MAXIMUM LIKELIHOOD LEARNING AND INFERENCE IN CONDITIONAL RANDOM FIELDS

This version contains corrections after the official hand-in.

"Learning without thought is labor lost."
Confucius, The Confucian Analects [2].

Iulian Vlad Serban

June 8, 2012

Bachelor Thesis in Mathematics.
Department of Mathematical Sciences,
University of Copenhagen.

Bachelorprojekt i matematik.
Institut for matematiske fag,
Københavns Universitet.

Supervisor: Christian Igel
Department Of Computer Science,
University Of Copenhagen.

Abstract

The topic of this thesis is learning and inference in Conditional Random Fields. Conditional Random Fields (CRFs) are introduced and a subset of these are defined as Log-Linear Neighbourhood Models (LN Models). The analytical properties of the LN Models are investigated, and the models are connected to the latest literature in Computer Vision. Contrastive Divergence learning is then introduced and analysed in relation to the LN Models. In particular, properties regarding the convergence and bias of the Contrastive Divergence are shown based on the results derived for the Restricted Boltzmann Machines. Furthermore Graph Cut methods for exact polynomial time inference in CRFs are reviewed. The main theorem presented establishes a class of functions, for which Graph Cut methods are applicable. From the theorem a subclass of LN Models is established, for which Graph Cut methods are also applicable. Finally an LN Model for image denoising is proposed and analysed in regard to the established theoretical results. A series of empirical tests are performed on the model. These are discussed in relation to the theoretical results in the thesis, as well as in relation to the empirical results obtained for the Restricted Boltzmann Machines.

Resumé på dansk: Emnet for dette bachelorprojekt er parameter estimering og inferens i Conditional Random Fields. Conditional Random Fields (CRFs) bliver introduceret, og en underklasse af disse bliver defineret som Log-Linear Neighbourhood Models (LN Models eller LN Modeller). LN Modellernes egenskaber bliver undersøgt, og de bliver derefter knyttet til den seneste litteratur i Computer Vision. Contrastive Divergence learning bliver introduceret og analyseret med henblik på LN Modellerne. Særligt bliver konvergens og bias egenskaber for Contrastive Divergence udledt, ved brug af resultater for Restricted Boltzmann Machines. Derefter bliver Graph Cut metoder til eksakt inferens i polynomisk tid præsenteret. Hovedteoremet etablerer en klasse af funktioner, som kan maskimeres ved brug af Graph Cut methoder. Udfra dette teorem bliver en underklasse af LN Modeller etableret, hvor Graph Cut metoder kan anvendes. Endelig bliver en LN Model til billedbehandling fremlagt og implementeret i et software program. En række empiriske forsøg udføres på denne model. Resultaterne af disse diskuteres i forhold til den udviklede teori, og til empiriske resultater opnået for Restricted Boltzmann Machines.

Contents

Chapter 1

Introduction

Conditional Random Fields are statistical models, which have been proposed to solve a variety of Machine Learning problems, including image segmentation, image denoising, natural language processing and a broad class of sequential data labeling problems. They have had considerable success in many of these areas and have, unlike some of their competitors, pleasant statistical properties, see [27] [18] [15].

However, in many cases Conditional Random Fields are intractable to compute with respect to both estimation of model parameters and inference. Because they are intractable the model parameters are often found using either approximate or heuristic methods. These methods are largely justified by their empirical ability to solve a given problem, see [19] [18] [22], but that leaves both our analytical understanding and our confidence in replicating and applying them to new problems in complete darkness. The primary purpose of this thesis is therefore to shed light on the subject using a maximum likelihood approach.

The second problem posed by Conditional Random Fields is that of inference. If these models are to be of great practical value inference should be computationally fast. If we are to entrust these models with solving critical problems inference needs also to be exact. The secondary purpose of this thesis will be to present an exact and fast method for performing inference.

I will concentrate on Conditional Random Fields (CRFs) that solve so called *supervised learning problems*. These are problems where a priori we are given a class of models $\mathbb{F}$, usually defined by a probability distribution, a parameter set ω and a training data set $\{\mathbf{x}_n, \mathbf{y}_n\}_{n=1}^{N}$. The objective is then to train the model, i.e. estimate the underlying model parameters $\boldsymbol{\lambda} \in \omega$, so that it can correctly map an input sample $\mathbf{x}$ to its correct output sample $\mathbf{y}$. In mathematical terms this boils down to finding a suitable function

$F : \Phi \rightarrow \Omega$, where $\mathbf{x}_n \in \Phi \quad \mathbf{y}_n \in \Omega$, from the class of functions $\mathbb{F}$. A principled way of finding F is by using maximum likelihood. This approach is widely applied in the literature, see for example [27] [15].

I will restrict myself to discrete classification, i.e. where Ω is a discrete set. This will allow me to build on existing results, such as theorems on the convergence of Gibbs chains and theoretical articles analysing approximate learning in the closely related Restricted Boltzmann Machine models. By restricting myself to the discrete case, I will further be able to present fast and exact algorithms for inference using the so called *Graph Cut* methods. As an important note, and as a secondary motivation, Graph Cut methods form the building blocks of one of the newest learning methods for CRFs. Although these learning methods are not maximum likelihood based the models presented in this thesis should be comparable to the latest research in learning CRFs.

The thesis is structured as follows. The next chapter will present CRFs in general. It will define a subset of CRF models, named Log-Linear Neighbourhood Models (LN Models), and investigate the analytical properties of these. It will further establish important connections between the LN Models and the latest applied research in Computer Vision image models. The third chapter will introduce and analyse the highly successful *Contrastive Divergence* (CD) procedure for the LN Models. Theoretical results are presented with proofs for the most important ones. Further, a bound on the expected bias of the CD procedure is derived based on results for the Restricted Boltzmann Machines. Throughout the chapter, discussions on the implications of the results are discussed. The fourth chapter will present the previously mentioned Graph Cut methods and connect them to the LN Models. The chapter will further introduce an alternative learning procedure called *Max-Margin learning* , which uses Graph Cut methods, and relate this procedure theoretically to maximum likelihood learning and CD learning. The fifth chapter will present an empirical investigation of the CD procedure to an application of image denoising. The results of the investigation are related to the theoretical results in the preceding chapters and results obtained for the Restricted Boltzmann Machines. The sixth and final chapter will give a conclusion and some directions for further work.

Chapter 2

Conditional Random Fields

2.1 Definitions

To begin understanding the CRF I will start with the definition of a Markov Random Field (MRF).

Definition 2.1 (Markov Random Field). *A Markov Random Field is an undirected graphical model $G = (V, E)$, where V denotes the set of vertices, also referred to as nodes, and E the set of edges. Indexed on V is a set of random variables $\{Y_v\}_{v \in V}$, with $Y_v \in \gamma \ \forall v \in V$, such that*

$$Y_v \perp\!\!\!\perp Y_u | \mathbf{Y}_{N(v)} \quad \forall u \notin N(v) \cup v, \tag{2.1}$$

where $N(v)$ is the set of vertices connected to v, defined as

$$N(v) \stackrel{def}{=} \{u \in V \mid \exists (u, v) \in E\}. \tag{2.2}$$

The random variables $\mathbf{Y}_{N(v)}$ correspond to the set $N(v)$. Let $\mathbf{Y} \in \Omega \stackrel{def}{=} \gamma^{|V|}$ be the vector of random variables $\{Y_v\}_{v \in V}$, where γ is any set.

The set $N(v)$ is also refered to as the neighbours of v. Two vertices u and v are said to be neighbours if and only if $u \in N(v)$ and $v \in N(u)$. Because $N(v)$ is defined by undirected edges in the graphical model it holds that $u \in N(v)$ if and only if $v \in N(u)$.

Inspired by Daphne Koller, Nir Friedman and Hanna M. Wallach, see [15] and [27], I define a CRF as a MRF with random variables on a discrete space conditioned upon an arbitrary data vector.

Definition 2.2 (Conditional Random Field). *A Conditional Random Field, given a vector $\mathbf{x} = (x_1, \ldots, x_M) \in \Phi$, is a Markov Random Field such that the probability density function (pdf) is:*

$$P(\boldsymbol{y}|\mathbf{x}, \boldsymbol{\lambda}) = \frac{1}{Z(\mathbf{x}, \boldsymbol{\lambda})} \exp\left(\sum_{k=1}^{K} F_k(\mathbf{y}, \mathbf{x}, \boldsymbol{\lambda})\right) \tag{2.3}$$

The constant $Z(\mathbf{x}, \boldsymbol{\lambda})$ is a normalization factor often refered to as the partition function. $\mathbf{F} = (F_1, \ldots, F_K)$ is a real-valued vector function such that $F_k : \Phi \times \Omega \times \omega \to \mathbb{R}$ for each $k \in \{1, \ldots, K\}$. The vector $\boldsymbol{\lambda} \in \omega$ denotes the model parameters. The set γ is a finite set and the sets ω and Φ are any sets.

Each F_k is called a *feature function* and the vector function $\mathbf{F}$ is called the vector of *feature functions*. Typically, each F_k is chosen such that it expresses some particular feature(s) of the data which should hold for the random variables $\{Y_v\}_{v \in V}$. This is done by assigning positively higher values for more likely configurations. If a specific F_k only depends on a single Y_v, i.e. changing all other Y_u where $u \neq v$ will not change the value of F_k, it is further called a *state feature function* for v. The feature functions are central to the definition of the CRF. They allow us to embed conditioned data into the model, such that the probability distribution changes with each $\mathbf{x}$. Thus, the CRF can be seen as an extension of the MRF under certain assumptions.

The conditional independence assumption constrains each F_k. This means that the F_k's have to be chosen such that for any $v, u \in V$ where $u \neq v$ and $u \notin N(v)$ the marginal distributions satisfy

$$P(y_v, y_u | \mathbf{x}, \boldsymbol{\lambda}, \mathbf{y}_{N(v)}) = P(y_v | \mathbf{x}, \boldsymbol{\lambda}, \mathbf{y}_{N(v)}) P(y_u | \mathbf{x}, \boldsymbol{\lambda}, \mathbf{y}_{N(v)}). \tag{2.4}$$

This constraint is analytically messy to work with. Instead one can choose each F_k such that it only depends on a given node $s \in V$ and its neighbours $N(v) \in V$. This is a notational convenience and does not restrict the general class of models allowed. Suppose a vector of feature functions $\mathbf{F}$ is given with a set of random variables $\mathbf{Y}$. One can then create a graphical model as follows. First, create a node for each random variable. Then for each feature function F_k find the set of random variables that are independent of it, i.e. all $v \in V$ such that for any $\mathbf{x}$ and $\mathbf{y}_{V \setminus v}$ changing y_v will not change the value of F_k. Add edges between all the nodes outside this set. If there is no edge between two nodes $v, u \in V$, then this will imply that the feature functions will factorize according to equation (2.4). Thus, any model which can be specified as a CRF Model can also be specified as a CRF Model where each feature function depends on a single node and its neighbouring nodes. Indeed, a large group of CRF models have feature functions that are naturally formulated in terms of their node connections, which makes their network structure particularly elegant.

I further choose to introduce the parameters $\boldsymbol{\lambda}$ linearly into the exponential function. This will be very convenient from an analytical viewpoint and is similar to the log-linear CRF Model proposed by Koller and Friedman, see [15, p. 125].

Definition 2.3 (LN Model). *Let $G = (V, E)$ be a Conditional Random Field. Denote for $k \in \{1, \ldots, K\}$ by $F_k(\mathbf{y}, \mathbf{x}, s)$ the feature function k at node $s \in V$ satisfying*

$$F_k(\mathbf{y}, \mathbf{x}, s) = F_k(\hat{\mathbf{y}}, \mathbf{x}, s) \quad \forall \mathbf{y}, \hat{\mathbf{y}}, \mathbf{x}, s \text{ with } \mathbf{y}_{N(s)\cup s} = \hat{\mathbf{y}}_{N(s)\cup s}, \tag{2.5}$$

such that the probability density function is:

$$P(\boldsymbol{y}|\mathbf{x}, \boldsymbol{\lambda}) = \frac{1}{Z(\mathbf{x}, \boldsymbol{\lambda})} \exp\left(\sum_{k=1}^{K}\sum_{s=1}^{|V|} \lambda_k F_k(\mathbf{y}, \mathbf{x}, s)\right) \tag{2.6}$$

As above $Z(\mathbf{x}, \boldsymbol{\lambda})$ is the partition function. Let γ be a finite set and $\mathbf{Y} \in \Omega := \gamma^{|V|}$. The vector $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_K) \in \omega \subseteq \mathbb{R}^K$ denotes the model parameters. Let $\mathbf{x} \in \Phi \subseteq \mathbb{R}^M$. The real-valued vector function $\mathbf{F} = (F_1, \ldots, F_K)$ such that $F_k : \Omega \times \Phi \times \{1, \ldots, |V|\} \to \mathbb{R}$ for $k \in \{1, \ldots, K\}$. This is defined as a Conditional Random Field Log-Linear Neighbourhood Model (or shorter an LN Model). Further the negative of the term in the exponential function of equation (2.6) is referred to as the energy or the energy function.

Writing out the marginal probabilities for (2.6) and applying some simple algebra one can check that it indeed satisfies equation (2.4) and, therefore, also equation (2.1). This means we can visualize the independence assumptions of the model in the graphical representation. If any path between $v \in V$ and $u \in V$ passes through a set of nodes C it holds that $Y_v \perp\!\!\!\perp Y_u | Y_C$. The restriction (2.5) seems natural to many problems, for example image analysis, where one would expect a single pixel to be related mainly to its neighbouring pixels. It is interesting to note, that the energy function is often regarded as analogous to a physical system, where states with high energies change often and therefore have low probabilities.

An Example: I can illuminate the above definition with a simple example. Suppose we are given some sensory data from a physical system, such as for example measurements of rainfall and wind speed at different geographical locations. Our task is then to infer the state of the physical system, for example whether or not a storm is taking place at a certain location. Suppose for simplicity that there are only three random variables, $\{a, b, c\}$, and that these are highly correlated. Let M be an LN Model with graph nodes $\mathbf{Y} = \{Y_a, Y_b, Y_c\}$, where $Y_a, Y_b, Y_c \in \{0, 1\}$, and conditioned data vector $\mathbf{x} = \{x_a, x_b, x_c\}$, where $x_a, x_b, x_c \in \mathbb{R}$. Suppose further that all nodes are connected in the graph and that we have the following two feature functions:

$$F_1(\mathbf{y}, \mathbf{x}, s) = |x_s y_s| \tag{2.7}$$

$$F_2(\mathbf{y}, \mathbf{x}, s) = \sum_{s' \in N(s)} |y_s y_{s'}| \tag{2.8}$$

With $\lambda_1, \lambda_2 \geq 0$. This model represents a probability distribution on $\mathbf{Y}$ where a certain random variable Y_s is more likely to equal 1 if X_s is large and if the neighbouring nodes equal 1. For the physical system this would translate into having a high probability of a storm at Y_s if we have observed storms in nearby locations and if the amount of rainfall and wind speed is high. According to the the definition of the LN Model the pdf is:

$$P(\mathbf{y}|\mathbf{x}, \boldsymbol{\lambda}) = \frac{1}{Z(\mathbf{x}, \boldsymbol{\lambda})} \exp\left(\sum_{k=1}^{2} \sum_{s \in \{a,b,c\}} \lambda_k F_k(\mathbf{y}, \mathbf{x}, s)\right) \tag{2.9}$$

$$= \frac{1}{Z(\mathbf{x}, \boldsymbol{\lambda})} \exp\left(\sum_{s \in \{a,b,c\}} \left(\lambda_1 |x_s y_s| + \lambda_2 \sum_{s' \in N(s)} |y_s y_{s'}|\right)\right) \tag{2.10}$$

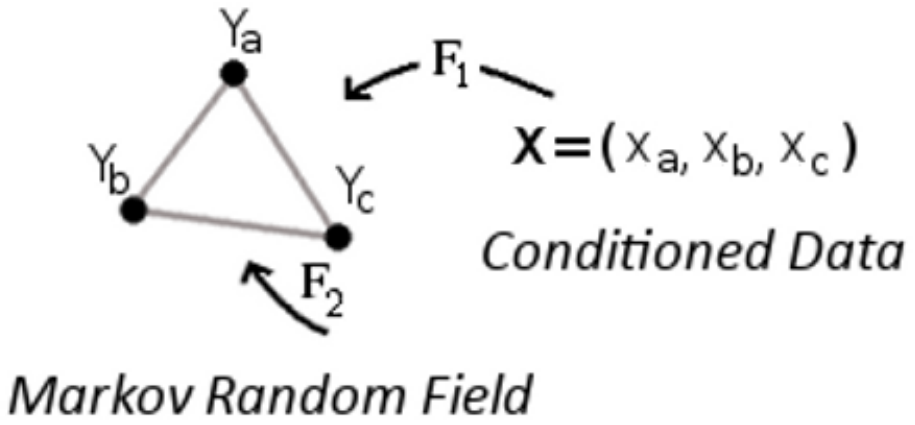

Figure 2.1: A graphical illustration of an example LN Model. The black circles represent random variables in the MRF.

As discussed earlier the model changes with x_s, yet no probability distribution has to be specified for x_s. We only have to state the relationship between the random variables and the observed data. This makes it easy to work with complicated or unknown distributions on the conditioned data and is one of the primary benefits of using CRFs. See Wallach [27, p. 2].

2.2 General Properties

I follow in the initial steps of Wallach, see [27]. Given a training data set $\{\mathbf{x}^n, \mathbf{y}^n\}_{n=1}^N$ one can write down the log-likelihood of (2.6) as

$$L(\boldsymbol{\lambda}) = \sum_{n=1}^{N} \left(\sum_{k=1}^{K} \sum_{s=1}^{|V|} \lambda_k F_k(\mathbf{y}^n, \mathbf{x}^n, s) - \log(Z(\mathbf{x}^n, \boldsymbol{\lambda})) \right). \tag{2.11}$$

By definition the partition function can be written as

$$Z(\mathbf{x}, \boldsymbol{\lambda}) = \sum_{\mathbf{y} \in \Omega} \exp \left(\sum_{k=1}^{K} \sum_{s=1}^{|V|} \lambda_k F_k(\mathbf{y}, \mathbf{x}, s) \right). \tag{2.12}$$

Differentiating this w.r.t. $\lambda_{k'}$ and dropping some indices gives

$$\frac{\partial Z(\mathbf{x}, \boldsymbol{\lambda})}{\partial \lambda_{k'}} = \sum_{\mathbf{y} \in \Omega} \exp \left(\sum_{k} \sum_{s} \lambda_k F_k(\mathbf{y}, \mathbf{x}, s) \right) \left(\sum_{s} F_{k'}(\mathbf{y}, \mathbf{x}, s) \right). \tag{2.13}$$

Now differentiating the log-likelihood w.r.t. $\lambda_{k'}$ yields

$$\begin{aligned} \frac{\partial L(\boldsymbol{\lambda})}{\partial \lambda_{k'}} =& \sum_{n} \left[\sum_{s} F_{k'}(\mathbf{y}^n, \mathbf{x}^n, s) \right. \\ & \left. - \frac{1}{Z(\mathbf{x}^n, \boldsymbol{\lambda})} \sum_{\mathbf{y} \in \Omega} \exp \left(\sum_{k,s} \lambda_k F_k(\mathbf{y}, \mathbf{x}^n, s) \right) \left(\sum_{s} F_{k'}(\mathbf{y}, \mathbf{x}^n, s) \right) \right] \\ \propto & \quad \mathrm{E}_{\hat{p}} \left[\sum_{s} F_{k'}(\mathbf{y}, \mathbf{x}, s) \right] - \frac{1}{N} \sum_{n} \mathrm{E}_{p} \left[\sum_{s} F_{k'}(\mathbf{Y}, \mathbf{x}^n, s) | \mathbf{x}^n, \boldsymbol{\lambda} \right] \\ =& \sum_{s} \mathrm{E}_{\hat{p}} \left[F_{k'}(\mathbf{Y}, \mathbf{x}, s) \right] - \frac{1}{N} \sum_{n} \sum_{s} \mathrm{E}_{p} \left[F_{k'}(\mathbf{Y}, \mathbf{x}^n, s) | \mathbf{x}^n, \boldsymbol{\lambda} \right]. \end{aligned} \tag{2.14}$$

Where $\hat{p}$ is the probability under $\{\mathbf{x}^n, \mathbf{y}^n\}_{n=1}^N$, i.e. the empirical distribution, and p is the probability of observing $\{\mathbf{y}^n\}_{n=1}^N$ given $\{\mathbf{x}^n\}_{n=1}^N$ and $\boldsymbol{\lambda}$, i.e. the probability distribution of the model. The meaning of this expression is important. For the maximum likelihood solution the equation will equal zero, and therefore the feature functions expected value w.r.t. the model are equal to the expected value w.r.t. to the empirical distribution. This is a consequence of the log-linear form the probability distribution assumes.

However, calculating this expression in a naive way will result in an algorithmic complexity of $O(N|V||\Omega| + N|V|) = O(N|V||\gamma|^{|V|} + N|V|)$. The factor $|\Omega| = |\gamma|^{|V|}$ comes from the summation over every configuration in the second term. Now take for example a single image of 32x32 pixels with 16 different states, say intensities of gray, for each pixel. This yields a complexity of $32^2 \cdot (16)^{32^2} + 32^2 \approx 10^{1236}$. By adding a new state to the model the complexity would have multiplied with a factor $32^2 = 1024$, i.e. the computation time would have been 1024 times longer. Even worse, by expanding the image with a single pixel in both width and height the complexity would have multiplied with a factor $33^2 \cdot (16)^{33^2} - 32^2 \cdot (16)^{32^2} \approx 1.8 \cdot 10^{1314}$. Thus, the computational complexity is exponential in the number of nodes and polynomial in the number of possible states. Adding more to the problem, as Wallach [27] notes, often there is no exact analytical solution to the problem and so iterative search methods (such as gradient ascent) has to be used. This would mean multiplying the previous complexity with the number of steps the algorithm requires to give a reasonable solution. In conclusion, the naive way of calculating the above expression is computationally intractable for even small models. It should, however, be noted that for certain graph structures the equation factorizes and can be calculated much faster. Wallach [27] gives an example of a chain graph with this property. From this point on, I shall assume that the graph does not factorize.

