Some investigations into energy-based models

by

Tijmen Tieleman

A thesis submitted in conformity with the requirements
for the degree of Master of Science
Graduate Department of Computer Science
University of Toronto

Abstract

Some investigations into energy-based models

Tijmen Tieleman

Master of Science

Graduate Department of Computer Science

University of Toronto

2007

Three questions about various energy-based probability models are asked and answered. The first is whether the Contrastive Divergence algorithm computes the gradient of any function at all - the answer is no. The second is whether there is a tractable Monte Carlo approximation to the gradient for variational learning in a large class of models including Sigmoid Belief Networks - the answer is yes. The third is how we might do early stopping for Restricted Boltzmann Machines, which have intractable objective functions - the problem is studied thoroughly, some old algorithms are reviewed and some new and better ones are introduced, and the way is pointed to the ultimate algorithm for this problem.

Acknowledgements

There can be no students without teachers, so I would like to thank my teachers here in the Machine Learning group of the University of Toronto, namely Geoff Hinton and Sam Roweis. Geoff Hinton, as my supervisor, is the source of most of the inspiration and ideas that went into this thesis.

Ilya Sutskever helped me by providing comments on drafts of this thesis, and the makers of Mathematica kindly made "The Integrator" available on the internet, which helped me find some integrals that I might not have found on my own.

Contents

Chapter 1

Introduction

In this thesis, I describe some findings of the past one and a half years of my research. Because that research covered multiple areas in machine learning, there will be several sections in this thesis, with only weak connections between them. The introduction brings in the necessary definitions and conventions. After that, the first section describes a finding about the Contrastive Divergence algorithm. The second section describes a gradient derivation for a different model. The third section describes how to do early stopping in energy-based models.

A large number of mathematical formulas appear in this thesis. I have attempted to make sure that they are not required for general understanding, by including much intuitive explanation. Einstein allegedly said once that an expert is he who can explain his knowledge to his grandmother, and since this Master's thesis is an attempt to show expertise, I have tried to make it as accessible as possible.

Since we stand on the shoulders of giants, much of this thesis is introduction to and explanation of the work on which my findings build. This thesis is, unfortunately, not aiming to be entirely understandable to my grandmother, so I chose to explain only briefly the oldest and hopefully best known work, and give more explanation of the newer work. Of course, references are included everywhere.

Chapter 2

Boltzmann Machines

2.1 General

One approach to unsupervised learning is the Boltzmann Machine (BM) neural network [6], which is briefly described in this section. A BM defines a probability distribution over binary vectors of some fixed length. It is usually visualized as a collection of units (neurons), and symmetric connections between them (which can be thought of as pairs of synapses). Such a connection connects two neurons, and not all pairs of neurons need be connected. Neurons are not connected to themselves. Neurons also have a bias, which can be interpreted as a connection to an invisible neuron that is always active.

2.2 Main concepts and notation

Let N be the number of neurons. The bias to the i^{th} unit can be denoted by b_i. The strength of the connection between unit i and unit j will be c_{ij}.

For simplicity, let us pretend that all connections are present. The ones that we want to be absent can have strength 0, which in effect makes them absent.

A configuration of a BM is an assignment of binary values (activities) to each of the units. Clearly, there are 2^N configurations. Let us denote the state (activity) of the i^{th}

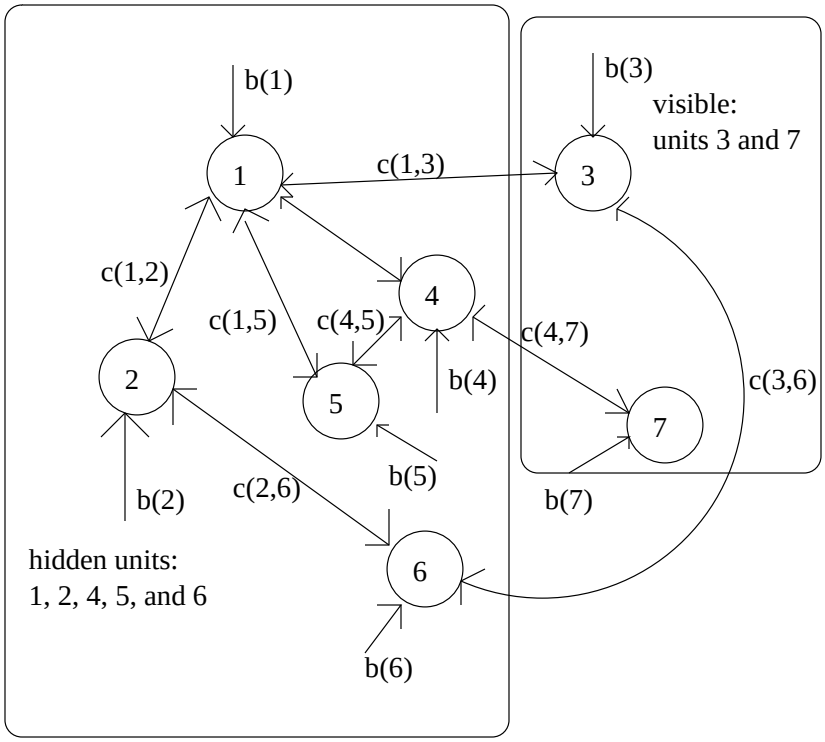

Figure 2.1: Visualization of a Boltzmann Machine. Some of the units are designated as visible units, while the others are hidden. Units, biases, and connections are named.

unit in configuration C by s_i^C.

The first concept is that of energy. Every configuration has an energy, which is defined to be $E(C) = -\sum_i s_i^C \cdot b_i - \sum_{i<j} s_i^C \cdot s_j^C \cdot c_{ij}$. From this, one can see that indeed, the biases can be thought of as connections to units that are always on.

From these energies, probabilities can be calculated. The probability of some configuration C is defined to be $P(C) = \frac{e^{-E(C)}}{Z}$, where the normalizing constant Z is defined to be $Z = \sum_\alpha e^{-E(\alpha)}$, with α standing for each of the 2^N configurations.

With these definitions, one has a model of data distributions, which can therefore be used for unsupervised learning. To model a data set D, consisting of binary vectors of length l, create a BM with l units, and somehow try to find biases and connection strengths that make the model's distribution close to the data distribution.

Alternatively, one can designate some of the units to be visible units, and the remaining ones to be hidden units. Then, to model the same data set, create a BM with l visible units and some freely chosen number of hidden units. The probability of a data vector x under this alternative BM is the sum of probabilities of all configurations in which the visible units' states are in accordance with x. There are 2^{n_h} such configurations, where n_h is the number of hidden units. This second approach to modeling is the one we will use from now on. Having hidden units has the advantage that more distributions can be modeled; if an unlimited number of them is available, any distribution that does not include probabilities of zero can be modeled.

2.3 Likelihood gradient for BMs

Before we continue, it is time to fix some more notation. Let us rename the individual neurons: v_i is the i^{th} visible unit, and h_i is the i^{th} hidden unit. Symbols $\vec{v}$ and $\vec{h}$ stand for configurations of the full set of visible and hidden units. For example, $P(\vec{v})$ stands for the probability that a BM assigns to some configuration $\vec{v}$ of the visible units.

Since a configuration of the visible units, together with a configuration of the hidden units, comprises a full configuration, the pair will have an energy, which we denote as $E(\vec{v}, \vec{h})$. $P_{\text{train}}(\cdot)$ stands for the training data distribution, which can be viewed both as a distribution over binary vectors of length l and as a (desired) distribution over configurations of the visible units. Thus, the goal in learning is to get a BM for which $P(\cdot)$ over configurations of the visible units (which represent data points) is similar to the training data distribution $P_{\text{train}}(\cdot)$.

Let us now look more closely at the process of finding a set of biases and connection strengths that yield a model distribution that is close to the data distribution. Since exhaustive search is too slow for current computers, we do this by gradient training. We start with some parameter vector θ which describes all biases and connection strengths, and try to change θ in such a way that the model gets better. In short, we repeatedly find the gradient vector of θ w.r.t. some objective function, and change θ a bit in the direction of that gradient. So first, we need an objective function.

Assuming the training data is a sequence of i.i.d. observations, one reasonable objective function is the product of probabilities of the training data points, also called likelihood. For more convenient notation and programming we instead prefer the logarithm of this likelihood, and to make it all look even more mathematically grounded, let us choose the average log likelihood, i.e. $\phi = \sum_{\vec{v}} P_{\text{train}}(\vec{v}) \log(P(\vec{v}))$. Let us also make the dependence on θ explicit: $\phi(\theta) = \sum_{\vec{v}} P_{\text{train}}(\vec{v}) \log(P^\theta(\vec{v}))$. This dependence also suggests switching to $E^\theta(C)$ (or $E^\theta(\vec{v}, \vec{h})$) for energy, and Z^θ for the normalizing constant. With all this new notation, the probability of some data point, or equivalently some configuration of the visible units, is $P^\theta(\vec{v}) = \frac{\sum_{\vec{h}} e^{-E^\theta(\vec{v}, \vec{h})}}{Z^\theta}$.

