ | ✓ | ✓ |

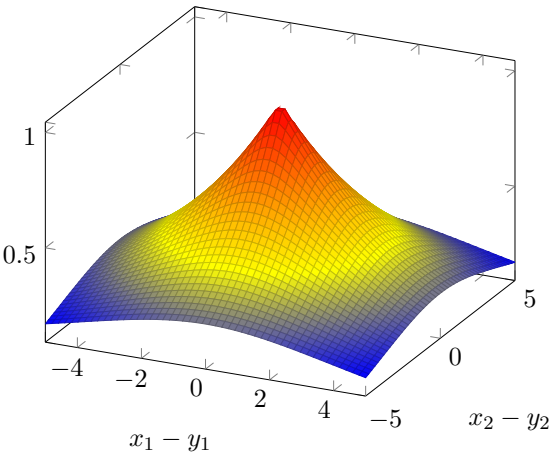
**Tabelle 2.1:** Positiv definite (p.d.) Kernfunktionen. $||z||_p = \sqrt[p]{\sum_i |z_i|^p}$ bezeichnet die $L_p$-Norm des Vektors $z$.



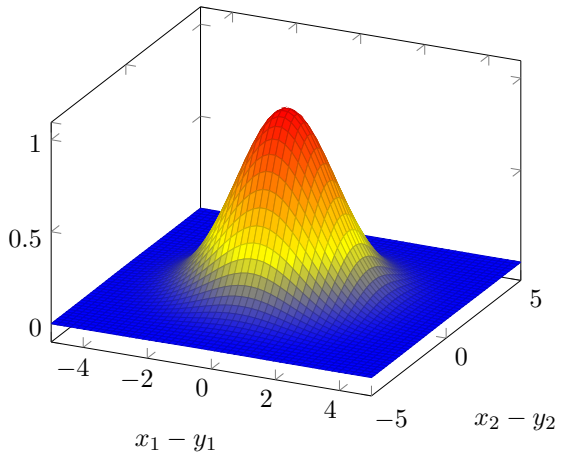

**(a)** Lineare Kernfunktion




**(b)** Laplace-Kernfunktion




**(c)** Exponentielle Kernfunktion




**(d)** Gauß-Kernfunktion

**Abbildung 2.2:** Nicht-stationäre Kernfunktion mit $x, y \in \mathbb{R}$: (a) Linear. Stationäre Kernfunktionen mit $x, y \in \mathbb{R}^2$: (b) Laplace, (c) Exponentiell und (d) Gauß.

**2.4.4 Korollar.** *Die Kernfunktion $\mathcal{K}'(x, y) = a \cdot \mathcal{K}(x, y)$ ist positiv definit, wenn $\mathcal{K}(x, y)$ positiv definit ist und $a \geq 0$.*

*Beweis.* Sei $\mathcal{K} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, $\{x_1, \ldots, x_n\} \in \mathcal{X}^n$ beliebig und $n \in \mathbb{N}, c_1, \ldots, c_n, a \in \mathbb{R}$ mit $a \geq 0$. Dann gilt:

$$\begin{aligned}\sum_i^n \sum_j^n c_i c_j \mathcal{K}'(x_i, x_j) &= \sum_i^n \sum_j^n c_i c_j (a \cdot \mathcal{K}(x_i, x_j)) \\ &= \underbrace{a}_{\geq 0} \cdot \underbrace{\sum_i^n \sum_j^n c_i c_j \mathcal{K}(x_i, x_j)}_{\geq 0} \geq 0\,.\end{aligned}$$

□

# Kapitel 3

# Algorithmen

In Kapitel 2 wurde die Berechnung von Zusammenfassungen unter Kardinalitätsbeschränkung als Optimierungsproblem

$$\max_{|S| \leq k} f(S)$$

für eine Zielfunktion $f$ formuliert. Auch wenn die Funktion $f$ submodular ist, bleibt das Problem im Allgemeinen $NP$-schwer und eine exakte Lösung ist (wahrscheinlich) nicht effizient berechenbar. Allerdings kann die Submodularität ausgenutzt werden, um approximative Lösungen zu berechnen. In diesem Kapitel werden zwei Approximationsalgorithmen für die Maximierung monoton submodularer Funktionen vorgestellt. Neben dem Berechnungsaufwand und dem Speicherbedarf werden sie auch durch ihre Approximationsgarantie $\alpha$ mit

$$f(L) \geq \alpha \cdot f(L^*)$$

charakterisiert. Dabei bezeichnet $L$ die berechnete Lösung und $L^*$ die (unberechenbare) exakte Lösung. Ein Algorithmus mit Approximationsgarantie 1 berechnet immer eine exakte Lösung, während ein Algorithmus mit Approximationsgarantie 0 beliebig schlechte Lösungen liefern kann.
Der *Greedy* Algorithmus berechnet durch $k$ Läufe über die Daten eine Lösung mit Approximationsgarantie $1 - \frac{1}{e} \approx 0.63$. Der *Sieve Streaming* Algorithmus wurde für die Verwendung auf Datenströmen entwickelt und bietet auf dem aktuellen Forschungsstand die höchste Approximationsgarantie für das Problem in einer Datenstromumgebung. In einem einzelnen Lauf über die Daten berechnet er bei konstantem Speicherbedarf eine $\frac{1}{2} - \epsilon$-Approximation. Die Funktionsweise dieser Algorithmen – und wie sie die Submodularität ausnutzen – wird jetzt genauer untersucht.

## 3.1 Greedy Algorithmus

**Algorithmus 3.1** Greedy Algorithmus [45]

*Eingabe:* Daten über der Grundmenge $V$ und Größe der Zusammenfassung $k$
*Ausgabe:* Zusammenfassung $S$

1: $S \leftarrow \emptyset$
2: **for** $i \leftarrow 1$ to $k$ **do**
3: $\quad S \leftarrow S \cup \{\arg\max_{e \in V} \Delta_f(e|S)\}$
4: **return** S

Der Greedy Algorithmus (Algorithmus 3.1) wurde 1978 von Nemhauser et al. veröffentlicht [45]. Er berechnet eine Zusammenfassung durch ein einfaches, gieriges Vorgehen. Angefangen mit der leeren Zusammenfassung werden die Daten $k$ mal durchlaufen. Nach jedem Lauf wird das Element mit dem größten Ertrag zur Zusammenfassung hinzugefügt. Der Ertrag eines Elements ist der absolute Anstieg im Funktionswert, wenn es zur Zusammenfassung hinzugefügt wird: $\Delta_f(e|S) = f(S \cup \{e\}) - f(S)$. Nach dem $k$-ten Lauf besteht die Zusammenfassung aus $k$ Elementen und wird zurückgegeben.
Bei $\mathcal{O}(kn)$ Zielfunktionsaufrufen für $n$ Datenpunkte und $\mathcal{O}(k)$ Speicherbedarf für die Zusammenfassung beträgt die Approximationsgarantie des *Greedy* Algorithmus trotz des simplen Vorgehens $1 - \frac{1}{e} \approx 0.63$. Feige zeigte 1998, dass (sofern $P \neq NP$) kein polynomieller Algorithmus mit einer höheren Approximationsgarantie für das Problem existiert [14].
Eine verbesserte Implementierung veröffentlichte Minoux 1978 als *Lazy Greedy* [41]. In dieser wird nicht in jedem Lauf über den gesamten Datensatz iteriert, sondern nur über notwendige Elemente. Wegen Submodularität ist der Ertrag eines Elements eine obere Schranke für den Ertrag des Elements in den nachfolgenden Iterationen. Die oberen Schranken der Elemente werden in einer absteigend sortierten Liste verwaltet. Durch Berechnung des Ertrags wird der Kopf der Liste solange aktualisiert, bis er das größte Element in der Liste bleibt. Dann ist garantiert, dass dies der größte Ertrag aller Elemente ist. Durch die Berechnung der Erträge nach Bedarf sind in der Praxis Einsparungen im Berechnungsaufwand um mehrere Größeneinheiten möglich, sodass *Lazy Greedy* als *state-of-the-art* Implementierung des *Greedy* Algorithmus benutzt wird [42].
Der *Greedy* Algorithmus wurde in vielen realen Anwendungen eingesetzt, wie die Platzierung von Sensoren [31] und die Zusammenfassung von Dokumenten [37] und Bildern [56]. In diesen Fällen berechnete er eine nahezu optimale Lösung, die weit besser war, also die durch die Approximationsgarantie vorgegebene *worst-case* Lösung. Für komplexere Einschränkungen existieren verschiedene Varianten des Algorithmus [16, 30].
Da $k$ Läufe über die Daten notwendig sind, kann der *Greedy* Algorithmus nicht in einer Datenstromumgebung eingesetzt oder an diese angepasst werden.

## 3.2 Sieve Streaming

**Algorithmus 3.2** Sieve Streaming [2]

*Eingabe:* Datenstrom $e_1, e_2, \ldots$ und Größe der Zusammenfassung $k$

*Ausgabe:* Zusammenfassung $S_v$

1: $O \leftarrow \{(1+\epsilon)^i | i \in \mathbb{Z}\}$
2: **for all** $v \in O$ **do**
3: $\quad S_v \leftarrow \emptyset$ (verwalte notwendige Mengen nach Bedarf)
4: $m \leftarrow 0$
5: **for** $i \leftarrow 1$ to $n$ **do**
6: $\quad m \leftarrow \max(m, f(\{e_i\}))$
7: $\quad O_i \leftarrow \{(1+\epsilon)^i | m \leq (1+\epsilon)^i \leq 2 \cdot k \cdot m\}$
8: $\quad$ Entferne alle $S_v$ mit $v \notin O_i$
9: $\quad$ **for all** $v \in O_i$ **do**
10: $\quad\quad$ **if** $\Delta_f(e_i|S_v) \geq \frac{v/2 - f(S_v)}{k - |S_v|}$ and $|S_v| < k$ **then**
11: $\quad\quad\quad S_v \leftarrow S_v \cup \{e_i\}$
12: **return** $\arg\max_{v \in O_n} f(S_v)$

*Sieve Streaming* (Algorithmus 3.2) wurde 2014 von Badanidiyuru et al. als Datenstromalgorithmus für die monoton submodulare Maximierung unter Kardinalitätsbeschränkung veröffentlicht [2]. Basierend auf parallelem *Thresholding* berechnet der Algorithmus eine Zusammenfassung in einem einzelnen Lauf über die Daten. Der Schwellenwert (*threshold*) eines Siebes gibt den notwendigen Ertrag eines Elements an, damit es zur Zusammenfassung hinzugefügt wird. Weil der optimale Schwellenwert nicht bekannt ist, werden mehrere Siebe mit unterschiedlichen Schwellenwerten und Zusammenfassungen verwaltet.
Für die Abschätzung des optimalen Schwellenwertes hilft der Zielfunktionswert der optimalen Lösung. Dafür ist wiederum der maximale Funktionswert einzelner Elemente hilfreich. Zur Bestimmung dieses Wertes wird eine Variable $m$ verwaltet, die in jeder Iteration aktualisiert wird. Wegen der Submodularität der Zielfunktion $f$ gilt dann für die optimale Zusammenfassung $L^*$ bisheriger Elemente:

$$m \leq f(L^*) \leq k \cdot m\,.$$

Wenn $m$ aktualisiert wird, werden auch die Siebwerte (*values*) $O_i$ auf das neue Intervall des Optimums angepasst. Für neue Siebwerte im Intervall werden Siebe generiert, während Siebe mit Siebwerten außerhalb des Intervalls entfernt werden. Wenn der Parameter $\epsilon$ klein genug gewählt wird, liegt mindestens einer der Siebwerte nahe am tatsächlichen Optimum. Daraufhin wird das aktuelle Element in die Zusammenfassung aller Siebe aufgenommen, deren Schwellenwert der Ertrag des Elements überschreitet. Für die Erfüllung der Kardinalitätsbeschränkung werden volle Siebe nicht mehr berücksichtigt. Der Schwellenwert

eines Siebes wird durch $\frac{v/2-f(S_v)}{k-|S_v|}$ berechnet. Badanidiyuru et al. [2] zeigen anhand eines Beispiels mit schiefen Daten, dass die Schwellenwerte zu hoch sind, wenn die Siebwerte nicht halbiert werden. Durch die Division der noch fehlenden Güte für den halbierten Siebwert $v$ durch die Anzahl der freien Plätze im Sieb, erreicht das Sieb nach Aufnahme von $k$ Elementen mindestens die Güte $\frac{v}{2}$. Damit sind die Zusammenfassungen voller Siebe $\frac{1}{2}$-Approximationen ihrer Siebwerte $v$. Die Halbierung der Siebwerte ist der Grund für die Erweiterung des Siebintervalls bis $2km$ in Zeile 7. Bei der Anlegung des Siebes mit dem Siebwert $2km$ besitzt es den Schwellenwert $\frac{2km/2}{k} = m$ und nimmt damit als letztes Sieb das größte bisher gelesene Element mit dem Funktionswert $m$ auf. Auf Anfrage oder nach dem Lesen einer endlichen Datenmenge der Größe $n$ wird die Zusammenfassung mit dem höchsten Zielfunktionswert zurückgegeben.