Despite its computational complexity, the LN Model still reveals some interesting statistical properties. From its log-linear form I will be able to show that the log-likelihood function is concave. This is a very important observation, because it implies that any local optimum is also a global optimum. To show that the log-likelihood is a concave function w.r.t $\boldsymbol{\lambda}$ is equivalent to showing that the Hessian matrix of $L(\boldsymbol{\lambda})$ is negative semidefinite. According to equation (2.14) this is true if the Hessian matrix of the negative

log-partition function is negative semidefinite, which it is:

$$
\begin{aligned}
&-\frac{\partial^2 \log(Z(\mathbf{x},\boldsymbol{\lambda}))}{\partial\lambda_{k'}\partial\lambda_{k''}} \\
&= \frac{1}{Z(\mathbf{x},\boldsymbol{\lambda})}\left[\frac{1}{Z(\mathbf{x},\boldsymbol{\lambda})}\sum_{\mathbf{y}\in\Omega}\exp\left(\sum_{k,s}\lambda_k F_k(\mathbf{y},\mathbf{x},s)\right)\left(\sum_s F_{k'}(\mathbf{y},\mathbf{x},s)\right)\right. \\
&\quad \exp\left(\sum_{k,s}\lambda_k F_k(\mathbf{y},\mathbf{x},s)\right)\left(\sum_s F_{k''}(\mathbf{y},\mathbf{x},s)\right) - \\
&\quad \left.\sum_{\mathbf{y}\in\Omega}\exp\left(\sum_{k,s}\lambda_k F_k(\mathbf{y},\mathbf{x},s)\right)\left(\sum_s F_{k'}(\mathbf{y},\mathbf{x},s)\right)\left(\sum_s F_{k''}(\mathbf{y},\mathbf{x},s)\right)\right] \\
&= \mathrm{E}_p\left[\sum_s F_{k'}(\mathbf{Y},\mathbf{x},s)\right]\mathrm{E}_p\left[\sum_s F_{k''}(\mathbf{Y},\mathbf{x},s)\right] \\
&\quad -\mathrm{E}_p\left[\left(\sum_s F_{k'}(\mathbf{Y},\mathbf{x},s)\right)\left(\sum_s F_{k''}(\mathbf{Y},\mathbf{x},s)\right)\right] \\
&= -\mathrm{Cov}_p\left(\sum_s F_{k'}(\mathbf{Y},\mathbf{x},s),\sum_s F_{k''}(\mathbf{Y},\mathbf{x},s)\right). \qquad (2.15)
\end{aligned}
$$

This yields the negative of the covariance matrix w.r.t. the random variables $\sum_s F_k(\mathbf{Y},\mathbf{x},s)$, $k = 1,\ldots,K$, which is negative semidefinite. Thus, maximum likelihood estimation is a convex optimization problem for the LN Model.

Another important observation is that the LN Model is an exponential family for fixed $\mathbf{x}$ and fixed feature functions. It can be written in the form of the following pdf

$$f(\mathbf{y}) = \frac{1}{c(\boldsymbol{\lambda})}\exp(\boldsymbol{\lambda}^T t(\mathbf{y})) \quad \mathbf{y}\in\Omega, \boldsymbol{\lambda}\in\omega, \qquad (2.16)$$

where

$$c(\boldsymbol{\lambda}) \stackrel{\text{def}}{=} Z(\boldsymbol{\lambda},\mathbf{x}), \qquad (2.17)$$

$$t(\mathbf{y}) \stackrel{\text{def}}{=} \left(\sum_s F_1(\mathbf{y},\mathbf{x},s), \quad \ldots, \quad \sum_s F_K(\mathbf{y},\mathbf{x},s)\right)^T. \qquad (2.18)$$

This can also be used to derive the concavity of the log-likelihood by using the positive-semidefinite covariance matrix of an exponential family, which is given by

$$\mathrm{Var}\left[t(\mathbf{Y})\right] = D^2_{\boldsymbol{\lambda}} \log c(\boldsymbol{\lambda}). \qquad (2.19)$$

See for example Nielsen [21, p. 26] for a derivation.

2.3 Generalization to Image Models in Computer Vision

I have now established the LN Model as a subset of the CRF Model. To justify it I will relate it to the latest image models in the Computer Vision literature. First, I will start with a definition of a submodel. This will allow me to present other models as submodels of the LN Model.

Definition 2.4. *For two statistical models M_1 and M_2 with respective probability density functions $P_1 : \Omega \times \Phi \times \omega_1 \to [0,1]$ and $P_2 : \Omega \times \Phi \times \omega_2 \to [0,1]$, M_1 is a submodel of M_2 if*

$$\forall \boldsymbol{\lambda}_1 \in \omega_1 \; \exists \boldsymbol{\lambda}_2 \in \omega_2 : \; P_1(\mathbf{y}|\mathbf{x}, \boldsymbol{\lambda}_1) = P_2(\mathbf{y}|\mathbf{x}, \boldsymbol{\lambda}_2) \quad \forall \mathbf{y} \in \Omega, \mathbf{x} \in \Phi, \tag{2.20}$$

where Ω is a finite set and Φ, ω_1 and ω_2 are any sets. The vectors $\boldsymbol{\lambda}_1$ and $\boldsymbol{\lambda}_2$ are the respective model parameters, and $\mathbf{Y}$ is the vector of random variables.

Lemma 2.5. *For two Conditional Random Fields M_1 and M_2 with respective probability density functions $P_1 : \Omega \times \Phi \times \omega_1 \to [0,1]$ and $P_2 : \Omega \times \Phi \times \omega_2 \to [0,1]$, M_1 is a submodel of M_2 if and only if*

$$\forall \boldsymbol{\lambda}_1 \in \omega_1 \; \exists \boldsymbol{\lambda}_2 \in \omega_2 : \; \sum_k^{K_1} F_k^1(\mathbf{y}, \mathbf{x}, \boldsymbol{\lambda}_1) = \sum_k^{K_2} F_k^2(\mathbf{y}, \mathbf{x}, \boldsymbol{\lambda}_2) \quad \forall \mathbf{y} \in \Omega, \mathbf{x} \in \Phi, \tag{2.21}$$

where

$$P_1(\boldsymbol{y}|\mathbf{x}, \boldsymbol{\lambda}_1) = \frac{1}{Z_1(\mathbf{x}, \boldsymbol{\lambda}_1)} \exp\left(\sum_k^{K_1} F_k^1(\mathbf{y}, \mathbf{x}, \boldsymbol{\lambda}_1) \right), \tag{2.22}$$

$$P_2(\boldsymbol{y}|\mathbf{x}, \boldsymbol{\lambda}_2) = \frac{1}{Z_2(\mathbf{x}, \boldsymbol{\lambda}_2)} \exp\left(\sum_k^{K_2} F_k^2(\mathbf{y}, \mathbf{x}, \boldsymbol{\lambda}_2) \right). \tag{2.23}$$

The sets $\Omega, \Phi, \omega_1, \omega_2$ and vectors $\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \mathbf{Y}$ are defined as above.

Proof. By definition of the partition function (2.21) is equivalent to (2.20). □

In other words when showing that one CRF Model is a submodel of another one only has to consider the energy functions, i.e. the terms in the exponential function of the pdf. With this observation I am ready to generalize the LN Model to other CRF models in the literature.

Binary Image Denoising: The LN Model is closely related to the CRF Model proposed by Korc and Forstner [18]. Their model has been acclaimed of having state-of-the-art performance in image denoising and considerable success in image labeling.

Definition 2.6. *The CRF Model presented by Korc and Forstner is defined by the probability density function*

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\left(\sum_{s\in V} A(y_s, \mathbf{x}) + \sum_{s\in V}\sum_{s'\in N(s)} I(y_s, y_{s'}, \mathbf{x})\right). \tag{2.24}$$

Here $\mathbf{x} \in \Phi$ defines a vector of pixel intensities, for example the brightness component of the pixel. As before $Z(\mathbf{x})$ is the partition function. The set of nodes on the graph is V, where each node represents a pixel position, with corresponding pixel intensity x_s, such that $y_s \in \{-1, 1\}\ \forall s \in V$. Let $A : \{-1, 1\} \times \Phi \to \mathbb{R}$ be the unary association potential function, depending only on the state y_s and x_s. Let $I : \{-1, 1\}^2 \times \Phi \to \mathbb{R}$ be the pairwise interaction potential function based on the two states s and s'. The potential functions are specified as

$$A(y_s, \mathbf{x}) = \sigma(y_s \mathbf{w}^T \mathbf{h}_s(\mathbf{x})) \tag{2.25}$$

$$I(y_s, y_{s'}, \mathbf{x}) = y_s y_{s'} \mathbf{v}^T \boldsymbol{\mu}_{ss'}(\mathbf{x}), \tag{2.26}$$

where σ is the non-linear sigmoid function

$$\sigma(t) = \frac{1}{1 + \exp(-t)}. \tag{2.27}$$

Here $\mathbf{h}_s$ and $\boldsymbol{\mu}_{ss'}$ are real-valued vector functions such that high values correspond to $y_s = 1$ and $y_s y_{s'} = 1$. The vectors $\mathbf{w} \in \mathbb{R}^{K_1}, \mathbf{v} \in \mathbb{R}^{K_2}$ are the model parameters, where K_1 and K_2 are the dimensions of vector functions $\mathbf{h}_s$ and $\boldsymbol{\mu}_{ss'}$. In particular, for the state-of-the-art model $\mathbf{h}_s = [1, x_s]$ and $\boldsymbol{\mu}_{ss'} = [1, |x_s - x_{s'}|]$.

In the pdf of the LN Model the parameters are multiplied with the feature functions in the energy function. This makes the LN Model an exponential family. In Korc and Forstner's CRF Model the parameters weighting each A are non-linear in a sum of the input variables. So how can it be rewritten into an LN Model? In the discrete case, when $x_s \in \{1, \ldots, C\}\ \forall s \in V$, one can define their model as a subset of the LN Model by appropriate choice of $\mathbf{F}$. One way to do it is through the indicator functions:

$$\begin{aligned} F_1(\mathbf{y}, \mathbf{x}, s) &= 1 - y_s \\ F_2(\mathbf{y}, \mathbf{x}, s) &= y_s 1_{\{x=1\}}(x_s) \\ &\ldots \\ F_{C+1}(\mathbf{y}, \mathbf{x}, s) &= y_s 1_{\{x=C\}}(x_s) \\ &\ldots \end{aligned}$$

For any choice of $\mathbf{w}$ one can choose:

$$\begin{aligned}\lambda_1 &= \sigma(0)\\ \lambda_2 &= \sigma(\mathbf{w}^T\mathbf{h}_s(1)))\\ &\dots\\ \lambda_{C+1} &= \sigma(\mathbf{w}^T\mathbf{h}_s(C)).\end{aligned}$$

It then follows that

$$\sum_{k=1}^{C+1}\lambda_k F_k(\mathbf{y},\mathbf{x},s) = \sigma(y_s\mathbf{w}^T\mathbf{h}_s(\mathbf{x})).$$

The pdfs are therefore identical, which means that their model is a submodel of the LN Model.

In the continuous case, when $\mathbf{x}\in\Phi$ and Φ is an infinite set, the non-linearity that Korc and Forstner's presents can never generally be captured by the LN Model. One way to show this is by observing that both their model and the LN Model are C^∞ functions. If Korc and Forstner's CRF Model is a submodel of the LN Model, according to Lemma 2.5, their derivatives for the energy functions should be equivalent for a certain set of parameters for the LN Model. Differentiating the energy function in the exponential function of the LN Model w.r.t. to a λ_k will always yield an expression without λ_k, while differentiating the sigmoid function will always yield a result dependent on λ_k. So in the continuous case Korc and Forstner's CRF Model is not a submodel of the LN Model.

A simple solution would be to consider a discretization of Φ, but this could lead to an enormous size of $\boldsymbol{\lambda}$, which would require an even greater number of training samples for effective estimation. Another option is to approximate their sigmoid function by a set of more general functions. One could approximate A by Taylor expanding the sigmoid function and specifying a new function F_k for each term in the expansion. Consider the particular case $\mathbf{h}_s = [1, x_s]$. If $y_s = 0$ then $A(y_s,\mathbf{x}) = \sigma(y_s\mathbf{w}^T\mathbf{h}_s(\mathbf{x})) = \sigma(0)$. If $y_s = 1$ Taylor expanding $\sigma(w_1 + w_2x)$ to the second order at $w_1 + w_2x = 0$ yields

$$\begin{aligned}&\frac{1}{2} + \frac{w_1 + w_2x}{4} - \frac{(w_1 + w_2x)^3}{48}\\ &= \frac{1}{2} + \frac{w_1}{4} + \frac{w_2x}{4} - \frac{1}{48}\left(w_1^3 + 3w_1^3w_2x + 3w_1w_2^2x^2 + w_2^3x^3\right)\end{aligned}$$

Observe that all the terms involving x can be written as a constant times x to the power of some positive integer. Select now the following $\mathbf{F}$:

$$\begin{aligned} F_1(\mathbf{y}, \mathbf{x}, s) &= 1 \\ F_2(\mathbf{y}, \mathbf{x}, s) &= y_s \\ F_3(\mathbf{y}, \mathbf{x}, s) &= y_s x_s \\ F_4(\mathbf{y}, \mathbf{x}, s) &= y_s x_s^2 \\ F_5(\mathbf{y}, \mathbf{x}, s) &= y_s x_s^3 \\ &\dots \end{aligned}$$

As before, for any choice of $\mathbf{w}$, one can choose $\boldsymbol{\lambda}$ so that the approximated Taylor expansion of their model is identical to the probability density function in the LN Model. This provides a method to approximate Korc and Forstner's model arbitrarily well. Using this approach one can indeed approximate any feature function to an arbitrary precision as long as the Taylor expansion does converge. For Korc and Forstner's model this approach is feasible in the continuous case. However, since the dimensionality of $\boldsymbol{\lambda}$ grows with the accuracy of the approximation, applying the above procedure becomes infeasible for larger models. Although it could be argued that even a limited approximation will contain enough complexity to capture the underlying model distribution.

Image Segmentation: An important and very promising application of CRF Models lies in image segmentation. Nowozin *et al.* [22] present a tree structured CRF Model for classifying objects. They use a segmentation method named *ultrametric contour maps* (UCM). Given an image it is first preprocessed by the UCM method, which will partition the image into coherent regions in a tree-structured hierarchical system. This procedure is carried out such that each child region is subset of its parent region, and such that joining all the regions, in each layer of the hierarchical system, will give back the complete image. The procedure is specified by an a priori granularity parameter, the so called *pruning edge strength*, which adjusts the number of child regions at the lowest layer in the hierarchical system.

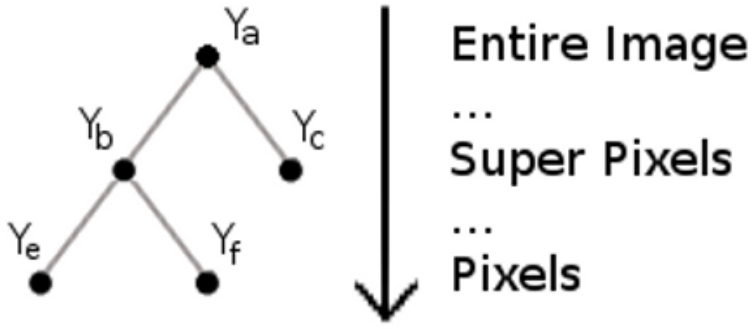

Figure 2.2: A Tree-Structured CRF using the UCM segmentation. The black circles represent random variables in the MRF.

After this process they apply a CRF Model where each region is a random variable taking values in a set of objects $\{1, \ldots, C\}$. The indices of each object represent a specific category, such as for example *background*, *person* and *bottle*. Each region is neighbour to both its parent region and its child regions as defined in the hierarchy by the UCM method. Illustration 2.2 shows the model structure. The interactions between nodes are then defined by two types of feature functions. First, the *unary observation factors*, which are state feature functions, i.e. they depend only on a single hidden random variable Y_s. They define as many as 3560 such features functions to represent textural and geometrical features of the data set. For details on these refer to *Nowozin et al.* [22]. Second, they define the *data-independent pairwise factors*, which are independent of the observed data. They are defined as fixed energy values for child-parent configurations. For a single image their model is equivalent to an LN Model with the following pdf

$$P(\mathbf{y}|\mathbf{x}, \boldsymbol{\lambda}) = \frac{1}{Z(\mathbf{x}, \boldsymbol{\lambda})} \exp\left(\sum_{s} \sum_{k=1}^{3560} \lambda_{U,k} x_{s,k} + \sum_{s} \sum_{s' \in N(s)} \left(\lambda_{T,y_s,y_{s'}} H(s, s') + \lambda_{T,y_{s'},y_s} H(s', s) \right) \right). \quad (2.28)$$

Here:

$$H(s, s') = \begin{cases} 1 & \text{if } s \text{ is parent to } s' \\ 0 & \text{otherwise} \end{cases} \quad (2.29)$$

The model parameters consists of the vector $\boldsymbol{\lambda}_U$, with 3560 entries, and the $C \times C$ matrix $\boldsymbol{\lambda}_T$. The matrix $\boldsymbol{\lambda}_T$ represents a table of energy values for child-parent node pairs, i.e. how likely a certain child-parent configuration is.

It can easily be turned into vector form to fit the definition of the LN Model. The data $\mathbf{x}$ is a matrix of size $|\mathbf{Y}| \times 3560$ with entries determined in the preprocessing stage, as described earlier. Extending the model to multiple images requires an important modification. The LN Model is specified by a fixed neighbourhood structure, but the model proposed by Nowozin *et al.* changes the neighbourhood structure with each input image according to the UCM method. The trick is then to concatenate a number of "artificial" nodes to every image preprocessed by the UCM method, so that all images have the exact same nodes and neighbourhood structure. Once this is done, one can then change equation (2.28), such that any feature function on an "artificial" node always takes the value zero. Let D denote the set of "artificial" nodes. One can write the pdf as

$$\begin{aligned} P(\mathbf{y}|\mathbf{x},\boldsymbol{\lambda}) = &\frac{1}{Z(\mathbf{x},\boldsymbol{\lambda})} \exp\Bigg(\sum_{s} \sum_{k=1}^{3560} \lambda_{U,k} x_{s,k} 1_{\{x \notin D\}}(s) \\ &+ \sum_{s} \sum_{s' \in N(s)} \Big(\lambda_{T,y_s,y_{s'}} 1_{\{x=obj_1 \wedge y=obj_2\}}(y_s, y_{s'}) H(s,s') 1_{\{x \notin D\}}(s) \\ &+ \lambda_{T,y_{s'},y_s} 1_{\{x=obj_1 \wedge y=obj_2\}}(y_{s'}, y_s) H(s',s) 1_{\{x \notin D\}}(s) \Big) \Bigg) \end{aligned} \tag{2.30}$$

After processing the model one can recursively set $y_s = y_{s'}$ for all $s \in D$ where s' is parent to s. This shows that the model proposed by Nowozin *et al.* can be defined as an LN Model with a fixed neighbourhood structure.

It should be noted that computations on their model, both both w.r.t. parameter estimation and inference, is fast and exact. However, because of the size of the model (i.e. the excessive number of feature functions, random variables and training images) it is only computationally tractable under the tree-structured hierarchical system, which ensures that the pdf factorizes. If one were to expand the model, for example by adding dependencies among adjacent image regions, exact computation in the model would become intractable. This makes the learning and inference methods presented in this thesis highly relevant to such models. There are two reasons for this. First, they will allow more complex models to fully capture the relevant dependencies. Second, they will accelerate the computations, both w.r.t. to learning and inference, which could open up for real-time applications.

The model presented by Nowozin *et al.* [22] cannot be regarded as a state-of-the-art model with regard to performance. However, the *associative hierarchical* model presented by Ladicky *et al.* [20] has been acclaimed state-of-the-art performance. Their model is an extension into a hierarchical neighbourhood system of a model proposed by Kohli, Ladicky and Torr in [14]. For the continuous case the model of Kohli *et al.* cannot be written down in the form of a LN Model, for similar reasons as stated earlier in regard to Korc and Forstner [18]. See [14, Equations (7), (10) and (12)] for the non-linear combination of parameters in the energy function. Nevertheless, their model can be approximated arbitrarily well as a submodel of a more general LN Model, due to the general form of the pdf, for the same reasons as described earlier. Furthermore, since the non-linear interactions between weight parameters involve only a small number of weights, in most cases less than 3 weights, the approximations should be very effective and accurate for even small datasets. Turning back to the CRF Model of Ladicky *et al.* , which is an extension of Kohli, Ladicky and Torr with the same pdf, it is also possible to approximate this arbitrarily well. Similar to Nowozin *et al.* , for a single fixed image their model also has a fixed neighbourhood structure. Therefore, by transforming the model using "artificial" nodes and then applying a Taylor expansion, it should be possible to approximate their model effectively. Thus, the LN Model appears to be a good candidate for effectively replicating a variety of state-of-the-art models in image segmentation, while still adhering to a number of preferential statistical properties.

I have now provided a general framework of three different methods for transforming and approximating other CRF Models:

1. Generalization through indicator functions.
2. Approximation through Taylor expansion.
3. Model transformation through "artificial" nodes.

These methods should allow a vast body of CRF models to be redefined as LN Models. Additionally, the widely applied natural language processing models proposed by Wallach [27] and Koller and Friedman [15, p. 145] are, from their definitions, submodels of the LN Model. Consequently, the theoretical results I show for the LN Model are highly relevant to state-of-the-art development in the areas of image denoising, image segmentation and natural language processing.

