CARL VON OSSIETZKY universität OLDENBURG

Probabilistic Models for Invariant Representations and Transformations

Von der Fakultät für Mathematik und Naturwissenschaften der Carl von Ossietzky Universität Oldenburg zur Erlangung des Grades

Doktor der Naturwissenschaften (Dr. rer. nat.)
angenommene Dissertation

von Herrn **Georgios Exarchakis**

geboren am 02. Februar 1985 in Larisa, Griechenland

georgios.exarchakis@uol.de

Erstgutachter: Prof. Dr. Jörg Lücke
Zweitgutachter: Prof. Dr. Bruno Olshausen
Tag der Disputation: 01. Dezember 2016

Abstract

English version

The central task of machine learning research is to extract regularities from data. These regularities are often subject to transformations that arise from the complexity of the process that generates the data. There has been a lot of effort towards creating data representations that are invariant to such transformations. However, most research towards learning invariances does not model the transformations explicitly.

My research is focused towards modeling data in ways that separate their "content" from the potential "transformations" it undergoes. I primarily used a probabilistic generative framework due to its high expressive power and the belief that any potential representation will be subject to uncertainty. To model data content I focused on sparse coding techniques due to their ability to extract highly specialized dictionaries. I defined and implemented a discrete sparse coding model that models the presence/absence of a dictionary element subject to finite set of scaling transformations. I extended the discrete sparse coding model with an explicit representation for temporal shifts that learns time-invariant representations for the data without loss of temporal alignment. In an attempt to create a more general model for data transformations, I defined a neural network that uses gating units to encode transformations from pairs of datapoints. Furthermore, I defined a non-linear dynamical system that expresses the dynamics in terms of a bilinear transformation that combines the previous state and a variable that encodes the transformation to generate the current state.

In order to examine the behavior of these models in practice I tested them with on a variety of tasks. Almost always, I tested the models on recovering parameters from artificially generated data. Furthermore, I discovered interesting properties in the encoding of natural images, extra-cellular neural recordings, and audio data.

German version

Die Hauptaufgabe des maschinellen Lernens ist es, aus gegebenen Daten Regelmäßigkeiten zu extrahieren. Diese Regelmäßigkeiten unterliegen oftmals verschiedenen Transformationen, die der Komplexität des Prozesses geschuldet sind, der die Daten erzeugt. In der Vergangenheit wurde viel Aufwand betrieben um Repräsentationen zu lernen, die invariant gegenüber solchen Transformationen sind. In den meisten Fällen werden jedoch beim Lernen solcher Invarianten die Transformationen nicht explizit berücksichtigt.

Der Schwerpunkt meiner Arbeit liegt darin, gegebene Daten so zu modellieren, dass der "Inhalt" und die möglichen "Transformationen", die der Inhalt durchläuft, voneinander getrennt werden können. Dabei habe ich hauptsächlich ein probabilistisches generatives Framework benutzt. Und zwar einerseits, weil es eine hohe Ausdruckskraft besitzt und andererseits aus dem Glauben heraus, dass jegliche mögliche Repräsentation der Daten immer auch eine gewisse Unsicherheit enthält. Um den Inhalt der Daten zu modellieren, verwende ich sogenannte "sparse coding" (dt. "spärliche Kodierung") Techniken, die es erlauben hoch spezialisierte Lexika zu extrahieren. Hierfür wurde ein "sparse coding" Modell definiert und implementiert, dass die An- bzw. Abwesenheit eines Lexikon-Eintrags modelliert, der durch eine endliche Anzahl an Skalierungsoperationen transformiert wurde. Das diskrete "sparse coding"-Modell wurde mit einer expliziten Repräsentation von zeitlichen Verschiebungen so erweitert, dass zeitinvariante Repräsentationen gelernt werden können ohne dabei die zeitliche Übereinstimmung zu verlieren. In einem Versuch ein allgemeineres Modell für Datentransformationen zu erstellen, wurde ein neuronales Netzwerk entwickelt, dass Transformationen zwischen Paaren von Datenpunkten kodiert. Weiterhin wurde ein nichtlineares System benutzt, dass die Dynamik durch eine bilineare Transformation beschreibt, die den vorigen Zustand und eine Variable, die die Transformation kodiert, kombiniert, um so den aktuellen Zustand zu generieren.

Um das Verhalten der oben genannten Modelle zu untersuchen, wurden sie auf verschiedenen Aufgaben getestet. Fast immer wurden die Modelle darauf getestet, aus künstlich erzeugten Daten bestimmte Parameter wiederherzustellen. Außerdem konnten interessante Eigenschaften beim Kodieren von natürlichen Bildern, extra-zellulären neuronalen Messungen und Audiodaten gewonnen werden.

Contents

1. Introduction

The brain has the capacity to identify regularities in natural stimuli in a highly efficient manner. During our lifetime we are exposed to an environment that is constantly changing and yet our brain is able to learn how to break it down into components that remain unaltered by those changes. In order to create artificial systems that integrate with the real world as efficiently as humans we need to learn how to extract information from natural data in a similar manner. Defining algorithms that assume as input a large amount of data and identify patterns and provide a succinct, flexible and powerful descriptions of the data is a subject of central interest in the field of Machine Learning.

Machine Learning is the field of study that deals with the development of algorithms that identify patterns in data. The task in itself is very broad and addresses several objectives from developing algorithms for classification and density estimation to improving the scaling of those algorithms for large numbers of variables, observed or unobserved. With such a broad set of research directions it often appears wise to draw inspiration from the natural environment in order to identify efficient learning techniques and the brain is the most apparent natural choice. In this work we define a variety of learning algorithms that we aspire to be applicable in wide range of problems and yet capable to provide a high quality representations of the data. For that reason, we often argue on the similarities between the behavior of our algorithms and the brain.

A prominent feature of the nervous system is the ability to separate sensory data in static and dynamic components. When we interact with nature we are often presented with stimuli that are subject to certain variations and yet we are able to perceive such stimuli and quite often the way it changes as well. For instance, when we see a moving car we are able to identify it as a car that follows a certain trajectory. That suggests that somewhere in our brain there is a representation of a car that is invariant to the change in the car's location. If we now see a bicycle moving in the same way as the car we can also identify the bicycle regardless of its transitions, more importantly though, we can identify that the car and the bicycle where moving in the same way, in other words, their state was subject to the same changes. This observation extends to other sensory modalities, for instance, when we hear someone talking we can identify what they say but we can also identify the voice of the speaker, whether they are moving away from us and so on. Creating invariant representations of "things" in our environment is of central interest in machine learning. (see e.g. Földiák, 1991; Tenenbaum and Freeman, 2000; Lewicki and Sejnowski, 2000; Memisevic and Hinton, 2007; LeCun et al., 2015, and many others).

With this in mind we define in this thesis a series of machine learning algorithms that

extract regularities from natural data. The models we deal with here use the learned structures to define different representations of the data that clearly separate content from change. We primarily use probabilistic generative models with latent variables but also artificial neural networks to learn these representations. In the following sections we will introduce elementary mathematical principles of the models we shall discuss later in more detail.

1.1. Probabilistic Generative Models

A highly expressive and versatile way to describe structure in a dataset are probabilistic generative models. Probabilistic generative models assume that there is some uncertainty regarding in the way the model describes the data and proceeds to model it by accounting for the probability of the data to be present. More specifically, an observation y is modeled by its probability of appearance:

$$p(y) \tag{1.1}$$

By sampling from the distribution of the data equation 1.1 we can generate data similar to the ones we observe, as described by our model, with slight perturbations that correspond to the uncertainty of the representation. When defining a distribution it is necessary to use some parameters that describe its structure. We usually denote the full set of parameters by the variable Θ. The parameters Θ are not the argument of our probabilistic model but they are necessary to define it and therefore we often denote the model as:

$$p(y|\Theta) = p(y) \tag{1.2}$$

Identifying the correct parameters Θ^* for the model given a set of observations is the objective of machine learning in the probabilistic setting. For that reason, it is useful to define a function of the parameters that takes the same values as the distribution at a point. This function is the likelihood[1] and we denote it as:

$$L(\Theta) = p(y|\Theta) \tag{1.3}$$

In general, when we seek the probabilistic model that is most likely to have generated the data we need to identify the model that maximizes the data likelihood, equation 1.3. This approach to learning is addressed as *Maximum Likelihood Estimation* (MLE) and is one of the most prominent methods for parameter estimation. MLE methods is what we mean when we discuss learning in this work.

The way we perceive the world implies structure beyond the one we interact with. Generative models describe this structure as unobserved/latent variables and assume that

[1] For N samples of y we assume that they are iid distributed and the likelihood is $L(\Theta) = p(\mathbf{y}|\Theta) = \prod_{n=1}^{N} p\left(y^{(n)}\right)$. We show the process for a $N = 1$ since it trivially extends to an arbitrary N.

the observed variables depend on them. In order to include these latent variables, x to our model we use the rule of total probability to extend equation 1.2 as:

$$p\left(y|\Theta\right) = \sum_{x} p\left(y, x|\Theta\right) \tag{1.4}$$

Typically the term generative models refer to latent variable generative models and they are defined by the joint probability of observed and latent variables, $p\left(y, x|\Theta\right)$, restructured as:

$$p\left(y, x|\Theta\right) = p\left(y|x, \Theta\right) p\left(x|\Theta\right) \tag{1.5}$$

where the probability $p\left(x|\Theta\right)$ is known as the prior of the latent variables and $p\left(y|x, \Theta\right)$ is commonly referred to as the noise model of the observed variables. When trying to infer something about the environment from our observations we often resort to the posterior of the latent variables given an observation, $p\left(x|y, \Theta\right)$ that is given to use by the Bayes' theorem:

$$p\left(x|y, \Theta\right) = \frac{p\left(y|x, \Theta\right) p\left(x|\Theta\right)}{p\left(y|\Theta\right)} = \frac{p\left(y|x, \Theta\right) p\left(x|\Theta\right)}{\sum_{x} p\left(y|x\Theta\right) p\left(x|\Theta\right)} \tag{1.6}$$

The Bayes' theorem relates the prior of the latent variables and the noise model of the observations with the posterior of the latent variables given the data. This means that, within the framework of probabilistic generative models, the only thing we need to define in order to model the structure of the natural world is the prior, and the noise model. One should note however that for a lot of generative models identifying the posterior is task of considerable difficulty. For instance, if x is a high dimensional discrete random variable the sum in the denominator of equation 1.6 can easily become computationally intractable. Also, if x takes values in a continuous domain the integrating over all the possible values requires solving an integral for the denominator of equation 1.6 that is not always analytically tractable.

1.1.1. Expectation Maximization

Now that we have defined the significant probabilistic quantities of generative models we need to define a method for learning the optimal parameters, Θ^*, of the model. As stated previously, the optimal parameters Θ^* are the ones that maximize the likelihood function. However, for both numerical and analytical reasons it is quite common to try to optimize the logarithm of the likelihood instead

$$\mathcal{L}\left(\Theta\right) = \log L\left(\Theta\right) = p\left(y|\Theta\right) \tag{1.7}$$

where $L\left(\Theta\right)$ is the likelihood function defined in equation 1.3. Since the logarithm is monotonically increasing function the optima of the log-likelihood are found for the same parameters, Θ^*, as the likelihood. There are many ways we can use to optimize the log-

likelihood e.g. gradient based methods, sampling and others. However, we will use an algorithm commonly known Expectation Maximization (EM) (Dempster et al., 1977) due to its nice theoretical and empirical convergence properties. Taking the logarithm of the likelihood of a latent variable model introduces a sum inside inside the logarithm which is difficult to handle. The EM framework works around that issue as follows:

$$\begin{aligned}
\mathcal{L}(\Theta) &= \log \sum_x p(y, x|\Theta) \\
&= \log \sum_x \frac{q(x)\, p(y, x|\Theta)}{q(x)} \\
&\geq \sum_x q(x) \log \frac{p(y, x|\Theta)}{q(x)} \\
&= \sum_x q(x) \log p(y|x, \Theta)\, p(x|\Theta) - \sum_x q(x) \log q(x) \\
&= \mathcal{Q}(\Theta, q) + H[q] = \mathcal{F}(q, \Theta)
\end{aligned} \tag{1.8}$$

where the inequality is Jensen's inequality and it introduces a distribution $q(x)$ over the latent variables. Equation 1.8 is called the *Free Energy* and it is a lower bound to log-likelihood that is much easier to optimize. $H[q]$ is the Shanon entropy of the distribution $q(x)$, and $\mathcal{Q}(\Theta, q)$ is the only part of the free energy that depends to the parameters and therefore serves as the objective function of the optimization problem. The EM is a two step iterative algorithm that optimizes the free energy. In the E-step we increase the free energy by making it equal to the log-likelihood, i.e. satisfying the condition:

$$\begin{aligned}
\mathcal{F}(q, \Theta) &= \sum_x q(x) \log p(y|x, \Theta)\, p(x|\Theta) - \sum_x q(x) \log q(x) \\
&= \sum_x q(x) \log p(x|y, \Theta)\, p(y|\Theta) - \sum_x q(x) \log q(x) \\
&= \sum_x q(x) \log p(y|\Theta) - \sum_x q(x) \log \frac{q(x)}{p(x|y, \Theta)} \\
&= \log p(y|\Theta) - KL[q(x) \,\|\, p(x|y, \Theta)] \\
&= \mathcal{L}(\Theta) - KL[q(x) \,\|\, p(x|y, \Theta)]
\end{aligned} \tag{1.9}$$

where $KL[q(x) \,\|\, p(x|y, \Theta)]$ is the Kullback–Leibler (KL) divergence between $p(x|y, \Theta)$ and $q(x)$. The KL divergence between two distributions is a equal to zero only when $p(x|y, \Theta)$ and $q(x)$ are equal. Therefore, identifying the posterior of the latent variable given the data defines the E-step of our algorithm.

The M-step of the EM algorithm is optimizing the free energy with respect to the parameters Θ. Typically, parameters Θ are real valued numbers and since the logarithm is

an monotonically increasing function identifying the values of the parameters that set the gradient of the free energy to zero is sufficient to maximize it:

$$\nabla_{\Theta}\mathcal{F}\left(q,\Theta\right)=0 \tag{1.10}$$

However, updating the parameters Θ changes the posterior of the latent variables given the data and therefore the updated free energy is no longer equal to the log-likelihood. It is therefore necessary to iterate between the E-step and M-step of the algorithm until convergence to identify the optima of the log-likelihood function.

Potential issues during training. There are many potential issues that can arise when applying the EM algorithm to generative models. In the E-step, identifying the posterior is often non trivial. In the case of continuous latent variables, the integral in the denominator of the posterior is sometimes not analytically tractable and one has to resort to numerical methods to find an approximate estimate of it. In the case of discrete latent variables, it is usually much easier to get an analytical solution for the posterior, however, with an increasing dimensionality for the latent variables the posterior becomes computationally intractable. In that case, we need to resort to sampling methods for an approximate estimate for the posterior or apply factored variational approximations, i.e. approximate the posterior with a distribution over the latents that treats them as independent random variables.

In this work we will present a novel approximation scheme for discrete latent variables, that is a straight forward extension of the truncated approximation in (Lücke and Eggert, 2010). Truncated approximations introduce a further step before the E-step that identifies a subset of the latent variables posterior that contains the majority of the posterior mass and estimates the posterior only over that set.

1.1.2. Sparse Coding

As an example generative model we introduce the Sparse Coding model. Sparse Coding (SC) (Olshausen and Field, 1997) was proposed as neural coding strategy of simple cells in the primary visual cortex of the mammalian brain, and it has since become a prominent information encoding paradigm on a diverse set of applications.

The rationale behind the definition of sparse coding suggests that in order for each latent variables to describe as much of the observations as possible it has to take values very close to zero most of the time and values far away from zero when it shares some common structure with a datapoint. The observations then are defined as a linear combination of those latent variables.

For observations $\vec{y}$ defined in $\mathbb{R}^D$, and latents $\vec{s}$ that take values in $\mathbb{R}^H$ the probabilistic

generative model is defined by the following two probability densities:

$$\text{Prior:} \qquad p\left(\vec{s}|\Theta\right) = \prod_h C\left(s_h, \Theta\right) \tag{1.11}$$

$$\text{Noise Model:} \qquad p\left(y|\vec{s}, \Theta\right) = \mathcal{N}\left(W\vec{s}; \sigma^2 \mathbb{1}\right) \tag{1.12}$$

where $C\left(\vec{s}|\Theta\right)$ is heavy tailed probability density, typically a Laplace or a Cauchy distribution. Notice, that the prior assumes that each latent variable is independent and identically distributed from the other and the noise model is a Gaussian density with a mean equal to the linear combination of the latent variables under a "dictionary" matrix W. The parameters we need to optimize are the dictionary elements, i.e. $\Theta = \{W\}$. Unfortunately, the posterior of the Sparse Coding model is analytically intractable and one has to resort to either sampling methods or identifying sophisticated point estimates in the latent space for inference and learning.

The Sparse Coding model has been central to the development of non-Gaussian encoding schemes in Machine Learning and it is the standard model for the behavior of simple cell receptive fields in the primary visual cortex. However, following the rationale of Sparse Coding one could further improve on the encoding by assuming discrete random variables for the latent space. Having a discrete latent space with at least one zero and one non-zero value would send a more transparent signal for the contribution or not of a variable in the observed data. In chapters 2, and 3 we will introduce algorithms that deal with such prior distributions.

1.2. Neural Networks

Artificial neural networks are a class of models that is popular for the analysis of natural data. Neural Networks are a class of models originally designed to model connectivity patterns in the brain that have developed to imitate considerable functional properties as well.

Originally proposed by Rosenblatt (1958) the perceptron is potentially the simplest form of neural network:

$$x_j = f\left(\sum_i w_{ji} y_i + b_j\right) \tag{1.13}$$

where f is a function defined as $f\left(x\right) = 1$, if $x > 0$, and $f\left(x\right) = 0$ otherwise. The perceptron, equation 1.13, models the activity of a neuron, x_j, as the weighted sum of the activity of the neurons it is connected to, y_i, if the corresponding weight w_{ji} is small/high then the connection between the two neurons is weak/strong. More recently, the word perceptron has been used to describe models with other nonlinear functions f.

One can use the output of a perceptron as input to another perceptron and do so repeat-

edly to create a multilayer perceptron.

$$x_k = g\left(\sum_j v_{kj} f\left(\sum_i w_{ji} y_i + b_j\right)\right) \tag{1.14}$$

Equation 1.14 presents the activation of a two-layer perceptron. Multilayer perceptrons are part of a branch of a machine learning branch known as Deep Learning. Deep Learning models seek alternative representations of the observations by giving them as arguments to a cascade of linear and non-linear functions (for a recent review see LeCun et al., 2015).

It is not uncommon to relate neural networks to probabilistic models by assigning probability density function as a further nonlinearity of the last layer. For the Gaussian case, for instance, a representation such as the one in equation 1.14 would become:

$$p(\vec{x}|\vec{y}) = \mathcal{N}\left(g\left(Vf(W\vec{y})\right), \Sigma\right) \tag{1.15}$$

where the mean $g\left(Vf\left(W\vec{y}\right)\right)$ is identical to the left hand side of 1.14, only now switched to a vectorial notation for the operations. Defining a likelihood for our model in terms of equation 1.15, deviates from our standard modelling methodology of generative models in the sense that our likelihood is a conditional probability of $\vec{x}$ given $\vec{y}$, $p\left(\vec{x}, \vec{y}\right)$, and we do not define their joint distributions $p\left(\vec{x}, \vec{y}\right)$. Models that define the conditional probability directly are known as *discriminative models*. Since discriminative models model the relationship between $\vec{x}$ and $\vec{y}$ they are typically used for supervised learning, i.e. learning in a setting where you have observations for both $\vec{x}$ and $\vec{y}$.

It is worth to note at this point that taking the logarithm of equation 1.15 and assuming and identity covariance matrix, Σ, reduces the learning problem to minimizing the squared error between the left and right hand side of equation 1.14. By far the most popular way to do that is via stochastic gradient descent in the parameter space of the model, $\Theta = \{W, V\}$. Since the mapping between $\vec{y}$ and $\vec{x}$ is a composition of linear and non-linear functions we can use the chain rule to identify the gradient, i.e. $(g \circ f)' = (f' \circ g) \cdot g'$ for instance for the parameters V we have:

$$\begin{aligned} \nabla_V \mathcal{L}(\Theta) &= \nabla_V \left\|\vec{x} - g\left(Vf\left(W\vec{y}\right)\right)\right\|_2^2 \\ &= 2\left(\vec{x} - g\left(Vf\left(W\vec{y}\right)\right)\right) \odot g'\left(Vf\left(W\vec{y}\right)\right) f\left(W\vec{y}\right)^T \end{aligned} \tag{1.16}$$

where f, g, and g' are elementwise functions and $\odot$ denotes an elementwise multiplication. Similarly, for the parameter W we have:

$$\begin{aligned} \nabla_W \mathcal{L}(\Theta) &= \nabla_W \left\|\vec{x} - g\left(Vf\left(W\vec{y}\right)\right)\right\|_2^2 \\ &= 2\left(\vec{x} - g\left(Vf\left(W\vec{y}\right)\right)\right) \odot g'\left(Vf\left(W\vec{y}\right)\right) V^T \odot f'\left(W\vec{y}\right)\vec{y}^T \end{aligned} \tag{1.17}$$

notice the reuse of the quantities in equations 1.16 and 1.17. The gradients described above by passing the error of the representation back through the transposed weights and the differentiated nonlinearities. This method of computing the gradients for neural networks is commonly referred to as back-propagation of errors, or simply *back-propagation*. Subtracting the gradients in equations 1.16 and 1.17 gives the update rule for the parameters of a neural network trained using gradient descent:

$$\Theta^{\text{new}} = \Theta^{\text{old}} - \epsilon \nabla_{\Theta} \mathcal{L}(\Theta) \tag{1.18}$$

where ϵ is the learning rate of the network. In stochastic gradient descent the sum over datapoints in $\mathcal{L}(\Theta)$ is limited to randomly selected subset of the datapoints. To make the learning more efficient one often includes momentum terms (see for instance Nesterov et al., 2007) or methods using second order derivatives (Battiti, 1992).

A particular type of neural network that we are going to deal with in this paper is the *autoencoder*. An autoencoder is an unsupervised neural network that tries to find a mapping of the data to itself. The simplest type of autoencoder is the linear autoencoder:

$$\mathcal{L}(W, V) \propto -\|\vec{x} - VW\vec{x}\| \tag{1.19}$$

where $W \in \mathbb{R}^{H \times D}$ is the encoding matrix, and $V \in \mathbb{R}^{H \times D}$ is the decoding matrix. When $W = V^T$, and $H < D$ the encoding, $\vec{h} = W\vec{x}$, of the linear autoencoder at the optimum has the same properties as the representation of the data offered by Principle Component Analysis (PCA) (Baldi and Hornik, 1989). Training the model described in 1.19 using stochastic gradient descent is more computationally efficient than most PCA algorithms and that sketches the appeal autoencoders have to machine learning research. By mapping the data onto itself using a neural network one can easily extract interesting representations of the data.

Most autoencoder research is focus on nonlinear and deep autoencoder models, i.e. representations like $\vec{h}$ are passed through elementwise nonlinearities and possibly through further encoding processes (Hinton and Salakhutdinov, 2006; Vincent et al., 2010). In chapter 4 that takes in a pair of datapoints and tries to relate them using their multiplicative interactions as a nonlinearity (see for instance Memisevic and Hinton, 2007).

The field of artificial neural networks has seen remarkable growth in recent years. The reason being that neural networks usually exhibit their advantages compared to other techniques when we deal with a large amount of data. The creation of large databases for machine learning tasks has exposed their potential in multiple tasks. Another reason for their recent growth is the fact that training neural networks with stochastic gradient descent is a task easily parallelizable through the use of graphical processing units (GPUs) and the recent advances in GPU architectures have made training neural networks orders of magnitude faster.

1.3. Dynamical Systems

So far we have discussed models that deal with datasets with independent observations, however, quite often we have to deal with observations that are not independent. When we study systems that exhibit dependencies across time we are discussing dynamical systems. Dynamical systems can be separated in two large categories discrete-time dynamical systems that treat time as a discrete variable and continuous-time dynamical systems that treat time as continuous variable. In this work we only refer to discrete-time dynamical systems and often we just refer to them as dynamical systems. Typically, when modeling dynamical systems we seek a different representation of the observations, a representation that expresses the state of the system at a point in time, and then we seek to identify structures that describe temporal dependencies in the system state. Identifying the system state at a point, is often directly analogous to seeking a posterior in a generative model. Temporal dependencies can be a bit more complex because they can exist across multiple time points. A commonly used methodology to simplify the modeling of temporal dependencies is to assume that the system state at a point in time depends only on the state at the immediately earlier point in time and not on any state before that. This is known as the Markov property and it is crucial for the inference process on dynamical systems that we use here.

1.3.1. Linear Dynamical Systems

Linear dynamical systems (LDS) assume that the relationship across consecutive states are linear transformations subject to uncertainty. The stochastic formulation of a linear dynamical system is:

$$y_t = Wx_t + u \tag{1.20}$$

$$x_t = Ax_{t-1} + v \tag{1.21}$$

where the variables x_t describe the system state and the variables y_t the observations at time t. The matrix A defines the dynamics of the model state and the matrix W is the dictionary or output matrix of the model. Variables u, and v represent the noise or uncertainty of the model, typically following a Gaussian distribution. In order to identify the model parameters in terms of the likelihood maximization framework discussed earlier we need to introduce the likelihood of the model in terms of the observations $\mathbf{y} = y_1, \ldots, y_T$ as:

$$\begin{aligned} p(\mathbf{y}) &= \prod_{t=1}^{T} p(y_t|x_t) \prod_{t=2}^{T} p(x_t|x_{t-1})\, p(x_1) \Leftrightarrow \\ &= \prod_{t=1}^{T} \mathcal{N}(y_t|Wx_t, \Sigma) \prod_{t=2}^{T} \mathcal{N}(x_t|Ax_{t-1}, \Gamma)\, \mathcal{N}\left(x_1|\hat{x}_1, \hat{V}_1\right) \end{aligned} \tag{1.22}$$

where $p(x_1)$ is the prior distribution of the latent variables of the model. Equation 1.22 defines the likelihood of a linear dynamical system and it can be maximized with a variant of the EM algorithm (originally Rauch et al. (1965) but also see Ghahramani and Hinton (1996); Shumway and Stoffer (2010). As in the case of the EM described earlier, the crucial quantity to estimate is the posterior distribution of the latent variables given the observations. $p(\mathbf{x}|\mathbf{y})$. Which becomes much harder to compute directly as the number of time-points increases. Due to the Markov property of the model the quantity that becomes relevant for the parameter updates is only the posterior at time t, $p(x_t|\mathbf{y}, \Theta)$, and it can be computed efficiently through a two-pass iterative process.

1.3.2. Non-Linear Dynamical Systems

In some cases the data we are trying to model are going through a more complex process than what can be described with linear dynamical system. In that case, it might be necessary to describe the data with a nonlinear dynamical system:

$$y_t = Wx_t + u \tag{1.23}$$
$$x_t = f(x_{t-1}) + v \tag{1.24}$$

where f is a non-linear function. A nonlinear dynamical system that we will discuss in chapter 5 expresses the dynamics as a bilinear map between the earlier state and another variable:

$$y_t = Wx_t + u \tag{1.25}$$
$$x_t = f(x_{t-1}, k_t) + v \tag{1.26}$$

where k_t is a variable that defines a linear transformation at time t. More explicitly, a bilinear map f is a function of two variables that is linear for both of them, i.e. $f(x, ay + b) = af(x, y) + b$, and $f(ax + b, y) = af(x, y) + b$. Therefore, maintaining the variable k_t fixed at equation 1.26 reduces it to a linear dynamical system, i.e. the variable k_t identifies the linear transformation that takes place between state $t - 1$, and t.

1.4. Overview

Based on the afore mentioned modeling methodologies we propose a set of algorithms that attempt to extract form invariant to a set of changes that are also explicitly modeled.

More specifically in chapter 2, we implement a Sparse Coding (Olshausen and Field, 1996) model with discrete latent variables which are invariant to a set of discrete scaling coefficients. We train the model using a variant of the EM algorithm that has worked well with binary latents in earlier work. An appealing property of this learning setup is that

we are able to learn a sparse dictionary as well as the prior for the activation probabilities and the variance of the noise model. We examine the learning behavior of the algorithm by applying it to artificially generated data with known parameters and discuss its ability to recover the data from random initialization. By applying the learning to natural images we learn structure that is found in typical sparse dictionaries only now we go beyond that by learning prior structure as well. We apply the algorithm on data captured from extra-cellular recordings of spiking neurons to learn interesting structure that challenges typical spike detection approaches. Using audio waveforms of human speech we learn sparse dictionaries that provide insights to the scaling behavior of the speech components.

We extend the discrete sparse coding algorithm to include a latent variable responsible for temporal alignment of the dictionary elements with the data in chapter 3. Using this additional variable we gain representations for the data that are time invariant without discarding temporal alignment. We apply the invariant discrete sparse coding algorithm to extra-cellular recordings of spiking neurons to find time-invariant dictionaries for spikes with explicit localization in time. Our results indicate that identifying the spiking profile and the spike timing of a neuron offers a simple and elegant solution to the spike sorting problems. On audio data, we use the algorithm to learn time invariant dictionaries and study the extracted parameters and compare to the parameters of discrete sparse coding model in chapter 2 to find surprisingly similar results.

To study change in the data in a more general setting we propose an autoencoder that learns how to identify relationships between pairs of datapoints in chapter 4. We test its potential to learn image relationships by applying it on pairs of images of items that change by a 3-dimensional rotation. We proceed to use the trained model to generate analogies of the transformations in the images.

In chapter 5, we propose a bilinear dynamical system that models dynamics in terms of state and transformation variables. The rationale behind it is that while the neural network in the earlier section would work between pairs of images the bilinear dynamical system could potentially scale to sequences of data of arbitrary size. We test this algorithm on artificial data to study its ability to identify transformations from a sequence of observation.

Finally, we discuss the theoretical and experimental results of our work in terms of lessons learned from applying our models to natural and artificial data. We talk about the issues regarding the model definitions and training, the results that we found particularly interesting and potential extensions of the presented algorithms. The end of this document also includes an appendix with some derivations that might facilitate the reader while going through parts of this work.

2. Discrete Sparse Coding

2.1. Introduction

In this chapter we discuss a variant of the Sparse Coding algorithm with discrete latent variables. The work in this chapter is published in Exarchakis and Lücke (2017). The mathematical description section and any comments on relevant research represents joint work between Georgios Exarchakis and Jörg Lücke.

The arguments in favor of sparsity stem from multiple research directions: classical computer vision results (Field, 1994), observed sparsity in brain recordings (Hubel and Wiesel, 1977), and the idea that the generative process of natural data consists of sparsely present structural elements (Olshausen and Field, 1997; Field, 1994). In this work we focus on the later idea, i.e., that it is common to perceive natural datasets as a large set of distinct structural elements that appear infrequently. Pursuing distinct selectivity of features in the data, as opposed to obfuscated overlapping responsibilities in earlier Gaussian approaches (Hancock et al., 1992), SC has commonly been associated with heavy tailed prior distributions. However, it is often argued that the SC principle encourages discrete distributions or distributions with a discrete component (Rehn and Sommer, 2007; Titsias and Lázaro-Gredilla, 2011; Goodfellow et al., 2012; Sheikh et al., 2014) since continuous distributions do not send a clear "yes" or "no" signal for the features that constitute a datapoint. Similarly, it is frequently pointed out in the closely related field of compressive sensing (see, e.g., Donoho, 2006; Eldar and Kutyniok, 2012; Sparrer and Fischer, 2014) that "hard" sparsity (in the form of an l_0 sparsity penalty) is preferable to softer sparsity as it would reflected by continuous prior distributions.

All non-Gaussian encodings of hidden variables typically pose difficulties in Machine Learning. While efficient approaches have been developed for approaches such as independent component analysis (Bell and Sejnowski, 1997; Bingham and Hyvärinen, 2000) or non-negative matrix factorization (Lee and Seung, 1999), we usually face severe analytical intractabilities if data noise is taken into account. For typical sparse coding models, we are therefore forced to apply approximation schemes (e.g. Olshausen and Field, 1997; Lee et al., 2007; Berkes et al., 2008; Mairal et al., 2010) to obtain efficient learning algorithms for parameter optimization. Several techniques have been used to overcome that problem (Aharon et al., 2006) based on either sophisticated point estimates of the posterior mode or sampling based methods (e.g. Berkes et al., 2008). Each of these methods offers its own set of advantages and disadvantages. Methods based on point estimates tend

to be computationally efficient by avoiding the intricacies of dealing with uncertainty in the posterior, for instance. Sampling based methods, on the other hand, offer a more advanced description of the posterior but usually at a cost of either computational complexity or convergence speed. In the case of discrete hidden variables, it is straight-forward to derive exact analytical solutions for the optimization of model parameters within the expectation maximization (EM) approach (e.g. Haft et al., 2004; Henniges et al., 2010, for binary latents) but such exact solutions scale very poorly with the number of latent dimensions. In order to overcome poor scalability during learning and inference for sparse coding with binary latents, factored or truncated variational approximations to a-posterior distributions have been used (Haft et al., 2004; Henniges et al., 2010; Bingham et al., 2009). Like in the continuous case, also sampling offers itself as a well-established and efficient approach (see, e.g., Zhou et al. (2009) for a 'hard' sparsity model or Griffiths and Ghahramani (2011) for a non-parametric approach with binary latents). In practice, however, deterministic factored or truncated approaches are frequently preferred (Haft et al., 2004; Zhou et al., 2009; Titsias and Lázaro-Gredilla, 2011; Sheikh et al., 2014) presumably due to their computational benefits in high dimensional hidden spaces. For discrete latents, truncated approximations to intractable posteriors (Lücke and Eggert, 2010; Puertas et al., 2010; Exarchakis et al., 2012; Henniges et al., 2014) have represented an alternative to sampling and factored variational methods. Like sampling (but unlike factored variational methods), truncated approximations do not make the assumption of *a-posteriori* independence. Like factored approaches (but unlike sampling), truncated approximations have been shown to be very efficient also in spaces with very large hidden spaces (Shelton et al., 2011; Sheikh et al., 2014). Truncated approaches can be expected to represent very accurate approximations if posterior masses are concentrated on relatively few states, which makes them well suited for our purposes.