The gradient can now be found through some juggling of equations.

$$\begin{aligned}
\frac{\partial \phi(\theta)}{\partial \theta} =& \frac{\partial}{\partial \theta} \sum_{\vec{v}} P_{\text{train}}(\vec{v}) \log(P^{\theta}(\vec{v})) \\
=& \frac{\partial}{\partial \theta} \sum_{\vec{v}} P_{\text{train}}(\vec{v}) \log \left(\frac{\sum_{\vec{h}} e^{-E^{\theta}(\vec{v},\vec{h})}}{Z^{\theta}} \right) \\
=& \frac{\partial}{\partial \theta} \sum_{\vec{v}} P_{\text{train}}(\vec{v}) \cdot \left(\log \sum_{\vec{h}} e^{-E^{\theta}(\vec{v},\vec{h})} - \log(Z^{\theta}) \right) \\
=& - \frac{\partial \log(Z^{\theta})}{\partial \theta} + \frac{\partial \sum_{\vec{v}} P_{\text{train}}(\vec{v}) \log \left(\sum_{\vec{h}} e^{-E^{\theta}(\vec{v},\vec{h})} \right)}{\partial \theta} \\
=& - \frac{\partial \phi^{-}(\theta)}{\partial \theta} + \frac{\partial \phi^{+}(\theta, P_{\text{train}})}{\partial \theta}
\end{aligned} \tag{2.1}$$

In the last line, the gradient was separated into two parts: the gradient of $\phi^{-}(\theta) = \log(Z^{\theta})$, the negative part, and the gradient of $\phi^{+}(\theta, P_{\text{train}}) = \sum_{\vec{v}} P_{\text{train}}(\vec{v}) \log \left(\sum_{\vec{h}} e^{-E^{\theta}(\vec{v},\vec{h})} \right)$, the positive part; named after how they appear in ϕ. They are somewhat independent and deserve independent attention.

Notice first, however, for any nonempty set S of configurations, that

$$\begin{aligned}
\frac{\partial \log \left(\sum_{C \in S} e^{-E^{\theta}(C)} \right)}{\partial \theta} =& \frac{1}{\sum_{C \in S} e^{-E^{\theta}(C)}} \cdot \frac{\partial \sum_{C \in S} e^{-E^{\theta}(C)}}{\partial \theta} \\
=& \frac{1}{\sum_{C \in S} e^{-E^{\theta}(C)}} \cdot \sum_{C \in S} \frac{\partial e^{-E^{\theta}(C)}}{\partial \theta} \\
=& \frac{1}{\sum_{C \in S} e^{-E^{\theta}(C)}} \cdot \sum_{C \in S} e^{-E^{\theta}(C)} \cdot \frac{\partial - E^{\theta}(C)}{\partial \theta} \\
=& \sum_{C \in S} \frac{e^{-E^{\theta}(C)}}{\sum_{C \in S} e^{-E^{\theta}(C)}} \cdot \frac{\partial - E^{\theta}(C)}{\partial \theta} \\
=& \sum_{C \in S} \frac{P^{\theta}(C)}{P^{\theta}(S)} \cdot \frac{\partial - E^{\theta}(C)}{\partial \theta}
\end{aligned} \tag{2.2}$$

Equation 2.2, in which $P^{\theta}(S)$ stands for the $\sum_{C \in S} P^{\theta}(C)$, can be seen as a P^{θ}-weighted average over S of $\frac{\partial - E^{\theta}(C)}{\partial \theta}$.

2.3.1 $\frac{\partial \phi^-(\theta)}{\partial \theta}$

Using equation 2.2 with S being the set of all configurations, it is immediate that

$$\begin{aligned}\frac{\partial \phi^-(\theta)}{\partial \theta} =& \frac{\partial \log(Z^\theta)}{\partial \theta} \\ =& \frac{\partial \log\left(\sum_C e^{-E^\theta(C)}\right)}{\partial \theta} \\ =& \sum_C P^\theta(C) \frac{\partial - E^\theta(C)}{\partial \theta}\end{aligned} \tag{2.3}$$

2.3.2 $\frac{\partial \phi^+(\theta, P_{\mathbf{train}})}{\partial \theta}$

Equation 2.2 is used here, too, once for each $\vec{v}$, with S being the set of all $(\vec{v}, \vec{h})$ pairs with that particular $\vec{v}$.

$$\begin{aligned}\frac{\partial \phi^+(\theta, P_{\text{train}})}{\partial \theta} =& \frac{\partial \sum_{\vec{v}} P_{\text{train}}(\vec{v}) \log\left(\sum_{\vec{h}} e^{-E^\theta(\vec{v},\vec{h})}\right)}{\partial \theta} \\ =& \sum_{\vec{v}} P_{\text{train}}(\vec{v}) \frac{\partial \log\left(\sum_{\vec{h}} e^{-E^\theta(\vec{v},\vec{h})}\right)}{\partial \theta} \\ =& \sum_{\vec{v}} P_{\text{train}}(\vec{v}) \sum_{\vec{h}} \frac{P^\theta(\vec{v},\vec{h})}{p^\theta(\vec{v})} \cdot \frac{\partial - E^\theta(\vec{v},\vec{h})}{\partial \theta} \\ =& \sum_{\vec{v}} P_{\text{train}}(\vec{v}) \sum_{\vec{h}} P^\theta(\vec{h}|\vec{v}) \cdot \frac{\partial - E^\theta(\vec{v},\vec{h})}{\partial \theta}\end{aligned} \tag{2.4}$$

2.3.3 Monte Carlo approximation

To get an unbiased estimate of $\frac{\partial \phi^+(\theta, P_{\text{train}})}{\partial \theta}$, take for each training data point $\vec{v}$ a sample $\vec{h}$ from $P^\theta(\cdot|\vec{v})$, and average $\frac{\partial - E^\theta(\vec{v},\vec{h})}{\partial \theta}$ (which is trivial to calculate) over these $(\vec{v}, \vec{h})$ pairs.

To get an unbiased estimate of $\frac{\partial \phi^-(\theta)}{\partial \theta}$, take a sample configuration C from $P^\theta(\cdot)$, and take $\frac{\partial - E^\theta(C)}{\partial \theta}$ of that configuration.

Finding these samples is, unfortunately, intractable in general. One could do it approximately using prolonged Gibbs sampling, but in the next section a tractable and reasonable approximation algorithm for a restricted class of BMs is presented.

2.4 Restricted Boltzmann Machines

Obtaining samples from various distributions in BMs is difficult but necessary. This suggests the use of a more restricted class of BMs, which still leaves sufficient complexity to enable interesting distributions, but somehow makes Gibbs sampling easier. That is indeed achieved by requiring that there are no connections from hidden units to other hidden units, and from visible units to other visible units. A BM that satisfies these requirements is called a Restricted Boltzmann Machine (RBM), first introduced in [9].

The states of the hidden units are now independent given the states of the visible units, and vice versa. This makes Gibbs sampling particularly easy and efficient: given the states of the visible units, states for all of the hidden units can be sampled at the same time, and vice versa.

The independence of the hidden units given the states of the visible units means that the derivative of ϕ^+ can quickly be found exactly (no Monte Carlo approximation needed any more). Getting the derivative of ϕ^- is still intractable, but a workable approximation can now be found.

2.4.1 Unit state probabilities in RBMs

The probability of the i^{th} hidden unit being in the active state (state 1), given the states of all other units, can be calculated from the probability of two configurations: C_0 and C_1. C_0 is the configuration with all units as given and the i^{th} hidden unit in state 0; C_1 is the same except that $s_{h_i}^{C_1} = 1$. These configurations are very similar, which can be highlighted by writing their energies using a common part: $E(C_0) = E_{\text{common}} + E_0$ and $E(C_1) = E_{\text{common}} + E_1$, for some wise choice of E_{common}, E_0, and E_1.

$$E(C_0) = -\sum_j s_j^{C_0} \cdot b_j - \sum_{j<k} s_j^{C_0} \cdot s_k^{C_0} \cdot c_{jk}$$
$$E(C_1) = -\sum_j s_j^{C_1} \cdot b_j - \sum_{j<k} s_j^{C_1} \cdot s_k^{C_1} \cdot c_{jk} \tag{2.5}$$

The wise choice of E_{common} (and with it E_0 and E_1) is

$$E_{\text{common}} = -\sum_{j\neq i} s_j^{C_0} \cdot b_j - \sum_{j,k:j<k,j\neq i,k\neq i} s_j^{C_0} \cdot s_k^{C_0} \cdot c_{jk} \tag{2.6}$$

$$E_1 = -b_{h_i} - \sum_{j\neq h_i} s_j^{C_0} \cdot c_{h_i,j} = -b_{h_i} + -\sum_{j=1}^{n_v} s_{v_j}^{C_0} \cdot c_{v_j,h_i} \tag{2.7}$$

$$E_0 = 0$$

The simplification in equation 2.7 uses the fact that in an RBM, the only connections to a hidden unit are those from visible units. Putting things together, we conclude that

$$\begin{aligned}
P(s_{h_i} = 1|\text{states of all other units}) =& \frac{P(C_1)}{P(C_0) + P(C_1)} \\
=& \frac{e^{-E(C_1)}}{e^{-E(C_0)} + e^{-E(C_1)}} \\
=& \frac{e^{-E_{\text{common}}-E_1}}{e^{-E_{\text{common}}-E_0} + e^{-E_{\text{common}}-E_1}} \\
=& \frac{e^{-E_{\text{common}}} \cdot e^{-E_1}}{e^{-E_{\text{common}}} \cdot e^{-E_0} + e^{-E_{\text{common}}} \cdot e^{-E_1}} \\
=& \frac{e^{-E_1}}{e^{-E_0} + e^{-E_1}} \\
=& \frac{1}{e^{E_1} + 1} \\
=& \frac{1}{e^{-\left(b_{h_i} + \sum_{j=1}^{n_v} s_{v_j} \cdot c_{v_j,h_i}\right)} + 1} \\
=& \sigma\left(b_{h_i} + \sum_{j=1}^{n_v} s_{v_j} \cdot c_{v_j,h_i}\right)
\end{aligned} \tag{2.8}$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$. Observe that this probability depends only on the states of the visible units.

