Aus dem Institut für Neuro- und Bioinformatik
der Universität zu Lübeck
Direktor: Prof. Dr. Thomas Martinetz

Soft-competitive learning of sparse data models

Inauguraldissertation
zur
Erlangung der Doktorwürde
der Universität zu Lübeck

Aus der Sektion Informatik/Technik

vorgelegt von

Kai Labusch

aus Lehrte

Lübeck 2011

Kai Labusch

1. Berichterstatter:	**Prof. Dr. rer. nat. Thomas Martinetz**
2. Berichterstatter:	**Prof. Dr. Ing. Alfred Mertins**
Tag der mündlichen Prüfung:	**30.9.2011**
Zum Druck genehmigt:	**Lübeck, den 5.10.2011**

Meinen Eltern

Zusammenfassung

Das Gehirn kann viele Problemstellungen meistern, die heutzutage noch nicht von maschinellen Lernmethoden bewältigt werden können. Ein vielversprechender Ansatz, um Forschritte in der maschinellen Informationsverarbeitung zu erzielen, ist, die Informationsverarbeitungsprozesse des Gehirns zu identifizieren und die gefundenen Mechanismen nachzuahmen.

Beobachtungen der Neurowissenschaften stützen die Hypothese, daß das Gehirn spärliche Kodierungen für die interne Repräsentation von Reizen verwendet. Lineare generative Modelle mit Spärlichkeitsrandbedingungen sind eine mathematische Abstraktion dieses Kodierungsprinzips. In vielen Problemstellungen, für deren Lösung lineare generative Modelle verwendet werden können, möchte man die Modellparameter aus gegebenen Daten erlernen, um das Modell optimal an die Aufgabenstellung anzupassen.

Im Rahmen dieser Arbeit wurden unüberwachte Lernmethoden zur Bestimmung optimaler Parameter eines linearen generativen Modells entwickelt: Sparse Coding Neural Gas (SCNG) und Neural Gas for Dictionary Learning (NGDL). Zusätzlich wurden verschiedene Anwendungen linearer generativer Modelle betrachtet: Datenrepräsentation, Bildrekonstruktion, Bilddekonvolution, blinde Quellentrennung und Merkmalsextraktion.

Anhand von Experimenten auf synthetischen Daten, die durch vorgegebene lineare Modelle erzeugt wurden deren Eigenschaften bekannt sind, konnte gezeigt werden, daß SCNG und NGDL die Modellparameter auf Basis der Trainingsdaten identifizieren können und dabei eine Verbesserung gegenüber dem Stand der Technik darstellen [1,3,6,9].

Es wurde ferner gezeigt, daß die hier vorgestellten unüberwachten Lernmethoden eine verbesserte Bildrekonstruktion und Bilddekonvolution ermöglichen [1,2,7]. In Bildrekonstruktionsexperimenten konnte gezeigt werden, daß bei begrenzter Anzahl an vorhandenen Trainingsdaten und begrenzter Lernzeit die Modellparameter, die durch NGDL gelernt wurden, eine bessere Bildrekonstruktion erlauben, als Modellparameter, die mit dem bisherigen Stand der Technik gelernt wurden [2,7]. In

Bilddekonvolutionsexperimenten konnte gezeigt werden, daß die Modellparameter an unterschiedliche Bildklassen adaptiert werden können [1].

Untersuchungen zu SCNG haben gezeigt, daß diese Methode die blinde Trennung linear gemischter Signale ermöglicht, insbesondere auch in übervollständigen Situationen, d.h., in Fällen, in denen mehr Quellensignale als observierte Mischsignale vorhanden sind [4,10]. Weiterhin wurde gezeigt, daß unter gewissen Vorbedingungen die Quellentrennung auch möglich ist, wenn die Mischmatrix zeitlicher Variation unterliegt [8].

Ferner wurde im Rahmen dieser Arbeit ein Ansatz für die Extraktion von Merkmalen zur Lösung bildbasierter Klassifikationsprobleme entwickelt [5,11]. Der Merkmalsextraktionsansatz wurde auf einem Benchmark-Datensatz für die Klassifikation handgeschriebener Ziffern evaluiert. Es konnte gezeigt werden, daß eine signifikante Verbesserung der Klassifikationsleistung einer Support-Vektor-Maschine erreicht werden kann. Die erreichte Klassifikationsleistung entspricht dem aktuellen Stand der Technik bei der Erkennung handgeschriebener Ziffern, wobei der hier vorgestellte Ansatz leicht für andere visuelle Klassifikationsprobleme verwendet werden kann.

Leistungsfähige Methoden für die Anpassung spärlicher linearer generativer Modelle an gegebene Daten können auf weitere Anwendungsprobleme übertragen werden und lassen auch dort interessante und vielversprechende Lösungsansätze erwarten.

Journalartikel:

[1] Kai Labusch, Erhardt Barth, and Thomas Martinetz. Robust and Fast Learning of Sparse Codes with Stochastic Gradient Descent. IEEE Transactions on Selected Topics in Signal Processing, 5(5):1048-1060, 2011.

[2] Kai Labusch, Erhardt Barth, and Thomas Martinetz. Soft-competitive Learning of Sparse Codes and its Application to Image Reconstruction. Neurocomputing, 74(9):1418-1428, 2011.

[3] Kai Labusch, Erhardt Barth, and Thomas Martinetz. Sparse Coding Neural Gas: Learning of Overcomplete Data Representations. Neurocomputing, 72(7-9):1547-1555, 2009.

[4] Kai Labusch, Erhardt Barth, and Thomas Martinetz. Demixing Jazz-Music: Sparse Coding Neural Gas for the Separation of Noisy Overcomplete Sources. Neural Network World, 19(5):561-579, 2009.

[5] Kai Labusch, Erhardt Barth, and Thomas Martinetz. Simple Method for High-Performance Digit Recognition Based on Sparse Coding. IEEE Transactions on Neural Networks, 19(11):1985-1989, 2008.

Konferenzbeiträge:

[6] Kai Labusch, Erhardt Barth, and Thomas Martinetz. Bag of Pursuits and Neural Gas for Improved Sparse Coding. In Gilbert Saporta, editor, Proceedings of the 19th International Conference on Computational Statistics, pages 327-336. Springer, 2010.

[7] Kai Labusch and Thomas Martinetz. Learning Sparse Codes for Image Reconstruction. In Michel Verleysen, editor, Proceedings of the 18th European Symposium on Artificial Neural Networks, pages 241-246. D-Side Publishers, 2010.

[8] Kai Labusch, Erhardt Barth, and Thomas Martinetz. Approaching the Time Dependent Cocktail Party Problem with Online Sparse Coding Neural Gas. In J.C. Principe and R. Miikkulainen, editors, Advances in Self-Organizing Maps - WSOM 2009, 7th International Workshop, St. Augustine, Fl, USA, June 2009, volume 5629 of Lecture Notes in Computer Science, pages 145-153. Springer, 2009.

[9] Kai Labusch, Erhardt Barth, and Thomas Martinetz. Learning data representations with sparse coding neural gas. In Michel Verleysen, editor, Proceedings of the 16th European Symposium on Artificial Neural Networks, pages 233-238. D-Side Publishers, 2008.

[10] Kai Labusch, Erhardt Barth, and Thomas Martinetz. Sparse Coding Neural Gas for the Separation of Noisy Overcomplete Sources. In Vera Kurková, Roman Neruda, and Jan Koutník, editors, Artificial Neural Networks - ICANN 2008, 18th International Conference, Prague, Czech Republic, September 3-6, 2008, Proceedings, Part II, volume 5163 of Lecture Notes in Computer Science, pages 788-797. Springer, 2008.

[11] Kai Labusch, Udo Siewert, Thomas Martinetz, and Erhardt Barth. Learning optimal features for visual pattern recognition. In Bernice E. Rogowitz, Thrasyvoulos N. Pappas, and Scott J. Daly, editors, Human Vision and Electronic Imaging XII, volume 6492. Proceedings of SPIE, 2007.

Contents

1 Introduction

By three methods we may learn wisdom: First, by reflection, which is noblest; Second, by imitation, which is easiest; and third by experience, which is the bitterest.

(Confucius)

In the scholastic tradition of mediaeval europe the principle of deduction dominated scientific reasoning for several centuries. The majority of scientists at that time were convinced that findings can only be obtained from the application of given axioms to a particular question. There was a controversial debate if the principle of induction, i.e., to derive a general rule from a particular observation, is a valid scientific approach. Around the 17th century the situation changed when more and more scientists such as for instance Galileo Galilei, Francis Bacon, or David Hume began to support the empirical approach and the principle of induction in science.

Today, due to their success, the empirical approach and the principle of induction are predominant. Researchers on the quest for new findings apply all sorts of data analysis techniques to measurement data. In politics and economics, it is common sense that decisions have to be supported by "objective" numbers that are derived from measurements such as opinion polls, accounting data, or goverment statistics. The amount of data that is analysed by the use of machine learning techniques is growing fast. Digital devices such as cameras, microphones, or GPS-receivers are ubiquitous and the internet enables us to efficiently access, distribute, and gather information. Social networks offer access to behavioural patterns and social relationships of a large number of persons. The collection of customer information is part of the business of many companies.

To perform specific observations in order to derive general postulations based on information that has been extracted from the observations by some formula or data model corresponds to the definition of categories of objects that can be described in a particular way. If only a subset of all possible categorizations is considered, a bias is inevitably introduced. Hence the question arises, if unbiased induction can be performed simply by consideration of all categorizatons that are possible. This question has been extensively studied by Satosi Watanabe. His "ugly duckling theorem" states that all objects become equally similar or dissimilar in this case [Watanbe, 1969]. Hence, induction is impossible without bias. A "natural" or "objective" answer to the question what is important cannot be given. In the end, it is a descision that is made by men which determines the categories that are considered to be important.

Often, scientists use classical statistical features, e.g., means, variances, or higher order moments in order to extract information from a set of observations. A huge library of scientific literature offers a broad range of more sophisticated methods that enable us to study various aspects of a given set of measurements. In many cases, users do not provide a justification, why a certain method has been preferred over other possibilities. Why does one prefer the mean over the median? Why does one perform a principal component analysis in order to determine directions of large variance that are pairwise orthogonal instead of considering a different set of directions? Of course, the availability of efficient algorithms guides the choice of the user.

A principle of science that has been proven to be very useful is the law of parsimony, i.e., Ockham's Razor. It is often used in order to justify the choice of certain features or models. According to this principle, among all models that can explain an observation, one should choose the most simple one. The simpler a model or feature is, the less bias is introduced by using it. However, what is simple in the language and axiomatic framework of men might look complicated from a different point of view.

Another factor that strongly influences decisions of scientists is experience. Those approaches that have been successfully employed in the past are likely to be considered as a possible solution in the future, i.e., often the implicit reason to prefer the mean over the median is that it performed well the last time. The more diverse the set of tasks is that have been tackled successfully by a solution strategy, the more confidence one has that it will perform well on a new problem.

During evolution, the information processing strategies of the brain have been proven to be competitive and applicable to a broad range of problems. Still many simple cognitive tasks that can be easily performed by humans or animals are out of reach of todays machine learning methods. An example of such a simple task is the ability of humans to concentrate on a single voice out of several voices in a conversation and to follow that voice even if background noise is present.

By identification and imitation of the strategies of information processing that are used by the brain scientists might be able to benefit from the experience of nature. The aim of this work is to make some progress in this direction. It studies a mathematical abstraction of a principle of information processing that recently has attracted a number of researchers due to some evidence that this principle is also implemented in the brain: the principle of sparse coding.

1.1 Sparse Coding and the Brain

One of the classical approaches that was used in order to gain insight into the information processing principles of the brain are electro-physiological measurements. Microelectrodes were placed in the brain tissue, then a stimulus, e.g., visual, accustical, or other was presented and the firing rate of the cells close to the electrodes were measured [Kuffler, 1953, Hubel, 1957].

Kuffler employed this method in order to study the properties of ganglion cells in the cats retina. The stimulus was a spot of light that was placed on the receptive field of a ganglion cell. In case of the ganglion cells he found the so-called center on/off cells whose receptive field can be devided in a concentric excitatory region where the firing rate of the cell increases if a spot of light is placed on it and a concentric inhibitory region that causes the firing rate of the cell to decrease if light falls on it [Kuffler, 1953].

Later, Hubel and Wiesel report on so-called simple [Hubel and Wiesel, 1959] and complex cells [Hubel and Wiesel, 1962] in the cats striate cortex. The receptive fields of the simple cells also can be divided in distinct excitatory and inhibitory regions. In contrast to the center on/off cells these regions are not concentric but possess a parallel arangement such that the firing of the cell can be triggered by a slit of light that has the width and orientation of the excitatory regions and is placed on their position whereas light that falls on the inhibitory regions causes a decrease of the firing rate [Hubel and Wiesel, 1959].

The complex cells showed a much more diverse behaviour. Their receptive field could not be clearly divided in distinct excitatory and inhibitory regions. For example, among them there was a class of cells whose most effective stimuli were vertically oriented edges that could be shifted horizontally over an unusually large region whereas a slight variation of the orientation stopped the firing of the cell. These cells did not respond to slits of light whether narrow or wide. Furthermore, it was crucial whether the bright area was to the left or to the right [Hubel and Wiesel, 1962]. A few years later the so-called end-stopped or hyper-complex cells were discovered that respond to termination of bars of light [Hubel and Wiesel, 1965]. While the behaviour of center on/off cells and simple cells can be described by a linear summation over the input, the behaviour of complex and hyper-complex cells cannot be easily explained by a linear model [Zetzsche and Barth, 1990]. Olshausen and Field [2004] argue that potentially only a small amount of all properties that might be observable in the visual cortex ($\approx 15\%$) have been properly documented by the scientific community, though the visual cortex is a subject of research since decades.

A long-standing hypothesis was that the simple cells remove redundancy from the input by performing a decorrelation of the input signal [Daugman, 1989]. Though this hypothesis explains the orientation and scale selectivity of the simple cells it does not explain their spatial localization since decorrelation can be also achieved by non-localized linear transformations such as the Fourier transformation or principle component analysis. It also has been argued that the spatial redundancies of natural images cannot be removed by decorrelation since they cannot be quantified by second-order statistics but only by higher-order statistics [Field, 1989, Zetzsche et al., 1993]. Furthermore, decorrelation can be performed by a linear mapping and does not explain the broad range of non-linear behaviour that has been observed in the visual cortex as for instance in case of complex and hyper-complex cells.

A hypothesis that could be used to derive the main properties of simple cells, e.g., spatial localization, ortientation selectivity, and selectivity with respect to different scales from natural image data was proposed by Olshausen and Field [Field, 1994, Olshausen and Field, 1995, 1996a,b]. Their assumption is that the output of the visual system corresponds to the hidden variables $\mathbf{a}$ of a linear generative model of natural images where each natural image $\mathbf{x} \in \mathbb{R}^N$ is obtained from a linear combination of a basis matrix C:

$$\mathbf{x} = C\mathbf{a} + \boldsymbol{\epsilon}, \quad C \in \mathbb{R}^{M\times N}. \tag{1.1}$$

In the probabilistic framework that they use, the hidden coefficients $\mathbf{a}$ stem from a prior density $P(\mathbf{a})$. This joint density is assumed to be a factorial distribution, i.e.,

$$P(\mathbf{a}) = \prod_{i=1}^{M} P((\mathbf{a})_i) \ . \tag{1.2}$$

The coefficients are sparse since their marginal distributions $P((\mathbf{a})_i)$ are considered to be leptokurtic. The residual $\boldsymbol{\epsilon}$ is assumed to be Gaussian [Olshausen and Field, 1997]. The model parameters C are determined by unsupervised learning such that the probability of obtaining the training data $\mathbf{x}_1, \dots, \mathbf{x}_L$, i.e., the data likelihood, is maximized:

$$\max_{C} P(\mathbf{x}_1, \dots, \mathbf{x}_L | C) \ . \tag{1.3}$$

Olshausen and Field proposed the Sparsenet algorithm for this task [Olshausen and Field, 1996b, 1997] (for details see Section 4.7) and have shown that the receptive fields that correspond to the basis functions C that are learned using the the Sparsenet algorithm on natural images of size 8×8 pixels possess the properties of simple cells that have been observed in the experiments mentioned before [Olshausen and Field, 1996b] . It is important to note that though the generative model (1.1) is linear, the mapping from the image $\mathbf{x}$ to its representation $\mathbf{a}$ that is implemented in the model is not linear. Given a basis C that has been learned, the coefficients are obtained by the maximization of the data likelihood:

$$\max_{\mathbf{a}} P(\mathbf{x}|C) \ . \tag{1.4}$$

Since Olshausen and Field explicitly consider overcomplete settings, i.e., $M \gg N$ [Olshausen and Field, 1997] this cannot be solved by simply inverting C but leads to a non-linear optimization problem. Even in case of $M = N$ due the prior of $\mathbf{a}$ which is non-uniform $\mathbf{a}$ has to be determined by a non-linear optimization process. Hence, this model theoretically offers enough degrees of freedom to also account for non-linear observations in the visual cortex. Due to the assumption (1.2) that the prior density of the coefficients is a factorial distribution, the new representations, i.e., the hidden variables $\mathbf{a}$, are statistically independent which implicates decorrelation but also extends to higher-order statistics. This shows that the probabilistic generative model of the Sparsenet algorithm, i.e., (1.1), can also be interpreted in the framework of independent component analysis (ICA). In Chapter 6, we show that the Olshausen and Field model can be successfully employed in order to built a very well performing technical solution for handwritten digit recognition [Labusch et al., 2008c].

ICA [Jutten and Herault, 1991, Bell and Sejnowski, 1995, Hyvärinen and Oja, 1997] is a another branch of research that provides an instructive view on the information processing principles that might be used in the brain. The model of Olshausen and Field, which can be understood as an ICA model [Kreutz-Delgado and Rao, 1999], assumes that the distributions of the hidden coefficients of the model are leptokurtic and that the hidden coefficients are stochastically independent. Another category of ICA approaches has been proposed which relies on weaker assumptions on the properties of the distributions of the hidden coefficients. Similar to the previously discussed approach of Olshausen and Field, in the ICA model of Hyvärinen and Oja [1997] a probabilistic setting is considered. In this setting an image $\mathbf{x} \in \mathbb{R}^N$ is obtained from a linear mixture of some hidden variables $\mathbf{a}$:

$$\mathbf{x} = C\mathbf{a}, \quad C \in \mathbb{R}^{N \times N}, \mathbf{a} \in \mathbb{R}^N. \tag{1.5}$$

Their sole hypothesis on the hidden variables $\mathbf{a}$ is that they are statistically independent, i.e.,

$$P(\mathbf{a}) = \prod_{i=1}^{M} P((\mathbf{a})_i) , \tag{1.6}$$

and that the marginal distributions $P((\mathbf{a})_i)$ are non-Gaussian. The distribution of a random variable that is obtained from a linear mixture of a number of non-Gaussian random variables tends towards a Gaussian distribution. The central limit theorem describes this behaviour for statistically independent random variables that are identically distributed and have finite means and variances (as discussed in detail for instance in [Ross, 2002]). In practice, often the Gaussianity of a linear mixture of a number of non-Gaussian random variables tends to be larger than the Gaussianity of the original random variables even if these original variables do not possess all poperties that are required by the central limit theorem, for instance, if the original random variables are not identically distributed. Due to this observation, it has been suggested [Hyvärinen and Oja, 1997] (see Section 5.1 for details) to estimate the mixing matrix C by the maximization of the non-Gaussianity of the resulting hidden variables $\mathbf{a}$. In some cases the maximization of non-Gaussianity is equivalent to the sparsification of the hidden variables $\mathbf{a}$. In these cases the basis that is obtained from Sparsenet and the mixing matrix that is obtained from these ICA methods are quite similar. ICA based on maximization of non-Gaussianity has been applied to natural images yielding receptive fields that are similar to the simple cells in the visual cortex [Hateren and Schaaf, 1998]. However, in their original form these approaches are limited to linear mappings. Hence, they do not offer a possible explanation for non-linear receptive fields.

Apart from observations in the visual cortex there has been further evidence that sparse representations might be used in other parts of the brain for instance in the auditory cortex [Hromadka and Zador, 2008] or the olfactory bulb of mice [Rinberg et al., 2006] and locusts [Jortner et al., 2007]. However, sparseness is not only considered to be a principle that is used in the first stages of signal processing of the brain but is also supposed to be present in higher level functions of the brain. Recently, the representation of objects in the medial temporal lobe (MTL) of humans has been studied by electrode measurements [Quian Quiroga et al., 2005, 2008]. This has been possible since the treatement of certain epilepsy patients requires the implantation of electrodes in this region of the brain in order to localize seizures in preparation of curative resection. In experiments that have been done with these patients, it could be shown that in the MTL there are neurons that show a very specific response to certain objects or persons such as actors or buildings which correspond rather to abstract concepts or perceptions and are highly invariant. For instance there are neurons that respond to the presentation of TV-actors that play in the same TV-series [Quian Quiroga et al., 2005, 2008]. It has been argued that this is a hint for a sparse representation in the brain, since in case of a “grandmother-cell” like representation that uses a single cell in order to represent a person, an object, or a concept, it would be very unlikely to find such a cell in the experiments [Quian Quiroga et al., 2008]. On the other hand, a densely distributed code would be contradictory to the very rare firing of neurons that has been observed in various experiments in this and other regions of the brain [Földiák and Young, 1998, Shoham et al., 2006]. Furthermore Valiant [2006] argues that the amount of time in which the brain solves certain tasks, the mean synaptic connection strength, and metabolic constraints such as wiring lengths and energetic constraints limit the number of neurons that are active in order to solve a particular task. His quantitative theory of neural computation predicts a neural representation that could be interpreted as a sparse code that neither is a “grandmother cell” code nor a dense distributed code.

1.2 Soft-competitive sparse coding

In the past, many learning methods have been proposed that can be used to determine an optimal basis or dictionary, i.e., the parameter C, in a linear generative model such as (1.1). In Chapter 4, we review the Method of Optimal Directions (MOD) [Engan et al., 1999], the method proposed in [Lee et al., 2007] which is quite similar to MOD but based on the dual of a slightly different optimization problem, the K-SVD algorithm [Aharon et al., 2006], and the Sparsenet algorithm [Olshausen and Field, 1995, 1996b].

A common problem of all these methods is that in order to perform an update step with respect to the dictionary, a fixed configuration of the hidden parameters of the model, i.e., the coefficients $\mathbf{a}$, has to be assumed. In the probabilistic interpretation that has been suggested for some of the algorithms, due to this approach, the maximization of the data-likelihood of the generative model, i.e., (1.3), is replaced with the maximization of a rather coarse approximation of the data-likelihood. In this approximation the maximal value of the likehood function is optimized instead of the actual likelihood, since a maximization of the actual likelihood would involve an intractable integration over the hidden parameters of the model. This aspect is discussed in more detail in Section 4.7.

In contrast to these approaches, we suggest to use many configurations of the hidden parameters instead of only a single one, in order to learn the dictionary. In a probabilistic interpretation, this corresponds to a better approximation of the data-likelihood that is considered in the optimization process by the use of many configurations of the hidden parameters in each learning step where each used configuration has a high likelihood according to the probabilistic generative model.

We use a mathematical abstraction of sparse coding that is not based on a certain prior density of the coefficients, but that relies on the hypothesis that each given observation $X = (\mathbf{x}_1, \ldots, \mathbf{x}_L)$, $\mathbf{x}_i \in \mathbb{R}^N$ can overall be generated as a linear combination of a few elements from an unknown dictionary $C = (\mathbf{c}_1, \ldots, \mathbf{c}_M)$, $\mathbf{c}_j \in \mathbb{R}^N$

$$\mathbf{x}_i = C\mathbf{a}_i + \boldsymbol{\epsilon}_i. \tag{1.7}$$

$\boldsymbol{\epsilon}_i$ and $\|\boldsymbol{\epsilon}_i\|_2$ are termed residual and representation error of sample $\mathbf{x}_i$. The requirement that only a few elements of C are used in order to generate a given observation $\mathbf{x}_i$ means that the coefficients $\mathbf{a}_i$ are sparse, i.e., they contain few non-zero entries compared to the dimensionality of the observations:

$$\|\mathbf{a}_i\|_0 \ll N \; . \tag{1.8}$$

$\|\mathbf{a}\|_0$ denotes the number of non-zero coefficients in $\mathbf{a}$. The number of dictionary elements M is a free model parameter. In case of overcomplete dictionaries, $M > N$ holds. Since in this case the number of columns of the matrix C, i.e., M, is larger than the dimension of the observations, i.e., N, one has to introduce constraints on the dictionary C and/or the coefficients $\mathbf{a}_i$ in order to obtain a well-defined representation. This can be achieved for instance by imposing sparsity constraints on the coefficients. See also Chapter 2.3 for results on the stability and identifiability of the representation (1.7). Apart from sparsity, we do not impose any further constraints on the dictionary or the coefficients.

1.2.1 Contributions of this thesis

Within this work, we propose soft-competitive unsupervised learning algorithms in order to tackle two different constrained optimization problems. Either we aim to minimize the mean of the squared representation error and constrain the coefficients of the dictionary such that the maximum number of non-zero entries of each coefficient vector $\mathbf{a}_i$ is upper bounded:

$$\min_C \frac{1}{L} \sum_{i=1}^{L} \|\mathbf{x}_i - C\mathbf{a}_i\|_2^2 \quad \text{subject to} \quad \|\mathbf{a}_i\|_0 \leq k \tag{1.9}$$

or we aim to minimize the mean number of non-zero entries of the coefficient vectors such that the representation error of each given data sample is bounded by some accuracy δ:

$$\min_C \frac{1}{L} \sum_{i=1}^{L} \|\mathbf{a}_i\|_0 \quad \text{subject to} \quad \|\mathbf{x}_i - C\mathbf{a}_i\|_2^2 \leq \delta \; . \tag{1.10}$$

In [Labusch et al., 2008a, 2009a] we have introduced the Sparse Coding Neural Gas algorithm (SCNG) which can be applied to (1.9). The SCNG algorithm is first introduced in Chapter 4.5 where it is also shown that it can be used in order to learn a dictionary from natural image data that possesses the properties of spatial localization, ortientation selectivity, and selectivity with respect to different scales.

It has been already mentioned that ICA and linear generative models are closely related. Indeed, any algorithm that determines the basis or dictionary of a linear generative model using a regularization term for the hidden parameters that obeys the property of separability can be understood as an ICA method [Kreutz-Delgado and Rao, 1999]. This aspect is also discussed in Section 5.1. In a similar way the SCNG algorithm is closely related to the domain of blind source separation. As it was first suggested in [Labusch et al., 2008b, 2009c] and [Labusch et al., 2009b], a slightly modified SCNG can be used to approach the optimization problem (1.10) which makes it possible to apply it to the problem of overcomplete blind source separation under the presence of noise and to the time dependent cocktail party problem in a difficult overcomplete setting which is discussed in Section 5.2 and 5.3.

The SCNG algorithm can be understood as a sparse approximation method, based on Optimized Orthogonal Matching Pursuit (OOMP), that modifies the dictionary during the pursuit. Therefore it is not possible to use an arbitrary approximation method for the coefficients in the learning process. In order to obtain a solution strategy for (1.9) that does not have this disadvantage, we have introduced a hard-competitive and soft-competitive stochastic gradient method [Labusch and Martinetz, 2010, Labusch et al., 2010] which we term Neural Gas for Dictionary Learning (NGDL) that can be combined with an arbitrary approximation method for the coefficients. NGDL provides significant performance improvements compared to MOD and K-SVD in terms of reconstruction of the underlying dictionary and minimization of the mean representation error as can be seen from the experiments that are presented in Section 4.6.5. The improvements in comparison to other state-of-the-art methods can also be seen in applications such as image reconstruction and deconvolution [Labusch and Martinetz, 2010, Labusch et al., 2011a,b]. Experiments that support these claims are presented in Chapter 4.6.7 and Chapter 4.6.8. Since NGDL can be combined with an arbitrary approximation method for the hidden parameters of the model, it is applicable whenever a probabilistic linear generative model is considered where the additive noise is assumed to be Gaussian. This means that it can also be applied if for instance a factorizable Laplacian prior is used for the hidden parameters. Hence, it also can be seen as a soft-competitive approach to noisy overcomplete ICA.

However, in order to employ NGDL for dictionary learning or ICA, one needs a sparse approximation method that determines many good configurations of the hidden parameters of the model. In the past, a number of approximation methods have been proposed that can be applied to the following NP-hard combinatorial optimization problem [Davis et al., 1997]:

$$\min_{\mathbf{a}} \|\mathbf{x}_i - C\mathbf{a}\|_2^2 \quad \text{subject to} \quad \|\mathbf{a}\|_0 \leq k \tag{1.11}$$

Here, C is a given fixed dictionary and k is a user-defined parameter. Among them there are greedy methods such as Matching Pursuit(MP) [S.Mallat and Zhang, 1993], Orthogonal Matching Pursuit(OMP) [Pati et al., 1993], and Optimized Orthogonal Matching Pursuit(OOMP) [Rebollo-Neira and Lowe, 2002]. Yet, all these sparse approximation methods provide only a single approximative solution for (1.11) and are therefore not applicable for soft-competitive learning with NGDL. In [Labusch and Martinetz, 2010] and [Labusch et al., 2010] we have proposed the Bag of Pursuits method (BOP) which is derived from OOMP and provides many good solutions for (1.11), which enables us to use NGDL for dictionary learning. The BOP method as well as other sparse approximation methods, i.e., greedy methods and relaxation methods, are discussed in Chapter 2.

2 Sparse approximation

Linear generative models as (1.1) can be used in a broad range of applications. For instance, they have been used for compression [Chang et al., 2000], reconstruction [Mairal et al., 2008], deconvolution [Figueiredo et al., 2007], denoising [Hoyer and Oja, 2000, Elad and Aharon, 2006], blind source separation [Lee et al., 1999, Zibulevsky and Pearlmutter, 2001], or feature extraction [Lee et al., 2000, Hyvärinen and Hoyer, 2000, Labusch et al., 2007, 2008c] tasks. In these applications, one is interested in the configuration of the hidden parameters of the model for given observations, once the dictionary C that is used in the model has been obtained either from an analytic framework (see Chapter 3) or from a dictionary learning method (see Chapter 4). Furthermore, most of the dictionary learning methods require to determine the coefficients of the dictionary elements during the learning process.

To determine the configuration of the hidden parameters that is optimal according to some criterion, yields a sparse approximation problem, if sparsity constraints are imposed on the hidden coefficients of the dictionary. This chapter discusses two categories of sparse approximation methods.

The first category, the greedy methods that are discussed in Section 2.1, either tackle the problem of finding the best k-term approximation of a given sample $\mathbf{x} \in \mathbb{R}^N$ in terms of a given dictionary $C \in \mathbb{R}^{N \times M}$

$$\min_{\mathbf{a}} \|\mathbf{x} - C\mathbf{a}\|_2^2 \quad \text{subject to} \quad \|\mathbf{a}\|_0 \leq k \tag{2.1}$$

or the problem of finding the sparsest representation of a given sample $\mathbf{x}$ under the constraint that the representation error is at most δ

$$\min_{\mathbf{a}} \|\mathbf{a}\|_0 \quad \text{subject to} \quad \|\mathbf{x} - C\mathbf{a}\|_2^2 \leq \delta \; . \tag{2.2}$$

Both (2.1) and (2.2) are NP-hard optimization tasks [Davis et al., 1997]. The greedy methods provide an approximative solution of these NP-hard tasks by an iterative construction of the sample $\mathbf{x}$ out of the elements of the dictionary C. If (2.1) is to be solved, the greedy methods stop after exactly k iterations have been performed, while in case of (2.2) the methods proceed until the approximation error drops below the user defined accuracy δ.

The second category, the relaxation methods that are discussed in Section 2.2, can be used to solve a least squares approximation problem where a regularization term has been added that enforces sparsity on the dictionary coefficients

$$\min_{\mathbf{a}} \|\mathbf{x} - C\mathbf{a}\|_2^2 + \lambda S(\mathbf{a}) \; . \tag{2.3}$$

Often $S(\mathbf{a})$, the measure of sparsity of the coefficients, is a relaxition of the zero norm such as the L_1 norm. This can be seen as a replacement of the difficult NP-hard optimization problem by an easier optimization task whose solution is under certain conditions close to the solution of the original NP-hard optimization problem [Donoho and Elad, 2003].

It is an important question whether the representation of given data in terms of a sparse linear combination of an overcomplete dictionary is well-defined, i.e., unique. This is indeed the case under certain conditions, which is discussed in Section 2.3, where it is also discussed under which conditions approximative greedy approaches can find these representations.

2.1 Greedy methods

This section refers to a class of greedy algorithms that are called pursuit methods. They iteratively construct a given vector out of the columns of a given matrix. The zero norm of the coefficient vector is equal to the number of approximation iterations that have been performed. Here, we describe Matching Pursuit [S.Mallat and Zhang, 1993], Orthogonal Matching Pursuit [Pati et al., 1993], Optimized Orthogonal Matching Pursuit [Rebollo-Neira and Lowe, 2002] and the Bag of Pursuits [Labusch et al., 2010].

2.1.1 Matching pursuit

Matching pursuit (MP) is a very simple sparse approximation method. An advantage of MP is its computational efficiency.

Let $C\mathbf{a}^{\mathrm{MP}}$ denote the current approximation of $\mathbf{x}$ in MP, and let $\boldsymbol{\epsilon} = \mathbf{x} - C\mathbf{a}^{\mathrm{MP}}$ denote the current residual that still has to be encoded. Initially, $\mathbf{a}^{\mathrm{MP}} = 0$ and

$\boldsymbol{\epsilon} = \mathbf{x}$. MP iteratively selects k columns of C by performing the following steps:

1. Select $\mathbf{c}_{l_{\text{win}}}$ by $\mathbf{c}_{l_{\text{win}}} = \arg\max_{\mathbf{c}_l} (\mathbf{c}_l^T \boldsymbol{\epsilon})^2$
2. Set $(\mathbf{a}^{\text{MP}})_{l_{\text{win}}} = (\mathbf{a}^{\text{MP}})_{l_{\text{win}}} + (\mathbf{c}_{l_{\text{win}}}^T \boldsymbol{\epsilon})$
3. Obtain new residual $\boldsymbol{\epsilon} = \mathbf{x} - C\mathbf{a}^{\text{MP}}$
4. Continue with step 1 until k iterations have been performed or $\|\boldsymbol{\epsilon}\| \leq \delta$

Even if we perform N iterations of MP, i.e., if we select as many columns of C as there are dimensions, it is not guaranteed that we will obtain $C\mathbf{a}^{\text{MP}} = \mathbf{x}$ and $\boldsymbol{\epsilon} = 0$, though the asymptotical convergence of MP for $k \to \infty$ has been proven [S.Mallat and Zhang, 1993].

2.1.2 Orthogonal matching pursuit

Let $C\mathbf{a}^{\text{OMP}}$ denote the current approximation of $\mathbf{x}$ in Orthogonal Matching Pursuit. In contrast to MP, this approximation fulfills $C\mathbf{a}^{\text{OMP}} = \mathbf{x}$ and $\boldsymbol{\epsilon} = 0$ after $k \leq N$ iterations [Pati et al., 1993]. Let U denote the set of indices of those columns of C that already have been used during Orthogonal Matching Pursuit. The number of elements in U, i.e., $|U|$, equals the number of iterations that have been performed so far. The columns of C that are indexed by U are denoted by C^U. Initially, $\mathbf{a}^{\text{OMP}} = 0$, $\boldsymbol{\epsilon} = \mathbf{x}$ and $U = \emptyset$. OMP works as follows:

1. Select $\mathbf{c}_{l_{\text{win}}}$ by $\mathbf{c}_{l_{\text{win}}} = \arg\max_{\mathbf{c}_l, l \notin U} (\mathbf{c}_l^T \boldsymbol{\epsilon})^2$
2. Set $U = U \cup l_{\text{win}}$
3. Solve the optimization problem $\mathbf{a}^{\text{OMP}} = \arg\min_{\mathbf{a}} \|\mathbf{x} - C^U \mathbf{a}\|_2^2$
4. Obtain current residual $\boldsymbol{\epsilon} = \mathbf{x} - C\mathbf{a}^{\text{OMP}}$
5. Continue with step 1 until k iterations have been performed or $\|\boldsymbol{\epsilon}\| \leq \delta$

2.1.3 Optimized orthogonal matching pursuit

An improved variant of the OMP algorithm is Optimized Orthogonal Matching Pursuit (OOMP) [Rebollo-Neira and Lowe, 2002]. In general, the columns of C are not pairwise orthogonal. Hence, the criterion of OMP that selects the column $\mathbf{c}_{l_{\text{win}}}, l_{\text{win}} \notin U$ of C that is added to U is not optimal with respect to the minimization of the residual that is obtained after the column $\mathbf{c}_{l_{\text{win}}}$ has been added. Therefore, Optimized Orthogonal Matching Pursuit uses a selection criterion that is optimal with respect to the minimization of the norm of the residual obtained: the algorithms runs through all columns of C that have not been used so far and selects the one that yields the smallest residual. Optimized Orthogonal Matching Pursuit works as follows:

1. Select $\mathbf{c}_{l_{\text{win}}}$ such that $\mathbf{c}_{l_{\text{win}}} = \arg\min_{\mathbf{c}_l, l \notin U} \min_{\mathbf{a}} \|\mathbf{x} - C^{U \cup l}\mathbf{a}\|$
2. Set $U = U \cup l_{\text{win}}$
3. Solve the optimization problem $\mathbf{a}^{\text{OMP}} = \arg\min_{\mathbf{a}} \|\mathbf{x} - C^{U}\mathbf{a}\|_2^2$
4. Obtain current residual $\boldsymbol{\epsilon} = \mathbf{x} - C\mathbf{a}^{\text{OMP}}$
5. Continue with step 1 until k iterations have been performed or $\|\boldsymbol{\epsilon}\| \le \delta$

The selection criterion of the OOMP algorithm (step 1) involves $M - |U|$ minimization problems, one for each column of C that has not been used so far. In order to reduce the computational complexity of this step, we use an implementation of the OOMP algorithm that employs a temporary matrix R that has been orthogonalized with respect to C^U. R is obtained by removing the projection of the columns of C onto the subspace spanned by C^U from C and setting the norm of the residuals $\mathbf{r}_l$ to one. The residual $\boldsymbol{\epsilon}^U$ is obtained in the same way, i.e., the projection of $\mathbf{x}$ to the subspace spanned by C^U is removed from $\mathbf{x}$. Initially, $R = (\mathbf{r}_1, \ldots, \mathbf{r}_l, \ldots, \mathbf{r}_M) = C$ and $\boldsymbol{\epsilon}^U = \mathbf{x}$. In each iteration, the algorithm determines the column $\mathbf{r}_l$ of R with $l \notin U$ that has maximum overlap with respect to the current residual $\boldsymbol{\epsilon}^U$:

$$l_{\text{win}} = \arg\max_{l, l \notin U} (\mathbf{r}_l^T \boldsymbol{\epsilon}^U)^2 \ . \tag{2.4}$$

Then, in the construction step, the orthogonal projection with respect to $\mathbf{r}_{l_{\text{win}}}$ is removed from the columns of R and $\boldsymbol{\epsilon}^U$:

$$\mathbf{r}_l = \mathbf{r}_l - (\mathbf{r}_{l_{\text{win}}}^T \mathbf{r}_l)\mathbf{r}_{l_{\text{win}}}, \tag{2.5}$$

$$\boldsymbol{\epsilon}^U = \boldsymbol{\epsilon}^U - (\mathbf{r}_{l_{\text{win}}}^T \boldsymbol{\epsilon}^U)\mathbf{r}_{l_{\text{win}}} \ . \tag{2.6}$$

After the projection has been removed, l_{win} is added to U, i.e., $U = U \cup l_{\text{win}}$. The columns $\mathbf{r}_l$ with $l \notin U$ may be selected in the subsequent iterations of the algorithm. The norm of these columns is set to unit length. The stopping criterion is either $|U| = k$ or $\|\boldsymbol{\epsilon}\| \le \delta$. The final entries of $\mathbf{a}^{\text{OMP}}$ can be obtained by recursively collecting the contribution of each column of C during the construction process, taking into account the normalization of the columns of R in each iteration.

2.1.4 Bag of Pursuits

In order to perform soft-competitive dictionary learning with NGDL (see Section 4.6), we need a sparse approximation method that not only determines a single approximation of the best coefficients but that determines many good approximations of a given sample. Here, we describe a method that has this property and which is termed "bag of pursuits" (BOP), since it performs a sequence of optimized orthogonal matching pursuits. The number of solutions that are provided is equal to the number of pursuits that are performed. The number of pursuits is a user-defined parameter and in the following is denoted by K_{user}.

The algorithm starts with $U_n^j = \emptyset$, $R_0^j = (\mathbf{r}_1^{0,j}, \ldots, \mathbf{r}_M^{0,j}) = C$ and $\boldsymbol{\epsilon}_0^j = \mathbf{x}$. The set U_n^j contains the indices of those columns of C that have been used during the j-th pursuit with respect to $\mathbf{x}$ up to the n-th iteration. R_n^j is a temporary matrix that has been orthogonalized with respect to the columns of C that are indexed by U_n^j. $\mathbf{r}_l^{n,j}$ is the l-th column of R_n^j. $\boldsymbol{\epsilon}_n^j$ is the residual in the n-th iteration of the j-th pursuit with respect to $\mathbf{x}$.