The main focus of this study is to derive a general learning algorithm for sparse coding with discrete latents. That is, for any sparse prior distribution over discrete values. Such a new prior allows us to study l_0 prior distributions with no constraints in the shape of the active coefficients, i.e., our approach does not assume any functional shape for the prior probabilities of our discrete states in contrast to the requirement of choosing a specific functional form of priors for SC with continuous latents (e.g., Cauchy and Laplace Olshausen and Field (1997), and many more, Student-t Berkes et al. (2008) or other heavy-tail distributions). The computational complexity of training a general discrete sparse coding model, which goes much beyond the complexity of earlier approaches, will be a main challenge of this study. Furthermore, we address the problem of how parameters of the prior for each state can be learned, as learning of such parameters from data essentially allows for learning of the shape of the prior distribution.

To demonstrate the newly derived approach and its capabilities, it is applied, e.g, to acquire information about the prior structure only conjectured in earlier work. Initially, we first demonstrate the effectiveness of the training scheme on artificial data to better expose

the intricacies of the learning procedure. We continue by testing the model with different configurations on natural images and aim at inferring prior shapes with a minimal scientific bias, in the course of this work we are also verifying the validity of the model by replicating and confirming preliminary earlier results (Henniges et al., 2010; Exarchakis et al., 2012) as special cases of our approach. Furthermore, we perform an analysis of data captured through extra-cellular recordings of spiking neurons using a configuration for the latent states that accounts for background activity as well as potential decays that occur in spike trains. Common methods of analysis of extra-cellular recordings use intricate pipelines for spike detection and identification and often rely on Gaussian priors to characterize a spike even though the spike is perceived as a discrete quantity. Here, we propose a model that takes into account the discrete nature of spikes as well as their varying amplitudes, which are due to spike trains, as well as potential overlaps with spikes of nearby neurons. Finally, we apply the model on a feature extraction task from human speech using the raw waveform. Human speech is widely perceived to be discrete in nature regardless of what its discrete components are (words, syllables, phonemes and so on), and that implies that a discrete encoding of speech in an unsupervised setting is particularly interesting.

2.2. Mathematical Description

Consider a set, Y, of N independent datapoints $\vec{y}^{(n)}$, with $n = 1, \ldots, N$, where $\vec{y}^{(n)} \in \mathbb{R}^D$. For these data the studied learning algorithm seeks parameters $\Theta^* = \{W^*, \sigma^*, \vec{\pi}^*\}$ that maximize the data log-likelihood:

$$L\left(Y|\Theta\right) = \log \prod_{n=1}^{N} p\left(\vec{y}^{(n)}|\Theta\right) = \sum_{n=1}^{N} \log p\left(\vec{y}^{(n)}|\Theta\right)$$

Sparse coding models are latent variable models and therefore the likelihood is defined as a function of unobserved random variables as follows

$$L\left(\vec{\boldsymbol{y}}|\Theta\right) = \sum_{n=1}^{N} \log p\left(\vec{y}^{(n)}|\Theta\right) = \sum_{n=1}^{N} \log \sum_{\vec{s}} p\left(\vec{y}^{(n)}|\vec{s},\Theta\right) p\left(\vec{s}|\Theta\right) \tag{2.1}$$

where the latent variables $\vec{s}$ are taken to have discrete values, and where the sum $\sum_{\vec{s}}$ goes over all possible vectors $\vec{s}$, i.e., over all possible combinations of discrete states. Let $\vec{s}$ be of length H, i.e. $\vec{s} = (s_1, \ldots, s_H)^T$, where each element s_h can take on one of K discrete values $\phi_k \in \mathbb{R}$, i.e. $\vec{s}_h \in \Phi = \{\phi_1, \ldots, \phi_K\}$. For such latents, we can define the following

prior:

$$p(\vec{s}\,|\,\Theta) \;=\; \prod_{h=1}^{H}\prod_{k=1}^{K} \pi_k^{\delta(\phi_k=s_h)}, \quad \text{with} \quad \sum_{k=1}^{K}\pi_k = 1, \tag{2.2}$$

where $\delta\left(\phi_k = s_h\right)$ is an indicator function which is one if and only if $\phi_k = s_h$ and zero otherwise. As for standard sparse coding, Equation 2.2 assumes independent and identical distributions for the latents s_h. The prior will be used to model sparse activity by demanding one of the values in $\Phi = \{\phi_1, \ldots, \phi_K\}$ to be zero and the corresponding probability to be relatively high. We will refer to the set of possible values Φ as a *configuration*. An example is to choose configuration $\Phi = \{0, 1\}$, which reduces (using $\pi = \pi_2$ and $(1-\pi) = \pi_1$) the prior (2.2) to the Bernoulli prior $p(\vec{s}\,|\,\Theta) \;=\; \prod_{h=1}^{H} \pi^{s_h}\,(1-\pi)^{1-s_h}$ (as used for binary sparse coding, Haft et al., 2004; Henniges et al., 2010; Bornschein et al., 2012). The notation used in (2.2) is similar to a categorical distribution but applies for latents with any values ϕ_k with any probabilities π_k. Its form will be convenient for later derivations.

Having defined the prior (2.2), we assume the observed variables $\vec{y} = (y_1, \ldots, y_D)^T$ to be generated as in standard sparse coding, i.e., we assume Gaussian noise with the mean set by a linear superposition of the latents:

$$p\left(\vec{y}\,|\,\vec{s}, \Theta\right) = \mathcal{N}\left(\vec{y}; W\vec{s}, \sigma^2\mathbb{1}\right) \tag{2.3}$$

with an isotropic covariance, $\sigma^2\mathbb{1}$, and mean $W\vec{s}$. We call the data model defined by (2.2) and (2.3) the discrete sparse coding (DSC) data model.

Given a set of datapoints $\vec{y}^{(1)}, \ldots, \vec{y}^{(N)}$ and the DSC data model, we now seek parameters $\Theta = (\vec{\pi}, W, \sigma)$ that maximize the likelihood (2.1). We derive parameter update equations using Expectation Maximization in its free-energy formulation (Neal and Hinton, 1998). In our case, exact EM update equations can be derived in closed-form but the E-step scales with the number of hidden states $\mathcal{O}(K^H)$, making the algorithm computationally intractable for large H.

In order to derive computationally tractable approximations for parameter optimization, we approximate the intractable a-posteriori probabilities $p(\vec{s}\,|\,\vec{y}, \Theta)$ by a truncated distribution:

$$p(\vec{s}\,|\,\vec{y}^{(n)}, \Theta) \;\approx\; q^{(n)}(\vec{s}; \Theta) \;=\; \frac{p(\vec{s}\,|\,\vec{y}^{(n)}, \Theta)}{\sum_{\vec{s}'\in\mathcal{K}^{(n)}} p(\vec{s}'\,|\,\vec{y}^{(n)}, \Theta)}\;\delta(\vec{s}\in\mathcal{K}^{(n)}), \tag{2.4}$$

where $\mathcal{K}^{(n)}$ is a subset of the set of all states, $\mathcal{K}^{(n)} \subseteq \{\phi_1, \ldots, \phi_K\}^H$, and $\delta(\vec{s}\in\mathcal{K}^{(n)})$ is again an indicator function (one if $\vec{s}\in\mathcal{K}^{(n)}$ and zero otherwise).

While truncated approximations have been shown to represent efficient approximation of high accuracy for a number of sparse coding generative models (Lücke and Eggert, 2010; Bornschein et al., 2013; Henniges et al., 2014), they have so far only been applied

to binary latents. Here, we will generalize the application of truncated distributions to variables with any (finite) number of discrete states.

Considering (2.4), we can first note that the assumptions for applying Expectation Truncation (ET; Lücke and Eggert, 2010) are fulfilled for the DSC model (2.2) and (2.3) such that we can derive a tractable free-energy given by:

$$\mathcal{F}\left(q,\Theta\right) = \sum_{n\in\mathcal{M}}\left[\sum_{\vec{s}} q^{(n)}\left(\vec{s};\Theta^{\mathrm{old}}\right)\left(\log p\left(\vec{y}^{(n)},\vec{s}\,|\,\Theta\right)\right)\right] + H\left(q\right) \tag{2.5}$$

where $q^{(n)}\left(\vec{s};\Theta^{\mathrm{old}}\right)$ is given in (2.4) and where $H(q)$ is the Shannon entropy. Notice that the summation over datapoints is no longer over the index set $\{1,\ldots,N\}$ but over a subset $\mathcal{M}$ of those datapoints that are best explained by the model. Since we use a truncated posterior distribution we expect that we do not explain well the entire dataset but rather a subset of it of size $\sum_{\vec{s}'\in\mathcal{K}^{(n)}} p(\vec{s}'|\Theta)/\sum_{\vec{s}} p(\vec{s}|\Theta)$. To populate $\mathcal{M}$ we use the datapoints with the highest value for $\sum_{\vec{s}\in\mathcal{K}^{(n)}} p\left(\vec{s},\vec{y}^{(n)}|\Theta^{old}\right)$. It can be shown for a large class of generative models (including the DSC model) (Lücke and Eggert, 2010), that maximizing the free-energy (2.5) then approximately maximizes the likelihood for the full dataset.

To get the optimal parameters for the model $\Theta^* = \{\vec{\pi}^*, W^*, \sigma^*\}$ we take the gradient of the free energy and seek the values of the parameters that set it to 0:

$$\begin{aligned}
\nabla\mathcal{F}\left(q,\Theta\right) &= \nabla\sum_{n\in\mathcal{M}}\left[\left\langle\log p\left(\vec{y}^{(n)}|\vec{s},\Theta\right)\right\rangle_{q^{(n)}} + \left\langle\log p\left(\vec{s}|\Theta\right)\right\rangle_{q^{(n)}}\right] \\
&= \nabla\sum_{n\in\mathcal{M}}\left[\left\langle-\frac{D}{2}\log\left(2\pi\sigma^2\right) - \frac{\sigma^2}{2}\|\vec{y}^{(n)} - W\vec{s}\|_2^2\right\rangle_{q^{(n)}}\right. \\
&\qquad\left. + \left\langle\sum_{h,k}\delta\left(\phi_k, s_h\right)\log\pi_k\right\rangle_{q^{(n)}}\right] = 0\,,
\end{aligned}$$

where we denote with $\left\langle g\left(\vec{s}\right)\right\rangle_{q(n)}$ the expectation value of a function $g\left(\vec{s}\right)$ under the distribution $q^{(n)}\left(\vec{s};\Theta^{old}\right)$. For W and σ the results are

$$\nabla_W\mathcal{F}\left(q,\Theta\right) = 0 \Leftrightarrow W^* = \left(\sum_{n\in\mathcal{M}}\vec{y}^{(n)}\left\langle\vec{s}^{T}\right\rangle_{q^{(n)}}\right)\left(\sum_{n\in\mathcal{M}}\left\langle\vec{s}\vec{s}^{T}\right\rangle_{q^{(n)}}\right)^{-1} \tag{2.6}$$

$$\nabla_\sigma\mathcal{F}\left(q,\Theta\right) = 0 \Leftrightarrow \sigma^* = \sqrt{\frac{1}{|\mathcal{M}|D}\left\langle\sum_{n\in\mathcal{M}}\|\vec{y}^{(n)} - W\vec{s}\|_2^2\right\rangle_{q^{(n)}}} \tag{2.7}$$

where $|\mathcal{M}|$ is the size of the set $\mathcal{M}$.

The prior parameter π_k can be obtained in the same way if one introduces the constraint to the free energy of having $\sum_k \pi_k = 1$ to maintain the normalized prior during the gradient procedure.

$$\nabla_{\pi_k} \mathcal{F}(q, \Theta) = 0 \Leftrightarrow \pi_k^* = \frac{\left\langle \sum_h \delta(\vec{s_h}, k) \right\rangle_{q^{(n)}}}{\left\langle \sum_{k,h} \delta(\vec{s_h}, k) \right\rangle_{q^{(n)}}} \tag{2.8}$$

The parameter update equations (2.6), (2.7), and (2.8) require the computation of expectation values $\langle g(\vec{s}) \rangle_{q(n)}$ (the E-step). By inserting the truncated distribution (2.4) we obtain:

$$\langle g(\vec{s}) \rangle_{q^{(n)}} = \sum_{\vec{s}} q^{(n)}(\vec{s}) \; g(\vec{s}) = \frac{\sum_{\vec{s} \in \mathcal{K}^{(n)}} p\left(\vec{s}, \vec{y}^{(n)} | \Theta^{old}\right) g(\vec{s})}{\sum_{\vec{s} \in \mathcal{K}^{(n)}} p(\vec{s}, \vec{y}^{(n)} | \Theta^{old})} \tag{2.9}$$

where $g(\vec{s})$ is a function of the hidden variable $\vec{s}$ (see parameter updates above). As can be observed, the expectation values are now computationally tractable if $|\mathcal{K}^{(n)}|$ is sufficiently small. At the same time, we can expect approximations with high accuracy if $\mathcal{K}^{(n)}$ contains the hidden variables $\vec{s}$ with the large majority of the posterior mass.

In order to select appropriate states $\mathcal{K}^{(n)}$ for a datapoint $\vec{y}$ we use the joint of each datapoint and the singleton posterior variables, i.e. variables that have only one non-zero dimension, to identify the features that are most likely to have contributed to the datapoint and only include those preselected states as in the posterior estimation. More formally, we define

$$\mathcal{K}^{(n)} = \left\{ \vec{s} \,|\, \forall i \notin I^{(n)} : s_i = 0 \text{ and } \|\vec{s}\|_0 \leq \gamma \right\}$$

where $\|\cdot\|_0$ is the non-zero counting norm and where $I^{(n)}$ is an index set that contains the indices of the H' basis functions that are most likely to have generated the datapoint $\vec{y}(n)$. The index set $I^{(n)}$ is in turn defined using a selection (or scoring) function. For our purposes, we here choose a selection function of the following form:

$$\mathcal{S}_h\left(\vec{y}^{(n)}\right) = \max_{\phi' \in \Phi} \left\{ p\left(s_h = \phi, s_{\not h} = 0, \vec{y}^{(n)} | \Theta^{old}\right) \right\}$$

$\mathcal{S}_h$ gives a high value for the index h if the generative field in the h-th column of W contains common structure with the datapoint $\vec{y}^{(n)}$ regardless of the discrete scaling that the model provides. In other words, the selection function uses the best matching discrete value for each generative field as for a comparison with the other generative fields. The H' fields with the largest values of $\mathcal{S}_h\left(\vec{y}^{(n)}\right)$ are then used to construct the set of states in $\mathcal{K}^{(n)}$. Using appropriate approximation parameters γ and H', the sets $\mathcal{K}^{(n)}$ can contain sufficiently many states to realize a very accurate approximation but sufficiently few states in order to warrant sufficiently efficient scalability with H. Crucially, H' can maintain a small value for many types of data while H increases.

The M-step equations (2.6) to (2.8) together with approximate E-step equations (2.9) using the truncated distributions 2.4 represent a learning algorithm for approximate maximization of the data likelihood under the discrete sparse coding model (Equations 2.2 and 2.3). We will refer to this algorithm as the *Discrete Sparse Coding* (DSC) algorithm, or simply DSC.

2.3. Numerical Experiments

We test the DSC algorithm on four different types of data: artificial data, natural images, extra-cellular neuronal recordings, and audio data of human speech. The artificial data are generated using the DSC generative model and they are used to confirm the ability of the DSC algorithm to learn the parameters of the generative model. The other three types of data are commonly encountered in real world scientific tasks. There we show that the DSC algorithm is capable of extracting interesting structure from the data while using discrete latents and small sets of parameters. Notably, the developed algorithm will enable learning of the K parameters of the prior distribution (alongside noise and generative fields).

2.3.1. Artificial Data

We used a linear barstest (Hoyer, 2002; Henniges et al., 2010) to evaluate the ability of the algorithm to recover optimal solutions for the likelihood. We generated $N = 1\,000$ datapoints using a DSC data model configuration with states $\Phi = \{-2, -1, 0, 1, 2\}$ and parameters $\vec{\pi} = (0.025, 0.075, 0.8, 0.075, 0.025)$ for the prior respectively. For the parameters $W \in \{0, 10\}^{H \times D}$, we used $H = 10$ different dictionary elements of $D = 25$ observed dimensions. To simplify the visualization of such high dimensional observations, we choose each dictionary element to resemble a distinct vertical or horizontal bar when reshaped to a 5×5 image, with the value of the bar pixels to be equal to 10 and the background 0, see Figure 2.1 **C**. The resulting datapoints were generated as linear superposition of the basis functions, scaled by a corresponding sample from the prior. Following the DSC generative model, we also added samples of a mean free Gaussian noise with a standard deviation of $\sigma = 2$ to the data, example datapoints can be seen in Figure 2.1 **A**.

Using the generated datapoints, we recovered the ground truth parameters by training the model as described in Section 2. We initialized the standard deviation of the noise model with the standard deviation of the observed variables σ_y, the parameters W with the mean datapoint plus Gaussian noise of with standard deviation of $\sigma_y/4$, and the prior parameters where initialized such that $p(\vec{s}_h = 0) = (H-1)/H$, and $p(\vec{s}_h \neq 0)$ was drawn from a uniformly random distribution and scaled to satisfy the constraint that $\sum_h p(\vec{s}_h) =$

1. The approximation parameters for the truncated approximation scheme were $H' = 7$, and $\gamma = 5$.

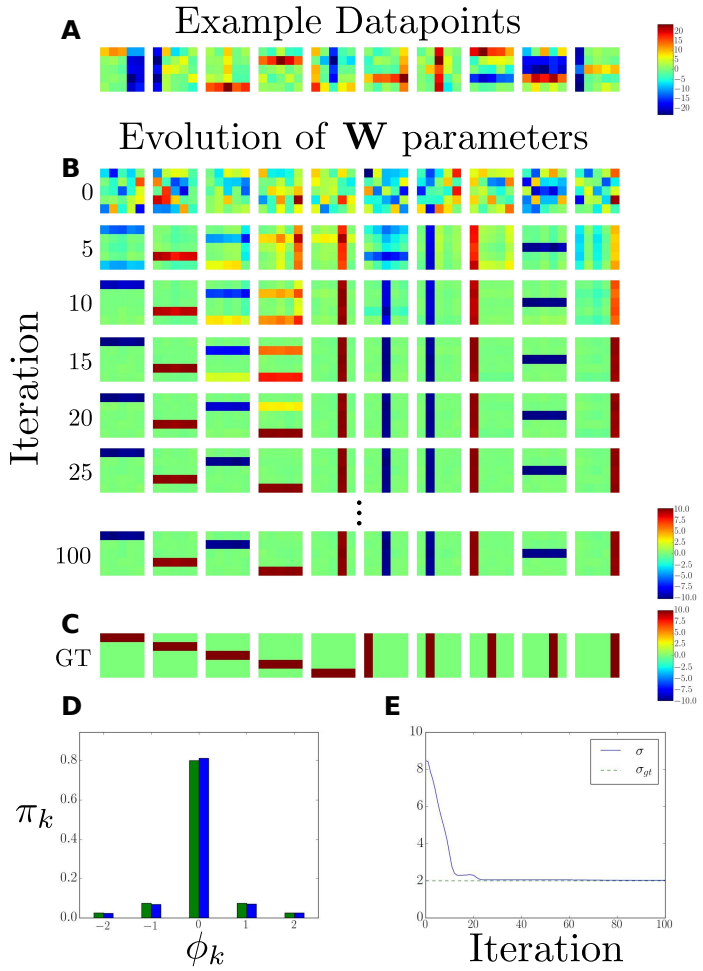

Figure 2.1.: Results from training on natural images using the binary DSC model. **A** Example datapoints sampled from the generative model. **B** The evolution of the dictionary over iterations. Iteration 0 shows the initial values and iteration 100 the dictionary after convergence, no interesting changes occur after iteration 25. **C** The ground truth values for the dictionary. **D** The learned prior parameters (green) compared to the ground truth prior parameters (blue). **E** The evolution of the model standard deviation (solid line) compared to the ground truth(dashed line). Notice that due to the symmetric state configuration the learned dictionary has identical structure with the ground truth but not necessarily the same sign.

We ran the DSC algorithm for 100 iterations using an annealing scheme described in (Ueda and Nakano, 1998; Sahani, 1999) with the value of the temperature parameter to be equal to 1 for the first 10 iterations and linearly decreased to 0, no annealing, by iteration 40. Furthermore, to avoid early rejection of datapoints we used all the datapoints for training for the first 60 iterations and then proceeded to decrease the number of training datapoints linearly to $|M|$ by iteration 90.

After convergence of the algorithm, the learned parameters for σ and $\vec{\pi}$ were observed to match the generating parameters with high accuracy, see Figure 2.1 **D**, and **E**. For the parameters W, we don't recover the exact ground truth parameters, see Figure 2.1 **B**, and **C**. The reason is that in this configuration of the model, and all symmetric ones, there are multiple maximum likelihood solutions since it is equiprobable for a dictionary element to contribute with either sign. Furthermore, we noticed that some configurations of the algorithm are more likely to converge to locally optimal solutions than others.

These results show that we can successfully learn a correct dictionary for the data while at the same time learning a value that parametrizes uncertainty of the discrete coefficients and the scale of the isometric noise of the observed space.

2.3.2. Image Patches

Sparse Coding (Olshausen and Field, 1996) was originally proposed as a sensory coding model for simple cell receptive fields in the primary visual cortex which was able to learn biologically plausible filters from natural image patches. Since then there has been a lot of effort in improving the original SC model, including approaches using alternatives to the originally suggested prior distributions. While by far most work kept focusing on continuous priors, discrete priors in the form of Bernoulli priors for binary latents have been investigated in previously (Haft et al., 2004; Henniges et al., 2010) (also compare non-parametric Bayesian approaches, e.g., Griffiths and Ghahramani (2011)). Furthermore, preliminary work for this study has investigated symmetric priors for three states (-1,0,1) (Exarchakis et al., 2012). We will use these earlier approaches, i.e., (Binary Sparse Coding, BSC, Henniges et al., 2010) and (Ternary Sparse Coding, TSC, Exarchakis et al., 2012), and their application to image patches as a further verification of the DSC algorithm before we proceed to the more general case for this data domain.

The Data. The data set we used for DSC were a selection of images with no artificial structures taken from the van Hateren image data set (van Hateren and van der Schaaf, 1998). We randomly selected $N = 200\,000$ image patches of size 16×16, thus setting the dimensionality of the data to $D = 256$ dimensions. As preprocessing, we first whitened the data using PCA-whitening and then we rotated the whitened data back to the original coordinate space using the set of highest principle components that corresponded 95% of the data variance, this technique is commonly referred to as zero-phase component

whitening or ZCA (Bell and Sejnowski, 1997).

Algorithm details. As in most SC variants, we were concerned with introducing an algorithm that is overcomplete in the absolute number of dimensions. It is worth noting at this point that the dimensionality of the model is not invariant of the model structure so for different configurations the size of the hidden space should also change in order to achieve the same level of accuracy. However, it is more clear to expose this behavior if we use models with constrained dimensionality and we do that by fixing the number of hidden dimension for all tasks to $H = 300$. To maintain similar results across different configurations of the model we use the same training scheme. We ran the DSC algorithm for 200 iterations. To avoid local optima we again used deterministic annealing, as described in (Ueda and Nakano, 1998; Sahani, 1999), with an initial temperature for $T = 2$ that is decreased linearly to $T = 0$ between iterations 10 and 80. Furthermore, in order not to reject any datapoints early in training we used the full data set for the first 20 iterations and linearly decreased it to the set of best explained datapoints $\mathcal{M}$ between iteration 20 and 60. In all cases, we used the same approximation parameters $H' = 8$ and $\gamma = 5$ to maintain a comparable effect of the approximation on the results.

Discrete Sparse Coding with binary latents The binary configuration of DSC (bDSC), with $\Phi = \{0, 1\}$, which recovers the binary sparse coding algorithm shows emergence of Gabor-like receptive fields as expected from (Henniges et al., 2010). The achievements highlighted in (Henniges et al., 2010) were primarily the high dimensional scaling of the latent space, inference of sparsity (a notable difference to Haft et al., 2004), and the recovery of image filters with statistics more familiar to those of primates (Ringach, 2002) than earlier algorithms (Olshausen and Field, 1997), even though later work reportedly improved on that further (Bornschein et al., 2013) (also compare Lücke, 2007, 2009; Rehn and Sommer, 2007; Zylberberg et al., 2011).

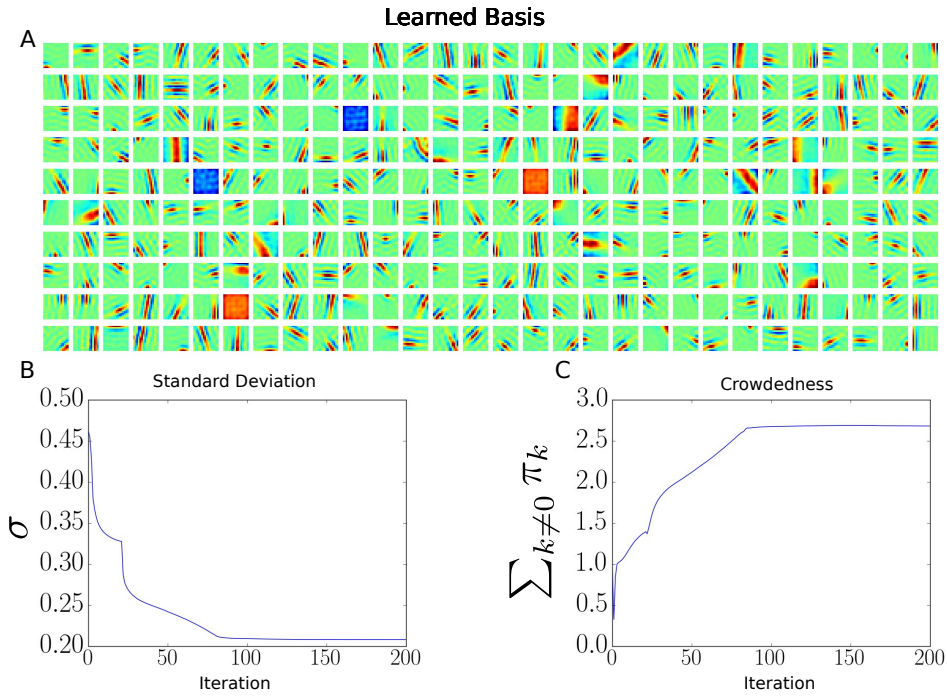

Figure 2.2.: Results from training on natural images using the binary DSC model. **A** Learned dictionary elements. **B** Model uncertainty parameter over EM iterations. **C** Average number of non zero coefficients "crowdedness" over EM iterations.

By applying bDSC reproduce earlier results, i.e., we recover, with Gabor-like and center surround filters, biologically plausible ensemble filters in a high dimensional latent space ($H = 300$), Figure 2.2 summarizes the obtained results. The work in (Henniges et al., 2010) showed results with an even higher number of observed and latent dimensions, however, the binary configuration of DSC has the same algorithmic complexity as BSC and it is trivial to show that DSC can scale to the same size. Here, we chose the lower dimensional observed and latent spaces to facilitate later comparison to the computationally more demanding DSC applications with more latent states. Also note that (Henniges et al., 2010) used a difference-of-Gaussian preprocessing instead of ZCA whitening chosen here and this may have an effect on the resulting parameters.

Discrete Sparse Coding with ternary latents The next more complex DSC configuration we tried is the ternary case (tDSC) in which we use the configuration $\Phi = \{-1, 0, 1\}$. Unlike bDSC, tDSC is symmetric in the state space and therefore shares more features with popular SC algorithms which utilize symmetric priors. However, in this work we study symmetry only in terms of the states (i.e., $-1, 0, 1$) but allow different

prior probabilities for each of these states (unlike Exarchakis et al., 2012, which assumed the same probabilities for states -1, and 1). The approximation parameters and training schedule were set to be identical to the bDSC in order to facilitate the comparison of the two configurations.

As the results in Figure 2.3 **C** show, tDSC converges to an almost symmetric prior, even with non-symmetric initializations of the prior probability of non zero states. For the DSC data model with configuration $\Phi = (-1, 0, 1)$ this means that any generative field is similarly likely as its negative version.

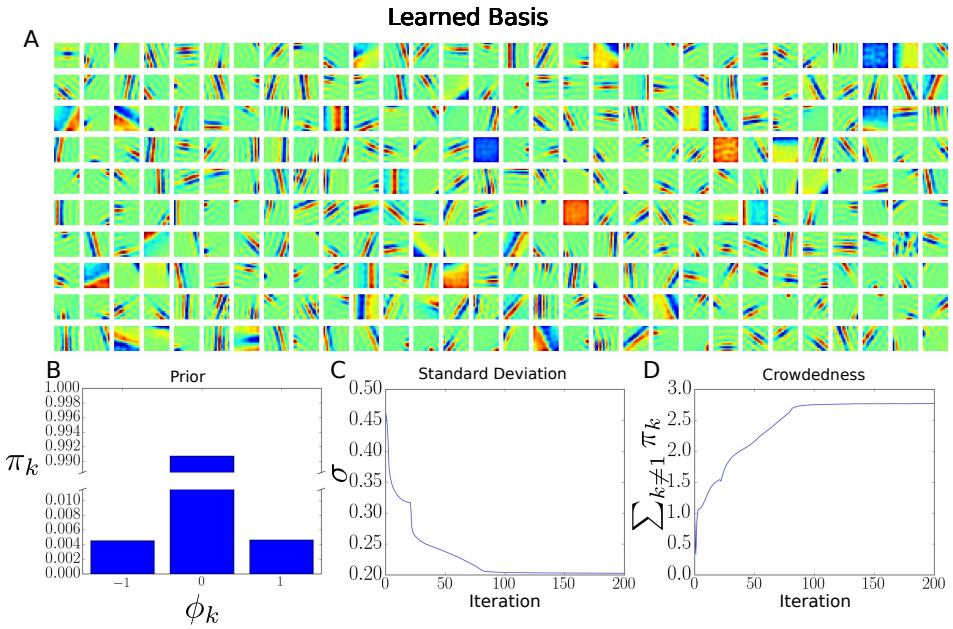

Figure 2.3.: Results from training on natural images using the ternary DSC model ,$-1, 0, 1$. **A** Learned dictionary elements. **B** Learned prior parameters. **C** Model uncertainty parameter over EM iterations. **D** Average number of non zero coefficients "crowdedness" over EM iterations.

Discrete Sparse Coding with multiple positive latents Finally, we turn to a DSC configuration with a greater number of discrete states, and use $\Phi = \{0, 1, 2, 3, 4\}$ to investigate prior probability structures that are not elucidated by the bDSC and tDSC models. Once more, the algorithmic details regarding this run can be viewed at the beginning of section 2.3.2 and they remain the same across all configurations. The only difference across the three different tests is the configurations of the algorithms.

The prior at convergence is monotonically decreasing with the increasing values of the states suggesting that states of higher value have an auxiliary character. Furthermore, the

shape of the prior distributions reinforces the argument for unimodal distributions. The increased number of states also shows a decreased scale for the noise model at convergence suggesting, once more, that an increased number of states provides a better fit for the data, compare to Figure 2.3 and Figure 2.2.