Similarly, the configuration probabilities for a visible unit, given the configuration of all hidden units, are independent of the configuration of the other visible units:

$$P(s_{v_i} = 1|\text{states of all other units}) = \sigma\left(b_{v_i} + \sum_{j=1}^{n_h} s_{h_j} \cdot c_{v_i,h_j}\right) \tag{2.9}$$

2.5 Likelihood gradient for RBMs

2.5.1 The derivative of ϕ^+

$\frac{\partial \phi^+}{\partial \theta}$ can now be computed exactly.

$$\begin{aligned}
\frac{\partial \phi^+(\theta, P_{\text{train}})}{\partial \theta} =& \sum_{\vec{v}} P_{\text{train}}(\vec{v}) \sum_{\vec{h}} P^\theta(\vec{h}|\vec{v}) \cdot \frac{\partial - E^\theta(\vec{v}, \vec{h})}{\partial \theta} \\
=& \sum_{\vec{v}} P_{\text{train}}(\vec{v}) \sum_{h_1=0}^{1} \sum_{h_2=0}^{1} \sum_{h_3=0}^{1} \cdots \sum_{h_{n_h}=0}^{1} \\
& P^\theta(h_1, h_2, h_3, \ldots, h_{n_h}|\vec{v}) \cdot \frac{\partial - E^\theta(\vec{v}, h_1, h_2, h_3, \ldots, h_{n_h})}{\partial \theta} \\
=& \sum_{\vec{v}} P_{\text{train}}(\vec{v}) \sum_{h_1=0}^{1} \sum_{h_2=0}^{1} \sum_{h_3=0}^{1} \cdots \sum_{h_{n_h}=0}^{1} \\
& P^\theta(h_1|\vec{v}) \cdot P^\theta(h_2|h_1, \vec{v}) \cdot P^\theta(h_3|h_1, h_2, \vec{v}) \cdot \ldots \cdot P^\theta(h_{n_h}|h_1, h_2, \ldots, h_{n_h-1}, \vec{v}) \cdot \\
& \frac{\partial - E^\theta(\vec{v}, h_1, h_2, h_3, \ldots, h_{n_h})}{\partial \theta} \\
=& \sum_{\vec{v}} P_{\text{train}}(\vec{v}) \sum_{h_1=0}^{1} \sum_{h_2=0}^{1} \sum_{h_3=0}^{1} \cdots \sum_{h_{n_h}=0}^{1} \\
& P^\theta(h_1|\vec{v}) \cdot P^\theta(h_2|\vec{v}) \cdot P^\theta(h_3|\vec{v}) \cdot \ldots \cdot P^\theta(h_{n_h}|\vec{v}) \cdot \\
& \frac{\partial - E^\theta(\vec{v}, h_1, h_2, h_3, \ldots, h_{n_h})}{\partial \theta}
\end{aligned} \tag{2.10}$$

The last step is done using the observation of equation 2.8 that the distribution over configurations of a hidden unit given the configuration of the visible units is independent of the configuration of the other hidden units.

From equation 2.10, it is easy to find the derivative of ϕ^+ w.r.t. the strength of the connection between the i^{th} visible unit and the j^{th} hidden unit. Notation $[x]$ stands for the function that is 1 when x is true and 0 when x is false.

$$\begin{aligned}
\frac{\partial \phi^+(\theta, P_{\text{train}})}{\partial c_{v_i,h_j}} &= \sum_{\vec{v}} P_{\text{train}}(\vec{v}) \sum_{h_1=0}^{1} \sum_{h_2=0}^{1} \sum_{h_3=0}^{1} \cdots \sum_{h_{n_h}=0}^{1} \\
&\quad P^\theta(h_1|\vec{v}) \cdot P^\theta(h_2|\vec{v}) \cdot P^\theta(h_3|\vec{v}) \cdot \ldots \cdot P^\theta(h_{n_h}|\vec{v}) \cdot \\
&\quad \frac{\partial - E^\theta(\vec{v}, h_1, h_2, h_3, \ldots, h_{n_h})}{\partial c_{v_i,h_j}} \\
&= \sum_{\vec{v}} P_{\text{train}}(\vec{v}) \sum_{h_1=0}^{1} \sum_{h_2=0}^{1} \sum_{h_3=0}^{1} \cdots \sum_{h_{n_h}=0}^{1} \\
&\quad P^\theta(h_1|\vec{v}) \cdot P^\theta(h_2|\vec{v}) \cdot P^\theta(h_3|\vec{v}) \cdot \ldots \cdot P^\theta(h_{n_h}|\vec{v}) \cdot \\
&\quad [s^{\vec{v}}_{v_i} = 1 \wedge s_{h_j} = 1] \\
&= \sum_{\vec{v}} P_{\text{train}}(\vec{v}) [s^{\vec{v}}_{v_i}] \cdot P^\theta(s_{h_j} = 1|\vec{v}) \\
&= \sum_{\vec{v}} P_{\text{train}}(\vec{v}) [s^{\vec{v}}_{v_i}] \cdot \sigma \left(b_{h_j} + \sum_{k=0}^{n_v} s^{\vec{v}}_{v_k} \cdot c_{v_k,h_j} \right)
\end{aligned} \tag{2.11}$$

For the biases it is even simpler:

$$\begin{aligned}
\frac{\partial \phi^+(\theta, P_{\text{train}})}{\partial b_{v_i}} &= \sum_{\vec{v}} P_{\text{train}}(\vec{v}) \sum_{h_1=0}^{1} \sum_{h_2=0}^{1} \sum_{h_3=0}^{1} \cdots \sum_{h_{n_h}=0}^{1} \\
&\quad P^\theta(h_1|\vec{v}) \cdot P^\theta(h_2|\vec{v}) \cdot P^\theta(h_3|\vec{v}) \cdot \ldots \cdot P^\theta(h_{n_h}|\vec{v}) \cdot \\
&\quad \frac{\partial - E^\theta(\vec{v}, h_1, h_2, h_3, \ldots, h_{n_h})}{\partial b_{v_i}} \\
&= \sum_{\vec{v}} P_{\text{train}}(\vec{v}) \sum_{h_1=0}^{1} \sum_{h_2=0}^{1} \sum_{h_3=0}^{1} \cdots \sum_{h_{n_h}=0}^{1} \\
&\quad P^\theta(h_1|\vec{v}) \cdot P^\theta(h_2|\vec{v}) \cdot P^\theta(h_3|\vec{v}) \cdot \ldots \cdot P^\theta(h_{n_h}|\vec{v}) \cdot \\
&\quad [s^{\vec{v}}_{v_i} = 1] \\
&= \sum_{\vec{v}} P_{\text{train}}(\vec{v}) \cdot [s^{\vec{v}}_{v_i} = 1]
\end{aligned} \tag{2.12}$$

And

$$\begin{aligned}
\frac{\partial \phi^+(\theta, P_{\text{train}})}{\partial b_{h_i}} =& \sum_{\vec{v}} P_{\text{train}}(\vec{v}) \sum_{h_1=0}^{1} \sum_{h_2=0}^{1} \sum_{h_3=0}^{1} \cdots \sum_{h_{n_h}=0}^{1} \\
& P^\theta(h_1|\vec{v}) \cdot P^\theta(h_2|\vec{v}) \cdot P^\theta(h_3|\vec{v}) \cdot \ldots \cdot P^\theta(h_{n_h}|\vec{v}) \cdot \\
& \frac{\partial - E^\theta(\vec{v}, h_1, h_2, h_3, \ldots, h_{n_h})}{\partial b_{h_i}} \\
=& \sum_{\vec{v}} P_{\text{train}}(\vec{v}) \sum_{h_1=0}^{1} \sum_{h_2=0}^{1} \sum_{h_3=0}^{1} \cdots \sum_{h_{n_h}=0}^{1} \\
& P^\theta(h_1|\vec{v}) \cdot P^\theta(h_2|\vec{v}) \cdot P^\theta(h_3|\vec{v}) \cdot \ldots \cdot P^\theta(h_{n_h}|\vec{v}) \cdot \\
& [s_{h_i} = 1] \\
=& \sum_{\vec{v}} P_{\text{train}}(\vec{v}) \cdot \sigma \left(b_{h_i} + \sum_{j=1}^{n_v} s_{v_j}^{\vec{v}} \cdot c_{v_j, h_i} \right)
\end{aligned} \tag{2.13}$$

2.5.2 The derivative of ϕ^-

$\frac{\partial \phi^-(\theta)}{\partial \theta}$ is intractable, also in RBMs, but there is a reasonable approximation. Recall that, given a sample from the RBM distribution over full configurations, we can easily get a Monte Carlo approximation of this gradient, by reporting $\frac{\partial - E^\theta(C)}{\partial \theta}$ for that configuration C. Getting this sample from exactly the RBM distribution is intractable, but one can use prolonged Gibbs sampling from some starting configuration to get a sample from a hopefully fairly similar distribution. The procedure for getting a more or less unbiased estimate of $\frac{\partial \phi^-(\theta)}{\partial \theta}$ is as follows:

Pick any configuration of the RBM. Repeat the following two steps until out of time or patience: (1) update the state of the hidden units by sampling from the Bernoulli distributions with mean $\sigma \left(b_{h_i} + \sum_{j=1}^{n_v} s_{v_j}^{\vec{v}} \cdot c_{v_j, h_i} \right)$ for the i^{th} hidden unit; (2) update the state of the visible units by sampling from the Bernoulli distributions with mean $\sigma \left(b_{v_i} + \sum_{j=1}^{n_h} s_{h_j}^{\vec{h}} \cdot c_{v_i, h_j} \right)$ for the i^{th} visible unit. Call the last configuration C. Now report $\frac{\partial - E^\theta(C)}{\partial \theta}$, which is trivial to compute for both connection strengths and biases (see section 2.5.1). This is the more or less unbiased estimate of $\frac{\partial \phi^-(\theta)}{\partial \theta}$. To make it more

unbiased, perform more updates.