In iteration n, the algorithm looks for that column of R_n^j whose inclusion in the linear combination leads to the smallest residual $\boldsymbol{\epsilon}_{n+1}^j$ in the next iteration of the algorithm, i.e., that has the maximum overlap with respect to the current residual. Hence, with

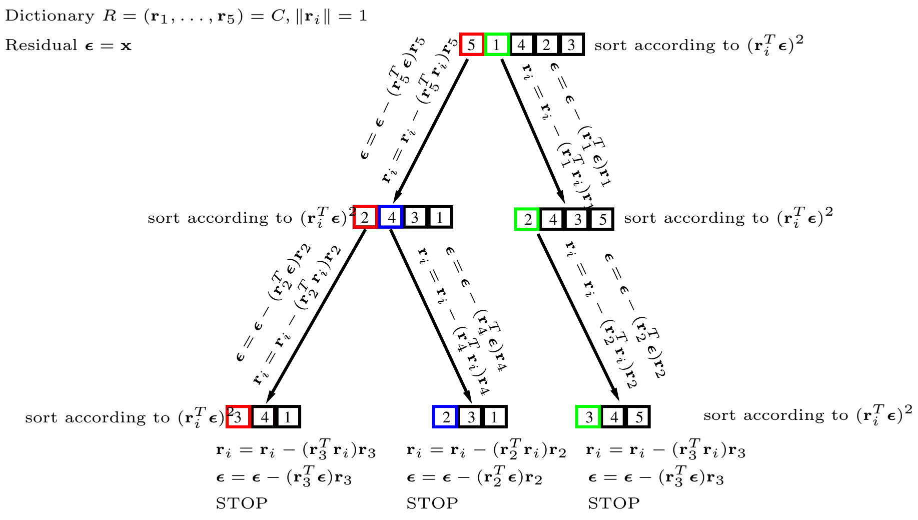
$$\mathbf{y}_n^j = \left(\frac{{\mathbf{r}_1^{n,j}}^T \boldsymbol{\epsilon}_n^j}{\|\mathbf{r}_1^{n,j}\|}, \ldots, \frac{{\mathbf{r}_l^{n,j}}^T \boldsymbol{\epsilon}_n^j}{\|\mathbf{r}_l^{n,j}\|}, \ldots, \frac{{\mathbf{r}_M^{n,j}}^T \boldsymbol{\epsilon}_n^j}{\|\mathbf{r}_M^{n,j}\|} \right) \tag{2.7}$$

it looks for $l_{\text{win}}(n,j) = \arg\max_{l, l \notin U_n^j} \left((\mathbf{y}_n^j)_l \right)^2$. Then, the orthogonal projection of R_n^j to $\mathbf{r}_{l_{\text{win}}(n,i,j)}^{n,i,j}$ is removed from R_n^j

$$R_{n+1}^j = R_n^j - \frac{\mathbf{r}_{l_{\text{win}}(n,i,j)}^{n,i,j} ({R_n^j}^T \mathbf{r}_{l_{\text{win}}(n,i,j)}^{n,i,j})^T}{{\mathbf{r}_{l_{\text{win}}(n,i,j)}^{n,i,j}}^T \mathbf{r}_{l_{\text{win}}(n,i,j)}^{n,i,j}} \,. \tag{2.8}$$

Furthermore, the orthogonal projection of $\boldsymbol{\epsilon}_n^j$ to $\mathbf{r}_{l_{\text{win}}(n,i,j)}^{n,i,j}$ is removed from $\boldsymbol{\epsilon}_n^j$

$$\boldsymbol{\epsilon}_{n+1}^j = \boldsymbol{\epsilon}_n^j - \frac{{\boldsymbol{\epsilon}_n^j}^T \mathbf{r}_{l_{\text{win}}(n,i,j)}^{n,i,j}}{{\mathbf{r}_{l_{\text{win}}(n,i,j)}^{n,i,j}}^T \mathbf{r}_{l_{\text{win}}(n,i,j)}^{n,i,j}} \mathbf{r}_{l_{\text{win}}(n,i,j)}^{n,i,j} \,. \tag{2.9}$$



Dictionary $R = (\mathbf{r}_1, \dots, \mathbf{r}_5) = C$, $\|\mathbf{r}_i\| = 1$
Residual $\boldsymbol{\epsilon} = \mathbf{x}$
5 1 4 2 3
sort according to $(\mathbf{r}_i^T \boldsymbol{\epsilon})^2$
$\boldsymbol{\epsilon} = \boldsymbol{\epsilon} - (\mathbf{r}_5^T \boldsymbol{\epsilon})\mathbf{r}_5$
$\mathbf{r}_i = \mathbf{r}_i - (\mathbf{r}_5^T \mathbf{r}_i)\mathbf{r}_5$
$\mathbf{r}_i = \mathbf{r}_i - (\mathbf{r}_1^T \mathbf{r}_i)\mathbf{r}_1$
$\boldsymbol{\epsilon} = \boldsymbol{\epsilon} - (\mathbf{r}_1^T \boldsymbol{\epsilon})\mathbf{r}_1$
sort according to $(\mathbf{r}_i^T \boldsymbol{\epsilon})^2$
2 4 3 1
2 4 3 5
sort according to $(\mathbf{r}_i^T \boldsymbol{\epsilon})^2$
$\boldsymbol{\epsilon} = \boldsymbol{\epsilon} - (\mathbf{r}_2^T \boldsymbol{\epsilon})\mathbf{r}_2$
$\mathbf{r}_i = \mathbf{r}_i - (\mathbf{r}_2^T \mathbf{r}_i)\mathbf{r}_2$
$\mathbf{r}_i = \mathbf{r}_i - (\mathbf{r}_4^T \mathbf{r}_i)\mathbf{r}_4$
$\boldsymbol{\epsilon} = \boldsymbol{\epsilon} - (\mathbf{r}_4^T \boldsymbol{\epsilon})\mathbf{r}_4$
$\mathbf{r}_i = \mathbf{r}_i - (\mathbf{r}_2^T \mathbf{r}_i)\mathbf{r}_2$
$\boldsymbol{\epsilon} = \boldsymbol{\epsilon} - (\mathbf{r}_2^T \boldsymbol{\epsilon})\mathbf{r}_2$
sort according to $(\mathbf{r}_i^T \boldsymbol{\epsilon})^2$
3 4 1
2 3 1
3 4 5
sort according to $(\mathbf{r}_i^T \boldsymbol{\epsilon})^2$
$\mathbf{r}_i = \mathbf{r}_i - (\mathbf{r}_3^T \mathbf{r}_i)\mathbf{r}_3$
$\boldsymbol{\epsilon} = \boldsymbol{\epsilon} - (\mathbf{r}_3^T \boldsymbol{\epsilon})\mathbf{r}_3$
STOP
$\mathbf{r}_i = \mathbf{r}_i - (\mathbf{r}_2^T \mathbf{r}_i)\mathbf{r}_2$
$\boldsymbol{\epsilon} = \boldsymbol{\epsilon} - (\mathbf{r}_2^T \boldsymbol{\epsilon})\mathbf{r}_2$
STOP
$\mathbf{r}_i = \mathbf{r}_i - (\mathbf{r}_3^T \mathbf{r}_i)\mathbf{r}_3$
$\boldsymbol{\epsilon} = \boldsymbol{\epsilon} - (\mathbf{r}_3^T \boldsymbol{\epsilon})\mathbf{r}_3$
STOP

Figure 2.1: The figure depicts the tree-like search procedure of the BOP method for the constraint $\|\mathbf{a}\|_0 \leq 3$. In this example $K_{\text{user}} = 3$ holds, i.e., the method determines three different solutions. The method starts by sorting the dictionary elements according to their overlap with respect to the residual (root of the tree). The dictionary element that has the largest overlap, i.e., element 5, is selected. All other dictionary elements as well as the residual are orthogonalized with respected to dictionary element 5. This procedure is repeated (elements 2 and 3 are selected) until at most three dictionaries elements have been used. Now, the second solution is determined. Among all overlaps that have been computed so far the largest one is selected (element 1 at root level). Again a sequence of orthogonalizations is performed until three dictionary elements have been used (elements 2 and 3 are selected). The third solution is obtained by repeating the entire procedure again. Solution 1: 5,2,3 Solution 2: 1,2,3 Solution 3: 5,4,2.

The algorithm stops if $\|\boldsymbol{\epsilon}^j_n\| \leq \delta$ or $n = k$. The j-th approximation of the coefficients of the best k-term approximation, i.e., $\mathbf{a}^j$, can be obtained by recursively tracking the contribution of each column of C that has been used during the iterations of pursuit j. In order to obtain a set of approximations $\mathbf{a}^1, ..., \mathbf{a}^{K_{\text{user}}}$, where K_{user} is chosen by the user, we want to conduct K_{user} matching pursuits. To obtain K_{user} different pursuits, we implement the following function:

$$Q(l,n,j) = \begin{cases} 0 : & \text{If there is no pursuit (among all pursuits that have been performed with respect to } \mathbf{x}\text{) that is equal to the } j\text{-th pursuit up to the } n\text{-th iteration, where in that iteration column } l \text{ has been selected} \\ 1 : & \text{else .} \end{cases} \tag{2.10}$$

Then, while a pursuit is performed, we track all overlaps $\mathbf{y}^j_n$ that have been computed during that pursuit. For instance if $\mathbf{a}^1$ has been determined, we have $\mathbf{y}^1_0, \dots, \mathbf{y}^1_n, \dots, \mathbf{y}^1_{s_1-1}$ where s_1 is the number of iterations of the 1st pursuit with respect to $\mathbf{x}$. In order to find $\mathbf{a}^2$, we now look for the largest overlap in the previous pursuit that has not been used so far

$$n_{\text{target}} = \arg \max_{n=0,\dots,s_1-1} \max_{l,Q(l,n,j)=0} \left((\mathbf{y}^1_n)_l\right)^2 \tag{2.11}$$

$$l_{\text{target}} = \arg \max_{l} (\mathbf{y}^1_{n_{\text{target}}})_l \, . \tag{2.12}$$

We replay the 1st pursuit up to iteration n_{target}. In that iteration, we select column l_{target} instead of the previous winner and continue with the pursuit until the stopping criterion has been reached. If m pursuits have been performed, among all previous pursuits, we look for the largest overlap that has not been used so far:

$$j_{\text{target}} = \arg \max_{j=1,\dots,m} \max_{n=0,\dots,s_j-1} \max_{l,Q(l,n,j)=0} \left((\mathbf{y}^j_n)_l\right)^2 \tag{2.13}$$

$$n_{\text{target}} = \arg \max_{n=0,\dots,s_{j_{\text{target}}}-1} \max_{l,Q(l,n,j_{\text{target}})=0} \left((\mathbf{y}^{j_{\text{target}}}_n)_l\right)^2 \tag{2.14}$$

$$l_{\text{target}} = \arg \max_{l,Q(l,n_{\text{target}},j_{\text{target}})=0} \left((\mathbf{y}^{j_{\text{target}}}_{n_{\text{target}}})_l\right)^2 \, . \tag{2.15}$$

We replay pursuit j_{target} up to iteration n_{target}. In that iteration, we select column l_{target} instead of the previous winner and continue with the pursuit until the stopping criterion has been reached. We repeat this procedure until K_{user} pursuits have been performed. A schematic view of the BOP method is shown in Figure 2.1.

2.2 Relaxation methods

In general, the optimization problems (2.1) and (2.2) that involve the zero norm as measure of sparsity are NP-hard [Davis et al., 1997]. Therefore, methods have been proposed that do not solve these optimization problems but related easier optimization problems, whose solution often is close to the solution of the primal NP-hard problems [Donoho and Elad, 2003]. These methods consider the optimization problem that is defined by (2.3) where the penalty or regularization term $S(\mathbf{a})$ is a relaxation of the zero norm. This means that it still enforces sparsity, but is not as discontinuous as the zero norm. Many choices are possible for $S(\mathbf{a})$, for instance:

$$S(\mathbf{a}) = \begin{cases} \|\mathbf{a}\|_1 \\ -e^{-\|\mathbf{a}\|_2^2} \\ \log(1+\mathbf{a}^2)\ . \end{cases} \tag{2.16}$$

If the regularization term that is used is differentiable, as for instance in case of $S(\mathbf{a}) = \log(1+\mathbf{a}^2)$, (2.3) can be approached by standard optimization methods such as gradient descent. This has been done in the Sparsenet algorithm [Olshausen and Field, 1995, 1996b]. However, if $S(\mathbf{a})$ is not convex, which is the case for $S(\mathbf{a}) = \log(1+\mathbf{a}^2)$, gradient optimization might run into local minima.

In the Basis Pursuit method [Chen et al., 1998], one chooses $S(\mathbf{a}) = \|\mathbf{a}\|_1$ and obtains as optimization task

$$\min_{\mathbf{a}} \|\mathbf{x} - C\mathbf{a}\|_2^2 + \lambda\|\mathbf{a}\|_1\ , \tag{2.17}$$

which has a target function that is convex but not differentiable. With $\hat{C} = (C, -C)$ this can be converted to the following equivalent quadratic program

$$\min_{\hat{\mathbf{a}}} \hat{\mathbf{a}}^T \hat{C}^T \hat{C} \hat{\mathbf{a}} - (2\mathbf{x}^T \hat{C} + \lambda\mathbf{1})\hat{\mathbf{a}} \quad \text{subject to } (\hat{a})_i \geq 0\ \forall i = 1, \ldots, 2M\ , \tag{2.18}$$

that can be approached with standard QP-solvers. (2.17) is closely related to the following minimization problem

$$\min_{\mathbf{a}} \|\mathbf{a}\|_1 \quad \text{subject to} \quad \| \mathbf{x} - C\mathbf{a}\|_2^2 \le \delta \; . \tag{2.19}$$

The constrained minimization problem (2.19) can be solved by an equivalent unconstrained maximization of its corresponding Lagrangian

$$\max_{\beta} \min_{\mathbf{a}} \|\mathbf{a}\|_1 + \beta(\| \mathbf{x} - C\mathbf{a}\|_2^2 - \delta) \quad \text{subject to} \quad \beta \ge 0 \tag{2.20}$$

where β is the Lagrange parameter. Suppose that the optimal value for β, i.e., β_*, is known, then (2.20) turns into

$$\min_{\mathbf{a}} \frac{1}{\beta_*} \|\mathbf{a}\|_1 + \| \mathbf{x} - C\mathbf{a}\|_2^2 \; . \tag{2.21}$$

For any choice of λ in (2.17) there is some δ in (2.19) such that $\lambda = 1/\beta_*$. For any δ in (2.19) there is some λ in (2.17) such that $\lambda = 1/\beta_*$.

A disadvantage of the Basis pursuit approach is that it leads to a quadratic minimization problem of size $2M \times 2M$, which becomes intractable for very large settings. Among the methods that are computationally more feasible and that can be used to find a solution of (2.17) (for a comprehensive review see [Bruckstein et al., 2009]) there are iteratively reweighted least squares methods [Rao et al., 2003], stepwise methods as for instance the LARS-LASSO algorithm [Osborne et al., 2000], iterative shrinkage methods [Daubechies et al., 2004], and related neuro-inspired dynamical systems [Rozell and Baraniuk, 2008].

2.3 Stability and identifiability

In this section, we briefly discuss some results on the stability and identifiability of the solution of the sparse approximization tasks (2.1) and (2.2). Here, stability is a property of the optimization problem and describes how a change of the noise level influences its domain of solutions. Identifiability describes the ability of some sparse approximation method to find such a solution within a certain accuracy. Hence, this property is only defined in connection with a specific approximation algorithm. Since in this work the OMP algorithm or one of its derivatives, e.g., OOMP or BOP, are used in order to solve sparse approximation problems, we here concentrate on results that were obtained for the OMP method.

Of course, the choice of the dictionary C has a strong influence on stability and identifiability. In particular, its mutual coherence has an impact on these properties. The mutual coherence $H(C)$ is defined as follows:

$$H(C) = \max_{1 \le i,j \le M, i \ne j} |\mathbf{c}_i^T \mathbf{c}_j| \ . \tag{2.22}$$

Identifiablility in the approximation domain: Let $\mathbf{x}^* = C\mathbf{a}^*$ be the best k-term approximation of a given sample $\mathbf{x}$ in terms of a dictionary C:

$$\mathbf{a}^* = \arg\min_{\mathbf{a}} \|\mathbf{x} - C\mathbf{a}\| \quad \text{subject to} \quad \|\mathbf{a}\|_0 \le k \ . \tag{2.23}$$

It has been shown [Tropp, 2004] that if

$$k < \frac{1}{2}\left(1 + \frac{1}{H(C)}\right) \tag{2.24}$$

holds, than OMP yields an $\mathbf{x}^{\mathrm{OMP}} = C\mathbf{a}^{\mathrm{OMP}}$ with

$$\|\mathbf{x} - \mathbf{x}^{\mathrm{OMP}}\| \le \sqrt{1 + 6k}\ \|\mathbf{x} - \mathbf{x}^*\| \ . \tag{2.25}$$

This also means that if (2.24) holds and $\mathbf{x} = \mathbf{x}^*$, OMP finds the exact solution of (2.1), i.e., it identifies the best k-term approximation of $\mathbf{x}$ that is $\mathbf{x}^*$.

Stability of the solution in the representation domain: The result on the stability of the solution in the representation domain, as it is presented in the following, is taken from [Bruckstein et al., 2009]. It was first reported in [Donoho et al., 2006]. Let $\mathbf{a}^*$ be some vector for which $\|\mathbf{x} - C\mathbf{a}^*\| \le \delta$ and

$$\|\mathbf{a}^*\|_0 < \frac{1}{2}\left(1 + \frac{1}{H(C)}\right) \tag{2.26}$$

holds. Now, let $\mathbf{a}^{**}$ be some solution of (2.2), i.e.,

$$\mathbf{a}^{**} = \arg\min_{\mathbf{a}} \|\mathbf{a}\|_0 \text{ subject to} \|\mathbf{x} - C\mathbf{a}\| \le \delta \ . \tag{2.27}$$

Then

$$\|\mathbf{a}^* - \mathbf{a}^{**}\| \le \frac{4\delta^2}{1 - H(C)(2\|\mathbf{a}^*\|_0 - 1)} \tag{2.28}$$

holds. Therefore, if (2.26) holds and $\delta = 0$, $\mathbf{a}^*$ is the unique solution of (2.2).

Identifiablility in the representation domain: We here cite a result on the identifiability of the solution of (2.2) for OMP as it is presented in [Bruckstein et al., 2009]. It was first reported in [Donoho et al., 2006].

Let $\mathbf{a}^*$ be some vector for which $\|\mathbf{x} - C\mathbf{a}^*\| \leq \delta$ and

$$\|\mathbf{a}^*\|_0 < \frac{1}{2}\left(1 + \frac{1}{H(C)}\right) - \frac{\delta}{H(C)\min_{\mathbf{a}_i^* \neq 0} |\mathbf{a}_i^*|} \tag{2.29}$$

holds. Let $\mathbf{a}^{\mathrm{OMP}}$ be an approximative solution of (2.2) that has been obtained from the OMP algorithm. Then

$$\|\mathbf{a}^* - \mathbf{a}^{\mathrm{OMP}}\| \leq \frac{\delta^2}{1 - H(C)(\|\mathbf{a}^*\|_0 - 1)} \tag{2.30}$$

holds. Furthermore, $\mathbf{a}^{\mathrm{OMP}}$ contains only non-zeros that also appear in $\mathbf{a}^*$. This also means that if (2.29) holds and $\delta = 0$, OMP identifies the exact solution of (2.2). For $\delta > 0$, local stability is given, as long as in (2.29) the absolute value of the smallest non-zero entry in $\mathbf{a}^*$, i.e., $\min_{\mathbf{a}_i^* \neq 0} |\mathbf{a}_i^*|$, is sufficiently large [Donoho et al., 2006].

3 Analytic dictionaries

In some of the experiments that are part of this work analytic data representations were used for comparison purposes. Analytic means in this case, that the dictionary that is used for the representation of given data is not obtained from a learning algorithm, but that the dictionary is defined by some mathematical framework. This chapter provides an overview over those analytic representations that have been used in the experiments. It is supposed to be a brief summary of the most important aspects, since a comprehensive discussion of the methods that are presented here would by far exceed the scope of this work. A comprehensive discussion of this domain can be found for instance in [Mallat, 2009].

3.1 Discrete Fourier Transformation

Suppose you are given a vector $\mathbf{x} \in \mathbb{R}^N$, $\mathbf{x} = (x_1, \dots, x_l, \dots, x_N)^T$. By computing its Fourier representation, it is implicitly considered to represent a discrete function $x(l)$ of period N

$$x(l) = x_{(l \ mod \ N)+1} \qquad l = -\infty, \dots, \infty \quad l \in \mathbb{Z}. \tag{3.1}$$

The set of functions

$$c_m^{\text{dft}}(l) = \cos\left(\frac{2\pi ml}{N}\right) + i \sin\left(\frac{2\pi ml}{N}\right) \ m = 0, \dots, N-1 \tag{3.2}$$

is an orthogonal basis of the space of discrete functions of period N [Mallat, 2009]. Using a matrix $C^{\text{DFT}} \in \mathbb{C}^{N \times N}$ with $C_{l,m}^{\text{DFT}} = c_{m-1}^{\text{dft}}(l-1)$ the coefficients of the discrete Fourier transformation of $\mathbf{x}(l)$ are obtained from $\mathbf{a} = C^{\text{DFT}}\mathbf{x}$. Given the Fourier coefficients $\mathbf{a}$, the vector representation of the function $x(l)$ can be reconstructed by applying the inverse discrete Fourier transformation $\mathbf{x} = \left(C^{\text{DFT}}\right)^T \mathbf{a}$. The Fourier coefficients $\mathbf{a} = (a_1, \dots, a_m, \dots, a_N)$ can be interpreted as the frequency dependent discrete function $a(m)$. The parameter l of the primal function has been replaced by the frequency parameter m of its Fourier transformation.

Even if $x(l)$ is smooth in the interval $[0, N-1]$, due to its periodicity the absolute values of the high-frequency Fourier coefficients are large, if $|x(0) - x(N-1)|$ is large. On the one hand this effect is somehow artificial, on the other hand it is even detrimental, in particular in applications such as image compression where, due to the blockwise processing of the images, N might be quite small. Therefore, in such applications often the discrete cosine transformation is used instead.

3.2 Discrete Cosine Transformation

For the discrete cosine transformation, the following set of basis functions is used:

$$c_m^{\text{dct}}(l) = f_m \sqrt{\frac{2}{N}} \cos\left(\frac{m\pi}{N}\left(l + \frac{1}{2}\right)\right) \quad m = 0, \dots, N-1 \tag{3.3}$$

with

$$f_m = \begin{cases} \frac{1}{\sqrt{2}} & : \text{ if } m = 0 \\ 1 & : \text{ else .} \end{cases} \tag{3.4}$$

Considering the matrix $C^{\text{DCT}} \in \mathbb{R}^{N \times N}$ with $C_{l,m}^{\text{DCT}} = c_{m-1}^{\text{dct}}(l-1)$, it follows from (3.3) that its columns are pairwise orthogonal and, due to the normalization factor $f_m \sqrt{\frac{2}{N}}$, the norm of each column is equal to one [Mallat, 2009]. Hence, C^{DCT} is an orthonormal basis of $\mathbb{R}^N$. Therefore, the coefficients of the discrete cosine transformation are obtained from $\mathbf{a} = C^{\text{DCT}}\mathbf{x}$. $\mathbf{x}$ can be reconstructed from the coefficients by applying the inverse discrete cosine transformation, i.e., $\mathbf{x} = \left(C^{\text{DCT}}\right)^T \mathbf{a}$. In contrast to the discrete Fourier transformation, the discrete cosine transformation does not suffer that strong from the problem of artificially introduced high-frequencies. This can be seen as follows:

Let us consider a vector $\hat{\mathbf{x}} \in \mathbb{R}^{2N}$ with

$$\hat{x}_l = \begin{cases} x_l & : \text{ if } l = 1, \dots, N \\ x_{2N-l+1} & : \text{ if } l = N+1, \dots, 2N \text{ .} \end{cases} \tag{3.5}$$

In the framework of the discrete Fourier transformation, this vector is implicitly considered as discrete function of period $2N$

$$\hat{x}(l) = \hat{x}_{(l \; mod \; 2N)+1} \qquad l = -\infty, \dots, \infty \quad l \in \mathbb{Z}. \tag{3.6}$$

This function is symmetric at $kN + (1/2)$, $k \in \mathbb{Z}$. Applying a discrete fourier transformation to this function that has been shifted by $(1/2)$ one obtains Fourier coefficients that can be used in order to reconstruct $\hat{x}(l)$ from the Fourier basis functions as follows

$$\hat{x}(l) = \sum_{m=0}^{2N-1} a_m^{\text{re}} \cos\left(\frac{2\pi m\left(l+\frac{1}{2}\right)}{2N}\right) + a_m^{\text{im}} \sin\left(\frac{2\pi m\left(l+\frac{1}{2}\right)}{2N}\right) . \tag{3.7}$$

Here a_m^{re} and a_m^{im} are the real and imaginary part of the Fourier coefficient a_m. Due to the symmetry properties of $\hat{x}(l)$, the imaginary part of the Fourier coefficients is equal to zero, i.e., $a_m^{\text{im}} = 0$, $\forall m$. Furthermore, the discrete cosine basis functions $c_m^{\text{dct}}(l)$, $m = 0, \ldots, N-1$, $l = 0, \ldots, N-1$ already provide an orthogonal basis that can represent any $\mathbf{x} \in \mathbb{R}^N$. Together with the symmetry properties of $\hat{x}(l)$ and the symmetry properties of the shifted cosine functions in (3.7), it follows that $a_m^{\text{re}} = 0$, $\forall m > N-1$. Hence, apart from normalization factors, the discrete cosine transformation can be understood as the discrete Fourier transformation of the periodic function $\hat{x}(l)$. $\hat{x}(l)$ is much smoother than $x(l)$ since $\hat{x}(0) = \hat{x}(2N-1)$ is guaranteed.

One dimensional bases such as the Fourier basis (3.2) or the cosine basis (3.3) can be extended to multiple dimensions by taking the pairwise tensor products of the one-dimensional basis vectors [Mallat, 2009]. In this work we use the two-dimensional discrete cosine transformation that uses a set of basis vectors that are products of one-dimensional discrete cosine functions

$$c_{h,v}^{\text{dct}}(l_h, l_v) = f_h f_v \frac{2}{N} \cos\left(\frac{h\pi}{D}\left(l_h + \frac{1}{2}\right)\right) \cos\left(\frac{v\pi}{D}\left(l_v + \frac{1}{2}\right)\right) \tag{3.8}$$

$$h = 0, 1, \ldots, D-1, v = 0, 1, \ldots, D-1 .$$

and provides an orthonormal basis of the image patches $\mathbf{x} \in \mathbb{R}^{D \times D}$, $D^2 = N$. Here, l_h and l_v are the horizontal and vertical position in the image. Furthermore, we use overcomplete cosine dictionaries that have $M > D^2$ many elements. Such an overcomplete cosine dictionary is obtained by using a finer sampling of the frequencies (here $\sqrt{M}$ is an integer number):

$$c^{\mathrm{dct}}_{h,v}(l_h, l_v) = f_h f_v \frac{2}{N} \cos\left(\frac{h\pi}{\sqrt{M}}\left(l_h + \frac{1}{2}\right)\right) \cos\left(\frac{v\pi}{\sqrt{M}}\left(l_v + \frac{1}{2}\right)\right) \tag{3.9}$$

$$h = 0, 1, \ldots, \sqrt{M} - 1, v = 0, 1, \ldots, \sqrt{M} - 1,\ \sqrt{M} > D\ .$$

Note that in the overcomplete case the representation in terms of the dictionary elements is not unique. Also the dictionary elements are not pairwise orthogonal any more. In order to obtain a well-defined representation constraints on the coefficients have to be imposed. The sparse approximation methods described in Chapter 2 can be used in order to determine a representation of given data in terms of such an overcomplete dictionary.

3.3 Gabor functions

In the Fourier representation the frequency resolution is maximal, whereas the dependency of an observation on its primal parameters, as for instance time or space, is completely removed. The more the basis functions of a new representation are localized in one of the domains, i.e., either frequencies or primal parameters, the higher the resolution is in that domain. High localisation is in this case equivalent to small variance. The Fourier basis functions are maximally localized in the frequency domain, since each of them represents negative and positive part of exactly one frequency. This means that the variance of the Fourier representation of the Fourier basis functions is equal to zero, whereas the variance of the Fourier basis functions in the primal parameter domain is infinite.

In order to obtain a representation that enables a parallel analysis in terms of both domains, i.e., frequencies and primal parameters, one needs basis functions whose variance in both domains is as small as possible. The uncertainty principle provides a lower-bound on the product of the variances of a function in its primal domain and its Fourier representation [Weyl, 1931]. As a consequence of this principle the maximum resolution in both domains cannot be achieved, but one has to trade frequency resolution against resolution in the primal parameter domain. In order to realize such a trade-off windowed Fourier transformations have been proposed [Mallat, 2009]. One important example is the product of the Fourier basis functions and a Gaussian, which localizes the infinite Fourier basis functions in the primal parameter domain:

$$c^{\text{gabor1D}}_{m,l_0,\sigma}(l) = \exp\left(-\frac{(l-l_0)^2}{\sigma^2}\right)\left[\cos\left(\frac{2\pi ml}{N}\right) + i\sin\left(\frac{2\pi ml}{N}\right)\right] . \tag{3.10}$$

The product of the variances in both domains of the obtained basis functions is equal to the theoretical minimum that is defined by the uncertainty principle [Gabor, 1946]. The frequency resolution can be controlled by the parameter σ, i.e., the larger σ is, the larger the frequency resolution becomes. The location in the primal parameter domain is controlled by the parameter l_0. Since this type of representation has been first analyzed and proposed by Gabor [Gabor, 1946] the resulting basis functions are termed Gabor-chirps. Gabor functions possess a bandpass characteristic and zero mean and are therefore also termed Gabor-wavelets. Gabor functions have been generalized to two dimensions [Daugman, 1980, 1985]:

$$\begin{aligned} c^{\text{gabor2D}}_{h,v,l_0,\sigma}(l_h, l_v) &= \exp\left(-\frac{(l_h - l_{h_0})^2}{\sigma_h^2} + \frac{(l_v - l_{v_0})^2}{\sigma_v^2}\right) \\ &\times \left[\cos\left(\frac{2\pi}{D}\left(h\left(l_h - l_{h_0}\right) + v\left(l_v - l_{v_0}\right)\right)\right)\right. \\ &\left. + \, i\sin\left(\frac{2\pi}{D}\left(h\left(l_h - l_{h_0}\right) + v\left(l_v - l_{v_0}\right)\right)\right)\right] . \end{aligned} \tag{3.11}$$

The 2-dimensional Gabor functions can be controlled with respect to frequency selectivity, spatial selectivity, and orientation selectitivity. It has been shown that given an appropriate parameterization, the projections of image patches on these Gabor functions are very similar to the responses of simple cells to the same visual stimuli [Daugman, 1980, 1985]. Usually, the projection on the Gabor functions is complex valued. Taking the real part is one of the valid approaches to obtain real valued projections to work with [Gabor, 1946, Daugman, 1985]. Additionally, we use a reparameterized version of (3.11). Considering a two-dimensional rotation of degree α

$$R_\alpha = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \tag{3.12}$$

the real part of a two-dimensional Gabor function that is determined by its orientation α, wavelength λ, phase ϕ, and center $\mathbf{l}_0 = (l_{h_0}, l_{v_0})^T$ is

$$c^{\text{RE(gabor2D)}}_{\mathbf{l}_0,\lambda,\sigma,\alpha,\phi}(\mathbf{l}) = \exp\left(-\frac{\|R_\alpha(\mathbf{l}-\mathbf{l}_0)\|^2}{2\sigma^2}\right) \, cos\left(\frac{2\pi}{\lambda}\left(\mathbf{1}R_\alpha(\mathbf{l}-\mathbf{l}_0)\right)+\phi\pi\right) \; . \quad (3.13)$$

Here, $\mathbf{l} = (l_h, l_v)^T$ is the position in the image. Measurements in the visual cortex of macaques have shown that the majority of the simple cells have a spatial bandwidth of approximately one octave, which is independent from the wavelength λ [Valois et al., 1982] . The spatial bandwidth of (3.13) is $\frac{\sigma}{\lambda}$ [Kruizinga and Petkov, 1999]. If the wavelength λ and bandwidth in octaves b are considered as parameters to be chosen by the user, the width σ of the Gaussian window is obtained according to [Kruizinga and Petkov, 1999] as:

$$\sigma = \frac{\lambda}{\pi}\sqrt{\frac{\ln 2}{2}}\frac{2^b+1}{2^b-1} \; . \quad (3.14)$$

The phase parameter ϕ controlls the type of receptive field that is obtained. $\phi = 0$ leads to symmetric center on fields whereas $\phi = 1$ leads to symmetric center-off fields. For $\phi = \frac{1}{2}$ and $\phi = -\frac{1}{2}$ asymmetic receptive fields with oposite polarities are obtained.

3.4 Orthogonal Wavelet Dictionaries

In the orthogonal wavelet framework data representations at different scales are considered. In order to represent a given function $\mathbf{x}(t) \in L^2(\mathbb{R})$ at a scale 2^s a set of basis functions is used that is obtained from dilations and translations of the so-called scaling function ϕ:

$$\phi_{t_0,s}(t) = \frac{1}{\sqrt{2^s}}\phi\left(\frac{t-2^s t_0}{2^s}\right) \qquad t_0, s \in \mathbb{Z},\; t \in \mathbb{R}. \quad (3.15)$$

ϕ is chosen such that [Mallat, 2009]:

- the set of functions $\phi_{t_0,s}(t)$ is an orthonormal basis of a subspace $V_s \subset L^2(\mathbb{R})$.
- the sequence of subspaces $V_0, V_1, \ldots, V_s, V_{s+1} \ldots$ is nested:

$$V_0 \supset V_1 \supset \cdots \supset V_s \supset V_{s+1}, \ldots \; . \quad (3.16)$$

- proceeding from a finer scale to a coarser scale removes only details that are orthogonal to the structures that remain in the subsequent subspaces. If $U^{s\perp s+1}$ denotes a subspace of functions that is contained in V_s where its elements are orthogonal to the functions contained in V_{s+1}:

$$U^{s\perp s+1} = \{\mathbf{x}(t) \in V_s \mid <\mathbf{x}(t), \mathbf{y}(t)> = 0 \ \forall \mathbf{y}(t) \in V_{s+1}\} \tag{3.17}$$

then

$$V_{s+1} = V_s \setminus U^{s\perp s+1} \tag{3.18}$$

holds.

- the union of all orthogonal complement spaces spans the entire function space

$$\bigcup_{s\in\mathbb{Z}} U_s = L^2(\mathbb{R}) \ . \tag{3.19}$$

Due to the subspace structure also

$$\left(\bigcup_{k\in\mathbb{Z},\ k<s_{\max}} U_{k\perp k+1} \right) \bigcup V_{s_{\max}} = L^2(\mathbb{R}) \ . \tag{3.20}$$

An orthogonal basis for the orthogonal complement spaces $U^{s\perp s+1}$ is obtained from dilations and translations of the wavelet function ψ that complements the scaling function ϕ:

$$\psi_{t_0,s}(t) = \frac{1}{\sqrt{2^s}} \psi \left(\frac{t - 2^s t_0}{2^s} \right) \quad t_0, s \in \mathbb{Z},\ t \in \mathbb{R}. \tag{3.21}$$

In this work, we consider finite dimensional vectors $\mathbf{x} \in \mathbb{R}^N$. An orthogonal basis for a finite dimensional vector space can be obtained from the continuous orthogonal wavelet framework by considering:

$$\begin{aligned} c_{l_0,s}^{\mathrm{Wav}(\phi,\psi)}(l) &= \mathbf{g}_s^{\phi,\psi}(l - l_0) \\ & s = 1, \ldots, s_{\max}, \quad l = 0, \ldots, N \quad l, s \in \mathbb{N}_0 \\ & l_0 = k2^s, \quad k = 0, \ldots, \frac{N}{2^s} - 1, \quad k \in \mathbb{N}_0 \end{aligned} \tag{3.22}$$

and

$$\begin{aligned} c^{\mathrm{Scale}(\phi)}_{l_0,s_{\max}}(l) &= \mathbf{h}^{\phi}_{s_{\max}}(l-l_0) \\ & \quad l=0,\dots,N \quad l\in\mathbb{N}_0 \\ & \quad l_0 = k2^{s_{\max}}, \quad k=0,\dots,\frac{N}{2^{s_{\max}}}-1, \quad k\in\mathbb{N}_0 \end{aligned} \tag{3.23}$$

where $s_{\max}$ is the maximum scale that is considered in the new representation and

$$g^{\phi,\psi}_s(l) = \left\langle \hat{\psi}_s\left(t\right), \hat{\phi}_0(t-l)\right\rangle \tag{3.24}$$

as well as

$$h^{\phi}_s(l) = \left\langle \hat{\phi}_s\left(t\right), \hat{\phi}_0(t-l)\right\rangle \ . \tag{3.25}$$

$\hat{\phi}_s(t)$ and $\hat{\psi}_s(t)$ are the periodized scaling function and the periodized wavelet:

$$\hat{\phi}_s(t) = \sum_{k=-\infty}^{\infty} \phi\left(\frac{t}{2^s}+kN\right) \tag{3.26}$$

$$\hat{\psi}_s(t) = \sum_{k=-\infty}^{\infty} \psi\left(\frac{t}{2^s}+kN\right) \quad s\in\mathbb{Z},\ t\in\mathbb{R}. \tag{3.27}$$

Note that there are other ways of handling the boundaries, as for instance folded wavelets [Mallat, 2009].

$\mathbf{h}^{\phi}_1(l)$ and $\mathbf{g}^{\phi,\psi}_1(l)$ can be understood as coefficients of discrete filters which enable a very efficient computation of the representation in the new basis by means of a filter-bank which does not explicitly use the expanded basis vectors (3.22) and (3.23) [Mallat, 2009]. However, in this work, the basis has been explicitly expanded in order to employ sparse approximation algorithms for the coefficient estimation in case of overcomplete bases.

A widely used orthogonal wavelet is the Haar-wavelet, that is defined as follows:

$$\psi^{\mathrm{Haar}}(t) = \begin{cases} 1 & \text{if } 0\le t<\frac{1}{2} \\ -1 & \text{if } \frac{1}{2}\le t<1 \\ 0 & \text{else .} \end{cases} \tag{3.28}$$

The Haar-Wavelet complements a piece-wise constant scaling function

$$\phi^{\text{Haar}}(t) = \begin{cases} 1 & \text{if } 0 \leq t < 1 \\ 0 & \text{else .} \end{cases} \tag{3.29}$$

A Haar-wavelet orthogonal basis for image patches of size $D \times D$ is obtained from the pairwise tensor products of one-dimensional Haar basis vectors

$$\begin{aligned} c^{\text{Wav(Haar)}}_{\mathbf{l}_0,s}(\mathbf{l}) &= c^{\text{Wav}(\phi^{\text{Haar}},\psi^{\text{Haar}})}_{l_{0h},s}(l_h)\, c^{\text{Wav}(\phi^{\text{Haar}},\psi^{\text{Haar}})}_{l_{0v},s}(l_v) \\ & \quad s = 1,\ldots,s_{\max}, \quad s \in \mathbb{N} \\ & \quad l_{0h} = k2^s, \quad k = 0,\ldots,\frac{D}{2^s}-1, \quad k \in \mathbb{N}_0 \\ & \quad l_{0v} = k2^s, \quad k = 0,\ldots,\frac{D}{2^s}-1, \quad k \in \mathbb{N}_0 \end{aligned} \tag{3.30}$$

and

$$\begin{aligned} c^{\text{Scale(Haar)}}_{\mathbf{l}_0,s^{\max}}(\mathbf{l}) &= c^{\text{Scale}(\phi^{\text{Haar}})}_{l_{0h},s^{\max}}(l_h)\, c^{\text{Scale}(\phi^{\text{Haar}})}_{l_{0v},s^{\max}}(l_v) \\ & \quad l_{0h} = k2^{s^{\max}}, \quad k = 0,\ldots,\frac{D}{2^{s^{\max}}}-1, \quad k \in \mathbb{N}_0 \\ & \quad l_{0v} = k2^{s^{\max}}, \quad k = 0,\ldots,\frac{D}{2^{s^{\max}}}-1, \quad k \in \mathbb{N}_0 \ . \end{aligned} \tag{3.31}$$

Here, $\mathbf{l} = (l_h, l_v)^T$ is the position in the image patch. We used $s_{\max} = \log_2 D$. In a very similar way as in the discrete cosine case, one can define an overcomplete Haar-wavelet dictionary. However, in this case overcompleteness is not introduced in terms of frequency but in terms of localisation, i.e., an overcomplete Haar dictionary is generated by smaller translation steps of the wavelet in the primal parameter domain:

$$\begin{aligned} c^{\text{Wav(Haar)}}_{\mathbf{l}_0,s}(\mathbf{l}) &= c^{\text{Wav}(\phi^{\text{Haar}},\psi^{\text{Haar}})}_{l_{0h},s}(l_h)\, c^{\text{Wav}(\phi^{\text{Haar}},\psi^{\text{Haar}})}_{l_{0v},s}(l_v) \\ & \quad s = 1,\ldots,s_{\max}, \quad s \in \mathbb{N} \\ & \quad l_{0h} = 0,\ldots,D-1, \quad l_{0h} \in \mathbb{N}_0 \\ & \quad l_{0v} = 0,\ldots,D-1, \quad l_{0v} \in \mathbb{N}_0 \ . \end{aligned} \tag{3.32}$$

If an overcomplete Haar-wavlet basis is used, again, the representation is not unique. Therefore constraints on the coefficients have to be imposed in order to render the representation unique. If sparseness constraints are imposed on the coefficients, again, the approximation methods described in Chapter 2 can be employed in order to determine the coefficients of the dictionary elements.

4 Dictionary Learning

This chapter discusses two general categorys of dictionary learning approaches that can be combined with the sparse approximation methods that are described in Chapter 2. The methods from the first category aim to minimize the representation error where constraints on the zero norm of the coefficients are explicitly implemented in the sparse approximation method, i.e., the constraint on the number of non-zero coefficients is not part of the target function but is considered in the optimization with respect to the coefficients. The zero norm as measure of the sparsity of the coefficients is invariant against a scaling of the elements of the dictionary. Due to the scaling invariant measure of sparsity, the methods from this category consider an unconstrained optimization problem with respect to the dictionary. Methods from this category are for instance vector quantization approaches, e.g., the k-means algorithm [Hartigan and Wong, 1979], the LBG-algorithm [Linde et al., 1980], and the Neural Gas algorithm [Martinetz and Schulten, 1991, Martinetz et al., 1993, Cottrell et al., 2006]. Also the Method of Optimal Directions (MOD) [Engan et al., 1999], the K-SVD algorithm (KSVD) [Aharon et al., 2006], the Sparse Coding Neural Gas algorithm (SCNG) [Labusch et al., 2008a, 2009a] as well as the combination of the Bag of Pursuits approach for sparse approximation (Section 2.1.4) and the Neural Gas algorithm [Labusch et al., 2010] belong to this group.

The methods from the second category employ a sparse approximation method in order to determine the dictionary coefficients that does not introduce explicit constraints on the number of non-zero coefficients but incorporates some regularization term in the target function in order to prevent trivial, i.e., non-sparse, solutions from being selected. These dictionary learning methods employ relaxation methods, as for instance those approaches discussed in Section 2.2 for the determination of the coefficients. Many methods from this second category possess a probabilistic interpretation in terms of a maximization of the data likelihood or the posteriori probability of the learned dictionary. Since the regularization or penalty term usually is a relaxation of the zero norm such as for instance the L_1 norm, it often is not scaling invariant, i.e., its influence could be minimized by simply increasing the norm of the dictionary elements. In order to prevent this, one has to constrain the norm of the dictionary elements to some fixed value. Hence, methods from the second category have to solve a constrained optimization problem with respect to the dictionary. In many cases, as for instance in the Sparsenet algorithm proposed in [Olshausen and Field, 1996b], the improved algorithm proposed in [Lewicki and Sejnowski, 1998, 2000], or the column normalized FOCUSS variant for dictionary learning, i.e., FOCUSS-CNDL [Kreutz-Delgado et al., 2003], the update of the dictionary is first performed as in an unconstrained optimization and than the dictionary is projected to the constrained solution space by setting the norm of the dictionary elements to a fixed value. Since this approach often leads to slow convergence, improved optimization approaches have been proposed that directly tackle the constrained optimization problem via the Lagrangian dual of the constrained target function [Lee et al., 2007].

4.1 Vector quantization

One of the most basic approaches for dictionary learning is vector quantization. In the domain of vector quantization the dictionary is termed codebook. Vector quantization aims to find a codebook C that minimizes the mean of the squared reconstruction error:

$$E_h = \frac{1}{L}\sum_{i=1}^{L} \|\mathbf{x}_i - C\mathbf{a}_i\|_2^2 \tag{4.1}$$

subject to the constraint

$$(\mathbf{a}_i)_k = \begin{cases} 1 & : \quad k = \arg\min_j \|\mathbf{x}_i - \mathbf{c}_j\|_2^2 \\ 0 & : \quad \text{else} . \end{cases} \tag{4.2}$$

Due to (4.2), vector quantization can also be understood as a model of maximum sparsity.