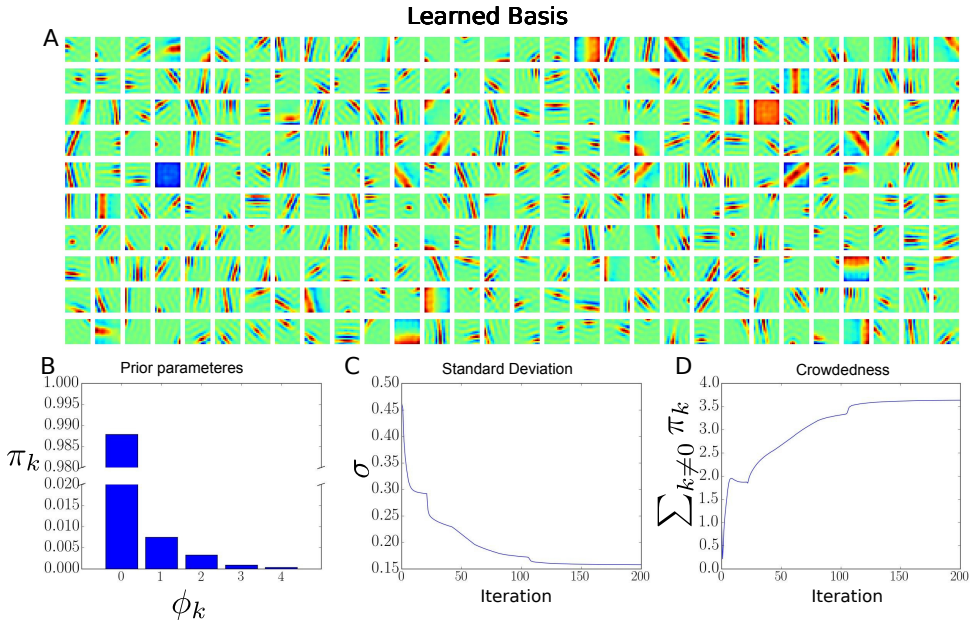

Figure 2.4.: Results from training on natural images using the DSC model with $0, 1, 2, 3, 4$ states. **A** Learned dictionary elements. **B** Prior parameters at convergence. **C** Model uncertainty parameter over EM iterations. **D** Average number of non zero coefficients "crowdedness" over EM iterations.

The learned dictionary, Figure 2.4 **A**, shows a greater preference to localized features suggesting that the algorithm is no longer trying to use multiple dictionary elements to account for the scale in an image. Therefore, scaling "invariance" allows the dictionary to explain finer detail in the structure.

Note that in this configuration the sparsity significantly decreases (crowdedness increases, Figure 2.4) compared to the earlier two DSC configurations. It seems that the scaling invariance, as discussed earlier, encourages the dictionary to explain fine detail and with an increasing number of features explaining fine detail the crowdedness (sparsity decreases) also increases.

The study of natural image statistics is largely centered around sparse coding algorithms with varying priors or introducing novel training schemes. In this work, we introduce an extension of the ET algorithm that is able to learn the structure of the prior distribution for arbitrary discrete states. With the three configurations described in this

section, we have used this feature to provide interesting insights about the structure of the prior. Namely, we deduce that symmetry around 0 is a valid assumption for the definition of a prior distribution and furthermore that distributions monotonically decreasing as they move away from 0 are also a sensible choice. This is the case at least for the frequently used linear superposition assumption and standard Gaussian noise.

2.3.3. Analysis of Neuronal Recordings

Information in the brain is widely considered to be processed in the form of rapid changes of membrane potential across neurons, commonly know as action potentials or spikes. This activity is often viewed as a natural form of discretization of continuous sensory stimuli for later processing in the cortex.

A cost effective way to study the behavior of these neurons and the spike generating process is to perform extra-cellular (EC) electrode recordings. However, when one observes the data obtained from an extra-cellular recording one sees various forms of noise either structured, for instance spikes from remote neurons, or unstructured, such as sensor noise. In this setting, we expect the DSC algorithm to provide interesting insights on the analysis of neural data. Using, different configurations one can either explain overlapping spikes, or as we attempt to show here, use the discrete scaling inherent to the algorithm to explain background spikes, i.e. spikes of remote neurons, and high scaling to explain relevant/near spikes, or the amplitude decay of spike trains using multiple high values.

In this work we will present a study of neural data using the DSC algorithm. To be concise, we will focus on a single configuration of DSC that we believe best elucidates most of the features of the algorithm.

Dataset We used the dataset described in (Henze et al., 2000, 2009). The dataset contains simultaneous intra-cellular and extra-cellular recordings from hippocampus region CA1 of anesthetized rats. We took the first EC channel of recording d533101, sampled at 10 kHz, and band-pass filtered it in the range of $400 - 4000$ Hz and then we sequentially extracted $2ms$ patches of the filtered signal with an overlap of 50%. We used those patches as the training datapoints for our algorithm. We also use the intra-cellular (IC) recording provided by the dataset to better illustrate the properties of the uncertainty involved in EC recordings.

Training We used a DSC configuration with 4 discrete states, $\Phi = \{0, 1, 6, 8\}$, to describe the structure of the data. This configuration was selected using the intuition that spikes of distant neurons will have roughly the same shape as spikes of the relevant neuron but at a smaller scale and therefore correspond to state 1 and the states 6, and 8 will explain features of the relevant neuron or nearby neurons for which we allow some variation in strength. Note that a configuration of $\Phi = \{0, 0.5, 3, 4\}$ would be equivalent because

of the unnormalized columns of W. To choose the best model configuration we could use the variance at convergence as a selection criterion, however, it is useful to make assumptions on a configuration by observing the data. The number of hidden variables, $H = 40$, was selected to be slightly higher than the number of observed variables, $D = 20$, which in turn correspond to $2ms$ of recording sampled at 10 kHz. The approximation parameters for the ET algorithm were set to $H' = 6$ and $\gamma = 4$.

We initialize the noise scale σ as the mean standard deviation of the observed variables, the columns of W using the mean of the datapoints plus a Gaussian noise with standard deviation $\sigma/4$, and the parameters $\vec{\pi}$ we initialized such that $p\left(s_h = 0\right) = \left(H - 1\right)/H$ and $p\left(s_h \neq 0\right)$ is sampled from a uniformly random distribution under the constraint that $\sum_h p\left(s_h\right) = 1$.

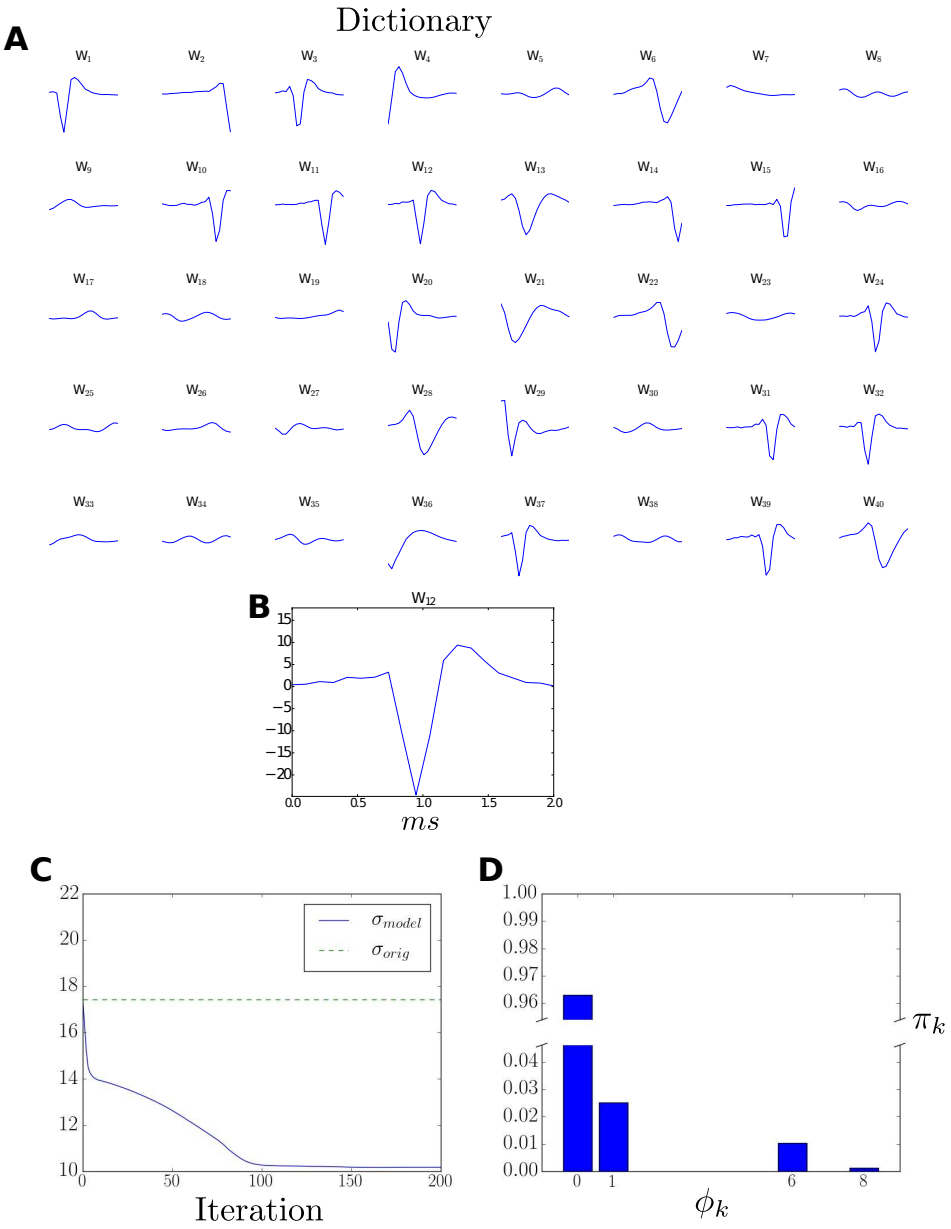

Figure 2.5.: **A** The learned dictionary. Some of the basis develop as extra-cellular recordings of spikes similar to those seen in earlier literature. We also discover components that can only be attributed to structured noise, e.g. from distant neurons. **B** $\vec{W}_{12}$, On the x-axis you see the duration in ms and y-axis the voltage in mV. **C** The evolution of the model σ next to the original data std (dashed). **D** The learned prior parameters $\vec{\pi}$

We let the algorithm run for 200 EM iterations using a deterministic annealing schedule (Ueda and Nakano, 1998; Sahani, 1999) with $T = 1$ for the first 10 iterations and proceed to linearly decreasing it to $T = 0$ by iteration 80. Furthermore, in order to avoid early rejection of interesting datapoints we force the algorithm to learn on all datapoints for the first 60 iterations and then decrease the number of datapoints to $|\mathcal{M}|$ by iteration 100, always maintaining the datapoints with the highest value for $\sum_{\vec{s}\in\mathcal{K}^{(n)}} p(\vec{y}_n, \vec{s})$, see Section 2.2.

In Figure 2.5 **A**, we see the dictionary as was formed at convergence. There we notice potential shifts similar to the ones reported in (Henze et al., 2000) for the extracellular recordings but also other elements that are more similar to finer details of potential changes. Such a decomposition of activity into district subspaces has been shown to improve classification in many tasks and it could prove useful in identifying spiking activity of different neurons in spike sorting systems. One should also take into account that since there is no built-in temporal invariance in the model and there is no spike alignment previously performed in the data, we sometimes observe similar features to appear shifted across the time axis. These temporal shifts emerge as the DSC algorithm addresses temporal alignment by populating the dictionary with time shifted elements. Figure 2.5 **C** shows the evolution of the model noise scale σ compared to the total standard deviation of the original signal. As expected, once our model accounts for the spikes, of near or distant neurons, the noise in the signal becomes smaller. It is worth noting at this point the correlation of σ_{orig} with the presence of spikes in the signal. Figure 2.8 **D** shows the learned prior. The result suggests that most spikes are active with a coefficient of 1 suggesting that they belong to the background noise (modeling distant spike events received by the electrode), then the most active coefficient is 6 suggesting that the dictionary element describes firing scaled down, perhaps due to a spike burst, and the lowest probability latent state is 8 which was intended to model spikes at their highest intensity.

To illustrate how well we were able to fit the data we reconstructed the extra-cellular signal using the latent variables values with highest (approximate) posterior probability), see Figure 2.7. More precisely, for each datapoint $\vec{y}^{(n)}$ we use the $\vec{s}^* \in \mathcal{K}^{(n)}$ that has the highest value for the truncated posterior $q_n(\vec{s})$ and we reconstruct the datapoint using the mean of the noise model $\vec{y}^{(n)} = W\vec{s}^*$. Since the datapoints were selected as consecutive patches of the original recording with a 50% overlap it was necessary to find a sensible way to appropriately reconstruct the overlapping region. We used the reconstruction contributed by the latent vector with the highest truncated posterior to determine the reconstruction at the overlapping region, i.e. $\hat{y}_n^+ = W^+\vec{s}^*$ with $\vec{s}^* = \mathrm{argmax}_{\vec{s}^*}\{q_n(\vec{s}^*), q_{n+1}(\vec{s}^*)\}$ where $\vec{y}_n^+$ is the last 50% of the vector $\vec{y}$ and W^+ the corresponding part of the rows of the W matrix. In Figure 2.7 **A**, we present the reconstruction (green line) of the original extra-cellular signal (blue line). The decomposition of the reconstruction in terms of generative fields (corresponding to the inferred states $\vec{s}^*$ is visualized in Figure 2.7 **B**, **C**, **D**.

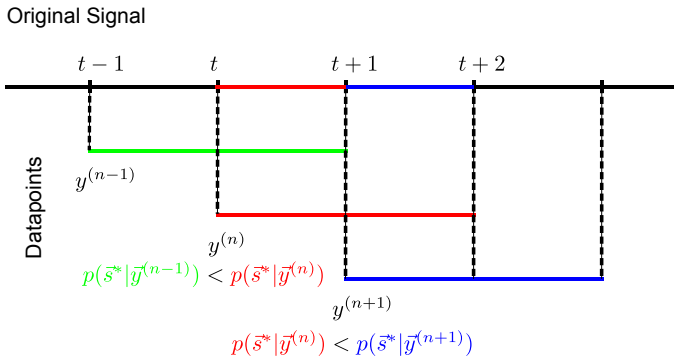

Figure 2.6.: Graphical representation of our treatment of a time series signal. We separate the time series in segments and each datapoint for DSC is a patch of the time series that starts at the first observation of each segment and covers two consecutive segments. To reconstruct a datapoint we use $W\vec{s}^*$, where $\vec{s}^*$ is the MAP vector for each datapoint. To reconstruct a segment that appears in more than one datapoint we use the reconstructed values of the datapoint with the highest approximate posterior.

In Figure 2.8, we show the difference between the reconstructed time series and the original. From the result, we observe that the model does very well at explaining background activity, however, on the locations of some action potentials it appears that there is still relatively high uncertainty. Potentially, the relative frequency of spikes to background is very low and therefore making the spikes a rare event and not captured very well by the model. One could improve on that by either creating higher overlap between consecutive patches allowing the dictionary to explain more details on the potential axis rather than the time axis or to use some spike sorting preprocessing routine, such as spike detection (Quiroga et al., 2004). In Figure 2.8 **C**, we see the corresponding IC recording. Note that only two spikes belong to the targeted neuron.

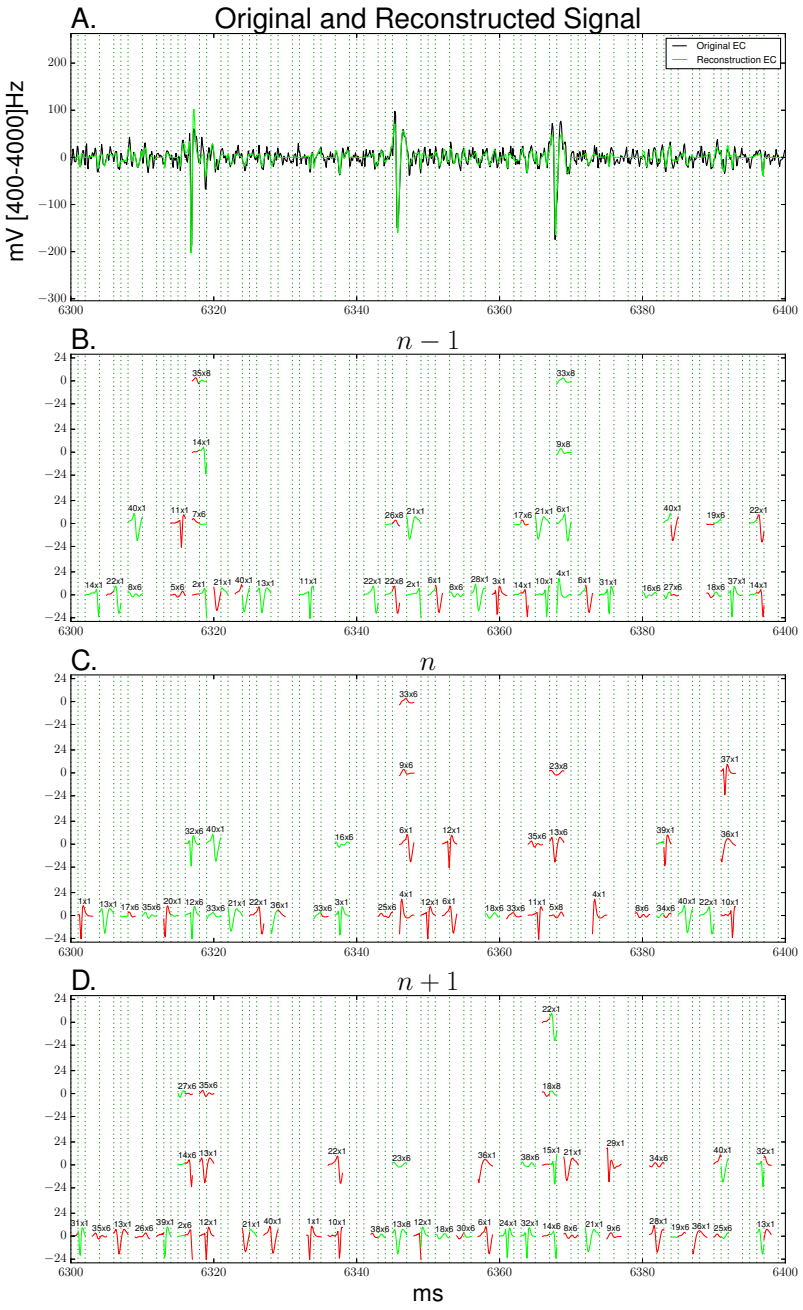

Figure 2.7.: **A** Reconstruction results of an EC recording. **B**-**D** Dictionary elements used to reconstruct the signal. The time-axis are aligned - the plots **B**-**D** represent three consecutive datapoints with 50% overlap. The text above each line denotes the element id times the scaling factor. The green(red) segments of the elements were used(cut out) to reconstruct the corresponding part of the time series.

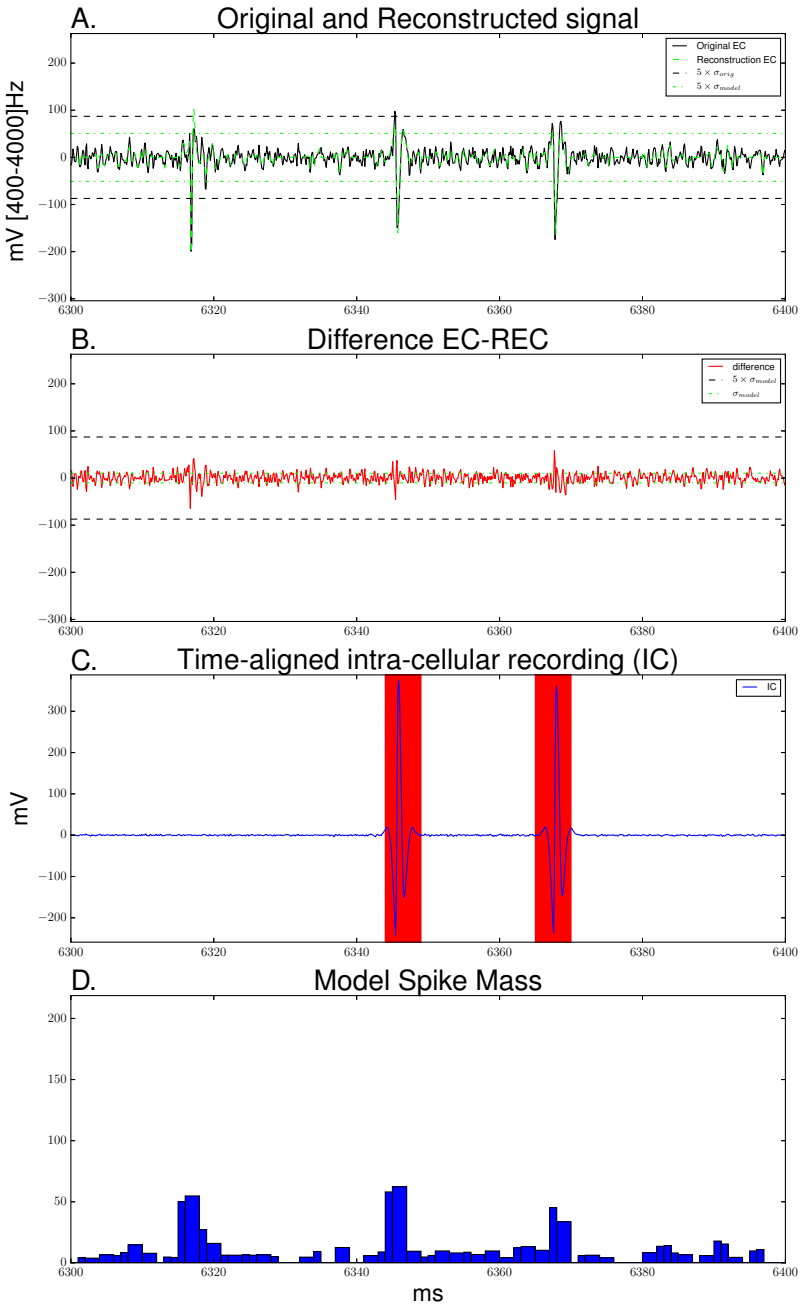

Figure 2.8.: **A** Reconstruction results of an EC recording. **B** The Difference between the reconstructed and the original signal. **C** Time-aligned IC recording - only two of the three clear spikes in **A** correspond to a spike from the targeted neuron. **D** The energy contained in each reconstructed segment estimated using $\sum_h s_h \frac{1}{T} \sum_d |W_{dh}|$.

Part of a spike sorting task is spike identification. Spike identification is usually performed used a threshold over the signal that is taken in a rather ad-hoc way to be at 5 times the standard deviation of the signal, e.g. see $5 \times \sigma_{orig}$ in Figure 2.8 **A** (Quiroga et al., 2004). In Figure 2.8 **D**, we propose an alternative based on the DSC model. The barplot shows the sum of the l_1 norm of all active dictionary elements at that point scaled by the corresponding latent, i.e. $\sum_h s_h \frac{1}{T} \sum_d |W_{dh}|$ for $s_h \in \vec{s}^*$ where $\vec{s}^*$ is the highest posterior latent vector. We expect this quantity to be a better spike detection measure since it is invariant of noise in the signal. For instance, if the neuron was spiking more frequently the threshold $5 \times \sigma_{orig}$ we see in Figure 2.8 **A** would increase but $5 \times \sigma_{model}$ would remain the same because the spikes would be explained by the latent $\vec{s}^*$. That means the threshold, $5 \times \sigma_{orig}$, applied in the signal varies with the neuron firing rate but any threshold imposed on Figure 2.8 **D** would only be affected very mildly by variations in the neuron firing rate.

2.3.4. Audio Data

For our final experimental setting we tested the algorithm on audio data of human speech. We used the TIMIT database (Garofolo et al., 1993) to extract $N = 100\,000$ datapoints $\vec{y}^{(n)}$ in the form of $D = 60$-dimensional consecutive waveforms with an overlap of 50%. We used $H = 100$ hidden random variables to describe the data under the DSC generative model with a configuration $\Phi = \{-2, -1, 0, 1, 2\}$.

For the training, we initialize each column of the dictionary matrix $W \in \mathbb{R}^{D \times H}$ using a non silent datapoint, defined as a datapoint with a norm greater than 1, i.e. $\|\vec{y}^{(n)}\|_1 > 1$. The prior parameter $\vec{\pi}$ was initialized such that $p(s_h = 0) = (H - 1)/H$ and the probabilities of the non zero states we randomly drawn from a uniform distribution under the constraint that $\sum_h p(s_h) = 1$. The scale of the noise model σ was initialized as the average standard deviation of each observed variable.

We ran the DSC algorithm for 200 iteration and verified convergence by a stability check of the parameters over EM iterations. During the run we used an annealing schedule described in (Ueda and Nakano, 1998) with the annealing parameter starting at $T = 1/9$ and decaying it linearly to 0 by iteration 50. We also avoided datapoint cutting until iteration 20 and then proceeded to linearly decrease the datapoints to $|M|$ by iteration 80 as per the algorithm description in section 2.2.

At convergence we noticed that the learned dictionary, see Figure 2.9, is composed of both temporally localized components and global components. The dictionary components are frequently constrained to a single frequency although frequency mixing is not unlikely. The prior emerges to be symmetric around zero even thought no such constraint was imposed by the model and we also see a considerable decrease in the scale of the noise model which suggests a good fit.

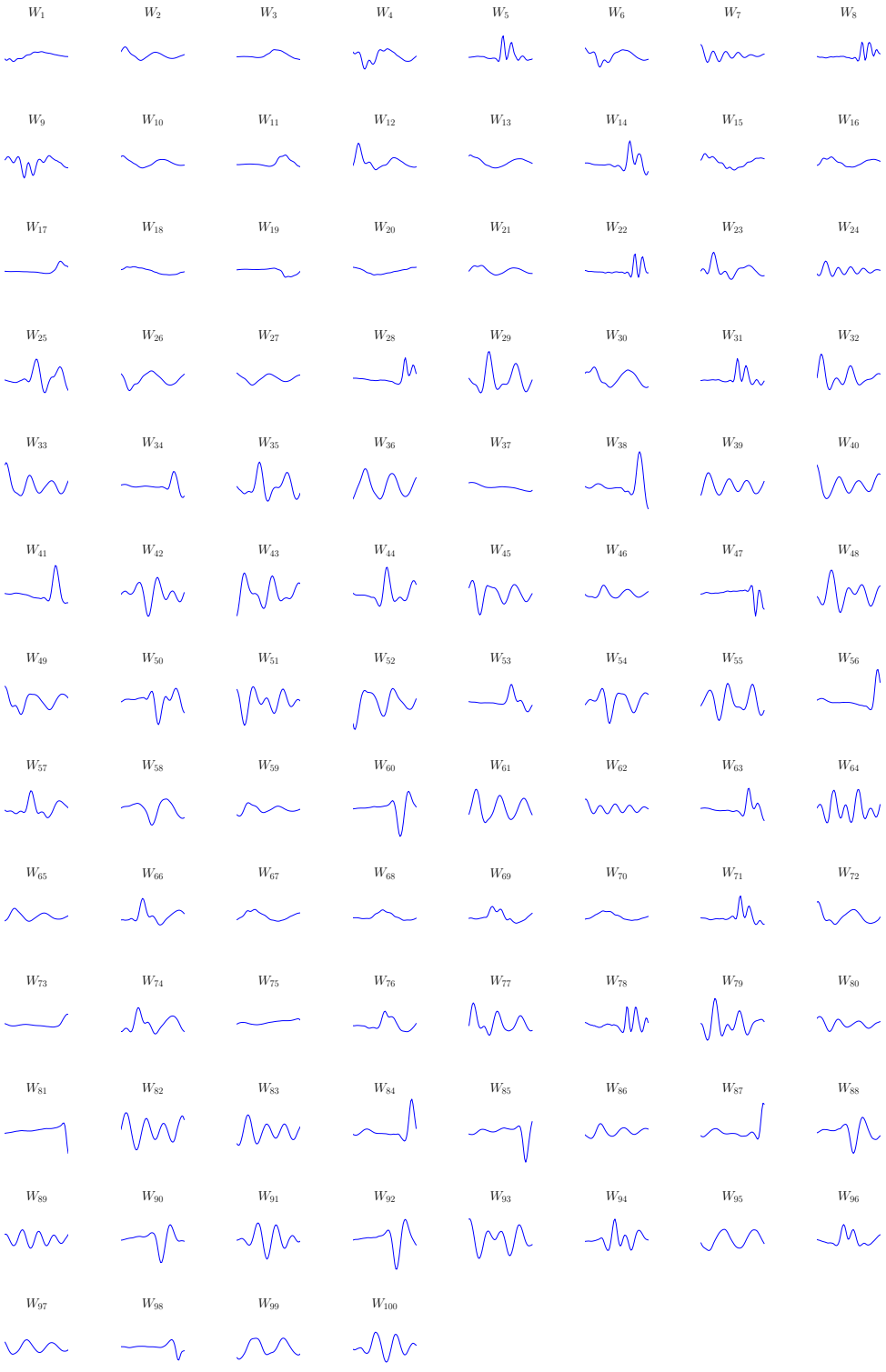

Figure 2.9.: Columns of dictionary matrix, W, after convergence of the algorithm.

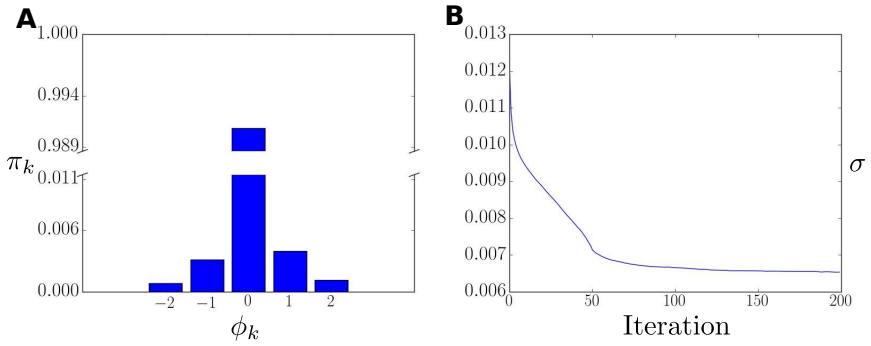

Figure 2.10.: **A** prior parameters at convergence. **B** the evolution of the standard deviation of the model during ET algorithm iterations

Similarly to the neural data analysis section 2.3.3, we used the reconstruction of a time series segment to evaluate how well we were able to fit the data. Once more, for each datapoint $\vec{y}_n$ we use the $\vec{s}^* \in \mathcal{K}^{(n)}$ that has the highest value for the truncated posterior $q_n(\vec{s})$ and we reconstruct the datapoint using the mean of the noise model $\hat{y}_n = W\vec{s}^*$. For the overlapping region we, again, use the reconstruction of the data point with the highest truncated posterior for $\vec{s}^*$. In Figure 2.11 **A**, we can see the reconstruction (red line) of the original waveform (blue line). The decomposition of the reconstruction can be seen in the following to subplots **B**, **C**, **D** over 3 consecutive datapoints $n-1, n, n+1$ respectively. The vertical lines are aligned in time across the four subplots and they represent the time limits for the reconstructed patches

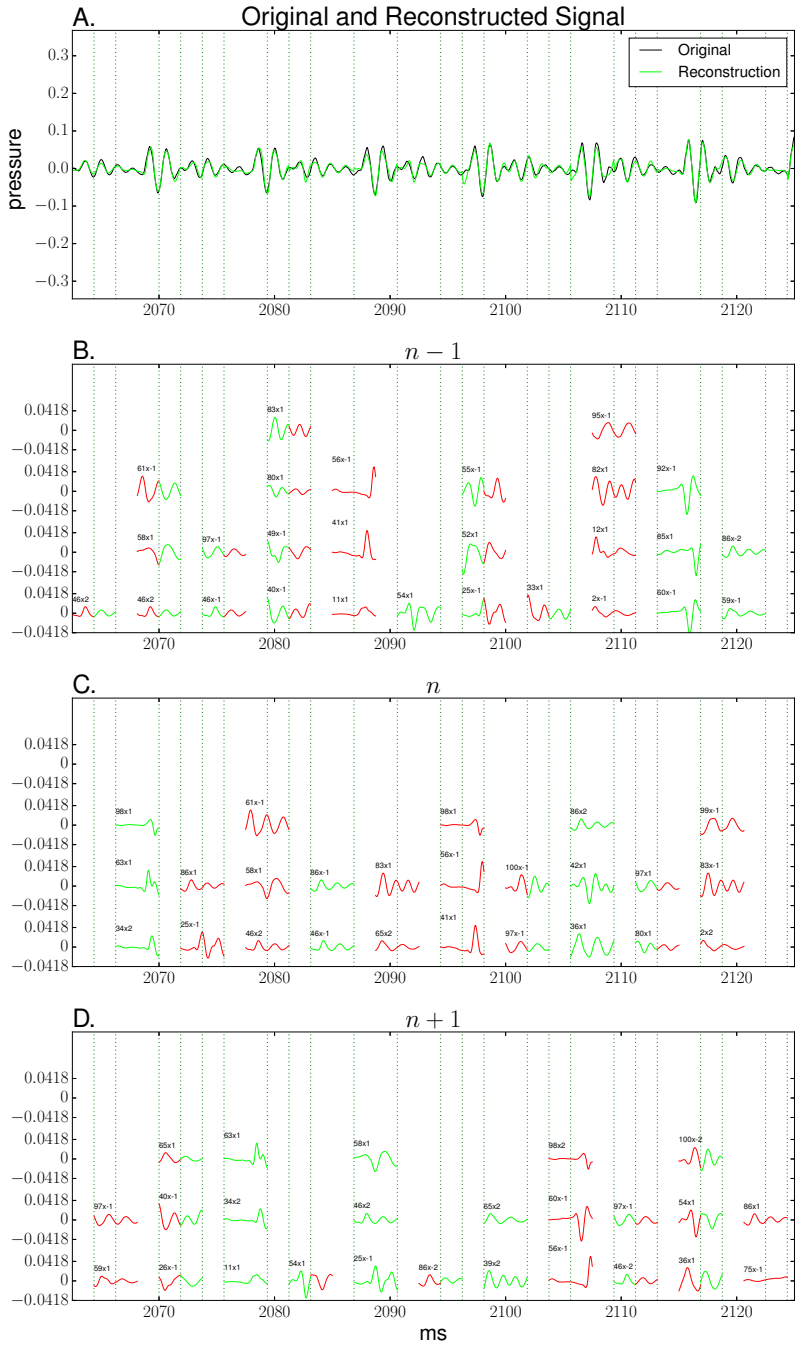

Figure 2.11.: **A** Reconstruction results of an audio waveform. **B**-**D** Dictionary elements used to reconstruct the signal. The time-axis are aligned - the plots **B**-**D** represent three consecutive datapoints. The text above each line denotes the element id times the scaling factor. The green(red) segments of the elements were used(rejected) to reconstruct the corresponding part of the time series.