2.6 Contrastive Divergence

Contrastive Divergence (CD) is an algorithm that approximates the data likelihood gradient in RBMs. It is based on the idea that one often does not need to have an accurate estimate of the gradient, as long as the estimate is in the right direction. It was introduced in [4] and is briefly explained here.

Because $\frac{\partial \phi^+}{\partial \theta}$ is tractable exactly, the difficulty is approximating $\frac{\partial \phi^-(\theta)}{\partial \theta}$. CD uses the approximation algorithm described in section 2.5.2, but some details remain to be specified.

First observe that starting the Gibbs sampling process from a configuration that is close to the RBM distribution is a good idea. Intuitively, the Gibbs sampling moves slowly in the direction of the RBM distribution, and if the distance is small, it will sooner get close. After some training, assuming a reasonable training algorithm, the training data will be close to the RBM distribution, so starting the Gibbs sampling at the training data indeed achieves a quick start. CD takes as many estimates of $\frac{\partial \phi^-(\theta)}{\partial \theta}$ as there are training data points, and for each estimate the Gibbs sampling starts from another training data point.

Second, observe that the final estimate of $\frac{\partial \phi^+ - \phi^-}{\partial \theta}$ is the difference of $\frac{\partial -E^\theta(C)}{\partial \theta}$ for some C, obtained by sampling the state of the hidden units given some training data point, and the same $\frac{\partial -E^\theta(C)}{\partial \theta}$ for another C, sampled from a distribution that is hoped to be close to the RBM distribution. This estimate is then used by changing the parameters θ a small amount in the direction of the gradient. Such a procedure would work equally well if the gradient estimate, summed over all training cases, was a rather poor estimate but was likely to be in the same direction as the true gradient (as measured by the dot product between the two). The main justification for CD assumes that indeed the Gibbs

sampling process moves (slowly) in the direction of the RBM distribution: CD takes only a few steps in that direction, by doing a few Gibbs sampling updates, and then takes the aforementioned difference to be an estimate of at least the direction of the gradient (although the magnitude may be inaccurate).

In summary, to get an estimate of the direction of the gradient, using the CD algorithm, average the following over training data points $\vec{v}$: $\frac{\partial -E^\theta(C)}{\partial \theta}$ for some configuration C obtained by sampling states of the hidden units given the state of the visible units as specified by $\vec{v}$, minus $\frac{\partial -E^\theta(C)}{\partial \theta}$ for some configuration C obtained by running the Gibbs sampling process up for n iterations, starting with configuration $\vec{v}$. n stands for the number of times we update first the visible units and next the hidden units, not counting the first time we update the hidden units. This n is a parameter, and the algorithm for a particular choice of this parameter can be called CD-n. CD-1 is the fastest and works well for most purposes. Higher values of n give more accurate approximations to the true likelihood gradient, but require more computation time and have greater variance.

Chapter 3

CD not a gradient

One might suspect that although the expected value of CD does not give the ML gradient, it does give another gradient. [1] found that CD has fixed points (in their toy RBM, at least), but did not show whether CD is a gradient or not. It turns out that this is not the case. To prove this point, let us think about the nature of gradients, so that later we can find evidence that CD fails to conform to this.

3.1 About gradients in general

Take any differentiable scalar function $f(\theta)$ of a vector θ of n real numbers. Consider two parameter vectors, for example the all 2 vector (denoted θ_2) and the all 3 vector (denoted θ_3). Consider the difference $d = f(\theta_3) - f(\theta_2)$. Now consider the infinitely many paths from θ_2 to θ_3. The integral of $\frac{\partial f(\theta)}{\partial \theta}$ over any such path should be d. Consider the analogy with a mountain landscape: whatever route one takes between two places, the total amount one goes up minus the total amount one goes down must be the same, and must in fact be the difference in elevation between the two places.

Let us now assume that CD is indeed the gradient of some such function f. Let us also set up a simple RBM in which many calculations can be done exactly. Then, we can travel from some RBM parameter vector θ_0 to some other RBM parameter vector θ_1.

Along the way, we will integrate the amount that CD tells us we are going up or down on this underlying function f of which CD is supposed to be the gradient. We must now find that whatever route we take from θ_0 to θ_1, this integral must total to $f(\theta_1) - f(\theta_0)$. While we have no idea of what $f(\theta_1)$ or $f(\theta_0)$ might be, we do know that the difference between the two is a constant, so our integral should be the same, whatever route is taken. As we will see later, this is not always the case for CD, which implies that CD is not the gradient of any function.

3.2 Details of the experiment

The simple RBM is very simple indeed: it has only one visible unit and one hidden unit. The training data is simple, too: the only training case is the one where the visible unit is in state 1. The RBM parameter vector θ contains three numbers, namely the bias of the visible unit (denoted θ_v), the strength of the connection between the visible and the hidden unit (denoted θ_w), and the bias of the hidden unit (denoted θ_h), in that order. Now let us fix θ_0 to be $[0, 0, 0]$, i.e. both the biases and the connection strength are 0. θ_1 will be $[1, 1, 0]$, i.e. the bias of the visible unit is 1, the strength of the connection between the two units is also 1, and the bias of the hidden unit is 0. Let us also fix two paths from θ_0 to θ_1. The first path leads from $[0, 0, 0]$ straight to $[1, 0, 0]$, and from there straight to $[1, 1, 0]$. The second path leads from $[0, 0, 0]$ straight to $[0, 1, 0]$ and from there straight to $[1, 1, 0]$.

Now we wish to integrate the CD value over these two paths. In moving from one parameter vector to another by changing only one of the three parameters, we must integrate the CD would-be gradient on that parameter. Since we never change the bias to the hidden unit, we only need to calculate the CD would-be gradient w.r.t. the bias on the visible unit and w.r.t. the strength of the connection between the units.

3.3 Notation and preliminary definitions

Let us denote by $\mathrm{CD}_v(\theta)$ the CD would-be gradient w.r.t. the bias on the visible unit, at parameter vector θ. Similarly, $\mathrm{CD}_w(\theta)$ will be the CD would-be gradient w.r.t. the weight on the connection between the visible and the hidden unit, and $\mathrm{CD}_h(\theta)$ will be the same for the bias to the hidden unit (although we will not calculate that). To remind ourselves of the gradient notation, where CD is supposed to give the gradient of some function $f(\theta)$, we can now write $\frac{\partial f(\theta)}{\partial \theta} = [\mathrm{CD}_v(\theta), \mathrm{CD}_w(\theta), \mathrm{CD}_h(\theta)]$.

To calculate these CD expected values, we need some notation for the probabilities of configurations. The CD-1 algorithm specifies that we start with a training vector on the visible units and then do a Gibbs update on the hidden units. Let us call the resulting configuration T, which stands for training data, since on the visible units there is a training vector. After this first update of the hidden units, we do an update on the visible units, and another update on the hidden units. Let us call the configuration that we have now reached R, which stands for reconstruction, since on the visible units there is a reconstruction of the training vector. Let $P_{T:v=a,h=b}(\theta)$ denote the probability, under some parameter vector θ, that starting with our one training point, the T configuration has the visible unit in state a and the hidden unit in state b. Similarly, let $P_{R:v=a,h=b}(\theta)$ denote the probability, under θ, that, in the reconstruction configuration, the visible unit is in state a and the hidden unit is in state b. For example, $P_{T:v=0,h=0}(\theta) = 0$ for all θ, because our training data specifies that the visible unit starts in state 1, so with no updates on the visible unit it will never be in state 0. Let us also give a name to the probability of the visible unit being in some particular state in the reconstruction configuration: define $P_{R:v=a,h=*}(\theta) = P_{R:v=a,h=0}(\theta) + P_{R:v=a,h=1}(\theta)$.

3.4 Configuration probabilities

Recall that θ_v is the bias to the visible unit in parameter vector θ, θ_w is the strength of the connection between the two units, and θ_h is the bias to the visible unit. θ_h, however, is always zero, so we simply write a 0 instead.

Now we can write down the equations for the configuration probabilities, starting with the configuration probabilities of configuration T.

$$P_{T:v=0,h=0}(\theta) = 0 \tag{3.1}$$

$$P_{T:v=0,h=1}(\theta) = 0 \tag{3.2}$$

$$P_{T:v=1,h=1}(\theta) = \sigma(\theta_w) \tag{3.3}$$

$$P_{T:v=1,h=0}(\theta) = \sigma(-\theta_w) \tag{3.4}$$

The distribution over the state of the visible unit in the reconstruction configuration is as follows.

$$P_{R:v=1,h=*}(\theta) = \sigma(\theta_w) \cdot \sigma(\theta_w + \theta_v) + \sigma(-\theta_w) \cdot \sigma(\theta_v) \tag{3.5}$$

$$P_{R:v=0,h=*}(\theta) = \sigma(\theta_w) \cdot \sigma(-\theta_w - \theta_v) + \sigma(-\theta_w) \cdot \sigma(-\theta_v) \tag{3.6}$$

Last, the configuration probabilities of the reconstruction configuration are as follows.

$$P_{R:v=0,h=0}(\theta) = P_{R:v=0,h=*}(\theta) \cdot \sigma(-\theta_w) \tag{3.7}$$

$$P_{R:v=0,h=1}(\theta) = P_{R:v=0,h=*}(\theta) \cdot \sigma(\theta_w) \tag{3.8}$$

$$P_{R:v=1,h=0}(\theta) = P_{R:v=1,h=*}(\theta) \cdot \sigma(-\theta_w) \tag{3.9}$$

$$P_{R:v=1,h=1}(\theta) = P_{R:v=1,h=*}(\theta) \cdot \sigma(\theta_w) \tag{3.10}$$

3.5 The CD expected values

Now

$$\begin{aligned} \mathrm{CD}_v(\theta) = & P_{T:v=1,h=0}(\theta) + P_{T:v=1,h=1}(\theta) \\ & -P_{R:v=1,h=0}(\theta) - P_{R:v=1,h=1}(\theta) \end{aligned} \tag{3.11}$$

Because of our special training data distribution, this reduces to

$$\mathrm{CD}_v(\theta) = 1 - P_{R:v=1,h=0}(\theta) - P_{R:v=1,h=1}(\theta) \tag{3.12}$$

This can be rewritten as

$$\mathrm{CD}_v(\theta) = 1 - P_{R:v=1,h=*}(\theta) \tag{3.13}$$

In a similar fashion, we have

$$\mathrm{CD}_w(\theta) = P_{T:v=1,h=1}(\theta) - P_{R:v=1,h=1}(\theta) \tag{3.14}$$

3.6 The integrals

We have four integrals to calculate, for the four moves we make in our parameter landscape. After calculating these four separately, we will add them up appropriately.