4.1.1 k-means

The k-means algorithm [Hartigan and Wong, 1979] is a batch method that can be used in order to determine a codebook that minimizes (4.1) constrained by (4.2). In this algorithm, first for some given codebook C that has been intialized for example with randomly selected data samples, for each given sample $\mathbf{x}_i$, the coefficients are determined according to (4.2). Now let L_l be the number of given samples $\mathbf{x}_i$ with $(\mathbf{a}_i)_l = 1$. Consider the derivative of (4.1) with respect to codebook vector $\mathbf{c}_l$

$$\frac{\partial E_h}{\partial \mathbf{c}_l} = \sum_{\mathbf{x}_i : (\mathbf{a}_i)_l = 1} 2\mathbf{c}_l - \mathbf{x}_i = L_l \mathbf{c}_l - \sum_{\mathbf{x}_i : (\mathbf{a}_i)_l = 1} \mathbf{x}_i \ . \tag{4.3}$$

A local minimum of (4.1) is obtained at

$$\begin{aligned} 0 &= L_l \mathbf{c}_l - \sum_{\mathbf{x}_i : (\mathbf{a}_i)_l = 1} \mathbf{x}_i \\ \mathbf{c}_l &= \frac{1}{L_l} \sum_{\mathbf{x}_i : (\mathbf{a}_i)_l = 1} \mathbf{x}_i \ . \end{aligned} \tag{4.4}$$

After each element of the codebook has been subsequently updated according to (4.4), new coefficients are determined according to (4.2). The order in which the codebook vectors are updated is not relevant, since due to the constraint on the coefficients, which ensures that each sample is encoded by only one codebook vector, the update of one codebook vector does not influence the encoding performance of those samples where it has not been used. The update of the codebook and the determination of the coefficients are repeated until some stopping criterion is met. In the most common implementation the algorithm stops if the change of the codebook is lower than some threshold.

4.1.2 LBG-algorithm

The LBG-algorithm [Linde et al., 1980] is a pattern-by-pattern method for vector quantization. In each iteration t, it considers one given sample $\mathbf{x}_t$ whose coefficients $\mathbf{a}_t$ with respect to some codebook C have been determined according (4.2). In iteration t, only the codebook element $\mathbf{c}_l$ with $(\mathbf{a}_t)_l = 1$ is updated. The update is computed from the gradient of $\|\mathbf{x}_t - C\mathbf{a}_t\|_2^2$ with respect to $\mathbf{c}_l$:

$$\mathbf{c}_l = \mathbf{c}_l + \alpha\left(\mathbf{x}_t - \mathbf{c}_l\right) \tag{4.5}$$

where α is some learning rate. If the learning rate is slowly reduced over time, the LBG algorithm performs a stochastic gradient descent on (4.1). Due to its stochastic nature, it can be more robust against local minima of the target function.

4.1.3 Neural Gas

Algorithms such as k-means or the LBG-algorithm consider only the winner for learning, i.e., the codebook vector $\mathbf{c}_l$ for which $(\mathbf{a}_i)_l = 1$ holds. This hard-competitive learning scheme often leads to a sub-optimal utilization of the codebook vectors with respect to (4.1), i.e., bad quantization. Furthermore, the algorithms can be initialization-sensitive and might exhibit slow convergence. Though the stochastic gradient approach of the LBG-algorithm is more robust, depending on the topology of the input data it can also run into these problems.

The Neural Gas algorithm [Martinetz and Schulten, 1991, Martinetz et al., 1993] remedies these deficiencies by means of a soft-competitive learning scheme that facilitates robust convergence to close to optimal distributions of the codebook vectors over the data manifold to be learned.

In the NG algorithm in each update step all possible configurations of the coefficients are considered, i.e.,

$$\mathbf{a}_i^1, \dots, \mathbf{a}_i^M \ , \ (\mathbf{a}_i^l)_l = 1 \ . \tag{4.6}$$

The configurations are sorted according to their reconstruction error and, in contrast to hard-competitive approaches, used in order to update all the codebook vectors $\mathbf{c}_l$, $l = 1, \dots, M$ in each learning iteration. The update is weighted according to the rank of the configuration that uses the l-th codebook vector. Let $\text{rank}(\mathbf{x}_i, \mathbf{a}_i^l, C) = p$ denote the number of configurations $\mathbf{a}_i^m$ with

$$\|\mathbf{x}_i - C\mathbf{a}_i^m\| \leq \|\mathbf{x}_i - C\mathbf{a}_i^l\| \ . \tag{4.7}$$

Now, in the NG algorithm a neighborhood function $h_{\lambda_t}(v) = e^{-v/\lambda_t}$ is introduced and the following modified error function is considered

$$E_s = \sum_{i=1}^{L} \sum_{l=1}^{M} h_{\lambda_t}(\text{rank}(\mathbf{x}_i, \mathbf{a}_i^l, C)) \|\mathbf{x}_i - C\mathbf{a}_i^l\|_2^2 \ . \tag{4.8}$$

It has been shown in [Martinetz et al., 1993] that the update

$$\Delta \mathbf{c}_l = \alpha_t h_{\lambda_t}(\text{rank}(\mathbf{x}_i, \mathbf{a}_i^l, C)) \, (\mathbf{x}_t - \mathbf{c}_l) \, , \tag{4.9}$$

Algorithm 1 The Neural Gas algorithm

```
1 initialize $C = (\mathbf{c}_1, \dots, \mathbf{c}_M)$ using uniform random values
  for $t = 1$ to $t_{\max}$ do
2   select random sample $\mathbf{x}$ out of $X$
3   calculate current size of neighbourhood and learning rate:
        $\lambda_t = \lambda_0 \left(\lambda_{\text{final}}/\lambda_0\right)^{t/t_{\max}}$
        $\alpha_t = \alpha_0 \left(\alpha_{\text{final}}/\alpha_0\right)^{t/t_{\max}}$
4   determine the sequence $l_0, \dots, l_{M-1}$ with:
        $\|\mathbf{x} - \mathbf{c}_{l_0}\| \leq \dots \leq \|\mathbf{x} - \mathbf{c}_{l_k}\| \leq \dots \leq \|\mathbf{x} - \mathbf{c}_{l_{M-1}}\|$
    for $k = 0$ to $M - 1$ do
5     update $\mathbf{c}_{l_k}$ according to $\mathbf{c}_{l_k} = \mathbf{c}_{l_k} + \alpha_t e^{-k/\lambda_t} \left(\mathbf{x} - \mathbf{c}_{l_k}\right)$
    end for
  end for
```

where α_t is an exponentially decreasing learning rate

$$\alpha_t = \alpha_0 \left(\frac{\alpha_{\text{final}}}{\alpha_0}\right)^{\frac{t}{t_{\max}}} , \tag{4.10}$$

is equivalent to a stochastic gradient descent with respect to (4.8) where the coefficients are chosen according to (4.6). Here, α_0 and α_{final} denote the initial and final learning rate. Also the neighborhood-size is exponentially decreasing

$$\lambda_t = \lambda_0 \left(\frac{\lambda_{\text{final}}}{\lambda_0}\right)^{\frac{t}{t_{\max}}} . \tag{4.11}$$

For $\lambda_t \to 0$, (4.8) becomes equal to (4.1). Due to the soft-competitive learning scheme, the NG algorithm shows robust convergence to close to optimal distributions of the codebook vectors over the data manifold. A pseudo code implementation of the NG method is shown in Algorithm 1. Complementary to the pattern-by-pattern version of the algorithm that has been described so far, a batch version of the algorithm has been proposed in [Cottrell et al., 2006].

4.2 Principle Component Analysis

While vector quantization can be understood as a model of maximum sparsity, the framework of principle component analysis (PCA) does not enforce any sparsity on the dictionary coefficients. PCA has been used in many applications [Fukunaga, 1990] and is often considered as a standard preprocessing step.

PCA can be seen as a framework that considers a generative model in which each given sample $\mathbf{x} \in \mathbb{R}^N$ stems from a linear combination of an orthogonal basis $C \in \mathbb{R}^{N \times M}$, $M \leq N$ plus residual

$$\mathbf{x} = C\mathbf{a} + \boldsymbol{\epsilon} \; . \tag{4.12}$$

The coefficients $(\mathbf{a})_j$ are pairwise decorrelated. The orthogonal basis C contains the eigenvectors of the covariance matrix of $\mathbf{x}$ that is $E[\mathbf{x}\mathbf{x}^T]$. In case $M < N$, it contains those eigenvectors that correspond to the largest M eigenvalues of the covariance matrix. Since C is an orthogonal basis, the new representation can be obtained as:

$$\mathbf{a} = C^T \mathbf{x} \; . \tag{4.13}$$

Due to the choice of C, the following properties hold for the new representation $\mathbf{a}$:

- PCA can be seen as a method which determines an orthogonal projection that maps given data $\mathbf{x}$ into a new representation $\mathbf{a}$ in which all dimensions, i.e., entries of $\mathbf{a}$, are pairwise uncorrelated. If all the original dimensions are Gaussian distributed, the new representation $\mathbf{a}$ is statistically independent, since PCA removes all linear dependencies in the new representation. Hence, PCA can be seen as a first step towards ICA. In ICA methods that are based on the maximization of non-Gaussianity, PCA is used as a preprocessing step (see also Section 5.1).

- PCA can be seen as a method that looks for a nested set of orthogonal projections into lower-dimensional subspaces such that the loss of information in terms of the mean squared error is minimized.

- PCA can be seen as a method that looks for a nested set of orthogonal projections into lower-dimensional subspaces which maximize the mutual information between the lower-dimensional representation and the original data [Földiák, 1989].

In the past, a neural network adaptation rule has been proposed that enables a single artificial neuron to learn the eigenvector that corresponds to the largest eigenvalue of the covariance matrix of given data. This eigenvector is often called the direction of maximum variance:

Oja's rule: In Oja [1982] it is shown that $\mathbf{c}$ converges to the direction of maximum variance in the input distribution of $\mathbf{x}$, if it is iteratively updated according to

$$\Delta\mathbf{c} = \alpha\, y\, (\mathbf{x} - y\, \mathbf{c}) \tag{4.14}$$

where $\mathbf{x}$ is randomly selected from the input data, $y = \mathbf{c}^T\mathbf{x}$, and α is the learning rate.

Furthermore, an update rule for a single layer network has been proposed that learns those eigenvectors that correspond to the M largest eigenvalues of the covariance matrix of given data:

Sanger's rule: In [Sanger, 1989] it is shown that $\mathbf{c}_1, \dots, \mathbf{c}_M$ converge to the eigenvectors that correspond to the M largest eigenvalues of the covariance matrix of $\mathbf{x}$, if each vector $\mathbf{c}_j$ is iteratively updated according to

$$\Delta\mathbf{c}_j = \alpha y_j \left(\mathbf{x} - \sum_{l<j} y_l \mathbf{c}_l - y_j \mathbf{c}_j \right) \tag{4.15}$$

where $\mathbf{x}$ is randomly selected from the input data, $y_l = \mathbf{c}_l^T\mathbf{x}$, and α is the learning rate. Note, that (4.15) can be seen as an application of (4.14) where the input data has been orthogonalized with respect to $\mathbf{c}_l$, $l < j$.

4.3 Method of optimal directions

The Method of Optimal Directions (MOD) [Engan et al., 1999] can be interpreted as a generalization of the k-means algorithm. First, we discuss the case in which the coefficients of the dictionary are chosen according to

$$\mathbf{a}_i = \arg\min_{\mathbf{a}} \|\mathbf{x}_i - C\mathbf{a}\|_2^2 \quad \text{subject to} \quad \|\mathbf{a}\|_0 \le k\ . \tag{4.16}$$

Hence, we now consider data representations that encode a given sample as an arbitrary linear combination of at most k dictionary elements.

4.3.1 Scaling invariant dictionary learning

In this case, we aim to minimize the representation error (4.1) where the coefficients have been determined according to (4.16).

In the first step of the MOD algorithm, dictionary coefficients $A = (\mathbf{a}_1, \dots, \mathbf{a}_L)$, $\mathbf{a}_i \in \mathbb{R}^M$ of a given set of training samples $X = (\mathbf{x}_1, \dots, \mathbf{x}_L)$, $\mathbf{x}_i \in \mathbb{R}^N$ with respect to a given dictionary C are determined with some sparse approximation method such as for instance the greedy methods discussed in Section 2.1. The dictionary can be initialized, for instance, with randomly selected data samples.

In order to improve the dictionary such that the representation error (4.1) is reduced, the gradient of (4.1) with respect to the dictionary C is considered:

$$\frac{\partial E_h}{\partial C} = -2(X - CA)A^T \ . \tag{4.17}$$

A local minimum is obtained at

$$\begin{aligned} \frac{\partial E_h}{\partial C} &= 0 \\ 0 &= -2(X - CA)A^T \\ C &= XA^T(AA^T)^{-1} \ . \end{aligned} \tag{4.18}$$

After the dictionary has been updated according to (4.18), new coefficients are determined according to (4.16). The update of the dictionary and the determination of the coefficients are subsequently repeated, until some stopping criterion is met or a maximum number of update steps has been performed.

4.3.2 Column normalized dictionary learning

MOD considers an unconstrained optimization problem with respect to the dictionary, i.e., the minimization of (4.1). This requires a measure of sparsity for the coefficients whose influence cannot be minimized by a simple increase of the norm of the dictionary elements. The zero-norm used in the greedy methods possesses this property. Of course, MOD can also be applied for dictionary learning, if the method that determines the coefficients employs a measure of sparsity that is sensitive to the scaling of the dictionary elements. This is the case, if, for instance, the relaxation methods that are discussed in Section 2.2 are used to determine the coefficients of the dictionary. In order to maintain the influence of the regularization term in the coefficient estimation, the norm of the elements of the dictionary has to be set to some constant value (one) once a new dictionary has been determined according to (4.18). First, one looks for a solution of the unconstrained problem, then this solution is projected to the constrained solution space.

A disadvantage of this projective approach is that it can cause slow convergence. In order to overcome this disadvantage in [Lee et al., 2007] an update mechanism for the dictionary has been proposed that directly solves the constrained optimization problem by a maximization of the corresponding Lagrangian dual.

Let $u > 0$ be some constant, then the target function of the constrained optimization problem is

$$E_{\text{CNDL}} = \sum_{i=1}^{L} \|\mathbf{x}_i - C\mathbf{a}_i\|_2^2 \text{ subject to } \|\mathbf{c}_j\|_2^2 \leq u \; . \tag{4.19}$$

The corresponding Lagrangian is,

$$\begin{aligned} L_{\text{CNDL}} &= \sum_{i=1}^{L} \|\mathbf{x}_i - C\mathbf{a}_i\|_2^2 + \sum_{j=1}^{M} \lambda_j (\|\mathbf{c}_j\|_2^2 - u) \\ &= \mathit{trace}((X - CA)^2) + \mathit{trace}(\Lambda(C^2 - uI)) \; . \end{aligned} \tag{4.20}$$

where $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_M)$, $\Lambda = \mathit{diag}(\boldsymbol{\lambda})$ are the Lagrangian parameters. The maximization of

$$\min_C \; \mathit{trace}((X - CA)^2) + \mathit{trace}(\Lambda(C^2 - uI)) \tag{4.21}$$

with respect to $\boldsymbol{\lambda}$ is equivalent to the constrained minimization of (4.19) with respect to C. In order to implement the minimization with respect to C in (4.21), we consider its derivative

$$\begin{aligned} \frac{\partial L_{\text{CNDL}}}{\partial C} &= 0 \\ 0 &= -2(X - CA)A^T + 2C\Lambda \\ 0 &= -XA^T + CAA^T + C\Lambda \\ XA^T &= CAA^T + C\Lambda \\ XA^T &= C(AA^T + \Lambda) \\ C &= XA^T(AA^T + \Lambda)^{-1} \; . \end{aligned} \tag{4.22}$$

By insertion of (4.22) in (4.20), one obtains the Lagrangian dual which shall be maximized with respect to Λ

$$\begin{aligned}
D_{\mathrm{CNDL}} &= \mathit{trace}\left(X^TX - 2X^TCA + (CA)^2\right) + \mathit{trace}\left(C^2\Lambda - \Lambda u\right) \\
&= \mathit{trace}\left(X^TX - 2X^TCA\right) + \mathit{trace}\left(C(AA^T+\Lambda)C^T\right) - \mathit{trace}\left(\Lambda u\right) \\
&= \mathit{trace}\left(X^TX - 2X^TCA\right) + \\
&\quad\ \mathit{trace}\left(XA^T(AA^T+\Lambda)^{-1}(AA^T+\Lambda)C^T\right) - \mathit{trace}\left(\Lambda u\right) \\
&= \mathit{trace}\left(X^TX - 2X^TCA\right) + \mathit{trace}\left(XA^TC^T\right) - \mathit{trace}\left(\Lambda u\right) \\
&= \mathit{trace}\left(X^TX - X^TCA\right) - \mathit{trace}\left(\Lambda u\right) \\
&= \mathit{trace}\left(X^TX - X^TXA^T(AA^T+\Lambda)^{-1}A\right) - \mathit{trace}\left(\Lambda u\right)\ . \qquad (4.23)
\end{aligned}$$

(4.23) can be maximized for instance with gradient descent. If the maximum of (4.23) with respect to Λ has been determined by

$$\Lambda_* = \arg\max_{\Lambda} \mathit{trace}\left(X^TX - X^TXA^T(AA^T+\Lambda)^{-1}A\right) - \mathit{trace}\left(\Lambda u\right), \qquad (4.24)$$

the dictionary is obtained as

$$C = XA^T(AA^T+\Lambda_*)^{-1}\ . \qquad (4.25)$$

The determination of the coefficients and the update of the dictionary according to (4.25) are subsequently repeated until some stopping criterion is met or a maximum number of learning iterations has been performed.

4.4 K-SVD

The K-SVD algorithm is even more similar to the k-means algorithm and has been proposed in order to improve on the MOD method. According to the authors [Aharon et al., 2006], it provides better convergence speed and robustness against problematic local minima compared to the MOD method. Given fixed coefficients that have been determined by some method according to (4.16), in the K-SVD approach a separate update of each dictionary element is subsequently performed. Since in this case an update of a single dictionary element depends on those updates that already have been performed, the order of the updates is chosen randomly.

For the update of the l-th dictionary element, those data samples are selected where that element has been used in the encoding. Let $S_l = \{\mathbf{x}_i \mid (\mathbf{a}_i)_l \neq 0\}$ be the set of samples where the l-th dictionary element has been used. Now the representation error is considered that is obtained for all $\mathbf{x}_i \in S_l$, if the l-th dictionary element is removed from the encoding:

$$E_l = \sum_{\mathbf{x}_i \in S_l} \|\mathbf{x}_i - C\hat{\mathbf{a}}_i\|_2^2 = \|R_l\|_F^2 \tag{4.26}$$

where

$$(\hat{\mathbf{a}}_i)_m = \begin{cases} 0 & : \text{ if } \quad m = l \\ (\mathbf{a}_i)_m & : \text{ else.} \end{cases} \tag{4.27}$$

Here R_l is a matrix that contains all the residuals that are obtained if the coefficients of the l-th dictionary element are set to zero. In order to choose a better dictionary element, i.e., to minimize E_l, a singular value decomposition of the matrix R_l is performed

$$R_l = U_l \Sigma_l V_l \ . \tag{4.28}$$

Σ_l is a diagonal matrix that contains the singular values of R_l. Let λ_{l_*} be the singular value that has the largest absolute value. The updated l-th dictionary element is the column of U_l that corresponds in (4.28) to this largest singular value. After the l-th dictionary element has been updated, also its coefficients have to be modified since they are likely to be used in subsequent updates. The updated coefficients of the l-th dictionary element are the entries of the l_*-th row of V_l multiplied by the largest singular value λ_{l_*}. In each iteration of the K-SVD algorithm all the dictionary elements $\mathbf{c}_l$, $l = 1, \dots, M$ are updated this way. Then the coefficients are re-determined using some sparse approximation method according to (4.16). This is repeated until some stopping criterion is met or a maximum number of update iterations has been performed.

4.5 Sparse Coding Neural Gas

All the methods for dictionary learning that have been discussed so far employ a single fixed configuration of the coefficients in order to update the dictionary in the learning process. However, often there are many configurations of the coefficients that are almost equally good in terms of reconstruction performance, and it is not clear why one configuration should be preferred for learning. Due to this shortcoming, we introduce the Sparse Coding Neural Gas algorithm (SCNG), which enables us to use many possible configurations of the coefficients in the dictionary learning process. The SCNG method is the first step on the way to translate the idea of soft-competitive unsupervised learning, as it is implemented in the Neural Gas method, into the domain of dictionary learning.

We start with a slight relaxation of the strict constraint on the coefficients that is used in vector quantization. We use a new constraint on the coefficients $\mathbf{a}_i$ in (4.1) such that a sample $\mathbf{x}_i$ can be represented in terms of an arbitrarily scaled single element of the dictionary. In other words, we start by looking for a set of one-dimensional subspaces that cover the data. Hence, we now want to determine a set of data directions instead of data modes as it is done in vector quantization. Furthermore, we require $\|\mathbf{c}_j\|_2^2 = 1$ without loss of generality. This leads to the following optimization problem, which can also be seen as a model of maximum sparseness:

$$\min_{\mathbf{c}_1,\ldots,\mathbf{c}_M} \sum_{i=1}^{L} \|\mathbf{x}_i - C\mathbf{a}_i\|_2^2 \quad \text{subject to} \quad \|\mathbf{a}_i\|_0 \leq 1 \quad \text{and} \quad \|\mathbf{c}_j\|_2^2 = 1\ . \tag{4.29}$$

First consider the marginal case of (4.29), where only one dictionary element is available, i.e, $M = 1$. Now (4.29) becomes:

$$\min_{\mathbf{c}} \sum_{i=1}^{L} \|\mathbf{x}_i - \mathbf{c}a_i\|_2^2 = \sum_{i=1}^{L} \mathbf{x}_i^T\mathbf{x}_i - 2a_i\mathbf{c}^T\mathbf{x}_i + a_i^2 \text{ subject to } \|\mathbf{c}\|_2^2 = 1\ . \tag{4.30}$$

If $\mathbf{x}_i$ and $\mathbf{c}$ are fixed, (4.30) becomes minimal in case of $a_i = \mathbf{c}^T\mathbf{x}_i$. As final optimization problem, one obtains:

$$\max_{\mathbf{c}} \sum_{i=1}^{L} (\mathbf{c}^T\mathbf{x}_i)^2 \quad \text{subject to} \quad \|\mathbf{c}\|_2^2 = 1\ . \tag{4.31}$$

Hence, in this marginal case, the problem of finding the dictionary element that minimizes (4.30) boils down to finding the direction of maximum variance. A well-known learning rule that solves (4.31), i.e., that finds the direction of maximum variance, is called Oja's rule:

$$\Delta\mathbf{c} = \alpha\ y\ (\mathbf{x} - y\ \mathbf{c}) \tag{4.32}$$

with $y = \mathbf{c}^T\mathbf{x}$ and learning rate α. See also Section 4.2 for a discussion of Oja's rule.

Now consider the general case, where $M > 1$ holds. In this case, the optimization problem (4.31) turns into

$$\max_{\mathbf{c}_1,\ldots,\mathbf{c}_M} \sum_{i=1}^{L} \max_l (\mathbf{c}_l^T \mathbf{x}_i)^2 \quad \text{subject to} \quad \|\mathbf{c}_l\|_2^2 = 1 \; . \tag{4.33}$$

We can generalize to this case by first determining the dictionary element that has maximum overlap with respect to the data:

$$l_{\text{win}} = \arg\max_l (\mathbf{c}_l^T \mathbf{x})^2 \; . \tag{4.34}$$

In order to minimize (4.29), we then update this dictionary element $\mathbf{c}_{l_{\text{win}}}$ according to Oja's rule. However, this approach suffers from the same problem as the k-means algorithm. Due to hard-competitive selection of the dictionary element to be updated, it may happen that the dictionary elements will be distributed sub-optimally with respect to the target function (see also Figure 4.1 in the experiments section). To prevent this, we modify the original Neural Gas algorithm (see Algorithm 1, Chapter 4.1.3) to solve the general case of (4.29).

In the Neural Gas algorithm, soft-competitive learning is achieved by controlling the update of each codebook vector by its rank in the sequence of distances of all codebook vectors with respect to a given sample. These distances are computed within the sample space (see Algorithm 1, steps 4 and 5). We replace the distance measure and now consider the following sequence of distances (see Algorithm 2, step 4):

$$-\left(\mathbf{c}_{l_0}^T \mathbf{x}\right)^2 \leq \cdots \leq -\left(\mathbf{c}_{l_k}^T \mathbf{x}\right)^2 \leq \cdots \leq -\left(\mathbf{c}_{l_{M-1}}^T \mathbf{x}\right)^2 \; . \tag{4.35}$$

The modified distance measure requires a new update rule to minimize the distances between the codebook vectors and the current training sample $\mathbf{x}$. By combination of Oja's rule with the soft-competitive update of the NG algorithm, we obtain (see Algorithm 2, step 5):

$$\Delta \mathbf{c}_{l_k} = \alpha_t e^{-k/\lambda_t} y \left(\mathbf{x} - y \mathbf{c}_{l_k}\right) \; . \tag{4.36}$$

As in the NG algorithm, α_t and λ_t are exponentially decreasing learning rate and neighbourhood size:

$$\alpha_t = \alpha_0 \left(\alpha_{\text{final}}/\alpha_0\right)^{t/t_{\max}} \; , \tag{4.37}$$

$$\lambda_t = \lambda_0 \left(\lambda_{\text{final}}/\lambda_0\right)^{t/t_{\max}} \; . \tag{4.38}$$

For $t \to t_{\max}$, one obtains equation (4.14) as the update rule. Because of the optimization constraint $\|\mathbf{c}_j\| = 1$, we normalize the dictionary elements in each learning step. The complete Sparse Coding Neural Gas algorithm is shown in Algorithm 2.

4.5.1 On the convergence of Sparse Coding Neural Gas

Consider the following maximization problem:

$$\max_{\mathbf{c}_1,\dots,\mathbf{c}_M} \sum_{i=1}^{L} \sum_{l=1}^{M} h_{\lambda_t}(k(\mathbf{c}_l,\mathbf{x}_i))(\mathbf{c}_l^T\mathbf{x}_i)^2 \quad \text{subject to} \quad \|\mathbf{c}_l\|_2^2 = 1, \tag{4.39}$$

with $h_{\lambda_t}(v) = e^{-v/\lambda_t}$. Let $k(\mathbf{c}_l,\mathbf{x})$ denote the number of dictionary elements $\mathbf{c}_j$ with $(\mathbf{c}_j^T\mathbf{x})^2 > (\mathbf{c}_l^T\mathbf{x})^2$. Note that for $\lambda_t \to 0$ this optimization problem is equivalent to the optimization problem defined by (4.33). In order to maximize (4.39), we consider the Lagrangian

$$L = \sum_{i=1}^{L} \sum_{l=1}^{M} h_{\lambda_t}(k(\mathbf{c}_l,\mathbf{x}_i))(\mathbf{c}_l^T\mathbf{x}_i)^2 - \beta_l(\mathbf{c}_l^2 - 1), \tag{4.40}$$

where we have introduced the Lagrangian multipliers β_l. We obtain

$$\frac{\partial L}{\partial \mathbf{c}_j} = 2\sum_{i=1}^{L} h_{\lambda_t}(k(\mathbf{c}_j,\mathbf{x}_i))(\mathbf{c}_j^T\mathbf{x}_i)\mathbf{x}_i - 2\beta_j\mathbf{c}_j + R_j \tag{4.41}$$

with

$$R_j = \sum_{i=1}^{L} \sum_{l=1}^{M} h'_{\lambda_t}(k(\mathbf{c}_l,\mathbf{x}_i))\frac{\partial k(\mathbf{c}_l,\mathbf{x}_i)}{\partial \mathbf{c}_j}(\mathbf{c}_l^T\mathbf{x}_i)^2 \tag{4.42}$$

and $h'_{\lambda_t}(v) = \frac{\partial h_{\lambda_t}(v)}{\partial v}$. Due to the arguments presented in [Martinetz et al., 1993], $R_j = 0$ holds. At the maximum we have

$$\frac{\partial L}{\partial \mathbf{c}_j} = 0 \Leftrightarrow \beta_j = \sum_{i=1}^{L} h_{\lambda_t}(k(\mathbf{c}_j,\mathbf{x}_i))(\mathbf{c}_j^T\mathbf{x}_i)^2 \ . \tag{4.43}$$

Using this, we finally obtain the gradient

$$\frac{\partial L}{\partial \mathbf{c}_j} = 2\sum_{i=1}^{L} h_{\lambda_t}(k(\mathbf{c}_j,\mathbf{x}_i))(\mathbf{c}_j^T\mathbf{x}_i)\mathbf{x}_i - (\mathbf{c}_j^T\mathbf{x}_i)^2\mathbf{c}_j \ . \tag{4.44}$$

Algorithm 2 The Sparse Coding Neural Gas algorithm.

```
1 initialize $C = (\mathbf{c}_1, \dots, \mathbf{c}_M)$ using uniform random values
  for $t = 1$ to $t_{\max}$ do
2   randomly select sample $\mathbf{x}$ out of $X$
3   set $\mathbf{c}_1, \dots, \mathbf{c}_M$ to unit length
4   calculate current size of neighbourhood and learning rate:
        $\lambda_t = \lambda_0 (\lambda_{\text{final}}/\lambda_0)^{t/t_{\max}}$
        $\alpha_t = \alpha_0 (\alpha_{\text{final}}/\alpha_0)^{t/t_{\max}}$
    determine $l_0, \dots, l_{M-1}$ with:
        $-(\mathbf{c}_{l_0}^T\mathbf{x})^2 \leq \dots \leq -(\mathbf{c}_{l_k}^T\mathbf{x})^2 \leq \dots \leq -(\mathbf{c}_{l_{M-1}}^T\mathbf{x})^2$
    for $k = 0$ to $M-1$ do
5     with $y = \mathbf{c}_{l_k}^T\mathbf{x}$, update $\mathbf{c}_{l_k}$ according to $\mathbf{c}_{l_k} = \mathbf{c}_{l_k} + \alpha_t e^{-k/\lambda_t} y(\mathbf{x} - y\mathbf{c}_{l_k})$
    end for
  end for
```

Hence, for a randomly chosen $\mathbf{x} \in (\mathbf{x}_1, \dots, \mathbf{x}_L)$ at time t with learning rate α_t, the update

$$\begin{aligned}\Delta\mathbf{c}_j &= \alpha_t h_{\lambda_t}(k(\mathbf{c}_j, \mathbf{x}))\left((\mathbf{c}_j^T\mathbf{x})\mathbf{x} - (\mathbf{c}_j^T\mathbf{x})^2\mathbf{c}_j\right) && (4.45)\\ &= \alpha_t e^{-k(\mathbf{c}_j,\mathbf{x})/\lambda_t} y\,(\mathbf{x} - y\mathbf{c}_j) && (4.46)\end{aligned}$$

performs a stochastic gradient descent with respect to (4.39) [Kushner and Clark, 1978]. Note that the multiplication of a dictionary element by -1 does not change (4.39), therefore the sign of the dictionary elements cannot be recovered by minimization of (4.39).

4.5.2 Generalized Sparse Coding Neural Gas

As in the MOD and K-SVD cases, the Generalized Sparse Coding Neural Gas (GSCNG) algorithm uses a linear combination of k elements of C to represent a given sample $\mathbf{x}_i$. Hence, it considers the same optimization problem:

$$\min_{\mathbf{c}_1,\dots,\mathbf{c}_M} \sum_{i=1}^{L} \|\mathbf{x}_i - C\mathbf{a}_i\|_2^2 \quad \text{subject to} \quad \|\mathbf{a}_i\|_0 \leq k \text{ and } \|\mathbf{c}_j\|_2^2 = 1\ . \qquad (4.47)$$

In contrast to the MOD and K-SVD methods, in the GSCNG algorithm the coefficients are not considered to be fixed in order to compute an update of the dictionary. Rather, they are subsequently determined in parallel to the computation of the dictionary updates.

The parallel computation of the dictionary coefficients and the dictionary update is achieved by performing an OOMP step in each iteration of the GSCNG method. The OOMP method is described in Section 2.1.3. In order to minimize (4.47), we apply an update of R and C prior to the construction step (2.5) and (2.6) in each of the k iterations of OOMP. The update step reduces the norm of the residual that is obtained in the current iteration of the OOMP method. The norm of the residual becomes small if

$$(\mathbf{r}_{l_{\text{win}}}^T \boldsymbol{\epsilon}_i^U)^2 \tag{4.48}$$

is large where $\mathbf{r}_{l_{\text{win}}}$ and $\boldsymbol{\epsilon}_i^U$ are the winning element of the dictionary and the current residual in the current iteration of the OOMP algorithm. Hence, we have to consider the optimization problem

$$\max_{\mathbf{r}_1,\ldots,\mathbf{r}_{M-|U|}} \sum_{i=1}^{L} \max_{l, l \notin U} (\mathbf{r}_l^T \boldsymbol{\epsilon}_i^U)^2 \quad \text{subject to} \quad \|\mathbf{r}_l\| = 1 \; . \tag{4.49}$$

The optimization problem (4.49) is very similar to (4.33), but now the data $\boldsymbol{\epsilon}_i^U$ as well as the dictionary elements $\mathbf{r}_l$ have been orthogonalized with respect to those dictionary elements C^U that have already been used in the previous OOMP iterations. As before, an optimization of (4.49) can be achieved by using Oja's rule. Instead of only the winning column of R, i.e, $\mathbf{r}_{l_{\text{win}}}$ that is updated, we again employ the soft-competitive learning approach of the NG algorithm in order to update each column of R that may be selected in the next iteration of the OOMP algorithm. Again, we determine a sequence of distances of the current training sample to the current dictionary elements. But now, we only consider distances in the subspace that is orthogonal to C^U (see Algorithm 3, step 3):

$$-\left(\mathbf{r}_{l_0}^T \boldsymbol{\epsilon}_i^U\right)^2 \leq \cdots \leq -\left(\mathbf{r}_{l_k}^T \boldsymbol{\epsilon}_i^U\right)^2 \leq \cdots \leq -\left(\mathbf{r}_{l_{M-|U|-1}}^T \boldsymbol{\epsilon}_i^U\right)^2 \quad , \; l_k \notin U \; . \tag{4.50}$$

As before, we combine Oja's rule and the soft-competitive update of the NG algorithm, but the update is now orthogonal to the subspace spanned by C^U. On the one hand, we apply the update to the temporary dictionary R; on the other hand, we accumulate the updates of all subsequent OOMP iterations in the learned dictionary C. Due to the orthogonal projection (2.5) and (2.6) performed in each iteration, these updates are pairwise orthogonal (see Algorithm 3, step 4):

$$\Delta \mathbf{r}_{l_k} = \Delta \mathbf{c}_{l_k} = \alpha_t e^{-k/\lambda_t} y \left(\boldsymbol{\epsilon}_i^U - y\, \mathbf{r}_{l_k}\right) \; . \tag{4.51}$$

This update rule corresponds to a stochastic gradient descent with respect to (4.49), because the arguments provided in Section 4.5.1 can be applied in the same way.

4.5.3 Computational time complexity

Each update step can be split into the following tasks:

- $M - |U|$ distances with respect to the current residual have to be computed. The time complexity of this operation is $\mathcal{O}(MN)$.
- The distances have to be sorted and the winning dictionary element has to be determined. This can be accomplished with a time complexity of $\mathcal{O}(M \log(M))$.
- The winning dictionary element as well as those $M - |U|$ elements of the dictionary that may be used in the subsequent steps have to be updated. This can be performed in $\mathcal{O}(MN)$ operations.
- The residual and $M - |U|$ remaining dictionary elements of size N that have not been used so far, have to be orthogonalized with respect to the winning element of the dictionary. The time complexity of this operation is $\mathcal{O}(MN)$.

Therefore, each update step has a computational time complexity of $\mathcal{O}(MN + M \log(M))$. Each iteration of the Generalized Sparse Coding Neural Gas algorithm performs k update steps, i.e., each iteration has a computational time complexity of $\mathcal{O}(k(MN + M \log(M)))$. Overall, $t_{\max}$ iterations are performed, therefore, the overall time complexity of the algorithm is $\mathcal{O}(t_{\max} k(MN + M \log(M)))$.

The entire Generalized Sparse Coding Neural Gas method is shown in Algorithm 3.

4.5.4 Experiments on synthetic data

We tested the Sparse Coding Neural Gas algorithm on synthetically generated sparse linear combinations. We did not consider the task to determine M and k, i.e., the size of the dictionary that was used to generate the samples and the number of non-zero coefficients in each linear combination; instead, we assumed M and k to be known.

The dictionaries and coefficients used to generate training samples were chosen from a uniform distribution. We varied the mutual coherence of the dictionaries, i.e.,

Algorithm 3 The Generalized Sparse Coding Neural Gas algorithm.

```
    initialize $C = (\mathbf{c}_1, \ldots, \mathbf{c}_M)$ using uniform random values
    for $t = 1$ to $t_{\max}$ do
1     select random sample $\mathbf{x}$ out of $X$
2     set $\mathbf{c}_1, \ldots, \mathbf{c}_M$ to unit length
      calculate current size of neighbourhood: $\lambda_t = \lambda_0 \, (\lambda_{\text{final}}/\lambda_0)^{t/t_{\max}}$
      calculate current learning rate: $\alpha_t = \alpha_0 \, (\alpha_{\text{final}}/\alpha_0)^{t/t_{\max}}$
      set $U = \emptyset$, $\boldsymbol{\epsilon}^U = \mathbf{x}$ and $R = (\mathbf{r}_1, \ldots, \mathbf{r}_M) = C = (\mathbf{c}_1, \ldots, \mathbf{c}_M)$
      for $h = 0$ to $K - 1$ do
3       determine $l_0, \ldots, l_k, \ldots, l_{M-h-1}$ with $l_k \notin U$ :
          $$-(\mathbf{r}_{l_0}^T \boldsymbol{\epsilon}^U)^2 \leq \cdots \leq -(\mathbf{r}_{l_k}^T \boldsymbol{\epsilon}^U)^2 \leq \cdots \leq -(\mathbf{r}_{l_{M-h-1}}^T \boldsymbol{\epsilon}^U)^2$$
        for $k = 0$ to $M - h - 1$ do
4         with $y = \mathbf{r}_{l_k}^T \boldsymbol{\epsilon}^U$, update $\mathbf{c}_{l_k} = \mathbf{c}_{l_k} + \Delta_{l_k}$ and $\mathbf{r}_{l_k} = \mathbf{r}_{l_k} + \Delta_{l_k}$ with
            $$\Delta_{l_k} = \alpha_t e^{-k/\lambda_t} y (\boldsymbol{\epsilon}^U - y \mathbf{r}_{l_k})$$
          set $\mathbf{r}_{l_k}$ to unit length
        end for
5       determine $l_{\text{win}} = \arg \max_{l \notin U} (\mathbf{r}_l^T \boldsymbol{\epsilon}^U)^2$
6       remove projection to $\mathbf{r}_{l_{\text{win}}}$ from $\boldsymbol{\epsilon}^U$ and $R$:
          $$\boldsymbol{\epsilon}^U = \boldsymbol{\epsilon}^U - (\mathbf{r}_{l_{\text{win}}}^T \boldsymbol{\epsilon}^U) \mathbf{r}_{l_{\text{win}}}$$
          $$\mathbf{r}_l = \mathbf{r}_l - (\mathbf{r}_{l_{\text{win}}}^T \mathbf{r}_l) \mathbf{r}_{l_{\text{win}}} \quad , l = 1, \ldots, M \wedge l \notin U$$
7       set $U = U \cup l_{\text{win}}$
      end for
    end for
```

(2.22), in order to study its impact on the reconstruction performance. In order to obtain a random dictionary with mutual coherence z, we repeatedly chose a matrix from a uniform distribution in $[-1, 1]$ until $\lceil 100H(C) \rceil = \lceil 100z \rceil$. Then, the norm of the columns of the dictionary was normalized to unit length. The mean variance of the training samples was normalized such that is equal to one. A certain amount of uniformly distributed noise was added to the training samples.

We considered a two-dimensional toy example where each training sample is a multiple of one of five dictionary elements, i.e., $M = 5, k = 1, N = 2$. The variance of the additive noise was set to 0.01. Figure 4.1 shows the training samples, the original dictionary C^{orig} (dashed lines) and the dictionary C^{learn} that was learned from the data (solid lines). The left part of the figure shows the result obtained by hard-competitive learning, i.e., $\lambda_0 = \lambda_{\text{final}} = 0$. Note that some of the original dictionary elements were not learned correctly due to the sub-optimal distribution of the learned dictionary with respect to the given training data. The right part shows the result obtained using soft-competitive learning, i.e., $\lambda_0 = 5/2, \lambda_{\text{final}} = 0.01$.

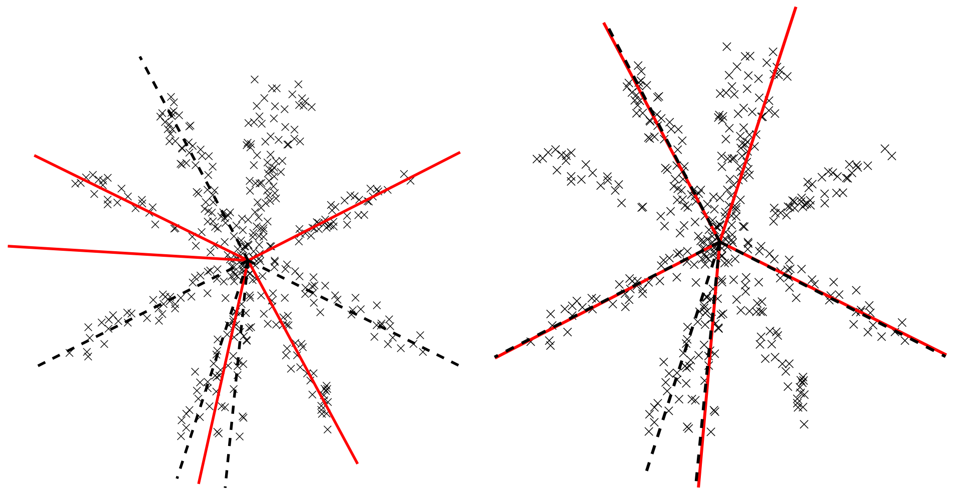


Figure 4.1: A two-dimensional toy example where each sample is a multiple of one of five dictionary elements plus additive noise. **Left:** hard-competitive learning, i.e., $\lambda_0 = \lambda_{\text{final}} = 0$. Some of the original dictionary elements (dashed lines) were not learned correctly. **Right:** soft-competitive learning $\lambda_0 = 5/2, \lambda_{\text{final}} = 0.01$. The original dictionary was obtained except for the sign of its elements. Note that though the data is radially arranged around the center of gravity in this toy example, this is not required for the method to work.

Note that the original dictionary was obtained except for the sign of its elements.

In a second experiment, a dictionary $C^{\text{orig}} \in \mathbb{R}^{40\times 100}$ was generated which consists of $M = 100$ elements of dimension 40. Linear combinations $\mathbf{x}_1, \dots, \mathbf{x}_{10000}$ of k elements were computed. The coefficients in the linear combinations were chosen uniformly distributed in $[-1, 1]$. We generated different dictionaries with mutual coherence $H(C) = 0.3, 0.4, 0.5, 0.6$. The learned dictionary C^{learn} was compared to the original dictionary C^{orig} that was used to generate the samples. This was done by considering the maximum overlap of each original dictionary element $\mathbf{c}_j^{\text{orig}}$ and the best matching learned dictionary element, i.e., $\max_i |\mathbf{c}_i^{\text{learn}}\mathbf{c}_j^{\text{orig}}|$. To assess how many of the learned dictionary elements could be assigned unambiguously to the original dictionary, we considered $|\hat{C}^{\text{learn}}|$, which is the size of the set

$$\hat{C}^{\text{learn}} = \{\mathbf{c}_k^{\text{learn}} : k = \arg\max_i |\mathbf{c}_i^{\text{learn}}\mathbf{c}_j^{\text{orig}}|,\ j = 1, \dots, M\} . \tag{4.52}$$

All experiments were repeated 10 times.