The red lines, in Figures 2.11 **B-D**, represent the reconstruction of the component with the highest truncated posterior from two consecutive datapoints - used to reconstruct the datapoint. The blue lines represent the component with the lower truncated posterior - rejected for the reconstruction.

2.4. Discussion

We have proposed a novel sparse coding algorithm with discrete latent variables, and we have shown that we are capable of efficiently learning model parameters for generative fields, noise and of the models discrete prior distribution. Efficient learning was realized by adapting truncated approximations (ET; Lücke and Eggert, 2010) to work on latent spaces of multiple discrete states.

In this section we will discuss the interpretation of discrete latents in the Discrete Sparse Coding setting, the significance of varying discrete state spaces in image modeling, the properties that make a Discrete Sparse Coding algorithm relevant to spike analysis of neural data and the efficiency of discrete sparse coding in fitting audio waveform.

Discrete Latent variables for Sparse Coding. Sparse Coding algorithms were originally proposed as a method that deviates from traditional Gaussian encoding schemes to make encoding more selective to an axis and therefore implicitly forcing the features to be more descriptive of a given data structure. Constraining the hidden space to binary values provides an on/off encoding scheme that is selective to image structure aligned with standard Sparse Coding. Discretizing in an arbitrary domain, however, utilizes the sparse coding principle to learn structure in the scale space of the data that would otherwise have been neglected or averaged out, for instance if we used continuous scaling values. The work presented in this paper shows that it is possible to efficiently learn a high dimensional discrete sparse coding model. Furthermore, we have shown that it is possible to learn a wider range of parameters than typical sparse coding algorithms such as the scale of the noise model and, more importantly here, parameters of a flexible prior.

Image Encoding. We have shown that our algorithm was able to scale to several real world high dimensional tasks. For image encoding, we have verified the functionality of our algorithm by first replicating previous (Henniges et al., 2010) and preliminary (Exarchakis et al., 2012) results using specific configurations of the DSC model. Furthermore, we have shown that scaling invariance in image encoding allows the filters to specialize in image structure rather than pixel intensity. Learning the prior parameters in different configurations of DSC has also shown us the distribution of the learned dictionary in scale space without any functional constraints like the ones commonly imposed by sparse coding algorithms with continuous latents (e.g., Laplace distribution/l_1 sparsity penalty).

Identifying the appropriate shape for the prior of the SC latents has been a persistent research area in natural image statistics research (Olshausen and Millman, 2000; Hyvärinen et al., 2005; Berkes et al., 2008; Mohamed et al., 2010; Goodfellow et al., 2013; Sheikh et al., 2014). Since the prior of DSC does not define a density function and it can take arbitrary discrete values, we can use it to try and sketch the necessary qualities of a natural image prior. The combination of the tDSC, and mDSC configurations used in this paper provides a description of desired properties for the form of the prior in sparse coding algorithms with linear superposition and Gaussian noise. Namely, the tDSC results support the use of symmetric priors that have been used ubiquitous in the field and the mDSC supports the argument for unimodal priors, since prior parameters value monotonically decrease away from zero. It is important to note, however, that convergence to local optima has been observed for artificial data making it difficult to guarantee the optimality of the learned shapes. Also note, that the shapes emerge under given the data distributions modeled by the DSC data model. While being general for discrete data, they do use standard assumptions of linear superposition and Gaussian noise model common. These assumptions are shared with the large majority of sparse coding approaches but alternatives models have been suggested in the past (Lücke and Sahani, 2008; Frolov et al., 2016; Bornschein et al., 2013; Henniges et al., 2014).

Discrete Latent variables for Neural Data Analysis. We used neural data to evaluate the performance of our algorithm due to their popular interpretation as sequences of discrete events. In this analysis, we showed that DSC can learn spike and sub-spike features that sufficiently describe neural recordings. Furthermore, carefully selecting the scale space makes it possible to discern physiological characteristics of temporal alignment, for instance whether a given spike is the initial event or a secondary spike in a spike burst. Notably, one of the most unique features of our algorithm, learning prior parameters was very informative about the structure of extra-cellular (EC) recordings. The learned prior interprets EC recordings as being composed of a multitude of spiking patterns coming from a population of neurons around the targeted neuron. Furthermore, the fact that we can learn a Gaussian noise model distinct from the spiking activity provides a more clear separation of noise from spikes than those traditionally seen in spike sorting tasks (Quiroga et al., 2004).

Discrete Latent variables for Audio Data The DSC algorithm was fitted to audio data successfully. The reconstruction has shown intelligible speech even though we did not use any hand crafted features of human speech suggesting that we were able to learn elementary short-time primitives human speech. The fact that a sparse discrete encoding of speech can capture these features implies that such a prior can be useful for speech encoding.

Conclusion To conclude we have derived, implemented and tested a novel sparse coding algorithm. Whenever it is reasonable to assume that the hidden variables are discrete, the studied approach offers itself to learn a statistical data model (which can then be used for different tasks). Furthermore, our model covers the general class of (finite) discrete priors under the canonical sparse coding assumptions of iid and sparsely distributed latents. Its parameterization of discrete latents is thus much more general than any example of sparse coding with continuous latents. As sparse coding plays an important role in Machine Learning and its many fields of application, we believe that Discrete Sparse Coding will be the method of choice for a large number of applications.

3. Time-Invariant Discrete Sparse Coding

3.1. Introduction

We have seen that discrete sparse coding can have many useful applications and provides interesting insights in a variety of applications. However, especially when we have to deal with data that have temporal structure there are issues that DSC does not address and we have to use rather crude ways of manipulating the data, e.g. overlapping patches. In the neural data example in particular we have a discrete encoding that represents spiking patterns in extra-cellular recordings. However, as we can see from the dictionary elements in order to represent spikes at different locations in time we need to use multiple dictionary elements. In a spike sorting task that would be an undesirable effect. Ideally, a generative model would assign one latent variable for each different neuron that affects the electrode. To address that issue in this section we introduce an extension to the DSC model that captures the temporal alignment of a dictionary element using and additional latent variable. This additional variable separates the signal into two structural element of the signal the location and its content. Introducing an additional latent variable increases the size of the posterior space significantly. However, it has been shown (Dai et al., 2013) that truncated methods can deal with features invariant to translation. It is our intension to introduce a similar extension for the DSC model in this work.

3.2. Mathematical Description

Consider a set, Y, of N independent datapoints $\vec{y}^{(n)}$, with $n = 1, \ldots, N$, where $\vec{y}^{(n)} \in \mathbb{R}^D$. For these data the studied learning algorithm seeks parameters $\Theta^* = \{W^*, \sigma^*, \vec{\pi}^*\}$ that maximize the data log-likelihood:

$$\mathcal{L}\left(Y|\Theta\right) = \log \prod_{n=1}^{N} p\left(\vec{y}^{(n)}|\Theta\right) = \sum_{n=1}^{N} \log p\left(\vec{y}^{(n)}|\Theta\right)$$

Sparse coding models are latent variable models and therefore the likelihood is defined as a function of unobserved random variables as follows

$$\mathcal{L}\left(\vec{\boldsymbol{y}}|\Theta\right) = \sum_{n=1}^{N} \log p\left(\vec{y}^{(n)}|\Theta\right) = \sum_{n=1}^{N} \log \sum_{\vec{s},\vec{t}} p\left(\vec{y}^{(n)}|\vec{s},\vec{t},\Theta\right) p\left(\vec{s},\vec{t}|\Theta\right) \tag{3.1}$$

where the latent variables $\vec{s}$, are taken to have discrete values, similarly to DSC, and the latent variables $\vec{t}$, are assumed to encode all possible translations for a dictionary element. Therefore, the sum $\sum_{\vec{s},\vec{t}}$ now goes over all different combinations of discrete states and all differente temporal displacements. Let $\vec{s}$ be of length H, i.e. $\vec{s} = (s_1, \ldots, s_H)^T$, where each element s_h can take on one of K discrete values $\phi_k \in \mathbb{R}$, i.e. $s_h \in \Phi = \{\phi_1, \ldots, \phi_K\}$. Also, let $\vec{t}$ be of length H, i.e. $\vec{t} = (t_1, \ldots, t_H)^T$, where each element t_h can take on one of τ discrete values representing indices in the range $\{1, \ldots, \tau\}$, i.e. $t_h \in \{1, \ldots, \tau\}$. For such latents, we can define the following prior:

$$p(\vec{s},\vec{t}\,|\,\Theta) \;=\; \prod_{h=1}^{H}\prod_{k=1}^{K} \frac{\pi_k}{\tau}^{\delta(\phi_k = s_h)}, \quad \text{with} \quad \sum_{k=1}^{K} \pi_k = 1, \tag{3.2}$$

where $\delta\left(\phi_k = s_h\right)$ is an indicator function which is one if and only if $\phi_k = s_h$ and zero otherwise. Similarly to DSC, we have a different probability of presence for different states, paramterized by $\vec{\pi}$, and the different states Φ describe, again, a different configuration. We assume that the prior over the variables t_h is uniform. As for standard sparse coding, Equation 3.2 assumes independent and identical distributions for each pair of latents s_h, t_h. The prior will be used to model sparse activity by demanding one of the values in $\Phi = \{\phi_1, \ldots, \phi_K\}$ to be zero and the corresponding probability to be relatively high.

Having defined the prior (3.2), we assume the observed variables $\vec{y}$ to be generated as follows. For each column of the dictionary matrix, $W \in \mathbb{R}^{D_w \times H}$, we crop a patch of size D_y starting from dimension t_h, we will denote this operation as $f\left(W, \vec{t}\right) = W^{(t)}$, where $W^{(t)} \in \mathbb{R}^{D_y \times H}$. The assumption here is that each datapoint $\vec{y}$ offers a view of the data that is constrained in time and through the use of the latent variable $\vec{t}$ we model the precise time in which an event took place. This definition for the generative process assumes that $D_w > D_y$, and $\tau = D_w - D_y + 1$. Once we have generated the temporally constrained dictionary matrix, we proceed to generating the datapoints as in DSC, i.e., we assume that $\vec{y}$ is governed by a Gaussian noise model with a mean set by a linear superposition of the latents s_h:

$$p\left(\vec{y}\,|\,\vec{s},\vec{t},\Theta\right) = \mathcal{N}\left(\vec{y}; f\left(W,\vec{t}\right)\vec{s}, \sigma^2\mathbb{1}\right) = \mathcal{N}\left(\vec{y}; W^{(t)}\vec{s}, \sigma^2\mathbb{1}\right) \tag{3.3}$$

with an isotropic covariance, $\sigma^2\mathbb{1}$, and mean $W^{(t)}\vec{s}$. We call the data model defined by

(3.2) and (3.3) the time-invariant discrete sparse coding (iDSC) data model.

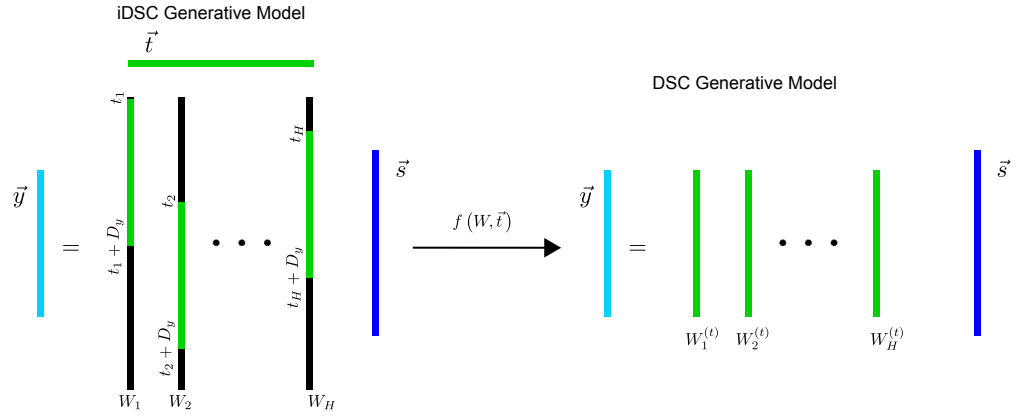

Figure 3.1.: A Graphical representation of the iDSC Generative Model. The figure shows the effect of the variable $\vec{t}$ on the dictionary elements. The black vertical bar are the columns of the dictionary matrix W. The vector $\vec{t}$ selects a subspace of each column that is best aligned with $\vec{y}$ and has the same dimensionality as $\vec{y}$ through the function $f\left(W,\vec{t}\right) = W^{(t)}$. Therefore, given a variable $\vec{t}$ we fall back to the DSC generative model $p\left(\vec{y},\vec{s}|\vec{t}\right)$

Given a set of datapoints $\vec{y}^{(1)},\ldots,\vec{y}^{(N)}$ and the DSC data model, we now seek parameters $\Theta = (W,\sigma,\vec{\pi})$ that maximize the likelihood (3.1). We derive parameter update equations using Expectation Maximization in its free-energy formulation (Neal and Hinton, 1998). In our case, exact EM update equations can be derived in closed-form but the E-step scales with the number of hidden states $\mathcal{O}(K^H\tau^H)$, making the algorithm computationally intractable for large H.

In order to derive computationally tractable approximations for parameter optimization, we approximate the intractable a-posteriori probabilities $p(\vec{s},\vec{t}\,|\,\vec{y},\Theta)$ by a truncated distribution:

$$p(\vec{s},\vec{t}\,|\,\vec{y}^{(n)},\Theta) \;\approx\; q^{(n)}(\vec{s},\vec{t};\Theta) \;=\; \frac{p(\vec{s},\vec{t}\,|\,\vec{y}^{(n)},\Theta)}{\sum_{\vec{t}'\in\mathcal{T}^{(n)},\vec{s}'\in\mathcal{K}^{(n)}} p(\vec{s}'\,|\,\vec{y}^{(n)},\Theta)}\,\delta(\vec{s}\in\mathcal{K}^{(n)})\delta(\vec{t}\in\mathcal{T}^{(n)}), \tag{3.4}$$

where $\mathcal{K}^{(n)}$ is a subset of the set of all states, $\mathcal{K}^{(n)} \subseteq \{\phi_1,\ldots,\phi_K\}^H$, $\mathcal{T}^{(n)}$ is a subset of the set of all states, $\mathcal{T}^{(n)} \subseteq \{0,\ldots,\tau\}^H$, and $\delta(\vec{s}\in\mathcal{K}^{(n)})$ and $\delta(\vec{t}\in\mathcal{T}^{(n)})$ are again an indicator functions.

In an earlier section 2.2, we covered a truncated approximation for DSC. Here, we will go through a further extension of a truncted EM approximation that covers latent variables for temporal shifts as well as scaling coefficients.

Considering (3.4), we can first note that the assumptions for applying Expectation Truncation (ET; Lücke and Eggert, 2010) are fulfilled for the iDSC model (3.2) and (3.3) such that we can derive a tractable free-energy given by:

$$\mathcal{F}\left(q,\Theta\right)=\sum_{n\in\mathcal{M}}\left[\sum_{\vec{s},\vec{t}}q^{(n)}\left(\vec{s},\vec{t};\Theta^{\text{old}}\right)\left(\log p\left(\vec{y}^{(n)},\vec{s},\vec{t}\,|\,\Theta\right)\right)\right]+H\left(q\right) \tag{3.5}$$

where $q^{(n)}\left(\vec{s},\vec{t};\Theta^{\text{old}}\right)$ is given in (3.4) and where $H(q)$ is the Shannon entropy. Notice that the summation over datapoints is no longer over the index set $\{1,\ldots,N\}$ but over a subset $\mathcal{M}$ of those datapoints that are best explained by the model. Since we use a truncated posterior distribution we expect that we do not explain well the entire dataset but rather a subset of it of size $\sum_{\vec{s}'\in\mathcal{K}^{(n)},\vec{t}'\in\mathcal{T}^{(n)}}p(\vec{s}',\vec{t}'|\Theta)/\sum_{\vec{s},\vec{t}}p(\vec{s},\vec{t}|\Theta)$. To populate $\mathcal{M}$ we use the datapoints with the highest value for $\sum_{\vec{s}\in\mathcal{K}^{(n)},\vec{t}\in\mathcal{T}^{(n)}}p\left(\vec{s},\vec{t},\vec{y}^{(n)}|\Theta^{\text{old}}\right)$. It can be shown for a large class of generative models (including the iDSC model) (Lücke and Eggert, 2010), that maximizing the free-energy (3.5) then approximately maximizes the likelihood for the full dataset.

To get the optimal parameters for the model $\Theta^{*}=\{W^{*},\sigma^{*},\vec{\pi}^{*}\}$ we take the gradient of the free energy and seek the values of the parameters that set it to 0:

$$\begin{aligned}\nabla\mathcal{F}\left(q,\Theta\right)&=\nabla\sum_{n\in\mathcal{M}}\left[\left\langle\log p\left(\vec{y}^{(n)}|\vec{s},\vec{t},\,\Theta\right)\right\rangle_{q^{(n)}}+\left\langle\log p\left(\vec{s},\vec{t}|\Theta\right)\right\rangle_{q^{(n)}}\right]\\&=\nabla\sum_{n\in\mathcal{M}}\left[\left\langle-\frac{D}{2}\log\left(2\pi\sigma^{2}\right)-\frac{\sigma^{2}}{2}\|\vec{y}^{(n)}-f\left(W,\vec{t}\right)\vec{s}\|_{2}^{2}\right\rangle_{q^{(n)}}\right.\\&\qquad\left.+\left\langle\sum_{h,k}\delta\left(\phi_{k},s_{h}\right)\log\frac{\pi_{k}}{\tau}\right\rangle_{q^{(n)}}\right]=0\,,\end{aligned}$$

where we denote with $\langle g\left(\vec{s}\right)\rangle_{q(n)}$ the expectation value of a function $g\left(\vec{s}\right)$ under the distribution $q^{(n)}\left(\vec{s};\Theta^{\text{old}}\right)$. For W and σ the results are

$$\nabla_{W}\mathcal{F}\left(q,\Theta\right)=0\Leftrightarrow W^{*}=\left(\sum_{n\in\mathcal{M}}\left\langle\vec{y}_{-\vec{t}}^{(n)}\vec{s}^{T}\right\rangle_{q^{(n)}}\right)\left(\sum_{n\in\mathcal{M}}\left\langle\vec{s}\vec{s}^{T}\right\rangle_{q^{(n)}}\right)^{-1} \tag{3.6}$$

$$\nabla_{\sigma}\mathcal{F}\left(q,\Theta\right)=0\Leftrightarrow\sigma^{*}=\sqrt{\frac{1}{|\mathcal{M}|D}\left\langle\sum_{n\in\mathcal{M}}\|\vec{y}^{(n)}-f\left(W,\vec{t}\right)\vec{s}\|_{2}^{2}\right\rangle_{q^{(n)}}} \tag{3.7}$$

where $|\mathcal{M}|$ is the size of the set $\mathcal{M}$. Notice that Equation 3.6 changes when compared to DSC in order to account for proper alignment of the datapoints. The vector $\vec{y}_{-t}^{(n)}$ is of the

same dimensionality as a column W and it contains the datapoint $\vec{y}^{(n)}$ in the dimensions specified by variable $\vec{t}$ and 0 everywhere else. [1]

The prior parameter π_k can be obtained in the same way if one introduces the constraint to the free energy of having $\sum_k \pi_k = 1$ to maintain the normalized prior during the gradient procedure.

$$\nabla_{\pi_k} \mathcal{F}(q, \Theta) = 0 \Leftrightarrow \pi_k^* = \frac{\left\langle \sum_h \delta(\vec{s_h}, k) \right\rangle_{q^{(n)}}}{\left\langle \sum_{k,h} \delta(\vec{s_h}, k) \right\rangle_{q^{(n)}}} \tag{3.8}$$

The parameter update equations (3.6), (3.7), and (3.8) require the computation of expectation values $\left\langle g\left(\vec{s}, \vec{t}\right) \right\rangle_{q(n)}$ (the E-step). By inserting the truncated distribution (3.4) we obtain:

$$\left\langle g\left(\vec{s}, \vec{t}\right) \right\rangle_{q^{(n)}} = \sum_{\vec{s},\vec{t}} q^{(n)}\left(\vec{s}, \vec{t}\right)\ g\left(\vec{s}, \vec{t}\right) = \frac{\sum_{\vec{s}\in\mathcal{K}^{(n)}, \vec{t}\in\mathcal{T}^{(n)}} p\left(\vec{s}, \vec{t}, \vec{y}^{(n)} | \Theta^{\text{old}}\right) g(\vec{s})}{\sum_{\vec{s}\in\mathcal{K}^{(n)}, \vec{t}\in\mathcal{T}^{(n)}} p\left(\vec{s}, \vec{t}, \vec{y}^{(n)} | \Theta^{\text{old}}\right)} \tag{3.9}$$

where $g(\vec{s})$ is a function of the hidden variable $\vec{s}$ (see parameter updates above). As can be observed, the expectation values are now computationally tractable if $|\mathcal{K}^{(n)}|$, and $|\mathcal{T}^{(n)}|$ are sufficiently small. At the same time, we can expect approximations with high accuracy if $\mathcal{K}^{(n)}$,$\mathcal{T}^{(n)}$ contain the hidden variables $\vec{s}$, and $\vec{t}$ with the large majority of the posterior mass.

In order to select appropriate states $\mathcal{K}^{(n)}$ for a datapoint $\vec{y}$ we use the joint of each datapoint, the singleton posterior variables $\vec{s}$, i.e. variables $\vec{s}$ that have only one non-zero dimension, and all the time shifts for the non-zero dimensions, to identify the features that are most likely to have contributed to the datapoint and only include those preselected states as in the posterior estimation. More formally, we define

$$\mathcal{K}^{(n)} = \left\{ \vec{s} \,|\, \forall i \notin I^{(n)} : s_i = 0 \text{ and } \|\vec{s}\|_0 \leq \gamma \right\}$$

where $\|\cdot\|_0$ is the non-zero counting norm, $I^{(n)}$ is an index set that contains the indices of the H' basis functions that are most likely to have generated the datapoint $\vec{y}(n)$. The index set $I^{(n)}$ is in turn defined using a selection (or scoring) function. For our purposes, we here choose a selection function of the following form:

$$\mathcal{S}_h\left(\vec{y}^{(n)}\right) = \max_{\phi' \in \Phi} \left\{ p\left(s_h = \phi, s_{\not h} = 0, t_h, \vec{y}^{(n)} | \Theta^{\text{old}}\right) \right\}$$

$\mathcal{S}_h$ gives a high value for the index h if the generative field in the h-th column of W contains common structure with the datapoint $\vec{y}^{(n)}$ regardless of the discrete scaling or the temporal shift that the model provides. The H' fields with the largest values of $\mathcal{S}_h\left(\vec{y}^{(n)}\right)$

[1] For more details see appendix B.1

are then used to construct the set of states in $\mathcal{K}^{(n)}$, i.e. their indices are used to form the index set $I^{(n)}$.

The set of time shifts $\mathcal{T}^{(n)}$ that we use for the truncated posterior of a datapoint $\vec{y}^{(n)}$ is populated, again, through the use of the joint probability of each datapoint $\vec{y}^{(n)}$, the singleton states $\vec{s}$, and all the time shifts each dictionary element only this time we maintain the most likely time shifts for each dicitonary element.

$$\mathcal{T}^{(n)} = \left\{ \vec{t} \,|\, t_h = \rho^{(n)}_{-\beta}(h) \right\}$$

where $\rho^{(n)}_{-\beta}(h)$ is a sequence of indices $t_h \in \{1, \ldots, \tau\}$ that yield the β highest values for $p\left(s_h \in \Phi, s_{\not h} = 0, t_h \,|\Theta^{\text{old}}\right)$. We therefore store in $\mathcal{T}^{(n)}$ the β different time shifts that align a dictionary element with the datapoint $\vec{y}_n$ best independently for each dictionary element. Using appropriate approximation parameters β, γ and H', the sets $\mathcal{K}^{(n)}$, and $\mathcal{T}^{(n)}$ can contain sufficiently many latent variables to realize a very accurate approximation at a sufficiently low computational cost even for high dimensional problems, i.e. our latent space no longer scales with the variable H.

Equations (3.6) to (3.8) and Equation (3.9) form the M-step and E-step of a truncated EM algorithm that maximizes the data likelihood of the iDSC generative model (Equations 3.2 and 3.3).

3.3. Numerical Experiments

In the Discrete Sparse Coding chapter (2) we introduced a method of applying the DSC algorithm on time series data by separating the sequence of observations into overlapping segments to account for the temporal structure in the data. That was a rather crude treatment of the temporal structure of the signal which forced the algorithm to compensate using multiple generative fields for the same spikes at different temporal locations. The iDSC algorithm is better equipped to deal with discrete events in time series data. Using the additional variables for temporal alignment, we expect to extract features that are invariant to temporal alignment. In this section we apply the iDSC algorithm to the extracellular neuron recordings dataset we used in section 2.3.3 to examine whether we can achieve a better separation of spiking neurons. We also apply the iDSC algorithm to the audio dataset we used in section 2.3.4 to examine the behavior of the effect of temporal alignment on audio data of human speech.

3.3.1. Spike Sorting

In 2.3.3 we saw the DSC algorithm applied to patches of extra-cellular (EC) recordings. The main idea there was to identify discrete events (e.g. spikes) in a continuous signal.

Spiking neurons are characterized by the shape of their spiking potential and the timing of the spiking event. In the neural data analysis, in 2.3.3, we noticed that the dictionary contained spikes of the same shape but slightly shifted left or right. This behavior suggests that the variable responsible to identify the presence of a spike or not in the signal ($\vec{s}$) was also responsible for the timing of the spike. Using the iDSC model, we have a clear separation between variables that encode the timing and variables that encode the presence of a spike. Therefore, we expect the iDSC model to characterize a spiking neurons contribution to the recording more rigorously and successfully separate across different spiking neurons, essentially solving the spike sorting problem.

Here, we will apply the iDSC algorithm on the same dataset of neural recordings as in 2.3.3 and compare the behavior of the two algorithms.

Dataset We used the same dataset (Henze et al., 2000, 2009) as in the neural data analysis of the earlier section 2.3.3. The dataset contains simultaneous intra-cellular and extra-cellular recordings from hippocampus region CA1 of anesthetized rats. Once more, we took the first EC channel of recording d533101, sampled at 10 kHz, and band-pass filtered it in the range of $400 - 4000$ Hz and then we sequentially extracted $2ms$ patches of the filtered signal. This time there was no overlap between consecutive patches since we assume that the temporal component of the iDSC data model will generate a smooth enough result. As in the earlier section we compare against the temporally aligned intra-cellular (IC) recording to evaluate the the correspondence between the representation learned by the iDSC algorithm and the behavior of the target neuron.

Training We used an iDSC configuration with 2 discrete states, $\Phi = \{0, 1\}$, to describe the structure of the data. We selected this configuration because we want to focus on the discrete nature of the spiking activity, and the timing of a spike and not any potential scaling. It is worth to note, however, that scaling behavior changes across DSC and iDSC algorithms suggesting that there is an interplay amongst latent variables responsible for time and scale[2], i.e. scaling appears to account for timing if it is not explicitly modeled. The number of hidden variables, $H = 10$, was selected as the anticipated number of distinct neurons that affect the electrode (as reported by Henze et al., 2000), $D_y = 20$, which in turn correspond to $2ms$ of recording sampled at 10 kHz. The dimensionality of the dictionary elements was set to be $D_w = 40$ so that a dictionary element can be aligned throughout the length of a datapoint. The approximation parameters for the iDSC algorithm were set to $H' = 6$, $\gamma = 4$, and $\beta = 10$ to maintain similar efficiency as in the DSC case.

We used the same initialization strategy as for the DSC algorithm. Therefore, the standard deviation σ of the Gaussian noise model (Equation 3.3) is initialized as the

[2] see Discussion at the end of the chapter.

mean standard deviation of the observed variables, the columns of the dictionary matrix, W, using the mean of the datapoints plus a Gaussian noise with standard deviation $\sigma/4$, and the prior parameters $\vec{\pi}$ we initialized such that $p\left(s_h = 0\right) = \left(H - 1\right)/H$, and $p\left(s_h = 1\right) = 1/H$. We have observed that this initialization strategy results in more stable behavior of the algorithm during the early stages of EM iterations.

The learning schedule remained consistent with DSC without noticeable issues. That is we let the algorithm run for 100 EM iterations using a deterministic annealing schedule (Ueda and Nakano, 1998; Sahani, 1999) with $T = 1$ for the first 5 iterations and proceed to linearly decreasing it to $T = 0$ by iteration 40. Furthermore, in order to avoid early rejection of interesting datapoints [3] we force the algorithm to learn on all datapoints for the first 30 iterations and then decrease the number of datapoints to $|\mathcal{M}|$ by iteration 50, always maintaining the datapoints with the highest value for $\sum_{\vec{s}\in\mathcal{K}^{(n)}} p(\vec{y}_n, \vec{s})$, see Section 3.2.

In Figure 3.2 **A**, we see the columns of the dictionary matrix at convergence. Each of the generative fields corresponds to a segment of the time series that is twice as long as the corresponding datapoint. We observe that most of the activity in a filter is gathered near the center, presumably because different views across a given filter, i.e. $\left(W_{t_h,h}, \ldots, W_{t_h+D_y,h}\right)^T$, can represent spikes at most of the different possible locations if the spikes are centered in the generative field. Since we include temporal alignment in the generative process we no longer see dictionary elements specializing for temporal alignment, as we did in the DSC case, and therefore the "presence" variable $\vec{s}$ is time-invariant. Similar, results were produced by (Henze et al., 2000), however, as in standard spike sorting pipelines (Rey et al., 2015) they use thresholding to identify spike events. We expect a time-invariant representation for the presence of the spike to be much better at spike sorting than features extracted by other methods. Figure 3.2 **C** shows the evolution of the standard deviation of the noise model, σ, compared to the total standard deviation of the original signal. We see that the standard deviation takes a long time to converge, especially when compared to DSC, however one must note that the posterior space in the iDSC algorithm is much larger than DSC and our approximation reduces it to a search space with comparable efficiency. Figure 3.2 **D** shows the crowdedness, i.e. $H\pi_1$, which represents the average number of generative fields that contribute to generate a datapoint. The prior is also quite slow to converge and seems to remain stable generally around 1 suggesting that there are not many overlapping neurons, i.e. neurons firing at the same time.

[3] spikes can be fairly rare events so the dictionary elements can develop to non spike features early on and the algorithm might cut out datapoints containing a spike

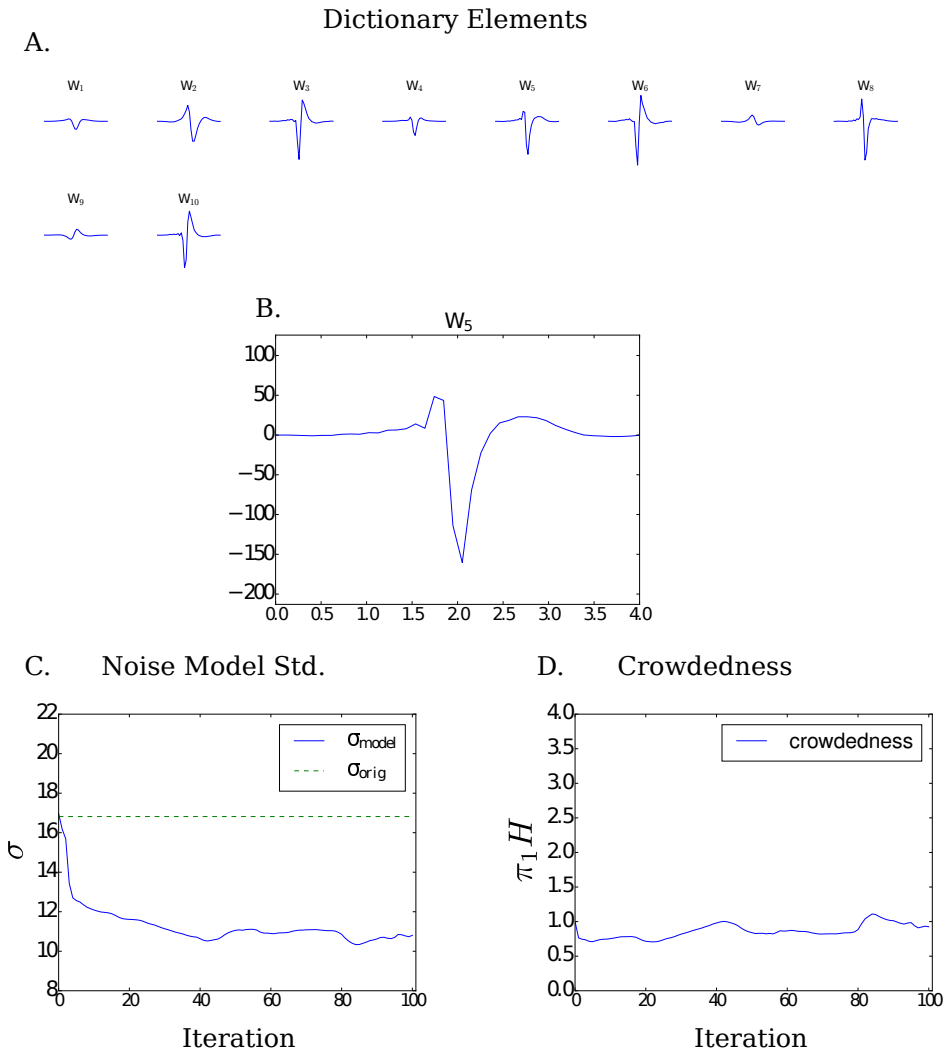

Figure 3.2.: **A.**The columns of the W matrix after we finished learning with the iDSC algorithm. Notice that the repetitive patterns along the time axis that we saw in DSC are missing. **B.** Here we show magnified copy of filter 5, the axes are consistent across filters the x-axis represent time in milliseconds and the y-axis recorded potential millivolt. **C.** The evolution of the standard deviation of the noise model (solid line) over EM iteration next to the standard deviation of the original signal (dashed line). The standard deviation decreases however it seems to take longer to converge than in the DSC case. **D.** Crowdedness over EM iterations. The crowdedness is the average number of active generative fields per data point.

