Erzeugung und Anwendung von kontextuellen Wörterbüchern

Tino Hartmann

30. November 2010

Inhaltsverzeichnis

Kapitel 1

Einleitung und Motivation

Eines der interessantesten und faszinierendsten Aspekte der Informatik ist der Begriff *Information*. Diesem Begriff ist inhärent, dass er je nach Kontext zu einem anderen Wert führt: Dieselben Daten können für einen Beobachter *viel*, für einen anderen Beobachter *viel mehr* Information enthalten. Dieser Begriff erstreckt sich notwendigerweise durch alle Gebiete der Informatik und kann damit ständig als gedanklicher Referenzpunkt verwendet werden.

1.1 Wörterbücher als Modelle

Diese Arbeit stellt ein verblüffendes Verfahren vor, Strukturinformation aus einer gegebenen Signalreihe zu extrahieren. Das resultierende Objekt — ein Wörterbuch — kann dann kontextsensitive Entscheidungen treffen.

Damit ist es möglich, bekannte Algorithmen an Beispieldaten anzupassen. Es gibt bereits Arbeiten, die kontextsensitive Bildkompression erlaubt: Ein Algorithmus, der speziell Gesichter sehr gut komprimieren kann. Daraus resultieren andere Anwendungen, wie zum Beispiel die Objektklassifizierung (wenn es sich gut komprimieren lässt, dann muss es wohl ein Gesicht sein).

Der interessante Aspekt von Wörterbüchern ist, dass mit ihnen sozusagen die Beschaffenheit der Daten gelernt werden kann, und das mit nur sehr wenig Einschränkungen der möglichen Daten oder Strukturen.

In den letzten Jahren hat dieses Gebiet zunehmen an Aufmerksamkeit gewonnen. Es wurden sehr große Fortschritte gemacht, die zu Verfahren führten, die es erlauben, solche Datenstrukturen effizient aus realen Daten zu extrahieren.

1.2 Form und Anspruch dieser Arbeit

Diese Arbeit hat den Anspruch eines illustrativen Überblicks. Ein großer Wunsch war dabei, die Mächtigkeit der vorgestellten Konzepte durch Experimente ertastbar zu machen, eine Rückmeldung zu zeigen, wie die Konzepte in der Realität angewendet werden können, das Verhalten in echter oder synthetischer Umgebung erfahren zu können.

Es ist schwierig, in natürlicher Sprache einen absoluten, kontinuierlichen Referenzpunkt aufrecht zu erhalten. Viele der vorgestellten wissenschaftlichen Arbeiten haben kein gemeinsames Vokabular oder benutzen inkonsistent verschiedene Namen für dieselben Konzepte. In dieser Arbeit wurde versucht, sprachliche Eindeutigkeit zu erhalten, auf Kosten der Abwechslung auf sprachlicher Ebene. Besonders in den Kapiteln 2 und 4 werden aus diesem Grund immer wieder Begriffe definiert, die dann konsequent verwendet werden. Kapitel 5 betrachtet den Begriff *Kontext* etwas genauer. Die dort vorgestellten Ideen finden dann in Kapitel 6 Verwendung, indem mögliche Anwendungsgebiete der Konzepte dargestellt werden. Kapitel 7 soll abschließend auf die Perspektive der Information zurückkommen, die Ergebnisse bewerten und die Grenzen der vorgestellten Verfahren ansprechen.

Kapitel 2

Wörterbücher und Repräsentationen

Dieses Kapitel beschreibt wesentliche Begriffe dieser Arbeit und motiviert sie. Es wurde versucht in der gesamten Arbeit eine einheitliche Benennung aufrecht zu erhalten, die zudem in jeder Situation intuitiv bleibt.

2.1 Von der Basis zum Wörterbuch

Der wohl am meisten verwendete Begriff dieser Arbeit soll den Anfang machen: das Wörterbuch.

Definition 1 (Wörterbuch und Wörter) *Ein* Wörterbuch W *ist eine Sammlung von* K *Prototyp-Signalen — genannt Wörter — gedacht zur Repräsentation von Signalen einer bestimmten Domäne. Ein Wörterbuch kann als Menge und als Matrix notiert werden.*

$$W = \{w_i\}_{i=1}^{K}, W \in \mathbb{R}^{n \times K}, K > n \tag{2.1}$$

Die Elemente der Menge — beziehungsweise die Spalten der Matrix — heißen Wörter.

Warum die neuartige Benennung, wenn es sich bei einem Wörterbuch um eine Matrix handelt? Der Grund dieser Namen ist die *Interpretation* der Daten. Ein Wörterbuch ist nicht einfach eine Matrix, auch wenn es als eine solche *repräsentiert* werden kann. *Wörterbuch* beschreibt das gedankliche Konzept und verhält sich zum Datenformat *Matrix* genauso wie *Präsentation* zu *Power Point*.

Um die Definition von *Wörterbuch* zu verstehen, benötigen wir ein Verständnis von *Signal* und *Repräsentation*:

Definition 2 (Signal) *Das Datenformat eines* Signals *ist ein Vektor* $s \in \mathbb{R}^n$. *Der Begriff* Signal *soll betonen, dass mit diesem Vektor auch eine bestimmte kontextabhängige Information verbunden ist.*

Die mit dem Signal verbundene Funktion kann zum Beispiel eine *wahre*, ursprüngliche Funktion $f : \mathbb{R} \to \mathbb{R}$ sein, und der Vektor s ist durch Diskretisierung eines Intervalls dieser Funktion entstanden.

Definition 3 (Repräsentation) *Ein Signal* s *lässt sich durch eine* Repräsentation $r(s) \in \mathbb{R}^K$ *mit dem Wörterbuch* W *durch Linearkombination der Wörter darstellen. Das Ursprungssignal ergibt sich durch Matrix-Vektor Multiplikation von Wörterbuch mit Repräsentation:*

$$s = Wr(s) \tag{2.2}$$

Gleichung 2.2 motiviert die Bezeichnung *Koeffizienten* für die skalaren Elemente von r. Ein Koeffizient in r_i bestimmt den Vorfaktor, mit dem das Wort w_i multipliziert wird, um s darzustellen. Genausogut kann dieser Koeffizient auch als Gewicht benannt werden.

Was bis hier beschrieben wurde ist nichts anderes als eine Basistransformation. Warum der Aufwand? Gleichung 2.2 beschreibt tatsächlich eine Basistransformation, falls W eine quadratische Matrix vollen Ranges ist. Jede Repräsentation, die Koordinaten von s in Basis W, ist dann eindeutig bestimmt:

$$r(s) = W^{-1}s = P_W(s) \tag{2.3}$$

Doch die Forderung, dass W^{-1} existieren muss, schränkt W sehr stark ein. Die Anzahl der Wörter in einem Wörterbuch wird damit durch die Dimension der Wörter diktiert. Falls diese Freiheit nicht aufgegeben werden will, und das will sie nicht, ist typischerweise $K \neq n$, und Gleichung 2.3 ist unnütz. Die Begriffe *Basis* und *Basistransformation* haben damit keinen Sinn.

Dennoch bleibt das Konzept einer Basis sinnvoll: Das Wörterbuch ist der Referenzpunkt der Darstellung von s, das in der Einheitsbasis vorliegt. Genau das ist der Grund warum die Begriffe Wörterbuch und Repräsentation eingeführt werden. Sie verdeutlichen in welchem Kontext vorhandene Daten zu interpretieren sind. *Darstellung von* s *in Basis* W wird zu *Repräsentation von* s *in Wörterbuch* W.

Gleichung 2.2 hat im Fall $K > n$ unendlich viele Repräsentationen für ein gegebenes s. Damit bezeichnet $r(s)$ eine unendliche große Menge. Um dies zu vermeiden gibt es das Konzept der Darstellungsdichte, die die Anzahl der Elemente in $r(s)$ einschränkt. Das Konzept der *dünnsten Repräsentation* im nächsten Abschnitt identifiziert sogar genau eine Repräsentation $r_{\min}(s)$.

In Kapitel 3 werden ein paar Algorithmen vorgestellt, die eine ganz bestimmte Repräsentationen von s berechnen, die dann durch die Nennung des Verfahrens (zum Beispiel *Matching Pursuit*) genau eine Lösung herausnehmen.

Es soll dafür zusätzlich die Idee der *Projektion* beibehalten werden: Repräsentiert man ein Signal in einem Wörterbuch, dann berechnet man seine Repräsentation in W oder *Projektion* auf W. Eine Repräsentation wird deswegen $r = P_W(s)$ genannt. Da $P_W(s)$ nicht eindeutig ist, wird oft ein Algorithmus aus Kapitel 3 zur Berechnung mit angegeben oder dieser ist aus dem Kontext klar.

$$r = P_W^{\mathrm{OMP}}(s) \tag{2.4}$$

2.2 Signalreihen

Da beim Berechnen von Wörterbüchern statt einzelnern Signale immer eine Menge Signale auftreten, soll eine einheitliche und praktische Notation eingeführt werden.

Definition 4 (Signalreihe) *Eine sortierte Menge von Signalen s_i heißt Signalreihe und kann als Matrix S notiert werden. Eine Spalte in S entspricht einem Signal s_i.*

Sollen alle Signale einer Signalreihe S durch einen Algorithmus A repräsentiert werden, wird dies einfach als $P_W^A(S)$ notiert. Das Ergebnis R ist eine Matrix, in der spaltenweise die Ergebnisse $P_W^A(s_i)$ aufgelistet sind.

$$R(S) = P_W^A(S) = P_W^A(s_i)_{s_i \in S} \tag{2.5}$$

Durch die Matrixnotation von Signalreihe S und Repräsentation R, lässt sich analog zu Gleichung 2.9 die Rückprojektion, die Berechnung des Bildes, einfach darstellen:

$$\hat{S} = WR \tag{2.6}$$

$\hat{S}$ heißt analog *Bild der Signalreihe*. Damit ergibt sich der Darstellungsfehler Δ der Signalreihe S für einen gegebene Norm $\|\cdot\|$:

$$\Delta = \|S - \hat{S}\| \tag{2.7}$$

2.3 Fehlerbehaftete Repräsentationen

Im Fall $K < n$ hat Gleichung 2.2 keine Lösung. In diesem Fall muss die harte Forderung von Gleichheit aufgegeben werden, um überhaupt eine Lösung zu bekommen.

Die Algorithmen, die im Laufe dieser Arbeit vorgestellt werden, arbeiten oft mit einem maximalen Fehler ϵ, der für eine Repräsentation akzeptabel ist. Das Signal s wird dabei nicht exakt repräsentiert

$$s \approx Wr^{\epsilon}(s), \|s - Wr^{\epsilon}(s)\| \leq \epsilon \tag{2.8}$$

Definition 5 (Bild eines Signals) *Wird ein Signal mit einer gewissen Fehlertoleranz ϵ dargestellt, entspricht die Rückprojektion $\hat{s} = Wr^{\epsilon}(s)$ nicht genau dem Signal s:*

$$s \approx \hat{s} = Wr^{\epsilon}(s) \tag{2.9}$$

Das rekonstruierte Signal $\hat{s}$ heißt Bild von *s.*

Abschnitt 2.4 wird einen weiteren Grund für den Parameter ϵ nennen.

2.4 Dünne Repräsentationen

Große Wörterbücher $K > n$ ermöglichen sehr viele Repräsentationen für ein gegebenes Signal, weswegen weitere Einschränkungen nützlich sein können. Die *Darstellungsdichte* k sei die Anzahl an Wörtern, die verwendet wird, um ein Signal mit einem Wörterbuch zu repräsentieren. Daraus ergeben sich die Definitionen von dünnen, k-dünnen und dünnsten Repräsentationen.

Definition 6 (dünne Repräsentation) *Eine Repräsentation heißt* dünn*, falls sie wenige Wörter des Wörterbuches benutzt, sprich, viele Koeffizienten der Repräsentation Null sind. Anders ausgedrückt: die Darstellungsdichte k ist relativ klein.*

Definition 7 (k-dünne Repräsentation) *Eine Repräsentation heißt k-dünn, falls sie k Wörter des Wörterbuches benutzt.*

Definition 8 (dünnste Repräsentation) *Eine Repräsentation R heißt* dünnste Repräsentation *eines Signals $s \in \mathbb{R}^n$ bezüglich eines Wörterbuchs W, falls*

$$R = \min_{R(s)} \{\|R(s)\|_0 : WR(s) = s\} =: R^W_{min}(s) \tag{2.10}$$

Es ist eine sehr starke Einschränkung von einer Repräsentation zu fordern, dass fast alle Koeffizienten Null sein sollen. Olshausen und Field haben deswegen eine Gewichtsfunktion eingeführt, die große Koeffizienten bestraft [19].

Definition 9 *Eine Repräsentation heißt* dünn bezüglich einer Gewichtsfunktion G, *falls* $\sum_i G(r_i)$ *klein ist.*

Der Nachteil hierbei ist aber, dass auch Repräsentationen als dünn bezeichnet werden, bei denen alle Koeffizienten klein, aber ungleich null sind. Dies entspricht fast nie dem Anwendungsziel.

Kombiniert man die Idee der dünnsten Repräsentation und die Fehlertoleranz aus Gleichung (2.8), bekommt man die *dünnste Repräsentation unter einem Fehler* ϵ.

Definition 10 *Eine Repräsentation* $R(s)$ *heißt* dünnste Repräsentation unter einem Fehler ϵ, *falls gilt:*

$$R = \min_{R(s)} \{\|R(s)\|_0 : \|s - WR^{\epsilon}(s)\| \leq \epsilon\} =: R^{W,\epsilon}_{min}(s) \tag{2.11}$$

Gesucht wird also nach der Repräsentation, die die wenigsten Wörter des Wörterbuches benutzt, gleichzeitig aber einen Fehler des Bildes zulässt. Dies hat mehrere Vorteile: Erstens ist die Lösung von (2.10) erwiesenermaßen NP-schwer, weswegen effizient nur nach einer genäherten Lösung $R^{W,\epsilon}$ gesucht werden kann [7]. Zweitens ermöglicht diese Definition Lösungen für kleine Wörterbücher mit $K < n$.

2.5 Anwendungen von dünnen Repräsentationen

Es gibt viele nützliche und verblüffende Anwendungsgebiete für dünne Repräsentationen von Signalen.

2.5.1 Funktionsapproximation

Das *Regressionsproblem* ist die Aufgabe eine multivariate Funktion $f : A \rightarrow B$ durch eine endliche Anzahl von Datenpunkten zu approximieren. Diese Datenpunkte sind N Paare:

$$D = \{(a_i, f(a_i)) \in A \times B\}_{i=1}^{N} \tag{2.12}$$

Ziel ist es häufig, für einen gegebenen Approximationsfehler die Paare $(a_i, f(a_i))$ so zu wählen, das N möglichst klein bleibt. Das Wörterbuch ist in diesem mehrdimensionalen Fall eine Menge von Kernelfunktionen. In [21] wird ein

Verfahren beschrieben, das bestimmte Punkte a_i findet, die im Wörterbuch der Kernelfunktionen eine dünne Darstellung haben. Durch diese Punkte kann dann die gesamte Funktion f sehr komprimiert dargestellt werden. Funktionsapproximation ist verwandt mit *Compressed Sensing*.

2.5.2 Compressed Sensing

Donoho hat eine interessante Theorie namens *Compressed Sensing*[9] eingeführt, die dünne Repräsentationen aus einem anderen Blickwinkel betrachten: Wenn es möglich ist, ein Signal $s \in \mathbb{R}^n$ mit wenigen Koeffizienten $k < n$ zu *speichern*, dann muss es doch auch möglich sein, dieses Signal durch wenige Parameter zu *messen*. Wäre im vorraus klar, welche Koeffizienten eine dünne Repräsentation des aktuellen Signals benutzt, dann könnten einfach genau diese gemessen werden. Deswegen ist die intuitive untere Schranke für diese Aufgabe k. Donoho zeigt, dass es tatsächlich möglich ist im Allgemeinen mit $O(k \log n)$ Messungen auszukommen, falls alle zu messenden Signale eine dünne Repräsentation in einem festen Wörterbuch haben.

Dieses Ergebnis hat einige bedeutende Implikationen. Wie oben gezeigt werden im Audio- und Videobereich sehr viele Daten erzeugt, nur um einen Großteil von ihnen zu vernachlässigen, nachdem *der wichtige Anteil* extrahiert wurde. Durch Compressed Sensing ist es möglich, diesen wichtigen Anteil direkt zu messen, was zum Beispiel zu geringeren Messzeiten oder kleineren Samplingraten führen kann.