Figure 4.2 shows the impact of the mutual coherence of the dictionary on the mean maximum overlap and on the mean of $|\hat{C}^{\text{learn}}|$ for $k = 1, \dots, 15$. It can be seen that the smaller the mutual coherence of the underlying dictionary is, the better the reconstruction performance becomes. The amplitude of the additive noise was set to 0.1. Figure 4.3 shows the impact of the variance of the additive noise on the mean maximum overlap and on the mean of $|\hat{C}^{\text{learn}}|$. An increasing noise level leads to decreasing performance, as expected. Figure 4.2 and Figure 4.3 show that the less sparse the coefficients are (the larger k is), the lower the quality of the dictionary reconstruction becomes (see also Tropp [2004], Donoho et al. [2006]).

In the third experiment, we fixed $k = 9$ and evaluated the target function (4.47) during the learning process while varying the noise amplitude and the mutual coherence of the dictionary. The coefficients used for reconstruction were determined by Optimized Orthogonal Matching Pursuit with k steps. Figure 4.4 shows that the reconstruction error decreases over time. The smaller the noise level is, the smaller the remaining reconstruction error becomes. The mutual coherence of the dictionary has only slight influence on the remaining reconstruction error.

Finally, in order to compare the performance of the algorithm to other methods, we repeated an experiment that has been described in [Aharon et al., 2006]. A dictionary $C^{\text{orig}} \in \mathbb{R}^{20\times 50}$ was generated, consisting of $M = 50$ elements of dimension 20. Linear combinations $\mathbf{x}_1, \dots, \mathbf{x}_{1500}$ of $k = 3$ elements were computed using uniformly distributed coefficients. We added Gaussian noise to obtain data with varying SNR. We obtained the learned dictionary by application of the Sparse Coding Neural Gas algorithm to the data. In Aharon et al. [2006], the number of learning iterations was set to 80 where each learning iteration uses the entire data. Therefore, in the SCNG method, we set $t_{\max} = 80 * 1500$. As in [Aharon et al., 2006], we compared the learned dictionary to the original one using the maximum overlap between each original dictionary element and the learned dictionary, i.e, whenever

$$\max_j \left(1 - |\mathbf{c}_i^{\text{orig}} \mathbf{c}_j^{\text{learn}}|\right) \tag{4.53}$$

is smaller than 0.01, we counted this as a success. We repeated this experiment 50 times with a varying SNR of 10dB, 20dB and 30dB including zero noise. As in [Aharon et al., 2006], for each noise level we sorted the 50 trials according to the number of successfully learned dictionary elements and ordered them in groups of ten experiments. Figure 4.5 shows the mean number of successfully detected dictionary elements for each of the ten groups for each noise level. For comparison, the results of the MOD method [Engan et al., 2000], the method of Kreutz-Delgado (MAP) [Kreutz-Delgado et al., 2003] and for the K-SVD method [Aharon et al.,

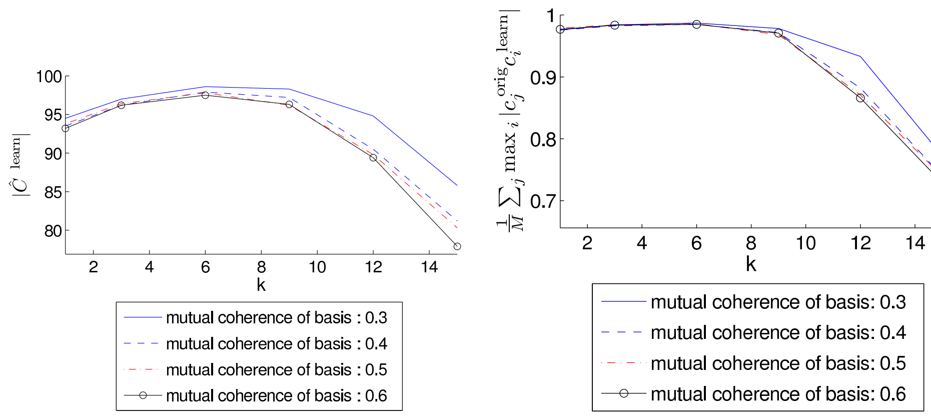


Figure 4.2: The impact of the mutual coherence $H(C)$ on the performance of Sparse Coding Neural Gas is shown. We used $M = 100$ dictionary elements of dimension 40. **Left:** mean size of $\hat{C}^{\text{learn}}$. **Right:** mean maximum overlap between original and learned dictionary. The larger the mutual coherence of the dictionary is and the less sparse the linear combinations are, the more the performance decreases. Sparse Coding Neural Gas parameters used: $\lambda_0 = M/2, \lambda_{\text{final}} = 0.01, \alpha_0 = 0.1, \alpha_{\text{final}} = 0.0001, t_{\max} = 10 * 10000$. The noise variance was set to 0.1.

2006], taken from [Aharon et al., 2006], are shown in Figure 4.5.

It can be seen that the Sparse Coding Neural Gas method outperformed the MAP method for all noise levels and performed as good as MOD for the 20dB and 30dB SNR and noise-free settings. Surprisingly, the performance of SCNG degraded at the 10dB SNR setting. K-SVD outperformed Sparse Coding Neural Gas. It should be noted that K-SVD and MOD are batch methods that use in each learning iteration the entire data in order to obtain the next update of the dictionary C, whereas Sparse Coding Neural Gas is a pattern-by-pattern online method that only uses one data sample at a time.

4.5.5 Experiments on natural image data

We used the SCNG algorithm to learn an overcomplete representation of random patches of natural images. The image patches of size 8×8 pixels were chosen randomly out of a number of landscape photographs published by Olshausen together with the Sparsenet algorithm. In order to reduce the influence of low frequencies on the reconstruction error, the images were bandpass filtered as described in [Olshausen and Field, 1996b]. The learned representation is 6.25 times overcomplete, i.e., it consists of 400 dictionary elements of size $8 \times 8 = 64$. k, the number of

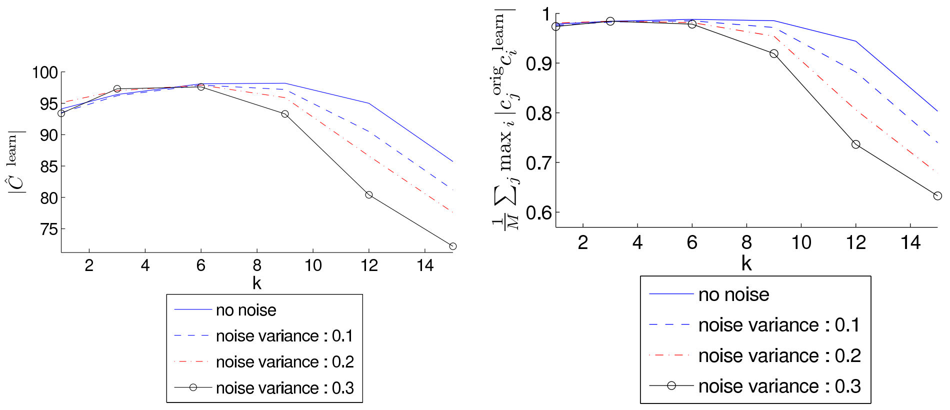


Figure 4.3: The impact of the noise level on the performance of Sparse Coding Neural Gas is shown. We used $M = 100$ dictionary elements of dimension 40. **Left:** mean size of $\hat{C}^{\text{learn}}$. **Right:** mean maximum overlap between original and learned dictionary. The more noise is present and the less sparse the linear combinations are, the more the performance decreases. Sparse Coding Neural Gas parameters used: $\lambda_0 = M/2, \lambda_{\text{final}} = 0.01, \alpha_0 = 0.1, \alpha_{\text{final}} = 0.0001, t_{\max} = 10 * 10000$. The coherence of the dictionary was set to 0.4.

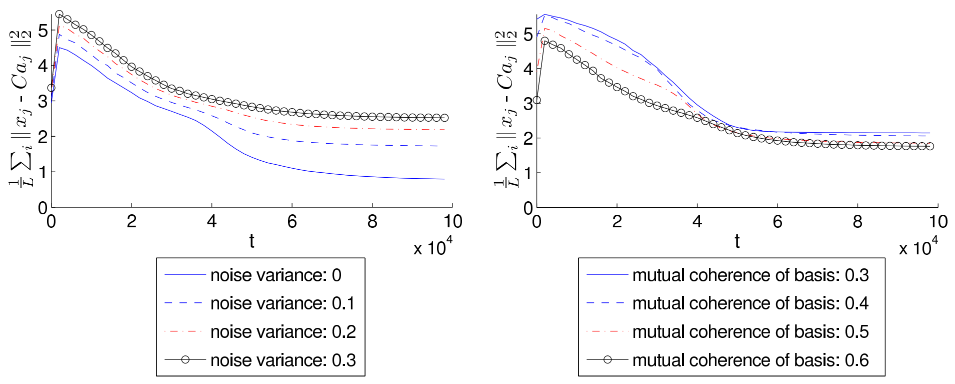


Figure 4.4: The mean reconstruction error over time is shown. We used $M = 100$ dictionary elements of dimension 40 and set $k = 9$. **Left:** Impact of different noise levels on the reconstruction performance. The mutual coherence of the dictionary was set to 0.4. **Right:** Impact of the mutual coherence of the dictionary on the reconstruction performance. The noise variance was set to 0.1. The more noise is present, the larger the remaining reconstruction error becomes. The mutual coherence has only slight influence on the remaining reconstruction error.

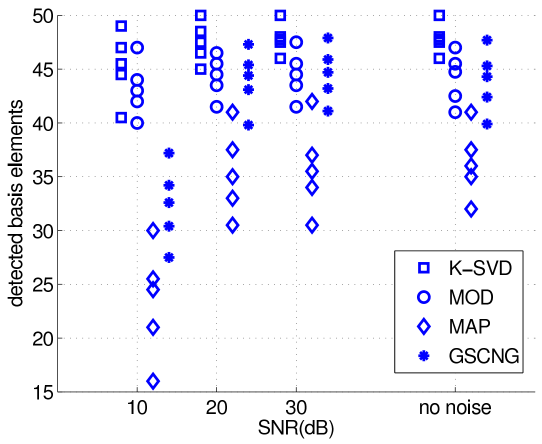


Figure 4.5: Comparison of the performance of Generalized Sparse Coding Neural Gas (GSCNG) with respect to the reconstruction of the original dictionary on synthetical data. The performance of MOD, K-SVD, MAP and GSCNG in the same setting are shown. The results for MOD, K-SVD and MAP were taken from Aharon et al. [2006]. GSCNG outperforms MAP and performs as good as MOD except on the 10dB SNR setting. K-SVD outperforms GSCNG.

non-zero entries per linear combination, was set to 30.

Similar experiments have been performed by a number of researchers. They report the emergence of dictionary elements that, like Gabor wavelets, resemble properties of simple cells in the visual cortex, i.e., they obtain bandpass-like basis functions that are localized in space and orientiation [Olshausen and Field, 1996b, Bell and Sejnowski, 1997, Hyvärinen and Hoyer, 2000]. An overcomplete basis of these patches of natural images obtained using the Sparse Coding Neural Gas algorithm is shown in Figure 4.6. It can be seen that the results reported by other researchers could be reproduced, i.e., we obtained bandpass-like structures ranging over different scales and localized in space and orientation.

4.6 Neural Gas for Dictionary Learning

In the previous section, the SCNG method for dictionary learning was introduced. In contrast to other dictionary learning methods it can use many configurations of the coefficients in the learning process. However, compared to MOD or K-SVD it has the disadvantage that it is directly derived from the OOMP method for sparse approxi-

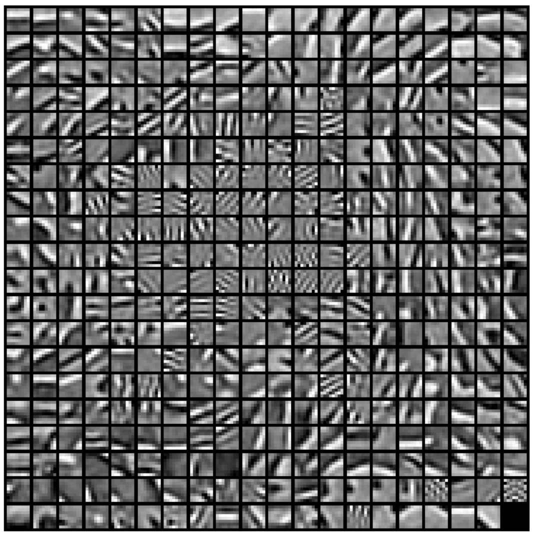


Figure 4.6: A 6.25-times overcomplete basis of patches of natural images of size 8×8 pixels that was obtained by applying Sparse Coding Neural Gas to natural image data. The basis functions were arranged by mapping the basis vectors to a 2D grid using a Kohonen map.

mation. Therefore, it cannot be combined with an arbitrary method for the estimation of the configuration of the coefficients. In this section, we present the Neural Gas for Dictionary Learning (NGDL), which translates the idea of soft-competitive unsupervised minimization of the reconstruction error as it is implemented in the Neural Gas even more directly into the domain of dictionary learning.

We again consider the mean squared reconstruction error (4.1) as target function. We want to minimize this reconstruction error under the constraint that the coefficients of the dictionary have been determined by an arbitrary sparse approximation method. For instance, one could use a greedy method which is based on the zero norm as measure of sparsity or a relaxation method which is based on the L_1-norm as measure of sparsity.

4.6.1 Hard-competitive stochastic gradient dictionary learning

The approach we begin with is a very simple way to minimize (4.1). Suppose that we are given coefficients $\mathbf{a}_t$ that have been estimated with an arbitrary sparse approximation method with respect to a sample $\mathbf{x}_t$ that is randomly selected from the training data in iteration t, $t = 0, \ldots, t_{\max}$. In each iteration, we update the dictionary C according to the gradient of (4.1) with respect to C:

$$\Delta C = \alpha_t(\mathbf{x}_t - C\mathbf{a}_t){\mathbf{a}_t}^T \tag{4.54}$$

α_t is the same exponentially decreasing learning rate, i.e. (4.10), that is also used in the SCNG algorithm.

Once an update of the dictionary according to (4.54) has been performed, the columns of the dictionary are renormalized to one. Then, a new training sample $\mathbf{x}_{t+1}$ is selected, the coefficients $\mathbf{a}_{t+1}$ are determined, and the next update of C can be performed. This simple procedure is fast, since it does not involve a singular value decomposition or a matrix inversion. Furthermore, it uses only one sample in each learning step and is therefore even applicable for online-learning. It also does not require to store a large set of training samples. Apart from the exponentially decreasing learning rate, the update (4.54) can be understood as the pattern-by-pattern variant of the update (4.86) which is used in the Sparsenet algorithm.

4.6.2 Experiments on hard-competitive stochastic gradient descent

In the experiments we again used synthetic data that actually can be represented as sparse linear combinations of some dictionary. We performed the experiments in order to asses two questions: (i) How good is the target function (4.1) minimized? (ii) Is it possible to obtain the true underlying dictionary only from the given data?

In the following $C^{\text{true}} = (\mathbf{c}_1^{\text{true}}, \ldots, \mathbf{c}_{50}^{\text{true}}) \in \mathbb{R}^{20\times 50}$ denotes a synthetic dictionary. Each entry of C^{true} was uniformly chosen in the interval $[-0.5, 0.5]$. Furthermore, we set $\|\mathbf{c}_l^{\text{true}}\| = 1$. Using such a dictionary, we created a training set $X = (\mathbf{x}_1, \ldots, \mathbf{x}_{1500})$, $\mathbf{x}_i \in \mathbb{R}^{20}$ where each training sample $\mathbf{x}_i$ is a sparse linear combination of the columns of the dictionary:

$$\mathbf{x}_i = C^{\text{true}}\mathbf{b}_i \ . \tag{4.55}$$

We chose the coefficient vectors $\mathbf{b}_i \in \mathbb{R}^{50}$ such that they contain k non-zero entries. We performed the selection of the position of the non-zero entries in the coefficient vectors according to three different data generation scenarios:

- **Random dictionary elements:** In this scenario all combinations of k dictionary elements are possible. Hence, the position of the non-zero entries in each coefficient vector $\mathbf{b}_i$ was uniformly chosen in the interval $[1, \dots, 50]$.

- **Independent Subspaces:** In this case the training samples are located in a small number of k-dimensional subspaces. We achieved this by defining $\lfloor 50/k \rfloor$ groups of dictionary elements, each group containing k randomly selected dictionary elements. The groups do not intersect, i.e., each dictionary element was at most member of one group. In order to generate a training sample, we uniformly chose one group of dictionary elements and obtained the training sample as a linear combination of the dictionary elements that belonged to the selected group.

- **Dependent subspaces:** In this case, similar to the previous scenario, the training samples are located in a small number of k-dimensional subspaces. In contrast to the previous scenario, the subspaces do highly intersect, i.e., the subspaces share basis vectors. In order to achieve this, we uniformly selected $k-1$ dictionary elements. Then, we used $50-k+1$ groups of dictionary elements where each group consists of the $k-1$ selected dictionary elements plus one further dictionary element. Again, in order to generate a training sample, we uniformly chose one group of dictionary elements and obtained the training sample as a linear combination of the dictionary elements that belong to the selected group.

The value of the non-zero entries was always chosen uniformly in the interval $[-0.5, 0.5]$. Finally the data was scaled such that the mean variance is equal to 1.

We applied MOD, K-SVD, and the hard-competitive variant of NGDL to the training data. In case of MOD and K-SVD, we used the implementations that were provided by the authors of [Aharon et al., 2006].

Let $C^{\text{learned}} = (\mathbf{c}_1^{\text{learned}}, \dots, \mathbf{c}_{50}^{\text{learned}})$ denote the dictionary that has been learned by one of these methods on the basis of the training samples. In order to measure the performance of the methods with respect to the minimization of the target function, we considered

$$E_h = \frac{1}{1500} \sum_{i=1}^{1500} \|\mathbf{x}_i - C^{\text{learned}} \mathbf{a}_i\|_2^2 \tag{4.56}$$

where $\mathbf{a}_i$ was obtained from the OOMP algorithm. In order to asses if the "true" dictionary can be reconstructed from the training data, we considered the mean

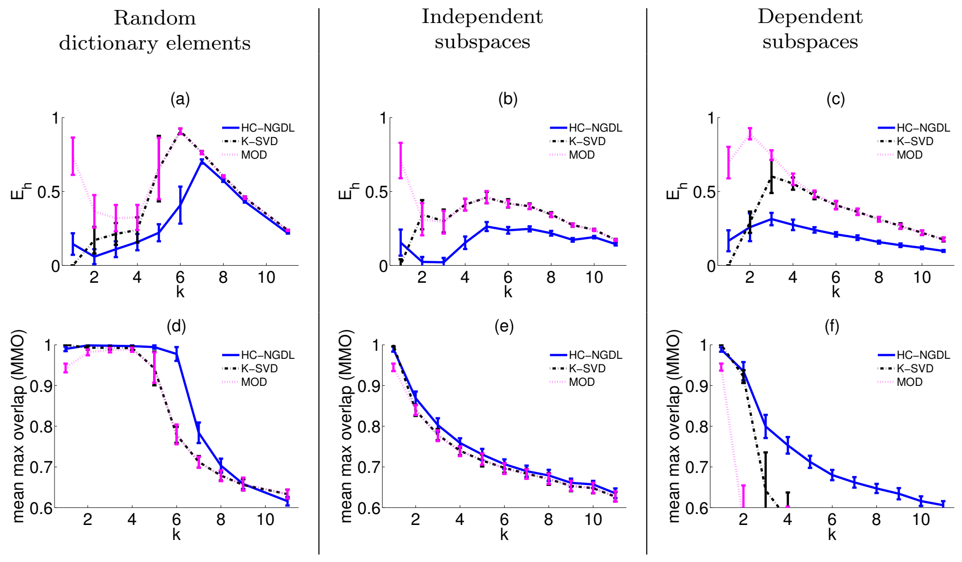


Figure 4.7: Experimental results for hard-competitive NGDL. See text for details. HC-NGDL: hard-competitive NGDL ($\alpha_0 = 0.1$, $\alpha_{\mathrm{final}} = 10^{-3}$). MOD and K-SVD: 100 learning iterations each using 1500 training samples. During learning for all methods the coefficients were obtained from OOMP. All experiments were repeated 50 times.

maximum overlap between each element of the true dictionary and the learned dictionary:

$$MMO = \frac{1}{50}\sum_{l=1}^{50} \max_{k=1,\dots,50} |\mathbf{c}_l^{\mathrm{true}}\mathbf{c}_k^{\mathrm{learned}}| \,. \tag{4.57}$$

k, the number of non-zero entries was varied from 1 to 11. For the hard-competitive NGDL method, we performed 100×1500 update steps, i.e., 100 learning epochs. For MOD and K-SVD, we performed 100 learning iterations each iteration using 1500 training samples. Note, that this yielded the same computational demand for all the methods used. We repeated all experiments 50 times and report the mean result over all experiments. For all dictionary learning methods, i.e., MOD, K-SVD, and hard-competitive NGDL the dictionary coefficients were determined with OOMP during learning.

The results of this experiment are depicted in Figure 4.7. In case of the random dictionary elements scenario (see (a) and (d)) hard-competitive NGDL clearly outperformed MOD and K-SVD. From the mean maximum overlap (see (d)), it can be seen that almost all dictionary elements are well reconstructed with up to 6 non-zero coefficients in the linear combinations. If the dictionary elements cannot be recon-

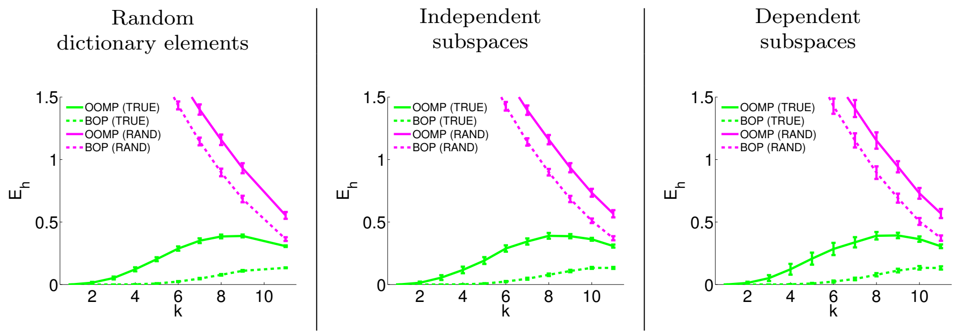


Figure 4.8: Comparison of the mean reconstruction error on the data described in Section 4.6.2. Both OOMP and BOP were provided with the true dictionary the data was generated from. It can be seen that if the “true“ dictionary is known, OOMP finds the “true” solution only for $k = 1$ whereas BOP with $K_{user} = 50$ finds the correct solution up to $k = 5$. For comparison purposes we also show the mean reconstruction error both methods achieve with a random dictionary. All experiments were repeated 50 times.

structed any more, i.e., for $k > 6$, the mean representation error E_h starts to grow (see (a)). In case of the independent subspaces ((b) and (e)) and dependent subspaces scenario ((c) and (f)) the hard-competitive NGDL method also outperformed MOD and K-SVD in terms of minimization of the representation error (see (b) and (c)) whereas in terms of dictionary reconstruction performance only in the intersecting subspaces scenario a clear performance gain compared to MOD and K-SVD can be seen (see (c) and (f)). This might be caused by the fact that in case of the independent subspaces scenario it is sufficient to find dictionary elements that span the subspaces where the data is located in order to minimize the target function, i.e., the scenario does not force the method to find the true dictionary elements in order to minimize the target function.

4.6.3 Evaluation of the Bag of Pursuits

In order to use NGDL in soft-competitive mode, we need a sparse approximation method that provides many possible configurations of the coefficients that can be used for learning. For this task, we proposed the Bag of Pursuits (BOP) method which is described in Section 2.1.4.

We performed an experiment in order to evaluate how far BOP improves OOMP. For that purpose, we again used our synthetical data from Section 4.6.2, however, this time the dictionary was not learned but the “true“ dictionary was assumed to be known. Then, only the best coefficients for reconstruction had to be determined.

This was done with OOMP and, for comparison, with BOP. The result with respect to the reconstruction error both methods achieve is shown in Figure 4.8. OOMP succeeds in finding the correct coefficients only in case $k < 2$. Perfect reconstruction for $k = 1$ can be expected from a theoretical result that was reported in [Tropp, 2004, Donoho et al., 2006] (see also Section 2.3), since the average mutual coherence of our synthetical dictionaries is 0.7 ± 0.05. In contrast, BOP is able to obtain perfect reconstruction even up to $k = 4$ or almost $k = 5$. Since the average mutual coherence is the same in all three data generation scenarios, the performance of BOP and OOMP is almost equal for all the scenarios, if the "true" dictionary was known or a random dictionary was used for the coefficient estimation. This is in contrast to the experiments where the dictionary had to be learned from the training data, and where the scenario had an influence on the ability of the learning method to determine the "true" dictionary (as can be seen for instance in Figure 4.7).

In Figure 4.9, we show the results of the experiment described in Section 4.6.2 that were obtained, if BOP was used to determine the coefficients instead of OOMP. Now, for all dictionary learning methods the coefficients were obtained from the best pursuit out of $K_{\text{user}} = 50$ pursuits that were performed according to the "Bag of Pursuits" approach described in Section 2.1.4. Compared to the results of the first experiment (see Figure 4.7), it can be seen that the computationally more demanding method for the approximation of the best coefficients leads to a significantly improved performance of MOD, K-SVD and hard-competitive NGDL with respect to the minimization of the reconstruction error (see (a)-(c)). The most obvious improvement can be seen in case of the dependent subspaces scenario where also the dictionary reconstruction performance significantly improves (see (c) and (f)). In the random dictionary elements (see (a) and (d)) and independent subspaces scenario (see (b) and (e)) there are only small improvements with respect to the reconstruction of the true dictionary.

4.6.4 Soft-competitive stochastic gradient descent

In the experiments in Section 4.6.3, we have shown that the learning performance can be significantly improved by using the best coefficients from the set of solutions that are provided by the BOP method. So far, only the best pursuit from a set of K_{user} pursuits has been used in the stochastic gradient descent. If a sparse approximation method such as BOP is used that subsequently improves on a set of intermediate solutions towards the final solution, it is desireable not only to use the final solution

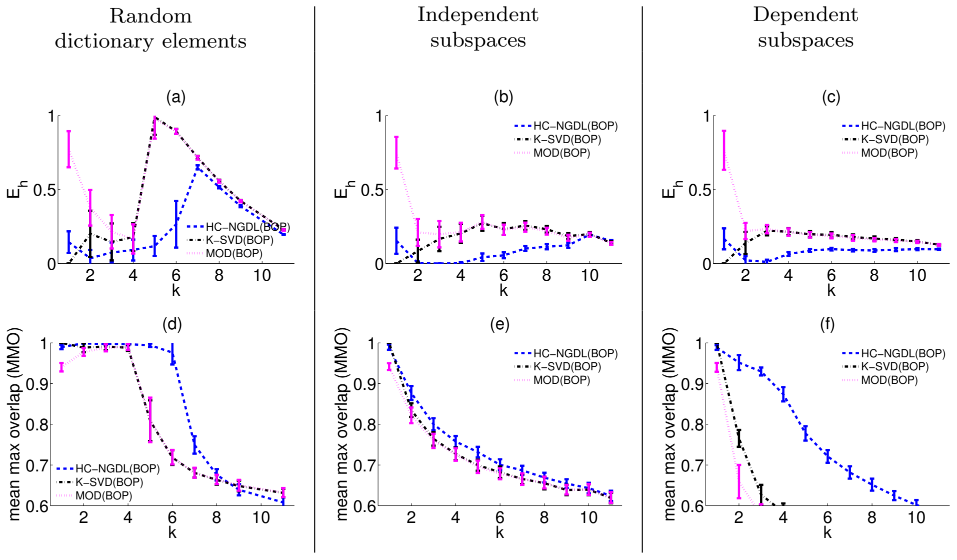


Figure 4.9: Experimental results for MOD, K-SVD, and hard-competitive NGDL. All methods employed the BOP algorithm with $K_{\text{user}} = 50$ in order to estimate the dictionary coefficients in the learning process. See text for details. HC-NGDL(BOP): hard-competitive NGDL ($\alpha_0 = 0.1$, $\alpha_{\text{final}} = 10^{-3}$). MOD and K-SVD: 100 learning iterations, each iteration using 1500 training samples. During learning for all methods the coefficient were obtained from the BOP method with $K_{\text{user}} = 50$. All experiments were repeated 50 times.

in the learning process but also to employ the information that might be contained in the set of intermediate solutions for the dictionary learning process. If possible, this should be implemented such that the use of the intermediate solutions does not introduce additional computational demand.

We now propose a soft-competitive dictionary learning strategy which uses a set of approximations of a training sample. Given the configurations of the hidden coefficients that correspond to the approximations, we aim to minimize the representation error (4.1). Generally, the approximations can be obtained from an arbitrary approximation algorithm that determines a set of approximations. In the following, we consider approximation methods that determine configurations of the coefficients that have at most k non-zero entries. However, the dictionary learning method does not depend on this measure of sparseness and could also be used with other constraints on the coefficients.

We want to directly apply the ranking approach of the Neural Gas algorithm to the learning of sparse codes. Similar to the NG algorithm, for each given sample $\mathbf{x}_i$, we consider all K possible coefficient vectors $\mathbf{a}_i^j$. In contrast to the NG setting, now the coefficients have at most k non-zero entries. Note that K, the number of configurations of the coefficients that have at most k non-zero entries, grows exponentially with M and k. The elements of each $\mathbf{a}_i^j$ are chosen such that $\|\mathbf{x}_i - C\mathbf{a}_i^j\|$ is minimal. We order the coefficients according to the representation error that is obtained by using them to approximate the sample $\mathbf{x}_i$

$$\|\mathbf{x}_i - C\mathbf{a}_i^{j_0}\| < \cdots < \|\mathbf{x}_i - C\mathbf{a}_i^{j_p}\| < \cdots < \|\mathbf{x}_i - C\mathbf{a}_i^{j_K}\| \; . \tag{4.58}$$

If there are coefficient vectors that lead to the same reconstruction error

$$\|\mathbf{x}_i - C\mathbf{a}_i^{m_1}\| = \|\mathbf{x}_i - C\mathbf{a}_i^{m_2}\| = \cdots = \|\mathbf{x}_i - C\mathbf{a}_i^{m_V}\| \; , \tag{4.59}$$

we randomly pick one of them and do not consider the others. Note that we need this due to theoretical considerations while in practice this situation almost never occurs. Let $\text{rank}(\mathbf{x}_i, \mathbf{a}_i^j, C) = p$ denote the number of coefficient vectors $\mathbf{a}_i^m$ with $\|\mathbf{x}_i - C\mathbf{a}_i^m\| < \|\mathbf{x}_i - C\mathbf{a}_i^j\|$. Introducing the neighborhood $h_{\lambda_t}(v) = e^{-v/\lambda_t}$, we consider the following modified error function

$$E_s = \sum_{i=1}^{L} \sum_{j=1}^{K} h_{\lambda_t}(\text{rank}(\mathbf{x}_i, \mathbf{a}_i^j, C)) \|\mathbf{x}_i - C\mathbf{a}_i^j\|_2^2 \tag{4.60}$$

which becomes equal to (4.1) for $\lambda_t \to 0$. In order to minimize (4.60), we consider the gradient of E_s with respect to C, which is

$$\frac{\partial E_s}{\partial C} = -2 \sum_{i=1}^{L} \sum_{j=1}^{K} h_{\lambda_t}(\text{rank}(\mathbf{x}_i, \mathbf{a}_i^j, C))(\mathbf{x}_i - C\mathbf{a}_i^j){\mathbf{a}_i^j}^T + R \tag{4.61}$$

with

$$\begin{aligned} R \quad = \quad & \sum_{i=1}^{L} \sum_{j=1}^{K} h^{'}_{\lambda_t}(\text{rank}(\mathbf{x}_i, \mathbf{a}_i^j, C)) \\ & \cdot \frac{\partial \text{rank}(\mathbf{x}_i, \mathbf{a}_i^j, C)}{\partial C} \|\mathbf{x}_i - C\mathbf{a}_i^j\|_2^2 \,. \end{aligned} \tag{4.62}$$

In order to show that $R = 0$, we adopt the proof given in [Martinetz et al., 1993] to our setting. With $\mathbf{e}_i^j = \mathbf{x}_i - C\mathbf{a}_i^j$, we write $\text{rank}(\mathbf{x}_i, \mathbf{a}_i^j, C)$ as

$$\text{rank}(\mathbf{x}_i, \mathbf{a}_i^j, C) = \sum_{m=1}^{K} \theta((\mathbf{e}_i^j)^2 - (\mathbf{e}_i^m)^2) \tag{4.63}$$

where $\theta(x)$ is the Heaviside step function. The derivative of the Heaviside step function is the delta distribution $\delta(x)$ with $\delta(x) = 0$ for $x \neq 0$ and $\int \delta(x)dx = 1$. Therefore, we can write

$$\begin{aligned} R \quad = \quad & 2 \sum_{i=1}^{L} \sum_{j=1}^{K} h^{'}_{\lambda_t}(\text{rank}(\mathbf{x}_i, \mathbf{a}_i^j, C))(\mathbf{e}_i^j)^2 \\ & \cdot \sum_{m=1}^{K} ((\mathbf{e}_i^m)(\mathbf{a}_i^m)^T - (\mathbf{e}_i^j)(\mathbf{a}_i^j)^T)\delta((\mathbf{e}_i^j)^2 - (\mathbf{e}_i^m)^2) \end{aligned} \tag{4.64}$$

Each term of (4.64) is non-vanishing only for those $\mathbf{a}_i^j$ for which $(\mathbf{e}_i^j)^2 = (\mathbf{e}_i^m)^2$ is valid. Since we explicitly excluded this case, we obtain $R = 0$. Hence, we can perform a stochastic gradient descent on (4.60) with respect to C by applying $t = 0, \ldots, t_{\max}$ updates of C using the gradient based learning rule

$$\Delta C = \alpha_t \sum_{j=1}^{K} h_{\lambda_t}(\text{rank}(\mathbf{x}_i, \mathbf{a}_i^j, C))(\mathbf{x}_i - C\mathbf{a}_i^j){\mathbf{a}_j^i}^T \tag{4.65}$$

for a randomly chosen $\mathbf{x}_i \in X$ where

$$\lambda_t = \lambda_0 \left(\frac{\lambda_{\text{final}}}{\lambda_0} \right)^{\frac{t}{t_{\max}}} \tag{4.66}$$

is an exponentially decreasing neighborhood-size. Again, α_t is an exponentially decreasing learning rate (see equation (4.10)). After each update has been applied, the column vectors of C are renormalized to one. Then the $\mathbf{a}_i^j$ are re-determined and the next update for C can be performed.

So far, for each training sample $\mathbf{x}_i$, all possible coefficient vectors $\mathbf{a}_i^j$, $j = 1, \dots, K$ with $\|\mathbf{a}_i^j\|_0 \leq k$ have been considered. K grows exponentially with M and k. Therefore, this approach is not applicable in practice. However, since in (4.60) all those contributions in the sum for which the rank is larger than the neighborhood-size λ_t can be neglected, we actually do not need all possible coefficient vectors. We only need the first best ones with respect to the reconstruction error. These are directly provided by the BOP method, at least approximately.

4.6.5 Experiments on soft-competitive NGDL

In an experiment, we used soft-competitive NGDL for dictionary learning. The configurations of the coefficients were obtained from the BOP with $K_{\text{user}} = 50$. Each of the configuration was used in the update step according to (4.65) with initial neighbourhood-size $\lambda_0 = 50$ and final neighbourhood-size $\lambda_{\text{final}} = 0.1$. We again used our synthetical data from Section 4.6.2. The results of this experiment are depicted in Figure 4.10. It can be seen that for less sparse scenarios, i.e. $k > 6$, the soft-competitive learning further improves the performance. Particularly in case of the dependent subspaces scenario a significant improvement in terms dictionary reconstruction performance can be seen for $k > 4$ (see (f)). For very sparse settings, i.e. $k \leq 4$, the hard-competitive approach seems to perform better than the soft-competitive variant. Again, in case of the independent subspaces only the representation error decreases (see (b)), whereas no performance gain for the dictionary reconstruction can be seen (see (e)). Again this might be caused by the fact that in case of the subspace scenario learning dictionary elements that span the subspaces is sufficient in order to minimize the target function.

Finally, we evaluated the influence of both, the number of given training data and performed training iterations/epochs, on the ability of the methods to estimate the underlying "true" dictionary. We again applied the soft-competitive NGDL

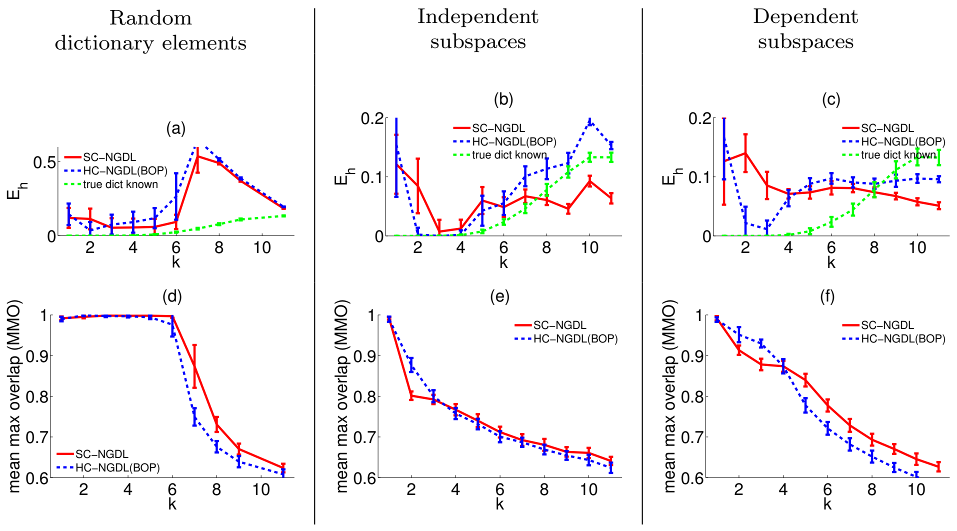


Figure 4.10: Experimental comparison of hard- and soft-competitive NGDL. See text for details. HC-SGD(BOP): hard-competitive NGDL ($\alpha_0 = 0.1$, $\alpha_{\text{final}} = 10^{-3}$). SC-SGD: soft-competitive NGDL ($\alpha_0 = 0.1$, $\alpha_{\text{final}} = 10^{-3}$, $\lambda_0 = 50$, $\lambda_{\text{final}} = 0.1$). All experiments were repeated 50 times. In the random dictionary elements and dependent subspaces scenarios further performance improvements with respect to the dictionary reconstruction can be observed in the soft-competitive case. For $k > 6$ a significant decrease of the mean reconstruction error was obtained. In the dependent and independent subspace scenarios, for $k > 8$ the mean reconstruction error obtained from the soft-competitive approach is even smaller than the mean reconstruction error obtained by using the "true" dictionary. This indicates that in these scenarios the methods are not forced to learn the true dictionary but learn a dictionary that spans the subspaces where the data is located. Obviously, there is a solution that is more incoherent than the true dictionary and still spans these subspaces.

and K-SVD to the synthetic data, but now we varied the number of given training samples and the number of training iterations/epochs. We only used K-SVD for the comparison, since it performed best among the non-gradient methods in the previous experiments. The number of non-zero coefficients in the linear combinations was set to 5. The results of the experiments are shown in Figure 4.11. In the random dictionary elements scenario and the dependent subspaces scenario, the dictionary reconstruction performance of NGDL can be significantly improved by an increase of the number of given training samples (see (a) and (c)). In contrast to this, the performance of the K-SVD method increases only marginally in these cases (see (d) and (f)). Furthermore, in the random dictionary elements scenario with 1000 training samples, with NGDL the dictionary is obtained after approximately 40 training epochs. For NGDL, convergence to the "true" solution can be seen even if only 500 training samples are given (see (a)). In contrast to that, K-SVD did not converge at all to the "true“ solution (see (d)).

In the independent subspaces scenario, the dictionary reconstruction performance of NGDL can also be improved by an increase of the number of given training samples (see (b)). However, in this case, the results indicate that it is not possible to estimate the dictionary with arbitrary accuracy, i.e., due to the structure of the subspaces, at some point, the method has learned a set of dictionary elements that span the subspaces and a further increase of the number of training samples does not lead to a better estimation of the "true" dictionary. Again, convergence to the "true“ solution could not be seen for the K-SVD method as well (see (e)).

4.6.6 Application to image encoding

We studied the influence of the soft-competive update of the dictionary in combination with the BOP method on the performance of the learned dictionary in an image encoding experiment under the condition of a limited number of training samples and limited time being available for learning. The measure of performance was the representation error obtained on unknown data.

We randomly extracted 6000 image patches of size 8×8 from a set of 11 images of size 400×600. Most of the training images are photographs of the city of Brugge. They are depicted in Figure 4.12. The training patches were selected randomly but with a variance within each patch of at least 0.1. We divided the set of random patches in a training and test part, each of size 3000. In a first experiment, we

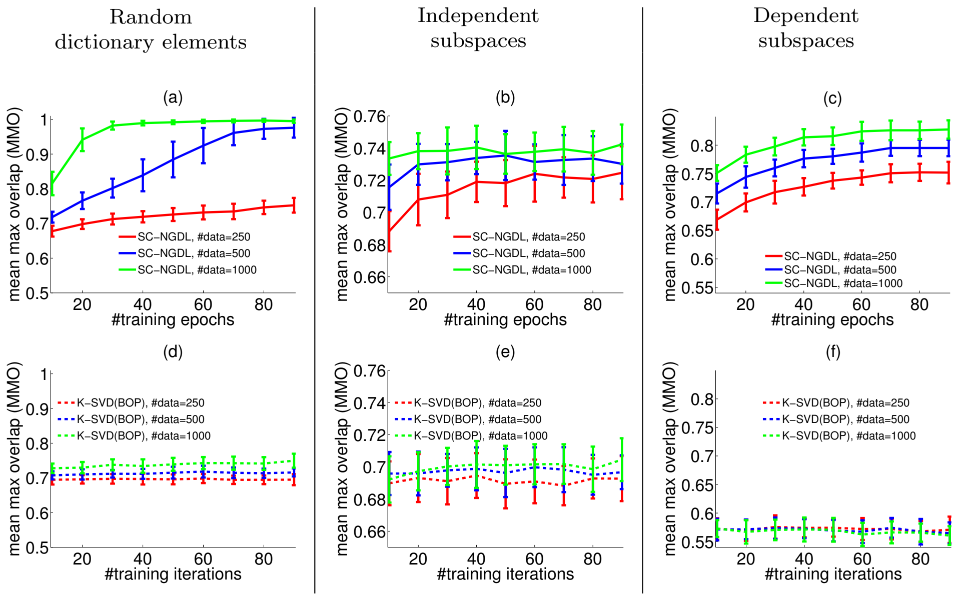


Figure 4.11: Experimental results for soft-competitive NGDL and K-SVD. The figures show the mean maximum overlap between the learned and the “true“ underlying dictionary. k, the number of non-zero entries, was set to 5 in all three scenarios. Both methods employed the BOP algorithm with $K_{\text{user}} = 50$ in order to estimate the dictionary coefficients in the learning process. We varied the size of the given training set and the number of training iterations/epochs. It can be seen that soft-competitive stochastic gradient descent is able to determine the ”true“ dictionary even for small training sets. In all scenarios SC-NGDL convergences significantly faster to significantly better results than K-SVD. SC-NGDL: soft-competitive NGDL ($\alpha_0 = 0.1$, $\alpha_{\text{final}} = 10^{-3}$, $\lambda_0 = 50$, $\lambda_{\text{final}} = 0.1$). All experiments were repeated 50 times.