Die Approximationsgarantie des *Sieve Streaming* Algorithmus hängt von der Anzahl der Siebe ab, die durch die Wahl des Parameters $\epsilon$ bestimmt wird. Mit einem Speicherbedarf von $\mathcal{O}\left(\frac{k\log(k)}{\epsilon}\right)$ und $\mathcal{O}\left(\frac{\log(k)}{\epsilon}\right)$ Funktionsaufrufen pro Iteration beträgt die Approximationsgarantie $\frac{1}{2} - \epsilon$. Der Algorithmus erreicht damit fast die 2015 von Buchbinder et al. aufgestellte obere Schranke $\frac{1}{2} + \varepsilon$ mit $\varepsilon > 0$ für die Approximationsgarantie von Datenstromalgorithmen, die monoton oder nicht-monoton submodulare Funktionen unter Kardinalitätsbeschränkung maximieren.

Auch dieser Algorithmus erreicht meistens bessere Approximationen als vorgegeben. In den Experimenten von Badanidiyuru et al. [2] und in Kapitel 5 erzielt er ähnliche Zielfunktionswerte wie der *Greedy* Algorithmus.

Bei der Berechnung der besten Zusammenfassung sind in der Regel nicht alle Siebe voll. Die nicht-vollen Siebe können durch die Aufnahme von zufälligen Elementen verbessert werden. Ohne Erhöhung des Berechnungsaufwandes können zufällige Elemente durch *Reservoir Sampling* (Algorithmus 3.3) beiläufig eingesammelt werden. Das Reservoir enthält zu jedem Zeitpunkt alle bisher gelesenen Elemente mit gleicher Wahrscheinlichkeit.

**Algorithmus 3.3** Reservoir Sampling [59]

*Eingabe:* Datenstrom $e_1, e_2, \ldots$ und Größe der Zusammenfassung $k$
*Ausgabe:* Zusammenfassung $S$

1: $S \leftarrow$ neue Liste
2: **for** $i \leftarrow 1$ to $k$ **do**
3: $\quad S[i] \leftarrow e_i$
4: **for** $i \leftarrow k+1$ to $n$ **do**
5: $\quad j \leftarrow$ Zufallszahl zwischen 1 und i
6: $\quad$ **if** $j \leq k$ **then**
7: $\quad\quad S[j] \leftarrow e_i$
8: **return** S

# Kapitel 4

# Implementierung

In Kapitel 3 wurde der *Sieve Streaming* Algorithmus als effizientes Verfahren zur Berechnung von Zusammenfassungen auf Datenströmen vorgestellt. Dieser wird für die Experimente in Kapitel 5 eingesetzt. Weil Badanidiyuru et al. [2] keine Implementierungsdetails angeben, wird der Fokus in diesem Kapitel auf konkrete Implementierungsmöglichkeiten gerichtet. Maßgeblich für den Berechnungsaufwand des Algorithmus ist die Anzahl der Siebe und der Aufwand pro Zielfunktionsauswertung. Besonders an diesen Stellen kann die Laufzeit durch geschickte Implementierung um mehrere Größenordnungen reduziert werden.
Eine vollständige Implementierung des *Sieve Streaming* Algorithmus in der Programmiersprache Java ist unter `www.bitbucket.org/maik-schmidt/sieve-streaming` frei verfügbar.

## 4.1 Anzahl der Siebe

Für die Anzahl der Funktionsaufrufe ist hauptsächlich die Anzahl der gleichzeitig verwalteten Siebe verantwortlich. Der *Sieve Streaming* Algorithmus verwaltet in jedem Schleifendurchlauf Siebe im Intervall $[m, 2km]$, dabei bezeichnet $m$ das größte bisher gelesene Element und $k$ die geforderte Maximalgröße einer Zusammenfassung und damit auch die maximale Anzahl an Elementen in einem Sieb.
Für die Anzahl der gleichzeitig verwalteten Siebe gilt für beliebige monoton submodulare Zielfunktionen:

$$\begin{aligned} |\text{Siebe}| &= |\{(1+\epsilon)^i | m \leq (1+\epsilon)^i \leq 2km, i \in \mathbb{Z}\}| \\ &= \left\lfloor \frac{\log(2km)}{\log(1+\epsilon)} - \frac{\log(m)}{\log(1+\epsilon)} \right\rfloor \\ &= \left\lfloor \frac{\log(2k)}{\log(1+\epsilon)} \right\rfloor \in \mathcal{O}\left(\frac{\log(k)}{\epsilon}\right). \end{aligned}$$

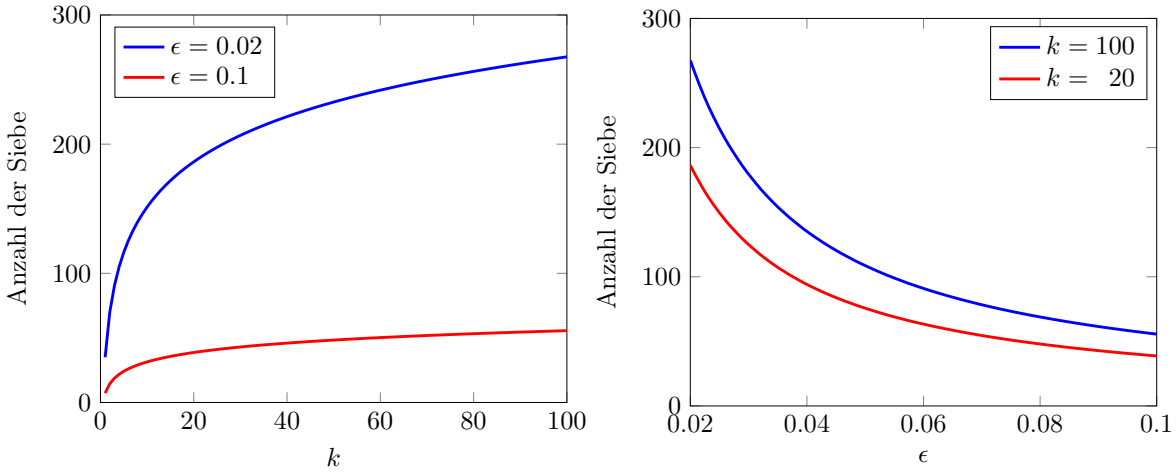



**Abbildung 4.1:** Anzahl der gleichzeitig verwalteten Siebe des *Sieve Streaming* Algorithmus mit originalem Siebintervall $[m, 2km]$. Der Parameter $k$ gibt die geforderte Größe der Zusammenfassung an, $\epsilon$ steuert die Approximationsgarantie des Algorithmus.

Die Anzahl der Siebe (Abbildung 4.1) ist stark vom Parameter $\epsilon$ abhängig, der die Approximationsgarantie des Algorithmus steuert. Es können leicht mehrere hundert Siebe generiert werden.

### 4.1.1 Reduktion der Anzahl der Siebe

Die Anzahl der Zielfunktionsaufrufe soll durch Vermeidung von überflüssigen Sieben reduziert werden. Die Intervallgrenzen $m$ und $2km$ der Siebe wurden in Kapitel 3 auf die Schranken $m$ und $km$ des maximalen Funktionswerts beliebiger monoton submodularer Funktionen zurückgeführt. Bei Verwendung der Zielfunktion $f(S) = \frac{1}{2}\log\det(\mathcal{I} + \sigma^{-2}\Sigma_S)$ ist das Intervall $[m, 2km]$ größer als notwendig und kann verkleinert werden. Gesucht ist das Intervall aller möglichen Funktionswerte $k$-elementiger Mengen.
Durch Betrachtung des charakteristischen Polynoms der Matrix $\sigma^{-2}\Sigma_S$ kann eine untere Schranke der Funktionswerte gefunden werden. Das charakteristische Polynom [6] einer $k \times k$ Matrix $A$ ist definiert als

$$C_A(x) = \det(x\mathcal{I} - A) = (x - \lambda_1)(x - \lambda_2)\cdots(x - \lambda_k)\,. \tag{4.1}$$

Dabei sind $\lambda_1, \ldots, \lambda_k$ die Eigenwerte der Matrix $A$. Durch Ausmultiplizieren erhält man die charakteristischen Koeffizienten $c_0, c_1, \ldots, c_k$ von $A$:

$$\begin{aligned} C_A(x) &= x^k - x^{k-1}\underbrace{(\lambda_1 + \lambda_2 + \cdots + \lambda_k)}_{c_1} + x^{k-2}\underbrace{(\lambda_1\lambda_2 + \cdots)}_{c_2} - \cdots + (-1)^k\underbrace{(\lambda_1\lambda_2\cdots\lambda_k)}_{c_k} \\ &= x^k - x^{k-1}c_1 + x^{k-2}c_2 - \cdots + (-1)^k c_k \\ &= \sum_{i=0}^{k}(-1)^i x^{k-i}c_i\,. \end{aligned} \tag{4.2}$$

Mithilfe des charakteristischen Polynoms kann die Determinante

$$\begin{aligned}
\det(\mathcal{I} + \sigma^{-2}\Sigma) &= (-1)^k \det(-\mathcal{I} - \sigma^{-2}\Sigma) \\
&\overset{(4.1)}{=} (-1)^k C_{\sigma^{-2}\Sigma}(-1) \\
&\overset{(4.2)}{=} (-1)^k \sum_{i=0}^{k} (-1)^i (-1)^{k-i} c_i \\
&= (-1)^{2k} \sum_{i=0}^{k} c_i \\
&= \sum_{i=0}^{k} c_i
\end{aligned}$$

in der Zielfunktion als Summe der charakteristischen Koeffizienten von $\sigma^{-2}\Sigma$ berechnet werden. Sei $\mathcal{K}$ die positiv definite Kernfunktion der Kernmatrix $\Sigma_S$ auf einer Menge $S = \{e_1, \ldots, e_k\}$. Dann ist die Kernfunktion $\sigma^{-2}\mathcal{K}$ wegen Korollar 2.4.4 für $\sigma > 0$ positiv definit und ihre Kernmatrix

$$\begin{pmatrix} \sigma^{-2}\mathcal{K}(e_1, e_1) & \cdots & \sigma^{-2}\mathcal{K}(e_1, e_k) \\ \vdots & \ddots & \vdots \\ \sigma^{-2}\mathcal{K}(e_k, e_1) & \cdots & \sigma^{-2}\mathcal{K}(e_k, e_k) \end{pmatrix} = \sigma^{-2}\Sigma_S$$

wegen Definition 2.4.1 positiv semi-definit. Die Eigenwerte $\lambda_1, \ldots \lambda_k$ der Matrix $\sigma^{-2}\Sigma$ sind also nicht-negativ, deshalb müssen auch alle charakteristischen Koeffizienten $c_0, c_1, \ldots, c_k$ als Summen über Produkte von Eigenwerten [6] nicht-negativ sein. Für die besonderen Koeffizienten $c_0$ und $c_1$ gelten:

$$\begin{aligned}
c_0 &= 1 \\
c_1 &= \sum_{i=0}^{k} \lambda_i = \operatorname{spur}(\sigma^{-2}\Sigma)\,.
\end{aligned}$$

Für alle in Kapitel 2 vorgestellten stationären Kernfunktionen kann die Spur der Matrix $\sigma^{-2}\Sigma$ wegen $\mathcal{K}(x, x) = 1$ direkt als $k\sigma^{-2}$ angegeben werden. Dies gilt auch für die in Anhang A beschriebene Normierung beliebiger positiv definiter Kernfunktionen. Damit kann die Determinante nach unten eingeschränkt werden durch

$$\det(\mathcal{I} + \sigma^{-2}\Sigma) = \sum_{i=0}^{k} c_i \geq c_0 + c_1 = 1 + k\sigma^{-2}$$

und $\frac{1}{2}\log(1+k\sigma^{-2})$ ist eine untere Schranke für die Funktionswerte von $\frac{1}{2}\log\det(1+\sigma^{-2}\Sigma)$. Ein Beispiel für eine $k \times k$ Kernmatrix mit minimalem Funktionswert ist die Einsmatrix

$$\Sigma_{S_{min}} = \begin{pmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{pmatrix}.$$

Seien $c = \begin{pmatrix} \sigma^2 & \cdots & \sigma^2 \end{pmatrix}^\intercal$ und $r = \begin{pmatrix} 1 & \cdots & 1 \end{pmatrix}$, sodass $cr = \sigma^{-2}\Sigma_{S_{min}}$, dann lässt sich der Zielfunktionswert mit Sylvesters Determinantentheorem (1) angeben als

$$\begin{aligned} f(S_{min}) &= \frac{1}{2}\log\det(\mathcal{I} + \sigma^{-2}\Sigma_{S_{min}}) \\ &= \frac{1}{2}\log\det(\mathcal{I} + cr) \\ &\overset{(1)}{=} \frac{1}{2}\log(1 + rc) \\ &= \frac{1}{2}\log(1 + k\sigma^{-2})\,. \end{aligned}$$

Damit ist $\frac{1}{2}\log(1 + k\sigma^{-2})$ der minimale Zielfunktionswert $k$-elementiger Mengen.
Eine obere Schranke $k \cdot m$ folgt aus der Submodularität der Zielfunktion. Ein Beispiel für eine $k \times k$ Kernmatrix mit maximalem Funktionswert ist die Einheitsmatrix

$$\Sigma_{S_{max}} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix} = \mathcal{I}\,.$$