Chapter 3

Contrastive Divergence Learning

3.1 Definitions

Given the intractable derivative of the log-likelihood, it would seem natural to search for an approximation of the gradient and particularly the derivative of the partition function. One such approximation is the Markov Chain Monte Carlo based method called *Contrastive Divergence*(CD). A short and clear description of CD can be found in the textbook of Hastie, Tibshirani and Friedman [12]. This method is widely applied in training the MRF models named Restricted Boltzmann Machines and has had considerable success. See the work of Hinton [13]. The general idea of CD is to run a number of Gibbs chains for each training sample $(\mathbf{x}^n, \mathbf{y}^n)$ to approximate the last term of equation (2.14). Although the Gibbs chain will be far away from the actual expression, for sufficiently many steps, the sample will effectively give the correct sign of the gradient w.r.t. each λ_k, which is enough to train the LN Model. I will start with a definition of the Gibbs sampler from Bremaud [6, Chapter 7, p. 287].

Definition 3.1 (The Gibbs Sampler)**.** *For $\mathbf{x} \in \Phi$, let M be a CRF Model $G = (V, E)$ with probability distribution, i.e. probability density function, $\boldsymbol{\pi}$. The Gibbs sampler is the homogeneous discrete time Markov chain over the space Ω, with initial starting point $\mathbf{y}_1$, produced by the following transition probabilities. For each step l, according to a probability distribution $\mathbf{Z}$ on V, the chain picks a node $s \in V$ to update, and then uses the following conditional probability to update the configuration*

$$P(\mathbf{Y}_{l+1} = \mathbf{y} | \mathbf{Y}_l = \hat{\mathbf{y}}) = z_s \boldsymbol{\pi}(y_s | \hat{\mathbf{y}}_{V \setminus s}) 1_{\{\mathbf{y}_{V \setminus s} = \hat{\mathbf{y}}_{V \setminus s}\}}, \tag{3.1}$$

where $z_s > 0 \quad \forall s \in V$. This is synonymously called a Gibbs chain. After a predefined number of steps L, the current configuration $\mathbf{y}_L$ is defined to be a sample from the Gibbs chain. This is called a Gibbs sample.

One can also choose to update the nodes periodically.

Definition 3.2 (The Periodic Gibbs Sampler). *For $\mathbf{x} \in \Phi$, let M be a CRF Model $G = (V, E)$ with probability distribution $\boldsymbol{\pi}$. The periodic Gibbs sampler is the homogeneous discrete time Markov chain over the space Ω, with initial starting point $\mathbf{y}_1$, produced by the following transition probabilities. Let $q : \{1, \dots, |V|\} \to V$ be a bijective function. For each step l, the chain picks a node $s \in V$ as the next node in the sequence $q(1), \dots, q(|V|), q(1), \dots$, where $l = 1$ is the first element of the sequence. The configuration $\mathbf{y}_{l+1}$ is then changed according to the following conditional probability:*

$$P(\mathbf{Y}_{l+1} = \mathbf{y} | \mathbf{Y}_l = \hat{\mathbf{y}}) = \boldsymbol{\pi}(y_s | \hat{\mathbf{y}}_{V \setminus s}) 1_{\{\mathbf{y}_{V \setminus s} = \hat{\mathbf{y}}_{V \setminus s}\}} \tag{3.2}$$

After a predefined number of steps L, where L is a multiplum of $|V|$, the current configuration $\mathbf{y}_L$ is defined to be a sample from the Gibbs chain. The finite sequence $q(1), \dots, q(|V|)$ is called the update order of the Gibbs sampler.

In the above definition updating the nodes $q(1), \dots, q(|V|)$ corresponds to a jump in the Markov chain. Therefore, when referring to a step l in the Gibbs chain, I will take it to mean updating the nodes $q(1), \dots, q(|V|)$ in total l times in correct order. Note that the update order is synonymously refered to as a scanning policy in the literature.

The Gibbs sampler has some very nice theoretical properties. These also hold for the periodic Gibbs sampler. As Bremaud shows in [6, p. 286] convergence is guaranteed to the unique stationary distribution $\boldsymbol{\pi}$, i.e. the distribution of the CRF Model, as $L \to \infty$, if only every possible state of each node $s \in V$ has a strictly positive probability under any data set. From equation (2.3) this holds. This is, indeed, a motivating result for the CD procedure.

Definition 3.3 (L-Steps R-Samples Contrastive Divergence Learning). *Let M be an LN Model with training data set $\{\mathbf{x}^n, \mathbf{y}^n\}_{n=1}^N$. Let $\boldsymbol{\lambda}$ be initialized from an arbitrary distribution. The derivative of the log-likelihood with respect to λ_k, equation (2.14), is then approximated by:*

$$\sum_s E_{\hat{p}} \left[F_k(\mathbf{y}, \mathbf{x}, s) \right] - \frac{1}{N} \frac{1}{R} \sum_n \sum_r \sum_s F_k(\mathbf{y}^n_{L,r}, \mathbf{x}^n, s) \tag{3.3}$$

Here $\hat{p}$ is defined as in (2.14). For each $r \in \{1, \dots, R\}$ and $n \in \{1, \dots, N\}$, calculate $\mathbf{y}^n_{L,r}$ by first initializing to a point $\mathbf{y}_1$ using either 1) the respective sample $\mathbf{y}^n$ or 2) a discrete uniform distribution on Ω. Then, use a periodic Gibbs sampler to sample $\mathbf{y}^n_{1,r} \to \cdots \to \mathbf{y}^n_{L,r}$. After the sampling, estimate the λ_k's with learning parameter $\eta > 0$ using

$$\lambda_k^{t+1} = \lambda_k^t + \eta \left(\sum_s E_{\hat{p}} \left[F_k(\mathbf{y}, \mathbf{x}, s) \right] - \frac{1}{N} \frac{1}{R} \sum_n \sum_r \sum_s F_k(\mathbf{y}^n_{L,r}, \mathbf{x}^n, s) \right), \tag{3.4}$$

where $\boldsymbol{\lambda}^t$ *defines the parameters of the LN Model at iteration* t*. This process is repeated iteratively until a maximum number of iterations is reached, or the absolute change in the parameters are sufficiently small w.r.t. a given* $\epsilon > 0$ *using*

$$|\lambda_k^{t+1} - \lambda_k^t| < \epsilon \quad \forall k \in \{1, \ldots, K\}. \tag{3.5}$$

Lemma 3.4. *For the L-Steps R-Samples Contrastive Divergence Learning procedure each Gibbs chain loops through every node in a predefined order, and chooses a new state for the corresponding random variable by the following conditional pdf, for a given state* s'*, point* $\mathbf{y}$ *and sample* $\mathbf{x}$*:*

$$P(y_{s'}|\mathbf{y}_{V\setminus s'}, \mathbf{x}, \boldsymbol{\lambda}) = \frac{\exp\left(\sum_k \lambda_k F_k(s', \mathbf{y}, \mathbf{x})\right)}{\sum_{\hat{\mathbf{y}}, \forall s \neq s' : \hat{y}_s = y_s} \exp\left(\sum_k \lambda_k F_k(s', \hat{\mathbf{y}}, \mathbf{x})\right)} \tag{3.6}$$

Proof. The proof follows from writing out the probability in equation (3.2) according to equation (2.6). □

Let τ be a predefined maximum number of iterations. For each training sample the procedure runs R Gibbs chains for L Gibbs iterations and then sums over both the empirical distribution and the Gibbs samples w.r.t. the feature functions. This bounds the algorithmic complexity by $O(\tau NRL|V| + 2N|V|)$. So the calculation cost does not grow exponentially anymore. Furthermore, if $L \to \infty$ and $R \to \infty$ the Ergodic Theorem in [6, p. 111] ensures almost-surely convergence in expectation, i.e. that the average of the CD samples in equation (3.3) converges almost-surely to the last term in equation (2.14).

However, there is a notable difference between the CD procedure for the LN Model and for the Restricted Boltzmann Machines. In the last CD step for the Restricted Boltzmann Machines, the expected value is taken over the last term in equation (3.3) for the *hidden units* given the *visible units*. See Hinton [13] for a description of the Restricted Boltzmann Machines and the reasoning behind taking expectation. As communicated by Asja Fischer, this corresponds to taking expectation over the feature function of the last updated node s, given all the other nodes. For CRF Models with a large number of nodes this step is useless, in part because one is taking expectation over just a single node and in part because the dependency structure is very likely to make a single state extremely probable and all others much less probable, conditioned on the neighbouring nodes. The Binary LN Model, discussed in the fifth chapter for image denoising, is a good example of this. For large images, the majority of pixels will be surrounded by neighbours with the same states. Changing a single node within such an area of the image, for example changing a node to the state 'white' when all of its neighbours are 'black', will often lead to a very low probability. Thus, for a large number of models this last step has little effect and can be omitted.

The initialization of the Gibbs samplers in the CD procedure is important. It seems best to start the Gibbs samplers using the respective samples $(\mathbf{y}^n)_{n=1}^N$ already given. This would in fact mean the CD procedure is trying to minimize the expected difference under the empirical distribution, $\hat{p}$, and another distribution situated *somewhere* between the empirical distribution and the model distribution. As L increases this other distribution converges to the model distribution as described before. However, since the Gibbs sampler is started at a given training sample, which is the very same sample the model is supposed to maximize the likelihood over, the procedure is biased towards believing that the model, with its current parameters $\boldsymbol{\lambda}$, is much better than it actually is. This will likely result in the absolute value of the gradient being underestimated. At first sight, the bias effects the gradient in absolute terms only and not the direction of the learning. By appropriate choice of η, this bias should not pose a problem. However, a more serious bias can appear when the model becomes deterministic. Once the model has become close to deterministic, i.e. it will rarely change any node states at all because that would induce a huge reduction in the energy function, starting the model at a given training sample will pose a serious problem. The Gibbs sampler would then need many more steps to sample other configurations. As a way of coping with this problem, I introduced a second initialization method into the definition of CD, where the $\mathbf{y}_0$ is initialized from a uniform distribution on Ω. Alternatively one could also use a procedure which alternates between the two initialization methods to harvest both the benefit of fast convergence to the stationary distribution and, at least to some extend, avoid the potential bias. I will show later, that the second initialization method yields an upper bound on the expected bias of the approximation, which is determinable a priori to training the model.

There exists an interesting variant of the Contrastive Divergence procedure called *Persistent Contrastive Divergence* (Persistent CD) which uses a different initialization method for the Gibbs samplers.

Definition 3.5 (L-Steps R-Samples Persistent CD Learning)**.** *Let M be an LN Model with training data set $\{\mathbf{x}^n, \mathbf{y}^n\}_{n=1}^N$. Let t be the current iteration of the algorithm. For $t = 1$, perform learning as Contrastive Divergence in Definition 3.3. When $t > 1$, for each $r \in \{1, \ldots, R\}$ and $n \in \{1, \ldots, N\}$, initialize the Gibbs chain to the previous $\mathbf{y}_{L,r}^n$ and continue the procedure as in Definition 3.3. The procedure is repeated until a maximum number of iterations is reached or the changes in the parameters are sufficiently small as in Definition 3.3.*

Despite the nice statistical properties discussed so far, Gibbs sampling can be drastically slow at converging in expectation. It is therefore of great practical interest to try to evaluate how fast the Gibbs sampling will converge

in expectation. This will be my next step toward understanding the LN Model with CD. To do this, I will first need some probability theory for the Gibbs sampler.

3.2 Properties of The Gibbs Sampler

I will need the next two results for calculating the expected bias of the CD procedure in the coming section. The following theorem is given in Bremaud [6, Chapter 6, p. 237].

Theorem 3.6. *Let $\mathcal{P}$ be a stochastic matrix on $E \times E$ where E is a finite set. Let $\boldsymbol{\mu}$ and $\mathbf{v}$ be two probability distributions on E. Then for $l \in \mathbb{N}$*

$$d_v(\boldsymbol{\mu}^T\mathcal{P}^l, \mathbf{v}^T\mathcal{P}^l) \leq d_v(\boldsymbol{\mu}, \mathbf{v})\delta(\mathcal{P})^l. \tag{3.7}$$

Here $\delta(P)$ is Dobrushin's ergodic coefficient defined by

$$\delta(\mathcal{P}) = \sup_{i,j\in E} d_v(\mathbf{p}_i, \mathbf{p}_j), \tag{3.8}$$

where $\mathbf{p}_i$ is the i'th row in $\mathcal{P}$. The variational distance d_v is then defined for probability distributions $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ on E as

$$d_v(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{2}|\boldsymbol{\alpha} - \boldsymbol{\beta}| = \frac{1}{2}\sum_{k\in E} |\alpha_k - \beta_k| \tag{3.9}$$

Bremaud [6, Chapter 6 p. 236] also presents the next lemma.

Lemma 3.7. *For a stochastic matrix $\mathcal{P}$ on $E \times E$ where E is a finite set*

$$\delta(\mathcal{P}) = 1 - \inf_{i,j\in E} \sum_{k\in E} \min(p_{ik}, p_{jk}) \leq 1 \tag{3.10}$$

Before I dive into the statistical properties of the CD procedure, I shall need yet another result from Bremaud. Bremaud [6, Chapter 6 p. 289] gives a proof showing that the variational distance between the Gibbs sampler's distribution and its stationary distribution decreases monotonically. The theorem and proof is given below.

Theorem 3.8 (Gibbs Sampler Monotonicity)**.** *Suppose M is an LN Model trained by the Contrastive Divergence procedure, using a periodic Gibbs sampler with arbitrary initial distribution $\boldsymbol{\mu}$ over Ω. Let $\boldsymbol{\pi}$ be the stationary distribution of the Gibbs sampler. Let $\mathbf{v}_s$ be the probability distribution of the Gibbs chain after a single update at some node $s \in V$. Then:*

$$d_v(\mathbf{v}_s, \boldsymbol{\pi}) \leq d_v(\boldsymbol{\mu}, \boldsymbol{\pi}). \tag{3.11}$$

Let $q(1), \ldots, q(|V|)$ be the update order of the Gibbs sampler, where $q : \{1, \ldots, |V|\} \to V$ is a bijective function. Denote by $\mathbf{v}_{q(i)}$ the probability distribution of the Gibbs sampler after updating nodes $q(1), \ldots, q(i)$. Then:

$$d_v(\mathbf{v}_{q(|V|)}, \boldsymbol{\pi}) \leq d_v(\boldsymbol{\mu}, \boldsymbol{\pi}), \tag{3.12}$$

Proof. By definition of the Gibbs sampler:

$$\mathbf{v}_s(\mathbf{y}) = \boldsymbol{\pi}(y_s|\mathbf{y}_{V\setminus s})\boldsymbol{\mu}(\mathbf{y}_{V\setminus s}) \tag{3.13}$$

Using this observation one can derive:

$$\begin{aligned}
d_v(\mathbf{v}_s, \boldsymbol{\pi}) =& \frac{1}{2}\sum_{\mathbf{y}\in\Omega} |\boldsymbol{\pi}(y_s|\mathbf{y}_{V\setminus s})\boldsymbol{\mu}(\mathbf{y}_{V\setminus s}) - \boldsymbol{\pi}(\mathbf{y})| \\
=& \frac{1}{2}\sum_{\mathbf{y}\in\Omega} |\boldsymbol{\pi}(y_s|\mathbf{y}_{V\setminus s})\boldsymbol{\mu}(\mathbf{y}_{V\setminus s}) - \boldsymbol{\pi}(y_s|\mathbf{y}_{V\setminus s})\boldsymbol{\pi}(\mathbf{y}_{V\setminus s})| \\
=& \frac{1}{2}\sum_{\mathbf{y}\in\Omega} |\boldsymbol{\pi}(y_s|\mathbf{y}_{V\setminus s}) \left(\boldsymbol{\mu}(\mathbf{y}_{V\setminus s}) - \boldsymbol{\pi}(\mathbf{y}_{V\setminus s})\right)| \\
=& \frac{1}{2}\sum_{\mathbf{y}_{V\setminus s}}\sum_{y_s} \boldsymbol{\pi}(y_s|\mathbf{y}_{V\setminus s})|\boldsymbol{\mu}(\mathbf{y}_{V\setminus s}) - \boldsymbol{\pi}(\mathbf{y}_{V\setminus s})| \\
=& \frac{1}{2}\sum_{\mathbf{y}_{V\setminus s}} |\boldsymbol{\mu}(\mathbf{y}_{V\setminus s}) - \boldsymbol{\pi}(\mathbf{y}_{V\setminus s})| \\
=& \frac{1}{2}\sum_{\mathbf{y}_{V\setminus s}} |\sum_{y_s} (\boldsymbol{\mu}(\mathbf{y}) - \boldsymbol{\pi}(\mathbf{y}))| \leq \frac{1}{2}\sum_{\mathbf{y}\in\Omega} |\boldsymbol{\mu}(\mathbf{y}) - \boldsymbol{\pi}(\mathbf{y})|
\end{aligned} \tag{3.14}$$

Applying the previous result recursively one obtains:

$$d_v(\mathbf{v}_{q(|V|)}, \boldsymbol{\pi}) \leq d_v(\mathbf{v}_{q(|V|-1)}, \boldsymbol{\pi}) \cdots \leq d_v(\mathbf{v}_{q(1)}, \boldsymbol{\pi}) \leq d_v(\boldsymbol{\mu}, \boldsymbol{\pi}). \tag{3.15}$$

□

Corollary 3.9. *Suppose M is an LN Model trained by either the Contrastive Divergence or the Persistent Contrastive Divergence procedure. Let $\mathcal{P}$ be the transition matrix of the Gibbs samplers at iteration t. If $L \geq L_1 > L_2$, then the following holds for all steps:*

$$d_v(\mathbf{v}\mathcal{P}^{L_1}, \boldsymbol{\pi}) \leq d_v(\mathbf{v}\mathcal{P}^{L_2}, \boldsymbol{\pi}) \tag{3.16}$$

Proof. Follows directly from Theorem 3.8.

□

Using some mathematical analysis, I can derive the following lemma for the update order of the periodic Gibbs sampler.

Lemma 3.10. *Suppose M is an LN Model trained by the Contrastive Divergence procedure using a periodic Gibbs sampler, with arbitrary initial distribution $\boldsymbol{\mu}$ over Ω. Let $\boldsymbol{\pi}$ be the stationary distribution of the Gibbs sampler. Then, the update orders with lowest obtainable variational distance satisfy*

$$\arg\min_{q} \sum_{\mathbf{y}\in\Omega} |\boldsymbol{\pi}(y_{q(|V|)})|\mathbf{y}_{V\setminus q(|V|)}) \sum_{\bar{y}_{q(|V|)}} \boldsymbol{\pi}(y_{q(|V|-1)}|\mathbf{y}_{V\setminus\{q(|V|-1),q(|V|)\}}, \bar{y}_{q(|V|)}) \sum_{\bar{y}_{q(|V|-1)}} \cdots \sum_{\bar{y}_{q(2)}} \boldsymbol{\pi}(y_{q(1)}|\bar{\mathbf{y}}_{V\setminus q(1)})(\boldsymbol{\mu}(\bar{\mathbf{y}}_{V\setminus q(1)}) - \boldsymbol{\pi}(\bar{\mathbf{y}}_{V\setminus q(1)}))|, \tag{3.17}$$

where $q(1),\ldots,q(|V|)$ is the update order of the Gibbs sampler and $q:\{1,\ldots,|V|\}\to V$ is a bijective function.

Proof. Let $\mathbf{v}_{q(i)}$ be the probability distribution of the Gibbs chain after updating nodes $q(1),\ldots,q(i)$. One can derive

$$\begin{aligned}
\mathbf{v}_{q(1)}(\mathbf{y}) &= \boldsymbol{\pi}(y_{q(1)}|\mathbf{y}_{V\setminus q(1)})\boldsymbol{\mu}(\mathbf{y}_{V\setminus q(1)}) \\
\mathbf{v}_{q(2)}(\mathbf{y}) &= \boldsymbol{\pi}(y_{q(2)}|\mathbf{y}_{V\setminus q(2)}) \\
&\quad \sum_{\bar{y}_{q(2)}} \boldsymbol{\pi}(y_{q(1)}|\mathbf{y}_{V\setminus\{q(1),q(2)\}}, \bar{y}_{q(2)})\boldsymbol{\mu}(\mathbf{y}_{V\setminus\{q(1),q(2)\}}, \bar{y}_{q(2)}) \\
&\quad \cdots \\
\mathbf{v}_{q(|V|)}(\mathbf{y}) &= \boldsymbol{\pi}(y_{q(|V|)})|\mathbf{y}_{V\setminus q(|V|)}) \sum_{\bar{y}_{q(|V|)}} \boldsymbol{\pi}(y_{q(|V|-1)}|\mathbf{y}_{V\setminus\{q(|V|-1),q(|V|)\}}, \bar{y}_{q(|V|)}) \\
&\quad \sum_{\bar{y}_{q(|V|-1)}} \cdots \sum_{\bar{y}_{q(2)}} \boldsymbol{\pi}(y_{q(1)}|\bar{\mathbf{y}}_{V\setminus q(1)})\boldsymbol{\mu}(\bar{\mathbf{y}}_{V\setminus q(1)})
\end{aligned} \tag{3.18}$$

Write $\boldsymbol{\mu}(\bar{\mathbf{y}}_{V\setminus q(1)}) = \boldsymbol{\pi}(\bar{\mathbf{y}}_{V\setminus q(1)}) + \delta_{\bar{\mathbf{y}}}$. Using that $\boldsymbol{\pi}$ is the stationary distribution derive

$$\begin{aligned}
\mathbf{v}_{q(|V|)}(\mathbf{y}) &= \boldsymbol{\pi}(y_{q(|V|)})|\mathbf{y}_{V\setminus q(|V|)}) \sum_{\bar{y}_{q(|V|)}} \boldsymbol{\pi}(y_{q(|V|-1)}|\mathbf{y}_{V\setminus\{q(|V|-1),q(|V|)\}}, \bar{y}_{q(|V|)}) \\
&\quad \sum_{\bar{y}_{q(|V|-1)}} \cdots \sum_{\bar{y}_{q(2)}} \boldsymbol{\pi}(y_{q(1)}|\bar{\mathbf{y}}_{V\setminus q(1)})(\boldsymbol{\pi}(\bar{\mathbf{y}}_{V\setminus q(1)}) + \delta_{\bar{\mathbf{y}}}) \\
&= \boldsymbol{\pi}(y_{q(|V|)})|\mathbf{y}_{V\setminus q(|V|)}) \sum_{\bar{y}_{q(|V|)}} \boldsymbol{\pi}(y_{q(|V|-1)}|\mathbf{y}_{V\setminus\{q(|V|-1),q(|V|)\}}, \bar{y}_{q(|V|)}) \\
&\quad \sum_{\bar{y}_{q(|V|-1)}} \cdots \sum_{\bar{y}_{q(2)}} \boldsymbol{\pi}(y_{q(1)}|\bar{\mathbf{y}}_{V\setminus q(1)}) \cdot \delta_{\bar{\mathbf{y}}} + \boldsymbol{\pi}(\mathbf{y}) \\
&= \boldsymbol{\pi}(y_{q(|V|)})|\mathbf{y}_{V\setminus q(|V|)}) \sum_{\bar{y}_{q(|V|)}} \boldsymbol{\pi}(y_{q(|V|-1)}|\mathbf{y}_{V\setminus\{q(|V|-1),q(|V|)\}}, \bar{y}_{q(|V|)}) \\
&\quad \sum_{\bar{y}_{q(|V|-1)}} \cdots \sum_{\bar{y}_{q(2)}} \boldsymbol{\pi}(y_{q(1)}|\bar{\mathbf{y}}_{V\setminus q(1)}) \ (\boldsymbol{\mu}(\bar{\mathbf{y}}_{V\setminus q(1)}) - \boldsymbol{\pi}(\bar{\mathbf{y}}_{V\setminus q(1)})) \\
&\quad + \boldsymbol{\pi}(\mathbf{y})
\end{aligned} \tag{3.19}$$

Writing this into $d_v(\mathbf{v}_{q(|V|)}, \boldsymbol{\pi})$ and taking minimum over q finishes the proof. □

While the lemma does not provide a divine insight into picking the update order of the Gibbs sampler, it does provide a reasonable idea. From the last product term in the equation, the variational distance is likely to be small when the chain is first updated at the node where $\boldsymbol{\mu}$ and $\boldsymbol{\pi}$ are believed to disagree the most. This also makes intuitive sense, since then initialized value of this random variable will never be used. Applying the idea recursively, the second node to update using the Gibbs chain should be the node where $\boldsymbol{\mu}$ and $\boldsymbol{\pi}$ are believed to disagree the second most, and so on.