Figure 4.12: The set of images where the training and test patches were extracted from.

evaluated the test error

$$E_{\text{TEST}} = \frac{1}{3000} \sum_{i=1}^{3000} \|\mathbf{x}_i - C\mathbf{a}_i\|_2^2. \tag{4.67}$$

where $\mathbf{x}_i$ are the elements of the test set. The coefficients $\mathbf{a}_i$ were determined by the BOP method with $K_{\text{user}} = 1, \ldots, 17$, and C was either an overcomplete DCT dictionary of size 441 or an overcomplete Haar dictionary of size 441. The number k of permitted non-zero coefficients was varied between $k = 3$ and $k = 13$. The results are shown in Figure 4.13. It can be seen that the DCT dictionary performed better than the Haar dictionary, and that the reconstruction error remained constant though the number of BOP-trials was increased. Hence, in this case, there is only a slight performance gain obtained by use of the BOP method instead of OOMP for the determination of the coefficients. Note, that for $K_{\text{user}} = 1$ BOP is equal to OOMP.

Next, we evaluated the performance that was obtained on the test set if the dictionary was learned on the training set. In this evaluation, we included the soft-competitive approach proposed in this section (NGDL plus BOP soft), the scaling invariant methods for dictionary learning MOD and K-SVD that have been discussed in Chapter (4), and the simple gradient descent update rule of the Sparsenet algorithm (4.86) (GRAD), which was discussed in Section 4.7. In order to evaluate whether soft-competitive learning actually makes a difference, we also learned a dictionary where the neighborhood-size used in the NGDL+BOP method was practically zero (denoted by NGDL plus BOP hard, $\lambda_0 = \lambda_{\text{final}} = 10^{-10}$), which corresponds to hard-competitive learning. Again, the parameter K_{user} was varied from $K_{\text{user}} = 1$ to $K_{\text{user}} = 17$ and the number of non-zero coefficients was varied between $k = 3$ and $k = 13$.

As already mentioned, we wanted to evaluate the best performance that can be obtained when both, the number of training samples and the available computation time, are limited. We used the training set consisting of 3000 samples of size 8×8 that has been introduced above. The computational demand of all the methods for dictionary learning that have been compared is dominated by the computational effort for the estimation of the coefficients of the dictionary. Therefore, for all methods we fixed the computation time by permitting 30000 estimations of the coefficients. In case of the pattern-by-pattern method NGDL+BOP this corresponds to 10 runs over the entire training set, i.e., 10 training epochs. All the other approaches are batch methods. For a fixed number of estimations of the coefficients, the number of updates of the dictionary is equal to $30000/B$ where B is the batch size. Of course, the batch size is upper-bounded by the number of training samples that are given. The use of all the training data in each update step would correspond to only 10 updates of the dictionary, due to the limited number of estimations of the coefficients. Since such a small number of dictionary updates might be suboptimal, the batch size B was varied and $30000/B$ updates of the dictionary were performed. We report the results for the best trade-off between number of updates of the dictionary and batch size. Apart from the batch size, further parameters had to be chosen for some of the methods. This was done in the following way:

- **Simple gradient (GRAD)**: For a fixed learning rate $\eta = 0.1$ the batch size was varied in 10 steps between 1 and 3000 and the obtained error on the test set was evaluated. For each tested batch size, $30000/B$ updates of the dictionary were performed. The batch size was set to the value that yielded the smallest error on the test set. Using the optimal batch size, the learning rate was varied in 10 steps between 0.01 and 1.0. We report the error on the test set for the optimal batch size and learning rate.

- **NGDL+BOP soft**: For $\lambda_0 = K_{\text{user}}, \alpha_0 = 1.0, \alpha_{\text{final}} = 0.01$ the final neighborhood-size was varied in 10 steps between $\lambda_{\text{final}} = 0.001$ and $\lambda_{\text{final}} = K_{\text{user}}$. Using the optimal final neighborhood-size with respect to the error on the test set the initial neighborhood-size was varied in 10 steps between $\lambda_0 = \lambda_{\text{final}}$ and $\lambda_0 = K_{\text{user}}$. Using the optimal initial and final neighborhood-sizes the final learning rate was varied in 10 steps between $\alpha_{\text{final}} = 0.01$ and $\alpha_{\text{final}} = 1.0$. Using the optimal initial and final neighborhood-sizes and the optimal final learning rate, the initial learning rate was varied in 10 steps between $\alpha_0 = \alpha_{\text{final}}$ and $\alpha_0 = 1.0$. We report the error on the test set that is obtained for optimal initial and final learning rate and neighborhood-size. Note that for $k > 3$ the optimal final neighborhood-size was always the largest tested value, i.e., $\lambda_{\text{final}} = K_{\text{user}}$. This was also the case for $k = 3$ for $K_{\text{user}} < 17$, while in case of $k = 3$ and $K_{\text{user}} = 17$ as optimal final neighborhood-size $\lambda_{\text{final}} = 11.9$ was selected.

- **NGDL+BOP hard**: The same initial and final learning rates were used that were optimal in the soft-competitive case. The initial- and final neighborhood-sizes were set to $\lambda_0 = \lambda_{\text{final}} = 10^{-10}$.

- **MOD**: The batch size B was varied in 10 steps between 442 and 3000. For each tested batch size, $30000/B$ updates of the dictionary were performed. We report the smallest error on the test set over all tested batch sizes.

- **K-SVD**: The batch size was varied in 10 steps between 442 and 3000. For each tested batch size, $30000/B$ updates of the dictionary were performed. We report the smallest error on the test set over all tested batch sizes.

For all methods except NGDL+BOP, only the coefficients of the best solution provided by BOP were used for learning. In order to remove the DC-component from the image patches, for all methods the first dictionary element was set to $\sqrt{1/64}$ and kept constant during learning. It was also forced to participate in each linear combination that was determined by the BOP method.

The results for the learned dictionaries are shown in Figure 4.14. It can be seen that in contrast to the DCT and Haar dictionaries the performance could be improved by increasing the number of BOP trials (K_{user}). NGDL+BOP in soft-competitive mode performed best while K-SVD performed second best. NGDL+BOP in hard-competitive mode performed as good as the standard gradient descent approach when using a constant learning rate (GRAD+BOP). We tested whether the

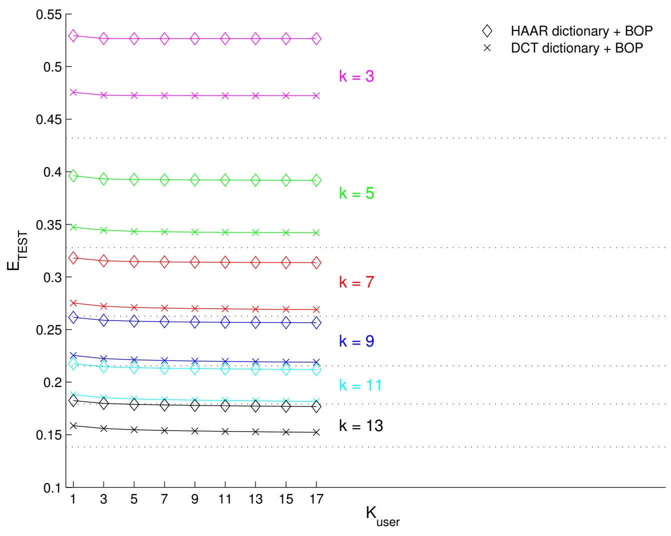


Figure 4.13: Mean squared representation error of the 3000 patches of size 8×8 of the test set that were randomly extracted from the images that are depicted in Figure 4.12. Either an overcomplete DCT dictionary or an overcomplete Haar dictionary were used for the encoding. In this case, the BOP method provides only a slight improvement. Note, that for $K_{\text{user}} = 1$ BOP is equal to OOMP.

performance gap between K-SVD+BOP and NGDL+BOP in soft-competitive mode could be reduced by performing 10× more learning iterations. Therefore, also results for 100 learning epochs with $K_{\text{user}} = 17$ for K-SVD+BOP and NGDL+BOP in soft-competitive mode are reported. It can be seen from Figure 4.14 that the gap was reduced but still noticeable.

4.6.7 Application to image reconstruction

Suppose, we are given an image that is incomplete. Our task is to reconstruct the original image, i.e., determine its missing pixel values from the the remaining pixels. Among other approaches, Wavelet representations have been successfully used to tackle this task [Antonini et al., 1992, Starck et al., 1994]. They have also been applied to the closely related problem of image denoising [Chang et al., 2000]. A key property for the success of certain Wavelet bases in these tasks is that natural image patches can be represented as a sparse linear combination of the Wavelets [Mallat, 2009]. Recently, for instance in [Aharon et al., 2006, Mairal et al., 2008],

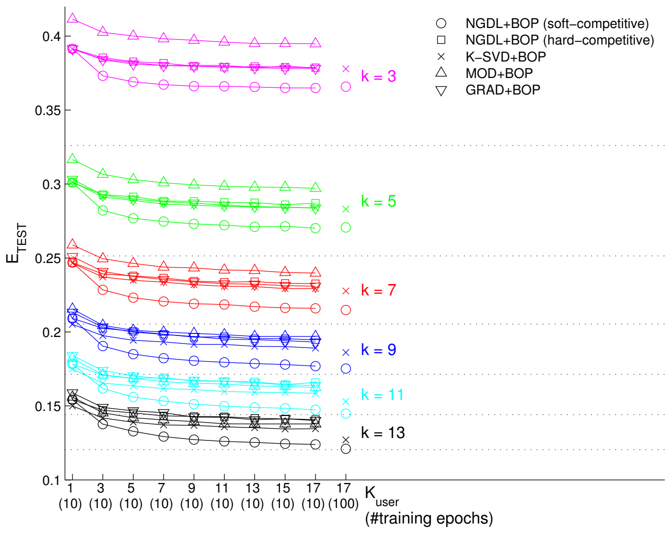


Figure 4.14: Mean squared representation error of the 3000 patches of size 8×8 of the test set. Results for varying number of permitted non-zero coefficients $k = 3, \ldots, 13$ are shown. MOD+BOP: Dictionary learned with the Method of Optimal directions. K-SVD+BOP: Dictionary learned with the K-SVD algorithm. NGDL+BOP(soft-competitive): Method proposed in this section in soft-competitive mode. NGDL+BOP(hard-competitive): Method proposed in this section in hard-competitive mode. GRAD+BOP: Gradient descent with constant learning rate combined with BOP. Note, that for $K_{\text{user}} = 1$ BOP is equal to OOMP. It can be seen that for all methods the performance improves if K_{user} is increased. NGDL+BOP in soft-competitive mode performed best, K-SVD second best. For all methods the user-defined parameters were optimized (see text for details). All methods used a separate training set.

it has been proposed to use dictionaries that have been learned from image data in order to solve this task. A possible advantage of the learned dictionaries is that they can be adapted to the specific properties of a subclass of images which is not always possible in case of Wavelet bases. However, in order to actually exploit this advantage, one needs a method for dictionary learning that is able to extract the subclass specific information from the training images.

In order to formalize the problem of image reconstruction, we consider a set of vectors $\mathbf{p}_1, \dots, \mathbf{p}_P$, $\mathbf{p}_i \in \mathbb{R}^D$. Each vector $\mathbf{p}_i$ contains the remaining pixels of a patch of size $n \times n$ at position i in the given image, where P is the number of all the patches of size $n \times n$ of the original image. If some of the image pixels are missing, this can be written as

$$\mathbf{p}_i = S_i \mathbf{q}_i \ , \quad i = 1, \dots, P \tag{4.68}$$

where S_i is a matrix that describes a projection to a lower-dimensional subspace. $\mathbf{q}_i \in \mathbb{R}^N$ contains the pixels of a patch of size $n \times n$, $n^2 = N$ at position i in the original image. The operator S_i might differ for each position of the image, depending on the pixels that are missing.

In order to obtain the original image, one has to invert the mapping S_i. Since in the given image certain pixels are missing, $D < N$ holds, i.e., in this case linear algebra tells us that, in general, S_i cannot be inverted. A common hypothesis is that the patches of the original image can be represented as a sparse linear combination of some dictionary C plus additive noise:

$$\mathbf{q}_i = C\hat{\mathbf{a}}_i + \boldsymbol{\epsilon}_i \Rightarrow \mathbf{p}_i = S_i C \hat{\mathbf{a}}_i + S_i \boldsymbol{\epsilon}_i \ . \tag{4.69}$$

$C = (\mathbf{c}_1, \dots, \mathbf{c}_M)$, $\mathbf{c}_j \in \mathbb{R}^N$, where $\|\hat{\mathbf{a}_i}\|_0 \leq k \ll D$ and $\|S_i \boldsymbol{\epsilon}_i\| \leq \delta$. Based on this hypothesis, it has been proposed (see Bruckstein et al. [2009] for review) to invert the mapping (4.69) by the solution of a sparse approximation problem

$$\mathbf{a}_i^{l_0} = \arg\min_{\mathbf{a}} \ \|\mathbf{a}\|_0 \quad \text{subject to } \|\mathbf{p}_i - S_i C \mathbf{a}\|_2 \leq \delta \tag{4.70}$$

where $C\mathbf{a}_i^{l_0}$ is the best approximation of $\mathbf{q}_i$ according to (4.70). Section 2.3 presents a result from [Donoho et al., 2006, Gribonval et al., 2006, Bruckstein et al., 2009], i.e., equation (2.28), which states that $\|\mathbf{a}_i^{l_0} - \hat{\mathbf{a}}_i\|_2^2$ is upper-bounded by a constant that is proportional to the square of the noise level δ under the condition that $\mathbf{a}_i^{l_0}$ is sparse enough. How sparse $\mathbf{a}_i^{l_0}$ has to be depends on the mutual coherence of the projected dictionary. On the one hand a smaller mutual coherence $H(S_i C)$ permits a larger number of non-zero entries in $\mathbf{a}_i^{l_0}$, one the other hand a smaller number of non-zero entries $\|\mathbf{a}_i^{l_0}\|_0$ permits a larger mutual coherence of the projected dictionary. The more pixels are missing, i.e., the lower-dimensional the subspace obtained from the projection S_i is, the larger the mutual coherence of the projected dictionary is expected to be.

We evaluated the influence of the soft-competitive learning on the performance in an image reconstruction task where the previously described approach based on sparse approximation is used in order to invert the projection to a lower dimensional subspace. In this experiment, we used those methods that performed best in the previous experiments, i.e., K-SVD+BOP and NGDL+BOP.

We created a larger training set, which better resembles the statistics of the images to be reconstructed. This larger set consists of 150000 image patches that were extracted randomly from the images that are depicted in Figure 4.12. For all methods, we used the optimal parameters that were determined in the previous experiment on image encoding, except for K-SVD where we increased the batch size by a factor of 3. We performed only 1 learning epoch, which means that each training sample was presented exactly once to the learning algorithm. However, due to the much larger training set, this corresponds to an increase of the number of dictionary updates by a factor of ≈ 2 in case of the K-SVD algorithm. Again, the number of permitted non-zero coefficients was varied between $k = 3$ and $k = 13$. We set K_{user} to the value that performed best in the previous experiment which is 17, as can be seen from Figure 4.14. The computation time for each k and method are shown in Table 4.5.

We used 78 test images of size 400×600, which are similar to the images depicted in Figure 4.12. From each test image, we removed certain percentages of pixels (0%, 30%, 50%, 70%, 90%). Then, for each 8×8 patch of the incomplete images we computed the coefficients with respect to a given dictionary using the BOP method ($K_{\text{user}} = 17$). Only the remaining pixels were used, i.e., the dimensions of the dictionary elements that correspond to the missing pixels were not considered. This approach corresponds to the solution of (4.70), where S_i is a projection into a low-dimensional subspace.

The minimum norm of the residual for the BOP stopping criterion, δ (noise level parameter), was varied between $0.00032 = \|0.00004 \cdot \mathbf{1}\|$, $0.0032 = \|0.0004 \cdot \mathbf{1}\|$,$0.032 = \|0.004 \cdot \mathbf{1}\|$,$\mathbf{1} \in \mathbb{R}^{64}$ which corresponds to an average error of approximately ± 0.01, ± 0.1, and ± 1 in 8-bit grayscale images. Note that δ was dynamically adjusted according to the number of pixels that are actually present in an image patch, i.e., a patch that contains n pixels was approximated such that the norm of the residual was less or equal to $\delta_n = \delta\sqrt{64/n}$. In order to reconstruct the image patch, we took the linear combination of the complete dictionary elements using the coefficients that minimize the norm of the residual. We performed the estimation of the coefficients only for non-overlapping patches, since this is computationally more efficient. Of course, the reconstruction can be improved by estimation of the coefficients of each possible patch of the image and use of the mean value of all estimated patches at a certain position as final estimator of the pixel value at that position. However, this is not required for the comparison of the performance of different methods for dictionary learning.

For each setting of the number of permitted non-zero coefficients k, each choice of the noise level parameter δ, and for each method, we computed the mean PSNR value over all 78 test images. In Table 4.1, the best mean PSNR value for different percentages of missing pixels and different methods is shown.

It can be seen that for all percentages of missing pixels the NGDL+BOP soft-competitive approach provided the best mean PSNR value for a certain choice of k and δ. Which k and δ the best choices are, can be seen from Table 4.3 and Table 4.4. From Table 4.3, which lists the best choice for the permitted number of non-zero entries, it can be seen that the more pixels are missing, the less coefficients should be estimated. From Table 4.4 it follows that the noise level δ should be chosen smaller as more pixels are available for the estimation of the coefficients.

The test images are quite different. Therefore, the standard deviation of the PSNR values over the 78 test images is around 4. Also the differences of the mean PSNR values are not that large. Therefore, we took a closer look at the results and counted for each percentage of missing pixels how often each method provided the best PSNR value over all 78 test images. The result is shown in Table 4.2. The best k and δ values were not selected for each image separately, but are exactly the same over all images according to Table 4.3 and Table 4.4. It can be seen from Table 4.2 that the soft-competitive approach not only provided the best mean PSNR value but that its performance is best for the majority of the test images. The more pixels are missing, the less clear the result is. For 90% missing pixels all methods performed equally bad. Those dictionaries learned with NGDL+BOP and K-SVD+BOP that performed best in the image encoding task, i.e., where no pixels are missing, are depicted in Figure 4.15.

percent missing	DCT	Haar	NGDL+BOP (soft)	NGDL+BOP (hard)	K-SVD
0%	37	36.3	39.6	38.9	39.1
30%	31	28.4	32.9	32.5	32.4
50%	27.8	25.6	29.7	29.6	29.6
70%	24.6	23.4	26.8	26.7	26.6
90%	17.5	17.9	18.1	18.1	18.1

Table 4.1: Best mean PSNR values over 78 test images for a certain choice of k, the number of permitted non-zero entries, and δ, the noise level parameter used as stopping criterion in the reconstruction of the images. Which choices of k and δ correspond to these mean PSNR values can be seen from Table 4.3 and Table 4.4. The dictionaries obtained with NGDL+BOP(hard/soft) and K-SVD+BOP that performed best in case of 0% missing pixels are depicted in Figure 4.15.

percent missing	DCT	Haar	NGDL+BOP (soft)	NGDL+BOP (hard)	K-SVD+BOP
0%	0	0	77	0	1
30%	0	0	69	9	0
50%	0	0	52	21	5
70%	0	0	57	15	6
90%	0	13	11	36	18

Table 4.2: For each percentage of missing pixels, the table shows how often a certain method provided the best PSNR value over the 78 test images. Note, that the best k and δ values were not chosen separately for each image, but are exactly the same over all images according to Table 4.3 and Table 4.4.

percent missing	DCT	Haar	NGDL+BOP (soft)	NGDL+BOP (hard)	K-SVD+BOP
0%	13	13	13	13	13
30%	11	13	11	11	11
50%	5	3	5	5	5
70%	3	3	3	3	3
90%	5	7	3	3	3

Table 4.3: Best choice for k, the number of permitted non-zero entries in the estimation of the dictionary coefficients. For instance, in case of 30% missing pixels NGDL+BOP(soft) achieves the best mean PSNR value over all 78 test images with $k = 11$ non-zero coefficients.

percent missing	DCT	Haar	NGDL+BOP (soft)	NGDL+BOP (hard)	K-SVD+BOP
0%	0.00004	0.0040	0.00004	0.00004	0.00004
30%	0.0004	0.0040	0.00004	0.00004	0.0004
50%	0.0004	0.0040	0.0004	0.0004	0.0004
70%	0.0040	0.0040	0.0040	0.0040	0.0040
90%	0.0040	0.0040	0.0040	0.0040	0.0040

Table 4.4: Best choice for δ, the noise level parameter that is used as stopping criterion in the estimation of the coefficients for reconstruction of the images. For instance, in case of 70% missing pixels NGDL+BOP(soft) achieves the best mean PSNR value over all 78 test images with $\delta = 0.004$ and $k = 3$ (see Table 4.3).

k	NGDL+BOP (soft)	NGDL+BOP (hard)	K-SVD+BOP
3	20	19.5	21.8
5	37.7	36.7	38.3
7	54.6	53.4	56.4
9	75	72.4	74.7
11	92	97.4	99.8
13	120.7	115.1	120.5

Table 4.5: Computation time in minutes on a recent desktop computer for each number of permitted non-zero coefficients and method. For all methods, 150000 coefficient estimations were performed. For all methods, the coefficients were determined with BOP ($K_{\text{user}} = 17$)

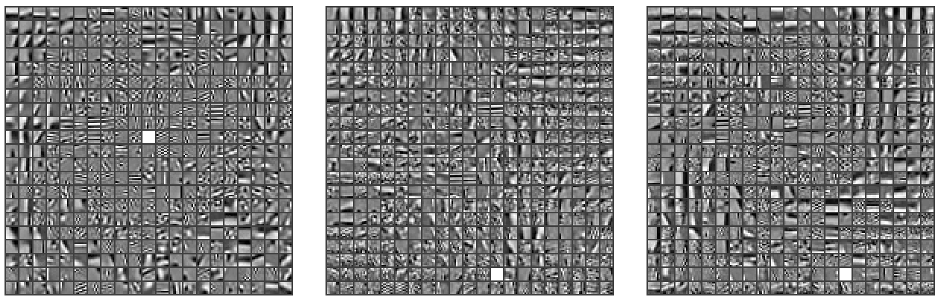


Figure 4.15: The dictionaries for the image reconstruction obtained with $K_{\text{user}} = 17$ and $k = 13$. **Left:** NGDL+BOP soft-competitive ($\lambda_0 = \lambda_{\text{final}} = 17$), **Center:** NGDL+BOP hard-competitive ($\lambda_0 = \lambda_{\text{final}} = 10^{-10}$), **Right:** K-SVD+BOP. The dictionary size is 441. The training set consisted of 150000 randomly extracted image patches of size 8×8. The patches were extracted from 11 images of size 400×600, which are shown in Figure 4.12.

4.6.8 Application to image deconvolution

Image deconvolution is another application domain of sparse linear generative models. In this section, we demonstrate how to deconvolve a given image by sparse approximation with dictionaries that have been learned from a set of training images that are similar to the image that is to be deconvolved.

The problem of image deconvolution can be formalized in a very similar way as it was done in the image reconstruction task. Let a vector $\mathbf{p}_i$ from the set of vectors $\mathbf{p}_1, \dots, \mathbf{p}_{N_p}$, $\mathbf{p}_i \in \mathbb{R}^D$ be a patch of size $d \times d$, $d^2 = D$ at position i in the convolved image. N_p denotes the number of valid patches of size $d \times d$ of the convolved image that has been obtained by application of a known convolution operator Q to the original image:

$$\mathbf{p}_i = Q\mathbf{q}_i \; i = 1, \dots, N_p \; . \tag{4.71}$$

For computational reasons, in the experiments, we considered only those patches that are located at even coordinates. Q is a matrix that describes the convolution operator. A vector $\mathbf{q}_i \in \mathbb{R}^N$ is a patch of size $n \times n$, $n^2 = N$ at position i in the original image that is to be reconstructed. In order to compute the original image from the convolved image, the operator Q has to be inverted. Again, linear algebra tells us that if $D < N$ holds, generally, Q cannot be inverted. Nevertheless, the problem can be approached in a sparse approximation framework by exploitation of the underlying sparse representation of the patches $\mathbf{q}_i$ of the original image.

The hypothesis that enables us to convert the problem into a sparse approximation problem is the same as in the image reconstruction task, namely that the patches of the original image can be represented as a sparse linear combination of some dictionary C plus some additive noise:

$$\mathbf{q}_i = C\mathbf{a}_i + \boldsymbol{\epsilon}_i \Rightarrow \mathbf{p}_i = QC\mathbf{a}_i + Q\boldsymbol{\epsilon}_i \tag{4.72}$$

with $C \in \mathbb{R}^{N \times M}$ and $\|\mathbf{a}_i\|_0 \leq k \ll D$. In order to estimate $\mathbf{q}_i$ on the basis of $\mathbf{p}_i$, we consider the following sparse approximation problem

$$\hat{\mathbf{a}}_i = \arg\min_{\mathbf{a}} \|\mathbf{a}\|_0 \quad \text{subject to} \quad \|\mathbf{p}_i - QC\mathbf{a}\|_2 \leq \delta \tag{4.73}$$

where $\hat{\mathbf{q}}_i = C\hat{\mathbf{a}}_i$ is the approximation of $\mathbf{q}_i$.

In the experiments, we considered two images that have been blurred by a Gaussian convolution kernel of size 7×7 with standard deviation 2.33. The first image depicts a mediaeval building of the city of Brugge (see Figure 4.16). It is also termed house image in the following. The second image depicts a closeup photograph of a flower (see Figure 4.18). It is termed flower image in the following. The blurred versions of the images are also depicted in Figure 4.16 and 4.18.

As already mentioned above, we used dictionaries that were learned from image data. In order to obtain suitable dictionaries, we took training sets of images that are similar to the convolved images. The first training set, which is termed town set in the following, consists of photographs of the city of Brugge (see Figure 4.17). These images mainly depict mediaeval buildings. The second training set, which is termed flowers set in the following, is a collection of closeup photographs of flowers (see Figure 4.19).

All images were linearly scaled such that the pixel values are in the interval $[0, 1]$. From the training images of each image class, we extracted a set of training patches that consists of 150.000 patches of size 16×16, i.e., $\mathbf{x}_1, \ldots, \mathbf{x}_{150000} \quad \mathbf{x}_i \in \mathbb{R}^{256}$. The training patches were selected randomly, but with a variance within each patch of at least 0.1. Then we applied the hard- and soft-competitive NGDL (Sections 4.6.1 and 4.6.4) in order to learn overcomplete dictionaries $C \in \mathbb{R}^{256 \times 1681}$ that are optimized in order to encode the images of the training set with at most k non-zero entries per sample. In the hard- and soft-competitive case, we used the BOP method with $K_{\text{user}} = 10$ in order to determine the linear combination of the elements of the dictionary. We evaluated different choices of k, i.e., $k = 12, 14, 16, 18, 20$. We performed only one learning epoch, i.e., a one-pass run over the entire training set. For comparison purposes the deconvolution experiments were repeated using an overcomplete Haar-wavelet frame, which is depicted in Figure 4.20.

In order to solve the optimization problem (4.73), we first applied the (known) convolution operator Q to the learned dictionary C. Each element of the dictionary can be interpreted as a 2-dimensional patch. These dictionary patches were blurred using the same operator that was applied to the house and flower image. The learned dictionary elements correspond to 16×16 image patches. The application of a 7×7 Gaussian filter to these dictionary elements leads to convolved dictionary elements of size 10×10 (it does not make sense to add a padding region). Hence, the estimation of the coefficients took place with blurred image patches and dictionary patches of size 10×10. The coefficients were determined with the BOP method ($K_{\text{user}} = 10$). As stopping criterion, we used an accuracy of $\delta = 0.0004$ and a maximum value for the number of dictionary elements in the linear combination. Due to the stopping criterion, the iteration of the BOP algorithm stopped, if the reconstruction error became smaller than 0.0004 or if the maximum number of dictionary elements in the linear combination had been reached. Since we used the BOP method for approximation, we obtained for each tile $\mathbf{q}_i$ of the original image a set of approximations, i.e., $\hat{\mathbf{q}}_i^j = C\mathbf{a}_i^j \; j = 1, \ldots, K_{\text{user}}$. We did not use the solution that provided the smallest reconstruction error but we selected from the set of approximations $\hat{\mathbf{q}}_i^{j*}$ that approximation which interferred the least with the estimation of the original image in the neighbourhood of position i.

The interference of an approximation $\hat{\mathbf{q}}_i^j$ with respect to its neighbourhood was computed on the basis of the set of approximations of those tiles of the original image that overlap position i. The tiles of the original image that overlap with position i are denoted as $\mathbf{q}_k \in N(i)$ in the following. Furthermore, let $\hat{\mathbf{q}}_k^l,\ l = 1, \ldots, K_{\text{user}},\ \mathbf{q}_k \in N(i)$ denote the set of approximations of these tiles that have been obtained from the BOP method. Additionally, let $O(i,k)$ and $O(k,i)$ be matrices that implement two mappings into a lower dimensional subspace such that the vector representations of the two tiles at position i and k only contain the corresponding pixels of the tiles that are located in the their overlapping region after the mapping has been applied. Now, $I(i,j)$, i.e., the interference of the j-th approximation at position i, is defined as the sum of the distances of the closest approximations of all overlapping tiles in the neighbourhood of position i:

$$I(i,j) = \sum_{\mathbf{q}_k \in N(i)} \min_{l=1,\ldots,K_{\text{user}}} \|O(i,k)\hat{\mathbf{q}}_i^j - O(k,i)\hat{\mathbf{q}}_k^l\|_2 \; . \tag{4.74}$$

For each tile, we swept through the set of its approximations and selected the one that led to the smallest interference:

$$j^* = \arg\min_j I(i,j) \; . \tag{4.75}$$

The final estimation of the pixel value of the primal image at position i is the mean of the approximations of all tiles that overlap with position i. For each tile, from the set of approximations that was obtained from the BOP method, the least interferring one was chosen accoding to (4.75).

4.6.9 Deconvolution results

The deconvolution results for the house and flower images are depicted in Figure 4.16 and 4.18 respectively. In order to compute a PSNR with respect to the original image, all results were linearly scaled to the interval $[0, 1]$ and centered such that the PSNR was maximized. Furthermore, the images were cut such that artifacts at the borders did not influence the PSNR measurement.

In both cases, i.e., house and flower, the PSNR of the sparse reconstruction improves compared to the PSNR of the blurred image. In case of the house image, it can be improved from $23.3dB$ to $26.3dB$ whereas in case of the flower image it can be improved from $25.6dB$ to $33.3dB$. Figure 4.16 depicts the PSNR values for the house image that were obtained with $k = 12, 14, 16, 18, 20$ and dictionaries learned by hard-competitive as well as soft-competitive stochastic gradient descent. Figure 4.16 also depicts results that were obtained if a dictionary that was learned on the flowers training set was used in order to reconstruct the house image. In Figure 4.18 the same kind of results for the flower image are depicted. Additionally, both figures show the results that were obtained with an overcomplete Haar-wavelet frame.

It can be seen that for both images the learned dictionaries outperformed the Haar-wavelet frame. The larger the number of non-zero entries in the linear combination becomes, the better the obtained performance is. Both figures show that the obtained performance depends on the set of training images that were used in order to learn the dictionaries. In both cases the dictionaries that were obtained from the set of images that are similar to the image that is subject to deconvolution outperformed those dictionaries that were learned from images that do not look similar to the convolved image. Due to computational reasons the number of pursuits performed by the BOP method was small ($K_{\text{user}} = 10$) given the size of the dictionary and the number of non-zero coefficients. Hence, the hard-competitive and soft-competitive method were quite similar in this application. Therefore, also the performance of hard-competitive and soft-competitive learning is rather similar though the soft-competitive version more often slightly outperformed the hard-comeptitive approach than the other way around.

The dictionaries that performed best and were learned from the town- and the flowers training set are depicted in Figure 4.17 and 4.19, respectively. It can be seen that there is a clear difference between the two dictionaries, e.g., the dictionary obtained from the town set contains more higher frequency components.

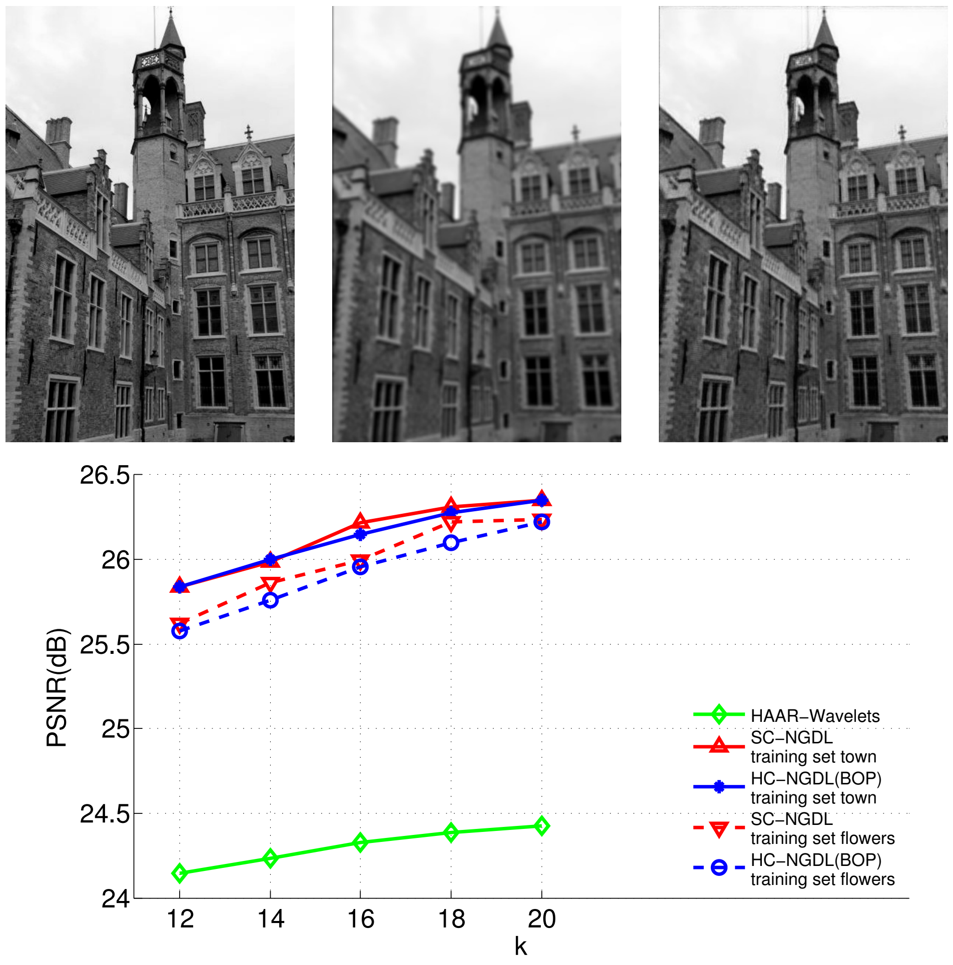


Figure 4.16: **Top Left:** primal image, **Top middle:** blurred image (PSNR: 23.3dB), **Top right:** best deconvolution result (PSNR: 26.3dB, SC-SGD, $k = 20$, training set town, dictionary is depicted in table 4.17). **Bottom:** PSNR of deconvoluted image with respect to primal image for different choices of k and methods. The larger k is, the better the deconvolution becomes. Best results were obtained if the town set (see table 4.17) was used for dictionary training instead of the flowers set. See text for details.

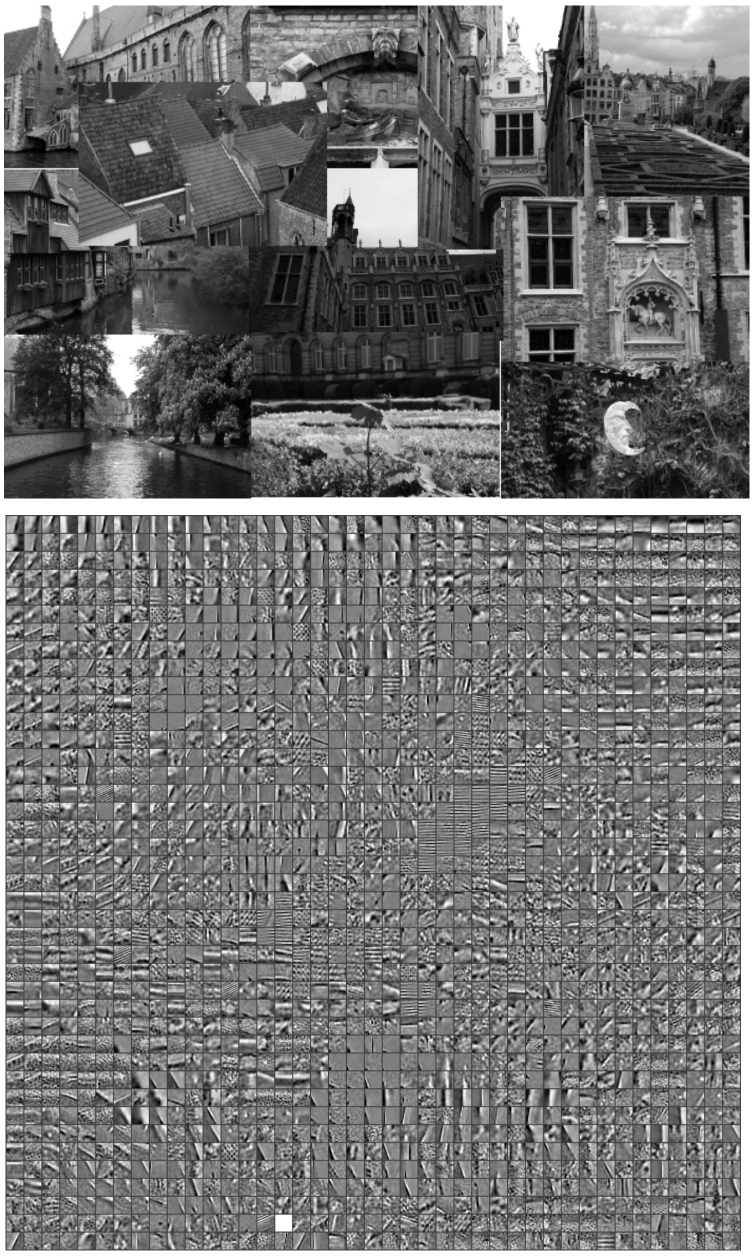


Figure 4.17: **Top:** Town training set. **Bottom:** Dictionary obtained from this set of images that performed best in the experiments depicted in Figure 4.16 (SC-SGD, $K_{\text{user}} = 10, \alpha_0 = 10^{-1}, \alpha_{\text{final}} = 10^{-3}, \lambda_0 = 50, \lambda_{\text{final}} = 0.1$, $k = 20$). The dictionary consists of 1681 elements of size 16×16. The 2D-arrangment was obtained from a Kohonen-Map.

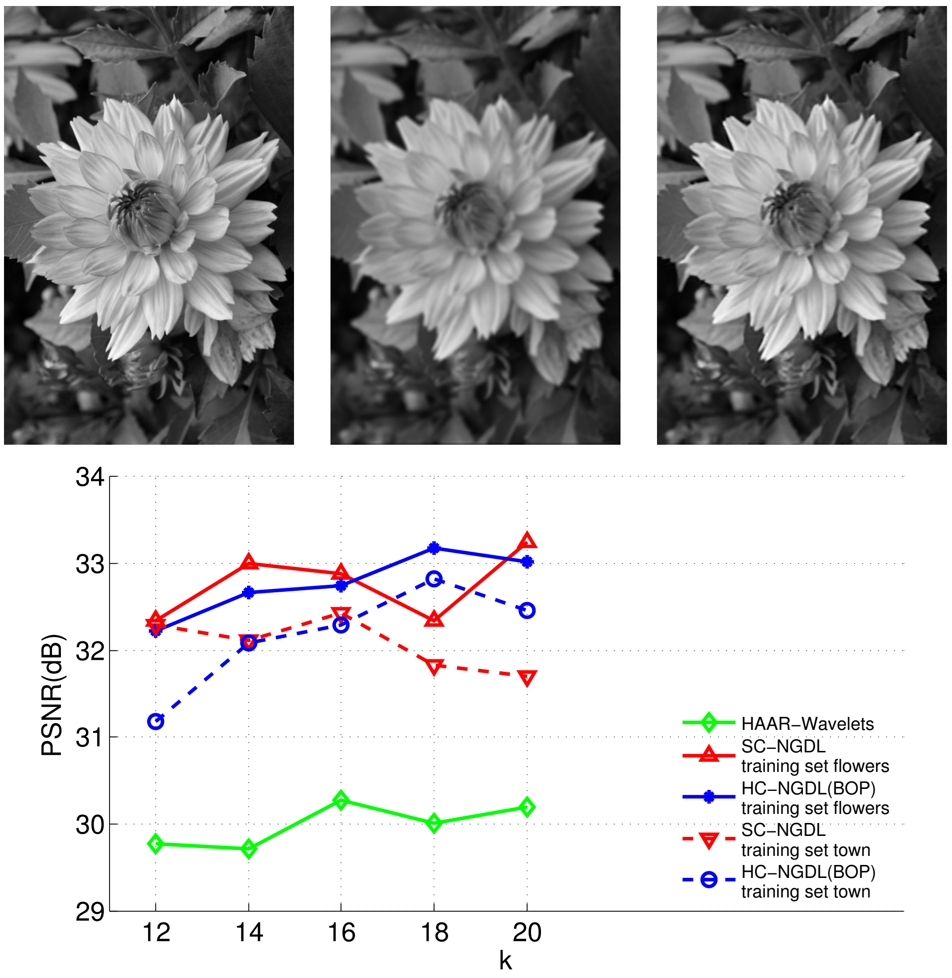


Figure 4.18: **Top left:** primal image, **Top middle:** blurred image (PSNR: 25.6dB), **Top right:** best deconvolution result (PSNR: 33.3dB, SC-SGD, $k = 20$,$\alpha_0 = 10^{-1}$, $\alpha_{\text{final}} = 10^{-2}$, $\lambda_0 = 10$, $\lambda_{\text{final}} = 10^{-2}$, training set flowers). **Bottom:** PSNR of deconvoluted image with respect to primal image for different choices of k and methods. Best results were obtained if the flowers set (see Figure 4.19) was used for dictionary training instead of the town set. See text for details.