Der Zielfunktionswert dieser Matrix lautet

$$\begin{aligned} f(S_{max}) &= \frac{1}{2}\log\det(\mathcal{I} + \sigma^{-2}\mathcal{I}) \\ &= \frac{1}{2}\log\det((1 + \sigma^{-2}) \cdot \mathcal{I}) \\ &= \frac{1}{2}\log((1 + \sigma^{-2})^k \cdot \det(\mathcal{I})) \\ &= \frac{k}{2}\log(1 + \sigma^{-2})\,. \end{aligned}$$

Betrachtet man den maximalen Funktionswert $m$ für einelementige Mengen, dann fällt auf, dass dieser wegen $\mathcal{K}(x, x) = 1$ unabhängig von den Daten konstant ist und sogar durch alle Elemente erreicht wird:

$$m = \max_{|S|=1} f(S) = \frac{1}{2}\log(1 + \sigma^{-2})\,. \tag{4.3}$$

Damit gilt: $f(S_{max}) = k \cdot m$. Weil Beispiele für die untere und die obere Schranke gefunden wurden, ist das Intervall $\left[\frac{1}{2}\log(1 + k\sigma^{-2}), km\right]$ das kleinste Intervall, das alle Funktionswerte $k$-elementiger Mengen enthält.
Aus (4.3) folgt, dass die Siebe nicht mehr dynamisch verwaltet werden müssen, stattdessen können sie bei der Initialisierung einmalig angelegt werden. Außerdem ist dadurch auch keine Erweiterung des Siebintervalls bis $2km$ mehr notwendig. Das Siebintervall kann sogar noch weiter auf das Intervall $\left[\log(1 + k\sigma^{-2}), km\right]$ reduziert werden. Weil die Siebwerte

für die Berechnung der Schwellenwertes halbiert werden, nehmen alle Siebe mit einem kleinerem Wert wahllos Elemente aus dem Datenstrom auf und erzeugen identische Zusammenfassungen, die aus den ersten $k$ Elementen bestehen. Das letzte Sieb, das wahllos Elemente aufnimmt, ist das Sieb mit dem Siebwert $\log(1 + k\sigma^{-2})$.
Mit dem neuen Intervall $\left[\log(1 + k\sigma^{-2}), km\right]$ gilt für die Anzahl der gleichzeitig verwalteten Siebe:

$$\begin{aligned} |\text{Siebe}^*| &= |\{(1+\epsilon)^i | \log(1 + k\sigma^{-2}) \leq (1+\epsilon)^i \leq km, i \in \mathbb{Z}\}| \\ &= \left\lfloor \frac{\log(km) - \log\log(1 + k\sigma^{-2})}{\log(1+\epsilon)} \right\rfloor . \end{aligned}$$

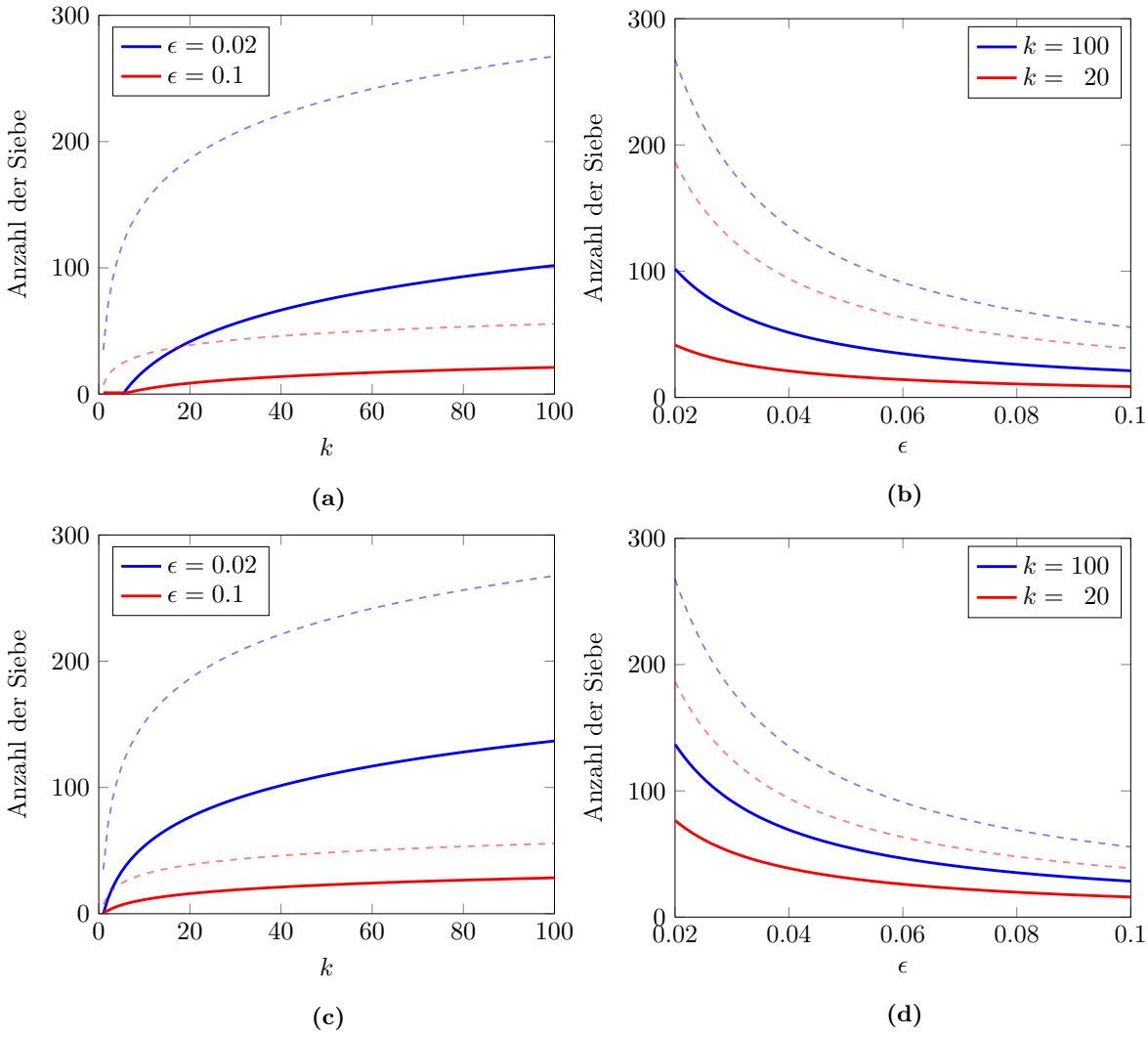



**Abbildung 4.2:** Reduktion der Anzahl der Siebe für $\sigma = 1$. Die Siebanzahl mit originalem Intervall von $m$ bis $2km$ ist gestrichelt dargestellt. Abbildungen (a) und (b) zeigen die Anzahl der Siebe mit minimalem Intervall von $\log(1 + k)$ bis $km$. Abbildungen (c) und (d) zeigen die Anzahl der Siebe mit großzügigem Intervall von $\log(1 + k)$ bis $2km$.

Die Anzahl der Siebe mit verkleinertem Intervall (Abbildung 4.2) wird dadurch für $\sigma = 1$ und $k = 20$ auf 22 % und für $k = 100$ auf 38 % der Siebanzahl mit originalem Intervall reduziert. Frappierend ist, dass für $k \leq 5$ unabhängig von den Daten und anderen Parametern nur ein einzelnes Sieb generiert wird. Das liegt daran, dass beliebige $k$-elementige Mengen eine $\frac{1}{2}$-Approximation der besten Zusammenfassung darstellen. Weil das keine zufriedenstellende Lösung liefern kann, ist eine Erweiterung des Intervalls wie beim originalen Siebintervall bis $2km$ ratsam. Dadurch ist das Erreichen höherer Zielfunktionswerte möglich, die Approximationsgarantie bleibt allerdings bei $\frac{1}{2} - \epsilon$. Die Anzahl der Siebe wird dann auf 41 % für $k = 20$ und 51 % für $k = 100$ reduziert. Die berechneten Zusammenfassungen mit diesem großzügigen und mit dem originalen Siebintervall sind immer identisch.

## 4.2 Aufwand eines Zielfunktionsaufrufs

### 4.2.1 Clusterungsbasierend

Die Auswertung der clusterungsbasierenden Zielfunktion wurde in Kapitel 2 mithilfe eines Reservoirs $W \subset V$ der Grundmenge für eine Zusammenfassung $S$ approximiert:

$$\begin{aligned} f_W(S) &= \frac{1}{|W|} \sum_{e \in W} f_e(S) \\ &= \frac{1}{|W|} \sum_{e \in W} \left( d(e, e_0) - \min_{v \in S \cup \{e_0\}} d(e, v) \right) \\ &= \frac{1}{|W|} \sum_{e \in W} d(e, e_0) - \frac{1}{|W|} \sum_{e \in W} \min_{v \in S \cup \{e_0\}} d(e, v) \,. \end{aligned}$$

Weil der Term $\frac{1}{|W|} \sum_{e \in W} d(e, e_0)$ konstant ist, müssen dafür lediglich die Abstände aller Reservoirpunkte zu den nächstgelegenen Elementen der Zusammenfassung berechnet werden. Der Berechnungsaufwand von $\mathcal{O}(|S| \cdot |W|)$ kann durch das Merken der minimalen Abstände $\min_{v \in S \cup \{e_0\}} d(e, v)$ in der nächsten Iteration erheblich verringert werden, denn es gilt:

$$\min_{v \in S \cup \{e_{k+1}\} \cup \{e_0\}} d(e, v) = \min \left\{ \min_{v \in S \cup \{e_0\}} d(e, v), d(e, e_{k+1}) \right\} .$$

Damit kann der Funktionswert nach dem Lesen eines neuen Elements $e_{k+1}$ berechnet werden durch:

$$f_W(S \cup \{e_{k+1}\}) = f_W(S) + \frac{1}{|W|} \sum_{e \in W} \max \left\{ 0, \min_{v \in S \cup \{e_0\}} d(e, v) - d(e, e_{k+1}) \right\} .$$

Der Aufwand eines Zielfunktionsaufrufs beträgt dann nur noch $\mathcal{O}(|W|)$.
Im *average-case* kann der Berechnungsaufwand weiter verringert werden, indem nur Distanzen zu Reservoirpunkten berechnet werden, wenn sie in einem Umkreis mit Radius

$\max_{e \in W} \min_{v \in S \cup \{e_0\}} d(e, v)$ um das neue Element $e_{k+1}$ liegen. Zu Punkten außerhalb dieses Umkreises kann das neue Element nicht am Nächsten liegen. Die geometrische Gestalt des Umkreises hängt dabei vom gewählten Distanzmaß ab. Durch Verwendung eines KD-Baums, der den Raum der Reservoirpunkte partitioniert, kann meistens der Hauptanteil der Distanzberechnungen übersprungen werden.

### 4.2.2 Entropiebasierend

Der Aufwand einer Auswertung der Zielfunktion $f(S) = \frac{1}{2} \log \det(\mathcal{I} + \sigma^{-2}\Sigma_S)$ wirkt sich stark auf die Laufzeit des *Sieve Streaming* Algorithmus aus. Durch geschickte Implementierung kann der Zeitaufwand für eine Zielfunktionsauswertung von $\mathcal{O}(k^3 + kd)$ auf $\mathcal{O}(k^2 + kd)$ reduziert werden. Dabei bezeichnet $k$ die Kardinalität der Menge $S$ und $d$ die Dimensionalität der Daten.
Weil sich in jeder Iteration des Algorithmus nur eine Zeile und Spalte der Kernmatrix ändert oder eine Zeile und Spalte hinzugefügt wird, liegt der Zeitaufwand für die Berechnung der neuen Kernmatrix bei $\mathcal{O}(kd)$.
Die naive Berechnung der Determinante einer $k \times k$ Matrix mit der Laplace-Entwicklung benötigt $\mathcal{O}(k!)$ Zeit. Üblicher ist die Berechnung der Determinante durch die LU-Zerlegung der Matrix. Eine reguläre $k \times k$ Matrix $A$ wird dabei in eine untere (*lower*) normierte Dreiecksmatrix $L$ und eine obere (*upper*) Dreiecksmatrix $U$ zerlegt, sodass $A = L \cdot U$ gilt. Anschließend kann die Determinante als Produkt der Diagonalelemente $u_{ii}$ von $U$ berechnet werden. Eine besondere Zerlegung in Diagonalmatrizen ist möglich, wenn die Matrix $A$ positiv definit ist. Die Cholesky-Zerlegung zerlegt sie dann in das Produkt einer unteren Dreiecksmatrix $L$ und ihrer Transponierten $L^\intercal$, sodass $A = L \cdot L^\intercal$ gilt. Mit dem Determinantenproduktsatz (1) kann die Determinante von $A$

$$\begin{aligned} \det(A) &= \det(LL^\intercal) \\ &\overset{(1)}{=} \det(L) \cdot \det(L^\intercal) \\ &= \prod_i^k l_{ii} \cdot \prod_i^k l_{ii} \\ &= \prod_i^k l_{ii}^2 \end{aligned} \tag{4.4}$$