2.5.3 Klassifikation

Wright et al. versuchen anhand von dünnen Repräsentationen Gesichter zu erkennen [22]. Dabei berechnen sie kein angepasstes Wörterbuch, sondern benuzten alle Trainingsdaten zur Darstellung der zu klassifizierenden Testbilder. Zur dünnen Darstellung eines Testsignals s wird Basis Pursuit verwendet. Die so berechneten Koeffizienten r werden daraufhin auf ihre Klassenzugehörigkeit analysiert. Sei $\delta_i : \mathbb{R}^K \to \mathbb{R}^K$ die charakteristische Funktion, die die Koeffizienten der Klasse i selektiert (Koeffizienten der anderen Klassen werden auf Null gesetzt). Um nun zu beurteilen, welche Klasse am meisten zur Repräsentation von s beiträgt, werden die Fehler e_i berechnet, die entstehen, falls zur Repräsentation nur die Koeffizienten der Klasse i verwendet werden.

$$e_i = s - W\delta_i(r) \tag{2.13}$$

s wird dann der Klasse zugeordnet, die den kleinsten Fehler verursachte, also den größten Beitrag zur Darstellung leistete. Siehe Algorithmus 1 für eine detaillierte Darstellung.

Algorithmus 1 Klassifizierung nach Wright et al. [22]

```
1: procedure class(s, W)
2:     r ← P_W^BP(s)                          ▷ Stelle s in W dar
3:     ∀i : e_i ← |s − Wδ_i(r)|
4:     return min_i{e_i}
5: end procedure
```

Dieses Verfahren wird in [22] als besonders robust gegenüber Störung und Verdeckung befunden. In der Praxis kann dieser Algorithmus einige Fragen aufwerfen.

Wie soll zum Beispiel mit einer sehr großen Menge an Trainingsdaten umgegangen werden? Welche Features müssen für diesen Algorithmus aus den Bildern extrahiert werden ([23])? Kann es hier sinnvoll sein ein angepasstes Wörterbuch zu erstellen? Außerdem ist ein Ansatz denkbar, in dem jede Klasse separat zur Darstellung von s herangezogen wird.

2.5.4 Kompression

Die Formate MP3 und JPEG sind so effektiv, weil Audio- und Videosignale in der Fourierbasis sehr viele kleine Koeffizienten haben. Intuitiv lässt sich das so formulieren: Anstatt einfach Pixel in einem Bild wegzulassen um Speicherplatz zu sparen, wird Information gelöscht, die sich auf jeden Pixel des Bildes verteilt auswirkt. Fällt diese Information weg, dann ist dies nicht offensichtlich.

Die Möglichkeiten von Wörterbüchern und Kompression sind groß. Ist zum Beispiel bekannt, welche Art von Signalen komprimiert werden soll, dann lässt sich der Speicheraufwand für ein Signal senken, da nur die Koordinaten im erwarteten Signalraum gespeichert werden müssen. Diese Information über die Art der Signale kann durch Beispiele oder durch spezifisches Expertenwissen gegeben sein. Bryt und Elad benutzten diese Tatsache um ein Verfahren zu entwickeln, das speziell Bilder von Gesichter komprimiert speichern kann [2]. Das Verfahren erzeugt vielversprechende Ergebnisse, an die bekannte Kompressionsverfahren wie JPEG oder JPEG-2000 nicht herankommen.

Auch in bewegten Bildern lassen sich dünne Repräsentationen effektiv anwenden [18]. Neff et al. zeigen Vorteile gegenüber MPEG [13] besonders bei sehr starken Kompressionsraten (20 kb/s).

2.5.5 Rauschentfernung

Rauschentfernung soll aus einem digitalen *Signal* störendes *Rauschen* entfernen.

$$s[n] = \langle x, W_n \rangle + \eta[n] \tag{2.14}$$

Sei x ein analoges Ursprungssignal und η ein normalverteiltes Rauschen, mit Milttelwert Null und Varianz σ, oft als $N(0, \sigma^2)$ notiert. $\langle x, W \rangle$ bezeichnet die Filterung von x mit der Sensorantwort W_n, die Digitalisierung von x.

$$\langle x, W_n \rangle = \int_{-\infty}^{+\infty} x(t) W_n(t) dt \tag{2.15}$$

Ein wahres Signal x wird also über ein Messgerät digitalisiert, doch die Messung unterliegt einer gewissen Ungenauigkeit, was zur Addition einer Störung η führt. Nun soll nach Observation von s das Ursprungssignal x geschätzt werden. [1] Dies ist nur möglich, wenn Information über die Natur des Signals und des Rauschens vorhanden ist, sonst kann nicht zwischen den beiden unterschieden werden. Falls die Wahrscheinlichkeitsverteilung von Rauschen und Signal bekannt ist, ist es möglich einen Schätzer zu berechnen, der den durchschnittlichen Schätzfehler minimiert [17]. Leider sind aber für reale Signale, wie zum Beispiel Audioaufnahmen oder Fotos, diese Verteilungen nicht bekannt oder viel zu komplex. Es werden stattdessen generelle Annahmen über die Signale getroffen, zum Beispiel dass das resultierende x *lokal glatt* sein sollte oder in einer bestimmten Basis dünn dargestellt werden kann.

Eine andere Perspektive ist der Beispiel-basierte Ansatz. Statt mathematische Eigenschaften der Signale zu raten, werden diese Eigenschaften über Beispiele berechnet. Eine Möglichkeit die Eigenschaften dieser Signale zu repräsentieren ist ein Wörterbuch — und dies rechtfertigt die Einbettung in diese Arbeit. Die Verwendung von Wörterbüchern zur Extraktion von x ist nach Elad und Aharon sehr vielversprechend und vergleichsweise effektiv [11]. In ihrer Arbeit zeigen sie die Möglichkeit auf, das Rauschen eines einzelnen gestörten Bildes zu reduzieren, und das durch ein Wörterbuch, das mit demselben Bild trainiert wurde (!).

Es gibt viele Möglichkeiten, Rauschen anhand von Wörterbüchern zu filtern.

Hard Thresholding

Die Idee des *Hard Thresholding* ist das gegebene Signal s nicht exakt mit einem bestimmten Wörterbuch darzustellen, sondern die Koeffizienten zu

[1] Oft wird auch einfach das *optimale digitale Signal* $\langle x, W \rangle$ geschätzt. Dies ist auch praktikabler, da für reale Signale das wahre x meist gar nicht vorhanden ist.

löschen, die kleiner als der *threshold* T sind.([17], Abschnitt 11.2.2).

Verlustbehaftete Kompression

Eine andere Perspektive, Thresholding zu betrachten, ist die der verlustbehafteten Kompression (zum Beispiel in [3]): Es werden nur T Wörter zur Repräsentation von s benötigt. Typischerweise ist T sehr klein, zumindest aber kleiner als die Dimension von s. Die Darstellung mit T Koeffizienten ist also in gewisser Weise Kompression. Da s meist nicht exakt dargestellt werden kann, ist die Kompression zudem verlustbehaftet.

Hard Thresholding und verlustbehaftete Kompression ist nicht dasselbe. Ersteres löscht nur kleine Koeffizienten, letzteres behält nur die T wichtigsten und löscht alle anderen. Je nach Datensatz kann dies aber auch zu gleichen Ergebnissen führen, da Signale, die eine k-dünne Darstellung besitzen nur k Koeffizienten benötigen, die restlichen sind Null, oder — bei einer kleinen Störung — sehr klein.

Soft Thresholding

Die Praxis hat gezeigt, dass Hard Thresholding durch seine abrupte Auslöschung von Koeffizienten oft Artefakte im rekonstruierten Signal erzeugt. Als Lösung hierfür wurde das Soft Thresholding vorgeschlagen. Hierbei werden für s alle Koeffizienten im Wörterbuch berechnet. Danach werden sie in Richtung Null um den Betrag T *geschrumpft* (mit einer sogenannten *shrinkage function*).

$$t(k) = sgn(k) \cdot \max\{|x| - T, 0\} \tag{2.16}$$

Die grundlegende Idee und sogar ein konkreter Wert für T wurde von Donoho ausführlich aufgezeigt [8].

2.6 Fachtermini

In der englischen referenzierten Literatur gibt es viele Begriffe in diesem Gebiet, die sich nur schlecht oder missverständlich übersetzen lassen, weil sie oft dasselbe bedeuten, nämlich das Auffinden einer (ungefähren) Lösung von Gleichung 2.10 oder 2.11. Diese sind hier für bessere Lesbarkeit kurz aufgezählt:

- *pursuit algorithm*, ein Algorithmus zur Repräsentation von Signalen in einem Wörterbuch

– *atomic decomposition*, der Vorgang, ein Signal zu repräsentieren. Die Betonung liegt hier darauf, dass das Signal in seine Bestandteile *zerlegt* wird

– *sparse representation*, die dünne Darstellung eines Signals

– *sparse coding*, entweder das Forschungsgebiet der dünnen Repräsentation von Signalen oder dasselbe wie *atomic decomposition* mit der Betonung darauf, dass die Repräsentation eines Signals auch als Code gesehen werden kann.

Kapitel 3

Verfahren zur dünnen Repräsentation

Das Verfahren, das eine möglichst dünne Repräsentation von einem Signal bezüglich eines Wörterbuchs findet, wird in der Literatur oft *pursuit* (engl. Verfolgungsjagd) genannt. Diese Benennung betont, dass hier oft iterative Verfahren verwendet werden, die auf *Verfolgungsjagd* mit der Lösung sind. Wir werden diesen Prozess aber weiterhin *Repräsentation* nennen. Im folgenden sind die bekanntesten dieser Verfahren kurz beschrieben.

3.1 Matching Pursuit

Der Matching Pursuit Algorithmus (MP) ist ein *greedy* Verfahren zur Berechnung einer möglichst dünnen Repräsentation. In jedem Iterationsschritt wird ein Wort aus dem Wörterbuch ausgewählt, das für die Darstellung des Signals zusätzlich benutzt werden soll.

Gegeben sei ein Signal $s \in \mathbb{R}^n$ und ein Wörterbuch $W \in \mathbb{R}^{n \times K}$. Gesucht ist eine Repräsentation $r = P_W^\epsilon(s) \in \mathbb{R}^K$, die möglichst wenige Wörter aus W benutzt, sprich, möglichst viele Koeffizienten r_i sollen Null sein so, dass trotzdem gilt:

$$\|s - Wr\| \leq \epsilon \tag{3.1}$$

Idee: In jedem Schritt wird versucht, einen möglichst großen Teil von s durch die Wörter in W darzustellen. Es bleibt ein Restfehler δ übrig, der in den folgenden Schritten weiter verkleinert wird. Dazu wird das Wort gesucht, das das größte Skalarprodukt mit δ hat, denn jenes zeigt damit am meisten in Richtung δ.

Algorithmus 2 Matching Pursuit

1: **procedure** $P_W^{\mathrm{MP}}(s, \epsilon)$
2: $\quad \delta \leftarrow s$
3: $\quad \forall i : r_i \leftarrow 0$ ▷ Initialisierung der Repräsentation ist der Nullvektor
4: $\quad$ **while** $\delta > \epsilon$ **do** ▷ Solange die Repräsentation zu ungenau ist
5: $\quad\quad i \leftarrow \max_i\{\langle \delta, w_i \rangle\}$ ▷ w_i
6: $\quad\quad r_i \leftarrow \langle \delta, w_i \rangle$
7: $\quad\quad \delta \leftarrow s - Wr$ ▷ Restfehler
8: $\quad$ **end while**
9: $\quad$ **return** r
10: **end procedure**

Mallat and Zhang haben gezeigt, dass Matching Pursuit für alle Signale konvergiert, die im Vektorraum liegen, der vom Wörterbuch W aufgespannt wird [16].

Man beachte, dass der Rechenaufwand von MP quadratisch ist, denn das gesamte Wörterbuch muss in jedem Schritt durchsucht werden. Ein schöner Nebeneffekt des greedy-Verfahrens ist, dass nach T Schritten eine Repräsentation mit T Wörtern vorliegt. Statt der While-Schleife kann auch eine For-Schleife verwendet werden, die genau T mal durchlaufen wird. Der Aufruf sei dann $P_W\mathrm{MP}(s, T)$. Dies ist praktisch für die *verlustbehaftete Kompression* (siehe Abschnitt 2.5.5). Durch die For-Schleife wird sicher gestellt, dass exakt T Koeffizienten zur Darstellung von s verwendet werden.

3.2 Orthogonal Matching Pursuit

Orthogonal Matching Pursuit (OMP,[20]) ist eine Variante von MP mit besserer Konvergenz. Es wird in jedem Schritt dafür gesorgt, dass der verbleibende Fehler δ orthogonal auf den schon benutzten Wörtern steht. Das Verfahren ist immernoch greedy, was bedeutet, dass die *Auswahl* der zu benutzenden Wörter nacheinander stattfindet und ein Wort, das einmal nominiert wurde auch für alle späteren Schritte zur Repräsentation verwendet wird. Der Unterschied zu MP ist, dass in jedem Schritt die Koeffizienten für alle Wörter neu optimiert werden und somit die Repräsentation bezüglich der aktuellen Auswahl an Wörtern optimal ist.

3.3 Basis Pursuit

MP und OMP haben wie alle greedy-Verfahren einen sehr großen Nachteil. Sollte sich der Algorithmus in einem sehr frühen Schritt für ein Wort entscheiden, das zwar lokal gesehen den Fehler am besten vermindert, in einer global optimalen Lösung aber gar nicht dabei ist, kann es vorkommen, dass sehr viele Schritte und damit Wörter benötigt werden um diesen anfänglichen Fehler wieder auszugleichen. Die daraus resultierende Lösung kann dann höchstens ein lokales Optimum sein.

Basis Pursuit setzt ein globales Lösungsverfahren an. Zwar ist wie bereits erwähnt Gleichung (2.11) nicht effizient zu lösen, durch die Änderung eines kleinen Details ergibt sich aber ein handhabbares Problem:

$$P_W^{\mathrm{BP}}(s, \epsilon) = \min_{R(s)} \{\|R(s)\|_1 : \|s - WR^{\epsilon}(s)\| \leq \epsilon\} \tag{3.2}$$

Die ursprüngliche Definition einer dünnen Repräsentation nach l_0 Norm verlangt, dass möglichst viele Koeffzienten Null sind. Die l_1 Norm hingegen misst die Summe der Beträge der Koeffizienten. Durch diesen linearen Zusammenhang kann 3.2 auf die Problemstellung der linearen Programmierung [6] abgebildet ([5], Abschnitt 3.1) und damit in nahezu linearer Zeit gelöst werden. Ein globales Minimum in l_1 Norm kann natürlich kein globales Minimum in l_0 garantieren, (2.11) wird also nicht gelöst. Die Methode liefert aber dennoch recht gute Ergebnisse und übertrifft in jedem Fall die Verfahren MP und OMP.

Basis Pursuit ist kein Algorithmus, sondern nur ein Minimierungsschema. Konkrete Algorithmen können aus der Linear Programming Literatur entnommen werden. Chen stellt einige dieser Verfahren vor [4].

Kapitel 4

Optimale Wörterbücher

Die oben genannten Algorithmen zur Repräsentation arbeiten immer mit einem festen Wörterbuch. Um die erwartete Repräsentation für eine gegebene Signalverteilung möglichst dünn zu halten gibt es die Möglichkeit ein geeignetes Wörterbuch zu erstellen. Der einfachste Ansatz ist aus bekannten Basen oder Wörterbüchern (z.B. Cosinusbasis und Haarmatrix) ein geeignetes Wörterbuch zusammenzusetzen. Diese Methode resultiert zwar in wenigen Parameter aber das Ergebnis kann nur für wenige Signalverteilungen gut werden. Es gibt aber auch die Möglichkeit gut geeignete Wörter zu finden für eine gegebene Wörterbuchgröße K. Eine schwierige Aufgabe, denn die Anzahl der Parameter ist dann nK. Für diese Aufgabe wurden schon einige effiziente Verfahren gefunden, die das Problem mit einem akzeptablen Fehlerradius lösen [15].

Für den Vergleich von Wörterbüchern benötigen wir ein Bewertungskriterium. Das Ziel soll sein, für eine gegebene Signalverteilung ein *optimales* Wörterbuch zu finden. Diese Optimalität kann auf mehrere Arten definiert werden:

- Für ein beliebiges Signal s soll der Erwartungswert der benötigten Koeffizienten möglichst klein sein, damit die Repräsentation nicht stärker als ϵ von s abweicht. (Wortoptimalität)

- Für ein beliebiges Signal s und eine Gewichtsfunktion G soll der Erwartungswert von $\sum_i G(r_i(s))$ möglichst klein sein (gewichtete Wortoptimalität).

- Für ein beliebiges Signal s soll der erwartete Darstellungsfehler δ möglichst klein sein, falls s mit maximal $k_{\max}$ Wörtern dargestellt wird. (Fehleroptimalität)

Der Unterschied dieser Sichtweisen ist zwar klein, führt aber letztendlich zu unterschiedlichen Wörterbüchern.

4.1 Welches Buch ist optimal?

Analog zur obigen Aufzählung verschiedener Möglichkeiten, wie Optimalität eines Wörterbuches definiert werden kann, sollen hier die verschiedenen Konzepte formal festgehalten werden.