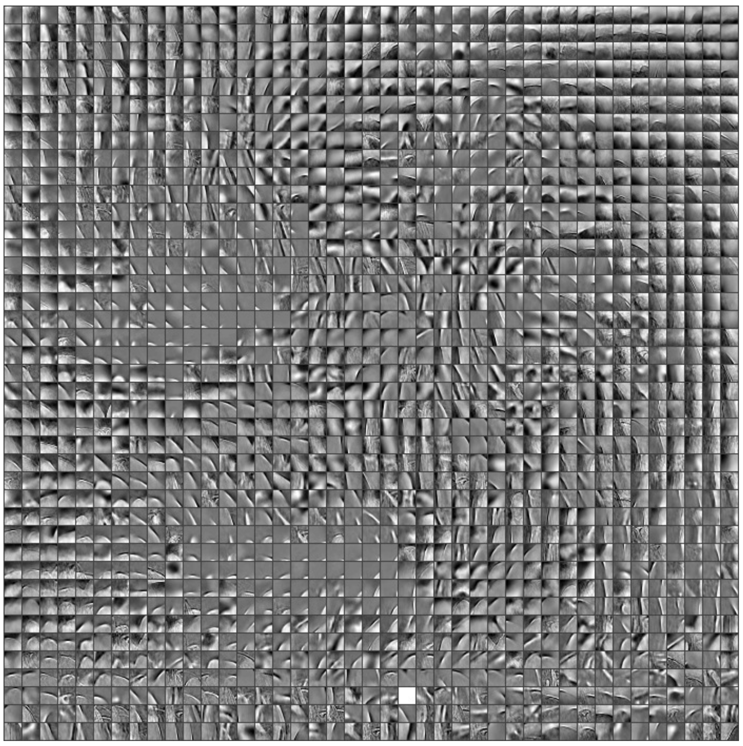


Figure 4.19: **Top:** Flowers training set. **Bottom:** Dictionary obtained from this set of images that performed best in the experiments depicted in Figure 4.18 (SC-SGD, $K_{\text{user}} = 10, \alpha_0 = 10^{-1}, \alpha_{\text{final}} = 10^{-2}, \lambda_0 = 10, \lambda_{\text{final}} = 10^{-2}$, $k = 20$). The dictionary consists of 1681 elements of size 16×16. The 2D-arrangment was obtained using a Kohonen-Map.

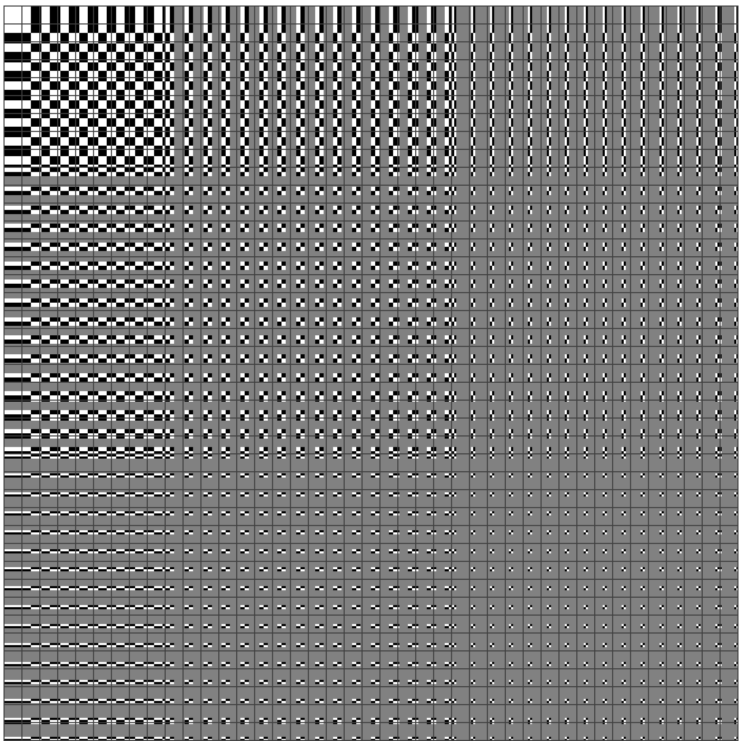


Figure 4.20: The Haar-wavelet frame that was used for comparison purposes.

4.7 The probabilistic point of view

In the probabilistic framework for dictionary learning that is discussed in this section and that has been used to motivate well-known algorithms such as the Sparsenet [Olshausen and Field, 1996b, 1997] and the algorithm proposed by Lewicki and Sejnowski [Lewicki and Sejnowski, 2000], one postulates that a given set of observations $\mathbf{x}_1, \dots, \mathbf{x}_L$, $\mathbf{x}_i \in \mathbb{R}^N$ stems from an unknown but fixed probability distribution with density $P^*(\mathbf{x})$. Furthermore, one considers a class of linear generative models that possess a likelihood of $P(\mathbf{x}|C)$ to generate a sample $\mathbf{x}$. The C is the parameter of the linear generative models that is to be determined.

4.7.1 Maximum likelihood methods

In order to find the best choice for the model parameters C, the similarity between the unknown "true" density $P^*(\mathbf{x})$ and the data likelihood $P(\mathbf{x}|C)$ of the model shall be maximized. The similarity of these two distributions is measured by their Kullback-Leibler divergence

$$KL[P^*(\mathbf{x}), P(\mathbf{x}|C)] = \int P^*(\mathbf{x}) \log \frac{P^*(\mathbf{x})}{P(\mathbf{x}|C)} \mathrm{d}\mathbf{x} \; . \tag{4.76}$$

The minimization of (4.76) corresponds to the maximization of the expectation value of the log-likelihood of the model, i.e.,

$$\int P^*(\mathbf{x}) \log(P(\mathbf{x}|C) \mathrm{d}\mathbf{x} = E\left(\log(P(\mathbf{x}|C)\right) \; , \tag{4.77}$$

since $P^*(\mathbf{x})$ cannot be influenced by the choice of C.

According to the data model that is used in the probabilistic methods, a sample $\mathbf{x}$ is obtained from a linear combination of the model parameters C plus additive noise

$$\mathbf{x} = C\mathbf{a} + \boldsymbol{\epsilon} \; . \tag{4.78}$$

The coefficients $\mathbf{a}$ are hidden variables that are sparsely distributed with density $P(\mathbf{a})$. Sparsely distributed means that their marginal densities $P(a_i)$ are leptocurtic. Furthermore, the joint density $P(\mathbf{a})$ is modeled as a factorial density, i.e., the hidden variables a_i are assumed to be statistically independent

$$P(\mathbf{a}) = \prod_{i=1}^{M} P(a_i) \ . \tag{4.79}$$

One possible choice of a leptocurtic marginal density is the Laplacian

$$P(a_i) = \frac{1}{2\beta} e^{-\frac{|a_i|}{\beta}} \ . \tag{4.80}$$

Usually, the noise $\boldsymbol{\epsilon}$ is considered to be normally distributed

$$P(\mathbf{x}|C, \mathbf{a}) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{\|\mathbf{x}-C\mathbf{a}\|_2^2}{2\sigma^2}} \ . \tag{4.81}$$

The log-likelihood of a given sample $\mathbf{x}$ according to the model (4.78) where coefficients and noise are distributed according to (4.80) and (4.81) is

$$\log\left(\int_{\mathbf{a}} P(\mathbf{x}|C, \mathbf{a}) P(\mathbf{a}) \mathrm{d}\mathbf{a}\right) = \log\left(\frac{1}{N_{\sigma,\beta}} \int_{\mathbf{a}} e^{-\frac{\|\mathbf{x}-C\mathbf{a}\|_2^2}{2\sigma^2} - \frac{M}{2\beta}\|\mathbf{a}\|_1} \mathrm{d}\mathbf{a}\right) \tag{4.82}$$

where $N_{\sigma,\beta}$ is some normalization constant. In order to maximize the log-likelihood, the integration over the hidden parameter $\mathbf{a}$ in (4.82) has to be performed. Since the integration with respect to $\mathbf{a}$ in (4.82) is intractable, an approximation of the integral is used. In the Sparsenet algorithm the likelihood function of the model is simply evaluated at its maximum [Olshausen and Field, 1997] which corresponds to the following approximation

$$\log\left(\int_{\mathbf{a}} e^{-\frac{\|\mathbf{x}-C\mathbf{a}\|_2^2}{2\sigma^2} - \frac{M}{2\beta}\|\mathbf{a}\|_1} \mathrm{d}\mathbf{a}\right) \approx \max_{\mathbf{a}} \ -\frac{\|\mathbf{x}-C\mathbf{a}\|_2^2}{2\sigma^2} - \frac{M}{2\beta}\|\mathbf{a}\|_1 \ . \tag{4.83}$$

An improved approximation based on the Gaussian integral has been proposed in [Lewicki and Sejnowski, 2000]. In this approach the function that is to be integrated is replaced by a Gaussian which is located at its maximum.

Both soft-competitive dictionary learning methods that have been presented in this work, i.e., SCNG which is presented in Section 4.5, and NGDL which is presented in Section 4.6, can be seen as approaches that use a better approximation of the intractable integral (4.82) in order to estimate the best parameters of the model. These methods consider for learning a large set of configurations of the hidden variables $\mathbf{a}$ that are sparse and lead to a small representation error with respect to $\mathbf{x}$, instead of only a single one as it is done in (4.83). In the SCNG algorithm this idea is implemented implicitly while in the NGDL approach the configurations are explicitly determined by the BOP method for sparse approximation.

A maximization of the expectation value of the data likelihood of the model is not possible, since the "true" distribution of the observations $\mathbf{x}$ is unknown. Instead, given a finite set of observations, a maximization of the empirical mean of the data likelihood is performed which corresponds, in case of statistical independence of the observations, to a maximization of

$$\max_C \log\left(P(\mathbf{x}_1, \ldots, \mathbf{x}_L|C)\right) = \max_C \log\left(\prod_{i=1}^{L} \int_{\mathbf{a}} P(\mathbf{x}_i|\mathbf{a}, C)P(\mathbf{a})\mathrm{d}\mathbf{a}\right) . \tag{4.84}$$

If the approximation (4.83) of the data likelihood is used, this finally leads to the following nested optimization problem

$$\min_C \sum_{i=1}^{l} \left(\min_{\mathbf{a}} \|\mathbf{x}_i - C\mathbf{a}\|_2 + \lambda\|\mathbf{a}\|_1\right) . \tag{4.85}$$

In the Sparsenet algorithm, the nested optimization problem (4.85) is tackeled by a two-fold optimization process. First, the dictionary C is considered fixed and the dictionary coefficients are determined by gradient descend with respect to the coefficients (see Chapter 2.2). Then the coefficients are considered fixed and a gradient descent with respect to the dictionary is performed. With learning rate η, the update of the dictionary is

$$\Delta C = \eta\,(X - CA)\,A^T \tag{4.86}$$

where $A = (\mathbf{a}_1, \ldots, \mathbf{a}_L)$ are the coefficients that have been determined for C.

The measure of sparsity used by the Sparsenet algorithm is not scaling invariant, i.e., the influence of the regularization term in (4.85) can be minimized by an increase of the norm of the dictionary elements. In order to prevent this undesired effect, in the Sparsenet algorithm the elements of the dictionary are normalized after the update of the dictionary has been performed. This normalization scales the elements of the dictionary such that a desired variance of the hidden variables $\mathbf{a}$ is obtained.

4.7.2 Maximum a posteri methods

In order to avoid the problematic integration in (4.84) with respect to the coefficients of the dictionary, Kreutz-Delgado et al. [2003] proposed the column normalized version of the FOCUSS algorithm for dictionary learning (FOCUSS-CNDL), which jointly maximizes the posterior probabilities of the model parameters C and $\mathbf{a}$

$$\max_{C,\mathbf{a}_1,\ldots,\mathbf{a}_L} P(C, \mathbf{a}_1, \ldots, \mathbf{a}_L | \mathbf{x}_1, \ldots, \mathbf{x}_L) \ . \tag{4.87}$$

This is equivalent to

$$\max_{C,\mathbf{a}_1,\ldots,\mathbf{a}_L} P(\mathbf{x}_1, \ldots, \mathbf{x}_L | C, \mathbf{a}_1, \ldots, \mathbf{a}_L) P(\mathbf{a}_1, \ldots, \mathbf{a}_L) P(C) \ . \tag{4.88}$$

Theoretically, the introduction of a prior with respect to C allows to directly incorporate constraints on the dictionary into the probabilistic setting. However, the learning rule of the FOCUSS-CNDL algorithm that is derived in [Kreutz-Delgado et al., 2003] is based on the assumption of a uniform prior in the solution space. The joint optimization with respect to C and $\mathbf{a}$ is performed by a parallel update of these variables. This means that initially a rather coarse optimization with respect to the coefficients is performed which is successively refined while being interrupted by optimization steps with respect to the dictionary. Similar to the Sparsenet algorithm, also in the FOCUSS-CNDL algorithm the column normalization is performed after the dictionary update has been applied.

All the probabilistic methods that have been presented in this section impose a regularization or penalty on the dictionary coefficients that is not scaling invariant. Hence, in order to obtain a feasible solution, a constraint on the norm of the dictionary elements is given. In all these algorithms the constrained optimization with respect to the dictionary is implemented by an unconstrained update of the dictionary that is supplemented by a projection to the constrained solution space. A disadvantage of this projective approach is that it can cause slow convergence. The update mechanism proposed in [Lee et al., 2007], which was discussed in Section 4.3, overcomes this disadvantage. It directly solves the constrained optimization problem by a maximization of the corresponding Lagrangian and could be used in some of the methods.

5 Blind source separation

A central task in the domain of blind source separation is the so-called "Cocktail Party Problem" that is to follow party conversations, i.e., to separate several speakers from noise and to focus on a single voice. A comprehensive review on the difficulties associated with this problem is provided in [Haykin and Chen, 2005].

The "Cocktail Party Problem" has been tackled by a number of researchers in a mathematical framework that is closely related to the generative linear models that have been considered for dictionary learning in Chapter 4. We are given a sequence of observations $\mathbf{x}(1),\ldots,\mathbf{x}(t), \ldots$ with $\mathbf{x}(t) \in \mathbb{R}^N$ and according to the hypothesis that is used in this framework, these observations stem from a linear mixture of a number of unknown sources $\mathbf{a}(1),\ldots,\mathbf{a}(t),\ldots$ with $\mathbf{a}(t) \in \mathbb{R}^M$:

$$\mathbf{x}(t) = C\mathbf{a}(t)\ . \tag{5.1}$$

In this chapter, we present solution strategies for two different scenarios. In the first scenario, called batch scenario, the sequence of observations is finite $\mathbf{x}(1),\ldots, \mathbf{x}(t), \ldots, \mathbf{x}(L)$ and the mixing matrix $C \in \mathbb{R}^{N\times M}$ is considered to be time invariant. In the second scenario, called online scenario, the sequence of observations is infinite and the mixing matrix $C(t) \in \mathbb{R}^{N\times M}$ is considered to be time dependent. We may regard the observations $\mathbf{x}(t)$ as what we hear and the sources $\mathbf{a}(t)$ as the voices of M speakers at time t. The sequence $\mathbf{s}_j = a(1)_j,\ldots,a(t)_j,\ldots$ is the voice track of speaker j. Now, the central question is, if it is possible to blindly separate the sources $\mathbf{s}_j$ from the mixtures $\mathbf{x}(t)$ without knowledge about the mixing matrix C. This task involves two sub-tasks. First, one has to determine the mixing matrix C from the observations and second, one has to estimate the sources on the basis of this matrix.

Among the methods that have been proposed to estimate the statements $\mathbf{s}_j$ and the matrix C, there are those methods for independent component analysis that solely rely on the statistical independence of the sources [Bell and Sejnowski, 1995, Hyvärinen and Oja, 1997, Hyvärinen, 1999a]. These methods require only the mixtures $\mathbf{x}(t)$ to be known and that one can assume $M = N$, i.e., that we are given as many observations as sources. An advantage of these methods is that they only impose weak constraints on the distributions of the sources, on the other hand statistical independence of the sources is not always given, as for instance, within music or human conversations where the sources can be highly dependent. Section 5.1 provides a brief discussion on some aspects of a method which is called FastICA that belongs to this category of methods. It has been proposed in [Hyvärinen and Oja, 1997, Hyvärinen, 1999a] and is used for comparison purposes in some of the experiments that are presented in this chapter.

In many situations, the number of sources is not equal to the number of observations, i.e., often $M > N$ holds. Humans have two ears but a large number of persons may be present at a party. In this overcomplete case stronger assumptions on the sources such as sparsity have to be introduced in order to enable a successfull identification of the mixing matrix and the sources [Hyvärinen et al., 1999, Lee et al., 1999, Theis et al., 2004, Davies and Mitianoudis, 2004]. Due to the presence of a certain amount of additional background noise the problem may become even more difficult. A model that accounts for a certain amount of additive noise,

$$\mathbf{x}(t) = C\mathbf{a}(t) + \boldsymbol{\epsilon}(t) \quad \|\boldsymbol{\epsilon}(t)\| \leq \delta, \tag{5.2}$$

has also been considered in the past [Hyvärinen, 1999b]. The SCNG algorithm which already has been introduced for the application of dictionary learning in Section 4.5 can also be applied to (5.2) if it is slightly modified. How this is done is discussed in Section 5.2. This application of the SCNG algorithm has been proposed first in [Labusch et al., 2008b] and has been applied to demixing of music in [Labusch et al., 2009c].

An even more realistic setting is obtained by the introduction of a mixing matrix that is time dependent:

$$\mathbf{x}(t) = C(t)\mathbf{a}(t) + \boldsymbol{\epsilon}(t) \quad \|\boldsymbol{\epsilon}(t)\| \leq \delta \quad C(t) = (\mathbf{c}_1(t), \dots, \mathbf{c}_M(t)), \mathbf{c}_j(t) \in \mathbb{R}^N. \tag{5.3}$$

For instance, in the case of the cocktail party, (5.3) can account for party guests who change their position during conversation. In [Labusch et al., 2009b], we have shown that the SCNG algorithm can also be applied to (5.3). This application of the SCNG-BSS variant is described in Section 5.3. In this application the SCNG method can be used to process an infinite sequence of observations, since the observations are used for learning in an online pattern-by-pattern mode, i.e., each observation is presented only once to the learning algorithm and the sources are estimated immediately. We do not make assumptions regarding the type of noise but require that the underlying sources $\mathbf{s}_j$ are sufficiently sparse, in particular, we require that the $\mathbf{a}(t)$ are sparse, i.e., only a few persons talk at the same time, which is desireable in a conversation. The noise level δ and the number of sources M have to be known. For the time dependent mixing matrix, we require $\|\mathbf{c}_j(t)\| = 1$ without loss of generality.

5.1 Independent component analysis

Due to the amount of work that has been done in the domain of Independent Component Analysis (ICA), a comprehensive discussion of this domain is out of the scope of this work. For a good overview of the field, we refer to [Hyvärinen et al., 2001].

Here, we concentrate on FastICA, since it was used as a reference in our experiments, it is also refered to in the introduction, and it is one of the most often used approaches. Initially, it has been proposed in [Hyvärinen and Oja, 1997, Hyvärinen, 1999a]. It considers the setting of (5.1) where no additive noise is present and the number of observed mixtures is equal to the number of underlying sources, i.e., $M = N$:

$$\mathbf{x}(t) = C\mathbf{a}(t) \tag{5.4}$$

with $\mathbf{x}(t) \in \mathbb{R}^N$ and $C \in \mathbb{R}^{N \times N}$. FastICA solely relies on the assumption that the hidden variables $\mathbf{a}(t)$ are statistically independent and non-Gaussian.

FastICA looks for model parameters C, which maximize the non-Gaussianity of the estimated sources. This is due to the empirical observation that the Gaussianity of a random variable, that is a linear combination of a number of statistically independent random variables, tends to be larger than the Gaussianity of the original random variables. In case of original random variables that are statistically independent, identically distributed, and have finite means and variances, this observation is explained by the central limit theorem (which is discussed in detail for example in [Ross, 2002])). However, empirically this observation seems to be true even if some of the preconditions of the central limit theorem do not hold, for instance, if the original random variables do not have the same distribution. Hence, it has been proposed to perform ICA by the maximization of the non-Gaussianity of the estimated original random variables [Hyvärinen and Oja, 1997, Hyvärinen et al., 2001].

As it is discussed in Section 4.2, PCA, i.e., decorrelation, can be seen as a first step of ICA. Therefore a preprocessing step of FastICA is the decorrelation and scaling of the observations such that the obtained variables are white, i.e., have unit variance:

$$\mathbf{y}(t) = \Lambda^{-1} V \mathbf{x}(t) \ . \tag{5.5}$$

V is a matrix which contains the eigenvectors of the covariance matrix $E[\mathbf{x}(t)\mathbf{x}(t)^T]$ of the data which, in practice, is estimated by avaraging. Λ is a diagonal matrix that contains the standard deviation of the corresponding principal components (square root of the eigenvalues) on its main diagonal. FastICA now looks for an additional linear mapping $W = (\mathbf{w}_1, \ldots, \mathbf{w}_N)^T$,$\mathbf{w}_j \in \mathbb{R}^N$ such that $C^{-1} = W\Lambda^{-1}V$ and therefore

$$\mathbf{a}(t) = W\Lambda^{-1}V\mathbf{x}(t) = W\mathbf{y}(t) \ . \tag{5.6}$$

The assumption that the hidden variables $\mathbf{a}(t)$ are statistically independent means in particular that they are white, i.e., their covariance matrix is equal to the unit matrix

$$E[\mathbf{a}(t)\mathbf{a}(t)^T] = \mathrm{diag}(\mathbf{1}) \ . \tag{5.7}$$

The whitening operation (5.5) ensures that the matrix W is constrained to being just an additional rotation, i.e., it is an orthonormal matrix:

$$\begin{aligned} E[\mathbf{a}(t)\mathbf{a}(t)^T] &= E\left[W\Lambda^{-1}V\mathbf{x}(t)\left(W\Lambda^{-1}V\mathbf{x}(t)\right)^T\right] \\ &= WE\left[\Lambda^{-1}V\mathbf{x}(t)\left(\Lambda^{-1}V\mathbf{x}(t)\right)^T\right]W^T \\ &= WW^T = \mathrm{diag}(\mathbf{1}) \ . \end{aligned}$$

FastICA now looks for a rotation W such that the non-Gaussianity of each source $a(t)_j$ is maximized. Since W is an orthonormal matrix, each of the sources can be considered separately, which is equivalent to a subsequent estimation of a single row of W. In order to determine a row of W, the observations $\mathbf{x}(t)$ are orthogonalized with respect to those rows of W that already have been obtained. The negentropy of a random variable is a good measure of its non-Gaussianity (as it is discussed for example in [Hyvärinen et al., 2001]). Hence, FastICA maximizes the negentropy of the source $a(t)_j$ in order to determine $\mathbf{w}_j$. Since the negentropy of a random variable is difficult to compute, an approximation of the negentropy is maximized, the so-called contrast function. The contrast function is chosen such that its maximization can be efficiently performed within a fixed-point iteration.

5.2 Sparse Coding Neural Gas for the Separation of Noisy Overcomplete Sources

Suppose that we determine the parameter C of the mixture model (5.2) for given mixtures $\mathbf{x}(1), \dots, \mathbf{x}(L)$ so that it is the solution of the following minimization problem

$$\min_C \sum_{t=1}^{L} \left(\min_{\mathbf{a}} \|\mathbf{x}(t) - C\mathbf{a}\| + \lambda S(\mathbf{a}) \right) . \tag{5.8}$$

This can be interpreted as independent component analysis [Kreutz-Delgado and Rao, 1999], if the regularization term $S(\mathbf{a})$ that is used in (5.8), fulfills the following conditions:

1. $S(\mathbf{a})$ has the property of separability:
$$S(\mathbf{a}) = \sum_{j=1}^{M} f(a_j) \tag{5.9}$$
2. $f(a_j)$ can be understood as the negative logarithm of some prior density of the source $\mathbf{a}_j$.

For instance in the Sparsenet method, which is discussed in Section 4.7, both above conditions, 1 and 2, are fulfilled. (5.8) can be seen as the Lagrangian dual of the following optimization problem

$$\min_C \sum_{t=1}^{L} \left(\min_{\mathbf{a}} S(\mathbf{a}) \quad \text{subject to} \quad \|\mathbf{x}(t) - C\mathbf{a}\| < \delta \right) \tag{5.10}$$

as explained in Section 2.2 for the case $S(\mathbf{a}) = \|\mathbf{a}\|_1$. This means that the determination of the solution of (5.10) is equivalent to the determination of the solution of (5.8) if δ and λ are appropriately chosen.

Here, we use $S(\mathbf{a}) = \|\mathbf{a}\|_0$ as measure of sparsity, which yields

$$\min_C \sum_{t=1}^{L} \left(\min_{\mathbf{a}} \|\mathbf{a}\|_0 \quad \text{subject to} \quad \|\mathbf{x}(t) - C\mathbf{a}\| < \delta \right) . \tag{5.11}$$

Due to

$$\|\mathbf{a}\|_0 = \sum_{j=1}^{M} \|a_j\|_0 \ , \tag{5.12}$$

this regularization term has also the property of separability. However, it cannot be interpreted as the negative logarithm of some prior density. Therefore, the method that is proposed in this section is closely related to ICA but cannot be interpreted in a probabilistic framework.

We begin with the batch case where a finite sequence of observations $\mathbf{x}(1), \ldots, \mathbf{x}(L)$ is given and the mixing matrix is time independent. We want to determine the mixing matrix $C = (\mathbf{c}_1, \ldots, \mathbf{c}_M)$ in (5.2) from the mixtures $\mathbf{x}(t)$ provided that we know the noise level δ and the number of underlying sources M. In this section, for simplicity, we denote $\mathbf{x}(j)$ as $\mathbf{x}_j$. We tackle this task with a variant of the SCNG algorithm that is modified so that it provides an approximative solution to (5.11).

In the SCNG algorithm the estimation of the configuration of the hidden variables $\mathbf{a}$ is obtained from the OOMP approximation method which is an improved variant of OMP (details of OMP and OOMP are discussed in Section 2.1.3). Donoho et al. [2006] have reported results which show that the sources actually can be identified with OMP given that they are sparse enough and that the mixing matrix C is incoherent enough. Details of these results can be found in Section 2.3.

As a consequence of the fact that the SCNG algorithm employs the OOMP method in order to estimate the configuration of the hidden variables $\mathbf{a}$, we are looking for a mixing matrix C that minimizes the number of non-zero entries of $\mathbf{a}_j^{\mathrm{OOMP}}$, i.e., the number of iteration steps of the OOMP algorithm, given a noise level δ

$$\min_C \sum_{j=1}^{L} \|\mathbf{a}_j^{\mathrm{OOMP}}\|_0 \quad \text{subject to} \quad \forall j \ : \ \|\mathbf{x}_j - C\mathbf{a}_j^{\mathrm{OOMP}}\| \leq \delta \ . \tag{5.13}$$

In order to solve (5.13), we perform an update of R and C prior to the construction step (2.5) and (2.6) in each iteration of the OOMP algorithm with respect to the currently used sample $\mathbf{x}_j$. The total number of OOMP iterations is reduced by minimization of the norm of the residual that is going to be obtained after the update has been performed. The norm of the residual becomes small if

$$(\mathbf{r}_{l_{\mathrm{win}}}^T \boldsymbol{\epsilon}_j^U)^2 \tag{5.14}$$

is large where $\mathbf{r}_{l_{\text{win}}}$ and $\boldsymbol{\epsilon}_j^U$ are the winning column in the current iteration of the OOMP method and the current residual with respect to the observation $\mathbf{x}_j$ (see Section 2.1.3 for details). Hence, we have to consider the optimization problem

$$\max_R \sum_{j=1}^{L} \max_{l, l \notin U} (\mathbf{r}_l^T \boldsymbol{\epsilon}_j^U)^2 \quad \text{subject to} \quad \|\mathbf{r}_l\| = 1 \; . \tag{5.15}$$

The method which minimizes (5.15) in order to determine the mixing matrix is allowed to iterate as often as necessary through the entire set of observations. We perform $t = 1, \ldots, t_{\max}$, $t_{\max} \geq L$ learning iterations and in each iteration a new sample $\mathbf{x}_t$ is selected from the set observations. Then, as it was already discussed in Section 4.5, the maximization problem (5.15) can be approached with Oja's rule, which is

$$\mathbf{r}_{l_{\text{win}}} = \mathbf{r}_{l_{\text{win}}} + \alpha \, y (\boldsymbol{\epsilon}_t^U - y \, \mathbf{r}_{l_{\text{win}}}) \tag{5.16}$$

with $y = \mathbf{r}_{l_{\text{win}}}^T \boldsymbol{\epsilon}_t^U$ and learning rate α. Instead of a sole update of the winning column of R, i.e., $\mathbf{r}_{l_{\text{win}}}$, we again employ the soft-competitive learning approach of the "Neural Gas" (NG) algorithm (see also Section 4.1.3) in order to update each column of R that might be selected in the next iteration of the OOMP algorithm. Hence, we determine the sequence

$$-\left(\mathbf{r}_{l_0}^T \boldsymbol{\epsilon}_t^U\right)^2 \leq \cdots \leq -\left(\mathbf{r}_{l_k}^T \boldsymbol{\epsilon}_t^U\right)^2 \leq \cdots \leq -\left(\mathbf{r}_{l_{M-|U|}}^T \boldsymbol{\epsilon}_t^U\right)^2 \quad , \; l_k \notin U \tag{5.17}$$

and combine Oja's rule with the soft-competitive update of the NG algorithm. We update the columns of the mixing matrix according to

$$\Delta \mathbf{r}_{l_k} = \Delta \mathbf{c}_{l_k} = \alpha_t e^{-k/\lambda_t} y \left(\boldsymbol{\epsilon}_t^U - y \, \mathbf{r}_{l_k}\right) . \tag{5.18}$$

Again, α_t and λ_t are the exponentially decreasing learning rate and neighborhood size at time t:

$$\lambda_t = \lambda_0 \left(\lambda_{\text{final}} / \lambda_0\right)^{t/t_{\max}} \tag{5.19}$$

$$\alpha_t = \alpha_0 \left(\alpha_{\text{final}} / \alpha_0\right)^{t/t_{\max}} \; . \tag{5.20}$$

In Section 4.5.1, we have shown that this update rule corresponds to a stochastic gradient descent with respect to

$$\max_R \sum_{j=1}^{L}\sum_{l=1}^{M} h_{\lambda_t}(k(\mathbf{r}_l,\boldsymbol{\epsilon}_j))(\mathbf{r}_l^T\boldsymbol{\epsilon}_j)^2 \quad \text{subject to} \quad \|\mathbf{r}_l\|_2^2 = 1, \tag{5.21}$$

with $h_{\lambda_t}(v) = e^{-v/\lambda_t}$. Here $k(\mathbf{r}_l,\boldsymbol{\epsilon}_j)$ denotes the number of columns of the temporary matrix R with $(\mathbf{r}_j^T\boldsymbol{\epsilon}_j)^2 > (\mathbf{r}_l^T\boldsymbol{\epsilon}_j)^2$. Note that for $\lambda_t \to 0$, (5.21) is equivalent to (5.15). For $t \to t_{\max}$ one obtains equation (5.16) as update rule. Note that (5.18) accumulates the updates of all iterations in the learned mixing matrix C. Due to the orthogonal projections (2.5) and (2.6) performed in each iteration of the OOMP algorithm, these updates are pairwise orthogonal. Furthermore, note that the columns of the original mixing matrix emerge in random order in the learned mixing matrix. The sign of the columns of the mixing matrix $\mathbf{c}_l$ cannot be determined because multiplication of $\mathbf{c}_l$ by -1 corresponds to multiplication of $\mathbf{r}_l$ by -1, which does not change (5.21). The entire SCNG method for blind source separation is shown in Algorithm 4.

5.2.1 Experiments

In order to evaluate the performance of the SCNG algorithm with respect to the reconstruction of the underlying sources, we performed a number of experiments on synthetical data. We generated sparse underlying sources $S = (\mathbf{s}_1,\ldots,\mathbf{s}_M)^T = (\mathbf{a}_1,\ldots,\mathbf{a}_L), \mathbf{s}_i \in \mathbb{R}^L, \mathbf{a}_j \in \mathbb{R}^M$. This was done by setting up to k entries of the $\mathbf{a}_j$ to uniformly distributed random values in $[-1,1]$. For each $\mathbf{a}_j$ the number of non-zero entries was obtained from a uniform distribution in $[0,k]$. We added Gaussian distributed noise $\boldsymbol{\epsilon}_j$ such that

$$\mathbf{x}_j = C\mathbf{a}_j + \boldsymbol{\epsilon}_j \ . \tag{5.22}$$

The noise parameter δ was obtained as

$$\delta = \frac{1}{L}\sum_{j=1}^{L}\|\boldsymbol{\epsilon}_j\| \ . \tag{5.23}$$

We scaled S and thereby the $\mathbf{a}_j$ so that $var(CS) = 1$. Then, the amplitude of the values in $\boldsymbol{\epsilon}_j$ was chosen such that the desired SNR was obtained. Again, in order

Algorithm 4 The sparse coding neural gas algorithm for source separation.

```
initialize $C = (\mathbf{c}_1, \dots, \mathbf{c}_M)$ using uniform random values
for $t = 0$ to $t_{\max}$ do
  select random sample $\mathbf{x}$ out of $X$
  set $\mathbf{c}_1, \dots, \mathbf{c}_M$ to unit length
  calculate current size of neighborhood: $\lambda_t = \lambda_0 \, (\lambda_{\text{final}}/\lambda_0)^{t/t_{\max}}$
  calculate current learning rate: $\alpha_t = \alpha_0 \, (\alpha_{\text{final}}/\alpha_0)^{t/t_{\max}}$
  set $U = \emptyset$, $\boldsymbol{\epsilon}^U = \mathbf{x}$ and $R = (\mathbf{r}_1, \dots, \mathbf{r}_M) = C = (\mathbf{c}_1, \dots, \mathbf{c}_M)$
  while $\|\boldsymbol{\epsilon}^U\| > \delta$ do
    determine $l_0, \dots, l_k, \dots, l_{M-|U|}$ with $l_k \notin U$ :
        $$-(\mathbf{r}_{l_0}^T \boldsymbol{\epsilon}^U)^2 \le \dots \le -(\mathbf{r}_{l_k}^T \boldsymbol{\epsilon}^U)^2 \le \dots \le -(\mathbf{r}_{l_{M-|U|}}^T \boldsymbol{\epsilon}^U)^2$$
    for $k = 1$ to $M - |U|$ do
      with $y = \mathbf{r}_{l_k}^T \boldsymbol{\epsilon}^U$ update $\mathbf{c}_{l_k} = \mathbf{c}_{l_k} + \Delta_{l_k}$ and $\mathbf{r}_{l_k} = \mathbf{r}_{l_k} + \Delta_{l_k}$ with
        $$\Delta_{l_k} = \alpha_t e^{-k/\lambda_t} y \, (\boldsymbol{\epsilon}^U - y \, \mathbf{r}_{l_k})$$
      set $\mathbf{r}_{l_k}$ to unit length
    end for
    determine $l_{\text{win}} = \arg\max_{l \notin U} (\mathbf{r}_l^T \boldsymbol{\epsilon}^U)^2$
    remove projection to $\mathbf{r}_{l_{\text{win}}}$ from $\boldsymbol{\epsilon}^U$ and $R$:
        $$\boldsymbol{\epsilon}^U = \boldsymbol{\epsilon}^U - (\mathbf{r}_{l_{\text{win}}}^T \boldsymbol{\epsilon}^U) \mathbf{r}_{l_{\text{win}}}$$
        $$\mathbf{r}_l = \mathbf{r}_l - (\mathbf{r}_{l_{\text{win}}}^T \mathbf{r}_l) \mathbf{r}_{l_{\text{win}}} \quad , l = 1, \dots, M$$
    set $U = U \cup l_{\text{win}}$
  end while
end for
```

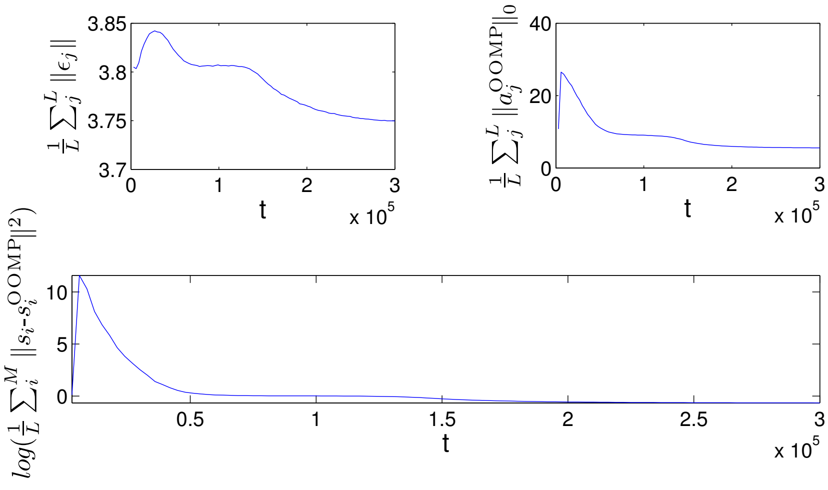


Figure 5.1: The figure shows the convergence of the SCNG algorithm. **Top left:** The mean norm of the residual $\boldsymbol{\epsilon}_j^U$ of the last iteration of the OOMP algorithm. **Top right:** Mean number of iterations performed by the OOMP algorithm until $\boldsymbol{\epsilon}_j^U \leq \delta$. **Bottom:** Logarithmic plot of the mean squared distance between the estimated sources s_i^{OOMP} and the true sources s_i. We used $M = 100$, $N = 50$, $H(C) = 0.4$, $k = 15$ and $SNR = 5dB$ which corresponds to $\delta = 3.96$. The fraction of non-zero entries in the sources was 7.5%, i.e., the avarage number of non-zero entries of the vectors $\mathbf{a}_j$ was 7.5.

to obtain a random mixture matrix $C \in \mathbb{R}^{N\times M}$ with coherence z, we repeatedly chose a matrix from a uniform distribution in $[-1,1]$ until $\lceil 100H(C)\rceil = \lceil 100z\rceil$. The norm of the columns of the mixture matrix was normalized to unit length.

In our experiments, we studied the error on the representation level. This means that for each observation $\mathbf{x}_j$, we evaluated the difference between the original contributions of the underlying sources, i.e., $\mathbf{a}_j$ to $\mathbf{x}_j$, and the contributions $\mathbf{a}_j^{\mathrm{OOMP}}$ that were estimated by the OOMP algorithm on the basis of the mixing matrix C that was learned by the SCNG algorithm:

$$\frac{1}{L}\sum_{j=1}^{L}\|\mathbf{a}_j - \mathbf{a}_j^{\mathrm{OOMP}}\|_2^2 = \frac{1}{L}\sum_{i=1}^{M}\|\mathbf{s}_i - \mathbf{s}_i^{\mathrm{OOMP}}\|_2^2. \tag{5.24}$$

Here $S^{\mathrm{OOMP}} = (\mathbf{s}_1^{\mathrm{OOMP}},\ldots,\mathbf{s}_M^{\mathrm{OOMP}})^T = (\mathbf{a}_1^{\mathrm{OOMP}},\ldots,\mathbf{a}_L^{\mathrm{OOMP}})$ are the underlying sources that were obtained from the OOMP algorithm. In order to evaluate (5.24), we had to assign the entries in $\mathbf{a}_j^{\mathrm{OOMP}}$ to the entries in $\mathbf{a}_j$, which is equivalent to assignment of the original sources $\mathbf{s}_i$ to the estimated sources $\mathbf{s}_i^{\mathrm{OOMP}}$. This problem arises due to the random order in which the columns of the original mixing matrix appear in the learned mixing matrix. For the assignment, we performed the following procedure:

1. Set $I_{\mathrm{orig}} : \{1,\ldots,M\}$ and $I_{\mathrm{learned}} : \{1,\ldots,M\}$.
2. Find and assign $\mathbf{s}_i$ and $\mathbf{s}_j^{\mathrm{OOMP}}$ with $i \in I_{\mathrm{orig}}, j \in I_{\mathrm{learned}}$ such that
$$\frac{|\mathbf{s}_j^{\mathrm{OOMP}}\mathbf{s}_i^T|}{\|\mathbf{s}_i\|\|\mathbf{s}_j^{\mathrm{OOMP}}\|} \text{ is maximal.}$$
3. Remove i from I_{orig} and j from I_{learned}.
4. If $\mathbf{s}_j^{\mathrm{OOMP}}\mathbf{s}_i^T < 0$ set $\mathbf{s}_j^{\mathrm{OOMP}} = -\mathbf{s}_j^{\mathrm{OOMP}}$.
5. Proceed with (2) until $I_{\mathrm{orig}} = I_{\mathrm{learned}} = \emptyset$.

For all experiments, we used $L = 20000$ observed mixtures, an initial learning rate $\alpha_0 = 0.1$, a final learning rate $\alpha_{\mathrm{final}} = 0.0001$, an initial neighborhood-size $\lambda_0 = M/2$, and a final neighborhood-size $\lambda_{\mathrm{final}} = 10^{-7}$. We repeated all experiments 10 times and report the mean result over the 10 runs. The number of learning iterations of the SCNG algorithm was set to $t_{\max} = 15 * 20000$.

In our first experiment, we evaluated the convergence of the SCNG algorithm over time in case of $N = 50$ observations of $M = 100$ underlying sources with up to $k = 15$ non-zero entries. The result is shown in Figure 5.1. We chose a SNR of $5dB$, which corresponds to $\delta = 3.96$. The norm of the residual of the final iteration $\boldsymbol{\epsilon}_j^U$ of the OOMP algorithm converged to a value close to δ. The number of iterations of the OOMP algorithm decreased over time and converged to a value close to the fraction of non-zero entries in the entire sources, which is 7.5%. Since the number of non-zero entries was uniformly chosen between 0 and 15 and $M = 100$ sources were used, the fraction of non-zero entries is equal to the average number of non-zero entries of the vectors $\mathbf{a}_j$. At the same time, also the error on the representation level is minimized. The 5 underlying sources that were estimated best as well as one of the mixtures from which they were obtained are shown in Figure 5.2.

In the next experiment, we used $N = 20$ observed mixtures, $M = 40$ underlying sources, and a SNR of $10dB$. We varied the coherence $H(C)$ of the mixing matrix and k, the number of non-zero entries of the underlying sources. The result is shown in Figure 5.4. The sparser the sources are and the smaller the coherence of the mixing matrix is, the better the obtained performance becomes. Then, we fixed $H(C) = 0.6$, $SNR = 20dB$, and $k = 5$ and varied the overcompleteness by setting $M = 20, \dots, 80$. In Figure 5.4, it can be seen that only slightly varying performance was obtained though the overcompleteness strongly increases. Furthermore, we varied N from 10 to 50, chose $M = 2N$, and $k = \lceil N/10 \rceil$. Figure 5.4 shows that almost the same performance was obtained, i.e., the obtained performance does not depend on the number of sources and observations if the fraction N/M as well as the sparseness of the sources remains constant.

The results that are shown in Figure 5.5 were obtained by choosing $N = 20$, $M = 40$, $H(C) = 0.6$, and varying the noise level as well as the sparseness of the sources. As expected, the more noise was present and the less sparse the sources were, the lower the obtained performance is. Finally, we set $k = 5$ and studied the obtained reconstruction performance depending on the coherence of the mixing matrix and the noise level. The result is also shown in Figure 5.5. It can be seen that in our experimental setting the noise level has a strong impact on the performance. The influence of the noise cannot be compensated by a smaller mutual coherence of the mixing matrix.

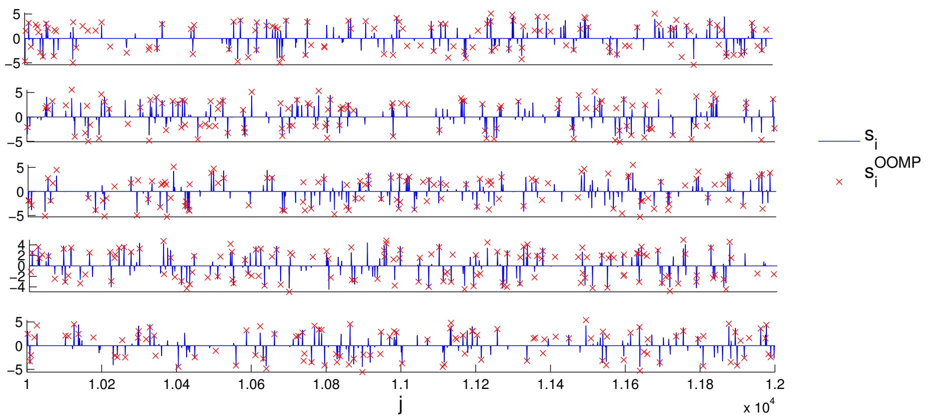


Figure 5.2: The figure shows the 5 out of $M = 100$ underlying sources $\mathbf{s}_i$ that were reconstructed best from the $N = 50$ mixtures observed. One of the mixtures is shown in Figure 5.3. The solid line depicts $\mathbf{s}_i$ whereas the crosses depict the estimated sources $\mathbf{s}_i^{\mathrm{OOMP}}$ that were obtained by applying the OOMP to the mixtures using the mixing matrix that was learned by the SCNG algorithm. The coherence of the mixture matrix was set to $H(C) = 0.4$. The data was generated using $k = 15$, $SNR = 5dB$. We used $\delta = 3.96$.