als Produkt der quadrierten Diagonalelemente $l_{ii}$ von $L$ berechnet werden.
Beide Zerlegungen benötigen asymptotisch $\mathcal{O}(k^3)$ Zeit, der Rechenaufwand der Cholesky-Zerlegung ist in der Praxis bei vielen Anwendungen aber nur halb so hoch [53], weil nur eine Dreiecksmatrix berechnet werden muss. Außerdem gehört sie zu den numerisch stabilsten Matrixoperationen [27]. Die Anforderungen sind hierfür allerdings höher, weil die Cholesky-Zerlegung einer Matrix nur bei Positiv-Definitheit funktioniert. Die Einhaltung dieser Anforderung soll jetzt überprüft werden, dafür muss die Matrix $\mathcal{I} + \sigma^{-2}\Sigma_S$ positiv

definit sein.
Sei $\mathcal{K}$ die positiv definite Kernfunktion der Kernmatrix $\Sigma_S$ auf einer Menge $S = \{e_1, \ldots, e_k\}$. Dann ist die Kernfunktion $\sigma^{-2}\mathcal{K}$ wegen Korollar 2.4.4 für $\sigma > 0$ positiv definit und ihre Kernmatrix

$$\begin{pmatrix} \sigma^{-2}\mathcal{K}(e_1,e_1) & \cdots & \sigma^{-2}\mathcal{K}(e_1,e_k) \\ \vdots & \ddots & \vdots \\ \sigma^{-2}\mathcal{K}(e_k,e_1) & \cdots & \sigma^{-2}\mathcal{K}(e_k,e_k) \end{pmatrix} = \sigma^{-2}\Sigma_S$$

wegen Definition 2.4.1 positiv semi-definit. Weil die Einheitsmatrizen positiv definit sind, ist die Matrix $\mathcal{I}+\sigma^{-2}\Sigma_S$ wegen Korollar 2.4.3 auch positiv definit. Die Cholesky-Zerlegung kann also bei Verwendung beliebiger positiv definiter Kernfunktionen und $\sigma > 0$ für die Berechnung von $\det(\mathcal{I} + \sigma^{-2}\Sigma_S)$ angewandt werden.
Einen weiteren Vorteil hat die Zerlegung für die Berechnung der Log-Determinante. Bei großen Matrizen kann die explizite Berechnung der Determinante numerische Probleme verursachen. Stattdessen kann die Zielfunktion

$$\begin{aligned} f(S) &= \frac{1}{2}\log\det(\mathcal{I} + \sigma^{-2}\Sigma_S) \\ &= \frac{1}{2}\log\det(LL^T) \\ &\overset{(4.4)}{=} \frac{1}{2}\log\left(\prod_i^k l_{ii}^2\right) \\ &= \sum_i^k \log(l_{ii}) \end{aligned} \tag{4.5}$$

durch Berechnung der Summe der logarithmierten Diagonalelemente $l_{ii}$ der Cholesky-Zerlegung von $\mathcal{I} + \sigma^{-2}\Sigma_S$ ausgewertet werden.
Berechnet man die Dreiecksmatrix $L$ sequentiell mit der Cholseky-Banachiewicz Methode

$$\begin{aligned} l_{ij} &= \frac{1}{l_{ij}}\left(\sigma^{-2}\mathcal{K}(e_i,e_j) - \sum_{k=1}^{j-1} l_{ik}l_{jk}\right) && \text{für } i > j \\ l_{ii} &= \sqrt{1 + \sigma^{-2}\mathcal{K}(e_i,e_i) - \sum_{j=1}^{i-1} l_{ij}^2}\,, \end{aligned}$$

dann ist die Zerlegung einer Reihe $i$ nur abhängig von Elementen in derselben Reihe und in vorherigen Reihen $j < i$. Damit ist sie unabhängig von den nachfolgenden Reihen. Weil sich in jeder Iteration nur die letzte Reihe (und Spalte) der Kernmatrix ändert, kann die Cholesky-Zerlegung der neuen Kernmatrix aus der alten Zerlegung und der neuen Reihe berechnet werden. Die Zerlegung der letzten Reihe ist in quadratischer Laufzeit möglich. Wegen (4.5) lautet der zu berechnende Funktionswert nach dem Lesen eines neuen Elements $e_{k+1}$:

$$f(S \cup \{e_{k+1}\}) = f(S) + \log(l_{k+1,k+1})\,.$$

Dabei gilt, wie in Kapitel 2 definiert, $f(\emptyset) = 0$. Die Zielfunktion kann also durch Kenntnis des vorherigen Zielfunktionswertes in $\mathcal{O}(k^2 + kd)$ ausgewertet werden.

## 4.3 Parallelisierung

Eine weitere Möglichkeit zur Optimierung der Laufzeit ist die Verwendung von mehreren *Threads*. Da jedes Sieb eine eigene Zusammenfassung verwaltet, lassen sich alle Funktionsauswertungen in einer Iteration problemlos parallel ausführen. Dafür wird jedes Sieb einem *Thread* zugeteilt. Damit die Aufteilung auf die *Threads* nach Entfernen und Hinzufügen von neuen Sieben gleichmäßig bleibt, sollte sie nach dem *Round Robin* Prinzip geschehen. Dabei werden die Siebe an die *Threads* in zyklischer Reihenfolge verteilt.
Bei Verwendung der Zielfunktion $f(S) = \frac{1}{2} \log\det(\mathcal{I} + \sigma^{-2}\Sigma_S)$ sind die Siebe zu Beginn bereits festgelegt, es werden keine Siebe hinzugefügt oder entfernt. Trotzdem sollte auch hier eine *Round Robin* Zuteilung verwendet werden. Weil benachbarte nicht-volle Siebe ähnlich stark befüllt sind, benötigen sie ähnlich hohen Aufwand für die Auswertung der Zielfunktion. Durch das *Round Robin* Prinzip werden die Siebe mit hohem Berechnungsaufwand gleichmäßig an die *Threads* verteilt.
Alternativ kann die Parallelisierung durch einen *Thread Pool* umgesetzt werden. Dieser besteht aus einer festen Menge von *Threads* und einer Warteschlange mit Aufträgen. In jeder Iteration werden die Siebe mit dem neuen Element in die Auftragswarteschlage eingetragen. Die Aufträge werden dann aus der Warteschlage an untätige *Threads* verteilt.

# Kapitel 5

# Experimente

## 5.1 Sieve Streaming mit verkleinertem Siebintervall

### 5.1.1 Datensatz

Der hier verwendete *Oxford Parkinson's Disease Telemonitoring* Datensatz [55] besteht aus 5875 biomedizinischen Beobachtungen. Die Daten wurden im Jahr 2009 über einen Zeitraum von sechs Monaten gesammelt. Dabei wurden die Stimmen von 42 Personen aufgenommen, die im Frühstadium der Parkinson-Krankheit litten. Eine Beobachtung entspricht einer Sprachaufnahme, aus der 16 Merkmale extrahiert wurden.

### 5.1.2 Aufbau des Experiments

In diesem Experiment wird der *Sieve Streaming* Algorithmus zur Berechnung einer Zusammenfassung des *Oxford Parkinson's Disease Telemonitoring* Datensatzes eingesetzt. Das Experiment wurde bereits von Badanidiyuru et al. [2] mit originalem Intervall durchgeführt. Das Ziel des Experiments ist der Vergleich der Performanz des *Sieve Streaming* Algorithmus mit originalem Siebintervall $[m, 2km]$ und mit dem in Kapitel 4 eingeführtem verkleinerten Siebintervall $\left[\log\det(1 + k\sigma^{-2}), km\right]$. Dabei bezeichnet $m$ den Zielfunktionswert einzelner Elemente $\frac{1}{2}\log(1 + \sigma^{-2})$ und $k$ die geforderte Größe der Zusammenfassung. Es werden die folgenden Vergleichswerte ermittelt:

- **Greedy:** Die Ausgabe des *Greedy* Algorithmus (Algorithmus 3.1).
- **Sieve Streaming:** Die Ausgabe von *Sieve Streaming* (Algorithmus 3.2) mit originalem Siebintervall.
- **Sieve Streaming$^*$:** Die Ausgabe von *Sieve Streaming* (Algorithmus 3.2) mit verkleinertem Siebintervall.
- **Zufall:** Die Ausgabe einer zufälligen Auswahl durch *Reservoir Sampling* (Algorithmus 3.3).

Als Wert einer Ausgabe wird der Zielfunktionswert der berechneten Zusammenfassung ermittelt. Die Berechnungskosten entsprechen der Anzahl der Auswertungen der submodularen Zielfunktion (*oracle queries*). Dadurch sind die Ergebnisse implementierungs- und plattformunabhängig. Weiterhin werden die Beobachtungsvektoren des Datensatzes auf Mittelwert null und Varianz eins normalisiert. Als Zielfunktion wird die monoton submodulare Funktion $f(S) = \frac{1}{2} \log\det(\mathcal{I} + \sigma^{-2}\Sigma_S)$ und als Kernfunktion die Gauß-Kernfunktion $\mathcal{K}(x, y) = \exp\left(-\frac{||x-y||_2^2}{2h^2}\right)$ benutzt. Außerdem werden die Parameter $\sigma = 1$, $h = 1$ und für die *Sieve Streaming* Algorithmen $\epsilon = 0.1$ gewählt. Insbesondere werden die Berechnungskosten für ein festes $k = 20$ verglichen.

### 5.1.3 Ergebnisse

Abbildung 5.1 (a) enthält Werte und Kosten der vier Algorithmen für eine feste Größe der Zusammenfassung $k = 20$. Die Ergebnisse wurden auf die des *Greedy* Algorithmus normalisiert. Der Datensatz ist klein genug für die Anwendung des *Greedy* Algorithmus, welcher hier den höchsten Zielfunktionswert erzielt. Gleichzeitig hat er den höchsten Berechnungsaufwand, weil er $k$ mal über den gesamten Datensatz iteriert. Die *Sieve Streaming* Algorithmen erreichen beide 96 % des Wertes des *Greedy* Lösung. Die Kosten liegen mit originalem Siebintervall bei 59 % und mit verkleinertem Siebintervall bei 18 % der Kosten der *Greedy* Lösung. Durch die Verkleinerung des Intervalls der Siebe wurde also ein identi-

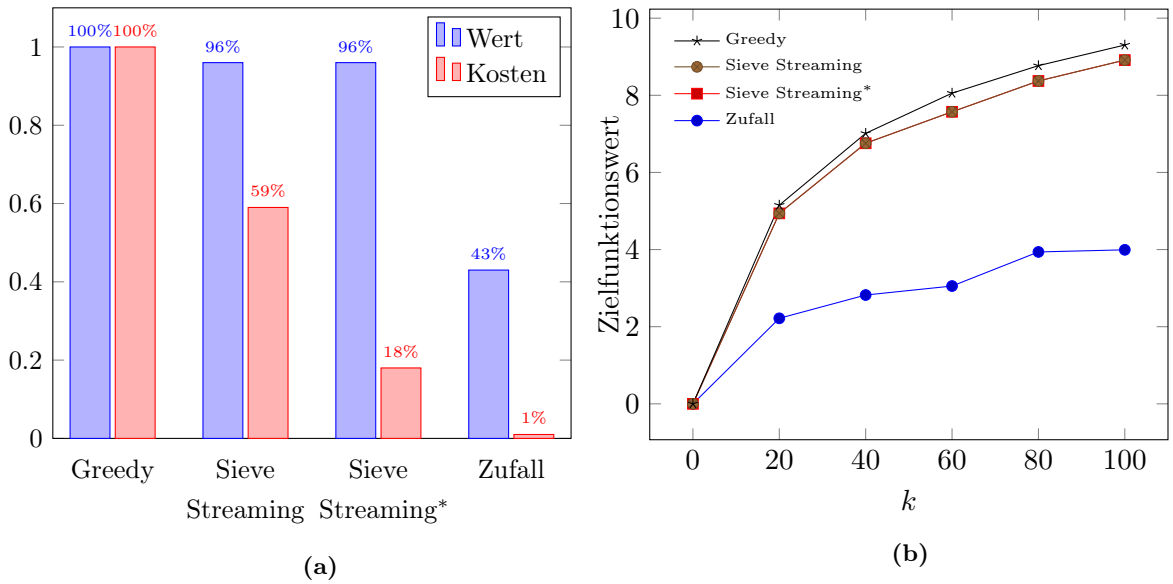



**Abbildung 5.1:** Werte und Kosten des *Sieve Streaming* Algorithmus mit originalem und mit verkleinertem Siebintervall im Vergleich zum *Greedy* Algorithmus und zur zufälligen Auswahl. Abbildung (a) zeigt die erhaltenen Zielfunktionswerte und den Berechnungsaufwand der Algorithmen relativ zum *Greedy* Algorithmus für ein festes $k = 20$. Abbildung (b) vergleicht die Zielfunktionswerte der Algorithmen für verschiedene Werte von $k$.

scher Zielfunktionswert ermittelt und zwei Drittel der Berechnungskosten eingespart. Die zufällige Auswahl einer Zusammenfassung hat die geringsten Kosten, erzielt aber nur 43 % des Wertes der *Greedy* Lösung.
Abbildung 5.1 (b) zeigt die Zielfunktionswerte der Ausgaben der vier Algorithmen für verschiedene Werte von $k$. Dabei erzielt der *Greedy* Algorithmus die höchsten Werte. Die Sieve Streaming Algorithmen erzielen für alle $k$ identische Lösungen und Zielfunktionswerte, die mindestens 94 % der Werte der *Greedy* Lösungen betragen. Die zufällige Auswahl erreicht nur Werte zwischen 40 und 45 % der Werte der *Greedy* Lösungen.
Das Experiment zeigt, dass es möglich ist, einen Großteil der Berechnungskosten durch Verwendung des verkleinerten Siebintervalls einzusparen. In diesem Beispiel wurden mit einem Drittel der Berechnungskosten identische Zusammenfassungen berechnet. Die erzielten Ergebnisse sind mit den Ergebnissen von Badanidiyuru et al. [2] vergleichbar.