Zunächst schieben wird die Verantwortung der Definition von Optimalität auf eine Fehler- oder Kostenfunktion E.

Definition 11 (Optimales Wörterbuch) *Ein Wörterbuch ist optimal bezüglich einer Fehlerfunktion E, falls W ein globales Minimum von E ist.*

$$W_{opt} = \min_{W}\{E(W, S)\} \tag{4.1}$$

Im Allgemeinen ist die Fehlerfunktion nicht abhängig von einer konkreten Signalreihe S wie in (4.1), sondern eine Funktion einer Signalverteilung $P(s)$. Ein *wortoptimales* Wörterbuch bezüglich einer Signalverteilung $P(s)$ und einem Fehler ϵ, ergibt sich dann, wenn die erwartete Repräsentation möglichst dünn ist, sprich, möglichst wenig Koeffizienten verwendet.

$$E_w(W, P(s)) = \int_{s \in S} P(s) \| P_W^{\min}(s, \epsilon) \|_0 \tag{4.2}$$

Für gegebene, konkrete Signalreihen ist die Signalverteilung aber im Allgemeinen unbekannt und E ist nur abhängig von W und S. Man muss sich hier mit dem Erwartungswert der Testreihe zufriedengeben.

Definition 12 *Ein Wörterbuch ist* wortoptimal *bezüglich einer Testreihe S und einem maximalen Fehler pro Signal ϵ, falls die erwartete Anzahl an benötigten Koeffizienten am kleinsten ist. Die zu minimierende Fehlerfunktion ist:*

$$E_w(W, S, \epsilon) = \sum_{s \in S} \| P_W^{min}(s, \epsilon) \|_0 \tag{4.3}$$

Zu beachten ist, dass E in der Praxis noch davon abhängt, welcher Repräsentationsalgorithmus P_W benutzt wird.

In der aktuellen Literatur wird oft ein *fehleroptimales* Wörterbuch gesucht. Dies sei hier diskret definiert, der kontinuierliche Fall ergibt sich analog und wird hier nicht gebraucht.

Definition 13 *Ein Wörterbuch ist* fehleroptimal *bezüglich einer Testreihe $S = s_i$ und einer maximalen Wortanzahl zur Darstellung $k_{\max}$, falls das Bild der Testreihe $\hat{S}$ möglichst nahe an S liegt, der Darstellungsfehler also möglichst gering ist. Die Fehlerfunktion ist:*

$$E_f(W, S, k_{\max}) = \|S - W P_W^{\min}(S, k_{\max})\|_F^2 = \|S - \hat{S}\|_F^2 \tag{4.4}$$

Dabei ist $\|\cdot\|_F$ die Frobenius Norm, die analog zur Euklischen Länge eines Vektors, die Größe einer Matrix berechnet.

$$\|S\|_F = \sqrt{\sum_{ij} S_{ij}} \tag{4.5}$$

4.2 Allgemeine Lösungsstrategie

Wie bereits erwähnt, definieren aktuelle Veröffentlichungen die Optimalität eines Wörterbuches mit Gleichung (4.4). In dieser Gleichung verstecken sich *zwei* Minimierungsaufgaben. Das Finden eines optimalen Wörterbuchs W_{opt} und die Berechnung einer minimalen Repräsentation $R = P_W^{\min}(S, k_{\max})$. Oft wird hier ein iterativer Ansatz gewählt, der zwei Phasen immer wieder ausführt und das Wörterbuch modifiziert bis ein Konvergenzkriterium $K(W, R)$ erfüllt ist. Typischerweise ist dieses Konvergenzkriterium das Erreichen einer bestimmten Fehlergrenze ϵ, die das Bild der Signalreihe von S abweichen darf.

$$K(W, R) = \begin{cases} True & \text{falls } |S - WR| \leq \epsilon \\ False & \text{sonst} \end{cases} \tag{4.6}$$

In der ersten Phase wird das Wörterbuch als fest angesehen und eine dünne Repräsentation R für die gesamte Signalreihe S berechnet. Hierbei kann oft ein beliebiger Repräsentationsalgorithmus, wie MP oder BP, benutzt werden. In der zweiten Phase wird die Repräsentation dazu benutzt, durch Anpassung des Wörterbuches einen kleineren Darstellungsfehler zu erzielen.

Für die erste Phase sei noch folgende Eigenschaft von (4.4) erwähnt: Es lässt sich der Darstellungsfehler der gesamten Signalreihe durch eine Summe der Darstellungsfehler aller Signale aufteilen.

$$E_f = \|S - WR(S)\|_F^2 = \sum_{i=1}^{n} |s_i - Wr(s_i)|_2^2 \tag{4.7}$$

Dadurch lässt sich jede Repräsentation eines Signals separat berechnen.

Alle vorgestellten Algorithmen werden W_{opt} nicht finden. Dies liegt nicht zuletzt an der NP-Schwere des Problems der minimalen Repräsentation. Doch selbst wenn minimale Repräsentationen effizient berechnet werden könnten ist W_{opt} schwer zu finden. Die Algorithmen dieser Arbeit beschränken sich auf einen iterativen Ansatz mit lokaler Aktualisierung des Wörterbuches, weswegen sie leicht in lokalen Minima hängen bleiben können. Ein Ausweg ist hier — wie immer — von mehreren Initialisierungen zu starten und das beste Wörterbuch als Lösung herauszusuchen. Eine Garantie W_{opt} zu finden kann aber nicht gegeben werden.

Die beiden beschriebenen Phasen der allgemeinen Lösungsstrategie haben große Ähnlichkeit mit dem Clusterverfahren K-Means. Da der wichtigste Algorithmus dieser Arbeit — KSVD — eine strenge Verallgemeinerung von K-Means darstellt, wird für besseres Verständnis im folgenden Abschnitt eine kleine Einführung der Clusteranalyse bereitgestellt.

4.3 Clusteranalyse

Ein Clusteralgorithmus (auch Clusteranalyse genannt) ist ein Verfahren zur Entdeckung von Strukturen, die in gegebenen Daten $S = \{s_i\}$ enthalten sind. Ein *Cluster* bildet eine Gruppe von ähnlichen Elementen. Die Ähnlichkeit zwischen zwei Elementen s_1, s_2 ist dabei von Anwendung zu Anwendung unterschiedlich über ein Abstandsmaß $abs(s_1, s_2)$ definiert. Typischerweise ist dies die l_2-Norm. Die Einteilung eines Signals in ein bestimmtes Cluster kann überlappend (weiches Clustering) oder nichtüberlappend (hartes Clustering) sein.

4.3.1 Clusteranalyse und dünne Repräsentationen

Es gibt eines sehr engen Zusammenhang zwischen Clusteranalyse und der Berechnung von dünnen Repräsentationen: Jedes Ergebnis einer Clusteranalyse kann als eine sehr dünne Darstellung interpretiert werden. Sei das Wörterbuch die Menge der Prototypsignale $W = \{w_i\}_{i=1}^{K}$, dann ist die Repräsentation eines Signals s, das zu Cluster j gehört:

$$r_i = \begin{cases} 1 & \text{i = j} \\ 0 & \text{sonst} \end{cases} \tag{4.8}$$

Die Darstellung durch ein Clusterverfahren ist also 1-dünn.

4.3.2 K-Means

Im K-Means Algorithmus werden für eine gegebene Signalmenge S k Prototypsignale p_i gesucht, von denen jedes ein *Cluster* erzeugt. Für ein hartes Clustering ergibt sich das Cluster c für ein Signal s durch den Abstand von den Prototypsignalen p_i:

$$c = \min_i \{abs(s, p_i)\} \tag{4.9}$$

Alle Signale im Cluster i, also das Cluster C_i, ist dann nach (4.9):

$$C_i = \{s_j : i = min_k\{abs(s_j, p_k)\}\} \tag{4.10}$$

Werden die Cluster als eine Repräsentationsmatrix $R(S) = (r_i)_{i=1}^n$ (r_i nach Gleichung (4.8)) notiert, dann ist das Erlernen der Clusterzugehörigkeit nichts anderes als die Berechnung einer dünnen Repräsentation (Phase 1). Ist die Zuordnung zu Clustern für jedes Signal bestimmt, berechnet K-Means neue Prototypsignale. Der neue Prototyp für ein Cluster ist einfach der Schwerpunkt aller zugehörigen Signale.

$$p_i = \frac{1}{|C_i|} \sum_{s \in C_i} s \tag{4.11}$$

Die Berechnung neuer Prototypen ist nichts anderes als die 2. Phase. Die Parallelität zwischen K-Means und einem Algorithmus zum Erlernen eines Wörterbuchs ist in Algorithmus 3 noch einmal genau dargestellt.

Algorithmus 3 K-Means aus der Perspektive eines Wörterbuches

procedure $W_{KM}(S)$
  $W \leftarrow (p_i \in \mathbb{R}^n)_{i=1}^k$ ▷ Wähle k zufällige Clusterzentren
  while not K **do** ▷ Solange Konvergenzkriterium nicht erfüllt
    ▷ Berechne dünne Repräsentationen (Clusterzugehörigkeit)
    $\forall i : C_i \leftarrow \{s_j : j = min_k\{abs(s_j, p_k)\}\}$
    ▷ Berechne neues Wörterbuch
    $\forall i : p_i \leftarrow \frac{1}{|C_i|} \sum_{s \in C_i} s$
  end while
  return W
end procedure

Der Ansatz aus Gleichung 2.2 lässt aber mehr Freiheiten, ist nicht so eingeschränkt wie eine Darstellung durch Cluster, die es nur erlaubt, einen der k Koeffizienten auf 1 und den Rest auf 0 zu setzen. Aus diesem Grund wird der Cluster-Ansatz durch K-SVD erweitert.

4.3.3 Verallgemeinerung von K-Means

Die beiden Phasen des K-Means können sehr einfach abstrahiert werden. Im ursprünglichen Algorithmus wird ein Cluster C_i durch ein Prototypsignal p_i definiert. Ersetzen wir nun das Prototypsignal mit einem *Prototypobjekt*.

Es gibt zwei Einschränkungen an die Klasse von Prototypobjekten.

- (Phase 1) Um ein Signal einem Cluster zuordnen zu können ist es notwendig, dass eine Distanz zwischen Prototypobjekt und Signal definiert werden kann. Dies entspricht im ursprünglichen K-Means einem gewöhnlichen Punkt-Punkt Distanzmaß. Zum Beispiel könnte das Objekt eine Gerade sein. Der Abstand zwischen Objekt und Signal ist dann definiert durch ein Punkt-Geraden Distanzmaß.

- (Phase 2) Um für eine gegebene Signalmenge eines Clusters ein neues Prototypobjekt zu bestimmen, muss es möglich sein, ein Objekt zu finden, dass die Summe aller Distanzen zu den Signalen minimiert. Im ursprünglichen K-Means Algorithmus ist dies der Schwerpunkt. Falls das Prototypobjekt eine Gerade ist, dann ist für eine gegebene Menge an Signalen die Prototypgerade die Regressionsgerade durch diese Punkte.

Im KSVD Algorithmus ist das Prototypobjekt ein Vektor und entspricht einem Wort im Wörterbuch.

4.4 Gradientenabstieg auf der Fehlerfunktion

Eine einfache Methode, ein gutes Wörterbuch für einen gegebene Testreihe S zu finden, ist ein Gradientenabstieg auf der Fehlerfunktion E_f (siehe Gleichung (4.7)). Dafür berechnet man die Ableitung von E_f bezüglich W:

$$\frac{\partial E_f}{\partial W} = 2\sum_{i=1}^{N}(s_i - Wr_i)r_i^T = 2(S - WR)R^T \tag{4.12}$$

Diese Gleichung zeigt, wie sich der Darstellungsfehler bezüglich W verändert (für feste Repräsentationen r_i). Da ein Minimum von E_f gesucht wird, wird W in jedem Schritt in die entgegengesetzte Richtung um eine Lernrate η verändert:

$$W_{n+1} = W_n - \eta\frac{\partial E_f}{\partial W} = W_n - \eta'\sum_{i=1}^{N}(s_i - W_n r_i)r_i^T \tag{4.13}$$

Dieses Verfahren verwenden Olshausen und Field in ihrer Arbeit [19] und wird Zukunft mit *naiv* referenziert. Nachteil dieses Verfahrens ist, dass durch η' ein zusätzlicher Parameter eingeführt wird, für den ein guter Wert gefunden werden muss. Es gibt aber Verfahren, die η' in jedem Schritt dynamisch anpassen.

4.5 Method of Optimal Directions

Engan et al. stellen eine andere interessante Variante vor [12]. Für jedes Wort im Wörterbuch wird versucht die *optimale Richtung* zu finden, die den Darstellungfehler minimiert — daher der Name. Statt wie in [19] Schrittweise in Richtung Minimum der Fehlerfunktion E_f zu laufen, wird diesen Minimum direkt berechnet. Dafür wird die Ableitung (Gleichung (4.12)) auf Null gesetzt:

$$\begin{aligned}
(S - WR)R^T &= 0 \\
SR^T - WRR^T &= 0 \\
WRR^T &= SR^T \\
W &= SR^T \cdot (RR^T)^{-1} \\
W_{n+1} &= SR_n^T \cdot (R_n R_n^T)^{-1}
\end{aligned}$$

4.6 K-SVD

K-SVD ist ein Algorithmus zum Erlernen eines geeigneten Wörterbuches für gegebene Datenbeispiele. Er ist eine Verallgemeinerung des Clustering-Verfahrens K-Means.

4.6.1 Singulärwertzerlegung

Da K-SVD im wichtigsten Schritt, nämlich dem Lernen eines Wörterbucheintrags, eine Singulärwertzerlegung vornimmt, soll dieses Konzept hier kurz erläutert und plausibilisiert, aber nicht bewiesen werden. Ein Beweis verwendeter Eigenschaften kann zum Beispiel in [7] nachgelesen werden.

Eine reelle Singulärwertzerlegung (engl. Singular Value Decomposition, SVD) stellt eine reelle Matrix A als Produkt von drei besonderen Matrizen dar.

$$A_{n\times m} = U_{n\times m}\Delta_{m\times m}V^T_{m\times m} \tag{4.14}$$

Diese drei Matrizen geben Auskunft über bestimmte Eigenschaften von A. Δ ist eine Diagonalmatrix, aufgebaut aus den *Singulärwerten* s_i von A.

$$\Delta_{ij} = \begin{cases} s_i & i = j \\ 0 & \text{sonst} \end{cases} \tag{4.15}$$

Konvention ist, dass die Singulärwerte sortiert sind ($s_1 \geq s_2 \geq \ldots$). Eine solche Zerlegung kann nun dazu benutzt werden, eine Matrix A von Rang n durch eine Matrix von Rang $r, r < n$ zu approximieren.

$$\hat{A}_{n \times m} = \hat{U}_{n \times r} \hat{\Delta}_{r \times r} \hat{V}^T_{r \times m} \tag{4.16}$$

Dafür werden aus Δ nur die r größten Singulärwerte benutzt. Dies entspricht bei sortierten Singulärwerten der linken oberen Ecke der Größe $r \times r$. $\hat{U}$ und $\hat{V}^T$ ergeben sich entsprechend aus den ersten r Spalten von U beziehungsweise V^T. Eine solche Approximation entspricht in der Sprache der Hauptkomponentenanalyse (PCA) einer Projektion auf die ersten r Hauptkomponenten. K-SVD benutzt diese Eigenschaft, um eine Fehlermatrix mit zwei Vektoren darzustellen. Dies enspricht einer Approximation 1. Ranges (siehe Algorithmus 4).

Algorithmus 4 SVD: Approximation 1. Ranges

```
1: procedure rank1(A)
2:     UΔV^T ← SVD(A)
3:     a ← U_1                                  ▷ Erste Spalte von U
4:     b ← Δ_{1,1} V_1        ▷ Größter Singulärwert und erste Spalte von V^T
5:     return a, b
6: end procedure
```

4.6.2 Der K-SVD Algorithmus

Wie im allgemeinen Ansatz beschrieben, arbeitet auch K-SVD mit zwei Phasen. Für die erste Phase kann jeder beliebige Algorithmus P^A_W zur Repräsentation verwendet werden. Die eigentliche Arbeit liegt in der Art und Weise, wie das Wörterbuch aktualisiert wird. Hierfür betrachten wir die Matrixmultiplikation WR zur Erstellung des Signalbildes:

$$\hat{S} = WR \tag{4.17}$$

Sei c_i Reihe i des Wörterbuches. c_i enthält alle Koeffizienten, die das Wort w_i betreffen. Gleichung (4.17) lässt sich umschreiben zu:

$$\hat{S} = \sum_{i=1}^{K} w_i c_i \tag{4.18}$$

Beachte, dass w_i ein Spaltenvektor, c_i ein Zeilenvektor ist und deshalb deren Produkt eine Matrix ergibt. Diese Matrix ist genau der Anteil in $\hat{S}$, der durch das Wort w_i erzeugt wird. Dies motiviert die Aufteilung der Fehlerfunktion (4.4) in zwei Teile: Ein Fehler, der durch ein bestimmtes Wort w_k erzeugt wird, und den restlichen Fehler E_k, der bliebe, wenn das Wort w_k nicht zur Darstellung benutzt werden würde.

$$\begin{aligned} S - WR = & \quad S - \textstyle\sum_{i=1}^{K} w_i c_i & (4.19)\\ = & \quad (S - \textstyle\sum_{i \neq k} w_i c_i) - w_k c_k & (4.20)\\ = & \quad E_k - w_k c_k & (4.21) \end{aligned}$$

Obere Gleichung sagt, dass der Darstellungsfehler kleiner ist, desto ähnlicher sich E_k und $w_k c_k$ sind. Dies benutzen Aharon et al. dafür, um nacheinander für jedes Wort w_k ein neues Wort w'_k und zugehörige Koeffizienten c'_k zu finden so, dass

$$w'_k c'_k \approx E_k \tag{4.22}$$

Hier wird offensichtlich, warum die Singulärwertzerlegung eingeführt wurde: E_k ist im Allgemeinen eine Matrix von Rang $n-1$ und $w'_k c'_k$ hat nur Rang 1, weswegen E_k nur approximiert werden kann. Dies ist auch plausibel. Könnte (4.22) mit Gleichheit gelöst werden, wäre der Darstellungsfehler durch eine Iteration bei Null und das Wörterbuch perfekt.