I will now use Lemma 3.7 and Bremaud [6, Chapter 7, Example 6.5] for the following theorem, which shall become very useful in the next section.

Theorem 3.11. *Suppose M is an LN Model trained by the Contrastive Divergence procedure using a periodic Gibbs sampler, with arbitrary initial distribution $\boldsymbol{\mu}$ over Ω. Let $\boldsymbol{\pi}$ be the stationary distribution of the Gibbs sampler. Let $\mathcal{P}$ be the stochastic matrix of the Gibbs chain. Then, for step $l \in \mathbb{N}$ of the Gibbs sampler,*

$$|\boldsymbol{\mu}\mathcal{P}^l - \boldsymbol{\pi}| \leq \frac{1}{2}|\boldsymbol{\mu} - \boldsymbol{\pi}|(1 - e^{-|V|\Delta})^l, \tag{3.20}$$

where:

$$\Delta = \max_{s' \in V} \delta_{s'} \tag{3.21}$$

$$\delta_{s'} = \max_{\hat{\mathbf{y}}, \check{\mathbf{y}}} \left\{ \left| \sum_k \sum_s \lambda_k F_k(\hat{\mathbf{y}}, \mathbf{x}, s) - \sum_k \sum_s \lambda_k F_k(\check{\mathbf{y}}, \mathbf{x}, s) \right| \middle| \hat{\mathbf{y}}_{V \setminus s'} = \check{\mathbf{y}}_{V \setminus s'} \right\} \tag{3.22}$$

Proof. Use Theorem 3.6 and that $\boldsymbol{\pi}$ is the stationary distribution to obtain:

$$|\boldsymbol{\mu}\mathcal{P}^l - \boldsymbol{\pi}\mathcal{P}^l| = |\boldsymbol{\mu}\mathcal{P}^l - \boldsymbol{\pi}| \leq \frac{1}{2}|\boldsymbol{\mu} - \boldsymbol{\pi}|\delta(\mathcal{P})^l \tag{3.23}$$

Denote by $p_{\hat{\mathbf{y}}\check{\mathbf{y}}}$ the entrance in $\mathcal{P}$, such that it is the probability that the Gibbs sampler will change from configuration $\hat{\mathbf{y}}$ to $\check{\mathbf{y}}$ after updating all the nodes $s \in V$. From Lemma 3.7 above one can bound $\delta(\mathcal{P})$ with:

$$\delta(\mathcal{P}) = 1 - \min_{\hat{\mathbf{y}}, \check{\mathbf{y}}} \sum_{\mathbf{y} \in \Omega} \min(p_{\hat{\mathbf{y}}\mathbf{y}}, p_{\check{\mathbf{y}}\mathbf{y}}) \leq 1 - |\gamma|^{|V|}(\min_{\hat{\mathbf{y}}, \check{\mathbf{y}}} p_{\check{\mathbf{y}}\hat{\mathbf{y}}}). \tag{3.24}$$

Now set:

$$m_s(\mathbf{y}) \stackrel{\text{def}}{=} \max_{\hat{\mathbf{y}}} \left\{ \sum_{k,s} \lambda_k F_k(\hat{\mathbf{y}}, \mathbf{x}, s) \middle| \hat{\mathbf{y}}_{V \setminus s} = \mathbf{y}_{V \setminus s} \right\}. \tag{3.25}$$

Let $p^{s'}_{\hat{\mathbf{y}}\check{\mathbf{y}}}$ be the probability that the Gibbs sampler will change from configuration $\hat{\mathbf{y}}$ to $\check{\mathbf{y}}$ when updating node s'. When updating node s' the Gibbs sampler will only change the value of $y_{s'}$ and therefore $p^{s'}_{\hat{\mathbf{y}}\check{\mathbf{y}}}$ will equal equation (3.6) for $\mathbf{y}_{V\setminus s'} = \hat{\mathbf{y}}_{V\setminus s'}$ and otherwise zero. From this one can derive:

$$\begin{aligned} p^{s'}_{\hat{\mathbf{y}}\check{\mathbf{y}}} =& \frac{\exp\left(\sum_k \lambda_k F_k(s', \check{\mathbf{y}}, \mathbf{x})\right)}{\sum_{\mathbf{y}:\ \mathbf{y}_{V\setminus s'}=\hat{\mathbf{y}}_{V\setminus s'}} \exp\left(\sum_k \lambda_k F_k(s', \mathbf{y}, \mathbf{x})\right)} \\ =& \frac{\exp\left(\sum_k \lambda_k F_k(s', \check{\mathbf{y}}, \mathbf{x}) - m_{s'}(\hat{\mathbf{y}})\right)}{\sum_{\mathbf{y}:\ \mathbf{y}_{V\setminus s'}=\hat{\mathbf{y}}_{V\setminus s'}} \exp\left(\sum_k \lambda_k F_k(s', \mathbf{y}, \mathbf{x}) - m_{s'}(\hat{\mathbf{y}})\right)} \geq \frac{e^{-\delta_{s'}}}{|\gamma|} \end{aligned} \tag{3.26}$$

The last inequality follows from the definition of δ_s and $m_s(\mathbf{y})$. This probability only applies to the update of node s' in the Gibbs chain and not to the probability for the entire step $p_{\hat{\mathbf{y}}\check{\mathbf{y}}}$. Therefore, denote the transition matrix of the Gibbs chain when updating node s by $\mathcal{P}_s$ and write

$$\mathcal{P} = \prod_s^{|V|} \mathcal{P}_s. \tag{3.27}$$

Using the previous inequality one obtains:

$$\min_{y,\hat{y}} p_{y\hat{y}} \geq \prod_s^{|V|} \frac{-e^{\delta_s}}{|\gamma|} \geq \frac{-e^{\Delta|V|}}{|\gamma|^{|V|}} \tag{3.28}$$

Using the bound on $\delta(\mathcal{P})$ from earlier one finally derives:

$$\delta(\mathcal{P}) \leq 1 - |\gamma|^{|V|} \frac{e^{-\Delta|V|}}{|\gamma|^{|V|}} = 1 - e^{-\Delta|V|} \tag{3.29}$$

□

3.3 Bounding the Expected Bias

The convergence of the Gibbs chain can be evaluated theoretically by providing an upper bound for the bias in expectation. To do this, I will use the approach of Fischer and Igel [9]. They build upon the results of Bengio and Delalleau [1], who present the following result.

Lemma 3.12. *Consider the Gibbs chain $\mathbf{y}_1 \to \mathbf{y}_2 \to \cdots$ starting at a data point $\mathbf{y}_1$. For any step l of the chain the log-pdf can be written as:*

$$\log P(\mathbf{y}_1) = \log \frac{P(\mathbf{y}_1)}{P(\mathbf{y}_l)} + \log P(\mathbf{y}_l) \tag{3.30}$$

Since this is true for every path in the chain, it can be calculated as the following expectations

$$\log P(\mathbf{y}_1) = E_{\mathbf{Y}_l}\left[\log \frac{P(\mathbf{y}_1)}{P(\mathbf{y}_l)} \middle| \mathbf{y}_1\right] + E_{\mathbf{Y}_l}[\log P(\mathbf{y}_l)|\mathbf{y}_1] \tag{3.31}$$

Where $E_{\mathbf{Y}_l}$ denotes expectation w.r.t. the variable $\mathbf{Y}_l$.

Proof. Equation (3.30) follows directly from the rules of logarithms. Equation (3.31) follows by the marginalization:

$$\log P(\mathbf{y}_1) = \sum_{\mathbf{y}_l} P(\mathbf{y}_l) \log P(\mathbf{y}_1) \tag{3.32}$$

□

Theorem 3.13. *Consider a converging Gibbs chain* $\mathbf{y}_1 \to \mathbf{y}_2 \to \cdots$ *starting at a data point* $\mathbf{y}_1$*. Equation* (3.30) *differentiated w.r.t.* λ_k *can be written as*

$$\begin{aligned}\frac{\delta}{\delta\lambda_k} \log P(\mathbf{y}_1) = &\sum_s \left(F_k(\mathbf{y}_1, \mathbf{x}, s) - E_{\mathbf{Y}_l}[F_k(\mathbf{Y}_l, \mathbf{x}, s)|\mathbf{y}_1]\right) \\ &+ \sum_s \left(E_{\mathbf{Y}_l} \left[\frac{\delta}{\delta\lambda_k} \log P(\mathbf{y}_l) \middle| \mathbf{y}_1 \right] \right),\end{aligned} \tag{3.33}$$

where the last term is the expected bias of the CD procedure, which goes to zero, as $l \to \infty$*.*

Proof. Take derivative of equation (3.30) w.r.t. λ_k

$$\begin{aligned}\frac{\delta}{\delta\lambda_k} \log P(\mathbf{y}_1) =& \frac{\delta}{\delta\lambda_k} \log \frac{P(\mathbf{y}_1)}{P(\mathbf{y}_l)} + \frac{\delta}{\delta\lambda_k} \log P(\mathbf{y}_l) \\ =& \frac{\delta}{\delta\lambda_k} \log \left(\exp\left(\sum_k \sum_s \lambda_k F_k(\mathbf{y}_1, \mathbf{x}, s) - \lambda_k F_k(\mathbf{y}_l, \mathbf{x}, s) \right) \right) \\ &+ \frac{\delta}{\delta\lambda_k} \log P(\mathbf{y}_l) \\ =& \sum_s \left(F_k(\mathbf{y}_1, \mathbf{x}, s) - F_k(\mathbf{y}_l, \mathbf{x}, s)\right) + \frac{\delta}{\delta\lambda_k} \log P(\mathbf{y}_l)\end{aligned} \tag{3.34}$$

Take expectation of this w.r.t. $\mathbf{Y}_l$ as in equation (3.31)

$$\begin{aligned}\frac{\delta}{\delta\lambda_k} \log P(\mathbf{y}_1) = &\sum_s \left(F_k(\mathbf{y}_1, \mathbf{x}, s) - E_{\mathbf{Y}_l}[F_k(\mathbf{Y}_l, \mathbf{x}, s)|\mathbf{y}_1]\right) \\ &+ E_{\mathbf{Y}_l} \left[\frac{\delta}{\delta\lambda_k} \log P(\mathbf{y}_l) \middle| \mathbf{y}_1 \right]\end{aligned} \tag{3.35}$$

From equation (2.14) one can see that the last term is the expected bias of the CD procedure. As in Bengio and Delalleau [1], one can express the convergence of the Gibbs chain through the probability

$$P(\mathbf{Y}_l = \mathbf{y}|\mathbf{y}_1) = P(\mathbf{y}) + \epsilon_l(\mathbf{y}), \tag{3.36}$$

where $\epsilon_l(\mathbf{y})$ is the error. Let $\epsilon_l \stackrel{\text{def}}{=} \max_y \epsilon_l(\mathbf{y}) \to 0$, as $t \to \infty$. This follows from the fact that the Gibbs chain converges to the stationary distribution

independent of the initial distribution, in which it was started. Now derive:

$$
\begin{aligned}
E_{\mathbf{Y}_l}\left[\frac{\delta}{\delta\lambda_k}\log P(\mathbf{Y}_l)\middle|\mathbf{y}_1\right] &= \sum_{\mathbf{y}_l} P(\mathbf{y}_l|\mathbf{y}_1)\frac{\delta}{\delta\lambda_k}\log P(\mathbf{y}_l) \\
&= \sum_{\mathbf{y}_l} P(\mathbf{y}_l)\frac{\delta}{\delta\lambda_k}\log P(\mathbf{y}_l) + \sum_{\mathbf{y}_l}\epsilon_l(\mathbf{y}_l)\frac{\delta}{\delta\lambda_k}\log P(\mathbf{y}_l) \\
&\leq \sum_{\mathbf{y}_l}|\epsilon_l(\mathbf{y}_l)|\frac{\delta}{\delta\lambda_k}\log P(\mathbf{y}_l) \leq \sum_{y}\epsilon_l\left|\frac{\delta}{\delta\lambda_k}\log P(\mathbf{y}_l)\right|
\end{aligned}
\tag{3.37}
$$

The last term goes to zero, as $t \to \infty$, which completes the proof. □

So given enough Gibbs steps, the expected bias of the CD procedure will become arbitrarily small. Consequently, a single Gibbs sampler in the CD procedure, i.e. $R = 1$, should be sufficient for approximating the log-likelihood gradient, at least in expectation.

I will use the previous result along with the following lemma to provide an upper bound on the expected bias of the CD procedure.

Lemma 3.14. *Let M be an LN Model. The following bound on the log-likelihood differentiated w.r.t. λ_k for a given data vector $\mathbf{x}$ holds:*

$$
\left|\frac{\delta}{\delta\lambda_k}\log P(\mathbf{y})\right| \leq \sum_s |F_k(\mathbf{y},\mathbf{x},s) - E_{\mathbf{Y}}[F_k(\mathbf{Y},\mathbf{x},s)]| \tag{3.38}
$$

$$
\leq \sum_s \max_{\hat{\mathbf{y}}} |F_k(\hat{\mathbf{y}},\mathbf{x},s)| \tag{3.39}
$$

Proof. Use equation (2.14) to derive

$$
\begin{aligned}
\left|\frac{\delta}{\delta\lambda_k}\log P(\mathbf{y})\right| &= \left|\sum_s F_k(\mathbf{y},\mathbf{x},s) - \sum_{s=1}^{|V|} E_{\mathbf{Y}}[F_k(\mathbf{Y},\mathbf{x},s)]\right| \\
&\leq \sum_s |F_k(\mathbf{y},\mathbf{x},s) - E_{\mathbf{Y}}[F_k(\mathbf{Y},\mathbf{x},s)]| \\
&\leq \sum_s \max_{\hat{\mathbf{y}}} |F_k(\hat{\mathbf{y}},\mathbf{x},s)|
\end{aligned}
\tag{3.40}
$$

□

Theorem 3.15 (Bound on the CD Bias). *Given an LN model $G = (V, E)$ trained by Contrastive Divergence, with a Gibbs sampler $\mathbf{y}_1 \to \mathbf{y}_2 \to \cdots$ over Ω, starting at a point $\mathbf{y}_1$. Let $\boldsymbol{\pi}$ be the stationary distribution of the Gibbs sampler and $\boldsymbol{\mu}$ the initial distribution. The expected bias of the L-Steps 1-Sample CD procedure, given in Theorem 3.13, can be bounded by, for given $\mathbf{x}$,*

$$\left| E_{\mathbf{Y}_l}\left[\frac{\delta}{\delta \lambda_k} \log P(\mathbf{y}_l) \middle| \mathbf{y}_1 \right] \right| \leq |\boldsymbol{\mu} - \boldsymbol{\pi}|(1 - e^{-|V|\Delta})^L \left(\sum_s \max_{\hat{\mathbf{y}}} |F_k(\hat{\mathbf{y}}, \mathbf{x}, s)| \right), \tag{3.41}$$

where

$$\Delta = \sup_{s' \in V} \sup_{\hat{\mathbf{y}}, \check{\mathbf{y}}} \left\{ \left| \left(\sum_k \sum_{s \in N(s')} \lambda_k \left(F_k(\hat{\mathbf{y}}, \mathbf{x}, s) - F_k(\check{\mathbf{y}}, \mathbf{x}, s) \right) \right) \right| \middle| \hat{\mathbf{y}}_{V \setminus s'} = \check{\mathbf{y}}_{V \setminus s'} \right\} \tag{3.42}$$

Proof. Use Theorem 3.13 with the bound in Lemma 3.14 to obtain

$$\begin{aligned}
&\left| E_{\mathbf{Y}_L}\left[\frac{\delta}{\delta \lambda_k} \log P(\mathbf{y}_L) \middle| \mathbf{y}_1 \right] \right| \\
&= \left| \sum_{\mathbf{y}_L} P(\mathbf{y}_L | \mathbf{y}_1) \left(\frac{\delta}{\delta \lambda_k} \log P(\mathbf{y}_L) \right) \right| \\
&= \left| \sum_{\mathbf{y}_L} (P(\mathbf{y}_L | \mathbf{y}_1) - P(\mathbf{y})) \left(\frac{\delta}{\delta \lambda_k} \log P(\mathbf{y}_L) \right) \right| \\
&\leq |\boldsymbol{\mu} - \boldsymbol{\pi}|(1 - e^{-|V|\Delta})^L \left| \frac{\delta}{\delta \lambda_k} \log P(\mathbf{y}_L) \right| \\
&\leq |\boldsymbol{\mu} - \boldsymbol{\pi}|(1 - e^{-|V|\Delta})^L \left(\sum_s \max_{\hat{\mathbf{y}}} |F_k(\hat{\mathbf{y}}, \mathbf{x}, s)| \right),
\end{aligned} \tag{3.43}$$

where

$$\begin{aligned}
\Delta &= \sup_{s' \in V} \sup_{\hat{\mathbf{y}}, \check{\mathbf{y}}} \left\{ \left| \left(\sum_k \sum_s \lambda_k F_k(\hat{\mathbf{y}}, \mathbf{x}, s) \right) \right. \right. \\
&\qquad \left. \left. - \left(\sum_k \sum_s \lambda_k F_k(\check{\mathbf{y}}, \mathbf{x}, s) \right) \right| \middle| \hat{\mathbf{y}}_{V \setminus s'} = \check{\mathbf{y}}_{V \setminus s'} \right\} \\
&= \sup_{s' \in V} \sup_{\hat{\mathbf{y}}, \check{\mathbf{y}}} \left\{ \left| \left(\sum_k \sum_{s \in N(s')}^{|V|} \lambda_k \left(F_k(\hat{\mathbf{y}}, \mathbf{x}, s) - F_k(\check{\mathbf{y}}, \mathbf{x}, s) \right) \right) \right| \middle| \hat{\mathbf{y}}_{V \setminus s'} = \check{\mathbf{y}}_{V \setminus s'} \right\}.
\end{aligned} \tag{3.44}$$

□

The theorem establishes an important relationship between the CD bias and the three distinct quantities:

- The variational distance between the initial distribution and the stationary distribution of the Gibbs chain.
- The number of Gibbs chain steps L coalesced with the maximum change in energy by changing a single node in a configuration, i.e. the largest change in energy the Gibbs chain make by updating a single node.
- The maximum value of F_k, given the observation $\mathbf{x}$.

Making either the variational distance smaller or the number of Gibbs chain steps L larger will lower the upper bound of the CD bias, and most likely also improve the performance of the CD procedure. The third quantity, however, requires a more sophisticated interpretation. For example, by simply multiplying each feature function with a very small scalar $0 < c < 1$ the parameters will grow and make the second term larger. Perhaps one way to improve the CD procedure, w.r.t. to the third quantity, is by smoothing the feature function values across the different configurations in Ω. Preferably one should avoid this, since it changes the actual model, but if there should exist unnecessary extreme values in the feature functions, one might improve the CD procedure considerably by removing them.