## 5.2 Anomalieerkennung

### 5.2.1 Das FACT Teleskop

Das *First G-APD Cherenkov Telescope* (FACT) ist seit 2011 auf Gran Canaria (Spanien) in Betrieb (Abbildung 5.2) und beobachtet kosmische Schauer. Durch die Interaktion mit Teilchen in der Atmosphäre emittieren sie Cherenkov-Licht. Das Cherenkov-Licht kann mithilfe einer Kamera aus 1440 hexagonalen *Geiger-mode avalanche photodiodes* (G-APDs) gemessen werden. Diese Ereignisse (*events*) dauern nur etwa 150 Nanosekunden an. Sie können durch verschiedene Auslöser erkannt und dann durch die Kamera mit einer Samplerate von zwei Gigahertz aufgenommen werden. Ein Ereignis besteht damit aus $1440 \cdot 300 = 432000$ Messwerten. Die Kamera kann dabei in fünf Minuten bis zu zehn Gigabyte Rohdaten aufnehmen [4]. Ein Ziel des Projekts ist die Analyse dieser Daten. Sowohl die große Menge der Daten als auch die hohe Geschwindigkeit, in der diese eintreffen, erschweren das Problem. Echtzeitanalysen erfordern den Einsatz von Datenstromalgorithmen. Eine Möglichkeit stellt die Extraktion der Anzahlen der Photonenankünfte (*photon arrival counts*) an jedem Pixel aus den Messwerten eines Ereignisses dar. So kann die Dimensionalität der Daten von 432000 auf 1440 reduziert werden.

### 5.2.2 Anomalien

Anomalien sind Datenpunkte mit ungewöhnlichen Eigenschaften. Es existieren zahlreiche Methoden zur Erkennung dieser besonders interessanten Daten. Die meisten wurden speziell für bestimmte Einsatzbereiche entwickelt und erfordern zuerst die Definition von gewöhnlichen Daten. Für eine generische Anomalieerkennung kann die Datenzusammenfassung durch Maximierung einer submodularen Funktionen eingesetzt werden. Bei einer geeigneten Konfiguration wird die Vielfalt der Zusammenfassung maximiert. Es werden

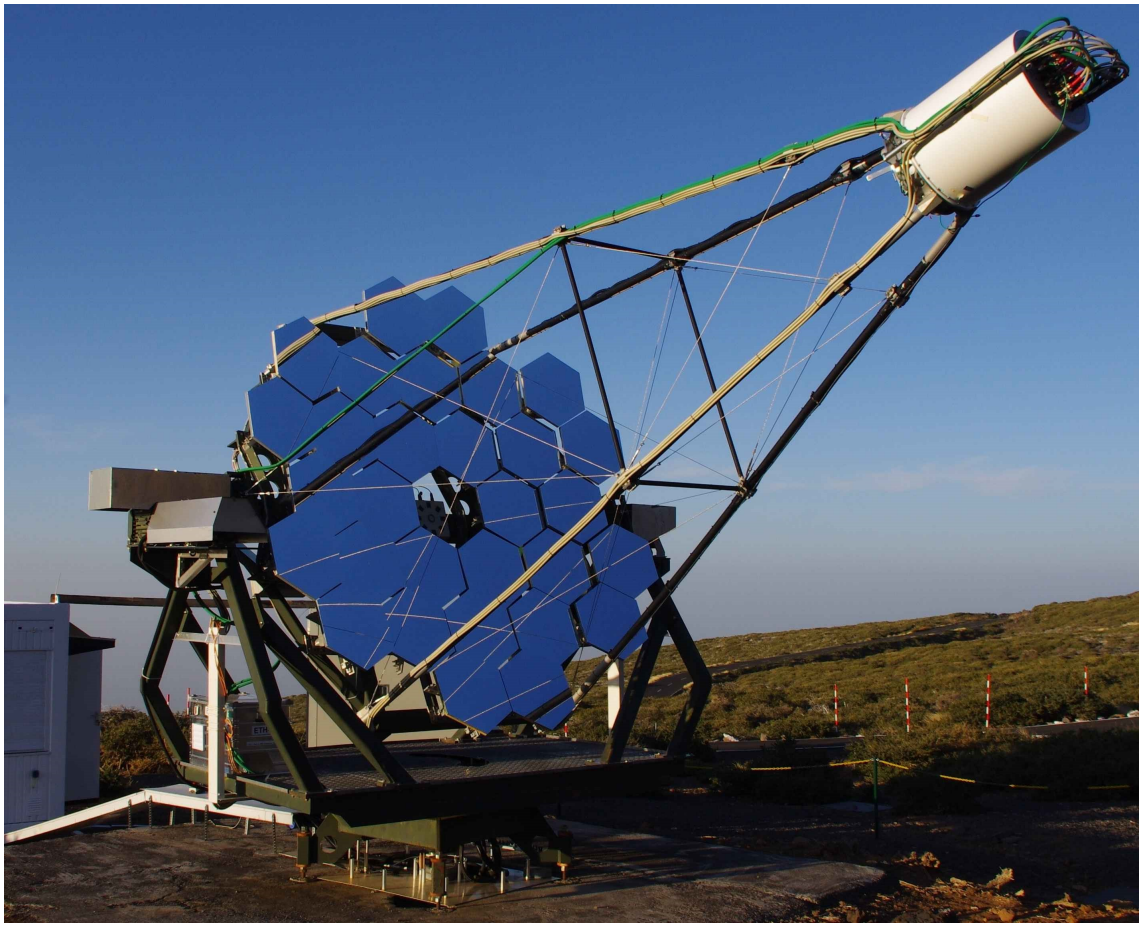


**Abbildung 5.2:** Das FACT Teleskop auf Gran Canaria [1]. Die Teleskopdaten entstehen durch die Aufnahmen einer Kamera (rechts oben), die auf eine Spiegelfläche gerichtet ist.

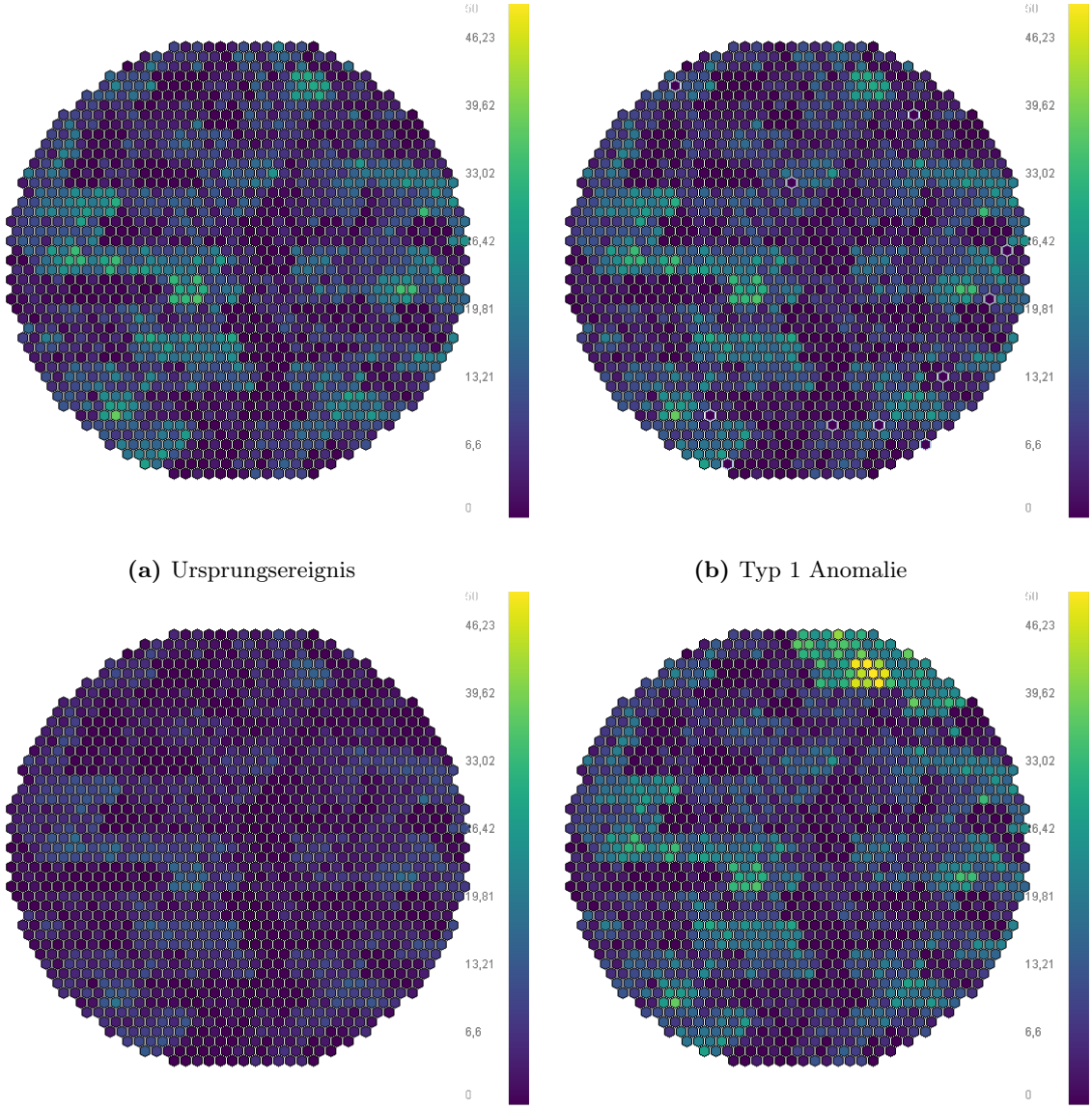
dann Datenpunkte in die Zusammenfassung aufgenommen, die sich stark von bisher gesehenen Daten unterscheiden. Dadurch werden gewöhnliche Daten implizit als die bisher gesehenen Daten definiert. Die erhaltene Zusammenfassung muss anschließend mit weiteren Verfahren oder manuell untersucht werden. Danach können Aussagen über unterschiedliche Klassen von Ereignissen im Datenstrom gemacht werden. Möglicherweise können die Ereignisse in der Zusammenfassung sogar direkt als Repräsentanten dieser Klassen interpretiert werden. Die Güte dieses Verfahrens wird jetzt experimentell untersucht.
Als interessante Ereignisse werden Anomalien durch Manipulation der Photonenanzahlen simuliert. So kann leicht festgestellt werden, ob sie in die Zusammenfassungen aufgenommen wurden. Es werden drei Typen von Anomalien erzeugt:

- **Typ 1:** Ereignis mit toten Pixeln. Die Anzahl der gemessenen Photonen wird für einzelne Pixel auf null gesetzt.

- **Typ 2:** Ereignis mit verringerter Photonenanzahl. Die Anzahl der gemessenen Photonen wird für alle Pixel um einen Faktor verringert.

- **Typ 3:** Ereignis mit erhöhter Photonenanzahl. Die Anzahl der gemessenen Photonen werden in einem Bereich von Pixeln um einen konstanten Wert erhöht.

Durch zufällige Anwendung dieser Manipulationen können große Mengen ungewöhnlicher Ereignisse generiert werden, aber auch Ereignisse, die möglicherweise auf technische Probleme hinweisen. Abbildung 5.3 enthält Beispiele für solche Anomalien für ein typisches Ursprungsereignis.