Um zu gewährleisten, dass nur mit dünnen Repräsentationen gearbeitet wird, darf das neue Wort w'_k aber nur von den Signalen verwendet werden, die auch vorher schon w_k verwendet haben.

$$c_k[i] = 0 \implies c'_k[i] = 0 \tag{4.23}$$

Um dies zu garantieren, werden vor der SVD die Spalten von E_k herausgestrichen, für die $c_k = 0$ ist. Für bessere Lesbarkeit sei diese Projektion einfach durch $P(E_k, c_k) =: \hat{E}_k$ gekennzeichnet (Aharon et al. notieren diese Operation als Matrixmultiplikation). Die Rückprojektion $P^{-1}(x, c_k)$ sei so definiert, dass in x entsprechende Null-Spalten wieder eingefügt werden. Durch Singulärwertzerlegung können zwei Vektoren a, b gefunden werden, die die gekürzte Matrix $\hat{E}_k$ gut approximieren. Das neue Wort w'_k und die zugehörigen Koeffizienten c'_k ergeben sich dann durch Rückprojektion von a, b.

Algorithmus 5 K-SVD: Lernen eines Wörterbuches

1: **procedure** $W_{KSVD}(S, K, \epsilon)$
2: $W^{(0)} \in \mathbb{R}^{n \times K}$ ▷ Initialisiere Wörterbuch
3: $\Delta \leftarrow \infty$
4: **while** $\Delta > \epsilon$ **do** ▷ Solange Fehler zu groß und Konvergenz nicht erreicht
5: ▷ Phase 1: Berechne Repräsentationen
6: $R \leftarrow P_W^A(S)$
7: ▷ Phase 2: Aktualisiere Wörterbuch
8: **for all** $w_k \in W$ **do**
9: $E_k \leftarrow (S - \sum_{i \neq k} w_i c_i)$
10: $\hat{E}_k \leftarrow P(E_k, c_k)$
11: $a, b \leftarrow \text{rank1}(\hat{E}_k)$
12: $w_k \leftarrow P^{-1}(a)$ ▷ Zuweisung meint Aktualisierung in W
13: $c_k \leftarrow P^{-1}(b)$ ▷ Zuweisung meint Aktualisierung in R
14: **end for**
15: $\Delta \leftarrow \|S - WR\|_F^2$ ▷ Berechne neuen Fehler
16: **end while**
17: **return** W
18: **end procedure**

4.6.3 Konvergenz von KSVD

KSVD ist neben seiner Ähnlichkeit zu K-Means insofern interessant, dass er ein sehr gutes Konvergenzverhalten zeigt. Konvergenz kann jedoch nicht garantiert werden. Wie alle hier vorgestellten Algorithmen zur Wörterbuchberechnung ist auch KSVD abhängig vom Repräsentationsalgorithmus. Falls P_W^A aber zu jeder Zeit die minimale Repräsentation zurückliefern würde, konvergiert KSVD. Dies liegt daran, dass in jedem Schritt die Singulärwertzerlegung (Zeile 11) den Darstellungsfehler verringert oder konstant lässt (in dieser Arbeit nicht bewiesen). Eine minimale Repräsentation im nächsten Schritt hat dann höchstens denselben Darstellungsfehler, sonst wäre sie nicht minimal. Der Konvergenzbeweis von KSVD ist damit analog zum Konvergenzbeweis von K-Means.

Kapitel 5

Kontext

Oft wird der Begriff *Kontext* sehr vage benutzt. In diesem Kapitel soll untersucht werden, was sich hinter diesem Begriff verbergen kann. Es werden zudem ein paar Fragen gestellt, die leider nicht alle im Rahmen dieser Arbeit beantwortet werden können, aber die Möglichkeiten dieses Forschungsgebietes aufzeigen sollen. Wie groß sind die Möglichkeiten für Wörterbücher, die speziell auf einen bestimmten Kontext trainiert wurden?

5.1 Signalmodelle

5.1.1 Signale als Überlagerung von Aspekten

Selten sind interessante oder wichtige Informationen in Daten offensichtlich. Meist ist in einer Untersuchung das gegebene Datensignal eine additive Überlagerung von vielen Anteilen, die für unterschiedliche Aspekte interessant sind oder je nach Kontext verschiedene Wichtigkeit erlangen.

Es sei zum Beispiel ein Audiosignal gegeben, dass durch einfache Mikrofonaufnahme entstanden ist und aus einer Überlagerung von Herzgeräuschen, Atemgeräuschen und Rauschen besteht. Für eine Klassifizierung von Herzkrankheiten können die Herzgeräusche, für Krankheiten, die die Lunge betreffen, können die Atemgeräusche relevant sein. In beiden Fällen ist das Rauschen interessant, einfach um es entfernen zu können.

In solchen Fällen wird im Allgemeinen ein *additives Signalmodell* erstellt.

$$s = \sum_{i=1}^{k} s_i \tag{5.1}$$

In diesem Modell entsteht ein Signal s aus einer additiven Überlagerung von

k Signalkomponenten s_i. Für jede Komponente ist die Wahrscheinlichkeitsverteilung $P(s_i)$ entweder bekannt oder sie wird geschätzt.

Mithilfe eines solchen Signalmodells lässt sich eine *wahrscheinlichste Erklärung* für ein gegebenes Eingabesignal berechnen. Diese *Erklärung* ist eine Zerlegung des Eingabesignals in seine additiven Bestandteile.

Im Allgemeinen sind interessante Parameter oder Aspekte eines Signals nicht additiv, sondern haben komplexe, vielleicht sogar Turing-vollständige Wechselwirkungen miteinander. Möchte man zum Beispiel aus mehreren Bildern eines Gebäudes ein dreidimensionales Modell M extrahieren, dann ist Gleichung (5.1) nicht mehr ausreichend. Vielmehr sucht man hier nach einem komplexen Modell M, mit dem eine generierende Funktion, in der Bildverarbeitung ein *Renderer* R, die Eingabebilder möglichst präzise rekonstruiert. In diesem Fall ist das Modell M die *Erklärung* der Eingabebilder und enthält Parameter wie Farbe und das Reflexionsverhalten von Wänden oder Position und Ausrichtung der Kamera. Der Renderer beschreibt in gewisser Weise indirekt die Wahrscheinlichkeitsverteilung der Bilder. Es ergibt sich also nur ein sehr Signalmodell:

$$s = R(M) \tag{5.2}$$

5.1.2 Definition von Kontext

Der Kontext, in dem ein Signal betrachtet wird, soll nun indirekt die wahrscheinlichste Erklärung definieren. Für den Kontext *Herzgeräusche* ergibt sich eine andere Berechnung als für *Atemgeräusche*, um bei dem Beispiel von Abschnitt 5.1.1 zu bleiben. Die Beschreibung eines Kontextes kann aber auch sehr komplex werden, was das Beispiel der Extraktion eines dreidimensionalen Modells gezeigt hat. Das Ziel oder der Wunsch des Wörterbuchansatzes soll sein, diesen Kontext durch eine Anzahl an Signalbeispielen indirekt zu beschreiben. Das daraus berechnete Wörterbuch soll ermöglichen, aus neuen Signalen die wahrscheinlichste Erklärung bezüglich des indirekt definierten Kontextes zu berechnen. Der Abschnitt *Experimente* dieser Arbeit soll untersuchen, inwieweit diese Aufgabe mit den Methoden des Wörterbuchansatzes umzusetzen ist.

5.1.3 Orthogonale Überlagerung

Ein sehr einfacher Fall der Überlagerung ist die orthogonale Überlagerung. Betrachten wird ein Signal das aus zwei Signalen aus verschiedenen Kontexten überlagert wurde. Das eine Signal sei das *Nutzsignal* x, das andere ein störendes *Rauschen* η.

$$s = x + \eta \tag{5.3}$$

Sei das Nutzsignal $x \in \mathbb{R}^n$ entstanden aus einer linearen Kombination aus $K < n$ linear unabhängigen Vektoren $w \in \mathbb{R}^n$.

$$x = \sum_{i=0}^{K-1} c_i w_i \tag{5.4}$$

Das Nutzsignal stammt also aus einem Untervektorraum von $\mathbb{R}^n$, notiert als W. Die Berechnung der Koeffizienten x nennt sich Projektion auf W und ist durch die Multiplikation mit der Inversen zu berechnen:

$$P_W(s) = W^{-1}s \tag{5.5}$$

Sein nun das Rauschen orthogonal zu diesem Untervektorraum, dann ist die Projektion von η auf W Null.

$$P_W(\eta) = 0 \tag{5.6}$$

Anders ausgedrückt: die Darstellung von η in W ist nicht möglich und damit Null, oder: η steht orthogonal auf W und damit auf allen x. Das Nutzsignal x kann dann durch Projektion auf W einfach extrahiert werden:

$$\begin{aligned} x &= P_W(s) \\ &= P_W(Wx + \eta) \\ &= W^{-1}(Wx + \eta) \\ &= W^{-1}Wx + W^{-1}\eta \\ &= x + 0 \end{aligned}$$

5.2 Kontextzugehörigkeit

Für die Klassifikation von Signalen ist es notwendig, die Zugehörigkeit eines Signales zu einem *Kontext* zu bestimmen. Sei $\{W_k\}_{k=1}^K$ eine Menge von Wörterbüchern, von denen jedes einen Kontext beschreiben soll. Eine Menge an Signalen $\{s_i \in \mathbb{R}^n\}_{i=1}^N$ soll nun *klassifiziert* werden. Es wird eine Funktion gesucht, die jedem s_i eine Klasse k zuordnet.

Sei nun eine Funktion $d(s, W)$ gegeben, die die Wahrscheinlichkeit misst, dass s aus dem Kontext W entstanden ist. Die Klasse k für jedes Signal ist dann einfach gegeben als:

$$k(s_i) = \max_k \{d(s_i, W_k)\} \tag{5.7}$$

Eine Möglichkeit $d(s, W)$ zu definieren wird dadurch motiviert, dass s in seinem ursprünglichen Wörterbuch W eine k-dünne Repräsentation hat.

Man kann nun versuchen zu schätzen, wie gut sich s in W darstellen lässt. Zum Beispiel lässt sich der Darstellungsfehler berechnen, falls s nur mit T Komponenten dargestellt wird.

$$d(s, W) = P_W^A(s, T) \tag{5.8}$$

In Kapitel 6 wird diese Definition von d untersucht. Eine weitere Möglichkeit ist, einen Repräsentationsfehler ϵ zu erlauben und dann zu berechnen, wieviele Wörter k' benötigt werden, um s in W darzustellen. Falls s eine k-dünne Repräsentation hat, dann ist je nach Repräsentationsalgorithmus k' klein, falls s aber keine dünne Repräsentation hat, dann ist k' sehr wahrscheinlich groß.

Eine andere Perspektive der Klassifikation wird in [22] verwendet. Statt die Kontextzugehörigkeit für jedes Signal getrennt zu berechnen, wird ein Signal mit dem kombinierten, zusammengefügten Wörterbuch $[W_1, W_2, \ldots]$ dargestellt. Danach wird analysiert, welches Wörterbuch am meisten Wörter zur Darstellung beigetragen hat. Dieser Klasse wird s dann zugeordnet. Falls s eine k-dünne Repräsentation in W_k hat, dann ist die Wahrscheinlichkeit hoch, dass die zugehörigen Wörter aus W_k auch zu Repräsentation verwendet werden. Es besteht jedoch die Möglichkeit, dass eine bessere Repräsentation im kombinierten Wörterbuch existiert, denn durch seine Größe gibt es mehr Freiheiten zur Darstellung. Die resultierende Darstellung muss nicht unbedingt etwas mit der ursprünglichen Darstellung zu tun haben und kann sich essentiell von ihr unterscheiden. Der Erfolg dieses Verfahrens hängt stark von der Beschaffenheit der Daten und der Ähnlichkeit der Wörterbücher ab.

5.3 Kontextanteile

Gegeben seien Signale, die aus einer additiven Überlagerung mehrerer Anteile entstanden sind (siehe (5.1)). Jeder Anteil s_i entstammt einem ursprünglichen Kontext, dem je ein Wörterbuch W_i zugeordnet ist.

$$s_i = W_i x_i, x_i \text{ ist } k\text{-dünn} \tag{5.9}$$

Im Gegensatz zum Filtern eines normalverteilten Rauschens soll nun versucht werden, aus s nacheinander seine Anteile $W_i x_i$ zu extrahieren. Ist dies mit einem einfachen Wörterbuchansatz möglich?

Ein einfacher Ansatz entspringt dem Gedanken, dass Anteil s_i des Signals nach Gleichung (5.9) eine dünne Repräsentation in W_i hat. Kann eine dünne Repräsentation von s in W_i die Koeffizienten x_i finden?

Die Antwort ist einfach: im Allgemeinen nicht! Im Allgemeinen sind die Wörterbücher, aus denen s aufgebaut ist, nämlich nicht paarweise orthogonal

zueinander. Betrachten wir den Spezialfall, dass s nur aus zwei Anteilen zusammengesetzt ist und die entsprechenden Wörterbücher identisch sind.

$$s = W_1 x_1 + W_2 x_2 = W_1(x_1 + x_2)$$

Die dünnste Repräsentation von s in $W_1 (= W_2)$ hat damit die Koeffizienten $x = x_1 + x_2$. Mit diesem Verfahren x_1 zu finden ist damit unmöglich.

Für die Extraktion einer Schätzung von x_1 und x_2 muss vermutlich ein Verfahren verwendet werden, dass die Information aller Wörterbücher W_i gleichzeitig benutzt und damit eine kombinierte wahrscheinlichste Erklärung berechnen kann. Zudem kann Information über die Koeffizientenverteilung $P(x_i)$ zusätzlich Aufschluss über den Ursprung von s geben.

Kapitel 6

Experimente

In diesem Kapitel sollen die vorgestellten Algorithmen an konkreten Datensätzen untersucht werden.

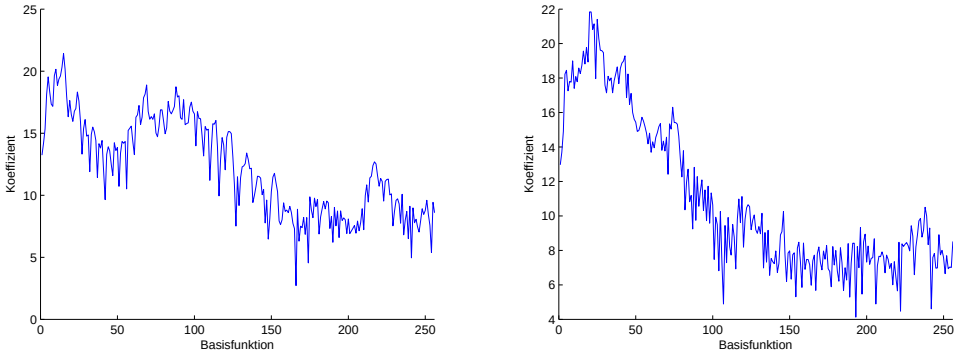
6.1 Datensätze

6.1.1 Allgemeines zu Signalreihen

Vor jeder Berechnung wird der gesamte Datensatz *zentriert* und *skaliert*. Ersteres ist notwendig, weil zum Beispiel KSVD eine Hauptkomponentenanalyse vollzieht, um ein neues Wort des Wörterbuchs zu berechnen. Falls die Daten nicht zentriert sind, zeigen die Hauptkomponenten tendentiell zum Schwerpunkt der Daten. Eine Analyse der tatsächlichen Struktur der Daten erfolgt nicht. Eine Zentrierung ist zudem keine dramatische Veränderung der Information, die in den Daten enthalten ist. Ein Repräsentationsalgorithmus kann nach Berechnung des Bildes durch einfache Addition des Schwerpunktes die Zentrierung der Daten rückgängig machen. Die *Skalierung* der Daten ist eine Frage der Aussagekraft der folgenden Diagramme. Die Signale wurden so skaliert, dass die durchschnittliche Länge 1 ist. Auch dies ist kein Eingriff in die Information der Daten und kann durch Multiplikation des Skalierungsfaktors rückgängig gemacht werden. Der Wert der mittleren quadratischen Abweichung (MSE) ist dann aber besser zu deuten.