A Heuristic Argument I can use Theorem 3.15 to give a heuristic argument in favor of Persistent CD for a sufficiently small learning rate $\eta > 0$ and sufficiently large update steps L. Suppose M is an LN Model with training data set $\{\mathbf{x}^n, \mathbf{y}^n\}_{n=1}^N$, which has been trained until learning iteration t. Let $\mathbf{v}$ and $\boldsymbol{\mu}$ be the initial distributions for Gibbs samplers using Persistent CD and Standard CD respectively. Denote by $\mathcal{P}_t$ and $\boldsymbol{\pi}_t$ the transition matrix and the stationary distribution of the Gibbs samplers at learning iteration t. In the degenerate case where $\eta = 0$ Theorem 3.8 yields

$$d_v(\mathbf{v}, \boldsymbol{\pi}_t) = d_v(\boldsymbol{\mu}\mathcal{P}_{t-1}, \boldsymbol{\pi}_t) = d_v(\boldsymbol{\mu}\mathcal{P}_t, \boldsymbol{\pi}_t) \leq d_v(\boldsymbol{\mu}, \boldsymbol{\pi}_t). \tag{3.45}$$

For a sufficiently small $\eta > 0$ applies $\boldsymbol{\pi}_t \approx \boldsymbol{\pi}_{t-1}$ and thus $\mathcal{P}_t \approx \mathcal{P}_{t-1}$. This means that applying the Persistent CD at learning step t corresponds to running the Gibbs chain for twice as many iterations than the Standard CD. Therefore, with a high probability, one would expect the variational distance to be smaller for the Persistent CD than for the Standard CD, according to Theorem 3.8. However, in the general case it could happen that the last sample obtained by the Persistent CD had a very low probability. This could change the Persistent CD model parameters in an arbitrary direction during the parameter updating step and hence push $\boldsymbol{\mu}\mathcal{P}_t$ far away from both the stationary distribution of the Persistent CD model and the Standard CD

model. So, from a deterministic point of view the variational distance is not smaller for the Persistent CD than the Standard CD. This unfortunate case only happens with a low probability, since the Gibbs sampler will update the model parameters in the correct direction, with a high probability, for sufficiently large L. However, for the same reason it should be noted that applying the Persistent CD procedure introduces an additional stochastic element in the process. While the Standard CD will not be affected by updating the parameters in the wrong direction, the Persistent CD will use the previous samples again and therefore risks updating in the wrong direction again if L is not sufficiently high. Therefore the proposed argument only applies for sufficiently small $\eta > 0$ and sufficiently large L. Under these conditions, and from Theorem 3.15, it is presumable that the Persistent CD will have a smaller bias than the Standard CD for most learning iterations.

Lastly, I can use Theorem 3.15 to establish an exact upper bound, when the Gibbs sampler is initialized from a uniform distribution in the CD procedure.

Corollary 3.16. *Given an LN model $G = (V, E)$ and a converging Gibbs chain $\mathbf{y}_1 \to \mathbf{y}_2 \to \cdots$ on Ω starting at a point $\mathbf{y}_1$. Let $\boldsymbol{\pi}$ be the stationary distribution of the Gibbs chain and $\boldsymbol{\mu}$ the initial distribution. If $\boldsymbol{\mu}$ is the uniform distribution on Ω, the expected bias of the L-Steps 1-Sample CD procedure can be bounded by, for given $\mathbf{x}$,*

$$\begin{aligned}&\left| E_{\mathbf{Y}_l}\left[\frac{\delta}{\delta\lambda_k}\log P(\mathbf{y}_l)\middle|\mathbf{y}_1\right]\right| \\ &\leq \left(1 - \frac{1}{|\gamma|^{|V|}}\right)(1 - e^{-|V|\Delta})^L \left(\sum_{s=1}^{|V|}\max_{\mathbf{y}}|F_k(\mathbf{y}, \mathbf{x}, s)|\right),\end{aligned}$$

where Δ is as in Theorem 3.15.

Proof. Given that $\boldsymbol{\mu}$ is the uniform distribution the largest variational distance, i.e. the largest d_v as defined in equation (3.9), occurs with respect to a probability distribution $\boldsymbol{\alpha}$ where $\boldsymbol{\alpha}(\mathbf{y}) = 1$ for a single $\mathbf{y} \in \Omega$. Use this observation with Theorem 3.15

$$\begin{aligned}\frac{1}{2}\sum_{\mathbf{y}\in\Omega}|\boldsymbol{\mu}(\mathbf{y}) - \boldsymbol{\pi}(\mathbf{y})| \leq \frac{1}{2}\sum_{k\in\Omega}|\boldsymbol{\mu}(\mathbf{y}) - \boldsymbol{\alpha}(\mathbf{y})| &= \frac{1}{2}\left(\frac{|\gamma|^{|V|} - 1}{|\gamma|^{|V|}} + 1 - \frac{1}{|\gamma|^{|V|}}\right) \\ &= 1 - \frac{1}{|\gamma|^{|V|}} \qquad (3.46)\end{aligned}$$

□

Chapter 4

Graph Cuts

4.1 Basic Concepts

So far I have presented and analyzed the Contrastive Divergence procedure for estimating the model parameters. Once the model parameters are found, one would like for almost every practical purpose to use the model on unobserved data. In other words one would like to find the most likely configuration in the CRF Model given a conditioned data vector. This is the task of inference and is equivalent to finding the configuration with the lowest energy value in equation (2.6). This is where Graph Cuts come in. Not only do Graph Cuts solve the inference problem, but they also function as the main building block in an alternative learning framework, which I shall present in the final section.

Graph Cuts is a novel approach for performing exact and fast inference in MRFs. Graph Cuts are by far the most popular, and for many graph structures so far the only, method for exact inference in polynomial time [17] [5] [4]. Simple enumeration of every single configuration to find the most likely one yields the same computational complexity as calculating the partition function, while the tractable alternatives in the literature are approximate methods with a high variance in the quality of their solutions, see for example Szeliski *et al.* [24]. Since Graph Cuts are applicable to MRFs they are also applicable to CRFs for fixed conditioned data vector $\mathbf{x}$.

I will follow the mathematically rigorous and general framework established by Kolmogorov and Zabih [17]. A short but less technical introduction is given by Boykov and Veksler [4]. I will start with some basic definitions inspired by Boykov and Veksler.

Definition 4.1. *A single-source single-sink flow network is a directed graphical model $G = (V, E)$ with terminal nodes $s, t \in V$. The node s is called the source node and the node t is called the sink node. Denote an edge by*

$(u, v) \in E$ *and further define the capacity* $c(u, v) : E \to \mathbb{R}$ *and the flow* $f(u, v) : E \to \mathbb{R}$ *such that:*

$$0 \leq c(u, v), \quad (v, u) \in E \quad \forall (u, v) \in E \tag{4.1}$$

$$f(u, v) \leq c(u, v) \quad \forall (u, v) \in E \tag{4.2}$$

$$f(u, v) = -f(v, u) \quad \forall (u, v) \in E \tag{4.3}$$

$$\sum_{w \in V} f(u, w) = 0 \quad \forall u \in V, u \notin \{s, t\} \tag{4.4}$$

If $f(u, v) = c(u, v)$ *for some edge* $(u, v) \in E$ *it is called a saturated edge. The capacity function is synonymously called the weight function. If not specified, the capacity of an edge* $c(u, v)$ *equals zero.*

The first and second condition limits the flow of each edge to a certain maximum capacity. The first and third conditions insures symmetry in the flow and together with the fourth condition constrains the network such that only the source and sink nodes can send and absorb flow. I will refer to a single-source single-sink flow network as simply a flow network. The flow of the network can now be defined.

Definition 4.2. *The flow of a flow network* $G = (V, E)$ *is defined as*

$$|f| = \sum_{(s,v) \in E} f(s, v). \tag{4.5}$$

The maximum flow is further defined as $\max_f |f|$.

An integral part of Graph Cut methods is the s-t-cut.

Definition 4.3. *An* s-t*-cut for a flow network* $G = (V, E)$ *with terminal nodes* $s, t \in V$ *is a partitioning of all the nodes* V *into two disjoint subsets* S *and* T*, i.e. for all* $v \in V$ *either* $v \in S$ *or* $v \in T$*. The cost of the* s-t*-cut is defined as*

$$C(S, T) = \sum_{(u,v) \in S \times T} c(u, v). \tag{4.6}$$

An edge $(u, v) \in E$ *where* $c(u, v) > 0$ *is called a severed edge if* $u \in S$, $v \in T$.

Definition 4.4. *For a flow network* $G = (V, E)$ *with terminal nodes* $s, t \in V$ *an* s-t*-cut with partitioning* S *and* T *is called a minimum* s-t*-cut if* $C(S, T) \leq C(S', T')$ *for all other partitionings* S', T'*. The cost is synonymously called the capacity.*

It can be shown that the minimum s-t-cut corresponds to a maximum flow solution of the network. Once this is done the literature on flow networks provides a wealth of polynomial time algorithms for finding the maximum

flow. The following theorem establishes this correspondance between maximum flow and the minimum s-t cut in a flow network. It is described in Boykov and Veksler [4, p. 3], but originally due to Ford and Fulkerson [10]. Please see the last reference for the proof.

Theorem 4.5. *For a flow network $G = (V, E)$, a minimum s-t-cut value $C(S, T)$ equals the maximum flow $\max_f |f|$ of the network. Furthermore, any saturated edge $(u, v) \in E$, i.e. $f(u, v) = c(u, v)$, where f is the maximum flow, corresponds to a severed edge in the minimum s-t-cut, i.e. $u \in S$ and $v \in T$.*

4.2 Graph Cut Minimization

I can now start employing the framework of Kolmogorov and Zabih [17]. They present the following definition.

Definition 4.6. *A function $E : \{0, 1\}^n \to \mathbb{R}$ is graph-representable if there exists a flow network $G = (V, E)$ with terminal nodes $s, t \in V$ and a subset of nodes denoted $V_0 = \{v_1, \ldots, v_n\} \subseteq V \setminus \{s, t\}$ such that, for any configuration $\{y_1, \ldots, y_n\} \in \{0, 1\}^n$, the minimum of the function $E(y_1, \ldots, y_n)$ is equal to a constant plus the cost of the minimum s-t-cut subject to the partitioning $v_i \in S$ if $y_i = 0$ and $v_i \in T$ if $y_i = 1$ $\forall i \in \{1, \ldots, n\}$.*

This definition is followed by a motivating lemma.

Corollary 4.7. *Suppose the energy function E is graph-representable by a flow network $G = (V, E)$ and a subset V_0. Then, it is possible to find the exact minimum of E in polynomial time by computing the minimum s-t-cut in G.*

Proof. The proof follows from Theorem 4.5 and the existence of polynomial time algorithms for finding maximum flow. □

One way of calculating the maximum network flow in polynomial time is through the use of the relabel-to-front algorithm described in Goldberg and Tarjan [11]. Their algorithm has a worst-case complexity of $O(|V|^2|E|)$. [11, See p. 8, Theorem 3.11]. So if the energy in equation 2.6 is graph-representable one can maximize it in polynomial time.

Kolmogorov and Zabih present a class of functions which are graph-representable. It should be noted that this class of functions does not necessarily have to be defined by a graphical model. From this class of graph-representable functions, I can establish a subclass of the LN Models for which Graph Cut methods apply. To begin I need an important result given by Kolmogorov and Zabih. See [17, Appendix].

Theorem 4.8. *The sum of two graph-representable functions is also graph-representable.*

Theorem 4.9. *Let $E : \{0,1\}^n \to \mathbb{R}$ be a function on the form*

$$E(y_1, \ldots, y_n) = \sum_i E_i(y_i) + \sum_i \sum_{i<j} E_{ij}(y_i, y_j), \tag{4.7}$$

where $E_i(y_i) : \{0,1\} \to \mathbb{R}$ for $i \in \{1, \ldots, n\}$ and $E_{ij}(y_i, y_j) : \{0,1\}^2 \to \mathbb{R}$ for $i, j \in \{1, \ldots, n\}$ for $i < j$. Then, E is graph-representable if and only if each term E_{ij} satisfies the inequality

$$E_{ij}(0,0) + E_{ij}(1,1) \leq E_{ij}(0,1) + E_{ij}(1,0). \tag{4.8}$$

Proof. I will only consider the *if part* of the proof which constructively shows that functions satisfying equation (4.8) are graph-representable. See [17, p. 5] for the other direction.

Let $E : \{0,1\}^n \to \mathbb{R}$ be a function satisfying equation (4.7) and (4.8). One can construct flow networks for each variable y_i and each pair y_i, y_j and afterwards merge these flow networks into a complete flow network by adding all nodes and edges from the smaller flow networks and summing their capacities for each edge. If each of the smaller flow networks are graph-representable then their sum, i.e. the complete set of nodes and the sum of the capacities for each edge, will also be graph-representable according to Theorem 4.8. The complete flow network will have $n + 2$ vertices where n comes from the number of binary variables and 2 from the terminal nodes.

Start by constructing a flow network for each term E_i. First, add a node v_i, the source node s and the sink node t to the flow network. If $E_i(0) < E_i(1)$ add an edge (s, v_i) and set the capacity $c(s, v_i) = E_i(1) - E_i(0)$. If $E_i(0) \geq E_i(1)$ add an edge (v_i, t) and set the capacity $c(v_i, t) = E_i(0) - E_i(1)$. In both cases one can add the remaining edge (s, v_i) or (v_i, t) with capacity zero to form a connected flow network. The minimum s-t-cut is cutting the edge with zero capacity. Hence, the value of the minimum s-t-cut will yield the minimum value of E_i plus a constant. E_i is therefore graph-representable.

Next construct a flow network for each term E_{ij} where $i < j$. Add the four nodes v_i, v_j, s, t. Then observe that E_{ij} can be written as Table 1 and Table 2 in Kolmogorov and Zabih [17].

$$\begin{pmatrix} E_{ij}(0,0) & E_{ij}(0,1) \\ E_{ij}(1,0) & E_{ij}(1,1) \end{pmatrix} = \begin{pmatrix} A & B \\ C & D \end{pmatrix} =$$
$$A + \begin{pmatrix} 0 & 0 \\ C-A & C-A \end{pmatrix} + \begin{pmatrix} 0 & D-C \\ 0 & D-C \end{pmatrix} + \begin{pmatrix} 0 & B+C-A-D \\ 0 & 0 \end{pmatrix} \tag{4.9}$$

The first term assigns the same value for all combinations and can therefore by Definition 4.6 be ignored. The second and third terms depend only on y_i and y_j respectively. For these create flow networks as before for E_i: add the necessary edges and capacities between v_i, v_j and the terminal nodes including the edges with zero capacity. The last term depends on both y_i and y_j. Therefore add edge (v_i, v_j) with corresponding capacity $c(v_i, v_j) = B + C - A - D$. By equation (4.8) this is non-negative. If there are no edges with strictly positive capacities to s then the minimum s-t-cut partitions $S = \{s\}, T = \{v_i, v_j, t\}$ and similarly if there are no strictly positive edges to t and if there is no edge between v_i and v_j. In these cases the minimum s-t-cut clearly coincides with minimizing the function E_{ij}. If there are strictly positive edges to both s and t the minimum s-t-cut will sever exactly one of the three non-zero edges spawned by the matrix in equation (4.9). The edge severed will be the edge with minimum capacity which corresponds to minimizing the value of E_{ij} plus a constant. E_{ij} is therefore graph-representable. Using this procedure one can construct flow networks for all E_i and E_{ij} and then use Theorem 4.8 to sum them together. Since each flow network was graph-representable their sum will also be graph-representable. This proves that E is a graph-representable function. □

The restriction $i < j$ for the functions E_{ij} is not a restriction since one can add the value of E_{ji} to E_{ij}. However, the restriction of equation (4.8) seriously limits the choices of E_{ij}. I can now use this result to establish a submodel of the LN Model for which Graph Cut methods apply.

Definition 4.10 (LP Model)**.** *Let M be an LN Model, where $\Omega = \{0,1\}^{|V|}$ satisfying for each $k \in \{1, \ldots, K\}$, $s \in V$ and any $\mathbf{y}$, $\mathbf{x}$*

$$F_k(\mathbf{y}, \mathbf{x}, s) = F_k^s(y_s) + \sum_{s' \in N(s)} F_k^{s,s'}(y_s, y_{s'}), \tag{4.10}$$

where $F_k^s : \{0,1\} \times \Phi \times \{1, \ldots, |V|\} \to \mathbb{R}$ (4.11)

$$F_k^{s,s'} : \{0,1\}^2 \times \Phi \times \{1, \ldots, |V|\}^2 \to \mathbb{R} \tag{4.12}$$

$$F_k^{s,s'}(1,1) + F_k^{s,s'}(0,0) \geq F_k^{s,s'}(1,0) + F_k^{s,s'}(0,1) \quad \lambda_k \geq 0. \tag{4.13}$$

The terms $\mathbf{x}$, s, s' are suppressed as variables in the F_k functions. Then, M is a CRF Log-Linear Pairwise Model (or shorter an LP Model).

Lemma 4.11. *Let M be a CRF Log-Linear Pairwise Model. For any given $\mathbf{x}$, one can perform inference on M, i.e. minimize the energy function in equation (2.6), in polynomial time using Graph Cut methods.*

Proof. Let M be a CRF Log-Linear Pairwise Model. Performing inference in the LN Model is equal to maximizing (or minimizing the negative of) the term in the exponential function in equation (2.6). Therefore, let F_k^s and

$F_k^{s,s'}$ correspond to functions $-E_i$ and $-E_{ij}$ in Theorem 4.9. One can then maximize the following expression w.r.t. $\mathbf{y}$

$$\sum_k \sum_s F_k(\mathbf{y}, \mathbf{x}, s) \tag{4.14}$$

by minimizing the following expression w.r.t. $\mathbf{y}$

$$\sum_k \sum_s -F_k^s(y_s, \mathbf{x}, s) - \sum_{s' \in N(s)} F_k^{s,s'}(y_s, y_{s'}, \mathbf{x}, s, s'). \tag{4.15}$$

By Definition 4.10 the terms in last part of equation (4.15) satisfy equation (4.8). Equation (4.15) is therefore graph-representable and, thus, one can maximize equation (4.14) in polynomial time. □

Due to the definition of the LN Model, I am in fact restricting myself further than required by Theorem 4.9. The optimized function in equation (4.14) is a linear function of $\boldsymbol{\lambda}$, which is not needed in Theorem 4.9. This of course is the necessary compromise of keeping the nice statistical properties of the LN Model and still having a model where Graph Cut methods apply. The result presented so far apply to single- and pairwise terms in the exponential function. Similar results are established for terms involving three random variables by Kolmogorov and Zabih [17].

4.3 An Introduction to Max-Margin Learning

In the previous two sections I derived Graph Cut methods, which were applicable to a subset of the LN Models. These could be used for inference, which serves a great practical interest, but they also form an integral part in estimating the parameters of the LN Model using *Max-Margin learning*. Max-Margin learning has some nice statistical properties, see [25] and [7], but it also has important connections to maximum likelihood estimation and Contrastive Divergence learning. These connections will serve as my motivation for introducing the Max-Margin learning.

Suppose M is an LN Model with training data $\{\mathbf{x}^n, \mathbf{y}^n\}_{n=1}^N$. Let

$$\hat{\mathbf{y}}^n = \arg\max_{\mathbf{y} \neq \mathbf{y}^n} \sum_k \sum_s \lambda_k F_k(\mathbf{y}, \mathbf{x}^n, s). \tag{4.16}$$

Using equation (2.11) the log-likelihood for an LN Model can be written as

$$\begin{aligned} L(\boldsymbol{\lambda}) &= \sum_n \left(\sum_{k,s} \lambda_k F_k(\mathbf{y}^n, \mathbf{x}^n, s) - \log(Z(\mathbf{x}^n, \boldsymbol{\lambda})) \right) \\ &= \sum_n \left(\sum_{k,s} \lambda_k F_k(\mathbf{y}^n, \mathbf{x}^n, s) - \sum_{k,s} \lambda_k F_k(\hat{\mathbf{y}}^n, \mathbf{x}^n, s) - \Gamma(\hat{\mathbf{y}}^n, \mathbf{x}^n) \right), \end{aligned} \tag{4.17}$$

where

$$\begin{aligned} \Gamma(\hat{\mathbf{y}}^n, \mathbf{x}^n) &\stackrel{\text{def}}{=} \log \left(\sum_{\mathbf{y}} \exp \left(\sum_{k,s} \lambda_k F_k(\mathbf{y}, \mathbf{x}^n, s) \right) \right) - \sum_{k,s} \lambda_k F_k(\hat{\mathbf{y}}^n, \mathbf{x}^n, s) \\ &= \log \left(\sum_{\mathbf{y}} \exp \left(\sum_{k,s} \lambda_k F_k(\mathbf{y}, \mathbf{x}^n, s) - \lambda_k F_k(\hat{\mathbf{y}}^n, \mathbf{x}^n, s) \right) \right) \geq 0. \end{aligned} \tag{4.18}$$

Suppose there is only a single sample $\mathbf{x}^n$. Instead of the partition function in the log-likelihood above, the Max-Margin learning procedure maximizes the following expression:

$$\sum_k \sum_s \lambda_k F_k(\mathbf{y}^n, \mathbf{x}^n, s) - \sum_k \sum_s \lambda_k F_k(\hat{\mathbf{y}}^n, \mathbf{x}^n, s) \tag{4.19}$$

By definition $\hat{\mathbf{y}}^n$ is the configuration with the highest probability, different from the correct configuration $\mathbf{y}^n$, for the parameters $\boldsymbol{\lambda}$. Maximizing equation (4.19) therefore boils down to minimizing the energy of the correct configuration $\mathbf{y}^n$, while inducing as large an *energy margin* as possible to the configuration with lowest energy, different from $\mathbf{y}^n$. This is where the name Max-Margin comes from. Because the energy margin can be made arbitrarily large by multiplying $\boldsymbol{\lambda}$ with a sufficiently big scalar, it is necessary to bound it. This is done by setting the 2-norm $||\boldsymbol{\lambda}|| = \psi$, where $\psi > 0$ is a constant. The difference between the Max-Margin expression and maximum likelihood expression is then bounded by:

$$\sum_n \max_{\boldsymbol{\lambda}:||\boldsymbol{\lambda}||=\psi} \Gamma(\hat{\mathbf{y}}^n, \mathbf{x}^n) \leq \max_{\boldsymbol{\lambda}:||\boldsymbol{\lambda}||=\psi} \sum_n \Gamma(\hat{\mathbf{y}}^n, \mathbf{x}^n) \tag{4.20}$$

I can now give a suggestion as to why $\Gamma(\hat{\mathbf{y}}^n, \mathbf{x}^n)$ decreases during Max-Margin learning. Suppose an LN Model is initialized with parameters $\boldsymbol{\lambda}$, such that the probability mass is distributed almost uniformly on Ω, subject to $||\boldsymbol{\lambda}|| = \psi$. This could for example be done by choosing $\boldsymbol{\lambda}$, subject to $||\boldsymbol{\lambda}|| = \psi$, such that the variational distance between the model distribution

and the uniform distribution is as small as possible. Suppose, as the training progresses, that the model starts to assign higher probabilities to a small subset of configurations $\{\bar{\mathbf{y}}^{n,1}, \ldots, \bar{\mathbf{y}}^{n,\kappa}\}$ and very low probabilities to all other configurations. If this did not happen, i.e. the model kept a large number of high probability samples, it is likely that the model would be either a poorly trained or simply unusable for the classification problem. Supposing it did happen, and that $\hat{\mathbf{y}}^n$ had a lower energy than $\mathbf{y}^n$, then by equation (4.18) gradually $\Gamma(\hat{\mathbf{y}}^n, \mathbf{x}^n) \to \log(\kappa)$. If further κ started to decrease, meaning the probability mass became more concentrated on even fewer samples, $\Gamma(\hat{\mathbf{y}}^n, \mathbf{x}^n)$ would decrease further. Indeed, $\Gamma(\hat{\mathbf{y}}^n, \mathbf{x}^n) \to 0$ as $\kappa \to 1$ as long as $\hat{\mathbf{y}}^n$ has a higher energy than $\mathbf{y}^n$.