**(a)** Ursprungsereignis

**(b)** Typ 1 Anomalie

**(c)** Typ 2 Anomalie

**(d)** Typ 3 Anomalie

**Abbildung 5.3:** Anzahlen gemessener Photonen vor und nach Manipulation des Ursprungsereignisses.

### 5.2.3 Wahl der Kernfunktionen

Die in Kapitel 2 vorgestellten stationären Kernfunktionen basieren auf dem Manhattan-Abstand oder dem Euklidischen Abstand

$$d_{L_2}(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

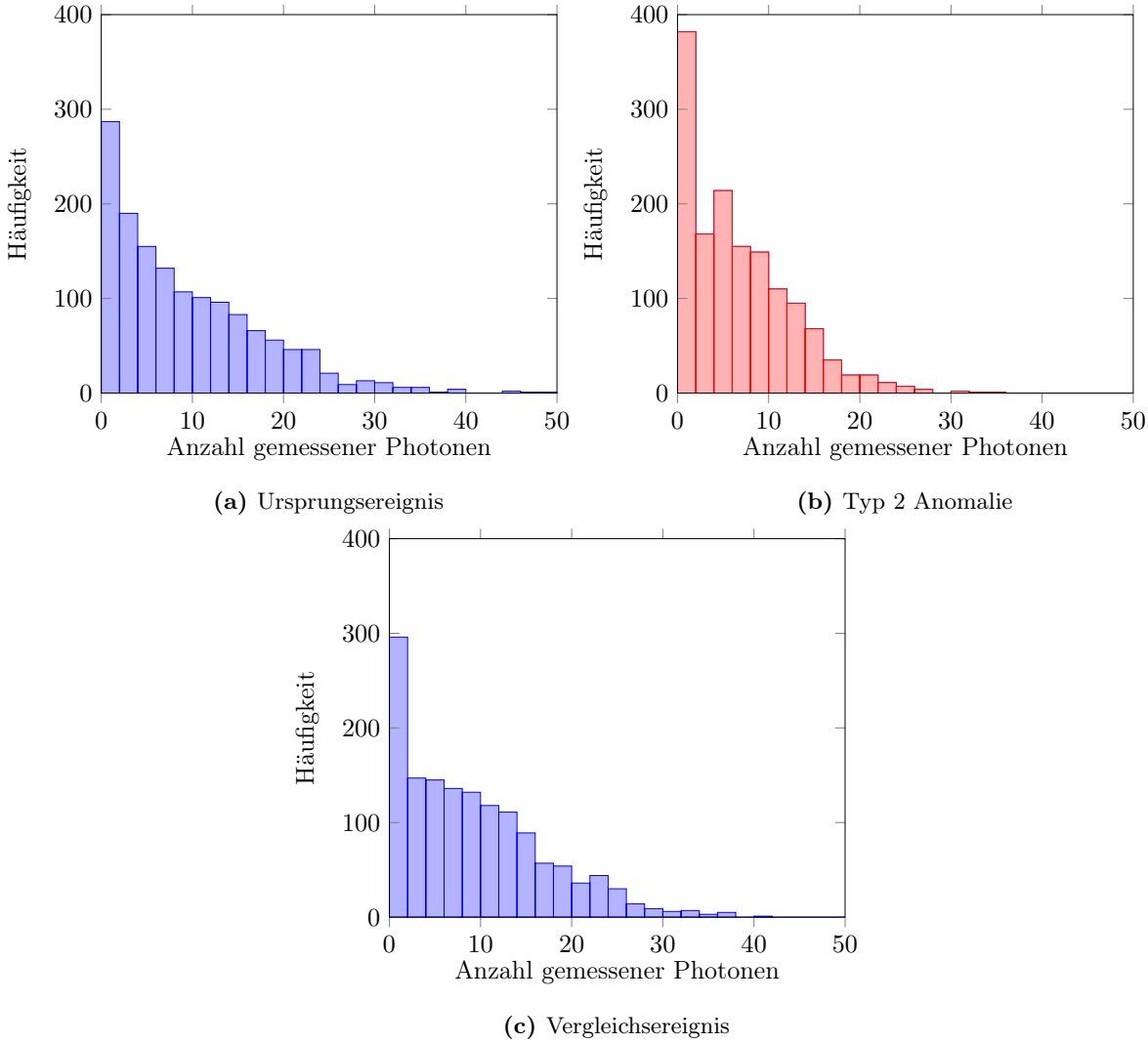
der Beobachtungen $x$ und $y$. Ein Beispiel soll auf eine Problematik des Euklidischen Abstands bei der Erkennung der beschriebenen Anomalien hinweisen. Dazu werden die Anzahlen der gemessenen Photonen eines Ursprungsereignis um 30 % reduziert (Typ 2 Anomalie). Der Euklidische Abstand eines unabhängigen Vergleichsereignisses zum Ursprungsereignis beträgt 217.28, während der Abstand vom Vergleichsereignis zur Anomalie 220.71 beträgt. Die Abstände unterscheiden sich nur sehr geringfügig, mit dem Euklidischen Abstand kann man Anomalien dieser Art nur vereinzelt erkennen.
Betrachtet man die Verteilung der gemessenen Photonen vor und nach der Manipulation (Abbildung 5.4) ist ersichtlich, dass Pearsons $\chi^2$-Test [47] den Unterschied der Verteilungen leicht erkennen kann. Der $\chi^2$-Test kann auf Gleichheit der Verteilungen von zwei Stichproben testen und basiert auf dem $\chi^2$-Abstand

$$d_{\chi^2}(\tilde{x}, \tilde{y}) = \sum_{i=1}^{m} \frac{(\tilde{x}_i - \tilde{y}_i)^2}{\tilde{x}_i + \tilde{y}_i},$$

einer gewichteten Summe der quadrierten Abstände der Häufigkeitsvektoren $\tilde{x}, \tilde{y} \in \mathbb{N}^m$ der Beobachtungen $x$ und $y$. Diese Metrik gewichtet die Abstände der Häufigkeiten mit dem Kehrwert der Anzahl des Auftretens. Der $\chi^2$-Abstand des Vergleichsereignisses zum Ursprungsereignis beträgt 41.43, während der Abstand vom Vergleichsereignis zur Anomalie 1083.13 beträgt. Das ist ein 25 mal höherer Wert, mit dem $\chi^2$-Abstand kann man Anomalien dieser Art sehr leicht als ungewöhnliche Ereignisse identifizieren. Untersuchungen zeigen die Überlegenheit des $\chi^2$-Abstand über dem Euklidischen Abstand bei Text- und Bildklassifizierungen [34]. Tabelle 5.1 vergleicht den euklidischen Abstand mit den $\chi^2$-Abstand für weitere Anomalien. Für Anomalie 1 wurde die Anzahl der gemessenen Photonen von 2 % der Pixel auf null gesetzt und für Anomalie 3 wurden die gemessenen Photonen in einem

| | | Anomalie | | |
|---|---|---|---|---|
| **Abstand** | Ursprungsereignis | Typ 1 | Typ 2 | Typ 3 |
| Euklid | 217.28 | 224.25 | 220.71 | 326.33 |
| $\chi^2$ | 41.43 | 50.67 | 1083.13 | 105.22 |

**Tabelle 5.1:** Euklidischer Abstand und $\chi^2$-Abstand eines Vergleichsereignisses zu einem Ursprungsereignis und zu Anomalien, die durch Manipulation des Ursprungsereignisses generiert wurden.




**(a)** Ursprungsereignis

**(b)** Typ 2 Anomalie

**(c)** Vergleichsereignis

**Abbildung 5.4:** Histogramme gemessener Photonen (a) vor und (b) nach Manipulation (Typ 2 Anomalie) im Vergleich mit (c) einem Vergleichsereignis. Der Euklidische Abstand vom Vergleichsereignis zum Ursprungsereignis und zur Anomalie ist fast identisch, während der $\chi^2$-Abstand zur Anomalie 25 mal höher ist als zum Ursprungsereignis.

Bereich von zehn Prozent aller Pixel um 20 erhöht.
Weil man mit dem $\chi^2$-Abstand bei diesen Beispielen die Anomalien deutlich besser identifizieren kann, werden zwei nicht-stationäre Kernfunktionen hergeleitet, die auf dem $\chi^2$-Abstand basieren. Der $\chi^2$-Abstand ist Grundlage für die $\chi^2$-Kernfunktion

$$\mathcal{K}_{\chi^2}(x, y) = 1 - \frac{1}{2} \sum_{i=1}^{n} \frac{(x_i - y_i)^2}{x_i + y_i} .$$

Der Gebrauch dieser Kernfunktion ist verbreitet bei Anwendungen der *Computer Vision* [58]. Außerdem ist sie parameterfrei und benötigt kein Wissen über die Daten. Sie ist aber nur bedingt positiv definit [51], das heißt die Gleichung aus Definition 2.4.1 ist nicht

für alle $c_i \in \mathbb{R}$ erfüllt. Aus einer bedingt positiv definiten Kernfunktionen $\mathcal{K} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ kann mit $z \in \mathcal{X}$ eine positiv definite Kernfunktion

$$\mathcal{K}'(x, y) = \mathcal{K}(x, y) - \mathcal{K}(x, z) - \mathcal{K}(z, x) + \mathcal{K}(z, z)$$

konstruiert werden [57]. Damit erhält man aus der $\chi^2$-Kernfunktion und $z = \vec{0}$ die positiv definite Kernfunktion

$$\begin{aligned}
\mathcal{K}'_{\chi^2}(x, y) &= \mathcal{K}_{\chi^2}(x, y) - \mathcal{K}_{\chi^2}(x, \vec{0}) - \mathcal{K}_{\chi^2}(\vec{0}, y) + \mathcal{K}_{\chi^2}(\vec{0}, \vec{0}) \\
&= 1 - \frac{1}{2} \sum_{i=1}^{n} \frac{(x_i - y_i)^2}{x_i + y_i} - 1 + \frac{1}{2} \sum_{i=1}^{n} \frac{x_i^2}{x_i} - 1 + \frac{1}{2} \sum_{i=1}^{n} \frac{y_i^2}{y_i} + 1 \\
&= \frac{1}{2} \sum_{i=1}^{n} (x_i + y_i) - \frac{1}{2} \sum_{i=1}^{n} \frac{(x_i - y_i)^2}{x_i + y_i} \,.
\end{aligned}$$

Für den Häufigkeitsvektoren $\tilde{x} \in \mathbb{N}^m$ von $x$ ist $\sum_{i=1}^{m} \tilde{x}_i = n$ konstant. Damit kann die Kernfunktion für Häufigkeitsvektoren vereinfacht werden:

$$\begin{aligned}
\mathcal{K}'_{\chi^2}(\tilde{x}, \tilde{y}) &= n - \frac{1}{2} \sum_{i=1}^{n} \frac{(\tilde{x}_i - \tilde{y}_i)^2}{\tilde{x}_i + \tilde{y}_i} \\
&= n \left( 1 - \frac{1}{2n} \sum_{i=1}^{m} \frac{(\tilde{x}_i - \tilde{y}_i)^2}{\tilde{x}_i + \tilde{y}_i} \right) \\
&= n \left( 1 - \frac{1}{2} \sum_{i=1}^{m} \frac{(\frac{\tilde{x}_i}{n} - \frac{\tilde{y}_i}{n})^2}{\frac{\tilde{x}_i}{n} + \frac{\tilde{y}_i}{n}} \right) \\
&= n \cdot \mathcal{K}_{\chi^2} \left( \frac{\tilde{x}}{n}, \frac{\tilde{y}}{n} \right) .
\end{aligned}$$

Mit Korollar 2.4.4 erhält man wieder die $\chi^2$-Kernfunktion, sie ist also für relative Häufigkeiten positiv definit.
Alternativ kann man aus einer bedingt positiv definiten Kernfunktion $\mathcal{K} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ eine positiv definite Kernfunktion

$$\mathcal{K}''(x, y) = \exp(t \cdot \mathcal{K}(x, y))$$

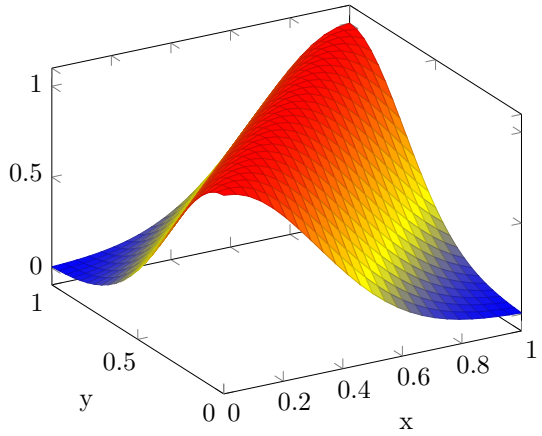
mit $t > 0$ konstruieren [50]. Für die $\chi^2$-Kernfunktion ergibt sich die Kernfunktion

$$\begin{aligned}
\mathcal{K}''_{\chi^2}(x, y) &= \exp \left( t - \frac{t}{2} \sum_{i=1}^{n} \frac{(x_i - y_i)^2}{x_i + y_i} \right) \\
&= \exp(t) \exp \left( -\frac{t}{2} \sum_{i=1}^{n} \frac{(x_i - y_i)^2}{x_i + y_i} \right) .
\end{aligned}$$

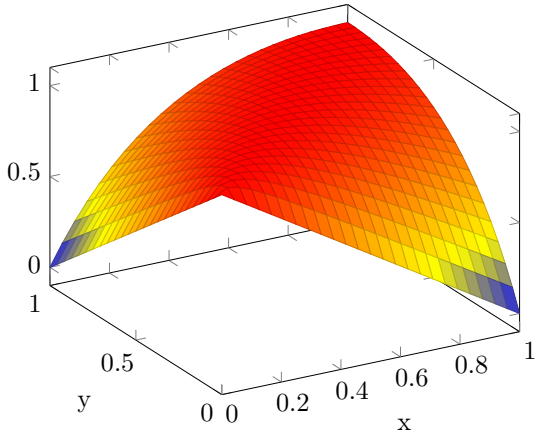
Mit Korollar 2.4.4 und $t = \frac{1}{h^2}$ erhält man die positiv definite Exponentielle $\chi^2$-Kernfunktion

$$\mathcal{K}_{\exp \chi^2}(x, y) = \exp \left( -\frac{1}{2h^2} \sum_{i=1}^{n} \frac{(x_i - y_i)^2}{x_i + y_i} \right) .$$

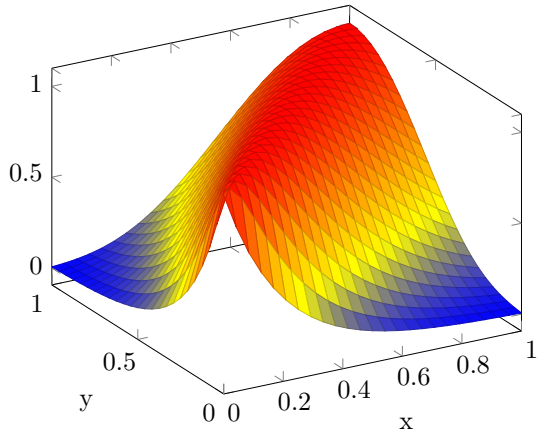
Diese Kernfunktion profitiert von Eigenschaften der $\chi^2$-Kernfunktion und der exponentiellen Kernfunktionen [54]. Die auf dem $\chi^2$-Abstand basierenden Kernfunktionen sind in Abbildung 5.5 im Vergleich mit der Gauß-Kernfunktion dargestellt.