3.6.1 From [0,0,0] to [1,0,0]

This is a move where we change only the bias to the visible unit, so we integrate the (supposedly) gradient that CD gives for that bias.

$$\begin{aligned}
\int_0^1 \mathrm{CD}_v([x,0,0])dx = \\
\int_0^1 1 - P_{R:v=1,h=*}([x,0,0])dx = \\
\int_0^1 1 - \left(\sigma(\theta_w)\cdot\sigma(\theta_w+\theta_v) + \sigma(-\theta_w)\cdot\sigma(\theta_v)\right)([x,0,0])dx = \\
\int_0^1 1 - \left(\sigma(0)\cdot\sigma(x) + \sigma(-0)\cdot\sigma(x)\right)dx = \\
\int_0^1 1 - \left(\frac{1}{2}\cdot\sigma(x) + \frac{1}{2}\cdot\sigma(x)\right)dx = \\
\int_0^1 1 - \sigma(x)dx = \\
\int_0^1 1 - \frac{1}{1+e^{-x}}dx = \\
[x - \log(1+e^x)]_0^1 = \\
1 - \log(e+1) + \log(2) \qquad (3.15)
\end{aligned}$$

3.6.2 From [0,1,0] to [1,1,0]

This is again an integral of CD_v, but this time the strength of the connection between the units is 1, instead of 0.

$$
\begin{aligned}
\int_0^1 \mathrm{CD}_v([x,1,0])dx = \\
\int_0^1 1 - P_{R:v=1,h=*}([x,1,0])dx = \\
\int_0^1 1 - \left(\sigma(\theta_w)\cdot\sigma(\theta_w+\theta_v) + \sigma(-\theta_w)\cdot\sigma(\theta_v)\right)([x,1,0])dx = \\
\int_0^1 1 - \left(\sigma(1)\cdot\sigma(1+x) + \sigma(-1)\cdot\sigma(x)\right)dx = \\
1 - \int_0^1 \sigma(1)\cdot\sigma(1+x)dx - \int_0^1 \sigma(-1)\cdot\sigma(x)dx = \\
1 - \sigma(1)\cdot\int_0^1 \sigma(1+x)dx - \sigma(-1)\cdot\int_0^1 \sigma(x)dx = \\
1 - \sigma(1)\cdot\left[\log(1+e^{1+x})\right]_0^1 - \sigma(-1)\cdot\left[\log(1+e^x)\right]_0^1 = \\
1 - \sigma(1)\cdot\left(\log(1+e^2) - \log(1+e)\right) - \\
\sigma(-1)\cdot(\log(1+e) - \log(2)) \qquad (3.16)
\end{aligned}
$$

3.6.3 From [0,0,0] to [0,1,0]

This is an integral of CD_w from 0 to 1, with both of the biases are fixed to 0.

$$
\begin{aligned}
\int_0^1 \mathrm{CD}_w([0,x,0])dx = \\
\int_0^1 P_{T:v=1,h=1}([0,x,0]) - P_{R:v=1,h=1}([0,x,0])dx = \\
\int_0^1 \sigma(x) - P_{R:v=1,h=*}([0,x,0]) \cdot \sigma(x)dx = \\
\int_0^1 \sigma(x) - (\sigma(x) \cdot \sigma(x) + \sigma(-x) \cdot \sigma(0)) \cdot \sigma(x)dx = \\
\int_0^1 \sigma(x)dx - \int_0^1 \sigma(x)^3 dx - \frac{1}{2}\int_0^1 \sigma(x) \cdot \sigma(-x) \cdot dx = \\
[\log(1+e^x)]_0^1 - \left[\frac{2\log(1+e^x)\,(1+e^x)^2 + 4e^x + 3}{2\,(1+e^x)^2}\right]_0^1 - \\
\frac{1}{2}\left[-\frac{1}{1+e^x}\right]_0^1 = \\
\log(1+e) - \log(2) - \frac{2\log(1+e)(1+e)^2 + 4e + 3}{2(1+e)^2} + \\
\log(2) + \frac{7}{8} + \frac{1}{2+2e} - \frac{1}{4} = \\
\frac{4e+3}{2(1+e)^2} + \frac{7}{8} + \frac{1}{2+2e} - \frac{1}{4} = \\
\frac{5}{8} - \frac{2+3e}{2\cdot(1+e)^2}
\end{aligned}
\tag{3.17}
$$

3.6.4 From [1,0,0] to [1,1,0]

The fourth integral is an integral of CD_w from 0 to 1, with the bias to the visible unit fixed at 1.

$$\int_0^1 \mathrm{CD}_w([1,x,0])dx =$$
$$\int_0^1 P_{T:v=1,h=1}([1,x,0]) - P_{R:v=1,h=1}([1,x,0])dx =$$
$$\int_0^1 \sigma(x) - (\sigma(x)\cdot\sigma(x+1) + \sigma(-x)\cdot\sigma(1))\cdot\sigma(x)dx =$$
$$\int_0^1 \sigma(x)dx - \int_0^1 \sigma(x)^2\cdot\sigma(x+1)dx - \sigma(1)\cdot\int_0^1 \sigma(x)\cdot\sigma(-x)dx =$$
$$-\left[\frac{\frac{(e-1)e}{1+e^x} + (e-2)e\cdot\log(1+e^x) + \log(1+e^{1+x})}{(e-1)^2}\right]_0^1 -$$
$$\sigma(1)\cdot\left[-\frac{1}{1+e^x}\right]_0^1 + [\log(1+e^x)]_0^1 =$$
$$\log(1+e) - \log(2) - \frac{\frac{(e-1)e}{1+e} + (e-2)e\cdot\log(1+e) + \log(1+e^2)}{(e-1)^2} +$$
$$\frac{\frac{(e-1)e}{2} + (e-2)e\cdot\log(2) + \log(1+e)}{(e-1)^2} + \sigma(1)\cdot\frac{1}{1+e} - \frac{1}{2}\cdot\sigma(1) \tag{3.18}$$

3.7 Conclusion

Now that we know the value of these integrals, we can check whether indeed the integral from $[0,0,0]$ to $[1,1,0]$ is independent of the path that is taken. When we add up the expressions we found for the integrals, no nice and simple expression comes out, so all we can do is numerically evaluate it. We find that going from $[0,0,0]$ to $[1,0,0]$ and from there to $[1,1,0]$ gives an integral of $0.379885 + 0.131057 \approx 0.510942$, while going from $[0,0,0]$ to $[0,1,0]$ and from there to $[1,1,0]$ gives an integral of $0.257753 + 0.238388 \approx 0.49614$. If CD truly were the derivative of some function, these two would sum to the same. Thus we conclude that CD is not the derivative of any function.

There is an important corollary to this. In our small RBM, if we would go from $[0,0,0]$ to $[1,0,0]$, from there to $[1,1,0]$, from there to $[0,1,0]$, and from there back to $[0,0,0]$, then we have seen, according to the CD approximations of gradient, an increase in the value of the objective function, even though clearly we are back at the same place. This

increase is approximately $0.510942 - 0.496148 = 0.014794$. Some thought now shows that a function $g(\theta)$ can be constructed, such that if one were to add the true derivatives of g to values that CD calculates, then the optimization would go around this square forever, believing that it gained 0.014794 in the objective function value during each pass.

We do occasionally add gradients of various functions to the gradient of our main objective function. The best known example is weight decay, but it also happens when there are two things of which we are maximizing the combined score. The current finding shows that if one is unfortunate in the choice of additional functions to include in the optimization, the $\mathrm{CD}(\theta) + \frac{\partial g(\theta)}{\partial \theta}$ optimization will not converge.

It must be emphasized that this finding is highly theoretical and may be of no practical relevance. As noted before, [1] found that in their toy experiments, CD did have fixed points, which were even somewhat close to fixed points of maximum likelihood learning. The current finding points out that CD need not always have fixed points, and that in our toy experiment, there are functions $g(\theta)$ such that CD combined with the gradient of g has no fixed points. However, the author has not yet encountered situations in which CD alone has no fixed points.

Chapter 4

Learning with variational inference

In RBMs, inference of hidden state given visible state is tractable, but learning is still somewhat difficult. In a different class of models, well exemplified by Sigmoid Belief Networks (SBNs, [8]), inference is intractable but once we have a sample from this intractable posterior, learning is easy. When working with this class of models, variational inference can, sometimes, stand in for proper inference (see [10] for an introduction to variational methods). In this section, the class of models is described, the basis of variational methods is described, a tractable Monte Carlo approximation to the gradient w.r.t. the inference parameters is introduced, and two examples of these models are briefly mentioned. A tractable Monte Carlo approximation to the gradient w.r.t. the parameters of the generative model has long been known and is mentioned here for completeness. Note that most of this section is review of old work: variational methods have long been known. The only new part is the tractable Monte Carlo approximation to the gradient w.r.t. the inference parameters.