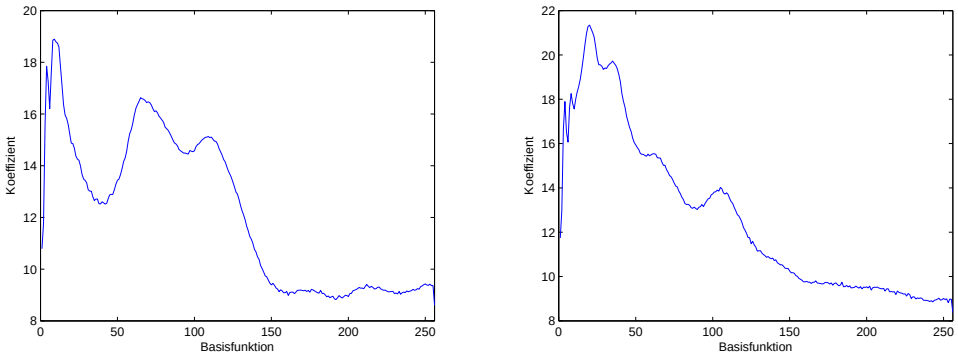
6.1.2 Phoneme

In dieser Arbeit wurde ein Ausschnitt des *Phonem*-Datensatzes verwendet, der von den Autoren des Buches *Elements of Statistical Learning*[14] zur



(a) Klasse 1 (b) Klasse 2

Abbildung 6.1: Beispielsignale des Phonem-Datensatzes



(a) Klasse 1 (b) Klasse 2

Abbildung 6.2: Durchschnittssignale des Phonem-Datensatzes

Verfügung gestellt wird[1]. Es handelt sich hierbei um die Frequenzbänder von Audiosignalen. Jedes Signal entspricht einem bestimmten Laut (Phonem) der das Signal einer Klasse zuordnet. Zudem wurde derselbe Laut von sehr vielen Personen eingesprochen (Signalreihe). Dieser Datensatz eignet sich für die folgenden Untersuchungen sehr gut, weil es sich um 1d-Signalreihen handelt, von denen jedes Signal in eine bestimmte Klasse zugeordnet wurde. Die Eindimensionalität ist wichtig für eine einfache Darstellung in dieser Arbeit, die Klassifizierung für eine Demonstration der Klassifizierungsfähigkeiten von Wörterbüchern. Die Unterschiede zwischen den verschiedenen Klassen sind zudem nichttrivial und die Signale enthalten sehr starkes Rauschen. Der Datensatz enthält 5 Klassen, die jeweils aus 600 bis 1200 Signalen bestehen.

[1]http://www-stat.stanford.edu/ tibs/ElemStatLearn/

6.1.3 Synthetische Datensätze

Für die Analyse geht man oft davon aus, dass eine Signalreihe das Ergebnis eines generierenden Wörterbuches ist.

$$s_i = W_{gen} x_i, \; x_i \text{ ist } k\text{-dünn} \tag{6.1}$$

Das Ziel ist dann die erzeugenden Koeffizienten x_i und das Wörterbuch W_{gen} zu finden.

In synthetischen Experimenten lassen sich umgekehrt für ein gegebenes Wörterbuch W_{gen} beliebig viele Signale erzeugen, die eine dünne Repräsentation haben. Der Vorteil dieser Methode ist nicht nur, dass genügend Signalbeispiele vorhanden sind, was sonst nicht immer der Fall ist, sondern auch, dass das *wahre* Wörterbuch W_{gen} bekannt ist und zudem dessen Größe gewählt werden kann. Außerdem ist klar, mit wie vielen Wörtern jedes Signal dargestellt werden kann. Zudem kann von Fall zu Fall unterschieden werden, wieviel Rauschen den Signalen hinzugefügt werden soll. Falls auf Rauschen verzichtet wird, kann damit das theoretische optimale Verhalten des Algorithmus' getestet werden. Theoretisch ist dieses Verhalten deswegen, weil in der Realität sehr selten Signale ohne Störung vorkommen.

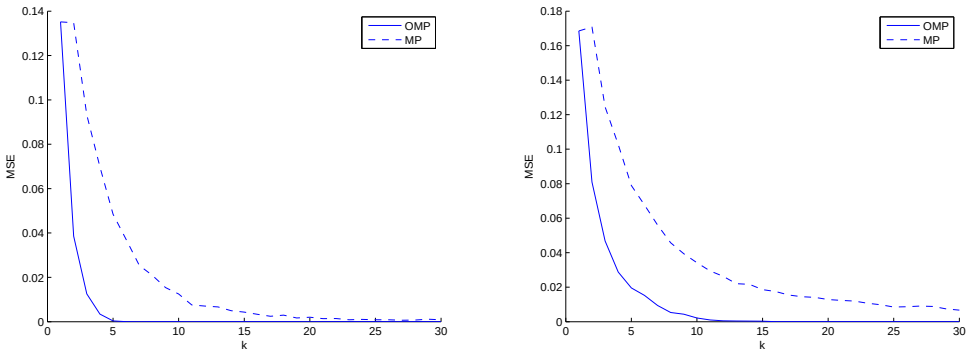
In einer solchen Testumgebung kann ein Wörterbuchalgorithmus in all seinen Parametern vollständig untersucht werden. In dieser Arbeit werden wir einige dieser Parameter betrachten.

Aharon et al. benutzen diese kontrollierte Testumgebung um den K-SVD-Algorithmus auf Effektivität zu überprüfen ([1] Abschnitt *V. Synthetic Experiments*). Dafür wird zuerst eine Signalreihe durch ein beliebiges, uniform verteiltes Wörterbuch W_{gen} erzeugt. Diese Reihe wird dann K-SVD übergeben, um ein Wörterbuch W_{ksvd} zu berechnen. Danach wird überprüft, ob gilt:

$$W_{gen} \approx W_{ksvd} \tag{6.2}$$

Obige Gleichung stellt die Frage, wie gut die *wahre* Datenkomplexität der Signalreihe rekonstruiert wurde. Dies ist nach Aharon et al. gleichzusetzen mit der Rekonstruktion des *wahren* Wörterbuches W_{gen}. Dieser Schritt ist zwar intuitiv eingängig, lässt sich aber kritisieren. Für diesen Schritt muss nämlich erst bewiesen werden, dass $W_{gen} = W_{opt}$. Für beliebige, zufällige Wörterbücher (wie sie in [1] verwendet werden) lässt sich hier schnell ein Gegenbeispiel finden: Falls W_{gen} zwei identische Wörter enthält, kann das tatsächlich optimale Wörterbuch W_{opt} dieses zusätzliche Wort dazu benutzen, ein anderes Signal noch dünner darzustellen.

Ein weiterer Kritikpunkt ist, dass nicht klar ist, wie der *Abstand* von zwei Wörterbüchern definiert ist. In [1] wird einfach für jedes Wort w_i in W_{gen} das



(a) $k_{wahr} = 5$ (b) $k_{wahr} = 10$

Abbildung 6.3: Rekonstruktionsfehler (MSE) für die Greedy-Algorithmen OMP und MP für Signale, die aus k_{wahr} Wörtern erstellt wurden. Anzahl der Wörter im Wörterbuch: 10^3, Länge der Signale: 10^2, Durchschnitt über 10^2 Signale. Man beachte, dass der mittlere Fehler für k_{wahr} nicht unbedingt Null ist.

nächstliegende Wort w_i' in W_{ksvd} gesucht. Eine *Rekonstruktion* von w_i soll dann stattgefunden haben, wenn w_i und w_i' nicht weiter als ϵ voneinander entfernt sind. Doch wie soll ein konkreter Wert für ϵ begründet werden?

Ein dritter Kritikpunkt ergibt sich bei näherer Betrachtung: Der Vergleich in Gleichung 6.2 ist nämlich in gewisser Weise *unfair* gegenüber K-SVD. Dieser ist nämlich, wie alle vorgestellten Algorithmen zur Berechnung von Wörterbüchern, grundlegend abhängig vom Repräsentationsalgorithmus P_W^A. Es könnte unter Umständen vorkommen, dass P_W^A selbst mit W_{gen} nicht in der Lage ist, ein von W_{gen} generiertes Signal s dünn zu repräsentieren (dies ist bei MP und OMP sogar sehr oft der Fall, siehe Abbildung 6.3), mit W_{ksvd} aber schon. Dass dann Gleichung 6.2 nicht gilt, liegt nicht an K-SVD sondern vielmehr an P_W^A. Für eine faire Bewertung sollte verglichen werden, wie sich die Darstellungsfehler der beiden Wörterbücher unter P_W^A zueinander verhalten.

$$S - W_{gen} P^A_{W_{gen}}(S) \approx S - W_{ksvd} P^A_{W_{ksvd}}(S) \tag{6.3}$$

Das Wörterbuch W_{ksvd} ist dann gut, falls der Darstellungsfehler in W_{ksvd} mit P_W^A nicht wesentlich größer ist, als der Darstellungsfehler in W_{gen}.

6.1.4 Bilder als Daten

Aharon et al. benutzen in [1] in einer Demonstration von KSVD Bilder als Signale. Der Algorithmus wurde in dieser Arbeit aber bisher nur im Zusammenhang mit 1d-Signalen erwähnt. Um auch für 2d-Signale 2d-Wörterbücher erstellen zu können werden hier einfach die Spalten eines Bildes zu einem langen 1d-Signal zusammengeklebt. Die Handhabung von Bildern ist also algorithmisch gesehen nicht anders als die von diskreten 1d-Signalen. Vielmehr sind Bilder durch feste Datenformate sogar eingeschränkt in ihrer Amplitude und y-Abtastung (Farbe/Helligkeit). Da 1d-Signale allgemeiner sind, soll hier weiterhin mit diesen gearbeitet werden.

6.2 Berechnen von Wörterbüchern

Es sollen nun die vorgestellten Algorithmen zur Wörterbuchberechnung verglichen werden. Da alle Algorithmen sehr unterschiedlich komplexe Iterationsschritte vollziehen, soll der Vergleich nicht auf der Schrittskala sondern auf der benötigten Rechenzeit in Sekunden erfolgen. Diese Skala ist keineswegs völlig korrekt normiert und sehr stark abhängig von den Implementierungen. Ein absoluter Vergleich soll hier auch nicht getroffen werden, vielmehr soll die Verzerrung vermindert werden. Der interessante Anteil der Diagramme bleibt bei jeder Skalierung die jeweiligen Konvergenzwerte des Approximationsfehlers.

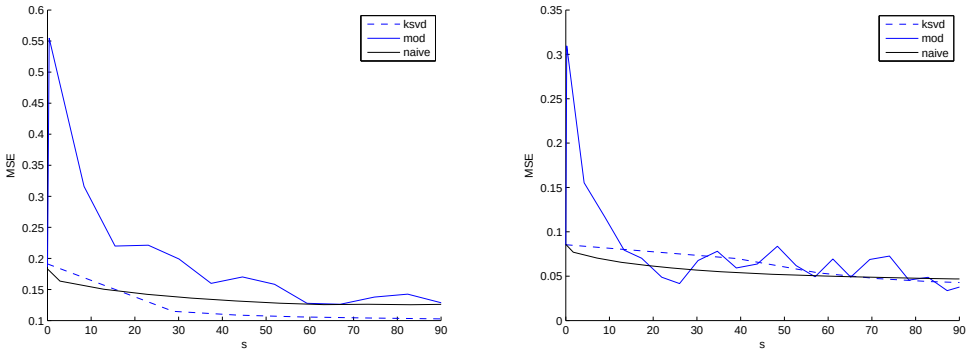
6.2.1 Große Wörterbücher

Die Ursprungsidee eines Wörterbuchs im Gegesatz zu einer Basis ist, dass ein Wörterbuch sehr groß werden kann. Durch wird die Möglichkit begünstigt, alle Signale aus einem bestimmten Kontext oder einer bestimmten Signalverteilung dünn darstellen zu können. Abbildung 6.4 zeigt das Konvergenzverhalten der vorgestellten Algorithmen auf den Phonem-Daten, falls alle Signale mit 7 Wörtern dargestellt werden sollen.

Abbildung 6.5 zeigt dasselbe für 3-dünne Darstellungen.

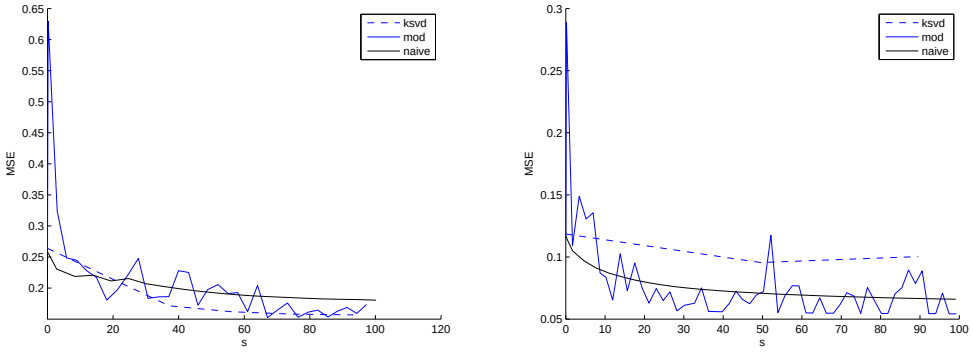
6.2.2 Kleine Wörterbücher

Im oberen Abschnitt wurden die Wörterbucher größer gewählt als die Signallänge. Dies ist auch in gewisser Weise sinnvoll, starteten wird doch mit dem Gedanken eine alternative, korrekte Darstellung für Signale einzuführen (Gleichung (2.2)). Falls das Wörterbuch kleiner ist als die Dimension der Signale ($K < n$), können gar nicht alle Signale aus $\mathbb{R}^n$ dargestellt werden.



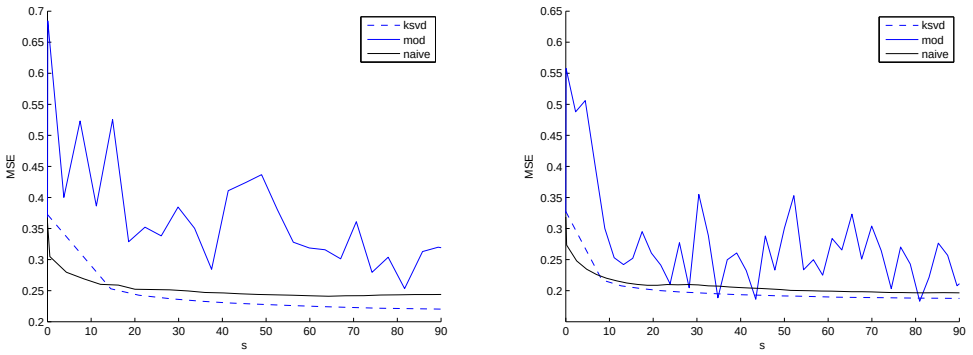
(a) Phonem Klasse 1 (b) Phonem Klasse 2

Abbildung 6.4: Rekonstruktionsfehler für die drei vorgestellten Algorithmen naive, MOD und KSVD für zwei Phonem-Klassen. Wörterbuchlänge ist die doppelte Signaldimension (512), alle Signale sollen mit $k = 7$ dünn dargestellt werden. Der Repräsentationsalgorithmus ist in allen Fällen OMP. Für den *naive*-Algorithmus ist $\eta = 0.1$.



(a) Phonem Klasse 1 (b) Phonem Klasse 2

Abbildung 6.5: Wie Abbildung 6.4, nur für dünnere Darstellungen $k = 3$. Der mittlere Fehler ist größer, weil für eine Approximation weniger Freiheiten gegeben sind.



(a) Phonem Klasse 1 (b) Phonem Klasse 2

Abbildung 6.6: Rekonstruktionsfehler für die drei vorgestellten Algorithmen naive, MOD und KSVD für zwei Phonem-Klassen. Wörterbuchlänge ist die halbe Signaldimension (128), alle Signale sollen mit $k = 7$ dünn dargestellt werden. Der Repräsentationsalgorithmus ist in allen Fällen OMP. Für den *naive*-Algorithmus ist $\eta = 0.1$.