As in section 2.3.3, we use the reconstruction of the signal, see Figure (3.3), to evaluate

how well we fit the data. In this case, we do not use overlapping datapoints so reconstruction is a simple matter of identifying the latent variables $\vec{t}^*$, and $\vec{s}^*$ that maximize the posterior $p\left(\vec{s}, \vec{t} | \vec{y}^{(n)}, \Theta\right)$ and use the mean of the noise model to reconstruct a datapoint, i.e. $\hat{\vec{y}}^{(n)} = f\left(W, \vec{t}^*\right) \vec{s}^*$. In Figure 3.3 **A**, you can see the reconstructed signal as a sequence of reconstructed datapoints (red line), separated by the vertical black line, plotted on top of the original extra-cellular signal (blue line) (filtered with a band-pass filter between 400Hz and 4kHz). In Figures 3.3 **B.1-10**, we see the contribution for the reconstruction of each individual generative field, i.e. we reconstructed the signal using $\hat{\vec{y}}^{(n)} = f\left(W, \vec{t}^*\right)_h s_h^*$ for each segment. There we see that the three most prominent spikes are separated cleanly by neurons 10, and 5 (also 2 contributes but with a smaller amplitude). This behavior suggests that the iDSC algorithm is capable of unsupervised separation across firing neurons that stimulate an electrode. Separating spiking neurons using a single algorithm from signal to neural coding is a significant improvement for the typical spike sorting task (e.g. compare Rey et al., 2015) A significant feature of the iDSC generative model, apart from the temporal invariance, is the ability to explain multiple overlapping neurons. For instance, in Figure (3.3) **B.4**, neuron 4 appears to be a third neuron that fires simultaneously with 5, and 10 at different times. Neurons 1, 7, and 9 have much smaller amplitude and fire more frequently. A possible assumption for the behavior low amplitude dictionary elements would be that they represent firing neurons that are fairly distant from the electrode.

A common way for spike sorting systems to identify spikes is to threshold the filtered signal at a value selected in a rather ad hoc way. A popular threshold in the literature is $5 \times \sigma_{orig}$, where σ_{orig} is the standard deviation of the filtered signal (see e.g. Quiroga et al., 2004), as exceeding that threshold would imply that this recording is a non-Gaussian event. In Figure 3.4 **B**, we show the difference between the reconstructed time series and the original recording and the result is well bellow the $5 \times \sigma$ threshold. Therefore, by standard community criteria, we do very well at extracting the non-Gaussian events (spikes). One should note at this point that some of the features we identify have a maximum amplitude below this threshold suggesting that typical spike sorting systems would probably neglect them. The iDSC algorithm does not require subjective criteria for spike identification since the latent variables $\vec{t}$ can localize the spike with very high accuracy. In Figure 3.4 **D**, for instance, we represent the activation of neuron 5 with a black vertical line at the center point of the generative field as it is aligned with the detected spike. In Figure 3.4 **C**, we see the intra-cellular recording of a neuron that is in a close distance with the extra-cellular electrode (for details see Henze et al., 2000). Note that only two spikes belong to the targeted neuron. When comparing Figures 3.4 **C** and **D** we find that neuron 5 perfectly predicts the spiking activity of the neuron of the intra-cellular recording for this part of the recording.

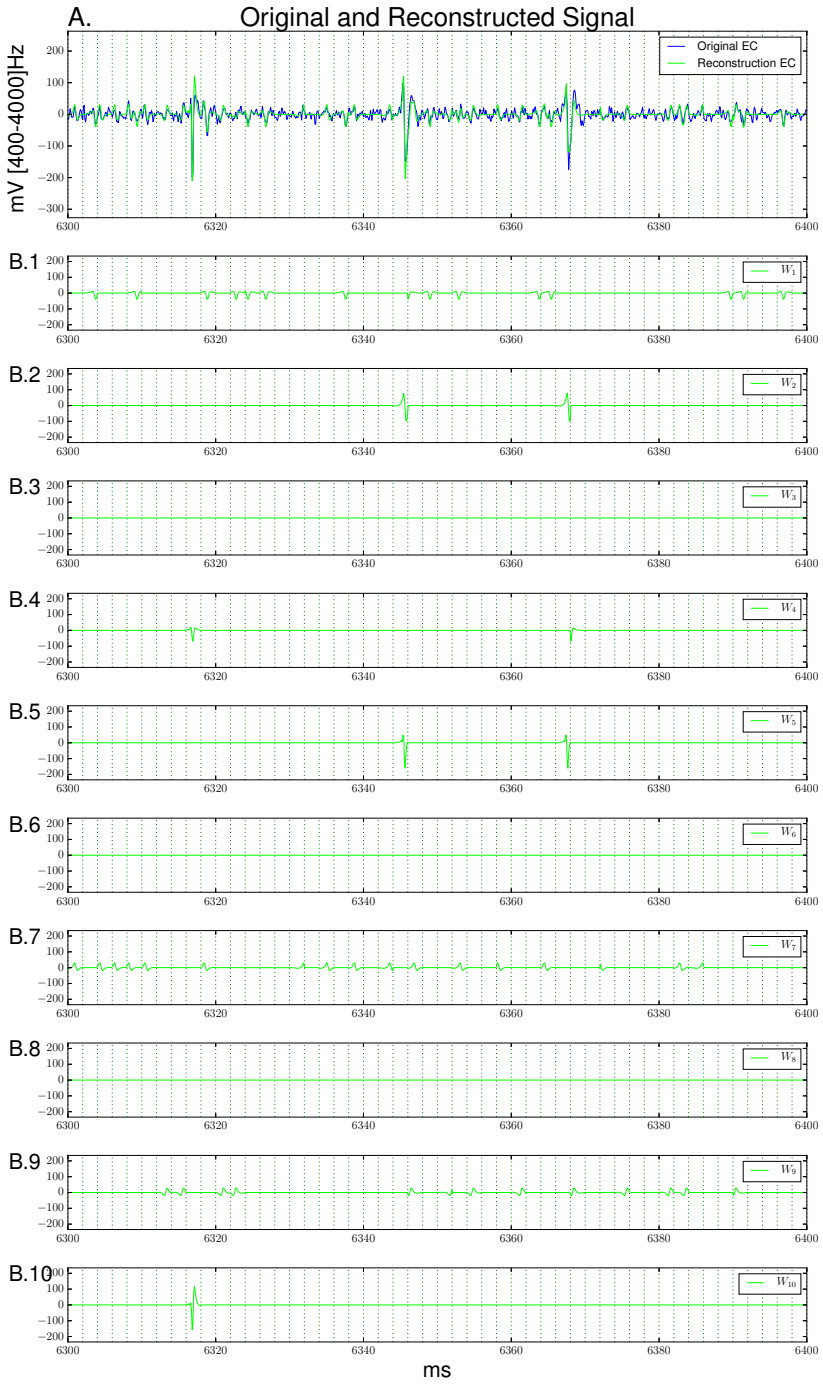

Figure 3.3.: **A** Reconstruction results of an EC recording. **B.1-10** Decomposition of the signal as in a set of dictionary elements. The reconstructed signal in Figure **A** is the sum of these plots. Notice how clearly W_5, and W_{10} separate the spikes - W_5 seems to represent the target neuron. Also notice , there high activity of neuron with of low amplitudes.

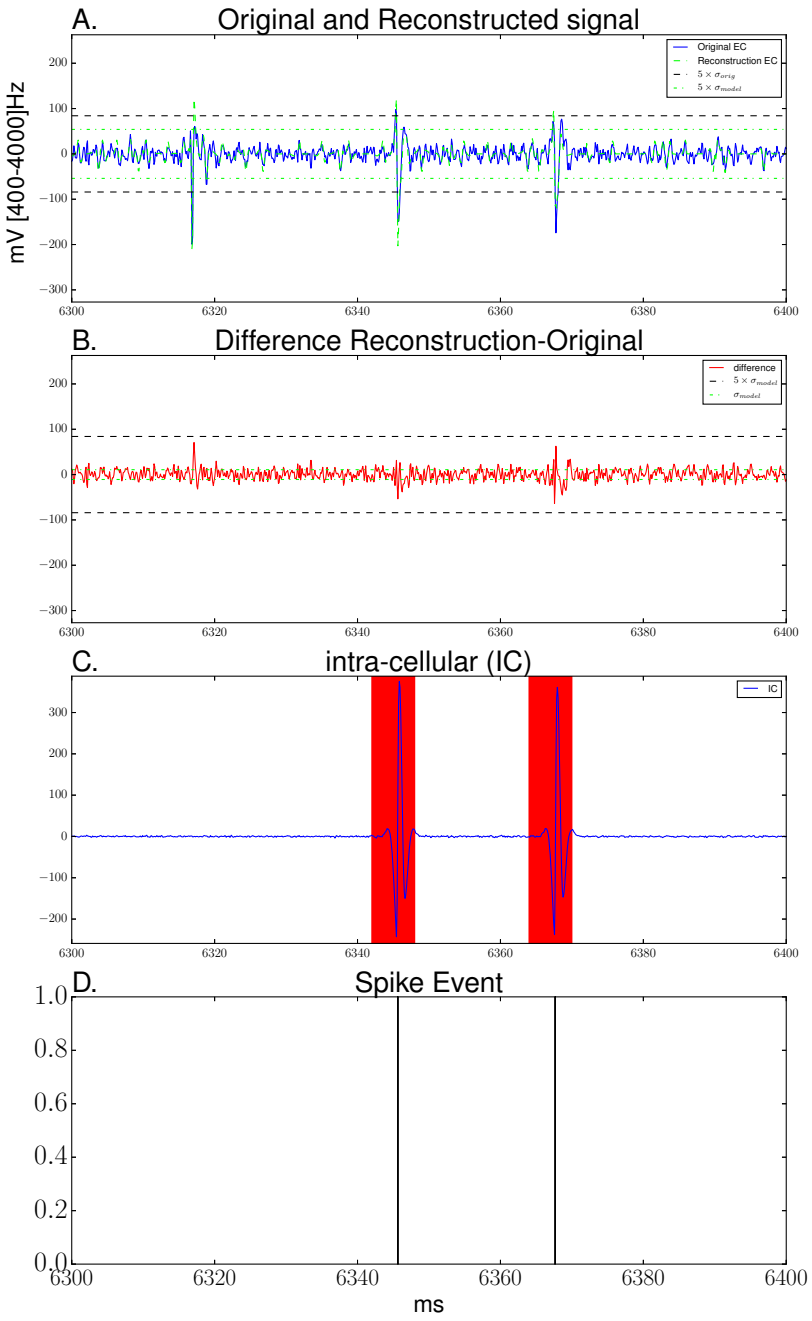

Figure 3.4.: **A** Reconstruction results of an EC recording. **B** The Difference between the reconstructed and the original signal. **C** Time-aligned IC recording - only two of the three clear spikes in **A** correspond to a spike from the targeted neuron. **D** The activation coefficient for dictionary element W_5 is highly correlated with the spiking activity of the target neuron.

3.3.2. Audio Data of Human Speech

Human speech can also be considered as a sequence of discrete events (phonemes, syllables, words etc.) that generate a continuous signal with a particular temporal structure. Identifying time-invariant structures in speech may very well provide interesting insights on speech encoding similar to the ones we saw for spiking neurons. We, therefore, applied the iDSC algorithm to the same audio data as in 2.3.4 to study the effect of an explicit time representation to audio encoding. We used the TIMIT database (Garofolo et al., 1993) to sequentially extract $N = 100\,000$ datapoints $\vec{y}^{(n)}$ in the form of $D = 60$-dimensional consecutive waveforms. We used $H = 60$ hidden random variables to describe the data under the DSC generative model with a configuration $\Phi = \{-2, -1, 0, 1, 2\}$.

For the training, the scale of the noise model σ was initialized as the average standard deviation of each observed variable. We initialize each column of the dictionary matrix $W \in \mathbb{R}^{D\times H}$ using the mean datapoint plus a mean free Gaussian noise. The prior parameter $\vec{\pi}$ was initialized such that $p\left(s_h = 0\right) = \left(H - 1\right)/H$, and $p\left(s_h = 1\right) = 1/H$.

We executed the DSC algorithm for 200 iterations following a deterministic annealing schedule described in (Ueda and Nakano, 1998; Sahani, 1999) with the annealing parameter starting at $T = 10$ and decaying it linearly to 0 by iteration 50. We also avoided datapoint cutting until iteration 20 and then proceeded to linearly decrease the datapoints to $|M|$ by iteration 80 as per the algorithm description in section 3.2. Furthermore, we add a Gaussian noise to the parameters W with a scale of 0.1 for the first 20 iterations and proceed to linearly decrease it to 0(no noise) by iteration 50. We find that adding noise to the dictionary elements enables us to learn more dictionary elements with high frequency and avoids local optima in the form of filters with 0 values. Such local optima arise in some cases when the preselection process of the iDSC algorithm excludes some dictionary elements repeatedly.

The dictionary after learning Figure 3.5 **A.** resembles wavelet-like filters of multiple frequencies. Contrary to the DSC case (Figure 2.9) were we saw localized structures with different temporal alignments most of the filters are centered. Thereby, verifying the observation that DSC dictionary elements are trying to account for temporal shifts. Having separated the task of proper alignment of the features iDSC provides a time-invariant representation that could be used in various auditory tasks, such as speech recognition. The standard deviation of the noise model in Figure 3.5 **A.**, significantly decreases after learning to less than half of the standard deviation of the original signal. However, after convergence it remains comparable to the DSC. One should note though that the DSC experiment on audio data had a larger dictionary making such a comparison less reliable for model comparison. The prior at convergence shows an average activation similar to the one from the DSC case, about 0.6 dictionary elements per datapoint. However, one should take into account that there are a lot of quiet segments in the data that could be biasing this measurement towards a lower value than necessary for speech encoding.

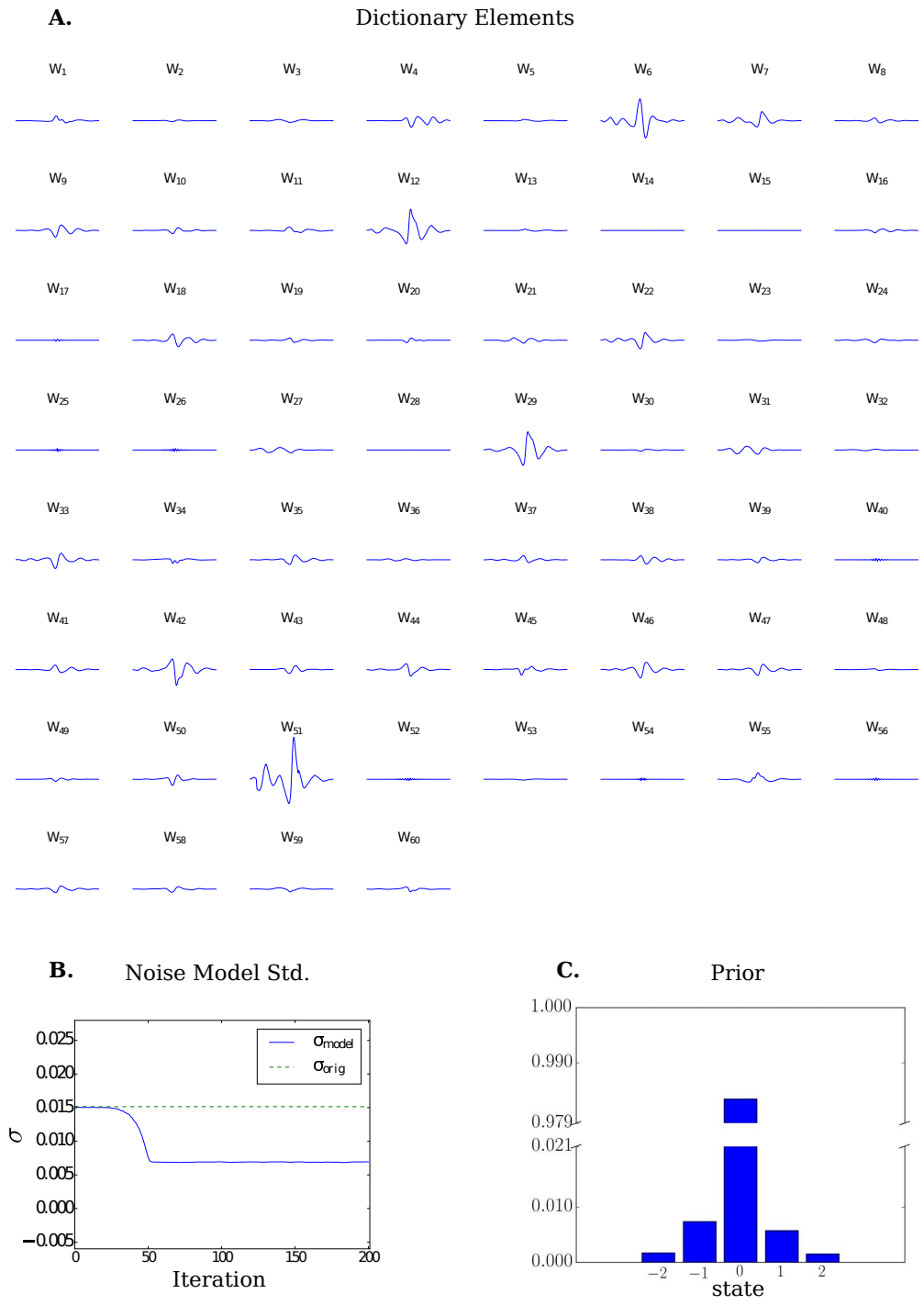

Figure 3.5.: **A.** Columns of dictionary matrix, W, after convergence of the algorithm. Notice that most of the activity is gathered at the center of the filter unlike the DSC case. **B.** The development of the standard deviation of the noise model per iteration of the iDSC algorithm. **C.** The development of the average activation of the dictionary elements per iteration.

Early on in the development of the prior we can also see very low activation values for the dictionary elements, a behavior we attribute to the annealing scheme. As we specified in our learning schedule, for the first 50 iterations we introduce some noise to the variables W. We do that to make sure that some patterns that appear frequently in the data will not dominate the dictionary during the early stages of the algorithm and leave the rest of the dictionary unused. This behavior is observed in some datasets and is considered to be a local optimum because it limits the expressiveness of the iDSC data model and results in lower values for truncated likelihood. However, adding a very high variance noise can move the filters very far away from the optimal value and in order to compensate the algorithm recovers by reducing the filters to very low values. When the filters of the iDSC model have low, approximately 0, values the effect of the generative process in describing the data is minimal so the latent variables, $\vec{s}$, with the highest posterior are the zero vector, thereby, pushing the prior to low values. This can break the algorithm since $p\left(s_h = 0\right) = 1$ means that no generative fields are ever considered by the model. Therefore, we have observed that adding a low variance Gaussian noise to the parameters can help avoid local optima but adding very high variance noise can push the algorithm towards trivial solutions.

Similarly to the neural data analysis section 2.3.3, we used the reconstruction of a time series segment to evaluate how well we were able to fit the data. In Figure 3.6 **A**, we can see the reconstruction (red line) of the original waveform (blue line). The decomposition of the reconstruction can be seen in the following to subplots **B**, **C**, **D** over 3 consecutive datapoints $n-1, n, n+1$ respectively. The vertical lines are aligned in time across the four subplots and they represent the time limits for the reconstructed patches

3.4. Discussion

We presented a sparse coding algorithm that learns how to separate the presence of a feature from its scale and temporal alignment. As in the Discrete Sparse Coding case, we were able to learn representations invariant to a set of discrete scaling effects. That allows us to learn a prior distribution that is not constrained by the typical functional constraints of most sparse coding priors. Furthermore, in this chapter we introduce a variable responsible for one dimensional translations of the data that improves the performance of the algorithm significantly when dealing with data that contain a temporal component. We were able to train the model by slightly modifying the truncated algorithm described in section 2 to take into account the additional variable (similarly to Dai et al., 2013, but using a linear SC model and one dimensional shifts). This extension maintains the properties of the algorithm in chapter 2 like learning the prior parameters and the variance of the noise model as well as a dictionary matrix for the data. However, the learned dictionaries are now better able to specialize on the profile of the spike rather than the timing.

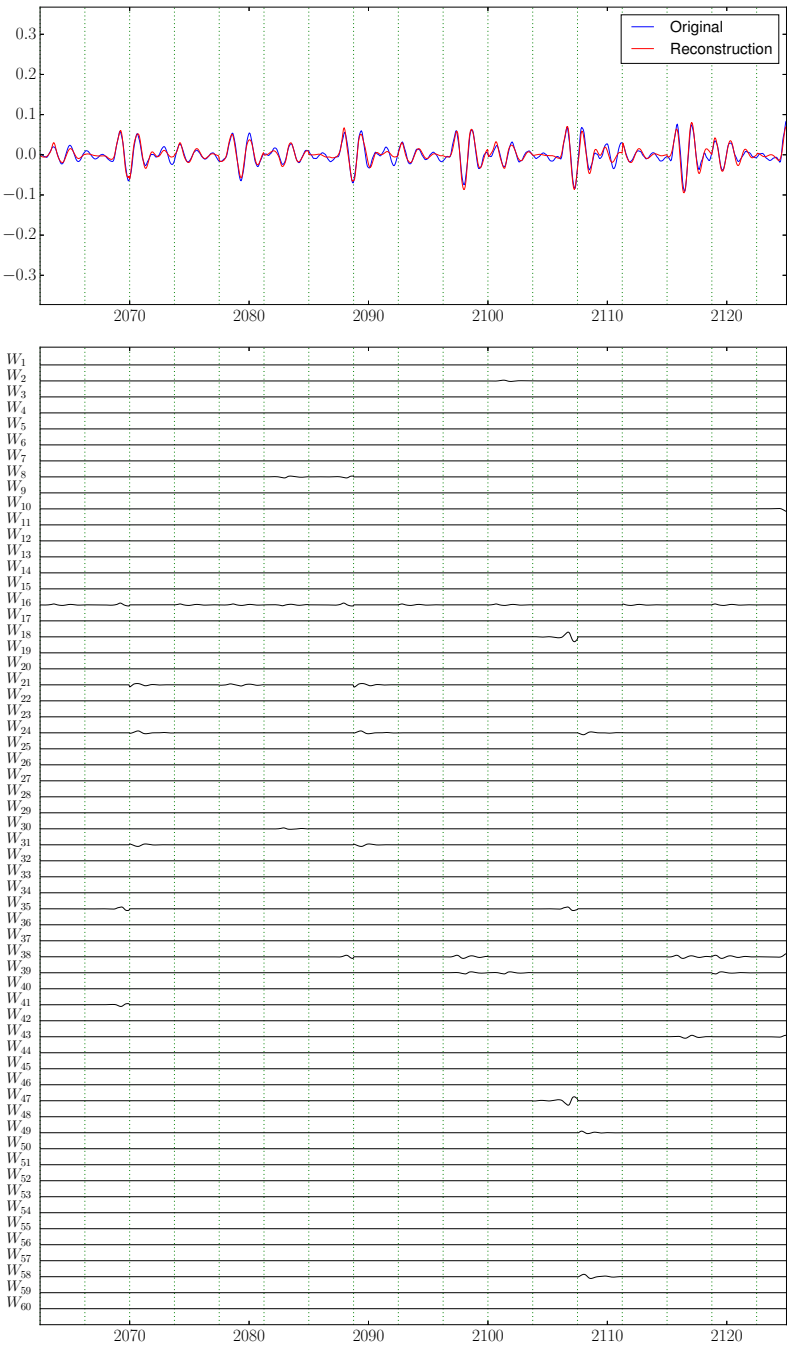

Figure 3.6.: **A** Reconstruction results of an audio waveform. **B** Dictionary elements used to reconstruct the signal. The time-axis are aligned - the plots represent the contribution of each hidden unit from W_1 to W_{60} (top to bottom)

We applied our algorithm to sequences of extra-cellular recordings from spiking neurons. The results showed that our algorithm can successfully separate spikes in a fully unsupervised manner. With the exception of passing the signal through a band-pass filter it doesn't require any preprocessing of the data that is typical to spike sorting algorithms. Typical spike sorting algorithms require multiple processing steps like, i.e. spike identification, feature extraction, and clustering (see for instance Quiroga et al., 2004; Harris et al., 2000; Lewicki, 1998, and many others). Our algorithm is effectively addressing all these steps under a single objective. Apart from the separation of any potential scientific bias that might interfere with the spike sorting process our algorithm addresses issues entirely neglected by such spike sorting pipelines such as overlapping spikes or, more importantly, the non-Gaussian description of the spike which is typically handled via an ad-hoc thresholding in a feature space. In the case where discrete latents are used it is typically based on a mixture model (Lewicki, 1998; Carlson et al., 2014) meaning that the potential for overlapping spikes is largely ignored. To the knowledge of the author the only paper that models overlapping spikes as well as temporal alignment for spikes is Pillow et al. (2013). In this work, the authors introduce a binary encoding to the waveform quite similar to the one present here, however, their algorithm uses a "binary pursuit" algorithm to identify the spiking neurons. Using this type of heuristic for the encoding limits the potential for inference in the latent space to search for the contribution of individual spikes to the objective rather than combinations of spikes (explaining away). Another drawback for that approach is the fact that it relies on an ad-hoc prior for the activation of a spike which we are able to learn through a more detailed posterior representation.

Our application on audio data extracts information about the prior and the noise that are similar to the DSC algorithm. The dictionaries are slightly more localized and they often have non-symmetric structure suggesting that they separate sounds with a distinct temporal characteristics, for instance, the beginning or the end of a phoneme. Furthermore, we are able to learn a lot of high frequency elements that were missing in the DSC case. The similarity between the learned parameters in the DSC and iDSC algorithms suggests that the audio data are not strongly dependent on temporal alignment with the exception perhaps of changes in phase of the dictionary elements. This observation could potentially be interesting in extending the work either by introducing more information in the design of the temporal alignment variables or taking into consideration models that separate phase and amplitude in the latent space (Turner and Sahani, 2011).

4. Learning Transformations with Neural Networks

4.1. Introduction

So far we have seen models that focus on learning a constrained set of transformations of the data, i.e. scaling, and temporal shifts. For the rest of this work, we will look at models that identify a more general set of transformations for the data. In this chapter we will discuss approaches based on neural networks applied on learning image transformations. Applications of such research directions can be found in any task that requires a level of abstraction higher than image content, such as action recognition and robotics (Montesano et al., 2010), as well as providing a better insight for the function of perceptual systems (Tenenbaum and Freeman, 2000; Cadieu and Olshausen, 2012).

Neural network architectures that generate invariant representations using multiplicative interactions of neurons have been proposed since the 80s (Hinton, 1981; Hinton and Lang, 1985; Anderson and Van Essen, 1987). Implementations of such architectures have appeared in image recognition systems (see for instance Wiskott et al., 1997) or for learning image correspondences (Lücke et al., 2008). For transformation learning, neural network implementations have been focusing on stacked architectures with multiplicative interactions in intermediate layers of to achieve gating in feature representation. This approach has been successful in learning transformation representations (Memisevic, 2011; Alain and Olivier, 2013), however, these architectures are typically constrained to learn a minimal number of transformation representations. Furthermore, earlier proposals for extracting intermediate level features implied using a single layer of fully connected weights (Hinton, 1981; Olshausen et al., 1993; Memisevic and Hinton, 2007) to find adaptive mappings between images or image and memory model. This approach seems to have a greater computational cost, but it offers a more intuitive encoding for the transformations and one would expect it to extract richer and more versatile transformation representations.

In Memisevic and Exarchakis (2013), we presented an algorithm that relates image sequences based on factored approaches. In that paper we introduced an experiment similar to the one that we include at the end of this chapter. The algorithm was proposed by Roland Memisevic while the implementation and testing was primarily performed by myself. The methods described in that paper considerably differ to what is described in

this chapter. Primarily, we are discussing a model that works on a pair of images and takes into account the full range products in the feature space while in Memisevic and Exarchakis (2013) we discussed a model that works on sequences of images of arbitrary length that use the squared responses of features for the encoding of the transformations.

4.2. Background on Transformation Learning

Consider two variables $\vec{y} \in \mathbb{R}^D$ and $\vec{x} \in \mathbb{R}^D$ related by a transformation $T \in \mathbb{R}^{D \times D}$. The transformation in T is usually modeled as the sum of $T = \sum_h L_k k_h$ of a set of transformations, $L_k \in \mathbb{R}^{D \times D}$, where the variable k_h is considered as a gating variable. This architecture has been used to relate variables $\vec{x}$ and $\vec{y}$ in a multitude of modeling settings. For instance, $\vec{x}$ and $\vec{y}$ take the form of visual stimulus and memory in Olshausen et al. (1993). In Tenenbaum and Freeman (2000), $\vec{k}$ and $\vec{x}$ take the form of style and content in an image.

In this work we will focus on relating two images with the same content but a varying viewpoint. Memisevic and Hinton (2007), with the same goal, present an autoencoder in which the variable $\vec{h}$ was identified as the transformation that relates $\vec{x}$ and $\vec{y}$ the most as $h_k = \sum_{ij} W_{kij} x_i y_j$. Using tied weights, the dictionary for the decoder was the same as for the encoder, i.e. the transformation matrix T between $\vec{x}$ and $\vec{y}$ was $T_{ij} = \sum_k W_{kij} h_k$. This model for transformations performed well in producing analogies of transformations of images that had global effect.

A potential issue with the autoencoder described in Memisevic and Hinton (2007) is the fact that the parameters scale cubicly with size of the observations requiring more data. To avoid this a number of approaches that factor the tensor W into matrices were introduced (see for instance Memisevic and Hinton, 2010; Memisevic, 2013; Alain and Olivier, 2013; Memisevic and Exarchakis, 2013). Typically, factored methods project the data in a feature space that serves as a distributed representation of a datapoint, i.e. each dimension carries some information about the entire datapoint. In order to encode the transformation, factored approaches use the multiplicative interactions in the feature space instead of the observed space directly. Then instead of taking all products across the variables in feature space they only take a subset of the products. This reduces the number of parameters to scale quadratically with respect to dimensionality of the input while maintaining good transformation encoding properties. It is noteworthy that factored approaches show very good results for more complex transformations in image space (Memisevic and Exarchakis, 2013). Presumably, this improvement is due to the fact that multiplicative interactions between datapoints where taken in a feature space which was a distributed representation

Although, factored representations were shown to be sufficient for non-trivial tasks we feel that it would be an oversight not to study an autoencoder that utilizes multiplicative interactions across feature representations for transformation encoding without studying

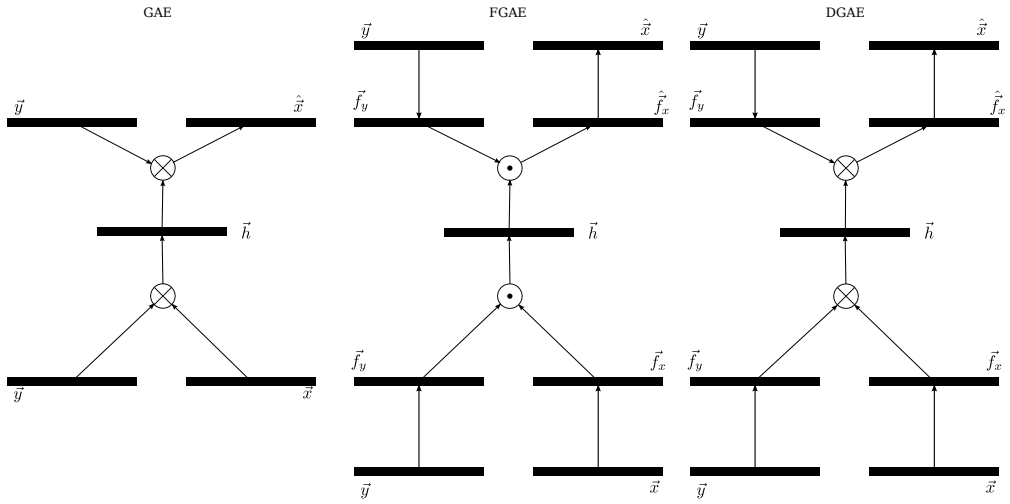

Figure 4.1.: The Gated Autoencoder (left) has all to all connections between the observations and the encoding variables. The Factored Gated Autoencoder (center) project the data in a feature space uses a subspace of the feature space products to generate the encoding. The Deep Gated Autoencoder (right) projects the input on a feature space and uses all products of the feature space to encoder the relationship.

the full range of products in the feature space. In this way we would be capable of describing a much wider range of transformations and in fact we should be able to encode transformations in very heterogeneous spaces like image-action or image-word.