4.1 A class of models

Consider probability density models that have a number of configurations, and that assign to each configuration a probability. RBMs are an example of this. As in RBMs,

the configurations must include some information that is considered 'visible', so that each possible data point is represented by the set of configurations of which the visible part indicates that data point. We require that the probability of configurations be tractable to calculate, up to a possibly unknown multiplicative factor. We also require that the log probability gradient for configurations be tractable: if θ denotes the model description, and C is any configuration, $\frac{\partial \log(P(C))}{\partial \theta}$ must be tractable. Note that SBNs meet both of these requirements, while RBMs only meet the first.

For our variational inference, we will need a second model, that describes a variational posterior distribution over states given the visible part. For example, a table would do, but there are more sophisticated methods. We require the ability to sample from this variational posterior, the ability to calculate probability of configurations in the variational posterior, up to possibly unknown multiplicative factor (which may be different for each configuration of the visible units), and again the log probability gradient, which now is conditional probability: $\frac{\partial \log(Q(C|\text{vis}))}{\partial \theta}$, where θ describes the inference model $Q(\cdot)$.

The reader may verify that as long as the variational inference gives the true posterior, these requirements allow us to obtain a Monte Carlo approximation of the log probability gradient of a training data set. In fact, we only need two of the mentioned requirements: the ability to sample from the (now correct) posterior, and the availability of the configuration log probability gradient of the density model. Now follows a description of the situation with incorrect posteriors.

4.2 Some pieces of statistical physics

First, it is time to bring in some basic statistical physics. Consider a set S of configurations of a system. Let each configuration C have an energy $E(C)$ in that system. A quantity called free energy of the set, $F(S)$, is defined as

$$F(S) = -\log\left(\sum_{C\in S} e^{-E(C)}\right) \tag{4.1}$$

There are many uses and interpretations of the concept of free energy. For now, just observe how it makes the following probability function sum to 1:

$$P(C) = \frac{e^{-E(C)}}{e^{-F(S)}} \tag{4.2}$$

With this probability distribution, we can rewrite the free energy into the following equivalent form:

$$F(S) = \sum_{C\in S} P(C)\cdot(E(C) + \log(P(C))) \tag{4.3}$$

From this we come to variational free energy (VFE). It works with any distribution $Q(\cdot)$ over configurations. The reader may verify that the two descriptions are indeed equal.

$$V(S,Q) = \sum_{C\in S} Q(C)\cdot(E(C) + \log(Q(C))) \tag{4.4}$$

$$= F(S) + K(Q,P) \tag{4.5}$$

In the second definition, K stands for the Kullback-Leibler divergence of $P(\cdot)$ from $Q(\cdot)$, which is defined as

$$K(P_1,P_2) = \sum_{C\in S} P_1(C)\log\frac{P_1(C)}{P_2(C)} \tag{4.6}$$

With this bit of statistical physics, a meaningful and tractable objective function can be defined.

4.3 The new objective function

The most common objective function for probability density modeling is log likelihood. Training data log likelihood is $L = \sum_{\vec{v}} P_{\text{train}}(\vec{v}) \log(P(\vec{v}))$. Let us assume for simplicity that there is only one training point: $L = \log(P(\vec{v}))$. The gradient of this objective function is intractable, so a different one is needed: one that uses the variational inference.

We have a density model that assigns a probability to each configuration, but the statistical physics is about energies, so first we need those. The energy of a configuration C will be $E(C) = -\log(P(C))$. Note that this makes the free energy of the set of all configurations be 0. Let $S(\vec{v})$ be the set of configurations of which the visible part indicates our one training point. Now clearly, $\log(P(\vec{v})) = -F(S(\vec{v}))$. We also have a variational posterior $Q(\cdot|\vec{v})$. The new objective function will be $-V(S(\vec{v}), Q(\cdot|\vec{v}))$. Note this this is at most $\log(P(\vec{v}))$, and is exactly that only when the inference distribution is correct: $\forall_C Q(C|\vec{v}) = P(C|\vec{v})$.

4.4 The gradient

The old objective function, log probability, does not mention the inference model, so it does not depend on its parameters in any way. The new one, however, does. Let us call the parameters of the inference model θ_{inf}, while the parameters of the main generative model will be θ_{gen}. The gradient w.r.t. the inference parameters is calculated separately from the gradient w.r.t. the generative parameters.

4.4.1 The gradient w.r.t. θ_{gen}

Thanks to the availability of inference, albeit incorrect inference, this gradient is tractable. Still assume just one training point, called $\vec{v}$.

$$\frac{\partial - V(S(\vec{v}), Q(\cdot|\vec{v}))}{\partial \theta_{\text{gen}}} = \tag{4.7}$$

$$-\frac{\partial \sum_{C \in S(\vec{v})} Q(C|\vec{v}) \cdot (E(C) + \log(Q(C|\vec{v})))}{\partial \theta_{\text{gen}}} = \tag{4.8}$$

$$-\frac{\partial \sum_{C} Q(C|\vec{v}) \cdot (E(C) + \log(Q(C|\vec{v})))}{\partial \theta_{\text{gen}}} = \tag{4.9}$$

$$-\sum_{C} Q(C|\vec{v}) \cdot \frac{\partial E(C)}{\partial \theta_{\text{gen}}} = \tag{4.10}$$

$$\sum_{C} Q(C|\vec{v}) \cdot \frac{\partial \log(P(C))}{\partial \theta_{\text{gen}}} \tag{4.11}$$

Equation 4.8 is obtained by expanding V. Equation 4.9 is obtained by observing that only configurations in $S(\vec{v})$ will have variational posterior probability greater than zero (assuming any reasonable posterior). Equation 4.10 is obtained by observing that $Q(C|\vec{v})$ does not depend on θ_{gen}. Finally, equation 4.11 is obtained by switching back from energy notation to probability notation. In equation 4.11 one can see that given the ability to sample from the variational posterior, and the tractability of the log probability gradient for configurations in the generative model, we have a nice Monte Carlo approximation: sample a configuration C from the variational posterior $Q(\cdot|\vec{v})$, and then take $\frac{\partial \log P(C)}{\partial \theta_{\text{gen}}}$ as the approximation.

4.4.2 The gradient w.r.t. θ_{inf}

For this gradient, we use the second notation of variational free energy, $V(S, Q(\cdot)) = F(S) + K(Q, P)$. This is more convenient because $F(S)$ does not depend on the inference distribution. Look under these equations for more detail on why they are valid.

$$\frac{\partial - V(S(\vec{v}), Q(\cdot|\vec{v}))}{\partial \theta_{\text{inf}}} = \tag{4.12}$$

$$-\frac{\partial F(S(\vec{v})) + K(Q(\cdot|\vec{v}), P(\cdot|\vec{v}))}{\partial \theta_{\text{inf}}} = \tag{4.13}$$

$$-\frac{\partial K(Q(\cdot|\vec{v}), P(\cdot|\vec{v}))}{\partial \theta_{\text{inf}}} = \tag{4.14}$$

$$-\frac{\partial \sum_C Q(C|\vec{v}) \cdot (\log(Q(C|\vec{v})) - \log(P(C|\vec{v})))}{\partial \theta_{\text{inf}}} = \tag{4.15}$$

$$-\sum_C \frac{\partial Q(C|\vec{v}) \cdot (\log(Q(C|\vec{v})) - \log(P(C|\vec{v})))}{\partial \theta_{\text{inf}}} = \tag{4.16}$$

$$-\sum_C \frac{\partial Q(C|\vec{v})}{\partial \theta_{\text{inf}}} \cdot (\log(Q(C|\vec{v})) - \log(P(C|\vec{v}))) -$$

$$\sum_C Q(C|\vec{v}) \cdot \frac{\partial \log(Q(C|\vec{v}))}{\partial \theta_{\text{inf}}} = \tag{4.17}$$

$$-\sum_C \frac{\partial Q(C|\vec{v})}{\partial \theta_{\text{inf}}} \cdot (\log(Q(C|\vec{v})) - \log(P(C|\vec{v}))) = \tag{4.18}$$

$$-\sum_C \frac{\partial Q(C|\vec{v})}{\partial \theta_{\text{inf}}} \cdot (F_{logQ,\vec{v}}(C) + \text{const}_{\vec{v}} - \log(P(C|\vec{v}))) = \tag{4.19}$$

$$-\sum_C \frac{\partial Q(C|\vec{v})}{\partial \theta_{\text{inf}}} \cdot (F_{logQ,\vec{v}}(C) + \text{const}_{\vec{v}} - \log(P(C)) + \log(P(\vec{v}))) = \tag{4.20}$$

$$-\sum_C \frac{\partial Q(C|\vec{v})}{\partial \theta_{\text{inf}}} \cdot (F_{logQ,\vec{v}}(C) + \text{const}_{\vec{v}} - \log(P(C))) = \tag{4.21}$$

$$-\sum_C \frac{\partial Q(C|\vec{v})}{\partial \theta_{\text{inf}}} \cdot (F_{logQ,\vec{v}}(C) + \text{const}_{\vec{v}} - F_{logP}(C)) = \tag{4.22}$$

$$-\sum_C \frac{\partial Q(C|\vec{v})}{\partial \theta_{\text{inf}}} \cdot (F_{logQ,\vec{v}}(C) - F_{logP}(C))$$

$$-\sum_C \frac{\partial Q(C|\vec{v})}{\partial \theta_{\text{inf}}} \cdot \text{const}_{\vec{v}} = \tag{4.23}$$

$$-\sum_C \frac{\partial Q(C|\vec{v})}{\partial \theta_{\text{inf}}} \cdot (F_{logQ,\vec{v}}(C) - F_{logP}(C)) = \tag{4.24}$$

$$-\sum_C Q(C|\vec{v}) \cdot \frac{\partial \log(Q(C|\vec{v}))}{\partial \theta_{\text{inf}}} \cdot (F_{logQ,\vec{v}}(C) - F_{logP}(C)) \tag{4.25}$$