Doch das ist gar nicht immer notwendig. Zum Beispiel wenn Signale nur approximiert dargestellt werden sollen. Außerdem kann es vorkommen, dass die Signalverteilung eines gegebenen Datensatzes nur einen sehr kleinen Untervektorraum von $\mathbb{R}^n$ ausfüllt. Die wahre Dimensionalität der Daten ist also gar nicht n, weshalb auch keine $n \times n$ Matrix zur Darstellung notwendig ist. Desweiteren ist durch die Restriktionen einer dünnen Darstellung jede Implikation von (2.2) zunichte gemacht. Es wird ja in keinem Fall die gesamte Matrix für eine Repräsentation benutzt, weswegen sich die Frage der Vollständigkeit auch für Matrizen mit $K > n$ stellt. Die Aussagekraft der Ergebnisse ist für kleine Wörterbücher vielleicht sogar noch stärker. Dies wird vielleicht deutlich, wenn man sich die K-Means Parallelität in Erinnerung ruft. Die Anzahl der Wörter in KSVD korrespondiert mit der Anzahl der Cluster in K-Means. Natürlich kann eine Clusteranalyse mit sehr vielen Clusterzentren den mittleren quadratischen Fehler sehr klein machen. Doch eine Aussage über die wahre Struktur der Daten ist damit nicht gefunden worden (siehe auch Abschnitt 6.4).

In Abbildung 6.6 ist die Konvergenz der Algorithmen für kleine Wörterbücher ($K = 0.5n$) zu sehen. Wie zu erwarten war, lässt sich KSVD auch in diesem Fall anwenden. Der resultierende MSE ist zwar größer, in Abschnitt 6.3 werden wir aber sehen, dass diese Werte sich nicht direkt vergleichen lassen.

6.3 Generalisierungsverhalten

Die oberen Diagramme zeigen den Prozess der Wörterbuchfindung und sollten verdeutlichen, wie schnell die vorgestellten Algorithmen konvergieren und wie sie sich zueinander verhalten. Es ist offensichtlich, dass für große Wörterbücher der gesamte mittlere quadratische Fehler kleiner ist, als für kleine Wörterbücher. Wenn W mehr Wörter enthält, kann auch mehr Information gespeichert werden, um die Signale S dünn zu repräsentieren. Im Extremfall kann ein Wörterbuch, das soviele Wörter enthält wie es Signale gibt, jedes Signal exakt mit einem Wort beschreiben. Das Wörterbuch ist dann einfach die Signalreihe selbst.

Meistens ist man aber an der sogenannten *Generalisierungseigenschaft* interessiert: Angenommen die Signale in S entstammen derselben Wahrscheinlichkeitsverteilung $P(s)$. Wie gut ist das Wörterbuch nun in der Beschreibung von zukünftigen, noch unbekannten Signalen aus dieser Verteilung? Das gesuchte Wörterbuch soll nicht die Signale beschreiben, die es in der Trainingsphase bekommt, sondern alle Signale, die aus derselben Verteilung stammen. Diese Eigenschaft kann nur geschätzt werden, denn $P(s)$ ist ja nicht bekannt.

Dafür werden meist die vorhandenen, bekannten Signale S in zwei Mengen aufgeteilt, S_{train} und S_{test}. Ein gegebener Algorithmus bekommt nur die Menge S_{train} für die eigentliche Berechnung, die Güte des Ergebnisses wird dann an S_{test} gemessen. S_{train} wird oft größer als S_{test} gewählt.

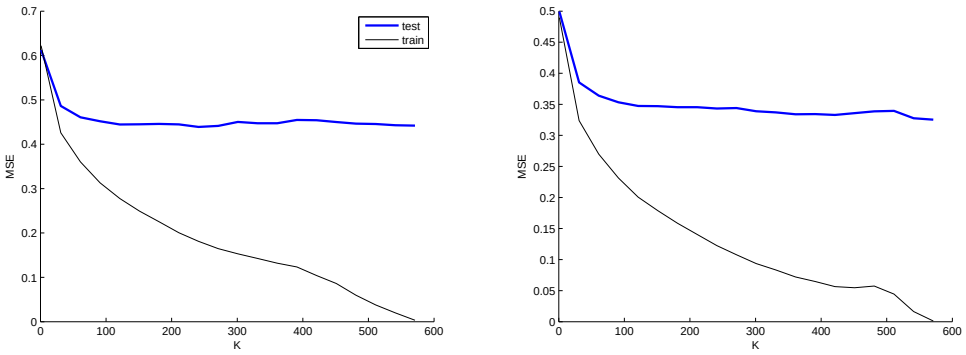
Damit diese beiden Mengen nicht zufällig vor- oder nachteilig gewählt werden gibt es verschiedene Verfahren. Eines davon, das wir auch hier verwenden werden, ist die sogenannte *Cross Validation*.

6.3.1 Cross Validation

Für die n-fache Cross Validation (siehe z.B. [10]) wird die Signalreihe $S = \{s_i\}$ in n gleichgroße Mengen eingeteilt.

$$S = \cup_{k=1}^{n} M_k, |M_k| = \lfloor \frac{|S|}{n} \rfloor \tag{6.4}$$

Die Klassifikation wird nun n mal durchgeführt. Im Schritt i ist M_i die Testmenge, und die Vereinigung aller restlichen M_k die Trainingsmenge. Durch diese Methode wird verhindert, dass ein Signal für seine eigene Vorhersage auch im Training verwendet wurde. Die Güte der Klassifizierung ist dann einfach die durchschnittliche Güte aller n Klassifikationen.



(a) $k = 3$ (b) $k = 7$

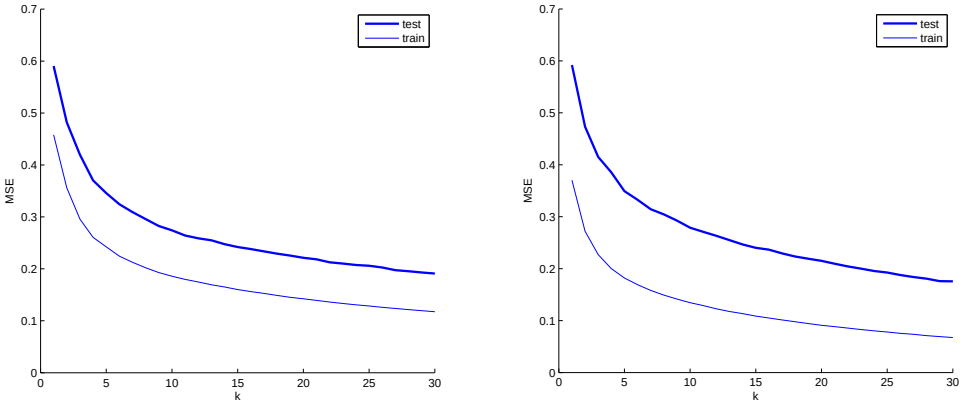
Abbildung 6.7: Mittlerer quadratischer Fehler für S_{test}. Zur Verstärkung des Effekts wurden S_{test} und S_{train} gleich groß gewählt. Da die Berechnung eines Wörterbuches für große K sehr rechenintensiv wird, wurde auf die Cross-Validation verzichtet.

6.3.2 Effekt der Wörterbuchgröße

Das Generalisierungsverhalten ist abhängig von vielen Parametern. Zwei von ihnen sollen hier betrachtet werden: Die Wörterbuchgröße K und die Darstellungsdichte k.

Betrachten wir als erstes die Wörterbuchgröße. Wird K zu klein, kann das Wörterbuch die Komplexität der Daten nicht erfassen, wird K zu groß, werden Details der Trainingsmenge gelernt, die sich nicht auf die Testmenge generalisieren lassen. Der mittlere quadratische Fehler kann dann nicht mehr sinken und sogar wieder ansteigen. Hinzu kommt, dass OMP für große Wörterbücher sehr langsam wird. Wie groß soll nun K gewählt werden? Dazu wird in Abbildung 6.7 der mittlere quadratische Fehler auf der Testreihe abhängig von der Wörterbuchgröße aufgetragen. Der Repräsentationsalgorithmus ist KSVD, da er in 6.2 am besten abgeschnitten hat.

Man vergleiche Abbildung 6.7(b) mit Abbildung 6.4(a): In Ersterer ist $K = 512$ und $k = 7$, aber mittlere quadratische Fehler geringer als $MSE(K = 512)$ in Abbildung 6.7(b). Dies ist konsistent mit den vorigen Überlegungen: In der unteren Abbildung hatte KSVD kein Wissen über S_{test}. Das resultierende Wörterbuch kann diese folglich schlechter darstellen. Man beachte, dass ab $K \approx 130$ keine Verbesserung von $MSE(S_{train})$ mehr stattfindet. Ab diesem Punkt kann KSVD den MSE der Trainingsmenge zwar noch verringern, die dabei extrahierten Strukturen können aber nicht für die unbekannten Signale der Testmenge verwendet werden und sind möglicherweise Rauschen.



(a) $K = 128$ (b) $K = 256$

Abbildung 6.8: MSE für S_{test} und S_{train} für verschiedene Darstellungen $k = k_1 = k_2$. Auch diese Berechnung ist so rechenintensiv, dass auf Cross-Validation verzichtet werden musste

6.3.3 Effekt der Darstellungsdichte

Es gibt zwei weitere Parameter in diesem Szenario, die es Wert sind untersucht zu werden:

- Wie dünn sollen die Signale dargestellt werden, wenn das Wörterbuch berechnet wird? Dieser Parameter sei k_1.
- Wie dünn sollen die Signale der Testmenge dargestellt werden, um den MSE zu berechnen? Dieser Parameter sei k_2.

Die Wahl von k_1 und k_2 ist nicht trivial und ist abhängig von der gewünschten Anwendung. Zudem ist der MSE auch noch abhängig von der Wörterbuchgröße K, die hier zwar getrennt untersucht wurde, in der Praxis aber kombiniert analysiert werden sollte, um ein optimales Verhalten zu erzielen.

Abbildung 6.8 zeigt, wie sich die Darstellungsdichte auf den mittleren quadratischen Fehler der Testmenge auswirkt. Wie erwartet unterscheiden sich die Fehlerkurven für die gewählten K nicht besonders. Lediglich $MSE(S_{train})$ unterscheidet sich, was damit begründet werden kann, dass Details in der Trainingsmenge gelernt werden, die sich nicht auf die Testmenge generalisieren lassen. Dies war zu erwarten, denn im vorigen Abschnitt wurde erkannt, dass ab $K \approx 130$ keine Verbesserung des MSE mehr möglich ist. Offensichtlich gilt dies auch für größere k.

Die Untersuchung von k_1 und k_2 muss hier nicht enden. Es sind auch Szenarios möglich, in denen zum Beispiel $k_1 < k_2$ ist.

6.3.4 Kompression

Das Szenario der Bildkompression, wie zum Beispiel in [2], stellt dieselben Fragen wie die oberen Abschnitte. Es sei eine Trainingsreihe S_{train} von Beispielsignalen gegeben, die dazu benutzt werden soll, ein Wörterbuch zu generieren. Dieses Wörterbuch W soll in der Zukunft unbekannte Signale S_{test} der gleichen (unbekannten) Signalverteilung möglichst gut komprimieren. Kompression ist in diesem Fall ein anderes Wort für die dünne Darstellung. Zum Beispiel haben die Signale des Phonem-Datensatzes 256 Einträge, die folglich auch gespeichert werden müssen. Falls sie mit einem erlernten Wörterbuch mit nur k Wörtern dargestellt werden können, müssen nur k Koeffizienten und die zugehörigen k Indizes der Wörter gespeichert werden. Folgende Fragen stellen sich:

- Wieviele Wörter soll W haben, also welches K ist effizient? Ein größeres K verlangsamt das Erlernen des Wörterbuchs und zusätzlich den Repräsentationsalgorithmus (OMP verhält sich quadratisch).
- Welches k_1 soll beim Erlernen von W benutzt werden?
- Welches k_2 zur Repräsentation der zukünftigen Signale ist ein guter Kompromiss zwischen Kompression und Approximation?

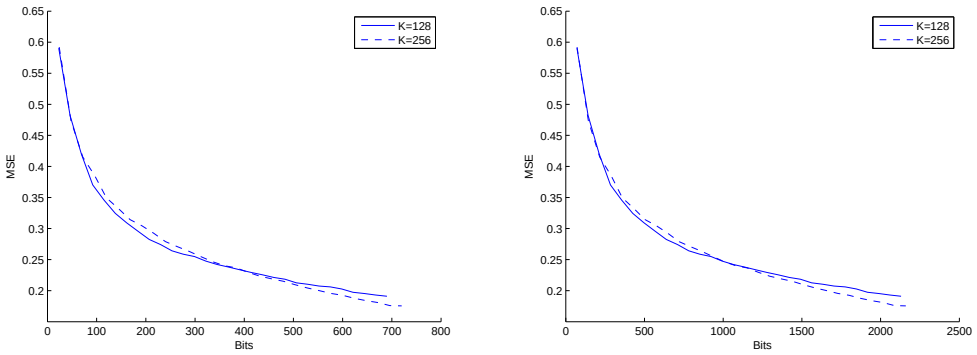
Die Beantwortung all dieser Fragen ist komplex und wurde selbst von [2] nicht ausführlich abgehandelt. Zur Veranschaulichung soll hier gezeigt werden, wie sich die Approximation der Signale aus S_{test} zum benötigten Speicherplatz verhält. Je größer k, desto mehr Bits $b(k)$ werden zur Speicherung des Signals benötigt.

$$b(k) = k(C + \log_2(K)) \tag{6.5}$$

Die Anzahl der Bits sind auch abhängig von $\log(K)$, da die benötigten Koeffizienten in K *adressiert* werden müssen. C sei die konstante Anzahl an Bits, die benötigt wird um einen Koeffizienten zu speichern. Die Wahl von C hat natürlich wieder Rückwirkung auf die Genauigkeit, also den MSE, was in Abbildung 6.9 aber vernachlässigt wurde.

Normalerweise wird diesen Verhalten in einem sogenannten Rate-Distortion-Diagramm gezeigt. Um Verwirrung zu vermeiden soll auf die Einführung dieser Einheiten verzichtet werden.

Abbildung 6.9 zeigt, dass ein kleineres K in manchen Fällen zu besserer Kompression führt. Die Wahl von K ist also nicht trivial: zu kleine Wörterbücher ermöglichen überhaupt keine gute Darstellung der Testsignale, zu große Wörterbücher können keine bessere Darstellung erzielen, kosten aber zusätzlichen Speicherplatz in der Adressierung der Koeffizienten.



(a) $C = 16$ (b) $C = 64$

Abbildung 6.9: MSE der Testsignale abhängig von der Kompression (umgerechnete Anzahl der Koeffizienten) für verschiedene Wörterbuchgrößen und Repräsentationsgenauigkeiten C der Koeffizienten.

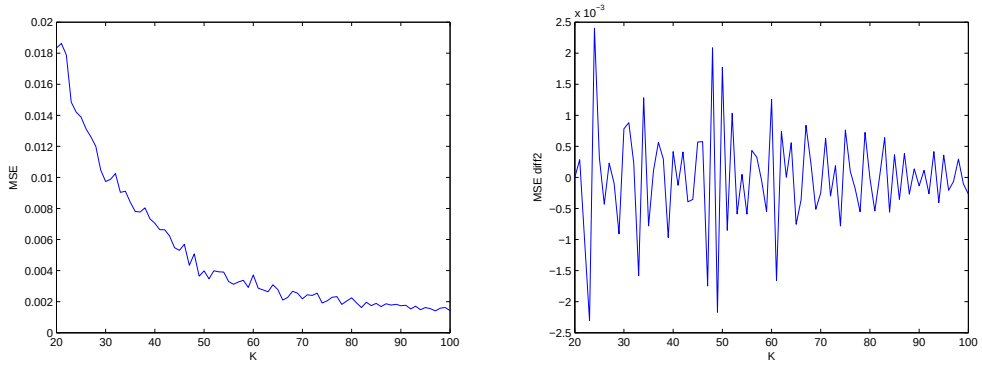
6.4 Wahre Datenkomplexität

Oft ist es wünschenswert, die *wahre* Komplexität eines Datensatzes herauszubekommen, um generelle Aussagen über die Domäne zu machen, aus die die Signale entstammen, oder um Parameter von Algorithmen zu rechtfertigen (siehe google news Beispiel weiter unten). Doch leider ist die *wahre* Komplexität in diesem Fall ein vager Begriff und lässt sich nicht direkt in eine Formel übersetzen.

Mit wie vielen Clustern lässt sich der Datensatz so darstellen, dass der erwartete mittlere Fehler minimal ist? Diese Frage lässt sich einfach beantworten: So viele Cluster wie Datenpunkte! Denn mit jedem neuen Cluster sinkt auch der mittlere Fehler. Doch dies entspricht in den wenigsten Fällen der *gemeinten, intuitiven* Datenkomplexität.