4.3. Deep Gated Autoencoder

Let the dataset consist of pairs of observations $\vec{x} \in \mathbb{R}^{D_x}$ and $\vec{y} \in \mathbb{R}^{D_y}$ related by some transformation. We seek a representation of the data in which each dimension carries information about the entire image. To achieve that we use an affine transformation on the data:

$$\vec{f^x} = U\vec{x} + \vec{b}_x \tag{4.1}$$

$$\vec{f^y} = V\vec{y} + \vec{b}_y \tag{4.2}$$

where $U \in \mathbb{R}^{K_x \times D_x}$, and $V \in \mathbb{R}^{K_y \times D_y}$ are the filtering matrices that we use to achieve an appropriate representation and $\vec{b}_x \in \mathbb{R}^{K_x}$, and $\vec{b}_y \in \mathbb{R}^{K_y}$ are vectors representing an additive bias. We combine $\vec{f}_x$, and $\vec{f}_y$ using a bilinear tensor $W \in \mathbb{R}^{H \times K_x \times K_y}$ to get a

transformation encoding, $\vec{k}$ as:

$$k_h = g\left(\sum_{ij} W_{hij} f_i^x f_j^y\right) \tag{4.3}$$

where g is a sigmoid function. Notice that if we think of W as several matrices stacked together along the dimension indexed by h then equation 4.3 will result in a higher value for the matrix that better connects $\vec{f^x}$ and $\vec{f^y}$.

The decoding stage of our algorithm uses the transformation representation $\vec{k}$ and one of the features representations $\vec{f^x}$, or $\vec{f^y}$ to reconstruct the other. Using the same, tied weights for both encoding and decoding that is:

$$\hat{f}_i^x = \sum_{hj} W_{hij} k_h f_j^y \tag{4.4}$$

$$\hat{f}_j^y = \sum_{hu} W_{hij} k_h f_i^x \tag{4.5}$$

Once we have a feature representation, we reconstruct the data using the same filters as in the encoding phase:

$$\hat{\vec{x}} = U^T \hat{\vec{f}}^x + \vec{c}_x \tag{4.6}$$

$$\hat{\vec{y}} = V^T \hat{\vec{f}}^y + \vec{c}_y \tag{4.7}$$

Our learning algorithm seeks the parameters $\Theta = \{W, U, V, b_x, b_y, c_x, c_y\}$ that optimize the objective of our algorithm. In a typical autoencoder neural network the objective is the reconstruction error, i.e.:

$$C_x = \|\vec{x} - \hat{\vec{x}}\|_2^2 \tag{4.8}$$

or

$$C_y = \|\vec{y} - \hat{\vec{y}}\|_2^2 \tag{4.9}$$

depending on whether we want to learn transformations from $\vec{y}$ to $\vec{x}$, or $\vec{y}$ to $\vec{x}$ respectively. In some cases, we might want to know how to learn bidirectional transformations in which case a cost like

$$C = C_x + C_y \tag{4.10}$$

might be more appropriate. One must be careful however because in the case of tied weights, i.e. using the same parameters for encoding and decoding, the learned transformations are urged to be orthogonal and depending on the application that may not always be desirable. We call the model described in equations 4.1 to 4.10 *Deep Gated Autoencoder* (DGAE)

In section 1.2 we discussed the similarities between autoencoders and probabilistic

models. In the case of the Deep Gated Autoencoder the negative of the cost function is proportional to the logarithm of the conditional probabilities of the variables

$$\log p\left(\vec{x}|\vec{y},\Theta\right) \propto -C_x \tag{4.11}$$

$$\log p\left(\vec{y}|\vec{x},\Theta\right) \propto -C_y \tag{4.12}$$

assuming we use a Gaussian noise model with an identity covariance matrix. Therefore, the cost function in Equation 4.10 corresponds to the product of the two conditonal distributions.

Training the model, or maximizing the likelihood as described in section 1.2, can be achieved through stochastic gradient descent in the space spanned by the parameters Θ (backpropagation).

4.4. Experiments

4.4.1. Image analogies

We use the NORB dataset (see LeCun et al., 2004) to create pairs of images of an object in different pose by varying the elevation and the azimuth of the camera while maintaining the same lightning conditions (a similar dataset was introduced in Memisevic and Exarchakis, 2013). The elevations vary by changing the camera viewing angle by either $-10, -5, 0, 5, 10$ degrees, and the azimuth changes by either $-20, 0, 20$ degrees. That creates a total of 15 different transformations in a 3 dimensional space. In this way we extract $N = 304200$ image pairs of dimensionality 96×96, i.e. $D_x = D_y = 9216$. The number of the observed dimensions is too large for our algorithm to work efficiently. We use Principle Component Analysis (PCA) (introduced in Hotelling, 1933), (also see Hyvarinen et al., 2009; Bishop, 2006) to reduce the dimensionality of the images while maintaining 99% of the variance of the data. This form of preprocessing reduced the dimensions to $D_x = D_y = 666$. For this experiment we use a feature space with the same dimensionality as the input $K_x = K_y = 666$. Even though we know that we only have 15 different transformation we can not be certain that we will achieve perfect separation of the used transformation. In an attempt to achieve a better separation of distinct transformation we use encodings with more dimensions than the transformations in our data, i.e. $H = 100$.

We train the model on the entire dataset using stochastic gradient descent for 1000 iterations over the full dataset using randomly selected subsets of the data (batches) of size 100. The update rules have been modified to include a momentum term as in Bengio et al. (2013) to enhance gradient values when we are close to the optimum and the gradient takes small values. Furthermore, the update rules in Bengio et al. (2013) also include a manipulation of the gradient similar to the one in Nesterov et al. (2007) in which they

attempt to estimate the value of the gradient at the point of the updated parameters and correct accordingly. The value of the momentum is set to $m = 0.9$ and the learning rate is set to $\epsilon = 10^{-5}$. We also implement a dropout regularization Hinton et al. (2012) scheme to find more robust encoding of the transformations. Dropout amounts to multiplying the transformation encoding $\vec{k}$ with samples from a Bernoulli distribution. Setting some of the dimensions of the encoding to zero urges the tensor W to use the sum of multiple matrices W_h to encode a transformation. This has no significant effect on the results presented here but it has been reported (for instance here Baldi and Sadowski, 2013) that it increases separability in the features space and typically the performance in classification tasks.

In order to generate analogies of image transformations we use two images to encode a transformation using equations 4.1 to 4.3 that produce a representation $\vec{k}$ for that transformation. We use $\vec{k}$ and a feature representation, $\vec{f}^{\,x}$ of a new (target) image and generate a reconstruction $\hat{\vec{y}}$ of the target image transformed in the same manner as the two original images that we used to get $\vec{k}$. If we are successful then the reconstruction will not carry any content information about the original images but only of the target image meaning that $\vec{k}$ is an encoding of the transformation between images and not the content.

In figure 4.2, we demonstrate the ability of our algorithm to identify transformations by producing analogies of the same transformations to a different object (target). We use 6 examples of typical transformations that have been selected to better demonstrate the potential of the algorithm. In each of the sub-figures 4.2 **A**-**F** we present a pair of images of the same object (the two top images) that define a transformation $\vec{k}$. The values in each dimension of $\vec{k}$ are plotted in the bottom bar-plot. Having the transformation encoding $\vec{k}$ we take an image of a new object (bottom left image) and apply the transformation. This process generates the image seen in the bottom right which is now transformed in manner analogous to the top two images. From the ensemble of image pairs we can observe that the algorithm is able to separate image transformations from image content successfully in many cases.

In figure 4.3, we present some failed examples of the transformations. In figure 4.3 **A** we have an example where image content of the top images (a car) seems to be present in the reconstruction. This behavior suggests that separability is not perfect and on occasion the transformation can focus on the object in a variety of orientations. In figure 4.3 **B** we see an example were image structure seems to be lost almost entirely and the result resembles a low frequency noise.

In figure 4.4, we present examples of image analogies were the transformation was particularly good. In the case of figure 4.4 **A**, we see that the target object is the same as in the seed image. It also has the same pose and lightning conditions and in that case it seems that the transformation is inferred almost perfectly. In the case of figure 4.4 **B** we see an image pair that is not transformed, i.e. it defines the identity transformation. It is common for our algorithm to infer the identity transformation correctly in most cases.

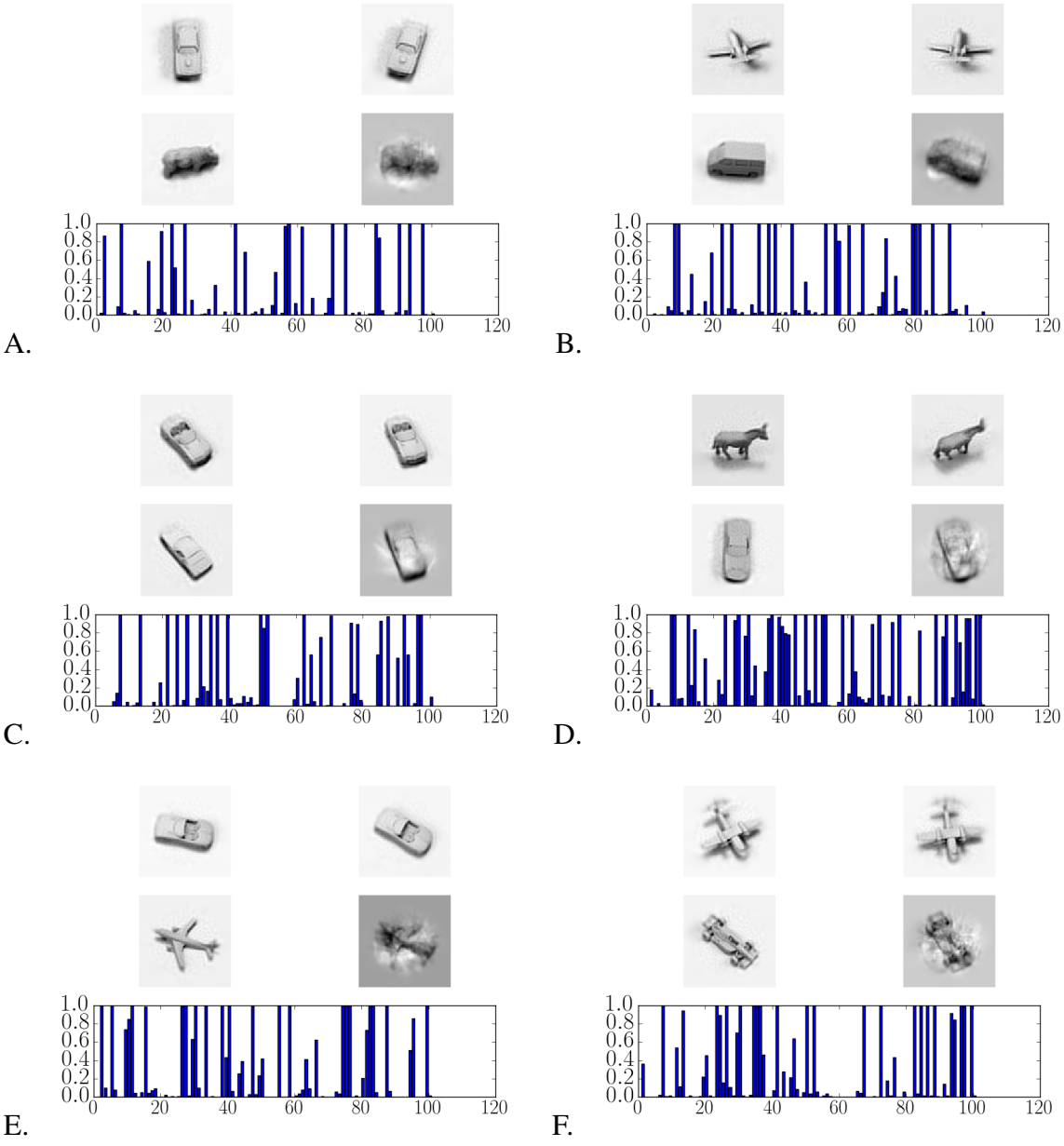

Figure 4.2.: Examples of image analogies. Each of the figures **A**-**B** shows a pair of images (top two images) that define an image transformation. The encoding of the image transformation is plotted in the bar plot at the bottom. The bottom left image is then used as a target for the transformation and produces the bottom right image. These examples show that the algorithm can extract an encoding of the transformation that is invariant of the form content in the image

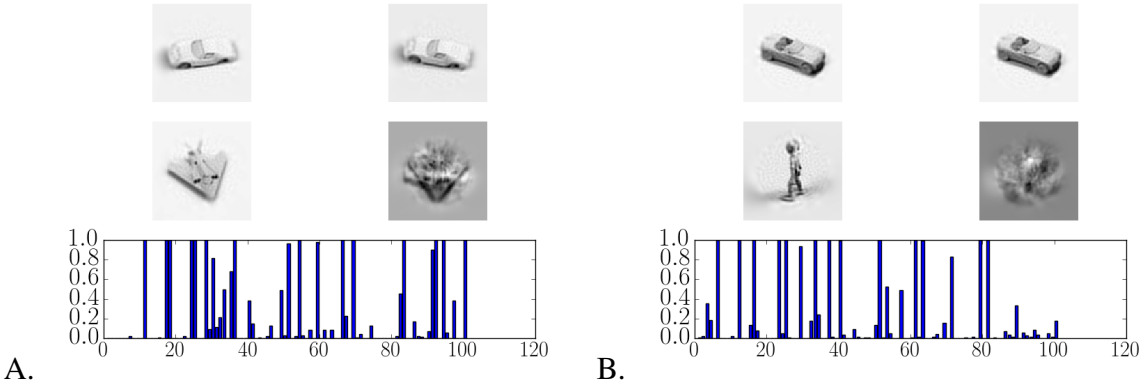

Figure 4.3.: Failed examples of image analogies: **A** shows an example were separating the transformation from the content of the image was not successful and the you can see features of the car in the top two images in the reconstructed image. **B** shows an example were separating the transformation completely distorts the image

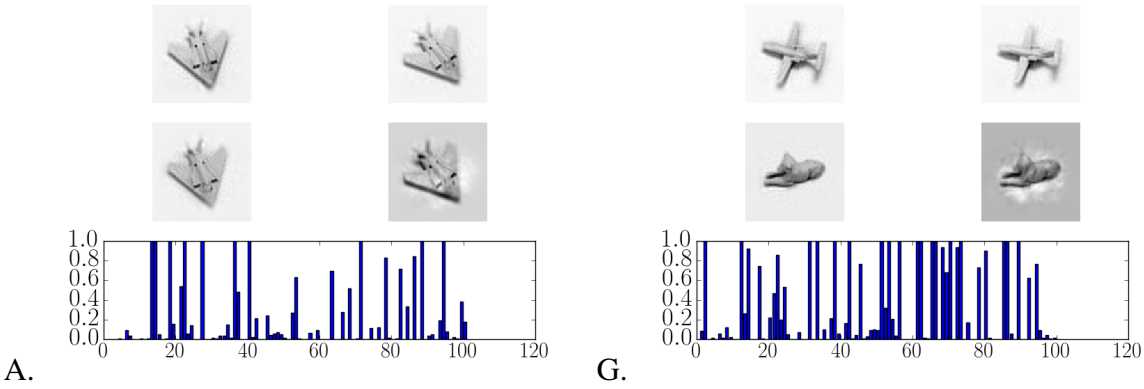

Figure 4.4.: Analogy examples that were unusually good. **A** shows an example in which the target was the same object in the same pose and lighting condition as the corresponding image used to define the transformation. Under these conditions we do very well at reconstructing the image. **B** shows an example in which the transformation was the identity transformation. Empirical results show that the identity transformation is easily learned.

4.5. Discussion

In this chapter we presented a new algorithm that learns transformations based on neural networks with gating connections. Earlier work (Memisevic and Hinton, 2007; Memisevic, 2013) has shown that learning image transformations is possible but it also suggests that factored methods are better capable of extracting local and more complex transformations than the standard gated Autoencoder. This is probably due to the fact that the factored gated autoencoder takes into account multiplicative interactions in feature space rather than the observed variables directly. Here we define an autoencoder that encodes transformation in a manner similar to GAE but instead of working with observations directly it seeks an alternative representation of the input first. To evaluate the potential of our algorithm to extract image transformations we applied it on a task of producing analogies of image transformations similar to the one in Memisevic and Exarchakis (2013). The results show that we are able to separate image transformations from the image content and in fact the transformation we learn appear to be more complex than the ones learned with the standard gated autoencoder (Memisevic and Hinton, 2007). Although, our method has a higher representational capacity than the factored gated autoencoder we have not been able to show considerable performance improvement, however to examine the full range of gating connections we need to scale the number of parameters cubicly with respect to the input while the parameters of the factored gated autoencoder only scale quadratically. That requires us to use more datapoints for training and longer training times which might be a reason for the competitive results of the factored method.

5. Bilinear Dynamical Systems

5.1. Introduction

The Deep Gated Autoencoder we described in the last chapter is efficient in high dimensions and has the demonstrated ability to extract a variety of complex transformations. However, as we have seen in its probabilistic formulation the encoding of these transformations is deterministically defined from the data. That constrains the study of the representation to happen on an example by example basis. Having a probabilistic model that defines an appropriate prior for the latent variables would enable us to sample from the prior and draw a more clear picture on the function of the algorithm. Added to that, we have the usual benefits of a probabilistic representation of a variable, e.g. using the posterior of the encoding given we have a model of the uncertainty of our representation. Another, limitation of DGAE is that it is defined to operate on pairs of datapoints related by certain transformation. Typically, a task that requires learning transformations from data would involve a sequence of more than a pair of datapoints.

In this chapter, we define a probabilistic model that can be fitted to sequential data. We intend to achieve an algorithm that separates static from dynamic content in a sequence of observations. We do that using a Bilinear State Space Model (BSSM) in which the state transitions change over time. This model adds to the work of the previous chapter in two ways, we have a definition of a prior on the variables that encode transformations and we are able to apply it iteratively on a sequence of observations thus extending the model to work on more than pairs of images.

Recurrent neural networks (Michalski et al., 2014) and other methods (Memisevic and Exarchakis, 2013), extending the gated autoencoder networks (Memisevic, 2013) discussed earlier, have the potential to work on arbitrary length sequences, however, these models maintain a deterministic encoding of the transformations. The work presented in this chapter bears more similarity to typical state space models (see for instance Shumway and Stoffer, 2010; Bar-Shalom and Li, 1993). Similar work has been attempted with the use of graphical models (for instance Ghahramani and Hinton, 2000; Bar-Shalom and Li, 1993), and other non-linear dynamical systems (Luttinen et al., 2014). Our work is most similar to the one in (Luttinen et al., 2014) with the exception that we assume independence in the "switching" variables and that they use a variational Bayes approximation for the parameter updates. The main difference with (Ghahramani and Hinton, 2000) is that we use "switch" variables only for the transformations and not the generative filters. The

model presented in this chapter is therefore a simplified version of earlier work. However, we support our modeling choices on the fact that the afore mentioned algorithms introduced complexity that is potentially unnecessary for the tasks that we address. We will refer to these models as hybrid state space networks since they typically include continuous variables to represent the state and discrete variables for switching.

5.2. Mathematical Description

Let the data $\mathbf{Y}$ be a set of N independent sequences of observations $\vec{y}_t^{(n)} \in \mathbb{R}^D$, with $n = 1, \ldots, n$, and $t = 1, \ldots, \tau$. We propose a dynamical system that describes the data as:

$$\vec{y}_t = W\vec{z}_t + u_t \tag{5.1}$$

$$\vec{z}_t = f\left(\vec{z}_{t-1}, \vec{k}\right) + v_t \tag{5.2}$$

where f is a bilinear map, and $\vec{k} \in \{0,1\}^C$, with $\left|\vec{k}\right| = 1$, is a "switching" variable that identifies the linear transformation from state $\vec{z}_{t-1}$ to $\vec{z}_t$. Therefore, we may sometimes write $f\left(\vec{z}_t, \vec{k}\right) = \sum_{ic} A z_t^i k_t^c$, or $f\left(\vec{z}_t, \vec{k}\right) = A^{(k)}\vec{z}_t$ where $A^{(k)} \in \mathbb{R}^{H\times H}$. u_t and v_t are stochastic variables representing mean-free Gaussian noise with covariance matrices Σ, and Γ respectively. The recursive rule requires a prior at time $t = 1$ which we set to be a Gaussian distribution with a mean $\vec{\mu}$ and a covariance matrix R.

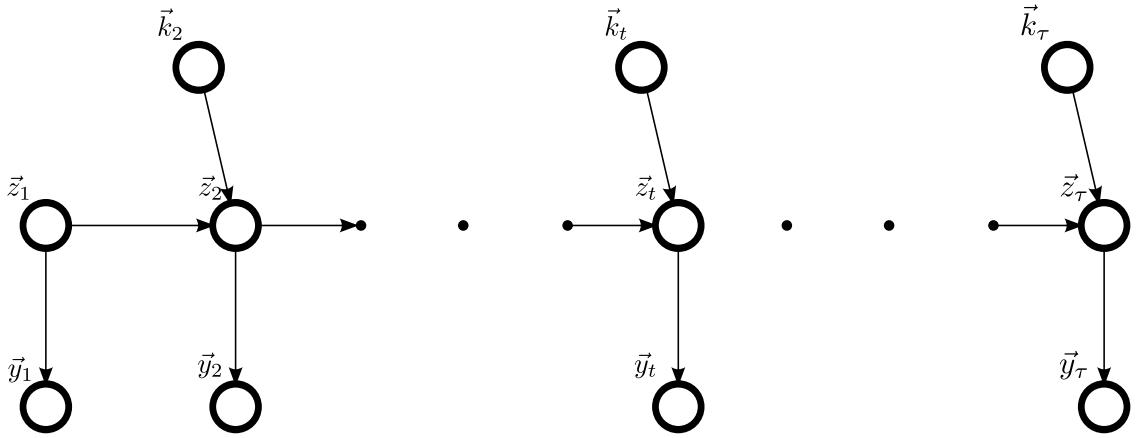

Figure 5.1.: Dependency Graph of the Generative Model.

Here we propose an algorithm that efficiently learns maximum likelihood parameters $\Theta^* = \{W^*, \Sigma^*, A^*, \Gamma^*, \vec{\pi}^*, \mu^*, R^*\}$ for the generative model. The likelihood of the model

can be written in terms of the latent variables as:

$$p\left(\mathbf{Y}\right)=\prod_{n}p\left(\vec{y}^{(n)}\right)=\prod_{n}\int_{\vec{z}}\sum_{\vec{k}}p\left(\vec{y}^{(n)},\vec{z},\vec{k}\right)d\vec{z} \tag{5.3}$$

since the data are independent over n all computations in the log-likelihood over n are reduced to summations. We will show the results for $N=1$ sequences but scaling to anything higher would be trivial since the sequences are assumed to be independent. One thing to note about our model is that since we assume that state changes from $\vec{z}_{t-1}$ to $\vec{z}_t$ can happen in C different ways with a certain probability then at state $\vec{z}_t$ is modeled as a mixture of Gaussians $\sum_c p\left(\vec{z}_t|\vec{z}_{t-1},\vec{k}\Theta\right)p\left(\vec{k}|\Theta\right)$. Following the recursion at time τ we would have a mixture with C^τ components which is easily intractable. For that reason we assume that at each time point the mixture of Gaussians merges to a single Gaussian. To find the maximum likelihood parameters we will use an EM algorithm modified for sequential data (Baum et al., 1970). The EM algorithm requires us to define an expectation value of the logarithm of the joint distribution of latent and observed variables:

$$\begin{aligned}p\left(\mathbf{y},\mathbf{z},\mathbf{k}\right)=&p\left(\mu\right)\prod_{t=2}^{\tau}p\left(\vec{z}_t|\vec{z}_{t-1},\vec{k},\Theta\right)p\left(\vec{k}|\Theta\right)\prod_{t=1}^{\tau}p\left(\vec{y}_t|\vec{z}_t,\Theta\right)\Leftrightarrow\\ \log p\left(\mathbf{y},\mathbf{z},\mathbf{k}\right)=&\log p\left(\mu\right)\sum_{t=2}^{\tau}\log p\left(\vec{z}_t|\vec{z}_{t-1},\vec{k},\Theta\right)p\left(\vec{k}|\Theta\right)\sum_{t=1}^{\tau}\log p\left(\vec{y}_t|\vec{z}_t,\Theta\right)\end{aligned} \tag{5.4}$$

where we take advantage of the Markov property assumed by the generative model. With this formulation of the joint the part of the free energy depending on Θ forms as follows:

$$\begin{aligned}\mathcal{Q}\left(\Theta\right)=\sum_{n}\Bigg(&-\frac{1}{2}\sum_{t}\int_{\vec{z}_t}q\left(\vec{z}_t\right)\left(\log\left|2\pi\Sigma\right|+\left(\vec{y}_t-W\vec{z}_t\right)\Sigma^{-1}\left(\vec{y}_t-W\vec{z}_t\right)^T\right)d\vec{z}_t\\&-\frac{1}{2}\sum_{t=2}\int_{\vec{z}_{t:t-1}}\sum_{k}q\left(\vec{z}_{t:t-1},k\right)\left(\log\left|2\pi\Gamma\right|+\left(\vec{z}_t-A^{(k)}\vec{z}_{t-1}\right)\Gamma^{-1}\left(\vec{z}_t-A^{(k)}\vec{z}_{t-1}\right)^T\right)d\vec{z}_{t:t-1}\\&+\sum_{t=2}\sum_{k}q\left(k\right)\sum_{c}k_c\log\left(\pi_c\right)\\&-\frac{1}{2}\int_{\vec{\mu}}q\left(\vec{z}_1\right)\left(\log\left|2\pi V_1\right|+\left(\vec{y}_t-W\vec{z}_1^{\tau}\right)V_1^{-1}\left(\vec{y}_t-W\vec{z}_1^{\tau}\right)^T\right)d\vec{\mu}\Bigg)\end{aligned} \tag{5.5}$$

where $q\left(x\right)=p(x|y_{1:\tau},\Theta^{\text{old}})$ is the posterior probability with respect to the old parameters Θ^{old}. Taking into account the linearity property of expectation values, the necessary

expectation values can be are identified and denoted as:

$$E_q\left[\vec{z}_t\right] = \hat{z}_t, \tag{5.6}$$

$$E_q\left[\vec{z}_t\vec{z}_t^T\right] = P_t, \tag{5.7}$$

$$E_q\left[\vec{z}_t\vec{z}_{t-1}^T|\vec{k}_t\right] = P_{t,t-1|k}, \tag{5.8}$$

$$E_q\left[\vec{k}_t\right] = \hat{k}_t. \tag{5.9}$$

equation 5.8 is the cross-covariance between two consecutive latent variables $\vec{z}$ and it is necessary for learning the parameters A, and Γ which are relevant to the system dynamics. We will not infer the cross-covariance with the algorithm presented here since it introduces a significant level of complexity to our algorithm and it is already possible to produce substantial functionality without it. Subsequently we will not be able to propose parameter updates for the system dynamics.

Equations 5.6 to 5.9 form the sufficient statistics for the algorithm and identifying them is a process analogous to the E-step of the EM algorithm We can infer the sufficient statistics by modifying the RTS smoother (Rauch, 1963; Ghahramani and Hinton, 1996) to include multiple transformations for the dynamics of the generative process. As in the linear case, we separate inference in two stages of recursion the forward pass, filtering, and the backward pass, smoothing.

In the filtering stage we identify the probabilities $p\left(\vec{z}_t|\vec{y}_{1:t-1}\right)$, $p\left(\vec{z}_t|\vec{y}_{1:t}\right)$, and $p\left(\vec{k}_t|\vec{y}_{1:t}\right)$ which we specify by their moments $\left(\hat{z}_t^{t-1}, V_t^{t-1}\right)$, $\left(\hat{z}_t^t, V_t^t\right)$, and $\left(\pi_c^t\right)$ respectively. The parameters of these distributions can be shown to emerge as follows,

$$\hat{z}_{t|k}^{t-1} = A_{|k}\hat{z}_{t-1}^{t-1} \tag{5.10}$$

$$V_{t|k}^{t-1} = A_{|k}V_{t-1}^{t-1}A_{|k}^T + \Gamma \tag{5.11}$$

$$\pi_c^t = \frac{p\left(y_t|y_{1:t-1}, k_c = 1\right)\pi_c^{t-1}}{\sum_{c'} p\left(y_t|y_{1:t-1}, k_{c'} = 1\right)\pi_{c'}^{t-1}} \tag{5.12}$$

$$\hat{z}_t^{t-1} = \sum_c \pi_c^t \hat{z}_{t|k_c=1}^{t-1} \tag{5.13}$$

$$V_t^{t-1} = \sum_c \pi_c^t \left(V_{t|k_c=1}^{t-1} + \left(\hat{z}_{t|k_c=1}^{t-1} - \hat{z}_t^{t-1}\right)\left(\hat{z}_{t|k_c=1}^{t-1} - \hat{z}_t^{t-1}\right)^T\right) \tag{5.14}$$

$$K_t = V_t^{t-1}W^T\left(\Sigma + WV_t^{t-1}W^T\right)^{-1} \tag{5.15}$$

$$\hat{z}_t^t = \hat{z}_t^{t-1} + K_t\left(y_t - W\hat{z}_t^{t-1}\right) \tag{5.16}$$

$$V_t^t = \left(I - K_tW\right)V_t^{t-1} \tag{5.17}$$

where $A_{|k}$ is the matrix that results from multiplying $\vec{k}$ with the tensor A, i.e. $\sum_c A_{cij} k_c$. The above equations approximate the filtering process by trying to get some information on the appropriate transformation before we make a step towards it. In particular, equations 5.10 to 5.11 are "peeking" ahead to find which transformation is more likely to be taken at the next step. Once the transformation is found we update the probability of the switching variable in equation 5.12 and we use that to merge our predictions to a single Gaussian function in equations 5.13 and 5.14 (see Lauritzen, 1996; Bar-Shalom and Li, 1993). We correct the prediction as in the typical Kalman Filter updates in equations 5.15 to 5.17.

In the smoothing stage we look for the probabilities $p\left(\vec{z}_t|\vec{y}_{1:\tau}\right)$, and $p\left(\vec{k}_t|\vec{y}_{1:\tau}\right)$ which we specify by their moments $(\hat{z}_t^\tau, V_t^\tau)$ and $\left(\pi_{c|\tau}\right)$, respectively.

$$\hat{z}_{t-1|k}^\tau = \left(\hat{z}_{t-1}^{t-1} + J_{t-1|k}\left(\hat{z}_t^\tau - \hat{z}_{t|k}^{t-1}\right)\right) \tag{5.18}$$

$$\hat{z}_{t-1}^\tau = \sum_c \hat{z}_{t-1|k_c=1}^\tau \pi_c^\tau \tag{5.19}$$

$$V_{t-1|k}^\tau = V_{t-1}^{t-1} + J_{t-1|k}\left(V_t^\tau - V_t^{t-1}\right) J_{t-1|k}^T \tag{5.20}$$

$$V_{t-1}^\tau = \sum_c \pi_c^\tau \left(V_{t-1|k_c=1}^\tau + \left(\hat{z}_{t-1|k_c=1}^\tau - \hat{z}_{t-1}^\tau\right)\left(\hat{z}_{t-1|k_c=1}^\tau - \hat{z}_{t-1}^\tau\right)^T\right) \tag{5.21}$$

$$\text{where } J_{t-1|k} = V_{t-1}^{t-1} A_{|k}^T \left(V_{t|k}^{t-1}\right)^{-1} \tag{5.22}$$

$$\pi_c^\tau = \frac{p\left(y_t^\tau|\hat{y}_{1:\tau}, k_c = 1\right)\pi_c^{\tau-1}}{\sum_{c'} p\left(y_t^\tau|\hat{y}_{1:\tau}, k_{c'} = 1\right)\pi_{c'}^{\tau-1}} \tag{5.23}$$

The backward recursions are initialized with $\hat{z}_\tau^\tau$, and V_τ^τ respectively as obtained from the filtering results while the prior update π_c^t remains the same for the backward as in the final step of the filtering.