Ignoring the 2-norm constraint on $\boldsymbol{\lambda}$, observe that the derivative w.r.t. λ_k of equation (4.19) becomes

$$\frac{\partial}{\partial \lambda_k}\left(\sum_k \sum_s \lambda_k F_k(\mathbf{y}^n, \mathbf{x}^n, s) - \sum_k \sum_s \lambda_k F_k(\hat{\mathbf{y}}^n, \mathbf{x}^n, s)\right) = \sum_s \left(F_k(\mathbf{y}^n, \mathbf{x}^n, s) - F_k(\hat{\mathbf{y}}^n, \mathbf{x}^n, s)\right). \tag{4.21}$$

This is analogous to the derivative of the log-likelihood in equation (2.14) by replacing the last expectation with $\sum_s F_k(\hat{\mathbf{y}}^n, \mathbf{x}^n, s)$. From a maximum likelihood perspective this is a conservative assumption. Suppose the model only has a single feature function, then equation (4.21) will be an upper bound on the log-likelihood derivative w.r.t. λ_k. Indeed, replacing the log-likelihood derivative w.r.t. λ_k with the above expression is identical to the saddle point approximation proposed by Kumar *et al.* [19, p. 6].

An even stronger connection can be drawn towards Contrastive Divergence learning. Equation (4.21) is closely related to the CD approximation (3.3), for fixed n. For convenience I will restate the CD approximation:

$$\sum_{s=1}^{|V|} F_k(\mathbf{y}^n, \mathbf{x}, s) - \frac{1}{R}\sum_{r=1}^{R}\sum_{s=1}^{|V|} F_k(\mathbf{y}^n_{L,r}, \mathbf{x}^n, s) \tag{4.22}$$

While Max-Margin learning is maximizing the energy margin between the observed sample and the highest probable sample of the model, the CD procedure is maximizing the energy margin between the observed sample and a set of samples from the model. In other words both the CD procedure and the Max-Margin procedure are maximizing energy margins, but with respect to slightly different samples. Conversely, one can view equation (4.21) as a CD parameter update, where the Gibbs samples are equal to the highest probable sample.

In general, as the CD training proceeds one would expect the LN Model to assign a high probability to a few configurations and a very low probability to all others. This presumption is similar to the one given for $\Gamma(\hat{\mathbf{y}}^n, \mathbf{x}^n)$ to decrease during training, but this time applied to the CD procedure instead of the Max-Margin learning procedure. This means that the CD procedure will, with a high probability, start to sample only a few highly probable configurations. Due to the log-linear construction of the pdf in the LN Model, this will yield samples where the feature function values are very close to each other, as well as to the configuration with the highest probability. Therefore, Max-Margin learning and CD learning are not so different.

By construction, equation (4.19) is a deterministic quantity, and, therefore, there are no stochastic elements in the parameter estimation process. Contrary to CD, we do not have to worry about the exactness of the solution found by the procedure. Once it has been empirically established that the procedure works for a certain class of problems, we can have a higher degree of confidence in applying it again to other problems in this class. However, the deterministic properties are paid for in terms of the introduced complexity in finding the parameters $\boldsymbol{\lambda}$, by having to specify a priori the parameters norm, and by deciding the exact margin definition, as will be discussed shortly.

The introduction so far has focused on a fixed sample n, but clearly one will need to consider all samples and somehow weight them accordingly to find $\boldsymbol{\lambda}$. Simply summing over all these as in the CD update step does not necessarily lead to a good classification model. Some samples could have a very small set of high probability configurations, e.g. samples that are easy to classify, and others a large set of configurations with almost the same energy levels, e.g. samples that are hard to classify. This problem does not occur in maximum likelihood estimation, because the maximization attempts to minimize the energy of all configurations different from the correct configuration. For margin learning one has to explicitly differentiate between samples. This is done using so called *slack variables*. This means, however, that the previous arguments comparing Max-Margin learning to maximum likelihood estimation and CD learning should also take this into consideration.

Following Szummer *et al.* [25], I define the Max-Margin learning procedure.

Definition 4.12 (Max-Margin Learning)**.** *Let M be a CRF Log-Linear Pairwise Model with training data set $\{\mathbf{x}^n, \mathbf{y}^n\}_{n=1}^N$. Let the penalty parameter $C > 0$ be given. Let $\boldsymbol{\lambda}^t$ be the parameters at learning iteration t. Initialize $\boldsymbol{\lambda}^1$ from an arbitrary distribution. Let $S^1, \ldots, S^N \subseteq \Omega$ be empty sets. For each learning iteration t the parameters $\boldsymbol{\lambda}$ are updated according to the following procedure. For every sample n, find:*

$$\hat{\mathbf{y}}^n = \arg\max_{\mathbf{y} \neq \mathbf{y}^n} \sum_k \sum_s \lambda_k^t F_k(\mathbf{y}, \mathbf{x}^n, s) \tag{4.23}$$

If $\hat{\mathbf{y}}^n \neq \mathbf{y}^n$ add $\hat{\mathbf{y}}^n$ to the set S^n. Find $\boldsymbol{\lambda}^{t+1}$ by solving the quadratic program:

$$\begin{aligned} &\min_{\boldsymbol{\lambda}^{t+1}} \frac{1}{2}||\boldsymbol{\lambda}^{t+1}||^2 + \frac{C}{N}\sum_n \xi_n \quad s.t. \\ &\quad \lambda_k^{t+1} F_k(\mathbf{y}^n, \mathbf{x}^n, s) - \lambda_k^{t+1} F_k(\mathbf{y}, \mathbf{x}^n, s) \geq 1 - \xi_n \quad \forall \mathbf{y} \in S^n \ \forall n \\ &\quad \xi_n \geq 0 \quad \forall n \end{aligned} \tag{4.24}$$

The procedure is stopped when $\boldsymbol{\lambda}^t$ remains unchanged after solving the quadratic program. The variables ξ_n are called the slack variables.

As described earlier, the parameter norm is assumed to equal some constant. To maximize equation (4.19) one would have to consider all the configurations $\mathbf{y} \in \Omega$, since changing the parameters can change $\hat{\mathbf{y}}$. This translates into solving the quadratic program:

$$\begin{aligned} &\max_{\boldsymbol{\lambda}^{t+1}: ||\boldsymbol{\lambda}^{t+1}|| = \psi} \epsilon \quad \text{s.t.} \\ &\quad \lambda_k^{t+1} F_k(\mathbf{y}^n, \mathbf{x}^n, s) - \lambda_k^{t+1} F_k(\mathbf{y}, \mathbf{x}^n, s) \geq \epsilon \quad \forall \mathbf{y} \in \Omega \ \forall n, \end{aligned} \tag{4.25}$$

Here ψ is a constant and ϵ denotes the smallest energy margin. Setting $S^n = \Omega \ \forall n$, the quadratic program above is transformed into (4.24) by using the variable transformation $||\boldsymbol{\lambda}^{t+1}|| \leftarrow 1/\epsilon$ and by adding the slack variables ξ_n. However, in practice setting $S^n = \Omega \ \forall n$ will become computationally intractable, which is why the sets S^n are build up iteratively during the learning procedure. This is where the Graph Cut methods come in. They allow us to find $\hat{\mathbf{y}}^n$ with a sufficiently high speed to make the Max-Margin procedure possible, see Szummer *et al.*[25]. The purpose of adding the slack variables is twofold. First, they allow the energy margin to be negative, i.e. that $\lambda_k^{t+1} F_k(\mathbf{y}^n, \mathbf{x}^n, s) - \lambda_k^{t+1} F_k(\hat{\mathbf{y}}^n, \mathbf{x}^n, s)$ can be negative. Second, they allow for noise in the data. See [25, p. 7] for further details. An analytical analysis of the convergence properties is given by Finley and Joachims [7]. They show that the procedure terminates in polynomial-time, see [7, p. 4], and that one can obtain a bound on the emperical risk, which gives a certain guarantee on the quality of the solution.

Chapter 5

An Experimental Framework

5.1 The Binary LN Model

The theoretical results presented in the preceding chapters established a deeper understanding of the learning and inference in CRFs, but they have also raised a number of important questions regarding both the theoretical analysis itself and the application of it. Therefore, experimentation is needed. In this chapter I will develop a representative model. The model will be of miniature size, so that it is tractable for large-scale experiments using exact log-likelihood. I will use the empirical results obtained on this model to analyse the convergence properties of the CD procedure, to compare different CD procedures and analyse the significance of the established theoretical results. I will also relate the empirical results to others in the literature.

To proceed I am interested in an LN Model of the form illustrated in figure 5.1. I will call it the Binary LN Model.

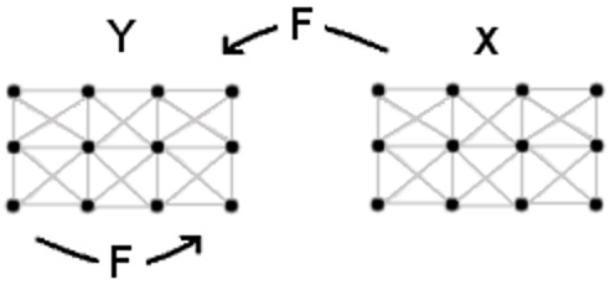

Figure 5.1: A Binary LN Model. The black circles under $\mathbf{Y}$ represent random variables in the MRF, while the black circles under $\mathbf{x}$ represent observed data.

Definition 5.1. *Let M be an LN Model $G = (V, E)$, where the nodes $V = \{S_{1,1}, \dots S_{A,B}\}$ such that $S_{a',b'} \in N(S_{a,b})$ if $|a - a'| \leq 1$ and $|b - b'| \leq 1$. Let $Y_v \in \{0, 1\}$ $\forall v \in V$. Let the conditioned data $\mathbf{x}$ be a vector with $|V|$ entries, such that each entry corresponds to a node in V. Then, define the feature functions as:*

$$F_1(\mathbf{y}, \mathbf{x}, s) = y_s \tag{5.1}$$

$$F_2(\mathbf{y}, \mathbf{x}, s) = y_s x_s \tag{5.2}$$

$$F_3(\mathbf{y}, \mathbf{x}, s) = -\frac{1}{8} \sum_{s' \in N(s)} |y_s - y_{s'}| \tag{5.3}$$

$$F_4(\mathbf{y}, \mathbf{x}, s) = -\frac{1}{8} \sum_{s' \in N(s)} |y_s - y_{s'}| \exp\left(|x_s - x_{s'}|^2\right) \tag{5.4}$$

Here x_s represents the entry in $\mathbf{x}$ corresponding to the node $s \in V$. Furthermore, restrict $\lambda_3 \geq 0, \lambda_4 \geq 0$. This model is defined as the Binary LN Model.

The model draws inspiration from both Korc and Forstner [18] and Boykov and Jolly [5, p. 5]. The following is a probabilistic justification of it. A binary image, i.e. an image consisting only of black and white pixels, of size $A \times B$ has been corrupted by noise. The noisy image is a grey-scale image. We are interested in recovering the original image, given the noisy one. Assume that only black pixels, and white pixels in their proximity, have been corrupted by noise. This would seem plausible if for example the image was transmitted as a signal through an analog circuit, with black pixels transmitted by a high current (or high voltage) and white pixels transmitted by a low current (or low voltage). Here the noise generated by the circuit will contaminate the black pixels considerably more than the white pixels. This assumption could also seem reasonable to other image denoising problems, such as medical images and space observations, where certain pixels (due to the underlying structure of the problem) will contain much more noise than others. To keep the model simple, I will suppose that the noise generated by a black pixel is *associated* with a normal distribution. Assuming that the noisy image pixels has been properly scaled, its marginal distribution will then be proportional to

$$\exp\left(|\mu - x_s|^2\right) = \exp\left(\mu^2 + x_s^2 - 2\mu x_s\right) = \exp\left(\mu^2\right) \exp\left(x_s^2\right) \exp\left(-2\mu x_s\right) \tag{5.5}$$

Setting $-2\mu = \lambda_2$ I introduce only the last term into the model as equation (5.2). This is justified by two reasons: 1) for $y_s = 1$ and observed x_s both the other two terms cancel out in the pdf, 2) Korc and Forstner [18] have applied the same feature function with considerable success. Suppose further that some general characteristics of the original image distribution is known.

Introduce the mean number of black pixels through equation (5.1). Suppose further that the original image has a certain structure, where neighbouring pixels with the same values are more likely to occur, i.e. black pixels are more likely to occur next to other black pixels. This assumption is widely applied in the literature, see [18], [5], [4] and [15, p. 112]. This structure is constructed in two ways. Firstly, as a consequence of the neighbourhood structure of the original image distribution, through equation (5.3). Secondly, by assuming that noise is *spilled over* from one pixel to its neighbouring pixels, in such a way that higher levels of noise have lower levels of probability. This is the smooth function in equation (5.4). The model comes closest to Boykov and Jolly [5].

To prepare for the experiments in the following sections, I will establish some central theory for the Binary LN Model. From the previous results, I can establish the following corollary.

Corollary 5.2. *Given a Binary LN Model $G = (V, E)$ and a converging Gibbs chain $\mathbf{y}_1 \to \mathbf{y}_2 \to \cdots$ on Ω starting at a point $\mathbf{y}_1$. Let $\boldsymbol{\pi}$ be the stationary distribution of the Gibbs chain and $\boldsymbol{\mu}$ the initial distribution. The expected bias of the L-Steps 1-Sample CD procedure, given in Theorem 3.13, can be bounded by*

$$\left| E_{\mathbf{Y}_l}\left[\frac{\delta}{\delta \lambda_k} \log P(\mathbf{y}_l) | \mathbf{y}_1 \right] \right| \leq AB|\boldsymbol{\mu} - \boldsymbol{\pi}| \left(1 - e^{-8AB(\lambda_1 + 2\lambda_2 + \lambda_3 + \lambda_4)}\right)^L. \tag{5.6}$$

If $\boldsymbol{\mu}$ is the uniform distribution on Ω, it can be bounded by

$$\left| E_{\mathbf{Y}_l}\left[\frac{\delta}{\delta \lambda_k} \log P(\mathbf{y}_l) | \mathbf{y}_1 \right] \right| \leq AB \left(1 - \frac{1}{|\gamma|^{|V|}}\right) \left(1 - e^{-8AB(\lambda_1 + 2\lambda_2 + \lambda_3 + \lambda_4)}\right)^L. \tag{5.7}$$

Proof. Since any node has a maximum of 8 neighbours and $-1 \leq x_s \leq 1$:

$$\max_{k,s,\mathbf{y},\mathbf{x}} |F_k(s, \mathbf{y}, \mathbf{x})| \leq 1 \tag{5.8}$$

$$\max_{k,s,\hat{\mathbf{y}},\check{\mathbf{y}},\mathbf{x}} |F_k(s, \hat{\mathbf{y}}, \mathbf{x}) - F_k(s, \check{\mathbf{y}}, \mathbf{x})| \leq 1 \quad k \neq 2 \tag{5.9}$$

$$\max_{s,\hat{\mathbf{y}},\check{\mathbf{y}},\mathbf{x}} |F_k(s, \hat{\mathbf{y}}, \mathbf{x}) - F_k(s, \check{\mathbf{y}}, \mathbf{x})| \leq 2 \quad k = 2 \tag{5.10}$$

Using these inequalities in the definition of Δ in Theorem 3.15 yields the first inequality. Using Corollary 3.16 yields the second inequality. □

The Binary LN Model has been constructed such that Graph Cut methods can be applied to it.

Lemma 5.3. *The Binary LN Model is a Log-Linear Pairwise Model.*

Proof. From Definition 4.10 one only need to check that the negative of each term in equations (5.4) and (5.3) satisfy equation (4.10). Since for any y_s and $y_{s'}$, $|y_s - y_s| = |y_{s'} - y_{s'}| = 0$ the equation is satisfied. □

One can construct a flow network representing the energy function of the LN Binary Model by following the proof of Theorem 4.9. For each y_s, one can represent F_1 and F_2 by constructing a flow network with y_s and the terminal nodes t, s. If $\lambda_1 \geq 0$, add an edge from y_s to t with capacity λ_1. See figure 5.2. If $\lambda_1 < 0$, add an edge from s to y_s with capacity $-\lambda_1$. Similarly for F_2. Next continue to represent flow networks for F_3 and F_4. For each pair of neighbours y_s and $y_{s'}$, construct a graph with nodes y_s, $y_{s'}$, s and t. Suppose the nodes have been ordered so that one only need consider each pair of neighbours once. Define:

$$a \stackrel{\text{def}}{=} \frac{1}{4}\left(\lambda_3 + \lambda_4 \exp\left(|x_s - x_{s'}|^2\right)\right). \tag{5.11}$$

The factor $1/4$ comes from each term $|y_s - y_{s'}|$ appearing twice in the energy function. Now use equation (4.9) to write:

$$\begin{pmatrix} 0 & 0 \\ a & a \end{pmatrix} + \begin{pmatrix} 0 & -a \\ 0 & -a \end{pmatrix} + \begin{pmatrix} 0 & 2a \\ 0 & 0 \end{pmatrix} \tag{5.12}$$

From the above expression add edges $(s, y_s), (y_{s'}, t), (y_s, y_{s'})$ with respective capacities $a, a, 2a$. See figure 5.2 for an illustration. Finally create the complete flow network by adding all the nodes, including the terminal nodes. Add the capacities between nodes such that they equal the sum across every subgraph.

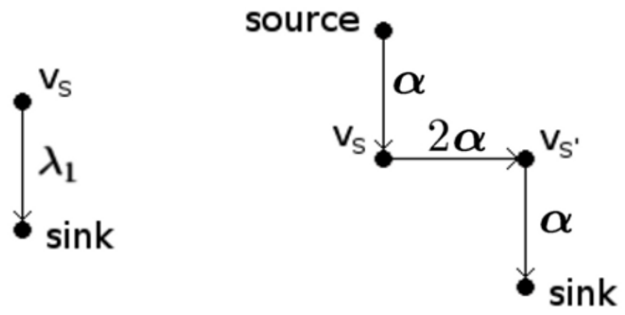

Figure 5.2: Left) Flow network representing F_1 for $\lambda_1 \geq 0$. Right) Flow network representing the terms in F_3 and F_4.

5.2 Experimental Setup

For the experiments I have developed a software program for experimentation on the LN Binary Model. The program is written in C++, using the Qt framework, and compiles under Ubuntu. Everything used in producing the program is freeware and easily accessible on the internet. The program implements the Binary LN Model with exact likelihood estimation, using gradient ascent, Contrastive Divergence learning and the above derived Graph Cut procedure for inference. The Graph Cut implementation uses the open-source max-flow algorithm of Yuri Boykov and Vladimir Kolmogorov [3] [16]. The program has been thoroughly documented. Questions regarding its implementation and performance are welcome. The reader is encouraged to run the program and review the documentation, to get a deeper understanding of the design and technical challenges underlying the experiments.

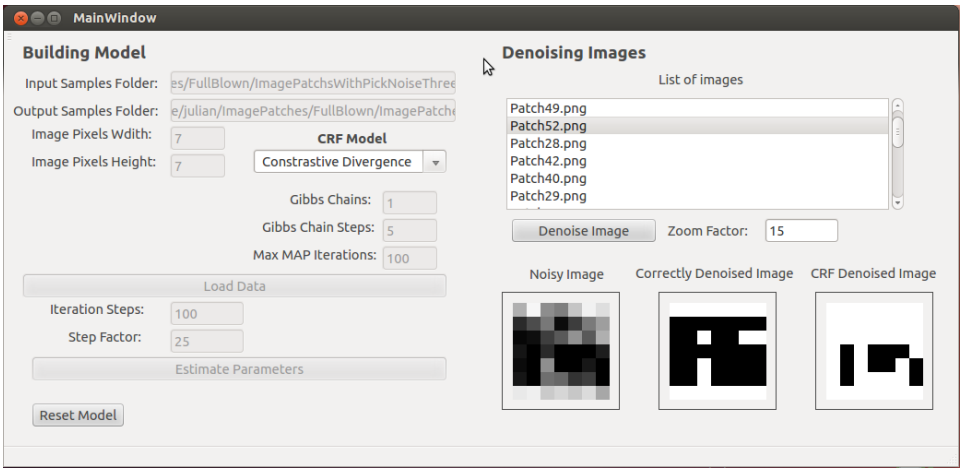

Figure 5.3: Screenshot of the software program. The Binary LN, trained with the CD procedure, is used to denoise a 7x7 image patch.