**(a)** Gauß-Kernfunktion




**(b)** $\chi^2$-Kernfunktion




**(c)** Expontentielle $\chi^2$-Kernfunktion

**Abbildung 5.5:** Vergleich zwischen der Gauß-Kernfunktion und auf dem $\chi^2$-Abstand basierenden Kernfunktionen mit $x, y \in \mathbb{R}$.

### 5.2.4 Aufbau des Experiments

Im Folgenden wird das Erkennen von Anomalien durch die Berechnung von Zusammenfassungen auf Teleskopdaten untersucht. Dazu werden simulierte Gammastrahlenereignisse aus 100 Simulationsläufen benutzt. Diese bestehen aus 512 bis 553 Ereignissen, aus denen die Anzahl der Photonenankünfte an jedem Pixel berechnet wird. Mit einer Wahrscheinlichkeit von zwei Prozent wird ein Ereignis zu einer der vorgestellten Anomalien manipuliert, sodass ein Lauf etwa zehn zufällige Anomalien beinhaltet. Anschließend wird jeder Lauf mit dem *Sieve Streaming* Algorithmus auf eine Zusammenfassung aus 20 Ereignissen reduziert. Dazu wird die Zielfunktion $f(S) = \frac{1}{2} \log \det(\mathcal{I} + \sigma^{-2}\Sigma_S)$ verwendet, weil sie die Vielfalt der Zusammenfassung maximiert.

Es werden zwei Experimente durchgeführt:

- **Experiment 1:** Beim ersten Experiment werden starke Anomalien erzeugt, von denen jeder Typ mindestens einmal in die Zusammenfassung aufgenommen werden soll. Für die Typ 1 Anomalie werden zehn Prozent der Pixel auf null gesetzt, die Typ 2 Anomalie reduziert die Photonenanzahl aller Pixel um 40 Prozent. Typ 3 erhöht die Photonenzahl in einem Bereich von zehn Prozent der Pixel um 20.

- **Experiment 2:** Beim zweiten Experiment werden Anomalien erzeugt, die nur noch sehr leicht von gewöhnlichen Daten abweichen. Von diesen soll auch hier jeder Typ mindestens einmal in die Zusammenfassung aufgenommen werden. Für die Typ 1 Anomalie werden nur zwei Prozent der Pixel auf Null gesetzt. Die Typ 2 Anomalie reduziert die Photonenanzahl um 15 Prozent, während Typ 3 sie in einem Bereich von zehn Prozent aller Pixel um zehn erhöht.

Dabei wird die Gauß-Kernfunktion mit den auf dem $\chi^2$-Abstand basierenden Kernfunktionen hinsichtlich der Erkennung der Anomalien verglichen. Weil für diese Kernfunktionen $\mathcal{K}(x, x) = 1$ gilt, kann durch die Erkenntnisse in Kapitel 4 ein verkleinertes Siebintervall genutzt werden. Für die Parameter $h$ der Gauß-Kernfunktion und der Exponentiellen $\chi^2$-Kernfunktion wurde initial die Anzahl der Dimensionen der Eingabedaten 1440 und 50 gewählt. Sie wurden anschließend jeweils experimentell auf 500 und 20 angepasst. Die Experimente werden für alle drei Kernfunktionen auf den Daten der 100 verschiedenen Simulationsläufe wiederholt. Dadurch kann die Wahrscheinlichkeit der Erfüllung der Anforderungen als Erfolgsanteil angegeben werden. Für einen Vergleich wird auch eine zufällige Menge aus Ereignissen durch *Reservoir Sampling* (Algorithmus 3.3) ausgewählt.

### 5.2.5 Ergebnisse

In Experiment 1 (Tabelle 5.2) wurden starke Anomalien erzeugt. Mit der Gauß-Kernfunktion wurde eine Anomalie des Typs 1 nur zu 50 % erkannt, während Typ 2 und 3 Anomalien mit jeweils 99 % und 100 % fast immer erkannt wurden. Für die $\chi^2$-Kernfunktion stellte die Anomalie des Typs 3 mit 94 % die größte Herausforderung dar. Typ 1 und 2 Anomalien wurden mit jeweils 99 % und 100 % fast immer erkannt. Das beste Ergebnis lieferte die Exponentielle $\chi^2$-Kernfunktion mit einem Erfolgsanteil von jeweils 99 %, 100 % und 99 %. Hier waren in fast allen Zusammenfassungen Anomalien jedes Typs.
In Experiment 2 (Tabelle 5.3) wurden nur noch sehr schwache Anomalien erzeugt. Mit der Gauß-Kernfunktion wurden Typs 1 und 2 Anomalien nicht mehr erkannt und nur noch zufällig zu jeweils 15 % und 12 % in die Zusammenfassung aufgenommen. Anomalien des Typs 3 wurden deutlich häufiger zu 57 % erkannt. Beide $\chi^2$-Kernfunktionen erzielten hier ähnliche Ergebnisse und erkannten Typ 1 Anomalien mit jeweils 75 % und 78 % und Typ 2 Anomalien mit jeweils 92 % und 90 %. Die Exponentielle $\chi^2$ Kernfunktion schneidet bei Anomalien des Typs 3 mit 81 % besser ab als die $\chi^2$-Kernfunktion mit 67 %, sodass sie insgesamt auch in diesem Experiment das beste Ergebnis liefert.

| | Erfolgsanteil | | |
|---|---|---|---|
| **Kernfunktion** | Typ 1 | Typ 2 | Typ 3 |
| Gauß-Kernfunktion | 51% | 99% | **100%** |
| $\chi^2$-Kernfunktion | **99%** | **100%** | 94% |
| Exponentielle $\chi^2$-Kernfunktion | **99%** | **100%** | 99% |
| Zufällige Auswahl | 14% | 13% | 16% |

**Tabelle 5.2:** Erfolgsanteil für verschiedene Kernfunktionen bei Verwendung starker Anomalien.

| | Erfolgsanteil | | |
|---|---|---|---|
| **Kernfunktion** | Typ 1 | Typ 2 | Typ 3 |
| Gauß-Kernfunktion | 15% | 12% | 57% |
| $\chi^2$-Kernfunktion | 75% | **92%** | 67% |
| Exponentielle $\chi^2$-Kernfunktion | **78%** | 90% | **81%** |
| Zufällige Auswahl | 10% | 12% | 15% |

**Tabelle 5.3:** Erfolgsanteil für verschiedene Kernfunktionen bei Verwendung schwacher Anomalien.

Die Ergebnisse sind vereinbar mit dem *No-Free-Lunch*-Theorem. Es existiert keine universelle Kernfunktion, die auf allen Daten die beste Lösung liefert. Dennoch erzielen die auf dem $\chi^2$-Abstand basierenden Kernfunktionen besonders in Experiment 2 deutlich bessere Ergebnisse. Die Exponentielle $\chi^2$-Kernfunktion benötigt allerdings eine sorgfältige Anpassung des Parameters, während die $\chi^2$-Kernfunktion parameterfrei ist und ohne Kenntnisse über die Daten sehr ähnliche Ergebnisse liefert. Nur bei Anomalien des Typs 3 hat sie einen signifikant kleineren Erfolgsanteil. Insgesamt scheint es möglich zu sein, stark ausgeprägte Anomalien durch die Berechnung einer Zusammenfassung fast sicher zu erkennen. Bei schwacher Ausprägung der Anomalien, sodass diese kaum noch als Anomalien bezeichnet werden können, ist hier trotzdem ein Erfolgsanteil von mindestens 78 % möglich.

# Kapitel 6

# Fazit und Ausblick

In der vorliegenden Arbeit wurde die Berechnung von Zusammenfassungen auf Datenströmen untersucht. Hierzu wurden zunächst submodulare Funktionen und wichtige Eigenschaften dieser vorgestellt. Es wurde gezeigt, dass sie in vielen realen Anwendungen auftauchen und zur Modellierung zahlreicher Optimierungsprobleme verwendet werden können. Die Zusammenfassung von Daten konnte als Maximierung einer monoton submodularen Zielfunktion unter Kardinalitätsbeschränkung modelliert werden.
Anschließend wurde der *Sieve Streaming* Algorithmus mit einer effizienten Implementierung als Datenstromalgorithmus mit der höchsten Approximationsgarantie zum aktuellen Forschungsstand vorgestellt. Für eine Zielfunktion ist dabei das unnötig große Siebintervall aufgefallen. Durch Abschätzung der möglichen Zielfunktionswerte konnte ein minimales Intervall hergeleitet werden, das mit deutlich weniger Sieben auskommt. Es bleibt zu untersuchen, ob das Siebintervall für weitere Zielfunktionen verkleinert werden kann. Festgestellt wurde auch die effektive Parallelisierbarkeit des Algorithmus.
Im ersten Experiment wurde die Performanz des *Sieve Streaming* Algorithmus mit verkleinertem Siebintervall ermittelt. Im Vergleich zum originalen Siebintervall konnten die Berechnungskosten durch die Verkleinerung auf ein Drittel reduziert werden. Die berechneten Zusammenfassungen waren mit beiden Intervallen identisch. Hiermit sei die Wiederholung des Experiments auf einem größeren Datensatz oder einem Datenstrom angeregt, um eine bessere Beurteilung der Einsparung zu ermöglichen.
Im zweiten Experiment wurde der *Sieve Streaming* Algorithmus für die Erkennung von Anomalien in einem Datenstrom eingesetzt. Durch Nutzung einer geeigneten Kernfunktion wurden fast alle Anomalien mit starker Ausprägung erkannt. Von sehr schwach ausgeprägten Anomalien konnte trotzdem noch ein Großteil erkannt werden. Es wurde gezeigt, dass zu den vielen Anwendungsgebieten der Optimierung von submodularen Funktion auch die Erkennung von Anomalien gezählt werden kann. Ein Vergleich mit üblichen Verfahren der Anomalieerkennung steht allerdings noch aus.

# Anhang A

# Normierung von Kernfunktionen

Es soll gezeigt werden, dass sich beliebige positiv definite Kernfunktionen in das Intervall $[-1, 1]$ normieren lassen und sie dabei positiv definit bleiben.
Sei $\mathcal{K} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ eine positiv definite Kernfunktion. Dann existiert eine Abbildung $\phi : \mathcal{X} \to \mathcal{F}$, sodass $\mathcal{K}(x, y)$ das Skalarprodukt der Bildvektoren von $x$ und $y$ im Skalarvektorraum $\mathcal{F}$ repräsentiert [40]:

$$\mathcal{K}(x, y) = \langle \phi(x), \phi(y) \rangle \,.$$

Es folgt mit der Cauchy-Schwarz-Ungleichung (1):

$$\begin{aligned} |\mathcal{K}(x, y)| &= |\langle \phi(x), \phi(y) \rangle| \\ &\overset{(1)}{\leq} ||\phi(x)||_2 \cdot ||\phi(y)||_2 \\ &= \sqrt{\langle \phi(x), \phi(x) \rangle} \cdot \sqrt{\langle \phi(y), \phi(y) \rangle} \\ &= \sqrt{\mathcal{K}(x, x)\mathcal{K}(y, y)} \,. \end{aligned} \tag{A.1}$$

Aus $\mathcal{K}$ kann eine normierte Kernfunktion $\mathcal{K}'$ konstruiert werden [49] mit

$$\mathcal{K}'(x, y) = \begin{cases} 0, & \text{falls } \mathcal{K}(x, x) = 0 \text{ oder } \mathcal{K}(y, y) = 0 \\ \frac{\mathcal{K}(x,y)}{\sqrt{\mathcal{K}(x,x)\mathcal{K}(y,y)}}, & \text{sonst.} \end{cases}$$

Wegen (A.1) gilt dann:

$$\begin{aligned} & -\sqrt{\mathcal{K}(x, x)\mathcal{K}(y, y)} \leq \mathcal{K}\,(x, y) \leq \sqrt{\mathcal{K}(x, x)\mathcal{K}(y, y)} \\ \Rightarrow \qquad & -1 \leq \mathcal{K}'(x, y) \leq 1 \,. \end{aligned}$$

Also bildet die Normierung $\mathcal{K}'$ positiv definiter Kernfunktionen auf das Intervall $[-1, 1]$ ab. Insbesondere gilt $\mathcal{K}'(x, x) = 1$, wenn $\mathcal{K}(x, x) \neq 0$.