Equation 4.13 is obtained by expanding V into free energy and KL divergence. Equation 4.14 is found by dropping the free energy, which does not depend on θ_{inf}. Equation 4.15 is found by expanding the expression of KL divergence. Equation 4.16 is a simple

rewrite. Equation 4.17 points out that change in θ_{inf} affects the KL divergence in two ways. Equation 4.18 drops the second term of the preceding equation, because it is easily shown to be zero. Equation 4.19 uses the assumed fact that we have a tractable function, here called $F_{logQ,\vec{v}}(\cdot)$, that calculates the log probability of configuration C in the variational posterior of $\vec{v}$, up to some possibly unknown additive constant, which may depend on $\vec{v}$. Equation 4.20 is found by expanding the conditional probability. Equation 4.21 is found by including $\log(P(\vec{v}))$ in $\text{const}_{\vec{v}}$. Equation 4.22 uses the assumed tractable $F_{logP}(\cdot)$ that gives $\log(P)$ up to some additive constant, while that constant disappears into $\text{const}_{\vec{v}}$. Equation 4.23 isolates $\text{const}_{\vec{v}}$. Equation 4.24 drops the second term of the preceding equation, because that term is zero. Equation 4.25 switches to log probability gradient and enables the Monte Carlo approximation: take a sample from $Q(C|\vec{v})$, and from there use three functions that are each assumed tractable, to get the estimate of $\frac{\partial -V(S(\vec{v}),Q(\cdot|\vec{v}))}{\partial \theta_{\text{inf}}}$.

In reality, we have multiple training points. As with likelihood, the VFE objective function is simply averaged over training points, and the same happens to the gradient.

4.5 Two examples

4.5.1 SBN

One example from this class of models has been mentioned already: Sigmoid Belief Networks, combined with some suitable inference model. One could, for example, set up an SBN for inference, in which the dependencies are in the direction opposite to the one in the generative model. SBNs meet all of the requirements, because exact probability of configurations is tractable, as is sampling. Thus, SBNs exceed the requirements by also allowing sampling from the main generative model, as well as exact probability calculation (as opposed to probability up to an unknown normalizing constant).

4.5.2 SBN-RBM

A second example is a model built as a combination of an SBN and an RBM, introduced in [3], combined with an SBN for inference. This model provides neither tractable sampling from the generative model, nor exact probability calculation in the generative model, but neither are necessary for the present gradient calculation, and the model has advantages that allow good initialization. One should note that for this model, $\frac{\partial \log(P(C))}{\partial \theta_{\text{gen}}}$ is not available exactly, but the CD approximation can be used for that.

4.6 Usability

Some practice and some theory suggest that this algorithm may be of limited value, due to the high variance of the gradient estimates. A different approximation algorithm, called Wake-Sleep [5], has more bias (i.e. it is not unbiased) and empirically less variance. One might try to take advantage of the absence of bias in the present algorithm by using it in the final stage of training, to do some fine-tuning. The details and elaboration on all of this, however, are future work.

Chapter 5

Early stopping on likelihood for RBMs

Gradient-based learning requires more than just an algorithm to calculate a gradient. Another component is an algorithm that decides which model to pick from the series found during optimization; this is also known as early stopping. Usually, one selects the model that has the best performance on a separate validation data set. For RBMs, however, this simple approach is infeasible, because the objective function that we optimize for is intractable. In this section, some alternatives are studied. First, there will be some comments about commonly used alternatives. Second, there will be another method, viewed in several different ways. And in conclusion, the way will be pointed to the ultimately best method.

5.1 The problem

As was mentioned before, the most commonly used objective function is average log likelihood: $\phi = \sum_{\vec{v}} P_{\text{train}}(\vec{v}) \log(P(\vec{v}))$. In RBMs, this is intractable, because the normalizing constant of RBMs is intractable.

5.2 Tractable alternatives

One approach is to use another goodness measure, and hope that it resembles our original measure, or that at least still the selected model will be good. One should keep in mind that these alternative objective measures are not used for training. They are evaluated only on the validation set, and we never use their gradients. A plot demonstrating performance of these methods is included at the end of this section.

5.2.1 Reconstruction error

The CD algorithm specifies how to get a reconstruction of each data point: the reconstruction distribution of some data point $\vec{v_1}$ is $P_{\text{rec},\vec{v_1}}(\vec{v_2}) = \sum_{\vec{h}} P(\vec{h}|\vec{v_1})P(\vec{v_2}|\vec{h})$, or in words, the probability that CD-1 will use $\vec{v_2}$ as negative data when positive data is $\vec{v_1}$. One might suspect that for a well trained RBM, this reconstruction of training data is usually similar to the training data itself. Note that this is not necessarily true. The quality of fit of an RBM has more to do with the reconstruction distribution averaged over all training points: that distribution should be similar to the training data distribution, but individual training points may be reconstructed as different training points. Anyway, it is an empirical finding that as RBM training proceeds, reconstructions of training data get closer to their original. This suggests setting up some measure of reconstruction error (RE), and then doing early stopping based on validation data reconstruction error.

An upside of this approach is that calculating this takes little time: simply sample some $\vec{h}$ from $P(\vec{h}|\vec{v_1})$, sample some $\vec{v_2}$ from $P(\vec{v_2}|\vec{h})$, and count the number of visible units that have different states in $\vec{v_1}$ and $\vec{v_2}$. A disadvantage is that there is randomness in this procedure, but to get a more reliable estimate one may repeat the procedure. A more serious consideration than these practical issues is whether this approach will result in stopping at the right time. Empirically, it appears not to do so: stopping occurs too late. There is an intuitive explanation of why this happens.

As training goes on, the connection strengths of the RBM tend to grow. In a manner quite independent of quality of fit, large weights tend to produce accurate reconstructions or, equivalently, slow mixing of the Markov Chain. Consider some hidden unit h with a strong positive connection to some visible unit v. Due to the strong connection, the probability that h will have the same state as v in samples from $P(\vec{h}|\vec{v_1})$, is large. The connection is symmetric, so the same goes for sampling from $P(\vec{v_2}|\vec{h})$. This shows that with large weights, it is quite probable that the reconstruction is similar to the original.

Quality of fit also reduces reconstruction error, and this is the effect that we are looking for and would like to isolate. But the effect of simply the size of the weights is significant, and results in stopping much too late. This makes stopping on validation data reconstruction error a poor approach.

5.2.2 Pseudo likelihood

A second approach is pseudo likelihood (PL). The pseudo likelihood of some configuration $\vec{v}$ of the visible units is $P_{\mathrm{PL}}(\vec{v}) = \prod_i P(\vec{v}|\vec{v}_{-i})$, where $\vec{v}_{-i}$ stands for the configuration $\vec{v}$ on all visible units except the i^{th}. One could take the average log PL of the validation data set as the alternative goodness measure. This is tractable in RBMs, but takes significantly more time than RE because of the cost of calculating exponentials and logarithms. It has the advantage over RE that it is deterministic. Like RE, however, PL tends to result in stopping too late, although the effect is less severe.

5.3 Approximating likelihood

Instead of using tractable alternatives, one could try to approximate true likelihood, and do early stopping on that approximation.

The first observation to be made is that truly approximating likelihood is very difficult, because it implies approximating the normalizing constant of the RBM, which is at the

moment considered a difficult problem. For early stopping, however, nobody needs the true value of the objective function: only differences between the scores of several (similar) models are needed, and this might be easier.

In this section, one basic algorithm is presented, and then viewed in a few different ways, each of which suggests a variation. First, however, we need some notation and conventions.

5.3.1 Preliminaries

Let us denote the validation data distribution by $P_{\text{val}}(\cdot)$. The validation score, which is a function of the validation distribution and the RBM parameter setting θ, is average log likelihood: $S(\theta, P_{\text{val}}) = \sum_{\vec{v}} P_{\text{val}}(\vec{v}) \log(P^{\theta}(\vec{v}))$. Not only configuration probability, but every concept from RBMs must from now on have a superscript of θ, because we are dealing with different RBMs. Therefore, we will also see Z^{θ} and $E^{\theta}(\cdot)$.

The task is formalized as follows. Given a sequence of RBMs, each only infinitesimally different from its neighbors, produce estimates of the difference in validation score between each pair of neighbors. Given estimates of these differences, one can make the usual early stopping plot, from which it is easy to pick a good model. The assumption of infinitesimal differences is, of course, one of those convenient assumptions that everybody knows to be false.

The task for a specific pair of neighbors is described conveniently by θ_1 and θ_2, the parameters of the first and the second RBM, respectively. The supposedly infinitesimal difference $\theta_2 - \theta_1$ will be abbreviated as $d\theta$.

5.3.2 The algorithm

The first description of the algorithm is the one that comes to mind most naturally. Later follows a more thorough analysis of what it really does.

The objective function, validation data log likelihood, is intractable, but we do have

a reasonable approximation to its gradient: Contrastive Divergence. And whoever has a gradient has enough information to estimate differences. Let us denote the CD-approximated gradient, which is a function of the training data distribution $P_{\text{train}}(\cdot)$ and of the RBM parameters θ, as follows: $\hat{\Delta}_{CD}(\theta, P_{\text{train}}(\cdot)) \approx \frac{\partial \sum_{\vec{v}} P_{\text{train}}(\vec{v}) \log(P^{\theta}(\vec{v}))}{\partial \theta} = \frac{\partial S(\theta, P_{\text{train}})}{\partial \theta}$. To estimate $S(\theta_2, P_{\text{val}}) - S(\theta_1, P_{\text{val}})$, simply take the dot product of $d\theta$ with the estimate of $\frac{\partial S(P_{\text{val}}, \theta)}{\partial \theta}\big|_{\theta_1}$. If instead of a deterministic $\hat{\Delta}_{CD}(P_{\text{val}}, \theta)$, we get an unbiased estimate of that (as is the case when using CD), the estimate of $S(P_{\text{val}}, \theta_2) - S(P_{\text{val}}, \theta_1)$ will also be unbiased, because of linearity of the dot product.