Parameter Updates We can find the parameter updates by taking the gradient of equation 5.5 with respect to each parameter and setting it to zero. The parameter updates require the sufficient statistics in the E-step which can be expressed in terms of the smoother equations as:

$$E_q\left[\vec{z}_t\right] = \hat{z}_t^\tau, \tag{5.24}$$

$$E_q\left[\vec{z}_t\vec{z}_t^T\right] = P_t = V_t^\tau + z_t^\tau z_t^{\tau T}, \tag{5.25}$$

$$E_q\left[\vec{k}_t\right] = \hat{k}_t = \pi_t^\tau. \tag{5.26}$$

This process gives closed form update rules for the parameters that optimize the objective.

$$\hat{z}_1^{\text{new}} = \hat{z}_1^{\tau(n)} \tag{5.27}$$

$$V_1^{\text{new}} = V_1^{\tau} + \left|\hat{z}_1^{\tau} - \hat{\bar{z}}_1^{\tau}\right| \left|\hat{z}_1^{\tau} - \hat{\bar{z}}_1^{\tau}\right|^T \tag{5.28}$$

$$\pi_c^{\text{new}} = \frac{\pi_c^{\tau}}{\sum_{c'} \pi_{c'}^{\tau}} \tag{5.29}$$

$$W^{\text{new}} = \left(\sum_{t=1}^{\tau} y_t \left(\hat{z}_t^{\tau}\right)^T\right) \left(\sum_{t=1}^{\tau} P_t\right)^{-1} \tag{5.30}$$

$$\Sigma^{\text{new}} = \frac{1}{T} \sum_t \left(y_t y_t^T - W \hat{z}_t^{\tau} y_t^T - y_t \left(\hat{z}_t^{\tau}\right)^T W^T + P_t\right) \tag{5.31}$$

For arbitrary N, we have to replace the sufficient statistics with their average over the datapoints. Our posterior approximation at this point is not able to correctly learn the parameter updates for the state transitions A, and Γ. This is potentially due to the fact that we merge the predictions of future states to a single Gaussian and that alters the direction of the transformations when described by the sufficient statistics. An alternative method that could potentially solve this issue is described in Bar-Shalom and Li (1993) where the authors present an algorithm (GPB2) for a hybrid state space model that maintains the projections across different filters for pairs for two consecutive time-points.

5.3. Numerical Experiments

In this section we present some results that verify the algorithms capacity to identify transformations in data. We do that by generating sequences of datapoints using the BDS model that has varying switching variables $\vec{k}$ over time. We show that we can infer the model state of these sequences fairly accurately and that we are able to identify when the dynamics change in a sequence.

5.3.1. Inference on artificially generated sequences

We generate $N = 200$ sequences of $T = 50$ observations each by drawing N samples from the state prior $p(\vec{z}; \vec{\mu}, R)$ and $N(T-1)$ samples from the prior of the transformation and use the BDS generative model to generate artificial observations. We evaluate our representation of the latent space by attempting to identify the latent variables that generated each sequence. We use the forward backward algorithm described in equations 5.10 to 5.17 to identify the posterior space. We initialize the state dynamics of our model as $A_{|c=1} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$, and $A_{|c=2} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$, and the state transition covariance as

$\Gamma = \begin{pmatrix} 0.05 & 0 \\ 0 & 0.05 \end{pmatrix}$. The dictionary $W = \begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix}$ and the covariance of the observations $\Sigma = 0.2\mathbb{1}$. The prior parameters are set to be $\vec{\pi} = \left(\frac{1}{3}, \frac{2}{3}\right)^T$, $\hat{\vec{z}}_1 = (1,1)^T$, and $\hat{V}_1 = \mathbb{1}$. An example of the generated data can be seen in figure 5.2 (red dotted line) together with the ground truth state that generated the observations (blue solid line). In the four plots of figure 5.2 we see the performance of different steps of our approximation and the reconstruction of the results. In the top plot we see the prediction for the model state as performed by equation 5.13. In the second plot we see the state estimate corrected by the datapoint in the filtering process 5.16. In the third plot we see the the state estimate of the filtering process 5.18 which is the mean of the posterior at time t with respect to the entire dataset. Comparing the top 3 plots one notices a progressive improvement in the state estimate on account of the additional data used for the estimate at each of these updates. In the last plot of figure 5.2 we present the reconstruction of our data from the posterior estimate. One notices that there is an error in the reconstruction that is due to the noise of the model. Since, the dynamics matrix $A_{|c=1}$ and the observation matrix W push the model harder along the first dimension the noise as well as the state are greater along the x-axis.

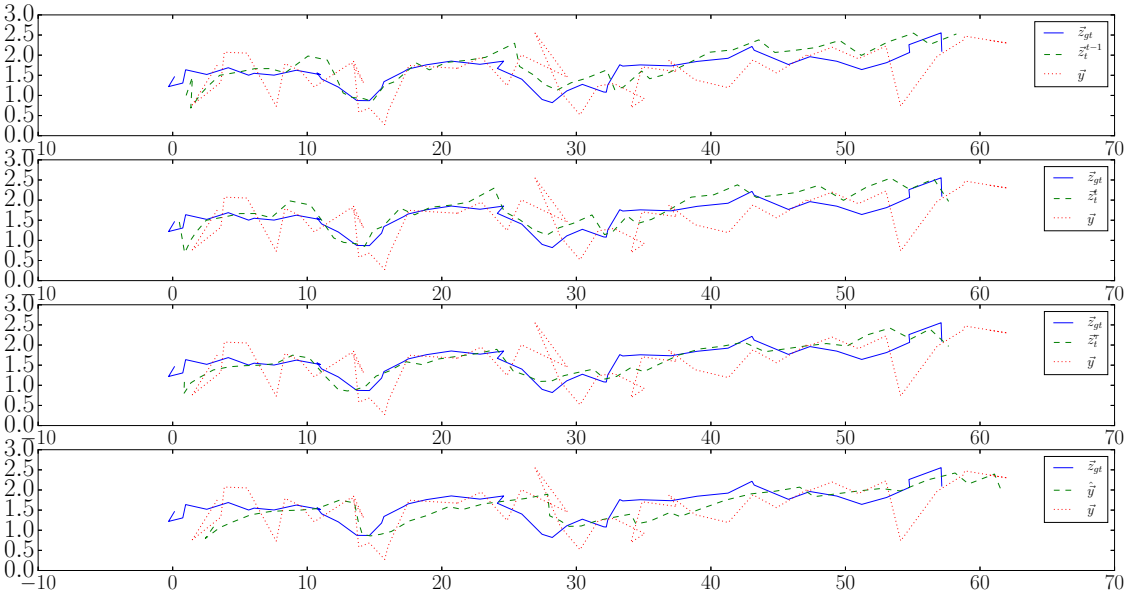

Figure 5.2.: **BDS state-space inference**: In the above plots you see an artificially generated sequence from the BDS data model, where $\vec{y}$ (red dotted line) are the generated observations, and $\vec{z}_{gt}$ are generated state space representations. (TOP-BOTTOM) In the first figure, we also plot the predicted state at each time point $\vec{z}_t^{t-1}$, i.e. the mean of $p\left(\vec{z}_t|\vec{y}_{1:t-1}\right)$. In the second figure, we plot the filtered state at each time point $\vec{z}_t^{t}$, i.e. the mean of $p\left(\vec{z}_t|\vec{y}_{1:t}\right)$. In the third figure, we plot the smoothened state representation at each time point $\vec{z}_t^{\tau}$, i.e. the mean of $p\left(\vec{z}_t|\vec{y}_{1:\tau}\right)$. Notice that each of them gets closer to the ground truth since it takes more observation about the sequence into account. In the last figure, we present the reconstruction of the observations $\hat{\vec{y}}_t = W\vec{s}_t^{\tau}$ from the smoothened states, i.e. the mean of $p\left(y_t|y_{1:\tau}\right)$.

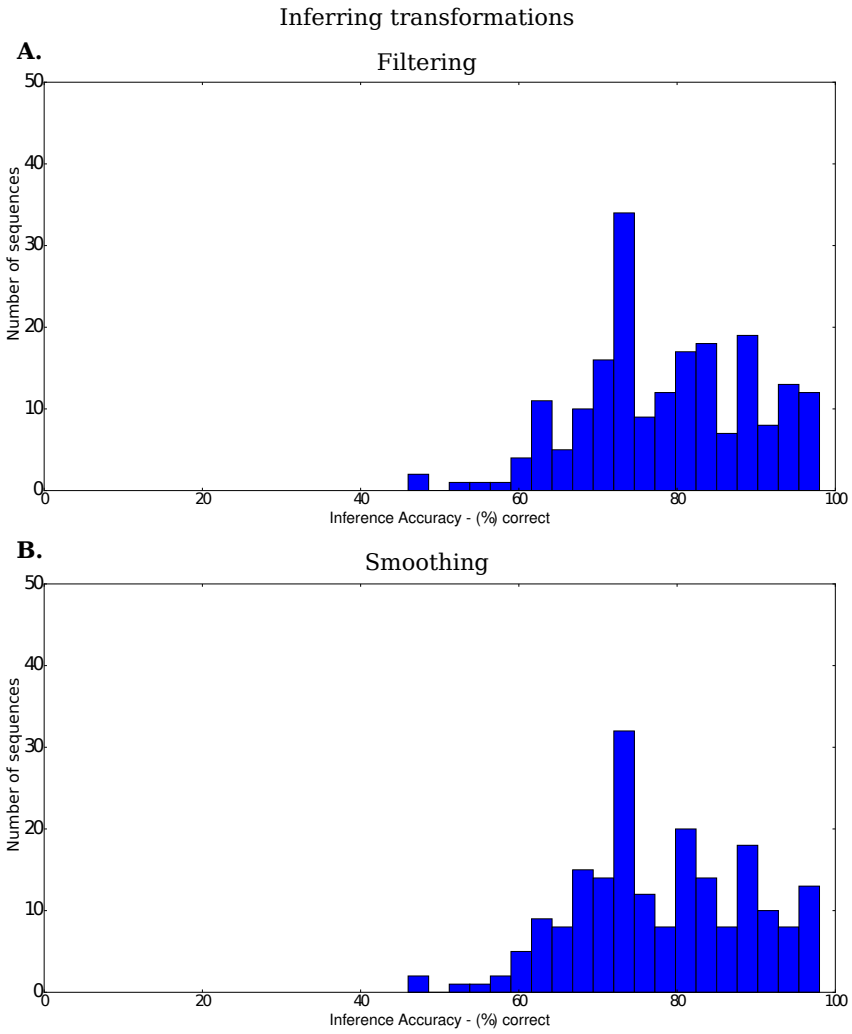

Figure 5.3.: **BDS transition inference**: In the two histograms above you see the percentage of correct state transitions inferred from artificial data. We used 200 sequences of length 50. Each sequence changes the model dynamics in manner specified by our samples of $p(k|\Theta)$. For the top histogram, we used the transition, $\vec{k}$, at time t of the filtering process that yields the highest posterior, i.e. $\arg\max_{\vec{k}_t} \left\{ p\left(\vec{k}_t | \vec{y}_{1:t}\right) \right\}$, and we achieved on average 79.6% correct predictions per sequence. For the other histogram, we used the transition, $\vec{k}$, at time t of the smoothing process that yields the highest posterior, i.e. $\arg\max_{\vec{k}_t} \left\{ p\left(\vec{k}_t | \vec{y}_{1:\tau}\right) \right\}$, and we achieved on average 80.4% correct predictions per sequence. Since the smoothing process takes into account "future" observations as well it is to be expected to have a higher accuracy for the prediction. It is interesting to see though that the filtering distribution is not falling far behind in terms of prediction performance.

This is probably due to the dynamics matrix $A_{|c=1}$ and the dictionary matrix W which are pushing values "harder" along the first dimension. This process increases the noise as well as the location estimate of the data.

In figure 5.3, we present the results on inference of the transformations. The results, show an error which is typical to models of this type (Ghahramani and Hinton, 2000). Notably, the inferred transformations extracted during the filtering process are fairly similar in performance to those extracted during the smoothing process where the full dataset is taken into account.

5.4. Discussion

In this chapter, we presented an algorithm that can be used to infer encodings of transformations from multi-dimensional sequential data. To infer the latent state of the model we used an algorithm similar to GBP1 in Bar-Shalom and Li (1993) adapted to the architecture of our model. The main difference of our work compared to most hybrid state space networks is that we assume that each transformation variable $\vec{k}$ is independent in time. Typical hybrid state space models appear to assume that if the state variables are going through certain dynamical process then the transition variables across state spaces will also go through a certain dynamical process (usually linear for the switching variables). This assumption stems from the observation that typically the behavior of "change" in dynamical systems is less volatile than the state of the system. However, we would expect that the data one would try to explain with a dynamical system of this form are sequences where the dynamics change abruptly and not in a particularly smooth manner. Another reason for assuming independence in the transformation variables is that it simplifies inference. Inferring the correct posterior in hybrid state space models (Ghahramani and Hinton, 2000; Luttinen et al., 2014) appears to be highly challenging throughout the literature even with ground truth parameters. It is our opinion that the literature in hybrid state space networks is mature enough in theoretical content but still rather difficult to manage when it comes to more realistic applications. Inference approaches exist that utilize MCMC methods (Carter and Kohn, 1996), variational approaches (Ghahramani and Hinton, 1996; Luttinen et al., 2014), truncated subspace approximations (Bar-Shalom et al., 1988), merging methods (Bar-Shalom and Li, 1993). These approaches while they maintain interesting properties are hard to manage in practice. Therefore, we chose to focus on a simpler hybrid state space network and learning process for our algorithm. We presented results in an inference task on artificial data that shows performance similar to established papers in the area (Ghahramani and Hinton, 2000). Our results show high accuracy in identifying transformations both during the filtering and the smoothing process. However, we need further work to be able to learn transformations rather than simply infer them from hard coded transformation matrices.

6. General Discussion and Conclusions

In this work we presented a number of probabilistic algorithms addressing primarily two machine learning topics, learning *invariant* representations and learning *transformations*. Discrete sparse coding and invariant discrete sparse coding were designed to learn representations of features in a dataset that were invariant to a set of discrete scaling units and one-dimensional translations. The deep gated autoencoder and the bilinear dynamical system were designed for learning and inference of transformations in groups of datapoints. We have presented a set of numerical experiments using those algorithms on both artificial and natural or "realistic" datasets which verify the functionality of our algorithms and outline significant technical features. The presented work was discussed, separately for each algorithm, at the end of each chapter. In this final chapter, we wish to compare some of the work discussed earlier across multiple chapters.

The two topics we deal with here are closely related, albeit they are clearly distinct, and we often find it necessary to discuss them jointly. In the case of invariant representations we are dealing with data that are similar to a degree of a certain set of variations. In our work these variations were explicitly modeled by the latent representation, however, it is quite common to endeavor invariant representations that simply discard the variations in the data. Having an explicit model of the change in your dataset is closely related to learning a transformation of the data. In the case of discrete sparse coding, for instance, inferring the scale of a datapoint can be used to produce a spike sequence that is normalized in terms of amplitude variations. This kind of normalization is quite similar to producing analogies of datapoint transformations as we did with deep gated autoencoder. This feature suggests that explicitly modeling invariances in data produces a transformation encoding similar to the one presented in chapters 4 and 5, although as we have defined it here it would be responsible for a more constrained set of transformations. Similarly, when defining an encoding of a transformation with the deep gated autoencoder you can use it to produce representations of datapoints that are altered by that transformation. Using an ensemble of different transformations would produce a set of representations of the datapoints that is as a whole a representation of the datapoint that is invariant to that set of transformations. Therefore, one could think of the transformation encoding process in the deep gated autoencoder as a method to implicitly define invariant representations of data. In fact, one could state that modeling "content" invariant to "change", or "change"

invariant to "content" is technically possible but there are strong dependencies in their definition.

Chapters 2 and 3 reveal some functional differences between the representations learned from discrete sparse coding and invariant discrete sparse coding. Especially, when using extra-cellular recordings the dictionary learned from the invariant discrete sparse coding appears to outperform an analysis done with the discrete sparse coding version. We expect that the differences between the two models would become less apparent on sequential data if the overlap between consecutive datapoints in the case of discrete sparse coding was higher, for instance, if two consecutive datapoints were responsible for the same observations in all except the first and last dimensions respectively. In that case we would be able to produce a maximum likelihood solution for the data using filters that do not account for the temporal structure in the data, i.e. more similar in structure to the ones in iDSC. This approach would be more similar to convolutional methods of sparse coding or in the discrete domain for spike analysis case more similar to the work presented in Pillow et al. (2013). The main issue we find in this work is that the timing for the spike is not taken into account, at least not during parameter estimation, and if it is then it is in some form separate processing stage. While this approach has shown interesting results we support our work in that learning based on temporal alignment could have considerable effects in the representation. Another, issue we have with this "convolution" based methods is that although overlapping spikes are taken into account (not so common but there is model based work here Pillow et al., 2013, and system based work in others) different combinations of temporal alignments for spikes, i.e. when trying to infer when a spike occurred the timing of that variable is treated independently of the spike timing of every other latent. This probably reduces the quality of the representation dramatically especially for neurons that exhibit a high degree of synchronization, as it happens for instance in the auditory cortex. The results of explicit time alignment during learning had strong effects in the neural data analysis but it does not leave unaffected other types of sequential data. In the audio dataset, for instance, we were able to extract high frequency features that we failed to see using discrete sparse coding. Interesting as that may be, temporal alignment does not appear to be as significant in audio recordings of human speech as it was on neural data. This is perhaps due to the fact that basic speech components , e.g. phonemes, are not strongly localized. In fact, they appear to vary based on the presence of an earlier or a latter phoneme, they change in duration and many other ways that would reduce the significance of temporal alignment and perhaps make it more important to introduce other types of invariances like filter size or introducing datapoint dependencies. In this case a "convolutional" variant for discrete sparse coding should also be considered. Probably the most prominent feature of our algorithm though that stands out both in DSC and iDSC is out ability to learn the prior and noise parameters. In other approaches, these parameters are typically set at ad-hoc values by the scientist and can lead to considerable biases in the results.

The other two models we wish to discuss are the deep gated autoencoder, presented in chapter 4, and the bilinear dynamical system, presented in chapter 5. They are both trying to achieve encodings of transformations to characterize the dataset and they are using a parametrized bilinear form to achieve that. However, there are also lot of very interesting and often quite subtle differences that could affect the behavior of the models. In the case of the bilinear dynamical system, we try to infer the encoding through a "search" in a latent random variable space while in the deep gated autoencoder we produce it deterministically using a parametrized bilinear operation in a feed-forward operation of a neural network. The "transformation" representation for the BDS model as is defined here is rather simple (only one transformation at a time $|\vec{k}| = 1$) and may never achieve the complexity of the representation produced by DGAE. However, the same modeling principles can be extended to work for any type of encoding with some refinement to the approximation of the posterior. Another, difference to note is that the two data points in the DGAE are projected to a different feature space (through different feature matrices U, and V) while the consecutive data points in the BDS model are connected by transformations defined in the same feature space. One could imagine BDS compensating on the state space representation by using a higher dimensional state space. However, the computational complexity of the inference process in the BDS algorithm is greater than the one in DGAE and therefore scaling in higher dimensions is more difficult. In any case, the fact that we introduce a probabilistic representation for the encoding allows us to work on transformation that are easier to manipulate and encodings that are more rigorously defined. Compared to other models defined in the area, we have focused our efforts to simplifying the model so that we make inference and learning more accurate but also faster and better scalable.

The work presented in this thesis outlines several aspects of unsupervised learning with latent variables that specialize in different types of structure in the data. Learning invariances or transformations from a dataset has proven to be quite difficult. However, we have shown ways to overcome several of the challenges involved with inference and learning in this context.

Appendix A.

Discrete Sparse Coding

A.1. M-step

To optimize the free energy with respect to W we find the gradient of the free energy and set it to zero

$$
\begin{aligned}
\nabla_w \mathcal{F} &= \nabla_w \sum_{n\in\mathcal{M}} \left[\left\langle \log p\left(\vec{y}^{(n)} | \vec{s}, \vec{t}, \Theta\right) \right\rangle_{q^{(n)}} + \left\langle \log p\left(\vec{s}, \vec{t} | \Theta\right) \right\rangle_{q^{(n)}} \right] \\
&= \nabla_w \sum_{n\in\mathcal{M}} \left[\left\langle -\frac{D}{2} \log 2\left(\pi\sigma^2\right) - \frac{\sigma^{-2}}{2} \left\| \vec{y}^{(n)} - W\vec{s} \right\|_2^2 \right\rangle_{q^{(n)}} \right. \\
&\quad \left. + \left\langle \sum_{h,k} \delta\left(\phi_k, s_h\right) \log \frac{\pi_k}{\tau} \right\rangle_{q^{(n)}} \right] = 0 \Leftrightarrow \\
&\quad \nabla_w \sum_{n\in\mathcal{M}} \left\langle \left\| \vec{y}^{(n)} - W\vec{s} \right\|_2^2 \right\rangle_{q^{(n)}} = 0 \Leftrightarrow \\
&\quad \sum_{n\in\mathcal{M}} \left\langle 2\left(\vec{y}^{(n)} - W\vec{s}\right) \vec{s}^T \right\rangle_{q^{(n)}} = 0 \Leftrightarrow \\
&\quad \sum_{n\in\mathcal{M}} \left\langle \vec{y}^{(n)} \vec{s}^T \right\rangle_{q^{(n)}} - \sum_{n\in\mathcal{M}} \left\langle W\vec{s}\vec{s}^T \right\rangle_{q^{(n)}} = 0 \Leftrightarrow \\
&\quad \sum_{n\in\mathcal{M}} \vec{y}^{(n)} \left\langle \vec{s}^T \right\rangle_{q^{(n)}} - W \sum_{n\in\mathcal{M}} \left\langle \vec{s}\vec{s}^T \right\rangle_{q^{(n)}} = 0 \Leftrightarrow \\
&\quad W = \sum_{n\in\mathcal{M}} \vec{y}^{(n)} \left\langle \vec{s}^T \right\rangle_{q^{(n)}} \left(\sum_{n\in\mathcal{M}} \left\langle \vec{s}\vec{s}^T \right\rangle_{q^{(n)}} \right)^{-1}
\end{aligned}
$$

Similarly, we can optimize the free energy with respect to σ by setting the free energy gradient w.r.t. to σ equal to zero.

$$
\begin{aligned}
\nabla_\sigma \mathcal{F} &= \nabla_\sigma \sum_{n\in\mathcal{M}} \left[\left\langle \log p\left(\vec{y}^{(n)} | \vec{s}, \Theta\right) \right\rangle_{q^{(n)}} + \left\langle \log p\left(\vec{s}|\Theta\right) \right\rangle_{q^{(n)}} \right] \\
&= \nabla_\sigma \sum_{n\in\mathcal{M}} \left[\left\langle -\frac{D}{2} \log\left(2\pi\sigma^2\right) - \frac{\sigma^{-2}}{2} \left\| \vec{y}^{(n)} - W\vec{s} \right\|_2^2 \right\rangle_{q^{(n)}} \right. \\
&\quad \left. + \left\langle \sum_{h,k} \delta\left(\phi_k, s_h\right) \log \pi_k \right\rangle_{q^{(n)}} \right] = 0 \Leftrightarrow \\
&\quad \sum_{n\in\mathcal{M}} \left\langle -D\frac{1}{\sigma^2}\sigma \right\rangle_{q^{(n)}} + \sum_{n\in\mathcal{M}} \left\langle \sigma^{-3} \left\| \vec{y}^{(n)} - W\vec{s} \right\|_2^2 \right\rangle_{q^{(n)}} = 0 \Leftrightarrow \\
&\quad -\frac{D\left|\mathcal{M}\right|}{\sigma} + \sigma^{-3} \sum_{n\in\mathcal{M}} \left\langle \left\| \vec{y}^{(n)} - W\vec{s} \right\|_2^2 \right\rangle_{q^{(n)}} = 0 \Leftrightarrow \\
&\quad \sigma^2 = \frac{1}{D\left|\mathcal{M}\right|} \sum_{n\in\mathcal{M}} \left\langle \left\| \vec{y}^{(n)} - W\vec{s} \right\|_2^2 \right\rangle_{q^{(n)}} \Leftrightarrow \\
&\quad \sigma = \sqrt{\frac{1}{D\left|\mathcal{M}\right|} \sum_{n\in\mathcal{M}} \left\langle \left\| \vec{y}^{(n)} - W\vec{s} \right\|_2^2 \right\rangle_{q^{(n)}}}
\end{aligned}
$$

To optimize the free energy with respect to π we find the gradient of the free energy and set it to zero under the constraint that $\sum_i \pi_i = 1$. To constrain the free energy appropriately we use the Lagrange multipliers method with $\sum_i \pi_i = 1$ as the Lagrangian.

$$
\begin{aligned}
\nabla_{\pi_i}\mathcal{F} &= \nabla_{\pi_i}\sum_{n\in\mathcal{M}}\left[\left\langle\log p\left(\vec{y}^{(n)}|\vec{s},\Theta\right)\right\rangle_{q^{(n)}}+\left\langle\log p\left(\vec{s},\vec{t}|\Theta\right)\right\rangle_{q^{(n)}}\right] \\
&\quad -\nabla_{\pi_i}\lambda\left(\sum_k\pi_k-1\right) \\
&= \nabla_{\pi_i}\sum_{n\in\mathcal{M}}\left[\left\langle-\frac{D}{2}\log\left(2\pi\sigma^2\right)-\frac{\sigma^{-2}}{2}\left\|\vec{y}^{(n)}-W\vec{s}\right\|_2^2\right\rangle_{q^{(n)}}\right. \\
&\quad \left.+\left\langle\sum_{h,k}\delta\left(\phi_k,s_h\right)\log\pi_k\right\rangle_{q^{(n)}}\right]-\nabla_{\pi_i}\lambda\left(\sum_k\pi_k-1\right)=0\Leftrightarrow \\
&\quad \nabla_{\pi_l}\sum_{n\in\mathcal{M}}\left\langle\sum_{h,k}\delta\left(\phi_k,s_h\right)\log\pi_k\right\rangle_{q^{(n)}}-\nabla_{\pi_l}\lambda\left(\sum_k\pi_k-1\right)=0\Leftrightarrow \\
&\quad \sum_{n\in\mathcal{M}}\left\langle\sum_{h,k}\delta\left(\phi_k,s_h\right)\frac{1}{\pi_i}\right\rangle_{q^{(n)}}-\lambda=0\Leftrightarrow \\
&\quad \pi_i=\frac{\sum_{n\in\mathcal{M}}\left\langle\sum_{h,k}\delta\left(\phi_k,s_h\right)\right\rangle_{q^{(n)}}}{\lambda}
\end{aligned}
$$

For λ we have:

$$
\begin{aligned}
&\lambda\left(\sum_k\pi_k-1\right)=0\Leftrightarrow \\
&\lambda\left(\sum_k\frac{\sum_{n\in\mathcal{M}}\left\langle\sum_{h,k}\delta\left(\phi_k,s_h\right)\right\rangle_{q^{(n)}}}{\lambda}-1\right)=0\Leftrightarrow \\
&\lambda=\sum_k\sum_{n\in\mathcal{M}}\left\langle\sum_{h,k}\delta\left(\phi_k,s_h\right)\right\rangle_{q^{(n)}}
\end{aligned}
$$

Appendix B.

Time-invariant Discrete Sparse Coding

B.1. M-step

To optimize the free energy with respect to W we find the gradient of the free energy and set it to zero

$$\begin{aligned}
\nabla_w \mathcal{F} &= \nabla_w \sum_{n \in \mathcal{M}} \left[\left\langle \log p\left(\vec{y}^{(n)} | \vec{s}, \vec{t}, \Theta\right) \right\rangle_{q^{(n)}} + \left\langle \log p\left(\vec{s}, \vec{t} | \Theta\right) \right\rangle_{q^{(n)}} \right] \\
&= \nabla_w \sum_{n \in \mathcal{M}} \left[\left\langle -\frac{D}{2} \log 2\left(\pi\sigma^2\right) - \frac{\sigma^2}{2} \left\| \vec{y}^{(n)} - f\left(W, \vec{t}\right) \vec{s} \right\|_2^2 \right\rangle_{q^{(n)}} \right. \\
&\quad \left. + \left\langle \sum_{h,k} \delta\left(\phi_k, s_h\right) \log \frac{\pi_k}{\tau} \right\rangle_{q^{(n)}} \right] = 0 \Leftrightarrow
\end{aligned}$$

The only component with a non-zero gradient is the quadratic:

$$
\begin{aligned}
&\frac{\partial}{\partial W_{k,l}} \left\langle \sum_{i=1,\dots,D_y} \left(y_i - \sum_j W_{i+t_j,j} s_j \right)^2 \right\rangle_{q^{(n)}} \\
&= \left\langle 2 \left(y_i - \sum_j W_{i+t_j,j} s_j \right) \frac{\partial}{\partial W_{k,l}} \sum_j W_{i+t_j,j} s_j \right\rangle_{q^{(n)}} \\
&= \left\langle 2 \left(y_i - \sum_j W_{i+t_j,j} s_j \right) \delta\left(k = i + t_j\right) \delta\left(l = j\right) s_j \right\rangle_{q^{(n)}} \\
&= \left\langle 2 \left(y_i - \sum_j W_{i+t_j,j} s_j \right) \delta\left(k = i + t_j\right) s_l \right\rangle_{q^{(n)}} \\
&= 2 \left\langle y_i \delta\left(k = i + t_j\right) s_l - \sum_j W_{i+t_j,j} s_j \delta\left(k = i + t_j\right) s_l \right\rangle_{q^{(n)}} = 0 \Leftrightarrow \\
&\left\langle y_i \delta\left(k = i + t_j\right) s_l \right\rangle_{q^{(n)}} = \left\langle \sum_j W_{i+t_j,j} s_j \delta\left(k = i + t_j\right) s_l \right\rangle_{q^{(n)}} \Leftrightarrow \\
&\left\langle y_{k-t_j} s_l \right\rangle_{q^{(n)}} = \sum_j W_{k,j} \left\langle s_j s_l \right\rangle_{q^{(n)}} \Leftrightarrow \text{ multiply with the inverse of } s_j s_l \\
&\left\langle y_{k-t_j} s_l \right\rangle_{q^{(n)}} \left\langle s_j s_l \right\rangle^{-1}_{q^{(n)}} = \sum_j W_{k,j} \delta\left(j = l\right) \Leftrightarrow \\
&W_{k,l} = \left\langle y_{k-t_j} s_l \right\rangle_{q^{(n)}} \left\langle \left(s_j s_l\right) \right\rangle^{-1}_{q^{(n)}} \Leftrightarrow \\
&W = \left\langle \vec{y}_{-t} \vec{s} \right\rangle_{q^{(n)}} \left\langle \left(s s^T\right) \right\rangle^{-1}_{q^{(n)}}
\end{aligned}
$$

The result aligns the datapoints with the corresponding column as prescribed by the shift variable

The M-step updates for the σ, and $\vec{pi}$ parameters of the iDSC algorithm do not change significantly when compared with the DSC case since they are not affected by the function $f\left(W, \vec{t}\right)$, see A.1.

Appendix C.

BiLinear Dynamical Systems

C.1. Properties of Mutlivariate Gaussian Distributions

Let $\vec{x}$, and $\vec{y}$ two vectors with dimensions D_x, and D_y respectively. We define the vector $\vec{z} = \begin{bmatrix} \vec{x} \\ \vec{y} \end{bmatrix}$ of dimensionality $D = D_x + D_y$ that is composed by concatenating $\vec{x}$, and $\vec{y}$. We assume that a multivariate gaussian over $\vec{z}$, i.e. $p(\vec{z}) = \mathcal{N}(\vec{z}|\vec{\mu}, \Sigma)$, is parametrized by:

$$\vec{\mu} = \begin{bmatrix} \vec{\mu}_x \\ \vec{\mu}_y \end{bmatrix} \tag{C.1}$$

$$\Sigma = \begin{bmatrix} \Sigma_x & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma \end{bmatrix} \tag{C.2}$$

where $\vec{\mu}_x \in \mathbb{R}^{D_x}$,$\vec{\mu}_y \in \mathbb{R}^{D_y}$,$\Sigma_x \in \mathbb{R}^{D_x \times D_x}$, and $\Sigma_y \in \mathbb{R}^{D_y \times D_y}$.
then the **marginal** distribution of $\vec{x}$ is:

$$p(\vec{x}) = \mathcal{N}(\vec{x}|\vec{\mu}_x, \Sigma_x) \tag{C.3}$$

and the **conditional** distribution of $\vec{x}$ given $\vec{y}$ is parametrized by:

$$\mu_{x|y} = \mu_x + \Sigma_{xy}\Sigma_{yy}^{-1}(\vec{y} - \vec{\mu}_y) \text{ , and} \tag{C.4}$$

$$\Sigma_{x|y} = \Sigma_{xx} - \Sigma_{xy}\Sigma_{yy}^{-1}\Sigma_{yx} \tag{C.5}$$

$$\text{i.e. } p(\vec{x}|\vec{y}) = \mathcal{N}\left(\vec{x}|\mu_x + \Sigma_{xy}\Sigma_{yy}^{-1}(\vec{y} - \vec{\mu}_y), \Sigma_{xx} - \Sigma_{xy}\Sigma_{yy}^{-1}\Sigma_{yx}\right) \tag{C.6}$$

We **merge** a mixture of Gaussians model to single Gaussian distribution with paramters:

$$\vec{\mu} = E[x] = E[E[x|k]] = \sum_c \pi_c \vec{\mu}_{c=1} \tag{C.7}$$

$$\begin{aligned}
\Sigma &= E\left[\vec{x}\vec{x}^T\right] - E[\vec{x}]\,E[\vec{x}]^T = E\left[E\left[\vec{x}\vec{x}^T|k\right]\right] - E[E[\vec{x}|k]]\,E[E[\vec{x}|k]]^T \\
&= E\left[\text{cov}[\vec{x}|k] + E[\vec{x}|k]\,E[\vec{x}|k]^T\right] - E[E[\vec{x}|k]]\,E[E[\vec{x}|k]]^T \\
&= E[\text{cov}[\vec{x}|k]] + E\left[E[\vec{x}|k]\,E[\vec{x}|k]^T\right] - E[E[\vec{x}|k]]\,E[E[\vec{x}|k]]^T \\
&= E[\text{cov}[\vec{x}|k]] + \text{cov}\left[E[\vec{x}|k]\,E[\vec{x}|k]^T\right] \\
&= \sum_c \pi_c \Sigma_c - \sum_c \pi_c (\vec{\mu}_c - \vec{\mu})(\vec{\mu}_c - \vec{\mu})^T \\
&= \sum_c \pi_c \left(\Sigma_c - (\vec{\mu}_c - \vec{\mu})(\vec{\mu}_c - \vec{\mu})^T\right)
\end{aligned} \tag{C.8}$$

for a detailed derivation of marginalization and conditioning over Gaussians see Bishop (2006).