Zum Schluss wird die Positiv-Definitheit von $\mathcal{K}'$ überprüft. Sei dazu $\{x_1, \ldots, x_n\} \in \mathcal{X}^n$ mit $n \in \mathbb{N}$ beliebig und $m$ die Dimensionalität des Merkmalsraums $\mathcal{F}$:

$$\begin{aligned}
\sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j \frac{\mathcal{K}(x_i, x_j)}{\sqrt{\mathcal{K}(x_i, x_i)\mathcal{K}(x_j, x_j)}} &= \sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j \frac{\langle \phi(x_i), \phi(x_j) \rangle}{\sqrt{\mathcal{K}(x_i, x_i)\mathcal{K}(x_j, x_j)}} \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j \frac{\sum_{k=1}^{m} \phi(x_i)_k \, \phi(x_j)_k}{\sqrt{\mathcal{K}(x_i, x_i)\mathcal{K}(x_j, x_j)}} \\
&= \sum_{k=1}^{m} \left( \sum_{i=1}^{n} c_i \frac{\phi(x_i)_k}{\sqrt{\mathcal{K}(x_i, x_i)}} \cdot \sum_{j=1}^{n} c_j \frac{\phi(x_j)_k}{\sqrt{\mathcal{K}(x_j, x_j)}} \right) \\
&= \sum_{k=1}^{m} \underbrace{\left( \sum_{i=1}^{n} c_i \frac{\phi(x_i)_k}{\sqrt{\mathcal{K}(x_i, x_i)}} \right)^2}_{\geq 0} \geq 0 \,.
\end{aligned}$$

Mit Definition 2.4.1 ist $\mathcal{K}'$ also positiv definit.

**Bemerkung.** Graf et al. [21] untersuchen die Normierung im Merkmalsraum und zeigen, dass für die implizite Merkmalsabbildung $\phi'$ von $\mathcal{K}'$ gilt:

$$\phi'(x) = \frac{\phi(x)}{||\phi(x)||_2} \,.$$

Dadurch kann die normierende Wirkung erklärt werden. Die Bildpunkte der Merkmalsabbildung werden auf den Rand der $m$-dimensionalen Einheitskugel verschoben.

# Tabellenverzeichnis

# Abbildungsverzeichnis

# Algorithmenverzeichnis

# Literaturverzeichnis


[1] Anderhub, H, M Backes, A Biland, Vittorio Boccone, I Braun, T Bretz, J Buss, Franck Cadoux, V Commichau, L Djambazov et al.: *Design and operation of FACT–the first G-APD Cherenkov telescope.* Journal of Instrumentation, 8(06):P06008, 2013.

[2] Badanidiyuru, Ashwinkumar, Baharan Mirzasoleiman, Amin Karbasi und Andreas Krause: *Streaming submodular maximization: Massive data summarization on the fly.* In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, Seiten 671–680. ACM, 2014.

[3] Barbosa, Rafael, Alina Ene, Huy Nguyen und Justin Ward: *The power of randomization: Distributed submodular maximization on massive datasets.* In: *International Conference on Machine Learning*, Seiten 1236–1244, 2015.

[4] Bockermann, Christian und Hendrik Blom: *The streams framework.* TU Dortmund University, Tech. Rep, 5:12, 2012.

[5] Boykov, Yuri Y und M-P Jolly: *Interactive graph cuts for optimal boundary & region segmentation of objects in ND images.* In: *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, Band 1, Seiten 105–112. IEEE, 2001.

[6] Brooks, Bernard P: *The coefficients of the characteristic polynomial in terms of the eigenvalues and the elements of an n× n matrix.* Applied mathematics letters, 19(6):511–515, 2006.

[7] Buchbinder, Niv, Moran Feldman, Joseph Seffi Naor und Roy Schwartz: *Submodular maximization with cardinality constraints.* In: *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, Seiten 1433–1452. Society for Industrial and Applied Mathematics, 2014.

[8] Buchbinder, Niv, Moran Feldman und Roy Schwartz: *Online submodular maximization with preemption.* In: *Proceedings of the Twenty-Sixth Annual ACM-SIAM*

*Symposium on Discrete Algorithms*, Seiten 1202–1216. Society for Industrial and Applied Mathematics, 2015.

[9] Buchbinder, Niv, Moran Feldman, Joseph Seffi und Roy Schwartz: *A tight linear time (1/2)-approximation for unconstrained submodular maximization.* SIAM Journal on Computing, 44(5):1384–1402, 2015.

[10] Chakrabarti, Amit und Sagar Kale: *Submodular Maximization Meets Streaming: Matchings, Matroids, and More.* arXiv preprint arXiv:1309.2038, 2013.

[11] Chekuri, Chandra, Shalmoli Gupta und Kent Quanrud: *Streaming algorithms for submodular function maximization.* arXiv preprint arXiv:1504.08024, 2015.

[12] Duvenaud, David: *Automatic model construction with Gaussian processes.* Doktorarbeit, University of Cambridge, 2014.

[13] Edmonds, Jack: *Submodular functions, matroids, and certain polyhedra.* Edited by G. Goos, J. Hartmanis, and J. van Leeuwen, 11, 1970.

[14] Feige, Uriel: *A threshold of ln n for approximating set cover.* Journal of the ACM (JACM), 45(4):634–652, 1998.

[15] Feige, Uriel, Vahab S Mirrokni und Jan Vondrak: *Maximizing non-monotone submodular functions.* SIAM Journal on Computing, 40(4):1133–1153, 2011.

[16] Fisher, Marshall L, George L Nemhauser und Laurence A Wolsey: *An analysis of approximations for maximizing submodular set functions—II.* Polyhedral combinatorics, Seiten 73–87, 1978.

[17] Fujishige, Satoru: *Theory of submodular programs: A Fenchel-type min-max theorem and subgradients of submodular functions.* Mathematical programming, 29(2):142–155, 1984.

[18] Fujishige, Satoru: *Submodular functions and optimization*, Band 58. Elsevier, 2005.

[19] Goel, Gagan, Chinmay Karande, Pushkar Tripathi und Lei Wang: *Approximability of combinatorial problems with multi-agent submodular cost functions.* In: *Foundations of Computer Science, 2009. FOCS'09. 50th Annual IEEE Symposium on*, Seiten 755–764. IEEE, 2009.

[20] Goemans, Michel X. und VS Ramakrishnan: *Minimizing submodular functions over families of sets.* Combinatorica, 15(4):499–513, 1995.

[21] Graf, Arnulf BA und Silvio Borer: *Normalization in support vector machines.* In: *Pattern Recognition: 23rd DAGM Symposium, Munich, Germany, September 12-14, 2001. Proceedings*, Seite 277. Springer, 2001.

[22] Grötschel, Martin, László Lovász und Alexander Schrijver: *The ellipsoid method and its consequences in combinatorial optimization.* Combinatorica, 1(2):169–197, 1981.

[23] Herbrich, Ralf, Neil D Lawrence und Matthias Seeger: *Fast sparse Gaussian process methods: The informative vector machine.* In: *Advances in neural information processing systems*, Seiten 625–632, 2003.

[24] Iwata, Satoru, Lisa Fleischer und Satoru Fujishige: *A combinatorial strongly polynomial algorithm for minimizing submodular functions.* Journal of the ACM (JACM), 48(4):761–777, 2001.

[25] Iwata, Satoru und Kiyohito Nagano: *Submodular function minimization under covering constraints.* In: *Foundations of Computer Science, 2009. FOCS'09. 50th Annual IEEE Symposium on*, Seiten 671–680. IEEE, 2009.

[26] Kaufman, Leonard und Peter J Rousseeuw: *Finding groups in data: an introduction to cluster analysis*, Band 344. John Wiley & Sons, 2009.

[27] Kiełbasiński, Andrzej: *A note on rounding-error analysis of Cholesky factorization.* Linear Algebra and its Applications, 88:487–494, 1987.

[28] Krause, Andreas und Daniel Golovin: *Submodular function maximization.* Tractability: Practical Approaches to Hard Problems, 3(19):8, 2012.

[29] Krause, Andreas und Ryan G Gomes: *Budgeted nonparametric learning from data streams.* In: *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, Seiten 391–398, 2010.

[30] Krause, Andreas und Carlos Guestrin: *A note on the budgeted maximization of submodular functions.* 2005.

[31] Krause, Andreas und Carlos E Guestrin: *Near-optimal nonmyopic value of information in graphical models.* arXiv preprint arXiv:1207.1394, 2012.

[32] Krause, Andreas, Ajit Singh und Carlos Guestrin: *Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies.* Journal of Machine Learning Research, 9(Feb):235–284, 2008.

[33] Lee, Jon, Vahab S Mirrokni, Viswanath Nagarajan und Maxim Sviridenko: *Non-monotone submodular maximization under matroid and knapsack constraints.* In: *Proceedings of the forty-first annual ACM symposium on Theory of computing*, Seiten 323–332. ACM, 2009.

[34] Li, Ping, Gennady Samorodnitsk und John Hopcroft: *Sign cauchy projections and chi-square kernel.* In: *Advances in Neural Information Processing Systems*, Seiten 2571–2579, 2013.

[35] Lin, Hui: *Submodularity in natural language processing: algorithms and applications.* University of Washington, 2012.

[36] Lin, Hui und Jeff Bilmes: *A class of submodular functions for document summarization.* In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, Seiten 510–520. Association for Computational Linguistics, 2011.

[37] Lin, Hui und Jeff A Bilmes: *Learning mixtures of submodular shells with application to document summarization.* arXiv preprint arXiv:1210.4871, 2012.

[38] Liu, Yuzong, Kai Wei, Katrin Kirchhoff, Yisong Song und Jeff Bilmes: *Submodular feature selection for high-dimensional acoustic score spaces.* In: *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, Seiten 7184–7188. IEEE, 2013.

[39] Lovász, László: *Submodular functions and convexity.* In: *Mathematical Programming The State of the Art*, Seiten 235–257. Springer, 1983.

[40] Mercer, James: *Functions of positive and negative type, and their connection with the theory of integral equations.* Philosophical transactions of the royal society of London. Series A, containing papers of a mathematical or physical character, 209:415–446, 1909.

[41] Minoux, Michel: *Accelerated greedy algorithms for maximizing submodular set functions.* Optimization techniques, Seiten 234–243, 1978.

[42] Mirzasoleiman, Baharan, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrák und Andreas Krause: *Lazier than lazy greedy.* In: *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[43] Mirzasoleiman, Baharan, Amin Karbasi, Rik Sarkar und Andreas Krause: *Distributed submodular maximization: Identifying representative elements in massive data.* In: *Advances in Neural Information Processing Systems*, Seiten 2049–2057, 2013.

[44] Narasimhan, Mukund, Nebojsa Jojic und Jeff A Bilmes: *Q-clustering.* In: *Advances in Neural Information Processing Systems*, Seiten 979–986, 2006.

[45] Nemhauser, George L, Laurence A Wolsey und Marshall L Fisher: *An analysis of approximations for maximizing submodular set functions—I.* Mathematical Programming, 14(1):265–294, 1978.

[46] Orlin, James B: *A faster strongly polynomial time algorithm for submodular function minimization.* Mathematical Programming, 118(2):237–251, 2009.

[47] Pearson, Karl: *X. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling.* The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 50(302):157–175, 1900.

[48] Queyranne, Maurice: *Minimizing symmetric submodular functions.* Mathematical Programming, 82(1-2):3–12, 1998.

[49] Rasmussen, Carl Edward und Christopher KI Williams: *Gaussian processes for machine learning*, Band 1. MIT press Cambridge, 2006.

[50] Schoenberg, Isaac J: *Metric spaces and positive definite functions.* Transactions of the American Mathematical Society, 44(3):522–536, 1938.

[51] Schölkopf, Bernhard und Alexander J Smola: *Learning with kernels: support vector machines, regularization, optimization, and beyond.* MIT press, 2002.

[52] Seeger, Matthias: *Greedy forward selection in the informative vector machine.* Technischer Bericht, Technical report, University of California at Berkeley, 2004.

[53] Seiler, Mary C und Fritz A Seiler: *Numerical recipes in C: the art of scientific computing.* Risk Analysis, 9(3):415–416, 1989.

[54] Sreekanth, V, Andrea Vedaldi, Andrew Zisserman und C Jawahar: *Generalized RBF feature maps for efficient detection.* 2010.

[55] Tsanas, Athanasios, Max A Little, Patrick E McSharry und Lorraine O Ramig: *Accurate telemonitoring of Parkinson's disease progression by noninvasive speech tests.* IEEE transactions on Biomedical Engineering, 57(4):884–893, 2010.

[56] Tschiatschek, Sebastian, Rishabh K Iyer, Haochen Wei und Jeff A Bilmes: *Learning mixtures of submodular functions for image collection summarization.* In: *Advances in neural information processing systems*, Seiten 1413–1421, 2014.

[57] Van Den Berg, C, JPR Christensen und P Ressel: *Harmonic Analysis on Semigroups: Theory of Positive Definite and Related Functions*, Band 100. Springer Science & Business Media, 2012.

[58] Vedaldi, Andrea und Andrew Zisserman: *Efficient additive kernels via explicit feature maps.* IEEE transactions on pattern analysis and machine intelligence, 34(3):480–492, 2012.

[59] Vitter, Jeffrey S: *Random sampling with a reservoir.* ACM Transactions on Mathematical Software (TOMS), 11(1):37–57, 1985.

[60] Wei, Kai, Rishabh Iyer und Jeff Bilmes: *Submodularity in data subset selection and active learning.* In: *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, Seiten 1954–1963, 2015.