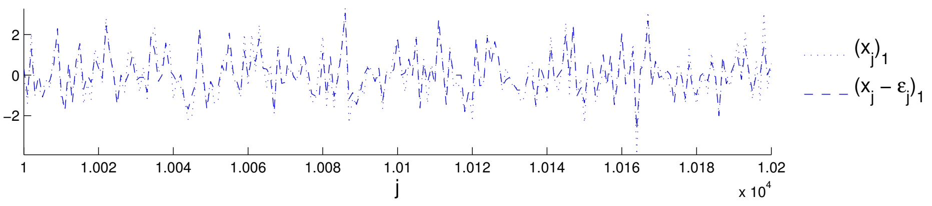


Figure 5.3: The figure shows one out of 50 mixtures from which the estimated sources that are shown in Figure 5.2 were obtained.
The dashed line depicts $(\mathbf{x}_j - \boldsymbol{\epsilon}_j)_1$ whereas the dotted line depicts $(\mathbf{x}_j)_1$.

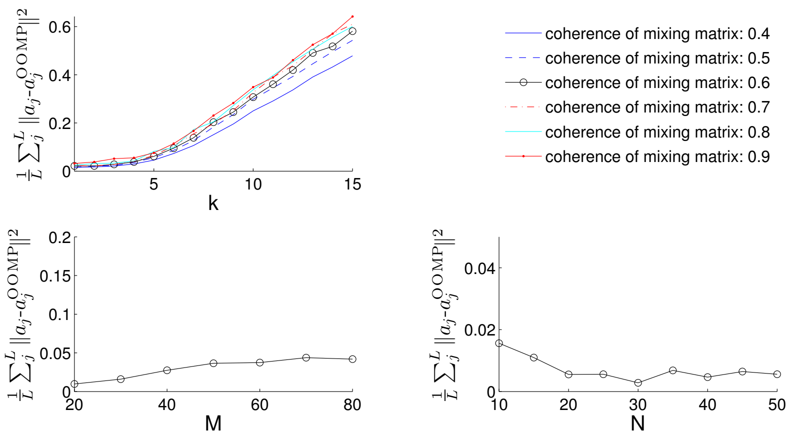


Figure 5.4: **Top left:** The influence of decreasing sparseness and increasing coherence of the mixing matrix with respect to the reconstruction error is shown. We used $N = 20$, $M = 40$, and $SNR = 10dB$. **Bottom left:** The obtained reconstruction error using $M = 20, \ldots, 80$,$SNR = 20dB$ and $k = 5$. **Bottom right:** The obtained reconstruction error for $N = 10, \ldots, 50$ with $M = 2N$,$k = \lceil N/10 \rceil$ and $SNR = 20dB$.

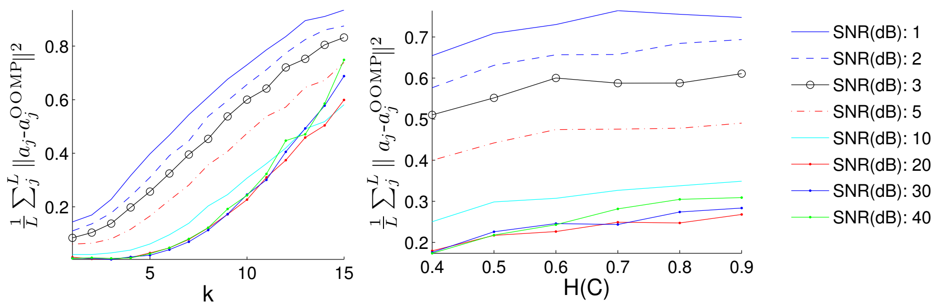


Figure 5.5: The obtained error with respect to the reconstruction of the sources depending on the noise level and the coherence of the mixing matrix is shown. **Left:** We used $N = 20, M = 40$. The coherence of the mixing matrix was set to 0.6. The reconstruction performance depending on the noise level as well as on the sparseness is shown. **Right:** The sparseness parameter k was set to 5.

5.2.2 An application to real world data: Demixing Jazz music

So far, we only have considered synthetical data in our experiments. In order to show that the SCNG algorithm can also be applied successfully to more realistic data, we applied the algorithm to audio data in a difficult setting.

Let us assume that we are given a recording of a piece of Jazz-music. The given piece is arranged for five instruments, vibraphone, piano, trumpet, percusion and bass. It is recorded with two microphones. Unfortunately, some additive noise is also present during the recording. Now, we want to reconstruct the part of each single instrument given only the recordings of the microphones.

In order to implement the setting described above, we took a Jazz midi-file and extracted the five instruments from that file. The obtained five audio tracks each containing a single instrument were used as ground truth. We obtained the simulated two-channel recording by applying a 2×5 mixing matrix to the five audio tracks and added some Gaussian noise to the mixture. Figure 5.6 shows the two recorded channels that were obtained. Then, we applied the SCNG algorithm to the recordings in order to estimate the mixing matrix.

In the same way as already described before, we then used the OOMP algorithm in order to estimate the original audio track of each instrument from the recordings. Figure 5.7 shows the original audio tracks and the estimated track for each instrument. It can be seen that one can clearly assign estimated and original part of each instrument. The auditory impression of the estimated parts is quite noisy but one can clearly assign each part to an instrument. However, periods of silence are estimated rather badly as can be seen for example from the percussion track. The result in this very difficult setting is promising and it might be improved by employing computationally more demanding methods such as basis pursuit using the mixing matrix provided by the SCNG method in order to reconstruct the tracks of the instruments.

5.2.3 Comparison to FastICA

Finally, we wanted to compare our method to a well-known method that also has been used to perform blind source separation, the FastICA algorithm which is briefly introduced in Section 5.1. The standard FastICA algorithm is not applicable in case of an overcomplete setting. Therefore, in our experiment, there are as many mixtures as sources, i.e., the mixing matrix is invertible. We wanted to study what the impact of additive noise on the performance of both methods with respect to

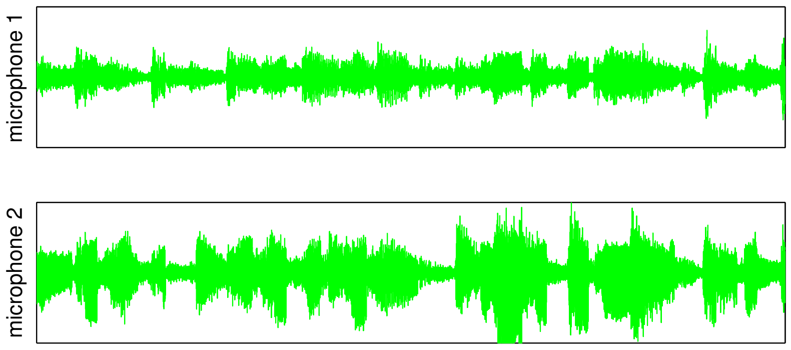


Figure 5.6: The figure shows the two recorded channels that have been obtained by applying a 2×5 mixing matrix to the tracks of the instruments that are shown in Figure 5.7. Gaussian noise has been added to the recorded channels. The coherence of the mixing matrix was set to 0.5. The SNR was set to $10dB$.

the determination of the mixing matrix is.

We randomly chose a mixing matrix $C^{\text{orig}} \in \mathbb{R}^{20 \times 20}$ with mutual coherence 0.5. As described before, we generated sparse sources $\mathbf{a}_j \in \mathbb{R}^{20}$ where the maximum number of non-zero entries of the sources $\mathbf{a}_j$ was set to 5. We applied the mixing matrix to the sources in order to obtain the mixtures $\mathbf{x}_j \in \mathbb{R}^{20}$. We employed the SCNG algorithm and the FastICA algorithm in order to obtain the estimated mixing matrices C^{SCNG} and C^{FICA}. We then compared the estimated mixing matrix to the original mixing matrix by computing the maximum overlap between each column of the original mixing matrix and the learned mixing matrix. For instance, in case of the solution provided by the SCNG algorithm, we computed

$$\max_j \left(1 - |\mathbf{c}_i^{\text{orig}} \mathbf{c}_j^{\text{SCNG}}|\right). \tag{5.25}$$

Whenever (5.25) was smaller than some accuracy level, we counted this as a success.

We repeated this experiment 9 times with varying SNR including zero noise. For each noise level, we sorted the 9 trials according to the number of successfully learned columns of the mixing matrix and ordered them in groups of 3 experiments. Figure 5.8 shows the mean number of successfully detected columns for each of the 3 groups for each noise level. We performed the experiment twice using accuracy

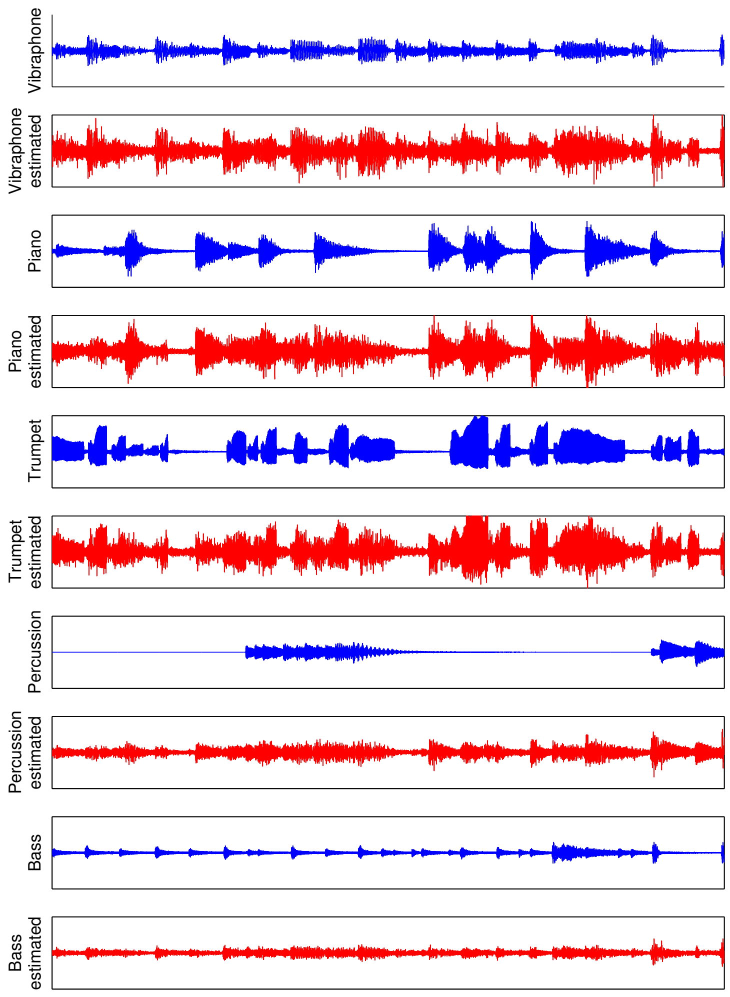


Figure 5.7: The figure shows the original audio track of each instrument as well as the estimated audio track that has been obtained by application of the OOMP algorithm to the two-channel recording that is shown in Figure 5.6 using the mixing matrix that was learned by the SCNG algorithm.

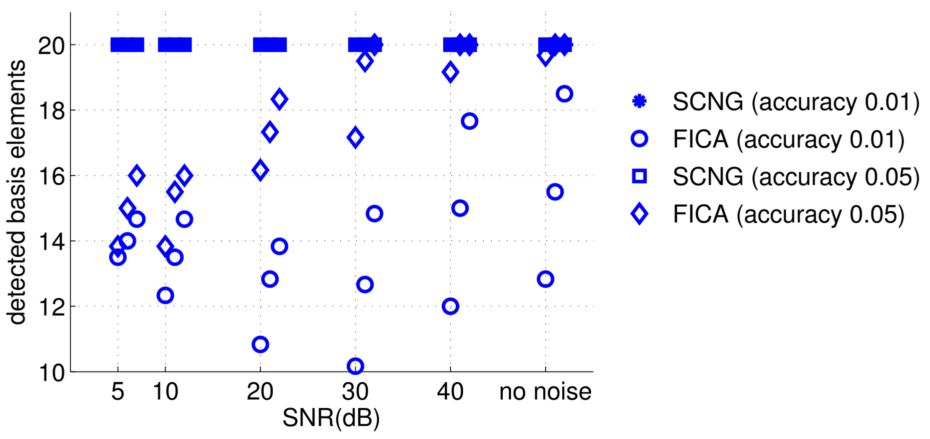


Figure 5.8: The figure shows the reconstruction performance of SCNG and FastICA on synthetically generated data, i.e., the number of successfully learned columns of the mixing matrix up to an accuracy of 0.01 and 0.05 respectively. We set $M = 20$, $N = 20$ and $k = 5$. The coherence of the mixing matrix was set to 0.5. We varied the SNR. In particular in case of the presence of strong noise SCNG outperforms FastICA.

levels of 0.01 and 0.05. In Figure 5.8, it can be seen that the SCNG algorithm is not sensitive to the additive noise, whereas the performance of the FastICA method degrades with increasing noise level.

In the next experiment, we set the noise level to $10dB$ and varied the number of non-zero entries of the sources $\mathbf{a}_j$. The results are shown in Figure 5.9. Though the performance of the SCNG algorithm degrades for $k = 9$, it outperforms the FastICA method on this noisy data.

In the last experiment, we used the audio data described in Section 5.2.2. For the audio data, a randomly chosen mixing matrix $C^{\text{orig}} \in \mathbb{R}^{5\times 5}$ with mutual coherence 0.5 was used. The results on the audio data are shown in Figure 5.10. It can be seen that for an accuracy level of 0.01 FastICA outperforms the SCNG method. The opposite result is obtained for accuracy 0.05. The audio data are less sparse than the data used in the synthetical setting, which might explain the reduced performance of the SCNG algorithm.

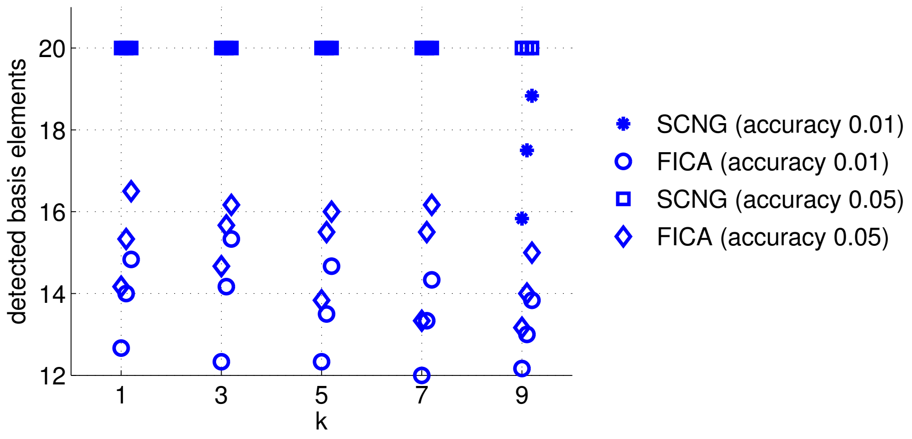


Figure 5.9: The figure shows the reconstruction performance of SCNG and FastICA on synthetically generated data, i.e., the number of successfully learned columns of the mixing matrix up to an accuracy of 0.01 and 0.05, respectively. We set $M = 20$, $N = 20$ and $SNR = 10dB$. The coherence of the mixing matrix was set to 0.5. We varied the parameter k which controls the sparseness of the underlying sources. The larger k is, the less sparse the sources are. Again SCNG outperforms FastICA. For large k the performance of SCNG degrades.

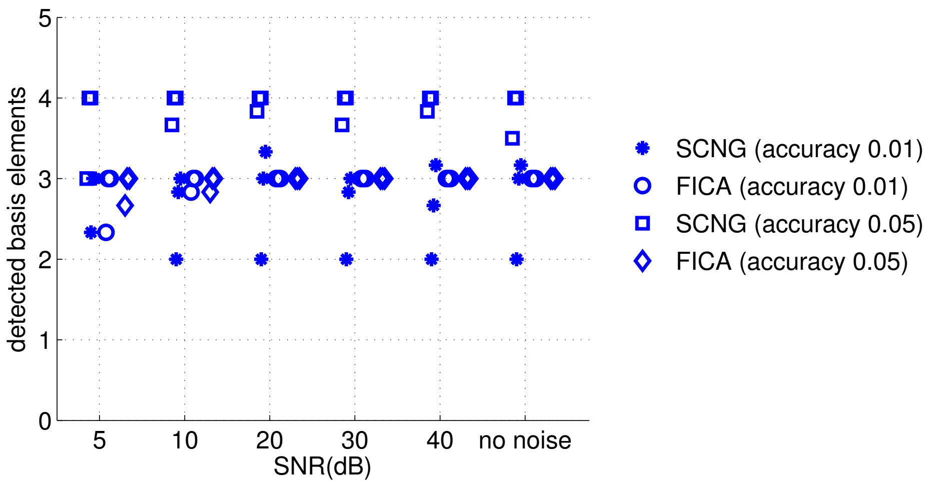


Figure 5.10: The figure shows the reconstruction performance of SCNG and FastICA on the audio data that is described in Section 5.2.2, i.e., the number of successfully learned columns of the mixing matrix up to an accuracy of 0.01 and 0.05, respectively. We set $M = 5$, $N = 5$. The coherence of the mixing matrix was set to 0.5. We varied the SNR. For accuracy 0.01 FastICA outperforms SCNG. For accuracy 0.05 one obtains the opposite result. In case of strong noise ($SNR = 5dB$) SCNG performs better than FastICA.

5.3 Approaching the Time Dependent Cocktail Party Problem

In Section 5.2, the mixing matrix was considered to be time invariant. Now, we want to estimate a mixing matrix that is time dependent, i.e., $C(t)$ where $C(t) \in \mathbb{R}^{M \times N}$. $C(t)$ is estimated from a potentially infinite sequence of mixtures $\mathbf{x}(1), \ldots, \mathbf{x}(t), \ldots$ under the assumption that the noise level δ is constant and known and that the number of underlying sources M is known. Furthermore, we assume that the mixing matrix changes only slowly over time such that $C(t)$ is approximately constant for some time interval $[t-T, t]$.

As in the time invariant case, we look for a mixing matrix that minimizes the number of iteration steps required by the OOMP algorithm to approximate an observation $\mathbf{x}(t)$ up to the noise level δ. However, now we do not consider all observations but only the time interval $[t-T, t]$:

$$\min_{C(t)} \sum_{t'=t-T}^{t} \|\mathbf{a}(t')^{\mathrm{OOMP}}\|_0 \quad \text{subject to} \quad \|\mathbf{x}(t') - C(t)\mathbf{a}(t')^{\mathrm{OOMP}}\| \leq \delta \,. \tag{5.26}$$

Again, a small norm of the current residual $\boldsymbol{\epsilon}(t')^U$ reduces the number of OOMP iterations that have to be performed until the stopping criterion $\|\boldsymbol{\epsilon}(t')^U\| \leq \delta$ has been reached, which is equivalent to a small $\|\mathbf{a}(t')^{\mathrm{OOMP}}\|_0$. In order to minimize the norm of the residuals in each iteration of the OOMP algorithm and thereby the expression (5.26), we have to maximize $(\mathbf{r}_{l_{\mathrm{win}}}\boldsymbol{\epsilon}(t')^U)^2$. Therefore, we consider the following optimization problem

$$\max_{\mathbf{r}_1,\ldots,\mathbf{r}_M} \sum_{t'=t-T}^{t} \max_{l, l \notin U} (\mathbf{r}_l^T \boldsymbol{\epsilon}(t')^U)^2 \quad \text{subject to} \quad \|\mathbf{r}_l\| = 1 \,, \tag{5.27}$$

which is the same as (5.15) except for the time interval that is considered. Again, we maximize (5.27) by updating R and $C(t)$ prior to the construction step of the OOMP algorithm, i.e., (2.5) and (2.6). This is done with the SCNG update rule for blind source separation (5.18). Again, we use a exponentially decreasing learning rate

$$\alpha(t) = \alpha_0 \left(\alpha_{\mathrm{final}}/\alpha_0\right)^{t/t_{\max}} \,, \tag{5.28}$$

and neighbourhood-size

$$\lambda(t) = \lambda_0 \left(\lambda_{\text{final}}/\lambda_0\right)^{t/t_{\max}} \ . \tag{5.29}$$

What happens for $t > t_{\max}$? Assuming that after $t_{\max}$ learning steps have been performed the current learned mixing matrix is close to the true mixing matrix, we track the slowly changing true mixing matrix by setting $\alpha(t) = \alpha_{\text{final}}$ and $\lambda(t) = \lambda_{\text{final}}$.

5.3.1 Experiments

We performed a number of experiments on synthetical data in order to study whether the underlying sources can be reconstructed from the mixtures in an online setting. We considered sequences

$$\mathbf{x}(t) = C(t)\mathbf{a}(t) + \boldsymbol{\epsilon}(t),\ t = 1, \dots, L, \tag{5.30}$$

where $\|\boldsymbol{\epsilon}(t)\| \leq \delta$, $\mathbf{x}(t) \in \mathbb{R}^N$, $\mathbf{a}(t) \in \mathbb{R}^M$. The true mixing matrix $C(t)$ slowly changes from state C_{i-1} to state C_i in P time steps. We randomly chose a sequence of true mixing matrices $C_i,\ i = 1, \dots, \lceil L/P \rceil$ with entries taken from a uniform distribution. The columns of these mixing matrices were normalized to unit norm. At time t with $(i-1)P \leq t \leq iP$, the true mixing matrix $C(t)$ was chosen according to

$$C(t) = \left(1 - \frac{(t-(i-1)P)}{P}\right) C_{i-1} + \frac{(t-(i-1)P)}{P} C_i \ . \tag{5.31}$$

The norm of the columns of each true mixing matrix $C(t)$ was then again normalized to unit norm. The sources $\mathbf{a}(t)$ were obtained by setting at most k entries of the $\mathbf{a}(t)$ to uniformly distributed random values in $[-1, 1]$. For each $\mathbf{a}(t)$, the number of non-zero entries was obtained from a uniform distribution in $[0, k]$. Uniformly distributed noise $\mathbf{e}(t) \in \mathbb{R}^M$ in $[-1, 1]$ was added such that

$$\mathbf{x}(t) = C(t)(\mathbf{a}(t) + \mathbf{e}(t)) = C(t)\mathbf{a}(t) + \boldsymbol{\epsilon}(t) \ . \tag{5.32}$$

We wanted to assess the error that is obtained with respect to the recontruction of the sources. Hence, we evaluated the difference between the sources $\mathbf{a}(t)$ and the estimation $\mathbf{a}(t)^{\text{OOMP}}$ that was obtained from the OOMP algorithm on the basis of the mixing matrix $C^{learn}(t)$ that was provided by the SCNG algorithm:

$$\|\mathbf{a}(t) - \mathbf{a}(t)^{\mathrm{OOMP}}\|_2 \,. \tag{5.33}$$

With $(\mathbf{s}_1^{\mathrm{OOMP}}, \ldots, \mathbf{s}_M^{\mathrm{OOMP}})^T = (\mathbf{a}(1)^{\mathrm{OOMP}}, \ldots, \mathbf{a}(L)^{\mathrm{OOMP}})$, we denote the estimated underlying sources obtained from the OOMP algorithm. Similar to the time-invariant setting, we had to assign the entries in $\mathbf{a}(t)^{\mathrm{OOMP}}$ to the entries in $\mathbf{a}(t)$ in order to evaluate (5.33). This is equivalent to assigning the true sources $\mathbf{s}_j$ to the estimated sources $\mathbf{s}_j^{\mathrm{OOMP}}$. This problem arises due to the random order in which the columns of the true mixing matrix appear in the learned mixing matrix. Due to the time dependent mixing matrix, the assignment may change over time. In order to obtain an assignment at time t, we considered a window of size s_w:

$$(\mathbf{w}_1(t)^{\mathrm{OOMP}}, \ldots, \mathbf{w}_M(t)^{\mathrm{OOMP}})^T = (\mathbf{a}(t - s_w/2)^{\mathrm{OOMP}}, \ldots, \mathbf{a}(t + s_w/2)^{\mathrm{OOMP}}) \tag{5.34}$$

and

$$(\mathbf{w}_1(t), \ldots, \mathbf{w}_M(t))^T = (\mathbf{a}(t - s_w/2), \ldots, \mathbf{a}(t + s_w/2)). \tag{5.35}$$

We obtained the assignment by performing the following procedure:

1. Set $I_{\text{true}} : \{1, \ldots, M\}$ and $I_{\text{learned}} : \{1, \ldots, M\}$.
2. Find and assign $\mathbf{w}_i(t)$ and $\mathbf{w}_j(t)^{\mathrm{OOMP}}$ with $i \in I_{\text{true}}, j \in I_{\text{learned}}$ such that
$$\frac{|\mathbf{w}_j(t)^{\mathrm{OOMP}} \mathbf{w}_i(t)^T|}{\|\mathbf{w}_i(t)\| \|\mathbf{w}_j(t)^{\mathrm{OOMP}}\|} \text{ is maximal.}$$
3. Remove i from I_{true} and j from I_{learned}.
4. If $\mathbf{w}_j(t)^{\mathrm{OOMP}} \mathbf{w}_i(t)^T < 0$ set $\mathbf{w}_j(t)^{\mathrm{OOMP}} = -\mathbf{w}_j(t)^{\mathrm{OOMP}}$.
5. Proceed with (2) until $I_{\text{true}} = I_{\text{learned}} = \emptyset$.

In all experiments an overcomplete setting was used that consisted of $M = 30$ underlying sources and $N = 15$ observed mixtures. At most $k = 3$ underlying sources were active at the same time. For all experiments, we used $L = 20000$ observed mixtures, an initial learning rate $\alpha_0 = 1$, an initial neighborhood-size $\lambda_0 = M/2$, a final neighborhood-size $\lambda_{\text{final}} = 10^{-10}$, and $t_{\max} = 5000$. We repeated all experiments 20 times and report the mean result over the 20 runs. For the

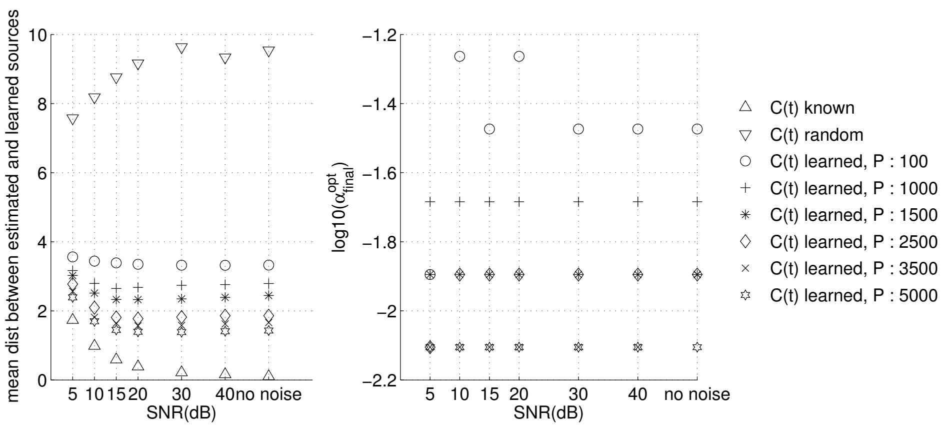


Figure 5.11: **Left:** Mean distance between $\mathbf{a}(t)$ and $\mathbf{a}(t)^{\mathrm{OOMP}}$ for different noise levels and different change rate of the mixing matrix (P). **Right:** Best performing final learning rate for each P and SNR. See text for details. Note that in case of a known or random mixing matrix the performance does not depend on P.

evaluation of the reconstruction error, the window size s_w was set to 30 and the reconstruction error was only evaluated in the time interval $t_{\max} < t < L$.

In the first experiment, we varied the SNR and the parameter P which controls the change rate of the true mixing matrix. The final learning rate α_{final} was optimized for each combination of P and SNR such that the minimal reconstruction error was obtained. For comparison purposes, we also computed the reconstruction error that was obtained by using the true mixing matrix as well as the error that was obtained by using a random matrix. The results of the first experiment are shown in Figure 5.11. On the left side, the mean distance between $\mathbf{a}(t)$ and $\mathbf{a}(t)^{\mathrm{OOMP}}$ is shown for different values of SNR and P. It can be seen that the larger the change rate of the true mixing matrix (the smaller P) is and the stronger the noise is, the more the reconstruction performance degrades. But even for strong noise and a fast changing true mixing matrix, the estimation provided by SCNG clearly outperforms a random matrix. Of course, the best reconstruction performance is obtained by using the true mixing matrix. On the right side of the figure, the best performing final learning rate for each P and SNR is shown. It can be seen that the optimal final learning rate depends on the change rate of the true mixing matrix but not on the strength of the noise.

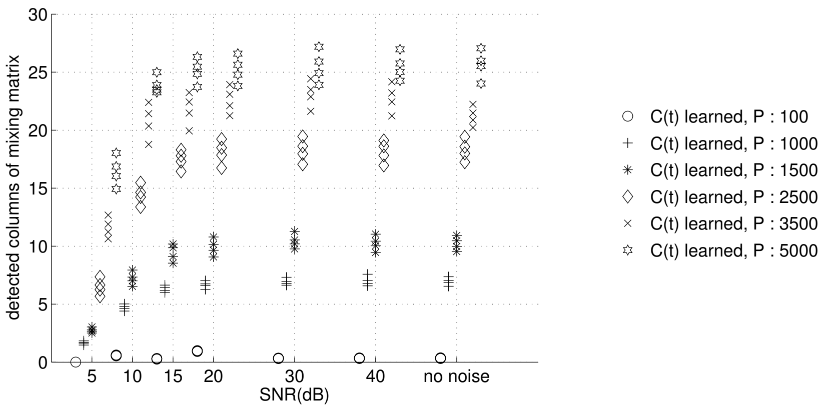


Figure 5.12: The figure shows the mean number of successfully detected columns of the mixing matrix for each of the five groups. We sorted the 20 trials according to the number of successfully learned columns of the mixing matrix and order them in groups of five experiments.

In order to assess how good the true mixing matrix was learned, we performed an experiment that is similar to an experiment that has been used to evaluate the performance of the K-SVD algorithm with respect to the learning of the true mixing matrix. This experiment was already introduced in batch settings in Section 4.5.4 and 5.2.1. Due to the time dependent mixing matrix which was given in the online-learning experiments, batch methods such as K-SVD or MOD which are discussed in Chapter 4 cannot be applied. We compared the learned mixing matrix to the true mixing matrix using the maximum overlap between each column of the true mixing matrix and each column of the learned mixing matrix. Whenever

$$\max_j \left(1 - |\mathbf{c}_i(t)\mathbf{c}_j^{\mathrm{learn}}(t)|\right) \tag{5.36}$$

was smaller than 0.05, we counted this as a success. We repeated the experiment 20 times with a varying SNR including zero noise. For each SNR, we sorted the 20 trials according to the number of successfully learned columns of the mixing matrix and ordered them in groups of five experiments. Figure 5.12 shows the mean number of successfully detected columns of the mixing matrix for each of the five groups for each SNR and P. The smaller the SNR and the change rate of the true mixing matrix are, the more columns are learned correctly. If the true mixing matrix changes very fast ($P = 100$), almost no column can be learned with the required accuracy.

6 Feature learning for visual pattern recognition

A common approach to solve a visual pattern recognition problem such as digit recognition is to divide the solution into the two parts of feature extraction and classification. A preliminary preprocessing step may be regarded as part of the feature extraction. In general it is not clear how learning methods can be used to obtain features that are optimal for a given task. Hence, methods for feature extraction are often selected according to heuristic principles based on experience and problem-specific knowledge.

In order to tackle the problem of object recognition one can match the unknown object against some reference [Keysers et al., 2007]. The matching result is then used to perform the recognition task. Though matching methods perform well in a number of tasks they are often complex and associated with a number of difficulties. For example they require to solve the computationally expensive correspondence problem [Keysers et al., 2007].

PCA or Gabor wavelets [Daugman, 1988] belong to another group of feature extraction methods. These methods do not perform an explicit matching but provide a new representation of the data. It is assumed that the new representation is advantageous with respect to recognition tasks. Based on the new representation a classifier is trained. However, it is not clear that the new representation is advantageous, since it is not guaranteed that it provides invariances adapted to the recognition problem.

It has been discussed in the introduction that there is evidence which supports the hypothesis that the visual system has adapted to the statistical properties of natural images by use of a linear generative model where objects are represented in terms of sparsely distributed coefficients of learned dictionary elements. Here, we present a method for learning visual features that is motivated by this hypothesis since the empirical observation that the brain performs very well in a broad range of pattern recognition tasks suggests that a sparse representation might be a feasible method of feature extraction in many settings. The method that is presented in this section is based on the Sparsenet approach, which has been discussed in Section 4.7. Additionally, our method is motivated by advanced models of the visual system which postulate that the output of simple cells is fed to a class of neurons which exhibit a maximum-selection behaviour [Riesenhuber and Poggio, 1999, Lampl et al., 2004].

Though the problem of digit recognition has been intensively investigated [LeCun et al., 1998, Decoste and Schölkopf, 2002, Simard et al., 2003], the improvement of digit recognition performance is still a major issue in a number of industrial applications, e.g. parcel sorting. We here describe a novel method for digit recognition that employs biologically inspired principles, i.e., a learned sparse representation and a local maximum operation. We evaluate the performance of our method for handwritten-digit recognition on the MNIST data set, being a very competitive benchmark for which many different methods already have been evaluated. In the same framework, we compare our results using a sparse code that was learned by the Sparsenet algorithm [Olshausen and Field, 1996b] against those obtained with more common feature-extraction methods such as PCA and Gabor wavelets.

6.1 Feature extraction

Pattern-recognition systems that are inspired by the neurosciences often tend to become quite complex. Is there a convenient, i.e., simple, way for practitioners to employ findings from vision research to actually solve a technical problem? An example of such a complex multi-stage recognition system being inspired by the neurosciences can be found in [Serre et al., 2007].

In this section, we propose a simple two-stage model for feature extraction. First, a number of coefficient images are computed. In order to determine the coefficients, a task specific dictionary is used that has been learned from image patches of the training data of the pattern-recognition task. Second, a local maximum operation is performed in order to obtain translation invariant features (see Figure 6.1).

We do not operate on the raw pixel values but apply a preprocessing to the image patches. This is required for the PCA and to improve the convergence of the Sparsenet algorithm. In the following, I denotes an entire image, whereas for an odd number N $\widehat{\mathbf{p}}_{x,y}$ denotes a vector that contains all N^2 pixel values of an image patch of size $N \times N$ centered at position (x, y) arranged in an appropriate scheme.

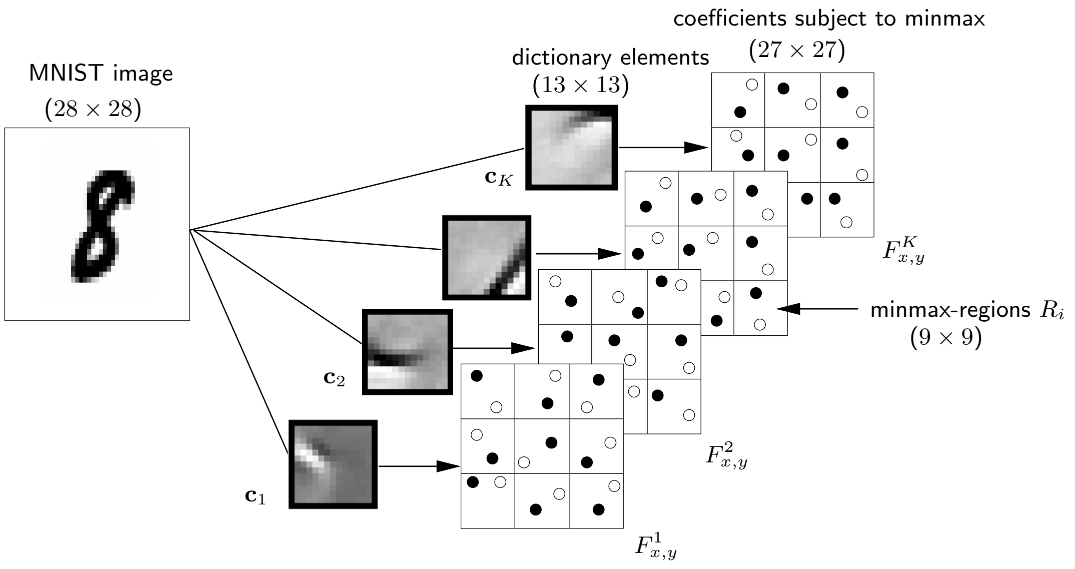


Figure 6.1: A schematic view of the proposed feature extraction method. First, the coefficients are extracted either by convolution of the input image with the elements of the dictionary or by solving (6.5). At each position in the image that is to be classified, coefficients are computed for each element of the dictionary. Second, each coefficient image is divided into regular non-overlapping regions, and for each region the minimal and maximal entry (indicated by the circles) are selected. These extrema are the features for the subsequent classification step.

The mean of the pixel values of a patch vector $\widehat{\mathbf{p}}_{x,y}$ is denoted by $\overline{\overline{\mathbf{p}}}_{x,y}$. Removal of the mean pixel value of the patch vector leads to

$$\widetilde{\mathbf{p}}_{x,y} = \widehat{\mathbf{p}}_{x,y} - \overline{\overline{\mathbf{p}}}_{x,y} \; . \tag{6.1}$$

With $\overline{\mathbf{p}}$ we refer to the mean of a large number of such $\widetilde{\mathbf{p}}_{x,y}$ that were obtained from many training images with $\widetilde{\mathbf{p}}_{x,y}$ placed at random positions. By removing $\overline{\mathbf{p}}$ we finally obtain the centered vectors $\mathbf{p}_{x,y}$:

$$\mathbf{p}_{x,y} = \widetilde{\mathbf{p}}_{x,y} - \overline{\mathbf{p}} \; . \tag{6.2}$$

In the first stage of the feature extraction, a new representation of each patch vector $\mathbf{p}_{x,y}$ of a given image I is computed. This new representation is based on a dictionary $C = (\mathbf{c}_1, \ldots, \mathbf{c}_M)$, $\mathbf{c}_j \in \mathbb{R}^{N^2}$. The way the new representation is determined depends on the framework that was used to obtain the dictionary.

6.1.1 Learning of data specific dictionaries

We want to learn a new representation of the centered patch vectors $\mathbf{p}_{x,y}$ that is adapted to the statistics of the image data that is used in the pattern-recognition task. In order to obtain such an adapted representation, we consider a linear generative model where each patch $\mathbf{p}_{x,y}$ is obtained from a linear combination of the columns of the dictionary matrix C plus some additive noise:

$$\mathbf{p}_{x,y} = C\mathbf{a}_{x,y} + \boldsymbol{\epsilon}_{x,y} \; . \tag{6.3}$$

$\mathbf{a}_{x,y}$ is the new adapted representation of the patch $\mathbf{p}_{x,y}$ whereas $\boldsymbol{\epsilon}_{x,y}$ is the residual, which depends on the method that is used in order to determine the new representation. The dictionary C is either obtained from the analytic framework of the Gabor wavelets, which are discussed in Section 3.3, or it is learned from the training data of the pattern recognition task. In order to learn a task specific dictionary we employ either PCA (see Section 4.2) or the Sparsenet algorithm (see Section 4.7).

6.1.2 Computation of coefficient images

The coefficient images F^j contain the coefficients $(\mathbf{a}_{x,y})_j$ of the dictionary columns $\mathbf{c}_j$ for all positions in the input image. In order to obtain coefficient images that have the same size as the initial image I, it is is required to enlarge I by setting the value of pixels outside the image to a fixed value. The method of obtaining the coefficients differs depending on the method that was used to learn the corresponding dictionary. In case of PCA and Gabor wavelets the coefficients are obtained by a convolution operation, i.e., for each centered patch vector $\mathbf{p}_{x,y}$ of the input image I we compute

$$F^j_{x,y} = \frac{\mathbf{p}^T_{x,y}\mathbf{c}_j}{\|\mathbf{c}_j\|}\ , j = 1, \ldots, K. \tag{6.4}$$

The Sparsenet coefficients are obtained by minimizing the objective function that was used to learn the basis:

$$(F^1_{x,y}, \ldots, F^K_{x,y})^T = \arg\min_{\mathbf{a}} \|\mathbf{p}_{x,y} - C\mathbf{a}\|_2 + \lambda S(\mathbf{a})\ . \tag{6.5}$$

As in the Sparsenet algorithm the minimization of (6.5) is performed via gradient descent. For a more detailed discussion of other methods that can be used to tackle (6.5) see also Chapter 2 and in particular Section 2.2.

6.1.3 Local maximum operation

The elements of the dictionaries are considered to represent relevant attributes of the image patches they were learned from since it is possible to reconstruct the image patches by a linear combination of only few elements. In case of sparse linear combinations only few elements of the dictionary are required to "explain" a certain image patch. A certain attribute is present at a certain location if the coefficient of the dictionary element that represents the attribute has a large absolute value. The absolute value of the coefficient can be interpreted as the similarity of the image at a certain position with respect to the element of the dictionary. Due to the nature of visual recognition tasks, some uncertainty with respect to the exact localisation of important attributes remains. Hence, we would like to allow for some spatial uncertainty to obtain local shift invariance. Assuming that those basis functions that are highly expressed, i.e., have large absolute coefficient values, are important, computing the maximum, as well as the minimum, in a local region localises the important attributes and achieves the desired local shift invariance. The attributes are considered independent, i.e. the positions where the minimum and maximum values are obtained differ for each element of the dictionary. There is some experimental evidence that the behaviour of complex cells in the visual cortex can be described by a local maximum operation [Lampl et al., 2004], and that human observers might account for position uncertainty by using the same principle [Barth et al., 1999]. The principle has been used recently in technical applications [Serre et al., 2007].

We implement this principle in a very simple way. Thereby we divide the input image into a set of regular, non-overlapping regions $R_i, i = 1, ..., M^2$ and take as local features the maximum and minimum of each region with respect to each coefficient image (see Figure 6.1):

$$F^j_{\max}(R_i) = \max_{x,y \in R_i} F^j_{x,y} \ . \tag{6.6}$$

$$F^j_{\min}(R_i) = \min_{x,y \in R_i} F^j_{x,y} \ . \tag{6.7}$$

The final feature vector (that is given as input to the classifier) of each input image consists of the maximum and minimum values of all regions with respect to all coefficient images:

$$\begin{aligned} \mathbf{f}_I \ = \ & (F^1_{\max}(R_1), \dots, F^1_{\max}(R_{M^2}), \dots, \\ & F^K_{\max}(R_1), \dots, F^K_{\max}(R_{M^2}), \\ & F^1_{\min}(R_1), \dots, F^1_{\min}(R_{M^2}), \dots, \\ & F^K_{\min}(R_1), \dots, F^K_{\min}(R_{M^2})) \ . \end{aligned} \tag{6.8}$$

Note, that in general the final feature vector $\mathbf{f}_I$ is not sparse. The coefficients of each patch are sparse but due to the maximum and minimum operation large positive or negative values are accumulated in the final feature vector.