Google News[2] ist ein gutes Beispiel für die Demonstration von Datenkomplexität. Hier müssen eine große Anzahl an Artikeln in Gruppen eingeteilt werden. Jede Gruppe von Artikeln soll über dasselbe Ereignis berichten. Die Anzahl der Ereignisse, über die die Artikel berichten, entspricht der wahren Datenkomplexität und ist nicht bekannt. Ein bewährtes Verfahren für diese Bestimmung berechnet die zweite Ableitung der $MSE(k)$-Funktion. Diese Funktion weist jedem k einen mittleren Fehler, der entsteht, wenn man den Datensatz mit k Cluster approximiert. Ein sehr große Veränderung in dieser Funktion (große erste Ableitung) bedeutet eine starke Reduzierung des Fehlers durch die Hinzunahme eines weiteren Clusters. Verändert sich

[2]http://news.google.de



(a) MSE für variables K

(b) Zweite Ableitung des MSE bzgl. K

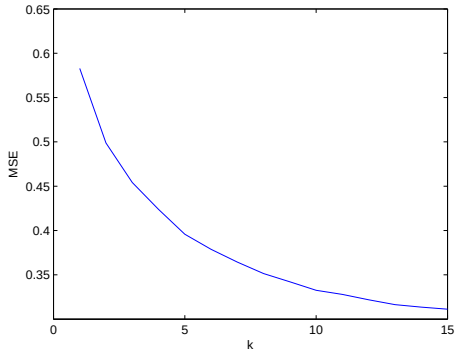
Abbildung 6.10: Berechnung der wahren Wörterbuchgröße für ein erzeugtes Wörterbuch und $K_{wahr} = 50$

diese Reduktion des Fehlers (zweite Ableitung) bedeutet dies, dass die Hinzunahme des Clusters einen größeren beziehungsweise kleineren Effekt hatte. Große Knicke (Extrema in der zweiten Ableitung) signalisieren wichtige Veränderungen in der Approximation der wahren Datenkomplexität. In unserem Beispiel heißt das: n Ereignisse werden erst durch n Cluster gut approximiert. Alle Clusterings mit mehr als n Cluster können den Fehler zwar noch weiter reduzieren, aber nicht so stark wie Cluster 1 bis n. Man kann sich nun streiten welcher Knick der $MSE(k)$ Funktion die *wahre* Komplexität signalisiert. Meistens nimmt man den größten oder einen Kompromiss zwischen dem größten und dem letzten.

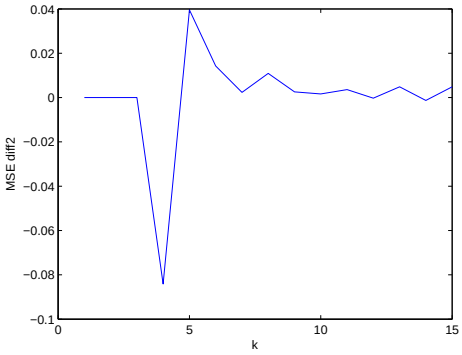
Ähnlich wie k-Means auf die Struktur der Daten schließen kann, könnte doch auch KSVD oder jeder andere Wörterbuchalgorithmus diese Aufgabe erfüllen. Abbildung 6.10 zeigt MSE(k) und die zugehörige zweite Ableitung für eine generierte Signalreihe, die aus einem Wörterbuch der Länge 50 erstellt wurde. Leider ist — wie so oft — die zweite Ableitung sehr stark verrauscht. Ein großer Knick bei $K = 50$ lässt sich nur erahnen.

Der zweite Parameter des Wörterbuchalgorithmus' ist die Darstellungsdichte k. Lässt sich diese vielleicht rekonstruieren? Dazu wird eine Signalreihe erstellt, für die k bekannt ist ($k_{wahr} \in \{5, 20\}$) und dann untersucht, wie sich der MSE abhängig von k verhält. Aber auch diese Ergebnisse (in Abbildung 6.11) lassen nicht eindeutig auf das ursprüngliche k zurückschließen.

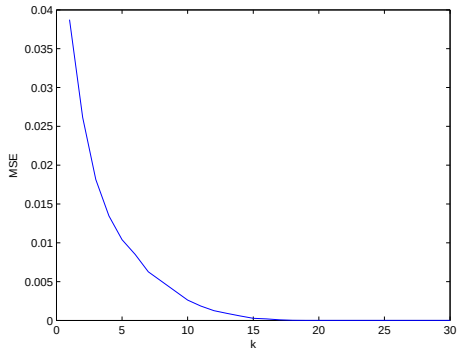
Da dieses Szenario sehr viele Parameter hat ist nicht auszuschließen, dass KSVD nicht dazu geeignet ist, k oder K aus einer Signalreihe zu rekonstruieren. In den Experimenten dieser Arbeit konnte jedoch keine geeignete Konfiguration gefunden werden.



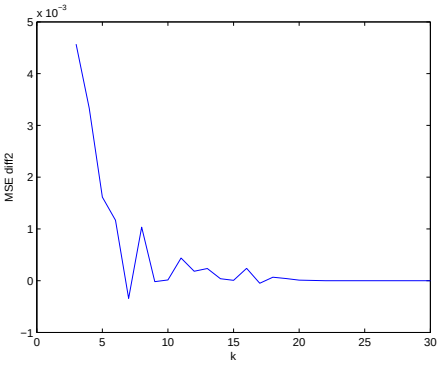
(a) MSE für variables k, $k_{wahr} = 5$, $W \in \mathbb{R}^{20\times 50}$



(b) Zweite Ableitung des MSE bzgl. k, $k_{wahr} = 5$



(c) MSE für variables k, $k_{wahr} = 20$, $W \in \mathbb{R}^{20\times 200}$



(d) Zweite Ableitung des MSE bzgl. k, $k_{wahr} = 20$

Abbildung 6.11: MSE abhängig von der Darstellungsdichte k für verschiedene *wahre* Darstellungsdichten

6.5 Klassifikation

Klassifikation ist die Einteilung von gegebenen Daten in bestimmte Klassen. Im Gegensatz zum Clustering, einer unüberwachten Lernmethode, soll hier überwacht gearbeitet werden. Die Klassen der einzelnen Signale sind im vorraus bekannt und das Verfahren darf mit diesen Informationen arbeiten. Meist ist die *Vorhersagekraft* oder *Verallgemeinerung* eines Verfahrens interessant. Wie gut kann das Verfahren die Klasse eines Signals vorhersagen, mit dem es nicht trainiert wurde? Für die Messung dieser Größe gibt es einige Verfahren. Sehr beliebt ist die oben beschriebene *Cross Validation*.

6.5.1 F-Maß

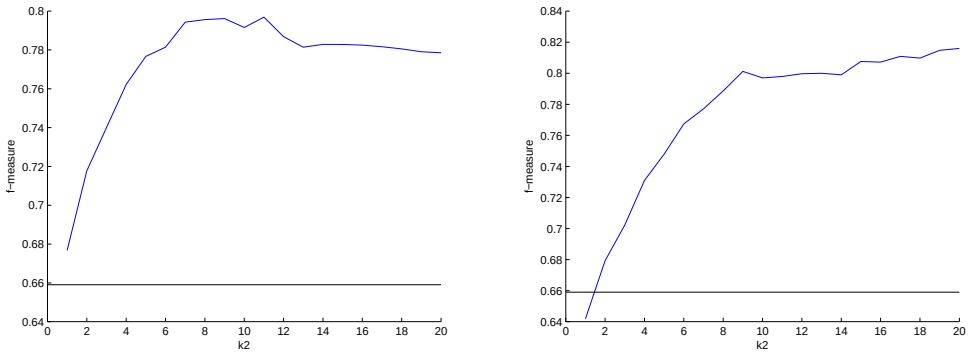
Die Anforderungen an Klassifikatoren sind nicht immer gleich. In einem Fall kann es wichtig sein, dass bezüglich einer Klasse besonders *präzise* gearbeitet wird, also keine falsche Einteilung in diese Klasse erfolgt (*precision*). Anders ausgedrückt: der Klassifikator darf bezüglich dieser Klasse keine *false positives* zurückliefern. Dies ist oft bei binärer Klassifizierung interessant, zum Beispiel der Einteilung von Suchergebnissen in die Klassen *relevant* und *nicht relevant*. In einem anderen Fall kann es genauso wichtig sein, dass der Klassifikator möglichst alle Elemente einer bestimmten Klasse richtig klassifiziert, egal wieviele Elemente dabei von anderen Klassen falsch zugeordnet werden (*recall*). Dies ist zum Beispiel bei der Suche in Gesetzestexten der Fall. Hier darf es nicht vorkommen, dass ein relevantes Ergebnis nicht angezeigt wird, es ist aber egal, wenn eine sehr große Menge an nicht relevanten Ergebnissen zusätzlich zurückgegeben wird.

Keine dieser beiden Maße, *precision* und *recall*, kann unabhängig voneinander optimiert werden. Deswegen gibt es einige kombinierte Maße, die den Klassifikator unter beiden Gesichtspunkten bewerten. Ein beliebtes kombiniertes Maß ist das F-Maß, das hier verwendet werden soll. Es berechnet das harmonische Mittel von $precision(p)$ und $recall(r)$:

$$F = 2\frac{p * r}{p + r} \tag{6.6}$$

6.5.2 Nearest Neighbor

Es ist oft nicht möglich Klassifikationsgüten intuitiv zu bewerten, weil die Schwierigkeit des Klassifikationsproblems abhängig von den verwendeten Daten stark schwankt. Deswegen werden neue Klassifikatoren oft mit den Leistungen alter Klassifikatoren verglichen. Der Vergleichsklassifikator in dieser Arbeit ist aufgrund seiner Einfachheit *Nearest Neighbor*.



(a) $k_1 = 5$ (b) $k_1 = 20$

Abbildung 6.12: Dreifach cross-validiertes Klassifikationsergebnis eines KSVD Wörterbuches für die Phonem-Klassen 1 und 2. Schwarz: Dreifach cross-validiertes Ergebnis der Nearest Neighbor Klassifikation.

Soll ein Signal s klassifiziert werden, wird das Signal s' aus der Trainingsmenge gesucht, das den kleinsten Abstand zu s hat (nach einer bestimmten Norm). Die Klasse von s ist dann die Klasse von s'.

6.5.3 Klassifikation mit Wörterbüchern

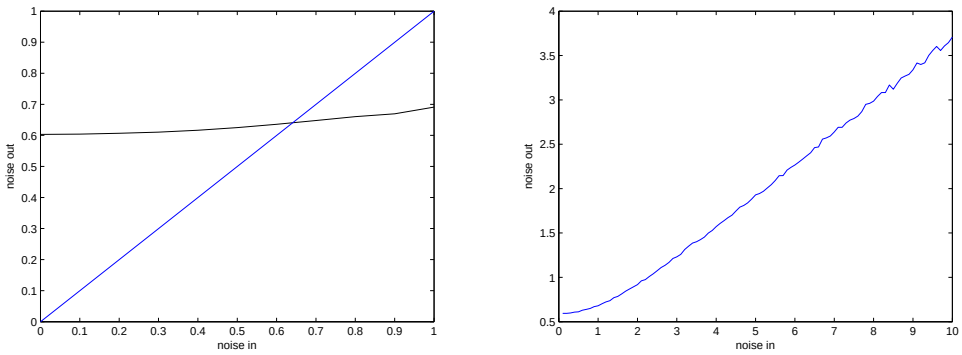
Es wird nun das Klassifikationsverfahren getestet, das in Abschnitt 5.2 beschrieben wurde. Das Beispiel versucht zwischen Signalen aus den Klassen 1 und 2 des Phonem-Datensatzes zu unterscheiden. Dazu werden mit KSVD zwei Wörterbücher W_1 und W_2 darauf trainiert, die Signale der Trainingsmengen k_1-dünn darzustellen. Danach wird für alle Signale s der Testmenge das Wörterbuch gesucht, mit dem es sich mit k_2 Wörtern besser darstellen lässt. Abbildung 6.12 zeigt das Ergebnis der Klassifikationen im F-Maß abhängig von k_2 für zwei verschiedene k_1.

Erstaunlich ist, dass für $k_1 = 5$ Darstellungen $k_2 > k_1$ bessere Ergebnisse liefern als $k_2 < k_1$. Das Wörterbuch klassifiziert in diesem Fall mit Darstellungen, für die es gar nicht trainiert wurde.

6.6 Filtern mit Kompression

Nun wird ein modifiziertes Denoising-Szenario (Gleichung (2.14)) untersucht:

$$s = x + \eta \tag{6.7}$$



(a) Rauschreduzierung für schwaches Rauschen (b) Rauschreduzierung für starkes Rauschen

Abbildung 6.13: Beispiel Rauschreduzierung für Klasse 1 des Phonem-Datensatzes mit falscher Wahl der Parameter: $K = 512$, $k_1 = k_2 = 5$, *noise in* ist die Varianz von η.

Wir vernachlässigen einen theoretischen, analogen Ursprung der Signale und fügen vorhandenen digitalen Signalen $X = \{x_i\}$ eine normalverteilte Störung hinzu.

Zuerst wird KSVD benutzt um ein Wörterbuch W zu trainieren. Die Extraktion von x' nach Beobachtung von s erfolgt dann durch Darstellung von s n W nach dem Verfahren der verlustbehafteten Kompression.

Das modifizierte Denoising Szenario ist insofern schwierig, weil die Signale x je nach Datensatz schon eine Störung enthalten können. Der verwendete Phonem-Datensatz weist ein solches starkes Rauschen auf. Durch diese unbekannte Störung ist gar nicht klar, ob ein Wörterbuch existiert, das alle x dünn darstellen kann. Da wir meist mit Approximationen rechnen wird diese Störung das Verfahren nicht unmöglich machen, das optimal mögliche Ergebnis aber verschlechtern. Es lässt sich dann nicht abschätzen, ob der Mißerfolg am Verfahren oder an den Daten selbst liegt. Desweiteren hat die Wahl der Parameter $K, k_1, k_2(= T)$ sowie die Anzahl der Iterationen von KSVD starke Auswirkungen auf das Ergebnis. Sollte das berechnete W gar nicht in der Lage sein, ein Testsignal mit einem kleinen Fehler mit k_2 Wörtern zu repräsentieren (verlustbehaftete Kompression), hat die Untersuchung des Rauschfilterverhaltens keinen Sinn. In Abbildung 6.13(a) ist dies gut zu sehen: Falls kein Rauschen hinzugefügt wird ergibt sich trotzdem ein sehr großer Fehler (≈ 0.8). Dadurch ist das geschätzte x' sehr weit vom ursprünglichen x entfernt, noch bevor x überhaupt gestört wurde.

6.6.1 Filtern synthetischer Daten

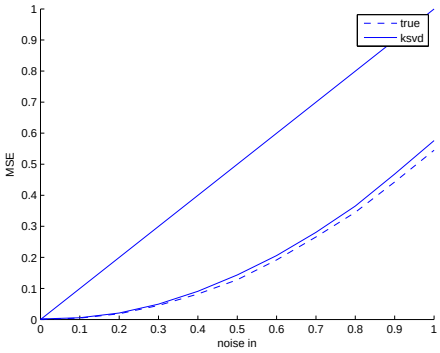
Im vorigen Abschnitt wurde beschrieben, dass die Entfernung von Rauschen beschränkt sein kann durch das Darstellungsvermögen von W. Deswegen soll das Denoising Szenario zuerst mit einem idealen Wörterbuch getestet, bevor reale Daten betrachtet werden.

Ein Wörterbuch W_{true} wird erstellt, dass eine k_1-dünne Signalreihen S_{train} und S_{test} erzeugt. KSVD berechnet anhand S_{train} ein Wörterbuch $W \in \mathbb{R}^{n \times K_2}$, das versucht, die Daten k_2-dünn darzustellen (k_1 und k_2, sowie K_1 und K_2 können ungleich sein, denn die Ursprungswerte k_1 und K_1 sind im Realfall unbekannt). Danach wird verglichen, wie gut die beiden Wörterbücher anhand von verlustbehafteter Kompression ($N = 5$) hinzugefügtes Rauschen aus den Signalen S_{test} extrahieren können. Die Anzahl der darstellenden Koeffizienten N im letzten Schritt ist hier willkürlich festgelegt, aber für beide Wörterbücher gleich. Die Diagramme in Abbildung 6.14 zeigen also nicht unbedingt das optimale Filterverhalten, sind aber dafür vergleichbar.

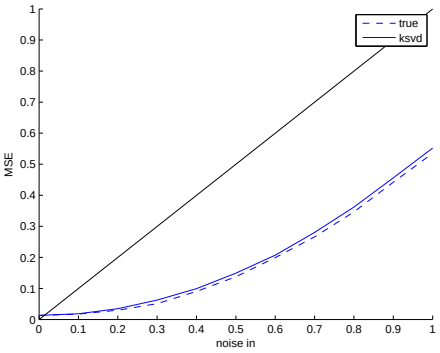
Diese Beispiele sollen zeigen, dass es durchaus möglich ist, das *optimale* Verhalten eines theoretischen, optimalen Wörterbuches mit einem berechneten Wörterbuch zu bekommen. Wie gut dieses optimale Verhalten ist, muss von Fall zu Fall entschieden werden und hängt stark von der Beschaffenheit der Daten ab.