Data: The experimental data was generated from a digital photo of a building in Hong Kong. The photo was grey-scaled and down-scaled. Afterwards it was manually turned into a binary black and white image using GIMP (www.gimp.org) according to the brightness of each pixel. The last step was done in such a way as to ensure that all windows appeared black and all the bricks surrounding the windows white (from the reflection of the sun). From this black and white image 80 non-overlapping 7x7 image patches were sampled systematically from left-to-right and top-to-bottom. The image was then contaminated by grey-scaled noise using the HSV Noise filter available in GIMP. The contamination procedure was repeated three times and 80 7x7 image patches were sampled from the contaminated image. The noise

was generated to provide *a serious doubt* regarding the color of individual pixels, i.e. whether they were black or white, but to leave a constellation of pixels in the same color recognizeable to the naked human eye. Each contaminated image patch would then correspond to the conditioned data, i.e. $\mathbf{x}^n$, where each pixel intensity was rescaled to the interval $[-1, 1]$, such that $x_s = 0$ if the pixel was completely white and $x_s = 1$ if the pixel was completely black. The corresponding non-contaminated image patch would correspond to the true configuration, i.e. $\mathbf{y}^n$, where each pixel intensity took values in $\{0, 1\}$. This setup will favor pairwise interactions. Consequently, the problem should contain sufficient structure for the Binary LN Model to learn in accordance with the justification presented earlier in this chapter. See Figure 5.4.

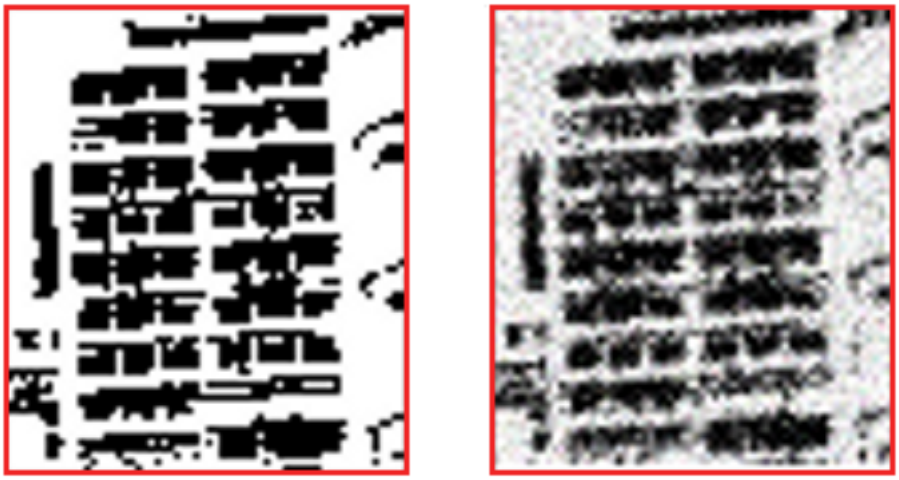

Figure 5.4: Left) Original image used. Right) Noisy image.

5.3 Experiments

In this section I will present the experimental results obtained for the Binary LN Model. Each experiment will present a hypothesis based on the theoretical results, the actual results of the experiment and a discussion of the the achieved results. It should be noted that the majority of the following experiments took several hours to perform, due to the computational complexity of estimating the parameters. This has limited the model size and the number of repeated experiments. This is taken into consideration in the following discussions, but will only affect a small subset of the conclusions. The experiments are primarily concerned with Gibbs samplers that are initialized using the training samples, since this has shown to be effective for MRF, see for example [13] and [26]. I further set $R = 1$.

Nano Model: Convergence of CD Perhaps the most fundamental experiment is to investigate whether the CD procedure converges at all, and, if it converges, then what it converges to. I will perform a total of three sub-experiments to understand the convergence properties. The first sub-experiment will be on the convergence of the parameters, w.r.t. the parameter 2-norm. The second sub-experiment will be on the uniqueness of the converged solutions, i.e. whether different CD procedures converge to the same solutions. The third sub-experiment will be on the convergence w.r.t. maximum likelihood.

Since convergence w.r.t. maximum likelihood can only be measured where the exact log-likelihood is tractable, I have to limit the experiment to 3x3 images. This is done by using only the top-left 3x3 pixels of the image patches. That is why I call it *the nano model*. Although the restriction to 3x3 images limits the neighbourhood interactions, it should still preserve some basic structure in the pixels for two reasons. Firstly, the neighbourhood structure applies from pixel to neighbouring pixel and therefore also exists in the 3x3 image model. Secondly, the conditioned data for a single pixel, i.e. x_s for y_s, provides a reasonable amount of information for each pixel, independent of the overall image size.

Because the log-likelihood gradient approximation produced by the Gibbs chain will converge in expectation to the gradient of the model and because of its empirical success for the MRF models named Restricted Boltzmann Machines, see for example [12, Chapter 17], there is reason to believe that the CD procedure will converge within a close proximity of the maximum likelihood solution. It is, however, unclear how close to the maximum likelihood solution the CD procedure will be, and how sensitive the produced solution is to the initial configuration (i.e. initial parameter values, learning rate etc.). A priori it is impossible to determine what a good Gibbs step size L would be. I therefore choose to test CD-1, i.e. Contrastive Divergence with $L = 1$, CD-5 and CD-20. For simplicity, I initialize the parameters $\boldsymbol{\lambda} = 1$. This initial setup should keep the numerical stability of the program during training. In other words, it should prevent the partition function from equaling infinity and avoid arithmetic imprecision. From some preliminary training, I found that learning rates in $[5, 25]$ seemed to give fast and reasonable results. I therefore set the learning rate $\eta = 25$ in the hope that differences between the different CD procedures will emerge clearly when the learning rate is higher.

Figure A.2, in the Appendix, shows the squared 2-norm of the parameters. Evidently, all the CD procedures converge fairly quickly. After 20 iterations it seems that the squared parameter norms have settled. It is, however,

unclear whether they have settled at the same solution or around different solutions. To perform the second sub-experiment, I will record histograms of the parameter values at iteration 100. Presumably, from the squared parameter norms, all the CD procedures should have converged at this point.

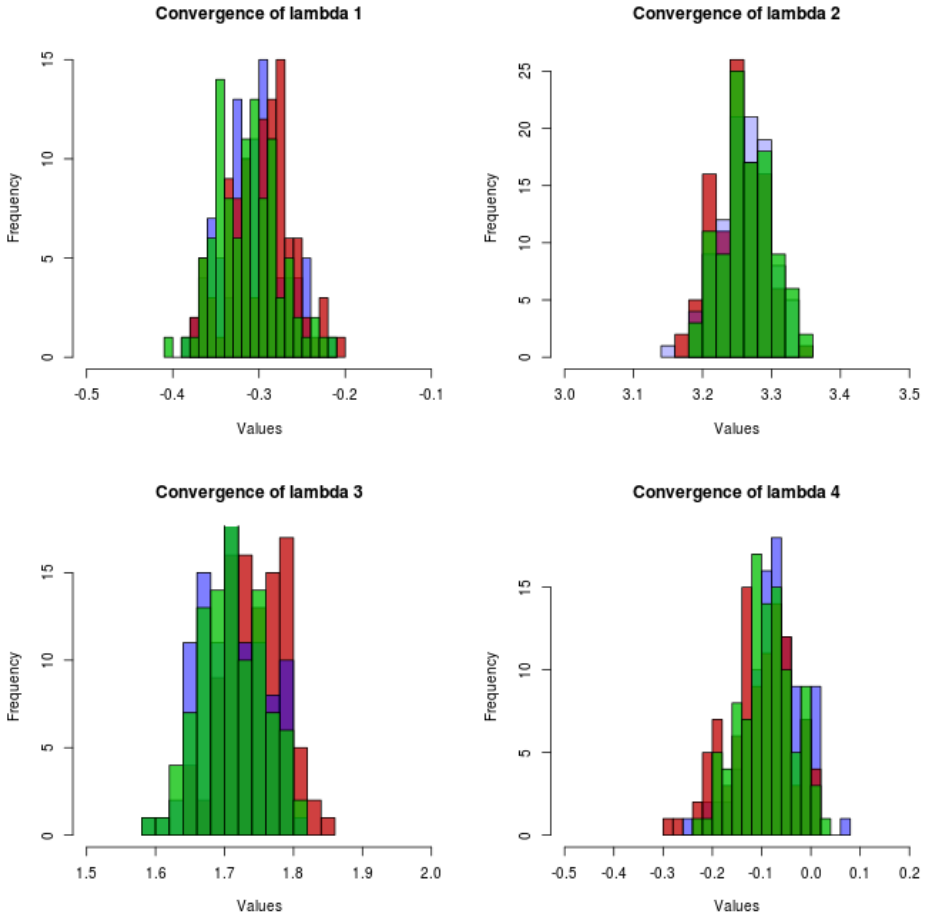

Figure 5.5: Histograms for each parameter value of the Binary LN Model, obtained by CD learning at iteration 100 for the 3x3 image patches. The colors red, blue and green correspond to respectively CD-1, CD-5 and CD-20 trained using learning rate $\eta = 25$. They were produced using 100 tests.

Figure 5.5 shows histograms for each parameter value. From these it seems clear that all the CD procedures have settled, within a close proximity, to the same parameter solution. This is also supported by the bell shaped curvature in the histograms. Consequently, this implies that the results I obtain on the convergence w.r.t. the maximum likelihood solution, will hold in general

for every CD procedure. It should also be noted, that there appears to be no difference between the CD procedures in the histograms.

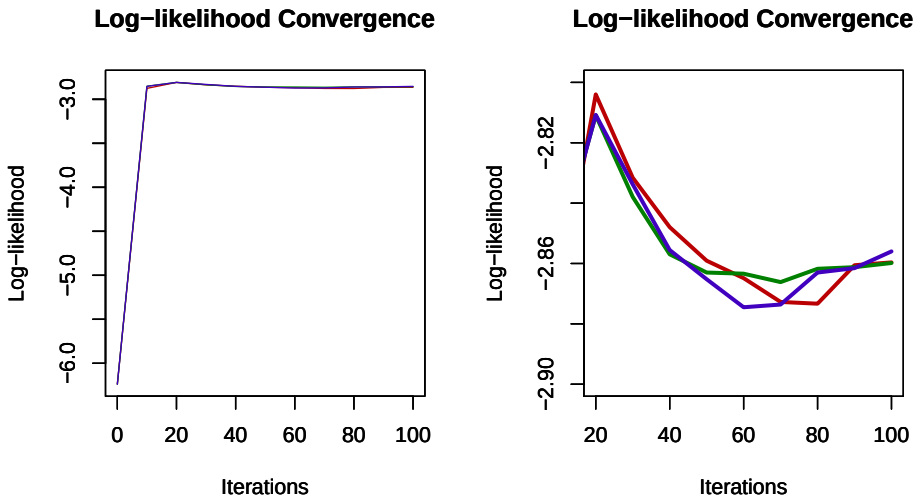

Figure 5.6: The two graphs show the log-likelihood of CD-1, CD-5 and CD-20 trained with $\eta = 25$ for the 3x3 image patches. Due to the small size of the above graphs the log-likelihood of the procedures lay on top of each other. They were produced using 50 tests measuring the log-likelihood at every 10th learning iteration. Left) Log-likelihood from learning iteration 0 to 100, Right) Log-likelihood from learning iteration 20 and onwards.

For the third sub-experiment, I used gradient ascent to find the exact maximum log-likelihood over all the samples. It is within a 0.01 decimal of -2.737. I then proceeded to analyse the log-likelihood of the CD procedures. Figure 5.6 shows the log-likelihood of each of the CD procedures, with $\eta = 25$, averaged over 50 tests. The left-side graph strongly indicates that the CD procedure converges within a close proximity of the log-likelihood. However, the right-side graph shows a disturbing effect. After the 20th learning iteration, the log-likelihood starts to fall and finally settle around the 60th iteration to a slightly worse log-likelihood than before. To investigate this divergence effect I performed an additional sub-experiment. I ran 100 tests, as in the third sub-experiment, and recorded the log-likelihood at the 20th iteration and at the 100th iteration. The result is shown in Figure 5.7 as density plots for each test. This confirms that the CD procedures diverge after the 20th iteration. It should, however, be noted that the divergence effect is rather small. This is evidenced in the substantial overlap in the density plots. Nevertheless, the divergence observation is important and will

be discussed later on. Overall from the experiment, I can conclude that the nano model is simple enough for even CD-1 to converge to a solution in close proximity to the maximum likelihood solution. This is an encouraging result for Contrastive Divergence learning.

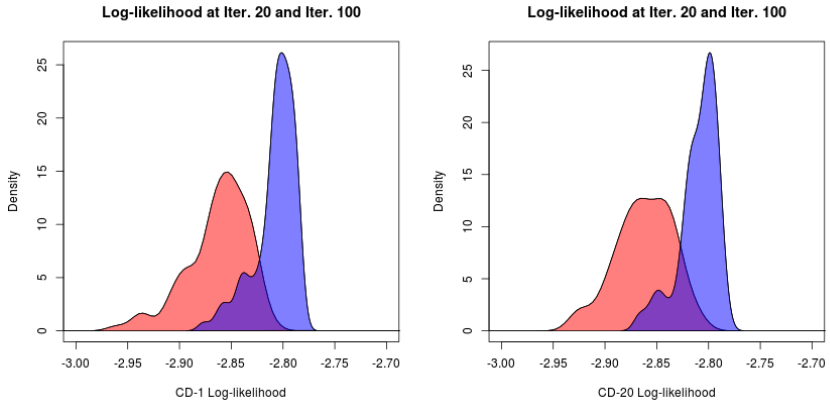

Figure 5.7: Density plots for the log-likelihood, produced by CD learning, at iteration 20 and 100 for the 3x3 image patches. The colors red and blue correspond to respectively iteration 100 and 20. They were produced using 100 tests. Left) Density plot of CD-1 log-likelihood trained using learning rate $\eta = 25$. Right) Density plot, with 512 Gaussian kernels, of CD-20 log-likelihood trained using learning rate $\eta = 25$.

Binary LN Model: Convergence of CD So far, I have experimented on a nano version of the Binary LN Model. I will now proceed to experiment on the full-scale Binary LN Model, i.e. the Binary LN Model with 7x7 image patches. As before, it is here important to analyse the convergence properties of the CD procedures. However, now the computational complexity has increased drastically. For the nano model a single test, measuring the log-likelihood until iteration 100 for every 10 iterations, would take about 2 minutes. Performing the same test on the full-scale Binary LN Model will take at least $2.2 \cdot 10^7$ years, assuming that the computer has sufficient memory and power to carry out the task. Thus, I cannot measure the exact log-likelihood. I will therefore only perform two sub-experiments to investigate the general convergence. The first sub-experiment will analyse the changes of the parameters, i.e. the squared parameter norms, as before. The second sub-experiment will analyse the uniqueness of the convergence, i.e. the parameter solutions, as before. I choose to test CD-1 and CD-20 in the first sub-experiment. I set the learning rate $\eta = 25$ and the same setup as in the last experiment. I initialize $\boldsymbol{\lambda} = \mathbf{1}$. I run 100 tests for 100 iterations and record the squared parameter norm for each iteration.

Figure A.2, in the Appendix, shows the squared 2-norm parameters. It indicates that all the CD procedures converge fairly quickly. It therefore seems reasonable to assume that all the CD procedures have converged after 100 iterations with $\eta = 25$. The second sub-experiment will be to analyse the parameter solutions at iteration 100. This will confirm whether or not the CD procedure has converged to the same distributions. It should also give some information on the solution quality, and, for the later experiments, some material to compare CD-1, CD-5 and CD-20. I run 100 tests using the same setup as before, but with CD-1 trained using both $\eta = 25$ and $\eta = 10$ and CD-20 trained using $\eta = 10$. Testing CD-1 with two different learning rates will give me some basic understanding of how the learning rate effects the final parameter solution. One would expect the lowest learning rate to have the most stable solution.

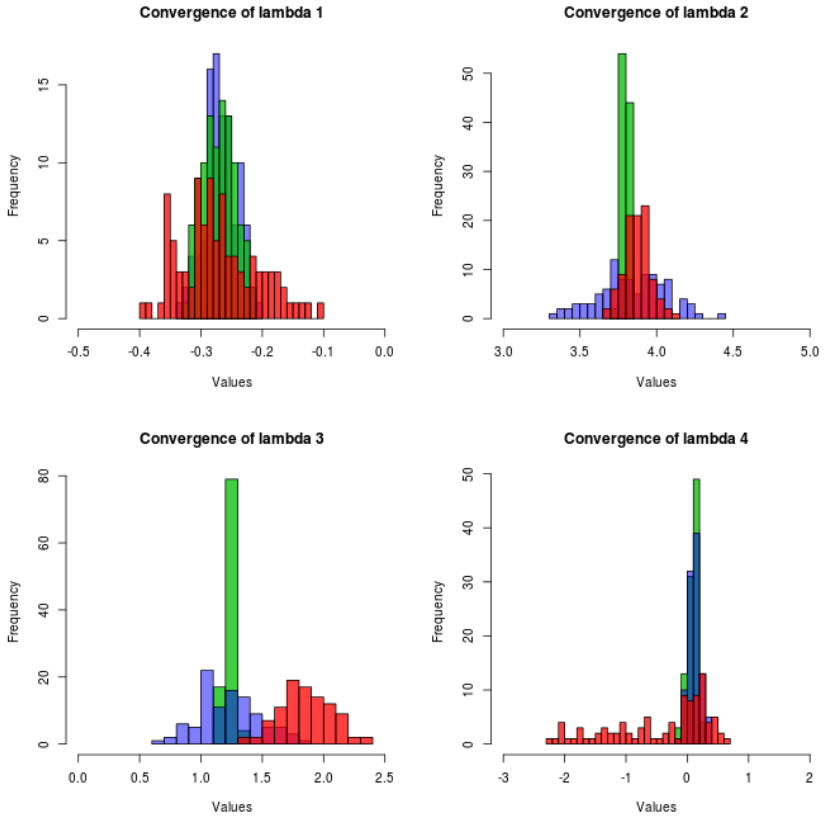

Figure 5.8: Histograms for each parameter value obtained by the CD procedure at iteration 100 for the 7x7 image patches. They were produced using 100 tests. The colors red, blue and green correspond to respectively CD-1 with $\eta = 25$, CD-1 with $\eta = 10$ and CD-20 with $\eta = 10$.

Figure 5.8 shows histogram plots for the different parameters. They strongly indicate that all CD procedures have converged within a proximity of the same parameters. The only parameter for which this conclusion is doubtful is λ_3, since the distribution for CD-1 with $\eta = 25$ seems to lean more to the right. However, given that all other parameters converge so clearly, it would seem unlikely that the CD procedure is drawing biased samples only for a single feature function. Given the high uniformity of CD-1 with $\eta = 25$, as is most evident in the bottom-right graph below, I will consider this to be an abnormality caused by either 1) an insufficient number of tests, or 2) an artifact of a high learning rate. Also, as expected, CD-20 with $\eta = 10$ gives the most stable solution, w.r.t. the parameter values, followed by CD-1 with $\eta = 10$ and lastly CD-1 with $\eta = 25$. I therefore conclude that the CD procedures converges to unique solutions for the Binary LN Models corresponding to 7x7 image patches.

For the 3x3 image patches I know that the CD procedure converges effectively close to the maximum likelihood solution. Because the Binary LN Model on the 7x7 image patches also converges, and with very high degree of confidence for CD-20 with $\eta = 25$, it is likely that the model is in fact converging close to the maximum likelihood solution. Thus, from the given experiments, I will reject that the CD procedures diverge significantly. Another argument supporting this conclusion will be discussed later, in reference to empirical results on the divergence of the CD procedure.

The CD Bias In the third chapter I established Theorem 3.15 for the upper bound on the expected bias of the CD procedure. A conclusion of the theorem was that a higher number of Gibbs chain steps L would likely result in a lower bias. This is also supported by, Theorem 3.8, which implies that higher L will lead to a Gibbs samples that are closer in expectation to the actual model. Nevertheless, in theory the actual bias could be up to $\max_k F_k$ large since every configuration has a strictly positive probability, which means that empirical results on the bias are necessary to improve our understanding of the Gibbs sampler. Performing a test on the bias requires knowing the exact log-likelihood gradient, which in turn yields the same computational complexity as the exact log-likelihood. I am therefore forced to limit the experiment to the nano model from before, i.e. the LN Binary Model on the 3x3 image patches. To be able to compare to the previous result I will keep $\eta = 25$ and the initialization $\boldsymbol{\lambda} = \mathbf{1}$. Like the previous experiment I will perform 50 tests and average the results over these. I will calculate two distinct quantities. First, the average bias, which is defined as the absolute difference between the true derivative and the derivative estimated from the Gibbs sample, averaged over all feature functions. Second, the maximum bias component, which is defined as the largest absolute difference between

the true derivative and the derivative estimated from the Gibbs sample, among all the parameters. From the theoretical results one should expect both components to become smaller the larger L gets.

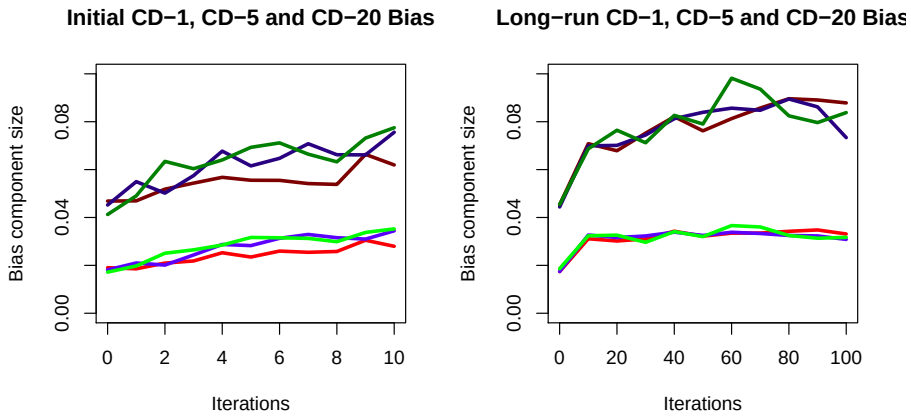

Figure 5.9: The graphs show the evolution of the two bias components: 1) the absolute average bias, and 2)the absolute maximum bias. They were produced using 50 tests measuring every 10th iteration. The colors red, blue and green correspond to CD-1, CD-5 and CD-20 trained using $\eta = 25$ for the 3x3 image patches. Left) Bias components for the first 10 iterations, Right) Bias components for the first 100 iterations.