C.2. Filtering - Forward Pass

The filtering process identifies the probabilities $p(\vec{z}_t|\vec{y}_{1:t-1})$, $p\left(\vec{k}_t|\vec{y}_{1:t}\right)$, and $p(\vec{z}_t|\vec{y}_{1:t})$.

$$E\left[\vec{z}_t|\vec{y}_{1:t-1},\vec{k}\right] \;=\; A_{|k}\hat{z}_{t-1}^{t-1} = \hat{z}_{t|k}^{t-1} \tag{C.9}$$

$$\text{cov}\left[\vec{z}_t|\vec{y}_{1:t-1},\vec{k}\right] \;=\; A_{|k}V_{t-1}^{t-1}A_{|k}^T + \Gamma = V_{t|k}^{t-1} \tag{C.10}$$

$$p(k_{c=1}|\vec{y}_{1:t}) = \frac{p(\vec{y}_t|\vec{y}_{1:t-1},k_c=1)\,\pi_c^{t-1}}{\sum_{c'} p(\vec{y}_t|\vec{y}_{1:t-1},k_{c'}=1)\,\pi_{c'}^{t-1}} = \pi_c^t \tag{C.11}$$

where $p(\vec{y}_t|\vec{y}_{1:t-1},k_c=1)$ is parametrized by:

$$E\left[\vec{y}_t|\vec{y}_{1:t-1},\vec{k}\right] = W\hat{z}_t^{t-1}$$

$$\text{cov}\left[\vec{y}_t|\vec{y}_{1:t-1},\vec{k}\right] = WV_t^{t-1}W^T + \Sigma$$

now we can use equation C.11 with C.7 andC.8 to merge the predictions into a single

Gaussian:

$$E\left[E\left[\vec{z}_t|\vec{y}_{1:t-1},\vec{k}\right]\right] = \sum_c \pi_c^t \hat{z}_t^{t-1} = \hat{z}_t \tag{C.12}$$

$$\begin{aligned} E\left[\text{cov}\left[\vec{z}_t|\vec{y}_{1:t-1},\vec{k}\right]\right] &= \sum_c \pi_c^t \left(V_{t|k_c=1}^{t-1} + \left(\hat{z}_{t|k_c=1}^{t-1} - \hat{z}_t^{t-1}\right)\left(\hat{z}_{t|k_c=1}^{t-1} - \hat{z}_t^{t-1}\right)^T\right) \\ &= V_t^{t-1} \end{aligned} \tag{C.13}$$

now that we have the prediction we can proceed with the correction as in the standard kalman filter case Bishop (2006); Murphy (2012)

$$K_t = V_t^{t-1} W^T \left(\Sigma + W V_t^{t-1} W^T\right)^{-1} \tag{C.14}$$

$$\hat{z}_t^t = \hat{z}_t^{t-1} + K_t\left(\vec{y}_t - W\hat{z}_t^{t-1}\right) \tag{C.15}$$

$$V_t^t = \left(I - K_t W\right) V_t^{t-1} \tag{C.16}$$

Equations C.12-C.13 parametrize $p\left(\vec{z}_t|\vec{y}_{1:t-1}\right)$, equation C.11 parametrizes $p\left(\vec{k}_t|\vec{y}_{1:t}\right)$, and equations C.15-C.16 parametrize $p\left(\vec{z}_t|\vec{y}_{1:t}\right)$. Given the transformation $\vec{k}$ the filtering equations fall back to the filtering equations of the Kalman filter. For equations C.9,C.10, and C.14-C.16 we Kalman filter recursice rules that can easily be found in the literature Bishop (2006); Murphy (2012).

C.3. Smoothing - Backward Pass

In the smoothing stage we identify $p\left(\vec{k}_t|\vec{y}_{1:\tau}\right)$, and $p\left(\vec{z}_t|\vec{y}_{1:\tau}\right)$. We achieve that by defining $p\left(\vec{z}_{t-1},\vec{z}_t|\vec{y}_{1:\tau},\vec{k}_t\right)$ and then identifying the conditional $p\left(\vec{z}_{t-1}|\vec{z}_t,\vec{y}_{1:\tau},\vec{k}_t\right)$.

The distribution $p\left(\vec{z}_{t-1},\vec{z}_t|\vec{y}_{1:\tau},\vec{k}_t\right)$ is parametrized by:

$$\begin{bmatrix} \hat{z}_{t-1}^{t-1} \\ \hat{z}_{t-1}^{t-1} \\ \hat{z}_{t|k}^{t-1} \end{bmatrix}$$

$$\begin{bmatrix} V_{t-1}^{t-1} & V_{t-1}^{t-1} A_{|k}^T \\ A_{|k}^T V_{t-1}^{t-1} & V_{t|k}^{t-1} \end{bmatrix}$$

conditioning on $\vec{z}_t$ using C.4 and C.5 we get:

$$\hat{z}_{t-1|k}^{t} = \hat{z}_{t-1}^{t-1} + V_{t-1}^{t-1} A_{|k}^{T} \left(V_{t|k}^{t-1}\right)^{-1} \left(\vec{z}_{t|k}^{t-1} - \hat{z}_{t|k}^{t-1}\right)$$

$$V_{t-1|k}^{t} = V_{t-1}^{t-1} - V_{t-1}^{t-1} A_{|k}^{T} \left(V_{t|k}^{t-1}\right)^{-1} A_{|k}^{T} V_{t-1}^{t-1}$$

defining $J_{t-1|k} = V_{t-1}^{t-1} A_{|k}^{T} \left(V_{t|k}^{t-1}\right)^{-1}$

$$\hat{z}_{t-1|k}^{t} = \hat{z}_{t-1}^{t-1} + J_{t-1|k} \left(\vec{z}_{t|k} - \hat{z}_{t|k}^{t-1}\right)$$

$$V_{t-1|k}^{t} = V_{t-1}^{t-1} - J_{t-1|k} V_{t|k}^{t-1} J_{t-1|k}^{T}$$

for the mean of $p\left(\vec{z}_t|\vec{y}_{1:\tau}\right)$ we have:

$$\begin{aligned}
\hat{z}_{t-1|k}^{\tau} &= E\left[\vec{z}_{t-1}|\vec{y}_{1:\tau}\right] = E\left[E\left[\vec{z}_{t-1}|\vec{y}_{1:\tau}, \vec{k}_t, \vec{z}_t\right]|\vec{y}_{1:\tau}\right] \\
&= E\left[E\left[\vec{z}_{t-1}|\vec{y}_{1:\tau}, \vec{k}_t, \vec{z}_t\right]|\vec{y}_{1:\tau}\right] \\
&= E\left[\hat{z}_{t-1}^{t-1} + J_{t-1|k}\left(\vec{z}_{t|k} - \hat{z}_{t|k}^{t-1}\right)|\vec{y}_{1:\tau}\right] \\
&= \hat{z}_{t-1}^{t-1} + J_{t-1|k}\left(\hat{z}_{t|k}^{\tau} - \hat{z}_{t|k}^{t-1}\right)
\end{aligned}$$

the mean of $p\left(\vec{z}_t|\vec{y}_{1:\tau}\right) = \sum_{\mathbf{k}} \hat{z}_{t-1|k}^{\tau}$ requires calculation over all the potential combinations of $\vec{k}$ over τ datapoints. Since this is intractable we introduce the term $\hat{z}_{t-1}^{\tau} = \sum_c \pi_c \hat{z}_{t|k_c=1}^{\tau}$ in the recursive rule above.

$$\hat{z}_{t-1|k}^{\tau} = \hat{z}_{t-1}^{t-1} + J_{t-1|k}\left(\hat{z}_{t}^{\tau} - \hat{z}_{t|k}^{t-1}\right)$$

and for the variance

$$
\begin{aligned}
V^{\tau}_{t-1|k} &= \text{cov}\left[\vec{z}_{t-1}|\vec{y}_{1:\tau}\right] \\
&= \text{cov}\left[E\left[\vec{z}_{t-1}|\vec{z}_t, \vec{k}_t, \vec{y}_{1:\tau}\right]|\vec{y}_{1:\tau}\right] + E\left[\text{cov}\left[\vec{z}_{t-1}|\vec{k}_t, \vec{z}_t, \vec{y}_{1:\tau}\right]|\vec{y}_{1:\tau}\right] \\
&= \text{cov}\left[\hat{z}^{t-1}_{t-1} + J_{t-1|k}\left(\vec{z}_{t|k} - \hat{z}^{t-1}_{t|k}\right)|\vec{y}_{1:\tau}\right] + E\left[V^{t-1}_{t-1} - J_{t-1|k}V^{t-1}_{t|k}J^{T}_{t-1|k}|\vec{y}_{1:\tau}\right] \\
&= J_{t-1|k}\text{cov}\left[\left(\vec{z}_{t|k} - \hat{z}^{t-1}_{t|k}\right)|\vec{y}_{1:\tau}\right]J^{t}_{t-1|k} + V^{t-1}_{t-1} - J_{t-1|k}V^{t-1}_{t|k}J^{T}_{t-1|k} \\
&= J_{t-1|k}V^{\tau}_{t|k}J^{t}_{t-1|k} + V^{t-1}_{t-1} - J_{t-1|k}V^{t-1}_{t|k}J^{T}_{t-1|k} \\
&= V^{t-1}_{t-1} - J_{t-1|k}\left(V^{\tau}_{t|k} - V^{t-1}_{t|k}\right)J^{T}_{t-1|k}
\end{aligned}
$$

once more we simplify the recursion by merging the covariances of the earlier step as:

$$
V^{\tau}_{t-1} = \sum_{c}\pi_c\left(V^{\tau}_{t|k_c=1} + \left(\hat{z}^{\tau}_{t-1|k_c=1} - \hat{z}^{\tau}_{t-1}\right)\left(\hat{z}^{\tau}_{t-1|k_c=1} - \hat{z}^{\tau}_{t-1}\right)^{T}\right)
$$

so the recursion rule becomes.

$$
V^{\tau}_{t-1|k} = V^{t-1}_{t-1} - J_{t-1|k}\left(V^{\tau}_{t} - V^{t-1}_{t|k}\right)J^{T}_{t-1|k}
$$

The approximations in the update rules have been show to approximate the posterior fairly well in section 5. An additional statistic that we would need to compute is the cross-covariance $V_{t-1,t|k}$ for which this approximation does not provide a sufficient description. One might look to extend this approximation in a manner analogous to the GPB2 in Bar-Shalom and Li (1993) where the probabilities are merged after every second time step.

C.4. Parameter Estimation

We take the objective that we need to optimize in the M-step from 5.5 :

$$
\begin{aligned}
Q &= \sum_n \sum_t \int_{\vec{z}_t} \sum_k q\left(\vec{z}_t, k|\Theta^{\text{old}}\right) \log p\left(\vec{y}_t^{(n)}|z_t, k, \Theta\right) \\
&+ \sum_{t=2} \int_{\vec{z}_{t,t-1}} \sum_k q\left(\vec{z}_{t,t-1}, k|\Theta^{\text{old}}\right) \log p\left(\vec{z}_t|k, \vec{z}_{t-1}, \Theta\right) p(k|\Theta) \\
&+ \int_{\vec{z}_1} \sum_k q\left(\vec{z}_1|\Theta^{\text{old}}\right) \log p\left(\vec{z}_1|\Theta\right) \\
&= \sum_n \left(\sum_t \left\langle -\frac{1}{2} \log\det\left(2\pi\Sigma\right) - \frac{1}{2}\left(\vec{y}_t^{(n)} - W\vec{z}_t\right)\Sigma^{-1}\left(\vec{y}_t^{(n)} - W\vec{z}_t\right)^T \right\rangle_q \right. \\
&+ \sum_{t=2} \left\langle -\frac{1}{2} \log\det\left(2\pi\Gamma\right) - \frac{1}{2}\left(\vec{z}_t - A_{|k}\vec{z}_{t-1}\right)\Gamma^{-1}\left(\vec{z}_t - A_{|k}\vec{z}_{t-1}\right)^T + \sum_c k_c \log\left(\pi_c\right) \right\rangle_q \\
&\left. + \left\langle -\frac{1}{2} \log\det\left(2\pi R\right) - \frac{1}{2}\left(\vec{y}_t^{(n)} - W\vec{z}_{1,k}^{\tau}\right)R^{-1}\left(\vec{y}_t^{(n)} - W\vec{z}_{1,k}^{\tau}\right)^T \right\rangle_q \right)
\end{aligned}
$$

For the dictionary matrix W we find:

$$
\begin{aligned}
\nabla_w Q = \nabla_w \sum_n \left(\sum_t \left\langle -\frac{1}{2}\left(\vec{y}_t^{(n)} - W\vec{z}_t\right)\Sigma^{-1}\left(\vec{y}_t - W\vec{z}_t\right)^T \right\rangle_q \right) = 0 \Leftrightarrow \\
\sum_n \left(\sum_t \left\langle \left(\vec{y}_t^{(n)} - W\vec{z}_t\right)\vec{z}_t^{T} \right\rangle_q \right) = 0 \Leftrightarrow \\
W^{\text{new}} = \sum_n \left(\vec{y}_t^{(n)} \sum_t \left\langle \vec{z}_t^{T} \right\rangle_q \right) \left(\sum_n \left(\sum_t \left\langle \vec{z}_t \vec{z}_t^{T} \right\rangle_q \right) \right)^{-1}
\end{aligned}
$$

For the observation covariance matrix Σ we find:

$$\nabla_\Sigma Q = \nabla_\Sigma \sum_n \left(\sum_t \left\langle -\frac{1}{2} \log\det\left(2\pi\Sigma\right) - \frac{1}{2}\left(\vec{y_t}^{(n)} - W\vec{z}_t\right)\Sigma^{-1}\left(\vec{y_t}^{(n)} - W\vec{z}_t\right)^T \right\rangle_q \right) = 0 \Leftrightarrow$$

$$\sum_n \left(\sum_t \left\langle -\frac{1}{2}\Sigma^{-1} + \frac{1}{2}\Sigma^{-1}\left(\vec{y_t}^{(n)} - W\vec{z}_t\right)\left(\vec{y_t}^{(n)} - W\vec{z}_t\right)^T\Sigma^{-1} \right\rangle_q \right) = 0 \Leftrightarrow$$

$$\sum_n \left(\sum_t \left\langle -\Sigma + \left(\vec{y_t}^{(n)} - W\vec{z}_t\right)\left(\vec{y_t}^{(n)} - W\vec{z}_t\right)^T \right\rangle_q \right) = 0 \Leftrightarrow$$

$$\Sigma^{\text{new}} = \frac{1}{NT}\sum_n \left(\sum_t \left\langle \left(\vec{y_t}^{(n)} - W\vec{z}_t\right)\left(\vec{y_t}^{(n)} - W\vec{z}_t\right)^T \right\rangle_q \right) \Leftrightarrow$$

$$\Sigma^{\text{new}} = \frac{1}{NT}\sum_n \left(\sum_t \left\langle \vec{y_t}^{(n)}\vec{y_t}^{(n)T} - W\vec{z}_t\vec{y_t}^{(n)T} - \vec{y_t}^{(n)}\vec{z_t}^T W^T + W\vec{z}_t\vec{z_t}^T W^T \right\rangle_q \right)$$

$$\Sigma^{\text{new}} = \frac{1}{NT}\sum_n \left(\sum_t \vec{y_t}^{(n)}\vec{y_t}^{(n)T} - W\left\langle\vec{z}_t\right\rangle_q \vec{y_t}^{(n)T} - \vec{y_t}^{(n)}\left\langle\vec{z_t}^T\right\rangle_q W^T + W\left\langle\vec{z}_t\vec{z_t}^T\right\rangle_q W^T \right)$$

the expectation values are directly computed by the equations described in chapter 5.

For the parameters A, and Γ we need expectation values that at this time our algorithm does not compute. For the parameters $\vec{\mu}$, R, and π the update rules are trivial and nearly identical to the kalman filter literature or the prior update derived earlier in the discrete sparse coding section.

List of Figures

Bibliography

Ackerson, G. and Fu, K. (1970). On state estimation in switching environments. *IEEE Transactions on Automatic Control*, 15(1):10–17.

Aharon, M., Elad, M., and Bruckstein, A. (2006). K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation. *Signal Processing, IEEE Transactions on*, 54(11):4311–22.

Alain, D. and Olivier, S. (2013). Gated autoencoders with tied input weights. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, volume 28, pages 154–162.

Anderson, C. H. and Van Essen, D. C. (1987). Shifter circuits: a computational strategy for dynamic aspects of visual processing. *Proceedings of the National Academy of Sciences of the United States of America*, 84(17):6297–301.

Baldi, P. and Hornik, K. (1989). Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*, 2(1):53–58.

Baldi, P. and Sadowski, P. J. (2013). Understanding dropout. In *Advances in Neural Information Processing Systems*, pages 2814–2822.

Bar-Shalom, Y., Fortmann, T. E., Tracking, Association, D., et al. (1988). Mathematics in science and engineering.

Bar-Shalom, Y. and Li, X.-R. (1993). Estimation and tracking- principles, techniques, and software. *Norwood, MA: Artech House, Inc, 1993.*

Battiti, R. (1992). First-and second-order methods for learning: between steepest descent and newton's method. *Neural computation*, 4(2):141–166.

Baum, L. E., Petrie, T., Soules, G., and Weiss, N. (1970). A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The annals of mathematical statistics*, 41(1):164–171.

Bell, A. J. and Sejnowski, T. J. (1997). The "independent components" of natural scenes are edge filters. *Vision Research*, 37(23):3327–38.

Bengio, Y., Boulanger-Lewandowski, N., and Pascanu, R. (2013). Advances in optimizing recurrent networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8624–8628. IEEE.

Berkes, P., Turner, R., and Sahani, M. (2008). On sparsity and overcompleteness in image models. *Advances in Neural Information Processing Systems*, 21.

Bingham, E. and Hyvärinen, A. (2000). A fast fixed-point algorithm for independent component analysis of complex valued signals. *International journal of neural systems*, 10(01):1–8.

Bingham, E., Kabán, A., and Fortelius, M. (2009). The aspect bernoulli model: multiple causes of presences and absences. *Pattern Analysis and Applications*, 12(1):55–78.

Bishop, C. (2006). *Pattern Recognition and Machine Learning*. Springer.

Bornschein, J., Henniges, M., and Lücke, J. (2013). Are v1 simple cells optimized for visual occlusions? a comparative study. *PLoS Computational Biology*, 9(6):e1003062.

Bornschein, J., Henniges, M., Puertas, G., and Lücke, J. (2012). Are V1 receptive fields shaped by low-level visual occlusions? *in prep.*

Cadieu, C. F. and Olshausen, B. A. (2012). Learning intermediate-level representations of form and motion from natural movies. *Neural Computation*, 24(4):827–866.

Carlson, D. E., Vogelstein, J. T., Wu, Q., Lian, W., Zhou, M., Stoetzner, C. R., Kipke, D., Weber, D., Dunson, D. B., and Carin, L. (2014). Multichannel electrophysiological spike sorting via joint dictionary learning and mixture modeling. *IEEE Transactions on Biomedical Engineering*, 61(1):41–54.

Carter, C. K. and Kohn, R. (1996). Markov chain monte carlo in conditionally gaussian state space models. *Biometrika*, 83(3):589–601.

Dai, Z., Exarchakis, G., and Lücke, J. (2013). What are the invariant occlusive components of image patches? a probabilistic generative approach. In *Advances in Neural Information Processing Systems*, pages 243–251.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society B*, 39:1–38.

Donoho, D. L. (2006). Compressed sensing. *IEEE Transactions on information theory*, 52(4):1289–1306.

Eldar, Y. C. and Kutyniok, G. (2012). *Compressed sensing: theory and applications*. Cambridge University Press.

Exarchakis, G., Henniges, M., Eggert, J., and Lücke, J. (2012). Ternary sparse coding. In *Proceedings LVA/ICA*, LNCS. Springer. in press.

Exarchakis, G. and Lücke, J. (2017). Discrete sparse coding. *(in press) Neural Computation*.

Field, D. J. (1994). What is the goal of sensory coding? *Neural Comput.*, 6(4):559–601.

Földiák, P. (1991). Learning invariance from transformation sequences. *Neural Computation*, 3(2):194–200.

Frolov, A. A., Húsek, D., and Polyakov, P. Y. (2016). Comparison of seven methods for boolean factor analysis and their evaluation by information gain. *IEEE transactions on neural networks and learning systems*, 27(3):538–550.

Garofolo, J. S., Lamel, L. F., Fisher, W. M., Fiscus, J. G., Pallett, D. S., and Dahlgren, N. L. (1993). {DARPA} {TIMIT} Acoustic Phonetic Continuous Speech Corpus {CDROM}.

Ghahramani, Z. and Hinton, G. E. (1996). Parameter estimation for linear dynamical systems. Technical report, Technical Report CRG-TR-96-2, University of Totronto, Dept. of Computer Science.

Ghahramani, Z. and Hinton, G. E. (2000). Variational learning for switching state-space models. *Neural computation*, 12(4):831–864.

Goodfellow, I., Courville, A. C., and Bengio, Y. (2012). Large-scale feature learning with spike-and-slab sparse coding. In *ICML*.

Goodfellow, I. J., Courville, A., and Bengio, Y. (2013). Scaling up spike-and-slab models for unsupervised feature learning. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1902–1914.

Griffiths, T. L. and Ghahramani, Z. (2011). The indian buffet process: An introduction and review. *Journal of Machine Learning Research*, 12(Apr):1185–1224.

Haft, M., Hofman, R., and Tresp, V. (2004). Generative binary codes. *Pattern Anal Appl*, 6:269–84.

Hancock, P. J. B., Baddeley, R. J., and Smith, L. (1992). The principal components of natural images. *Network: Computation in Neural Systems*, 3:61 – 70.

Harris, K. D., Henze, D. A., Csicsvari, J., Hirase, H., and Buzsáki, G. (2000). Accuracy of Tetrode Spike Separation as Determined by Simultaneous Intracellular and Extracellular Measurements. *Journal of Neurophysiology*, 84(1):401–414.

Henniges, M., Puertas, G., Bornschein, J., Eggert, J., and Lücke, J. (2010). Binary sparse coding. In *Proceedings LVA/ICA*, LNCS 6365, pages 450–57. Springer.

Henniges, M., Turner, R. E., Sahani, M., Eggert, J., and Lücke, J. (2014). Efficient occlusive components analysis. *Journal of Machine Learning Research*, 15:2689–2722.

Henze, D. A., Borhegyi, Z., Csicsvari, J., Mamiya, A., Harris, K. D., and Buzsáki, G. (2000). Intracellular features predicted by extracellular recordings in the hippocampus in vivo. *Journal of neurophysiology*, 84(1):390–400.

Henze, D. A., Borhegyi, Z., Csicsvari, J., Mamiya, A., Harris, K. D., and Buzsáki, G. (2009). Simultaneous intracellular and extracellular recordings from hippocampus region ca1 of anesthetized rats. CRCNS.org.

Hinton, G. (1981). Shape representation in parallel systems. *Proceedings of the 7th international joint conference on ...*.

Hinton, G. E. and Lang, K. J. (1985). Shape recognition and illusory conjunctions. In *IJCAI*, pages 252–259.

Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Holmes, E. E. (2013). Derivation of an em algorithm for constrained and unconstrained multivariate autoregressive state-space (marss) models. *arXiv preprint arXiv:1302.3919*.

Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24.

Hoyer, P. O. (2002). Non-negative sparse coding. In *Neural Networks for Signal Processing XII: Proceedings of the IEEE Workshop on Neural Networks for Signal Processing*, pages 557–65.

Hubel, D. H. and Wiesel, T. N. (1977). Functional architecture of macaque visual cortex. *Proceedings of the Royal Society of London B*, 198:1 – 59.

Hyvärinen, A., Hoyer, P. O., Hurri, J., and Gutmann, M. (2005). Statistical models of images and early vision. In *Proceedings of the Int. Symposium on Adaptive Knowledge Representation and Reasoning (AKRR2005)*.

Hyvarinen, A., Hurri, J., and Hoyer, P. O. (2009). *Natural Image Statistics*. Springer, 1st edition.

Lauritzen, S. L. (1996). *Graphical models*, volume 17. Clarendon Press.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

LeCun, Y., Huang, F. J., and Bottou, L. (2004). Learning methods for generic object recognition with invariance to pose and lighting. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–97. IEEE.

Lee, D. D. and Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–91.

Lee, H., Battle, A., Raina, R., and Ng, A. (2007). Efficient sparse coding algorithms. In *Advances in Neural Information Processing Systems*, volume 20, pages 801–08.

Lewicki, M. S. (1998). A review of methods for spike sorting: the detection and classification of neural action potentials. *Network: Computation in Neural Systems*, 9(4):R53–R78.

Lewicki, M. S. and Sejnowski, T. J. (2000). Learning overcomplete representations. *Neural computation*, 12(2):337–365.

Lücke, J. (2007). A dynamical model for receptive field self-organization in V1 cortical columns. In *Proc. International Conference on Artificial Neural Networks*, LNCS 4669, pages 389 – 398. Springer.

Lücke, J. (2009). Receptive field self-organization in a model of the fine-structure in V1 cortical columns. *Neural Computation*, 21(10):2805–45.

Lücke, J. and Eggert, J. (2010). Expectation truncation and the benefits of preselection in training generative models. *Journal of Machine Learning Research*, 11:2855–900.

Lücke, J., Keck, C., and von der Malsburg, C. (2008). Rapid convergence to feature layer correspondences. *Neural Computation*, 20(10):2441–2463.

Lücke, J. and Sahani, M. (2008). Maximal causes for non-linear component extraction. *Journal of Machine Learning Research*, 9:1227–67.

Luttinen, J., Raiko, T., and Ilin, A. (2014). Linear state-space model with time-varying dynamics. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 338–353. Springer Berlin Heidelberg.

Mairal, J., Bach, F., Ponce, J., and Sapiro, G. (2010). Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 11.

Memisevic, R. (2011). Learning to relate images: Mapping units, complex cells and simultaneous eigenspaces. pages 1–32.

Memisevic, R. (2013). Learning to relate images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1829–1846.

Memisevic, R. and Exarchakis, G. (2013). Learning invariant features by harnessing the aperture problem.

Memisevic, R. and Hinton, G. (2007). Unsupervised Learning of Image Transformations. *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8.

Memisevic, R. and Hinton, G. E. (2010). Learning to represent spatial transformations with factored higher-order Boltzmann machines. *Neural computation*, 22(6):1473–92.

Michalski, V., Memisevic, R., and Konda, K. (2014). Modeling deep temporal dependencies with recurrent grammar cells"". In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 1925–1933. Curran Associates, Inc.

Mohamed, S., Heller, K., and Ghahramani, Z. (2010). Sparse exponential family latent variable models. NIPS Workshop.

Montesano, L., Díaz, M., Bhaskar, S., and Minguez, J. (2010). Towards an intelligent wheelchair system for users with cerebral palsy. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 18(2):193–202.

Murphy, K. P. (2012). Machine learning: a probabilistic perspective.

Neal, R. and Hinton, G. (1998). A view of the EM algorithm that justifies incremental, sparse, and other variants. In Jordan, M. I., editor, *Learning in Graphical Models*. Kluwer.

Nesterov, Y. et al. (2007). Gradient methods for minimizing composite objective function. Technical report, UCL.

Olshausen, B. and Field, D. (1996). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–9.

Olshausen, B. and Millman, K. (2000). Learning sparse codes with a mixture-of-Gaussians prior. *Advances in Neural Information Processing Systems*, 12:841–847.

Olshausen, B. A. (2000). Sparse coding of time-varying natural images. In *Proc. of the Int. Conf. on Independent Component Analysis and Blind Source Separation*, pages 603–608. Citeseer.

Olshausen, B. A., Anderson, C. H., and Essen, D. C. V. (1993). A neurobiological model of visual attention and invariant pattern recognition based on dynamic routing of information. *The Journal of Neuroscience*, 13(11):4700–4719.

Olshausen, B. A. and Field, D. J. (1997). Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research*, 37(23):3311–3325.

Pillow, J. W., Shlens, J., Chichilnisky, E., and Simoncelli, E. P. (2013). A model-based spike sorting algorithm for removing correlation artifacts in multi-neuron recordings. *PloS one*, 8(5):e62123.

Puertas, G., Bornschein, J., and Lücke, J. (2010). The maximal causes of natural scenes are edge filters. In *Advances in Neural Information Processing Systems*, volume 23, pages 1939–47.

Quiroga, R. Q., Nadasdy, Z., and Ben-Shaul, Y. (2004). Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering. *Neural Comput.*, 16(8):1661–1687.

Rauch, H. (1963). Solutions to the linear smoothing problem. *IEEE Transactions on Automatic Control*, 8(4):371–372.

Rauch, H. E., Striebel, C., and Tung, F. (1965). Maximum likelihood estimates of linear dynamic systems. *AIAA journal*, 3(8):1445–1450.

Rehn, M. and Sommer, F. T. (2007). A network that uses few active neurones to code visual input predicts the diverse shapes of cortical receptive fields. *Journal of Computational Neuroscience*, 22(2):135–46.

Rey, H. G., Pedreira, C., and Quiroga, R. Q. (2015). Past, present and future of spike sorting techniques. *Brain research bulletin*, 119:106–117.

Ringach, D. L. (2002). Spatial structure and symmetry of simple-cell receptive fields in macaque primary visual cortex. *Journal of Neurophysiology*, 88:455–63.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

Sahani, M. (1999). *Latent variable models for neural data analysis*. PhD thesis, Caltech.

Sheikh, A.-S., Shelton, J. A., and Lücke, J. (2014). A truncated em approach for spike-and-slab sparse coding. *Journal of Machine Learning Research*, 15:2653–2687.

Shelton, J. A., Bornschein, J., Sheikh, A.-S., Berkes, P., and Lücke, J. (2011). Select and sample - a model of efficient neural inference and learning. *Advances in Neural Information Processing Systems*, 24.

Shumway, R. H. and Stoffer, D. S. (1991). Dynamic linear models with switching. *Journal of the American Statistical Association*, 86(415):763–769.

Shumway, R. H. and Stoffer, D. S. (2010). *Time series analysis and its applications: with R examples*. Springer Science & Business Media.

Sparrer, S. and Fischer, R. F. (2014). Adapting compressed sensing algorithms to discrete sparse signals. In *Smart Antennas (WSA), 2014 18th International ITG Workshop on*, pages 1–8. VDE.

Tenenbaum, J. B. and Freeman, W. T. (2000). Separating Style and Content with Bilinear Models. *Neural Computation*, 12(6):1247–83.

Titsias, M. K. and Lázaro-Gredilla, M. (2011). Spike and slab variational inference for multi-task and multiple kernel learning. In *Advances in Neural Information Processing Systems*, volume 24.

Titsias, M. K. and Lázaro-Gredilla, M. (2011). Spike and slab variational inference for multi-task and multiple kernel learning. In Shawe-Taylor, J., Zemel, R. S., Bartlett, P. L., Pereira, F., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 24*, pages 2339–2347. Curran Associates, Inc.

Turner, R. E. and Sahani, M. (2011). *Bayesian Time Series Models*, chapter Two problems with variational expectation maximisation for time-series models. Cambridge University Press.

Ueda, N. and Nakano, R. (1998). Deterministic annealing EM algorithm. *Neural Networks*, 11(2):271–82.

van Hateren, J. H. and van der Schaaf, A. (1998). Independent component filters of natural images compared with simple cells in primary visual cortex. *Proceedings of the Royal Society of London B*, 265:359–66.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(Dec):3371–3408.

Wiskott, L., Fellous, J.-M., Krüger, N., and von der Malsburg, C. (1997). Face recognition by elastic bunch graph matching. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(7):775–779.

Zhou, M., Chen, H., Ren, L., Sapiro, G., Carin, L., and Paisley, J. W. (2009). Non-parametric bayesian dictionary learning for sparse image representations. In *Advances in neural information processing systems*, pages 2295–2303.

Zylberberg, J., Murphy, J. T., and Deweese, M. R. (2011). A Sparse Coding Model with Synaptically Local Plasticity and Spiking Neurons Can Account for the Diverse Shapes of V1 Simple Cell Receptive Fields. *PLoS Computational Biology*, 7(10):e1002250.