6.6.2 Filtern realer Daten

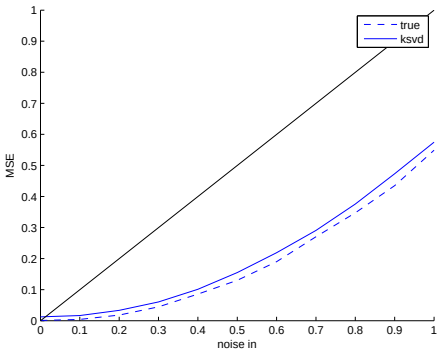
Lässt sich das oben beschriebene Verfahren auf reale Daten übertragen? Der Fokus soll nun darauf liegen, welche Parameter sich wie auswirken können. Wir haben zum Beispiel in Abbildung 6.13 gesehen, dass eine schlechte Wahl von k_1 jede Filterwirkung der verlustbehafteten Kompression zunichte macht. Algorithmus 6 berechnet das Filterverhalten eines KSVD Wörterbuches. In Abbildung 6.15 sieht man die Ergebnisse dieses Verfahrens für Signale der Phonem Klasse 1. Da die Signale dieses Datensatzes stark verrauscht sind, lässt sich kaum einschätzen, ob diese Ergebnisse gut oder schlecht sind. Dies könnte in einer weiterführenden Arbeit bestimmt werden, indem Vergleiche zu anderen Rauschfiltern gezogen werden. Im Rahmen dieser Arbeit soll das Ergebnis genügen, dass das vorgestellte Verfahren zumindest qualitativ in der Lage ist, Störungen aus Signalen zu entfernen.



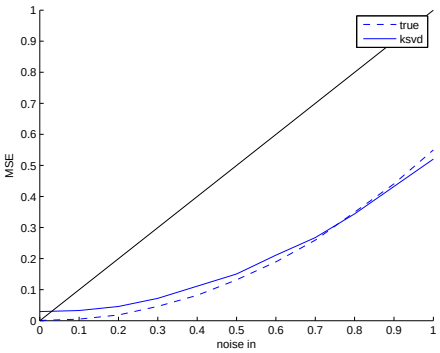
(a) $k_2 = k_1 = 5$ und $K_1 = K_2$



(b) $k_1 > k_2$ und $K_1 = K_2$



(c) $k_2 > k_1$ und $K_1 = K_2$

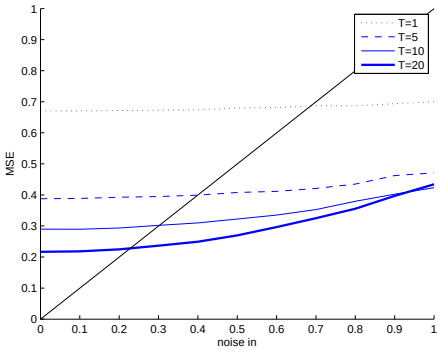


(d) $k_2 = k_1 = 5$ und $K_1 > K_2$

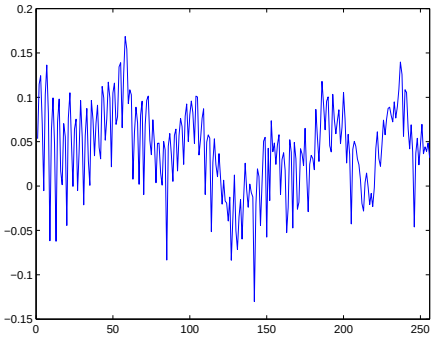
Abbildung 6.14: Beispiel Rauschreduzierung synthetischer Daten. $W_{true} \in \mathbb{R}^{50\times200}$, 2000 Signale, 10% davon Testmenge.

Algorithmus 6 Berechne das Filterverhalten eines KSVD Wörterbuches

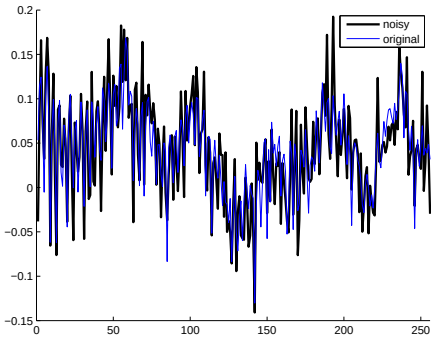
1: **procedure** FILTER-RESPONSE($S_{train}, S_{test}, K, \epsilon, T$)
2: $W \leftarrow W_{KSVD}(S_{train}, \epsilon)$ ▷ Berechne KSVD Wörterbuch
3: ▷ Taste den Rauschparameter ab, zum Beispiel in 0.1 Schritten
4: **for** $\eta \in [0..1]$ **do**
5: $e_\eta = \text{MSE}(W, S_{test}, \eta)$
6: **end for**
7: **return** e
8: **end procedure**
9: **procedure** MSE(W, S, σ)
10: $e \leftarrow 0$
11: **for all** $x_i \in S$ **do**
12: $s \leftarrow x + \eta(\sigma)$ ▷ Addiere normalverteiltes Rauschen
13: $s' \leftarrow P_W^{\text{OMP}}(s_i, T)$ ▷ OMP mit T Iterationen
14: $e \leftarrow e + |s_i - s'|^2$
15: **end for**
16: **return** $\frac{e}{|S|}$
17: **end procedure**



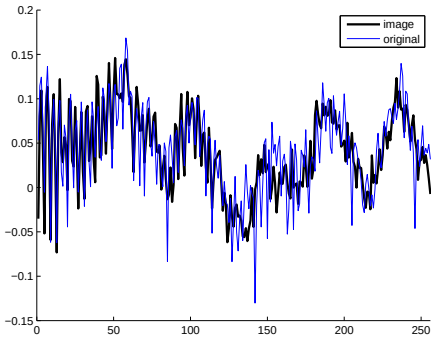
(a) Rauschreduzierung für verschiedene *thresholds* T



(b) Beispielsignal x aus Testreihe



(c) Das gestörte Signal $s = x + \eta$



(d) Das extrahierte Signal x'

Abbildung 6.15: Beispiel Rauschreduzierung die Phonem Klasse 1.

Kapitel 7

Diskussion

7.1 Schwachpunkte der Wörterbuchalgorithmen

Alle vorgestellten Algorithmen haben eine sehr große Schwachstelle gemein, die sich durch ein einfaches Beispiel illustrieren lässt. Man stelle sich ein generierendes Wörterbuch vor, das aus zwei gleichgroßen Mengen $M_1 = \{m_i^1\}, M_2 = \{m_i^2\}$ von Wörtern besteht. M_1 hat nur Werte ungleich null für die erste Hälfte der Wortlänge, bei M_2 ist es umgekehrt. Man generiere nun eine dünne Signalreihe, wobei für jedes Signal Wörter aus beiden Mengen gezogen werden müssen.

Beobachten wir nun K-SVD bei der Arbeit: In Zeile 11 wird eine Approximation der Fehlermatrix ersten Ranges gesucht. Sei w das aktuelle Wort, das durch ein w' aktualisiert werden soll. Sei c der Vektor der Koeffizienten. Sagen wir, der Algorithmus hätte bis hierhin gute Arbeit geleistet und nur Signale selektiert, die ein generierendes Wort auf der linken Seite gemeinsam haben und w sieht auf dieser Seite diesem generierenden Wort schon ziemlich ähnlich. Auf der rechten Signalseite allerdings wurde die selektierten Signale aus völlig unterschiedlichen Wörtern generiert. Das neue Wort w' und die zugehörigen Koeffizienten sollen bei der Rang-1-Approximation den Fehler E_k bestmöglich minimieren. w'_k wird also auf beiden Hälften versuchen, möglichst gute Arbeit zu leisten. Doch auf der rechten Signalseite ist dies gar nicht richtig. w' kann hier im besten Fall nur soetwas wie einen Mittelwert erzeugen. Dies ist in gewisser Weise *greedy*. Hier wäre es sinnvoll, wenn w' den Fehler lokal nicht minimiert sondern das Restsignal von anderen Wörtern darstellen lässt. Hieraus ergibt sich, dass K-SVD in diesem Szenario frühestens mit $|M_1| * |M_2|$ Wörtern zu einem guten Ergebnis kommt, weil alle Kombinationen der m_i^1, m_j^2 abgedeckt sein müssen. Doch selbst das ist unrealistisch, weil alle m_i^k kaum korrekt erkannt werden können.

Die Aussage hier soll sein, dass sich schon *methodisch* die theoretische Untergrenze von $|M_1| + |M_2|$ niemals erreichen lässt. Im Allgemeinen ist dies noch schwerwiegender, da generierende Wörterbücher aus mehr als zwei lokalen Bereichen aufgebaut sein können.

7.2 Komplexität des Wörterbuchproblems

Ein optimales Wörterbuch für eine gegebene Signalverteilung zu finden, ist eine sehr komplexe Aufgabe.

Es muss nicht nur berechnet werden, wie die generierenden Wörter aussehen, sondern auch aus genau welchen Wörtern jedes Signal entstanden ist. Verändert man das Wörterbuch, dann ändern sich auch die Koeffizienten der Signale. Hat das generierende Wörterbuch die Länge n, dann gibt es für die Auswahl von nur 2 Wörtern schon $n(n-1)$ Möglichkeiten.

Diese Komplexität betrachtet K-SVD in gewisser Weise gar nicht. Ein *Cluster* besteht aus den Signalen, die alle ein gemeinsames Wort benuzten. Doch die übrigen Wörter werden gar nicht betrachtet. Es stellt sich die Frage, ob aus der Kombination von Koeffizienten zusätzliche Information extrahiert werden kann.

7.3 Schwachpunkt des Wörterbuchmodells

Neben den konkreten Algorithmen, hat auch das Wörterbuchmodell selbst einen Schwachpunkt. Die indirekte Definition eines Kontextes/einer Signalverteilung durch ein Wörterbuch vernachlässigt die Verteilung und Kombination der Koeffizienten.

Sei ein generierendes Wörterbuch W_{gen} gegeben. Nun werden zwei k-dünne Signalreihen S_1,S_2 erzeugt, die jeweils unterschiedliche Kombinationsmöglichkeiten von Wörtern zulassen. Sei zum Beispiel $k = 3$. Ein Signal aus S_1 sei nun zusammengesetzt aus der Wortkombination w_a, w_b und w_c. Die Einschränkung an Signalreihe S_2 sei, dass sich keine dieser Kombinationen aus S_1 in S_2 wiederholt. Wie schon im vorherigen Abschnitt erwähnt unterscheidet K-SVD nicht zwischen Koeffizientenkombinationen, aber auch das Wörterbuchmodell selbst hat keine Möglichkeit, diese Information aufzunehmen. K-SVD und jeder andere Wörterbuchalgorithmus, der nur mit dem Datenspeicher einer Matrix arbeitet, wird die Signalreihen S_1 und S_2 optimal unterscheiden können. Genau dasselbe Ergebnis bekommt man, wenn man die Verteilung der Koeffizientenamplituden beschränkt.

7.4 Zusammenfassung

Diese Arbeit hat gezeigt, dass ein angepasstes Wörterbuch für eine gegebene Signalreihe ein sehr großes Anwendungsspektrum hat. Die referenzierten Arbeiten zeigen große, vielversprechende Fortschritte in Kompression und Denoising. Bekannte, altbewährte Verfahren wie MPEG oder JPEG, die nicht kontextsensitiv sind, werden durch diesen kontextsensitiven Ansatz weit abgehängt. Besonders bestechend ist die Einfachheit: Für die Klassifikation von Signalen zum Beispiel entfällt das mühsame Erstellen und Auswählen von Features. Features können im Wörterbuchansatz in gewisser Weise automatisch gelernt werden (das Wörterbuch), denn der Wörterbuchansatz hat keine Probleme mit hochdimensionalen Daten. In [22] wird demonstriert, dass die Perspektive der dünnen Repräsentation auch auf Feature-Vektoren funktionert.

Der Wörterbuchansatz hat erst in den letzten Jahren große Aufmerksamkeit erlangt und es lässt sich kaum erahnen, was in den nächsten Jahren damit gemacht werden wird.

Ich bin gespannt.

Literaturverzeichnis

[1] M. Aharon, M. Elad, and A. Bruckstein. K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation. *Signal Processing, IEEE Transactions on [see also Acoustics, Speech, and Signal Processing, IEEE Transactions on]*, 54(11):4311–4322, 2006. doi: 10.1109/TSP.2006.881199. URL `http://dx.doi.org/10.1109/TSP.2006.881199`.

[2] Ori Bryt and Michael Elad. Compression of facial images using the k-svd algorithm. *J. Visual Communication and Image Representation*, 19(4):270–282, 2008. URL `http://dblp.uni-trier.de/db/journals/jvcir/jvcir19.html`.

[3] Grace Chang, Bin Yu, and Martin Vetterli. Image denoising via lossy compression and wavelet thresholding, 1997.

[4] S. S. Chen. Basis pursuit. *Ph.D. Thesis, Department of Statistics, Stanford University*, 1995.

[5] Scott Shaobing Chen, David L. Donoho, and Michael A. Saunders. Atomic decomposition by basis pursuit. *SIAM Rev.*, 43(1):129–159, 2001. ISSN 0036-1445. doi: http://dx.doi.org/10.1137/S003614450037906X.

[6] George B. Dantzig. *Linear programming and extensions.* Rand Corporation Research Study. Princeton Univ. Press, Princeton, NJ, 1963.

[7] G. Davis, S. Mallat, and M. Avellaneda. Adaptive greedy approximations. *Constructive Approximation*, 13(1): 57–98, March 1997. doi: 10.1007/BF02678430. URL `http://dx.doi.org/10.1007/BF02678430`.

[8] D. Donoho. Denoising by soft-thresholding. In *IEEE Trans. Inform. Theory 41, pp. 613–627*, 1995.

[9] David L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, 2006. URL http://dblp.uni-trier.de/db/journals/tit/tit52.html.

[10] B. Efron and G. Gong. A Leisurely Look at the Bootstrap, the Jackknife, and Cross-Validation. *The American Statistician*, 37:36–48, February 1983.

[11] Michael Elad and Michal Aharon. Image denoising via learned dictionaries and sparse representation. In *In CVPR*, pages 17–22, 2006.

[12] K. Engan, S. O. Aase, and J. Hakon Husoy. Method of optimal directions for frame design. In *Proceedings of the Acoustics, Speech, and Signal Processing, 1999. on 1999 IEEE International Conference - Volume 05*, ICASSP '99, pages 2443–2446, Washington, DC, USA, 1999. IEEE Computer Society. ISBN 0-7803-5041-3. doi: http://dx.doi.org/10.1109/ICASSP.1999.760624. URL http://dx.doi.org/10.1109/ICASSP.1999.760624.

[13] D. Le Gall. Mpeg: A video compression standard for multimedia applications. *Communications of the ACM*, pages 46–58, 1991.

[14] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. Springer, corrected edition, July 2003. ISBN 0387952845. URL http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20.

[15] K. KreutzDelgado, J. F. Murray, B. D. Rao, K. Engan, T. W. Lee, and T. J. Sejnowski. Dictionary learning algorithms for sparse representation. *Neural Computation*, 15:349–396, 2003.

[16] S. G. Mallat and Zhifeng Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41(12):3397–3415, Dec 1993. ISSN 1053587X. doi: 10.1109/78.258082. URL http://dx.doi.org/10.1109/78.258082.

[17] Stphane Mallat. *A Wavelet Tour of Signal Processing, Third Edition: The Sparse Way*. Academic Press, 2008. ISBN 0123743702, 9780123743701.

[18] Ralph Neff and Avideh Zakhor. Very low bit rate video coding based on matching pursuits, 1997.

[19] B. A. Olshausen and D. J. Field. Natural image statistics and efficient coding*. *Network*, 7(2):333–339, May 1996. ISSN 0954-898X. URL http://view.ncbi.nlm.nih.gov/pubmed/16754394.

[20] Y. Pati, R. Rezaiifar, and P. Krishnaprasad. Orthogonal Matching Pursuit: Recursive Function Approximation with Applications to Wavelet Decomposition, 1993. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.55.1254.

[21] Tomaso Poggio and Federico Girosi. A sparse representation for function approximation. *NEURAL COMPUTATION*, 10:1445–1454, 1998.

[22] John Wright, Allen Y. Yang, Arvind Ganesh, Shankar S. Sastry, and Yi Ma. Robust Face Recognition via Sparse Representation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(2):210–227, February 2009. ISSN 0162-8828. doi: 10.1109/TPAMI.2008.79. URL http://dx.doi.org/10.1109/TPAMI.2008.79.

[23] Allen Y. Yang, John Wright, Student Member, Yi Ma, Senior Member, and S. Shankar Sastry. Feature selection in face recognition: A sparse representation perspective. Technical report, 2007.