The results, shown in Figure 5.9, shows no clear difference between CD-1, CD-5 and CD-20. They all appear to behave equally well with respect to the bias. This is not what I would expect from the my theoretical results established earlier. There can be two reasons for this: 1) either the model is so simple, that it simply cannot capture the differences, or 2) neither the upper bound nor the monotonicity of the Gibbs sampler is a good indicator of the actual bias.

However, the results brings to light a new observation. The bias appears to be increasing logarithmically during the learning procedure. Since the data-initialization in the CD procedure is getting closer to the model as the training proceeds, given the theorem on the upper bound of the bias, one would instead expect the bias to become smaller. The explanation for the growing bias can best be accounted for by the increasing parameters in the model. As the parameters grow the sampling becomes more deterministic and, thus, the Gibbs sampler will need more iterations to escape certain high-probability configurations. This explanation is also found in the literature, see Fischer and Igel [8].

Comparison of Different CD Procedures In the last experiments, I established that the different CD procedures were very similar, almost to the point of non-detection, w.r.t. the bias and the general convergence in log-likelihood. However, it could be that the different CD procedures performed differently w.r.t. speed of convergence. A more subtle analysis is required to investigate this. I will use the Binary LN Model for the 3x3 image patches, since I know that the solution of these converge in proximity to to the true log-likelihood for all the CD procedures. I will therefore use the previously obtained data on the squared parameter norms to see if there is any difference between the CD procedures. This is done by subtracting the CD-20 squared parameter norms with that of the CD-1 and CD-5.

Figure A.3, in the Appendix, shows that the squared parameter norm of the CD-1 procedure is growing slightly slower in the initial iterations than the CD-20 procedure, but that there is no difference between CD-5 and CD-20. The observation is not conclusive, but does give an indication that CD-1 "bounces around" more, e.g. moves the parameters in different directions for different learning iterations. The effect could be similar to the difference in convergence between iterative first-order optimization methods (for example gradient ascent) and second-order optimization methods (for example Newton's method). When applied to quadratic functions, the second-order optimization methods will often converge faster, while the first-order optimization method will "bounce around".

To further compare the different CD procedures I investigated the difference between CD-1 and Persistent CD-1. In particular, it would be interesting to test empirically the heuristic argument in favor of Persistent CD, given in chapter three. I therefore performed 50 tests using Persistent CD-1 with $\eta = 25$ for the 3x3 image patches and recorded the log-likelihoods. These are comparable to CD-1 with $\eta = 25$ from earlier.

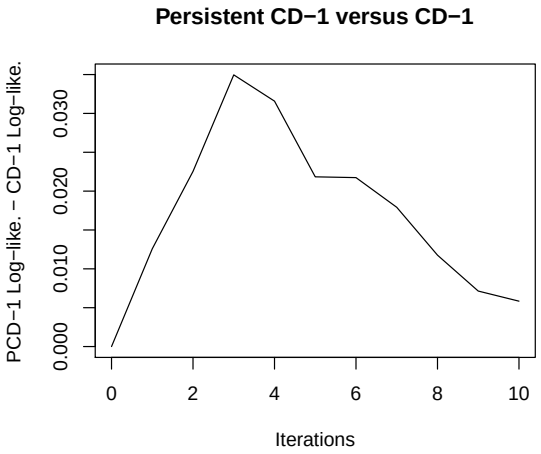

Figure 5.10: The graph shows the log-likelihood of Persistent CD-1 minus the log-likelihood of CD-1. The CD procedures were both trained with $\eta = 25$ for the 3x3 image patches. The graph was produced using 50 tests.

In Figure A.4, in the Appendix, the difference between CD-1 and Persistent CD-1 are very small. However, in Figure 5.10 the differences become more apparent. It indicates that the convergence speed of Persistent CD-1 is slightly faster than CD-1 in the initial iterations. Overall, however, the result is inconclusive. Further testing on a larger model could lead to more significant results.

CD Divergence Fischer and Igel [8] present a substantial number of results concerning the divergence of the CD learning procedure in Restricted Boltzmann Machines. See Hinton [13] for a description of the Restricted Boltzmann Machines and a discussion on preventing CD divergence when training these. When the CD procedure is applied to complex models the log-likelihood starts to decrease steadily after a certain number of iterations. This divergence is explained by the bias term in the CD procedure, which appears to grow very large when the 2-norm of the parameters grow. This is analogous to Theorem 3.15, where the upper bound on the bias term depends on size of the parameters. Since the Restricted Boltzmann Machines are closely related to the Conditional Random Fields, the results presented by Fischer and Igel are of significant importance.

However, the CRF Model is different from the Restricted Boltzmann Machine in a number of ways. The CRF Model is uniquely determined by each observed data sample, whereas the Restricted Boltzmann Machine is a priori defined. For the absolute bias term of the CD procedure to grow large, it has to grow in the same wrong direction for each k across the majority of Gibbs samples. If there are training samples where the feature function F_k is negative, and training samples where the feature function F_k is positive, this would intuitively make the CRF Model robust towards the bias term.

Secondly, the restriction of the Restricted Boltzmann Machine to be a bipartite graph, i.e. a graph consisting of two sets of hidden and visible nodes respectively, where connections only go from hidden to visible and visible to hidden, requires the Restricted Boltzmann Machines to have an excessive number of connections in order to model the data correctly. Fischer and Igel [8] present some relatively small Restricted Boltzmann Machines, where each node has 8 and 16 connections respectively, but in practice scientists typically work with thousands of connections for each node, see for example Hinton [13]. This could be a serious problem for the Gibbs sampler. If there was no connections between nodes, i.e. every node is independent of every other node, a single step of the Gibbs sampler would correspond to sampling from the actual distribution, which implies the bias is zero. As more connections are added to the network the Gibbs samplers performance is degraded. For many classification problems, CRF Models have the potential to avoid this excessive number of connections, since it is possible to construct rich models with relatively few connections. This could be done by specifying a larger number of feature functions, instead of adding connections between some nodes. As an example of this, Bremaud [6, p. 271] presents a *pixel-edge* model, which only has 4 connections for each node. In the more recent literature, Wallach [27] proposes a chain CRF Model where each node only has two neighbours.

These arguments are not conclusive and, indeed, I have observed the above divergence in my experiments on a small scale. However, it seems possible that CRF Models are by definition, and for some problems by construction, more robust to divergence than Restricted Boltzmann Machines. From an empirical viewpoint CD appears to be an effective and promising learning method for CRF Models.

Chapter 6

Conclusion

This thesis deals with Conditional Random Fields (CRFs). I defined and analysed a subset of CRFs named the Log-Linear Neighbourhood Models (LN Models). In particular, I confirmed that exact maximum likelihood estimation (MLE) becomes computationally intractable even for small LN Models, and that MLE is a convex optimization problem. Following this I connected LN Models to the latest applications of CRFs in the literature, and justified their importance from a practical perspective. This highlighted the importance of LN Models from both a theoretical point of view and for practical applications.

Next, I defined the Contrastive Divergence (CD) learning procedure. In relation to it, important results for the Gibbs sampler were presented. Based on results for Restricted Boltzmann Machines, I derived an upper bound on the expected bias of the CD procedure. I further developed some theoretical results for the CD procedure, including a result on the Gibbs chain update order and a heuristic argument in favor of Persistent CD. In the light of these theoretical results, I provided a number of discussions and ideas bridging theory with practical applications for the LN Models.

Then, Graph Cut methods for polynomial time function minimization were presented. A central theorem defining a class of functions, which Graph Cut methods could be applied to, was presented with proof. I established a subclass of LN Models, which Graph Cut methods could be applied to. In relation to the Graph Cut methods, I introduced the Max-Margin learning procedure, and connected it analytically to the maximum likelihood learning.

In the final part of the thesis, I developed an image denoising model named the Binary LN Model. I implemented this model as a software program with gradient ascent MLE, CD learning and Graph Cut methods. I then used the model to trial and investigate the developed theory on CD learning. In

particular, I analysed the convergence properties and related the empirical results to the established theoretical results. Lastly, the divergence of the CD procedure was discussed in relation to empirical results obtained for the Restricted Boltzmann Machines. Overall, my experiments on the model showed encouraging results for the CD procedure.

Directions for Further Work The work presented in this thesis has taken a broad approach to CRFs. It is therefore likely that further theoretical results are obtainable for the LN Model and general CRFs w.r.t. CD learning. Indeed, results for CD learning in the LN Model are especially important because they show both promising theoretical and empirical results. Therefore, the heuristic argument provided in favor of Persistent CD should be explored further and, if possible, be put on a formal grounding. The latest research in a variant of Persistent CD, proposed by Tieleman and Hinton [26], and the method of Parallel Tempering, see Salakhutdinov [23], are also likely to be successful alternatives to the CD procedures presented in this thesis. Another important aspect, which has not been discussed in this thesis, would be to investigate learning rates that change with each learning iteration. These could improve the CD procedure significantly.

While continuing research on CD learning, it is critical to compare it to state-of-the-art methods, such as Max-Margin learning using Graph Cuts and the results presented by Korc and Forstner [18]. Further work on developing the analytical and empirical connection between maximum likelihood learning and Max-Margin learning is also needed. Work in this direction could benefit from the theoretical results given by Finley and Joachims [7].

The empirical results presented in this thesis begs additional experimentation. It would be of great potential benefit to theoretical and practical developments, that further empirical studies are carried out on CD learning in CRFs. A good starting point is the chain CRF proposed by Wallach [27], which is also an LN Model. In this model exact log-likelihood is tractable and therefore the CD procedure can be evaluated w.r.t. to the exact maximum likelihood solution. The model proposed by Nowozin *et al.* [22] is also exact log-likelihood tractable, and could also be used for experimentation. An empirical comparison between CD learning and pseudo-likelihood learning, proposed by Korc and Forstner [18], could also be fruitful. In general, further analysis should be carried out to compare the different CD procedures.

Appendix A

Experimental Results

This Appendix lists a number of empirical results for the Binary LN Model, which are discussed in chapter five of the thesis.

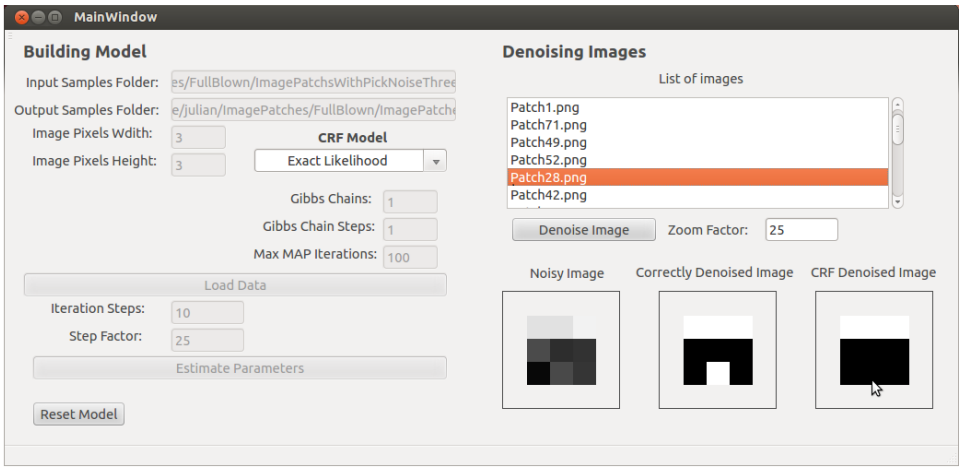

Figure A.1: Screenshot of the software program. The Binary LN, trained with gradient ascent exact likelihood, is used to denoise a 3x3 image patch.

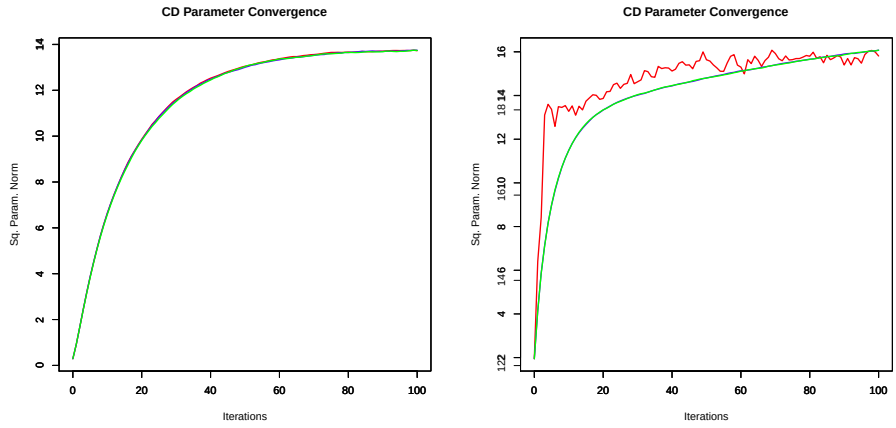

Figure A.2: The squared parameter norm of the CD procedures. They were produced using 100 tests. Left) Nano model, where the colors red, blue and green correspond to respectively CD-1, CD-5 and CD-20 trained with $\eta = 25$. Right) Full-scale model, where the colors red, blue and green correspond to CD-1 trained using $\eta = 25$ and CD-1 and CD-20 trained using $\eta = 10$ respectively.

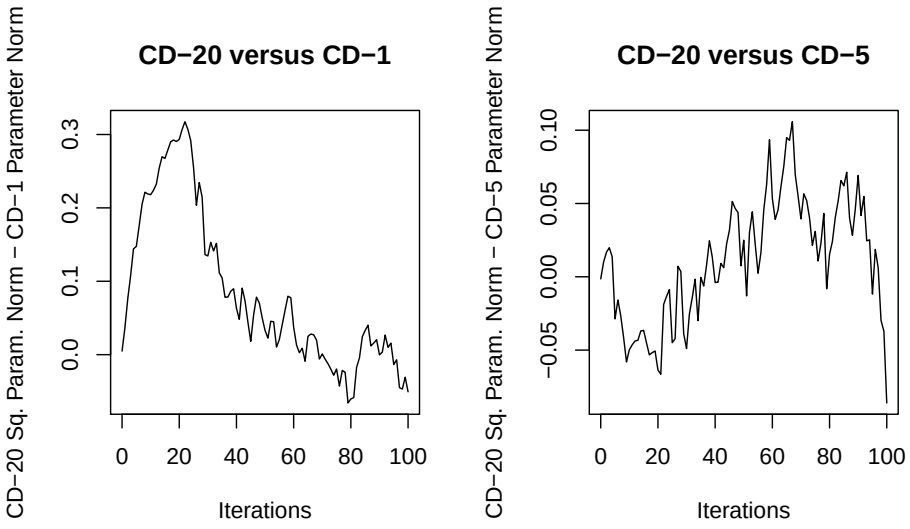

Figure A.3: The graphs show the CD procedures trained with $\eta = 25$ for the 3x3 image patches. They were produced using 50 tests. Right) The squared parameter norm of CD-20 minus the squared parameter norm of CD-5.

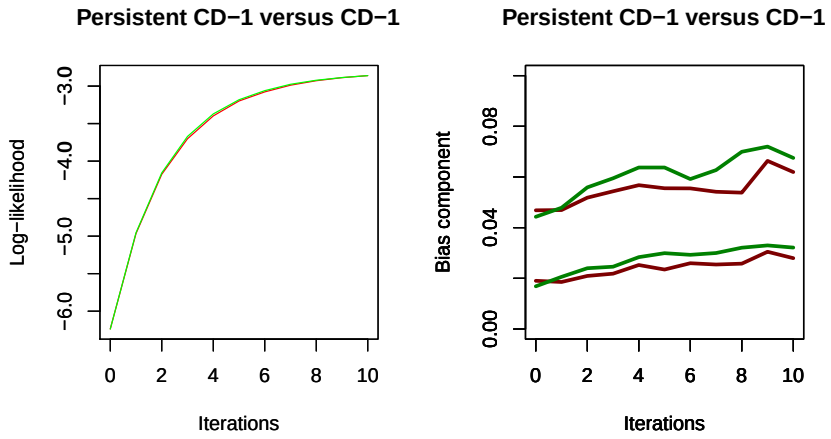

Figure A.4: The colors green and red represent Persistent CD-1 and CD-1, respectively, trained with $\eta = 25$ for the 3x3 image patches. The graphs were produced using 50 tests. Left) The log-likelihood of CD-1 and Persistent CD-1. Right) The bias components of CD-1 and Persistent CD-1.

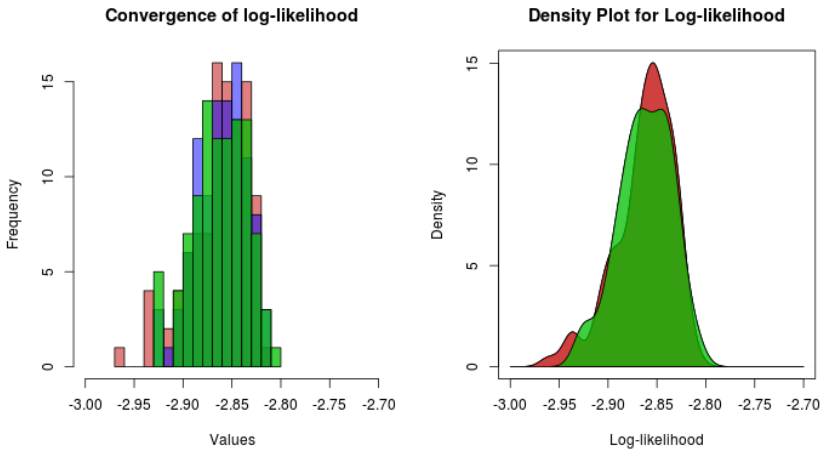

Figure A.5: Histogram and density plots for the log-likelihood, at iteration 100, for the 3x3 image patches. The colors red, blue and green correspond to respectively CD-1, CD-5 and CD-20 trained using learning rate $\eta = 25$. They were produced using 100 tests. Left) Histogram plot of CD-1, CD-5 and CD-20. Right) Density plot of CD-1 and CD-5 with 512 Gaussian kernels.

Bibliography

[1] Y. Bengio and O. Delalleau. Justifying and generalizing contrastive divergence. *Neural Computation*, 21(6):1601–1621, 2009.

[2] A. H. Black. *Man and Nature in the Philosophical Thought of Wang Fu-chih*. University of Washington Press, first edition, 1989.

[3] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 26(9):1124–1137, 2004.

[4] Y. Boykov and O. Veksler. Graph cuts in vision and graphics: Theories and applications. In N. Paragios, , Y. Chen, and O. Faugeras, editors, *Handbook of Mathematical Models in Computer Vision*, chapter 4, pages 79–96. Springer US, first edition, 2006.

[5] Y. Y. Boykov and M. Jolly. Interactive graph cuts: for optimal boundary and region segmentation of objects in n-d images. *Proceedings of the Internation Conference on Computer Vision (ICCV)*, 10:105–112, 2001.

[6] P. Bremaud. *Markov Chains: Gibbs Fields, Monte Carlo Simulation, and Queues*. Springer, 1999.

[7] T. Finley and T. Joachims. Training structural svms when exact inference is intractable. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, pages 304–311. ACM, 2008.

[8] A. Fischer and C. Igel. Empirical analysis of the divergence of gibbs sampling based learning algorithms for restricted boltzmann machines. *International Conference on Artificial Neural Networks (ICANN)*, LNCS 6354:208–217, 2010.

[9] A. Fischer and C. Igel. Bounding the bias of contrastive divergence learning. *Neural Computation*, 23:664–673, 2011.

[10] L. R. Ford and D. R. Fulkerson. *Flows in Networks*. Princeton University Press, first edition, 1962.

[11] A. V. Goldberg and R. E. Tarjan. A new approach to the maximum flow problem. *Journal of the ACM (JACM)*, 35(4):921–940, 1988.

[12] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer, second edition, 2009.

[13] G. E. Hinton. *A Practical Guide to Training Restricted Boltzmann Machines, Version 1.* Department of Computer Science, University of Toronto, 2010.

[14] P. Kohli, L. Ladicky, and P. H. S. Torr. Robust higher order potentials for enforcing label consistency. *International Journal of Computer Vision (IJCV)*, 82(3):302–324, 2009.

[15] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques.* The MIT Press, 2009.

[16] V. Kolmogorov. Personal webpage listing the authors software, May 2012. http://pub.ist.ac.at/ vnk/software.html.

[17] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 26(2):147–159, February 2004.

[18] F. Korc and W. Forstner. Approximate parameter learning in conditional random fields: An empirical investigation. *In Proceedings of the 30th DAGM symposium on Pattern Recognition*, pages 11–20, 2008.

[19] S. Kumar, J. August, and M. Hebert. Exploiting inference for approximate parameter learning in discriminative fields: An empirical study. *Proceedings of the 5th international conference on Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR)*, pages 153–168, 2005.

[20] L. Ladicky, C. Russell, P. H. S. Torr, and P. Kohli. Associative hierarchical crfs for object class image segmentation. In *in Proceedings of the International Conference on Computer Vision (ICCV)*, pages 739–746. IEEE Computer Society, 2009.

[21] S. F. Nielsen. *Matematisk statistik.* Department Of Mathematical Sciences, fourth edition, 2010.

[22] S. Nowozin, P. V. Gehler, and C. H. Lampert. On parameter learning in crf-based approaches to object class image segmentation. In *Proceedings of the 11th European conference on Computer vision: Part VI*, pages 98–111. Springer-Verlag, 2010.

[23] R. Salakhutdinov. Learning in markov random fields using tempered transitions. *Advances in Neural Information Processing Systems (NIPS)*, 22:1598âĂŞ1606, 2009.

[24] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother. A comparative study of energy minimization: Methods for markov random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(6):1068–1080, 2006.

[25] M. Szummer, P. Kohli, and D. Hoiem. Learning crfs using graph cuts. In *European Conference on Computer Vision (ECCV): Part II*, pages 582–595. Springer-Verlag, 2008.

[26] T. Tieleman and G.E Hinton. Using fast weights to improve persistent contrastive divergence. volume 22, page 1033âĂŞ1040. ACM, 2009.

[27] H. M. Wallach. Conditional random fields: An introduction. *University of Pennsylvania*, CIS Technical Report MS-CIS-04-21, 2004.