6.2 Experiments

We tested Sparsenet, PCA, and Gabor dictionaries on the well-known MNIST benchmark of handwritten digit images. The MNIST benchmark consists of 60000 training and 10000 test images of handwritten digits of size 28×28 pixels.

The PCA and Sparsenet dictionaries were obtained by determining the parameter C of a linear generative model as it is described in Section 6.1.1. The patch vectors that were used in order to determine the dictionaries were extracted at random positions from randomly chosen training images. The noise level parameter (λ) of the Sparsenet algorithm was chosen such that the best mean validation error was obtained. Additionally, as mentioned before, the performance of a simple set of Gabor wavelets as basis functions was evaluated. We did not optimize the parameters of the Gabors explicitly but took the best parameters out of a limited set we experimented with.

We obtained the features for the classifier as described in Section 6.1. The size of the elements of the dictionaries was 13×13 pixels ($N = 13$). Though the number of dictionary elements may be optimized with respect to recognition performance, for simplicity we chose as many dictionary elements as the number of pixels in the image patches, i.e. 169. The set of Gabor wavelets that provided the best results on the digit benchmark in our experiments consists of 160 filters, which is close to the number of dictionary elements used in the Sparsenet and PCA setting (We also tried larger Gabor sets). We used a layout of 9 minmax-operator regions of size 9×9 pixels as shown in Figure 6.1. Since the size of the coefficient images F^j equals the size of the input images, we did not select entries from the bottom row and the last column of the coefficient images, which corresponds to dropping the last row and column of each input image.

As classifier we used a standard 2-norm soft margin Support-Vector-Machine (SVM) with Gaussian kernels [Vapnik, 1995, Christianini and Shawe-Tylor, 2003]. In order to train the SVM, the SoftDoubleMaxMinOver learning algorithm [Martinetz et al., 2009] was used. We normalized the training data such that the mean norm of the feature vectors $\mathbf{f}_I$ was set to one. The hyperparameters of the SVM were optimised using a validation scheme where seven realisations of test and training data were considered. In each realisation the training and test set are disjoint and consist of 10000 samples that were randomly chosen from the entire training set. We took the hyperparameters providing the best mean classification error on the validation test sets in a grid search over the trade-off parameter, which controls the softness of the classification and the Gaussian kernel parameter γ. The search uses a logarithmic grid and proceeds recursively from a coarse grid to a fine grid. The start range was $[1, 10^4]$ for the trade-off parameter and $[1, 10^3]$ for the parameter γ of the Gaussian kernel $K(x, z) = exp(-\gamma \|x - z\|^2)$. The best hyperparameters are shown in Table 6.1. Using the best hyperparameters, each final classifier is trained

	γ	trade-off parameter
raw data	21.5	1291.5
PCA	5.5	31.5
Gabors	5.5	75
Sparsenet	2.5	93

Table 6.1: The best SVM hyperparameters .

on the entire set of feature vectors of all training samples.

Due to the multiple minmax-regions and the number of dictionary elements, the dimensionality of the data increases from 784 to $9 \times 169 \times 2 = 3042$ resp. $9 \times 160 \times 2 = 2880$ feature dimensions. We had to solve a ten-class problem since we have to differentiate ten digit classes (0-9). To accomplish this task using a SVM we trained 45 two-class classifiers, each of which separates two different digits (one against one). The decisions of all the two-class classifiers were then counted and finally the class with the majority of votes was selected. We used the same hyperparameters for all the 45 two-class classifiers.

6.3 Digit recognition results

The MNIST data set is quite popular and results obtained with many state-of-the-art methods are available for comparison [LeCun et al., 1998, Decoste and Schölkopf, 2002, Simard et al., 2003]. Currently, the best results reported on the MNIST data set were obtained with convolutional neural networks plus elastic distortions (0.4% error rate [Simard et al., 2003]) and Virtual SVM with deskewing and jittering preprocessing (0.56% error rate [Decoste and Schölkopf, 2002]). A recent approach that uses sparse representations and elastic distortions obtains an error rate of 0.39% [Ranzato et al., 2007]. In [Lauer et al., 2007] a method is proposed where a LeNet5 [LeCun et al., 1995] is used to train a feature extraction layer that is fed to a set of SVMs. Using elastic distortions these authors report an error rate of 0.54%.

We consider the use of an extended training set that was generated using a problem specific distortion model as data specific knowledge. Our method belongs to the class of methods that do not use such additional knowledge. In an evaluation of several matching methods that also belong to this class of methods an error rate of 0.52% is obtained [Keysers et al., 2007]. In [Belongie et al., 2002] an error rate of 0.63% is reported using a shape matching approach. In [Steinert et al., 2006] the authors report a positive influence of sparseness on the recognition performance with the MNIST set, though they cannot obtain state-of-the-art performance.

The different types of dictionaries obtained from and used on the MNIST set of handwritten digits are shown in Figure 6.2. Both Table 6.2 and Table 6.3 refer to the final results that were obtained by using these dictionaries for feature extraction as explained in Section 6.1 and training a SVM on these features where the optimal hyperparameters according to Table 6.1 were used. Table 6.2 shows the mean

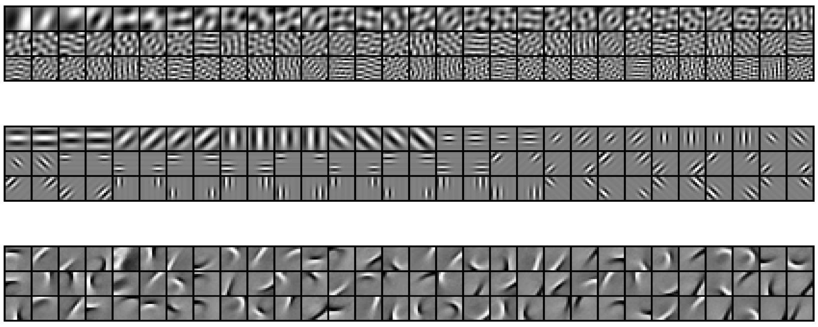


Figure 6.2: Subset of dictionaries used on the digit benchmark. From top to bottom: PCA dictionary, Gabor wavelets, dictionary that was obtained from the Sparsenet algorithm applied to a large set of randomly extracted patches of the digit images.

	mean validation error rate	error rate on MNIST test set
raw data	2.95%(±0.17)	1.42%
PCA	1.29%(±0.08)	0.80%
Gabors	1.24%(±0.10)	0.71%
Sparsenet	1.00%(±0.09)	0.59%

Table 6.2: SVM results: The second column of the table shows mean and standard deviation of the test error obtained with the best hyperparameters that were determined by validation on the 7 realisations of 10000 training and 10000 test samples. In the last column, the error on the MNIST test set is shown.

	#SVs per digit class									
	0	1	2	3	4	5	6	7	8	9
raw data	9785	5994	13241	13244	10976	12989	9842	10429	14015	12033
PCA	3293	2293	4580	4731	4041	4223	3676	3747	5246	4930
Gabors	4298	2970	5959	5690	4647	5371	4527	4986	6582	5630
Sparsenet	2698	1812	3667	3650	2833	3505	2770	3089	4216	3610

Table 6.3: SVM results: The columns refer to the classifiers that were obtained by using the optimal hyperparameters from table 6.1 to train a SVM with the SoftDoubleMaxMinOver algorithm with the complete MNIST training set. This training set consists of 60000 samples. The sum of the number of SVs of all classifiers of a particular digit class are shown.

validation error of the best hyperparameter combination as well as the final error on the MNIST test set. The mean validation error is worse than the final test error, since only 10000 instead of 60000 training samples were used for training. Table 6.3 shows the number of support vectors (SVs) that are used by the classifiers of each digit class (note that we use a one-against-one scheme, therefore we have 9 classifiers per digit class and the number of support vectors is the sum over all 9 SVMs).

All methods significantly outperformed the direct classification of the raw data. Gabor Wavelets clearly outperformed PCA. In the PCA experiment a complete basis was used. This means that for each image-patch vector an error free representation was obtained. The PCA result shows on the one hand that some performance gain can be attributed to the minimum and maximum operation since without it error free PCA yields the same result as the raw data. On the other hand the Gabor and Sparsenet results show that a sparse representation further improves performance.

The result using a learned sparse code is significantly better than the results obtained with Gabor wavelets and PCA. The number of support vectors of the best method using a learned sparse code is reduced by about a factor of three compared to the result on raw data, indicating that the feature extraction that we perform successfully implements invariances of the task. Note also that compared to earlier SVM results, for instance the virtual SVM reported in [Decoste and Schölkopf, 2002], our method uses significantly less support vectors (about a factor of four).

If a dictionary is used that has been obtained from a dictionary learning method that imposes sparseness constraints on the coefficients, the proposed feature extraction requires to solve the optimization problem of equation (6.5) via gradient descent for each patch of a given input image. Therefore PCA and Gabor wavelet feature extraction is more efficient from a computational complexity point of view. However, a number of more recent algorithms that can be used in order to determine the coefficients are available (see Chapter 2). Some of these methods could be used to determine the coefficients in a computationally more efficient way.

7 Discussion

Within this work, we have considered some domains in which the concept of sparse linear generative models, which is a mathematical abstraction of the principle of sparse coding, can be deployed. In Chapter 4, we have considered optimal representations of data in terms of representation error, image reconstruction, and image deconvolution. In Chapter 5, we have employed these models for blind source separation. Finally, in Chapter 6, we have described an approach that uses sparse linear generative models for feature learning in visual pattern recognition tasks.

In order to use a linear generative model in an application, one has to determine its parameters, i.e., the dictionary, that is used in the model. Some well-known mathematical frameworks that can be used to derive dictionaries for certain common signal classes have been discussed in Chapter 3. A more recent approach is, to derive a dictionary from given observations so that it is adapted to the data that is used in a particular task. In Chapter 4, we have reviewed many of the algorithms that have been proposed by other researchers in order to learn an appropriate dictionary on the basis of given training data.

A central limitation of all these methods is, that they use only one out of many possible configurations of the hidden parameters of the model, i.e., the coefficients, in order to learn the dictionary or mixing matrix. In the maximum likelihood interpretation that has been proposed for some of the methods (see Section 4.7), this corresponds to an approximation of the data likelihood of the model by its maximum value in order to avoid an intractable integration over the hidden coefficients. Improved methods that try to use a better approximation of the data-likelihood in order to avoid the intractable integration over the hidden parameters have been proposed, and have been briefly discussed in Section 4.7.

Within this work, we have introduced new soft-competitive methods that can be used to learn the free parameters of a linear generative model with additive noise from given data that can be seen as a novel way of dealing with this problem. These new methods are termed Sparse Coding Neural Gas (SCNG) and Neural Gas for Dictionary Learning (NGDL). In contrast to all the state-of-the-art methods, within these new methods, the number of possible configurations of the hidden parameters that can be included in each learning step, is only limited by the computational resources that are available.

Sparse Coding Neural Gas (SCNG): The SCNG algorithm is a new soft-competitive method that learns an overcomplete dictionary for the sparse encoding of given training data such that the mean representation error with respect to the given data is minimized. SCNG has been introduced in Section 4.5. In the experiments that have been described in Section 4.5.4, we used synthetic data that were actually generated from a sparse linear combination of a given dictionary. We assessed the performance of the SCNG method with respect to the reconstruction of that original dictionary. We evaluated the influence of additive noise, and the mutual coherence of the original dictionary on the reconstruction performance. The experiments show that the obtained performance depends on the sparsity of the coefficients, the strength of the additive noise, and on the mutual coherence (degree of non-orthogonality) of the underlying dictionary. The sparser the coefficients are, the smaller the mutual coherence of the dictionary is, and the lower the noise level is, the better the original dictionary can be reconstructed. On a synthetic set of data that has been used by others as a benchmark problem, we have obtained with SCNG similar results as with other state-of-the-art methods. In this experiment, though SCNG is an online method that learns pattern-by-pattern, it performed as well as state-of-the art batch methods although the batch methods could use the entire training data within each learning iteration.

In Section 4.5.5, we have shown that if the SCNG algorithm is applied to natural image data, bandpass-like dictionary elements are obtained that are localized in space and orientation. This reproduces results that have been reported by others for the sparse encoding of natural images and shows that the SCNG algorithm works robustly on real data. A further benefit of the algorithm, which can be seen from the experiment on natural image data, is that it converges even in the case of highly overcomplete dictionaries.

Sparse Coding Neural Gas for Blind Source Separation: Linear generative models are closely related to the domain of blind source separation as it has been discussed in the Introduction and in Chapter 5. A central problem in the domain of blind source separation is the "Cocktail Party Problem".

In Section 5.2, we have described how a variant of the SCNG algorithm can be used to tackle the “Cocktail Party Problem“ in a realistic and therefore difficult setting, where one has to blindly estimate the underlying sources of a linear mixture in a overcomplete situtation, i.e., where more sources than recorded mixtures are given, and additive noise is present. We have shown that SCNG can be successfully applied in this situation. An advantage of our method is that it does not rely on strong assumptions regarding the distribution of the sources or the distribution of the noise. However, it requires that the sources are sparse and that the noise level as well as the number of the underlying sources are known or can be estimated.

Based on the mixing matrix that was learned by the SCNG algorithm, we evaluated the performance on the representation level by employing the OOMP algorithm in order to obtain the sources from the observations. We analyzed the performance that can be achieved with respect to the reconstruction of the original sources. We studied the influence of the coherence of the mixing matrix, the noise level, and the sparseness of the underlying sources. We also evaluated the influence of the overcompleteness with respect to the obtained performance. These experiments and results have been presented in Section 5.2.1. If the sources are sufficiently sparse and the coherence of the mixing matrix and the noise level are sufficiently small, the SCNG algorithm is able to learn the mixing matrix and the sources can be reconstructed. The results show that sufficiently sparse sources can be reconstructed even in highly overcomplete settings.

In Section 5.2.2, we have reported on an experiment, which shows that SCNG also can be successfully applied to more realistic data. We considered the difficult setting of a piece of Jazz music that is played by five instruments and recorded by two microphones. Given only the recordings of the two microphones the track of each single instrument can be reconstructed even in the presence of some additive Gaussian noise.

Moreover, we have compared the performance of SCNG with FastICA in an experiment that has been presented in Section 5.2.3. For the comparison, we used the synthetical data as well as the more realistic audio data. In case of the synthetically generated data, we could show that SCNG outperforms FastICA and is more resistent to additive noise. In case of the audio data and a low noise level, SCNG does not outperform FastICA but if strong additive noise is present, SCNG provides better results.

In Section 5.2, the mixing matrix has been considered to be time invariant. Furthermore, a finite sequence of observed mixtures has been used. The learning of the mixing matix has been performed in batch mode, i.e., the learning algorithm could iterate several times over the entire mixture data in order to determine the mixing matrix.

An even more realistic model of the "Cocktail Party Problem" has been considered in Section 5.3. This more realistic model allows for more sources than observations, additive noise, and a mixing matrix that is time dependent, which corresponds to speakers that change their position during the conversation. Furthermore, the sequence of observed mixtures is infinite, which means that the SCNG algorithm works in a pattern-by-pattern mode. The estimation of the underlying sources is provided immediately once a new learning step with respect to the mixing matrix has been performed. From the experiments in Section 5.3.1, it can be seen that the sources have to be sparse enough, the mixing matrix must not change too quickly, and that the additive noise must not be too strong in order to enable a successfull estimation of the mixing matrix and the underlying sources.

Neural Gas for Dictionary Learning (NGDL): The SCNG algorithm performs the learning updates in a sequence of orthogonal subspaces, since it implements an Optimized Orthogonal Matching Pursuit (OOMP) that modifies the dictionary during the pursuit. Due to this methodology, it can only be applied if the sparsity constraint on the coefficients is the zero norm. In Section 4.6, we have introduced NGDL, which also is a new soft-competitive method for dictionary learning and is even more directly linked to the Neural Gas (NG) vector quantization method. NGDL is more versatile since it can be combined with different sparse approximation methods. This makes it applicable, for instance, whenever a probabilistic linear generative model is considered where the additive noise is assumed to be Gaussian. NGDL is powerful because it avoids matrix inversion or singular value decomposition and, due to its stochastic nature, is not so easily trapped in local minima. Results in Section 4.6.2 and 4.6.5 that have been obtained with typical synthetic benchmark data show that the simple and fast NGDL is competitive, and in many cases even superior to computationally more intensive state-of-the-art methods such as Method of Optimal Directions (MOD) or K-SVD, both in terms of how well the representation error is minimized and how well the dictionary is reconstructed.

In order to apply NGDL, one needs a sparse approximation method that does not provide only one possible configuration of the coefficients but rather a set of possible configurations. We have proposed the novel bag of pursuits (BOP) method for sparse approximation, which is derived from OOMP. BOP determines a set of good configurations of the coefficients with respect to a given dictionary instead of just a single configuration. BOP has been described in detail in Section 2.1.4. BOP can be computationally expensive, but in Section 4.6.3 we have shown on synthetic data that it can significantly improve both the approximation of the data and the reconstruction of the original dictionaries.

Sparse approximation for image reconstruction and deconvolution: Inverse problems such as image reconstruction and image deconvolution can be tackeled within a sparse approximation framework as it has been discussed in Section 4.6.7 and 4.6.8. In order to successfully solve these tasks in a sparse approximation framework, one needs a dictionary that provides a sparse representation of the image that is to be reconstructed or deconvolved. An advantage of learned dictionaries is that they can be adapted to a particular sub-class of images, which might lead to better results.

In Section 4.6.7, we have reported on image-reconstruction experiments. In these experiments from a set of 78 test images a certain percentage of pixels was set to $\emptyset$. Then, all the 78 test images were reconstructed by application of BOP as sparse approximation method. The experiment was repeated for a number of different dictionaries. We have shown that a dictionary that has been learned by the NGDL+BOP method can outperform dictionaries that have been obtained from a number of state-of-the-art methods in terms of reconstruction performance. Limited training data and limited computation time were available for the dictionary learning task. The learned dictionaries also outperformed dictionaries that were obtained from an analytical framework, e.g., an overcomplete Haar dictionary and an overcomplete DCT dictionary.

In Section 4.6.8 we have presented experiments that illustrate the potential usefulness of dictionaries that have been learned with NGDL+BOP in an image-deconvolution task. The deconvolution experiment was performed for two different classes of images, namely buildings and flowers. Dictionaries have been learned separately for each class. We then blurred two test images, i.e., an image that depicts a flower and another image that depicts a building. We have then used both dictionaries to invert the image blur, with known blurring kernel.

When using the buildings dictionary for deblurring the buildings, results were significantly better than when using the flowers dictionary. Conversely, the flowers dictionary yielded better results on the flower images. We have therefore shown that our method is able to learn dictionaries that adapt to a particular class of images. Moreover, all results that were obtained with the learned dictionaries are clearly better than those obtained with the Haar wavelets that we used as a reference.

Feature extraction for visual pattern recognition: In order to show that sparse linear models can be successfully applied to visual pattern recognition tasks, we have proposed a simple feature extraction method for visual pattern recognition. The feature extraction is based on unsupervised learning of sparse data representations. A detailed description of this method has been provided in Chapter 6. As an example of a visual pattern recognition task, the problem of handwritten digit recognition has been considered in the experiments.

In our feature extraction method, the Sparsenet algorithm is employed in order to learn a dictionary that provides a sparse representation of patches of the training images of the pattern recognition task. Then, for each training and test image, the sparse coefficients of its patches are determined with respect to the learned dictionary. A new representation of the training and test images is thus obtained, where within a certain region the minimum and maximum values of the coefficients of each dictionary element are used.

We trained a Support Vector Machine (SVM) on the features of the training images and tested its classification performance on the features of the test images. In the experiments, we used the training and test images of the MNIST handwritten digit benchmark. The results of the SVM training show that the number of support vectors is significantly reduced compared to training on the raw data. This indicates that our new learned representation incorporates invariances of handwritten digits.

The classification performance improves significantly, even though the final feature vector used for classification is not sparse in general and the dimensionality of the data increases. We compared a representation based on a dictionary that was learned with the Sparsenet algorithm with more traditional representations based on PCA and Gabor wavelets. Gabor wavelets can be seen as a sparse basis of natural images. According to our experiments, a better performance could be achieved by using a sparse code that was obtained by unsupervised learning from the MNIST training data.

Despite its simplicity, our approach performs as well as state-of-the-art methods that do not use prior knowledge specific to the handwritten digit recognition problem. The classification performance of our method is comparable with the performance of the image matching methods as evaluated in [Keysers et al., 2007]. For example, methods proposed in [Decoste and Schölkopf, 2002, Simard et al., 2003, Ranzato et al., 2007, Lauer et al., 2007] employ an elastic deformation modell for digits to boost their performance, while our method implicitly extracts deformation invariances by the unsupervised learning of a sparse code.

We have shown that a sparse feature representation, combined with a biologically plausible maximum operation leads to highly competitive classification performance. Since our method does not employ task specific prior knowledge and is simple, it can be applied to a broad range of visual pattern recognition problems. In cases where a specific image distribution is given that deviates significantly from natural images, a learned sparse feature representation may lead to a significant increase in classification performance.

Conclusion

In the Introduction, we have presented state-of-the-art results that support the hypothesis that the principle of sparse coding is an information processing strategy used by the brain. Since the information processing strategies of the brain have been proven useful and competitive during evolution, it is sensible to use similar strategies in machine learning algorithms. Within this work, we have shown that this principle can be used to make significant progress in a number of machine learning tasks.

Sparse linear generative models are a powerful framework that implements the principle of sparse coding. To adapt the unknown parameters of such a model to particular observations is an induction step that involves a difficult non-convex optimization problem.

We have proposed new powerful methods for dictionary learning that enable us to tackle this difficult learning problem. We demonstrated in experiments that these new methods are able to identify the underlying ground truth of given observations under conditions where other state-of-the-art methods fail. We presented results that show that the domains of data representation, denoising, deconvolution, blind source separation, and feature extraction can benefit from these new approaches. Due to the versatility of sparse linear models, we think that in the future a broad range of application domains can utilize the approaches that have been proposed in this work.

Bibliography

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.

M. Antonini, M. Barlaud, Pierre Mathieu, and I. Daubechies. Image coding using wavelet transform. *IEEE Trans. on Image Proccess.*, 1(2):205–220, 1992.

E. Barth, B. L. Beard, and A. J. Ahumada. Nonlinear features in vernier acuity. In *IS&T/SPIE Symposium on Human Vision and Electronic Imaging*, volume 3644, 1999.

A. J. Bell and T. J. Sejnowski. The "independent components" of natural scenes are edge filters. *Vision Res*, 37(23):3327–3338, December 1997. ISSN 0042-6989.

Anthony J. Bell and Terrence J. Sejnowski. An Information-Maximization Approach to Blind Separation and Blind Deconvolution. *Neural Computation*, 7(6):1129–1159, 1995.

S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(4):509–522, 2002. ISSN 0162-8828.

A. M. Bruckstein, D. L. Donoho, and M. Elad. From sparse solutions of systems of equations to sparse modeling of signals and images. *SIAM Review*, 51(1):34–81, 2009.

S. G. Chang, Y. Bin, and M. Vetterli. Adaptive wavelet thresholding for image denoising and compression. *IEEE Trans. on Image Process.*, 9(9):1532–1546, 2000.

Scott Shaobing Chen, David L. Donoho, and Michael A. Saunders. Atomic Decomposition by Basis Pursuit. *SIAM Journal on Scientific Computing*, 20(1):33–61, 1998.

N. Christianini and J. Shawe-Tylor. *Support Vector Machines*. Cambridge University Press, 2003.

Marie Cottrell, Barbara Hammer, Alexander Hasenfuß, and Thomas Villmann. Batch and median neural gas. *Neural Netw.*, 19(6):762–771, 2006. ISSN 0893-6080.

I. Daubechies, M. Defrise, and C.De-Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Commun. Pure Appl. Math.*, 57:1413–1457, 2004.

J. G. Daugman. Two-dimensional spectral analysis of cortical receptive field profiles. *Vision Res.*, 20:847–856, 1980.

J. G. Daugman. Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters. *J Opt. Soc.*, 2(7): 1160–1169, 1985.

J. G. Daugman. Complete discrete 2-D gabor transforms by neural networks for image analysis and compression. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 36(7):1169–1179, 1988.

J. G. Daugman. Entropy reduction and decorrelation in visual coding by oriented neural receptive fields. *IEEE Transactions on Biomedical Engineering*, 36:107–114, 1989.

M. Davies and N. Mitianoudis. Simple mixture model for sparse overcomplete ICA. *IEEE Proceedings on Vision, Image and Signal Processing*, 151(1):35–43, 2004.

G. Davis, S. Mallat, and M. Avellaneda. Greedy adaptive approximation. *J. Constr. Approx.*, 13:57–89, 1997.

Dennis Decoste and Bernhard Schölkopf. Training Invariant Support Vector Machines. *Mach. Learn.*, 46(1-3):161–190, 2002. ISSN 0885-6125.

David L. Donoho and Michael Elad. Optimally sparse representation in general (nonorthogonal) dictionaries via l^1 minimization. *PNAS*, 100(5):2197–2202, 2003.

David L. Donoho, Michael Elad, and Vladimir N. Temlyakov. Stable recovery of sparse overcomplete representations in the presence of noise. *IEEE Transactions on Information Theory*, 52(1):6–18, 2006.

Michael Elad and Michal Aharon. Image Denoising via Sparse and Redundant Representations Over Learned Dictionaries. *IEEE Trans on Image Processing*, 15 (12):3736–3744, 2006.

K. Engan, S. O. Aase, and J. Hakon Husoy. Method of optimal directions for frame design. In *ICASSP '99: Proceedings of the Acoustics, Speech, and Signal Processing, 1999. on 1999 IEEE International Conference*, pages 2443–2446, Washington, DC, USA, 1999. IEEE Computer Society. ISBN 0-7803-5041-3.

Kjersti Engan, Sven Ole Aase, and John H**r**akon Husøy. Multi-frame compression: theory and design. *Signal Process.*, 80(10):2121–2140, 2000. ISSN 0165-1684. doi: http://dx.doi.org/10.1016/S0165-1684(00)00072-4.

D. J. Field. What the statistics of natural images tell us about visual coding. *SPIE Human Vision, Visual Processing, and Digital Display*, 1077:269–276, 1989.

David J. Field. What is the goal of sensory coding? *Neural Computation*, 6(4): 559–601, 1994. ISSN 0899-7667.

M. A. T. Figueiredo, R. D. Nowak, and S. J. Wright. Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. *IEEE Journal of Selected Topics in Signal Processing*, 1:586–597, 2007.

Peter Földiák. Adaptive network for optimal linear feature extraction. *Neural Networks*, 1:401 – 405, 1989.

Peter Földiák and Malcolm P. Young. Sparse coding in the primate cortex. pages 895–898, 1998.

Keinosuke Fukunaga. *Introduction to statistical pattern recognition (2nd ed.)*. Academic Press Professional, Inc., San Diego, CA, USA, 1990. ISBN 0-12-269851-7.

D. Gabor. Theory of communication. *J. Inst. Electr. Eng.*, 93:429–457, 1946.

R. Gribonval, R. Figueras, and P. Vandergheynst. A simple test to check the optimality of a sparse signal approximation. *Signal Process.*, 86:496–510, 2006.

J. A. Hartigan and M. A. Wong. A K-means Clustering Algorithm. *Applied Statistics*, 28:100–108, 1979.

J. H. Van Hateren and A. Van Der Schaaf. Independent component filters of natural images compared with simple cells in primary visual cortex. In *Proceedings of the Royal Society London*, number 265, pages 359–366, 1998.

Simon Haykin and Zhe Chen. The cocktail party problem. *Neural Computation*, 17 (9):1875–1902, 2005. ISSN 0899-7667.

Patrik Hoyer and Erkki Oja. Image denoising by sparse code shrinkage. In *Intelligent Signal Processing*. IEEE Press, 2000.

M. R. Hromadka, T. DeWeese and A. M. Zador. Sparse representation of sounds in the unanesthetize auditory cortex. *PLoS Biology*, 6, 2008.

D. H. Hubel. Tungsten microelectrode for recording from single units. *Science*, 125: 549–550, 1957.

D. H. Hubel and T. N. Wiesel. Receptive fields of single neurones in the cat's striate cortex. *J. Physiol*, 148:574–591, 1959.

D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *J. Physiol*, 160:106–154, 1962.

D. H. Hubel and T. N. Wiesel. Receptive fields and functional architecture in two non-striate visual areas (18 and 19) of the cat. *J. Neurophysiol*, 28:229–289, 1965.

A. Hyvärinen. Fast and Robust Fixed-Point Algorithms for Independent Component Analysis. *IEEE Transactions on Neural Networks*, 10(3):626–634, 1999a.

A. Hyvärinen. Gaussian moments for noisy independent component analysis. *IEEE Signal Processing Letters*, 6(6):145–147, 1999b.

A. Hyvärinen, R. Cristescu, and E. Oja. A fast algorithm for estimating overcomplete ICA bases for image windows. *Proceedings of the International Joint Conference on Neural Networks, IJCNN'99*, 2:894–899, 1999.

Aapo Hyvärinen and Patrik Hoyer. Emergence of Phase- and Shift-Invariant Features by Decomposition of Natural Images into Independent Feature Subspaces. *Neural Comput.*, 12(7):1705–1720, 2000. ISSN 0899-7667.

Aapo Hyvärinen and Erkki Oja. A fast fixed-point algorithm for independent component analysis. *Neural Comput.*, 9(7):1483–1492, 1997. ISSN 0899-7667.

Aapo Hyvärinen, Juha Karhunen, and Erkki Oja. *Independent Component Analysis.* Wiley-Interscience, May 2001.

R. A. Jortner, Farivar S. S., and Laurent G. A simple connectivity scheme for sparse coding in an olfactory system. *J. Neuroscience*, 27(7):1659–1669, 2007.

C. Jutten and J. Herault. Blind separation of sources, Part I: An adaptive algorithm based on neuromimatic architecture. *Signal Processing*, 24(1):1–10, 1991.

Daniel Keysers, Christian Gollan, Thomas Deselaers, and Hermann Ney. Deformation models for image recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29 (8):1422–1435, 2007. ISSN 0162-8828.

Kenneth Kreutz-Delgado and Bhaskar D. Rao. Sparse basis selection, ICA, and majorization: Towards a unified perspective. In *Proceedings of ICASSP '99. International Conference on Acoustics, Speech, and Signal Processing*, pages 1081 – 1084, Phoenix, AZ , USA, 1999. IEEE Computer Society.

Kenneth Kreutz-Delgado, Joseph F. Murray, Bhaskar D. Rao, Kjersti Engan, Te-Won Lee, and Terrence J. Sejnowski. Dictionary learning algorithms for sparse representation. *Neural Comput.*, 15(2):349–396, 2003. ISSN 0899-7667.

P. Kruizinga and N. Petkov. Nonlinear operator for oriented texture. *IEEE Transactions on Image Processing*, 8(10):1395–1407, 1999.

S. W. Kuffler. Discharge patterns and functional organization of mmalian retina. *J. Neurophysiol*, 16:37–68, 1953.

H. J. Kushner and D. S. Clark. *Stochastic Approximation Methods for Constrained and Unconstrained Systems.* Springer, 1978.

Kai Labusch and Thomas Martinetz. Learning Sparse Codes for Image Reconstruction. In Michel Verleysen, editor, *Proceedings of the 18th European Symposium on Artificial Neural Networks*, pages 241–246. D-Side Publishers, 2010.

Kai Labusch, Udo Siewert, Thomas Martinetz, and Erhardt Barth. Learning optimal features for visual pattern recognition. In Bernice E. Rogowitz, Thrasyvoulos N. Pappas, and Scott J. Daly, editors, *Human Vision and Electronic Imaging XII*, volume 6492. Proceedings of SPIE, 2007.

Kai Labusch, Erhardt Barth, and Thomas Martinetz. Learning data representations with Sparse Coding Neural Gas. In Michel Verleysen, editor, *Proceedings of the 16th European Symposium on Artificial Neural Networks*, pages 233–238. D-Side Publishers, 2008a.

Kai Labusch, Erhardt Barth, and Thomas Martinetz. Sparse Coding Neural Gas for the Separation of Noisy Overcomplete Sources. In Vera Kurková, Roman Neruda, and Jan Koutník, editors, *Artificial Neural Networks - ICANN 2008, 18th International Conference, Prague, Czech Republic, September 3-6, 2008, Proceedings, Part II*, volume 5163 of *Lecture Notes in Computer Science*, pages 788–797. Springer, 2008b.

Kai Labusch, Erhardt Barth, and Thomas Martinetz. Simple Method for High-Performance Digit Recognition Based on Sparse Coding. *IEEE Transactions on Neural Networks*, 19(11):1985–1989, 2008c.

Kai Labusch, Erhardt Barth, and Thomas Martinetz. Sparse Coding Neural Gas: Learning of Overcomplete Data Representations. *Neurocomputing*, 72(7-9):1547–1555, 2009a.

Kai Labusch, Erhardt Barth, and Thomas Martinetz. Approaching the Time Dependent Cocktail Party Problem with Online Sparse Coding Neural Gas. In J.C. Principe and R. Miikkulainen, editors, *Advances in Self-Organizing Maps - WSOM 2009, 7th International Workshop, St. Augustine, Fl, USA, June 2009*, volume 5629 of *Lecture Notes in Computer Science*, pages 145–153. Springer, 2009b.

Kai Labusch, Erhardt Barth, and Thomas Martinetz. Demixing Jazz-Music: Sparse Coding Neural Gas for the Separation of Noisy Overcomplete Sources. *Neural Network World*, 19(5):561–579, 2009c.

Kai Labusch, Erhardt Barth, and Thomas Martinetz. Bag of Pursuits and Neural Gas for Improved Sparse Coding. In Gilbert Saporta, editor, *Proceedings of the 19th International Conference on Computational Statistics*, pages 327–336. Springer, 2010.

Kai Labusch, Erhardt Barth, and Thomas Martinetz. Soft-competitive Learning of Sparse Codes and its Application to Image Reconstruction. *Neurocomputing*, 74 (9):1418–1428, April 2011a.

Kai Labusch, Erhardt Barth, and Thomas Martinetz. Robust and Fast Learning of Sparse Codes With Stochastic Gradient Descent. *IEEE Transactions on Selected Topics in Signal Processing*, 5(5):1048 – 1060, 2011b.

Ilan Lampl, David Ferster, Tomaso Poggio, and Maximilian Riesenhuber. Intracellular measurements of spatial integration and the max operation in complex cells of the cat primary visual cortex. *J Neurophysiol*, 92(5):2704–2713, 2004.

Fabien Lauer, Ching Y. Suen, and Gérard Bloch. A trainable feature extractor for handwritten digit recognition. *Pattern Recogn.*, 40(6):1816–1824, 2007. ISSN 0031-3203.

Y. LeCun, L. D. Jackel, L. Bottou, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, U. A. Muller, E. Sackinger, P. Simard, and V. Vapnik. Learning algorithms for classification: A comparison on handwritten digit recognition. In J. H. Oh, C. Kwon, and S. Cho, editors, *Neural Networks: The Statistical Mechanics Perspective*, pages 261–276. World Scientific, 1995.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.

Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Y. Ng. Efficient sparse coding algorithms. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 801–808. MIT Press, Cambridge, MA, 2007.

Jong-Hwan Lee, Ho-Young Jung, Te-Won Lee, and Soo-Young Lee. Speech feature extraction using independent component analysis. *Acoustics, Speech, and Signal Processing, IEEE International Conference on*, 3:1631–1634, 2000.

Te-Won Lee, M.S. Lewicki, M. Girolami, and T.J. Sejnowski. Blind source separation of more sources than mixtures using overcomplete representations. *IEEE Signal Processing Letters*, 6(4):87–90, 1999.

Michael S. Lewicki and Terrence J. Sejnowski. Learning nonlinear overcomplete representations for efficient coding. In *NIPS '97: Proceedings of the 1997 conference on Advances in neural information processing systems 10*, pages 556–562, Cambridge, MA, USA, 1998. MIT Press. ISBN 0-262-10076-2.

Michael S. Lewicki and Terrence J. Sejnowski. Learning Overcomplete Representations. *Neural Computation*, 12(2):337–365, 2000.

Y. Linde, A. Buzo, and R. Gray. An algorithm for vector quantizer design. *IEEE Transactions on Communications*, 28:84–95, 1980.

J. Mairal, M. Elad, and G. Sapiro. Sparse representation for color image restoration. *IEEE Trans on Image Proccess.*, 17(1), 2008.

Stephane Mallat. *A wavelet tour of signal processing. The sparse way.* Elsevier, 2009.

T. Martinetz and K. Schulten. A "Neural-Gas Network" Learns Topologies. *Artificial Neural Networks*, I:397–402, 1991.

T. Martinetz, S. Berkovich, and K. Schulten. "Neural-gas" Network for Vector Quantization and its Application to Time-Series Prediction. *IEEE-Transactions on Neural Networks*, 4(4):558–569, 1993.

Thomas Martinetz, Kai Labusch, and Daniel Schneegaß. SoftDoubleMaxMinOver: Perceptron-like Training of Support Vector Machines. *IEEE Transactions on Neural Networks*, 20(7):1061–1072, 2009.

E. Oja. A simplified neuron model as a principal component analyzer. *J. Math. Biol.*, 15:267–273, 1982.

B. Olshausen and D. Field. Sparse coding of natural images produces localized, oriented, bandpass receptive fields. Technical Report CCN-110-95, Department of Psychology, Cornell University, 1995.

Bruno Olshausen and David Field. What is the other 85% of V1 doing? In T. Sejnowski and L. van Hemmen, editors, *Problems in Systems Neuroscience.* Oxford University Press, 2004.

Bruno A. Olshausen and David J. Field. Natural image statistics and efficient coding. *Network*, 7(2):333–339, 1996a.

Bruno A. Olshausen and David J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, (381):607–609, 1996b.

Bruno A. Olshausen and David J. Field. Sparse coding with an overcomplete basis set: a strategy employed by V1? *Vision Research*, 37(23):3311–3325, 1997.

M. R. Osborne, B. Presnell, and B. A. Turlach. A new approach to vairable selection in least squares problems. *J. Numer. Anal.*, 20:389–403, 2000.

Y. Pati, R. Rezaiifar, and P. Krishnaprasad. Orthogonal Matching Pursuit: Recursive Function Approximation with Applications to Wavelet Decomposition. *Proceedings of the 27 th Annual Asilomar Conference on Signals, Systems,*, November 1993.

R. Quian Quiroga, L. Reddy, G. Kreiman, C. Koch, and I. Fried. Invariant visual representation by single neurons in the human brain. *Nature*, 435:1102–1107, 2005.

R. Quian Quiroga, G. Kreiman, C. Koch, and I. Fried. Sparse but not 'grandmother-cell' coding in the medial temporal lobe. *Trends Cogn Sci*, 12(3):87–91, 2008.

Marc'Aurelio Ranzato, Christopher Poultney, Sumit Chopra, and Yann LeCun. Efficient learning of sparse representations with an energy-based model. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 1137–1144. MIT Press, Cambridge, MA, 2007.

B. D. Rao, K. Engan, S. F. Cotter, J. Palmer, and K. Kreutz-Delgado. Subset selection in noise based on diversity measure minimization. *IEEE Trans. Signal Process.*, 51:760–770, 2003.

L. Rebollo-Neira and D. Lowe. Optimized orthogonal matching pursuit approach. *IEEE Signal Processing Letters*, 9(4):137–140, 2002.

M. Riesenhuber and T. Poggio. Hierarchical models of object recognition in cortex. *Nature Neuroscience*, 2(11):119–125, 1999.

D. Rinberg, A. Koulakov, and A. Gelperin. Sparse odor coding in awake behaving mice. *J. Neuroscience*, 26(34):8857–8865, 2006.

Sheldon Ross. *A First Course in Probability.* Prentice Hall, 2002.

C. J. Rozell and D. H. Johnson R. G. Baraniuk. Sparse coding via thresholding and local competition in neural circuits. *Neural Computation*, 20:2526–2563, 2008.

Terence D. Sanger. Optimal Unsupervised Learning in a Single-Layer Linear Feedforward Neural Network. *Neural Networks*, 2:459–473, 1989.

Thomas Serre, Lior Wolf, Stanley Bileschi, Maximilian Riesenhuber, and Tomaso Poggio. Robust Object Recognition with Cortex-Like Mechanisms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(3):411–426, 2007. ISSN 0162-8828.

Shy Shoham, Daniel H. O'Connor, and Ronen Segev. How silent is the brain: is there a "dark matter" problem in neuroscience? *Journal of Comparative Physiology*, 192(8):777–784, 2006.

Patrice Y. Simard, Dave Steinkraus, and John C. Platt. Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis. In *ICDAR '03: Proceedings of the Seventh International Conference on Document Analysis and Recognition*, page 958, Washington, DC, USA, 2003. IEEE Computer Society. ISBN 0-7695-1960-1.

S.Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41:3397–3415, 1993.

J. L. Starck, A. Bijaoui, B. Lopez, and C. Perrier. Image reconstruction by the wavelet transform applied to aperture synthesis. *Astron and Astrophysics*, 283 (1):349–360, 1994.

Rebecca Steinert, Martin Rehn, and Anders Lansner. Recognition of handwritten digits using sparse codes generated by local feature extraction methods. In *ESANN*, pages 161–166, 2006.

F. Theis, E. Lang, and C. Puntonet. A Geometric Algorithm for Overcomplete Linear ICA. *Neurocomputing*, 56:381–398, 2004.

J. A. Tropp. Greed is good: algorithmic results for sparse approximation. *IEEE Transactions on Information Theory*, 50(10):2231–2242, 2004.

Leslie G. Valiant. A quantitative theory of neural computation. *Biol. Cybern.*, 95 (3):205–211, 2006. ISSN 0340-1200.

R. L. De Valois, D. G. Albrecht, and L. G. Thorell. Spatial frequency selectivity of cells in macaque visual cortex. *Vision Res.*, 22:545–559, 1982.

Vladimir N. Vapnik. *The nature of statistical learning theory.* Springer-Verlag New York, Inc., New York, NY, USA, 1995. ISBN 0-387-94559-8.

Satoshi Watanbe. *Knowing and Guessing, A Quantitative Study of Inference and Information.* Wiley, 1969.

H. Weyl. *The Theory of Groups and Quantum Mechanics.* Dutton, 1931.

Christoph Zetzsche and Erhardt Barth. Fundamental limits of linear filters in the visual processing of two dimensional signals. *Vision Research*, 30:1111–1117, 1990.

Christoph Zetzsche, Erhardt Barth, and Bernhard Wegmann. The importance of intrinsically two-dimensional image features in biological vision and picture coding. In Andrew B. Watson, editor, *Digital Images and Human Vision*, pages 109–38. MIT Press, OCT 1993.

Michael Zibulevsky and Barak A. Pearlmutter. Blind Source Separation by Sparse Decomposition in a Signal dictionary. *Neural Computation*, 13(4):863–882, 2001.

Danksagung

Ich danke meinen Kollegen am INB für interessante anregende Zeiten, für Diskusionen zu unterschiedlichsten Fragestellungen und die gemeinsame Suche nach Erkenntnis. Insbesondere danke ich Thomas Martinetz und Erhardt Barth für die Betreuung und Förderung meines Promotionsvorhabens, interessante Fragestellungen und die gute Zusammenarbeit.

Ferner danke ich meinen Brüdern und Eltern und insbesondere meiner Frau Ulrike, die mich immer in meinen Vorhaben bestärkt und unterstützt haben.