This algorithm is easy to implement and works reasonably well, as can be seen in the plot at the end of this section. However, like PL, it results in stopping a bit too late. The most logical explanation for this is some unknown but systematic difference between the CD approximated gradient and the true gradient.

5.3.3 A more thorough analysis

The CD algorithm has two parts: the estimate of a 'positive' gradient and the estimate of a 'negative' gradient. Let us recall why that is so.

$$\begin{aligned} S(\theta, P_{\text{val}}) &= \sum_{\vec{v}} P_{\text{val}}(\vec{v}) \log(P^{\theta}(\vec{v})) \\ &= \sum_{\vec{v}} P_{\text{val}}(\vec{v}) \log\left(\frac{\sum_{\vec{h}} e^{-E^{\theta}(\vec{v},\vec{h})}}{Z^{\theta}}\right) \\ &= -\log(Z^{\theta}) + \sum_{\vec{v}} P_{\text{val}}(\vec{v}) \log\left(\sum_{\vec{h}} e^{-E^{\theta}(\vec{v},\vec{h})}\right) \\ &= -\phi^{-}(\theta) + \phi^{+}(\theta, P_{\text{val}}) \end{aligned} \tag{5.1}$$

The symbols ϕ^+ and ϕ^- were introduced in exactly this way in section 2.3.

ϕ^+ is completely tractable, so $\phi^+(\theta_2, P_{\text{val}}) - \phi^+(\theta_1, P_{\text{val}})$ is tractable even when $d\theta$ is not infinitesimally small. Our remaining concern is $\phi^-(\theta_2) - \phi^-(\theta_1)$, or $\frac{\partial \phi^-(\theta)}{\partial \theta}\big|_{\theta_1}$ (assuming

infinitesimally small $d\theta$). Recall equation 2.3:

$$\frac{\partial \phi^-(\theta)}{\partial \theta} = \sum_C P^\theta(C) \frac{\partial - E^\theta(C)}{\partial \theta} \tag{5.2}$$

From this equation we see that to get an unbiased estimate of $\frac{\partial \phi^-(\theta)}{\partial \theta}$, we need a sample configuration C from $P^\theta(\cdot)$. The algorithm described above does that by doing some Gibbs sampling, starting at the training data, and hoping to move away from the training data towards the RBM distribution. But there might be other ways to do it; ways that might get a sample from a distribution closer to the RBM distribution.

If we do use CD, we can at least use CD-n for some $n > 1$. The greater this n, the closer the approximation to the true likelihood gradient. It was said before that although CD-1 gives a bad approximation of the true gradient, it often still gets the direction more or less correct, but the fact remains that the greater this n, the better the approximation, and that includes the approximation to the direction.

5.3.4 Alternative ways to get samples

There are other ways to get samples from $P^\theta(\cdot)$. The quality of our approximation of $\frac{\partial \phi^-(\theta)}{\partial \theta}$ depends only on how close our sampling distribution is to the RBM distribution, so any improvements must be achieved by changing the sampling process.

The canonical way to obtain such a sample is to start a Gibbs sampling procedure with some randomly chosen initial configuration. CD has a major improvement over that method by starting not at a random configuration but at a training data configuration. But there is an even better way to initialize. Because we are estimating the gradient for many different θ, all very similar to each other, we can start the Gibbs sampling procedure at a configuration at which it left off for the previous θ. Since the θ values are similar, the previous RBM distribution will be a very good approximation to the current one, and this way, as we analyze each θ, we get closer and closer to the RBM distribution,

even though it is changing. Note that, unfortunately, this method can give estimators that have greater variance than those taken from the normal CD approach, because in the normal CD approach, the Markov chain is always started from the training data, while with this new approach, the chain is started from different points each time.

This method, which can reasonably be called CD-*memory*, works very well for the present purpose (see the plot at the end of this section). To the author's knowledge, it has not been tested for obtaining a gradient on training data. It might work well there, too, but there is also a chance that samples from the previous RBM distribution will not be good samples from the current RBM distribution, because the gradient update that was calculated based on them has made these configurations less probable under the RBM-in-training.

5.3.5 Concluding notes

Given that $d\theta$ is not quite infinitesimally small, there are some tricks for better approximating the gradient between θ_1 and θ_2. See for example [7].

Note also that $\phi^-(\theta_2) - \phi^-(\theta_1)$ is a difference of logarithms of normalizing constants, or, equivalently, a ratio of normalizing constants. The task of approximating ratios of normalizing constants is well studied in the field of statistics, see for example [2]. Trying to invent new methods for it without using the knowledge that has been built up by much excellent research in the field of statistics seems a waste of effort. Therefore, I conclude this section by noting that the ultimate way of doing early stopping on validation likelihood for RBMs is to ask a good statistician for his favorite way of estimating ratios of normalizing constants. This method, however, has not (yet) been tried by the author.

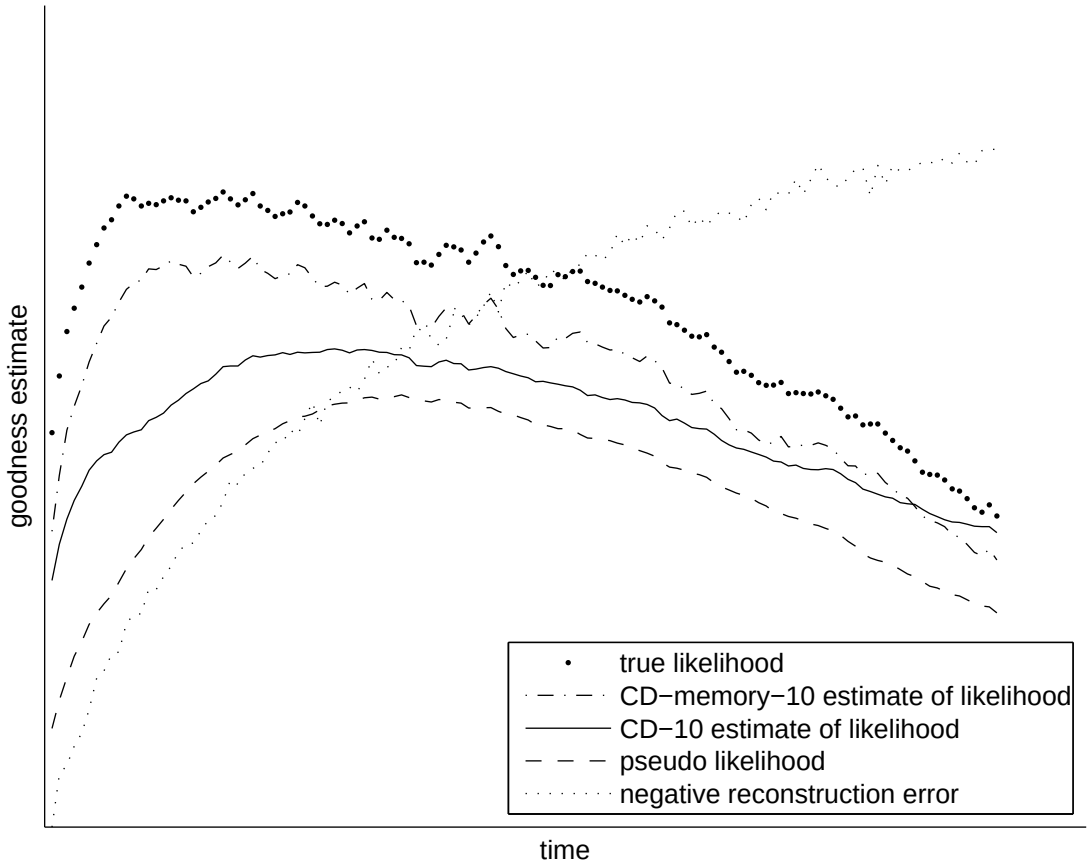

Figure 5.1: This plot allows one to compare the various early stopping methods discussed here. An RBM was trained on images of handwritten twos from the USPS data set. On a separate validation set, these measures were evaluated. The RBM had only 20 hidden units, so the true likelihood could be calculated. Note that the scale of the Y axis differs across measures (from bits to to pseudo bits to squared pixel reconstruction errors), so numbers have been omitted. Also note that as the CD-based measures only estimate differences, the y=0 line is of no significance. The only thing that can be compared in this plot is where the peaks are. In this experiment, validation data reconstruction error only decreased, but RE is not as bad as it looks here: with weight decay, it eventually starts to increase (the current experiment was done without weight decay).

Bibliography

[1] M.A. Carreira-Perpinan and G.E. Hinton. On contrastive divergence learning. *Artificial Intelligence and Statistics, 2005*, 2005.

[2] A. Gelman and X.L. Meng. Simulating normalizing constants: from importance sampling to bridge sampling to path sampling. *Statist. Sci*, 13(2):163–185, 1998.

[3] G. E. Hinton, S. Osindero, and Y. W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18, 2006.

[4] G.E. Hinton. Training Products of Experts by Minimizing Contrastive Divergence. *Neural Computation*, 14(8):1771–1800, 2002.

[5] G.E. Hinton, P. Dayan, B.J. Frey, and R.M. Neal. The wake-sleep algorithm for unsupervised neural networks. *Science*, 268(5214):1158–1161, 1995.

[6] G.E. Hinton and T.J. Sejnowski. Learning and relearning in Boltzmann machines. *Mit Press Computational Models Of Cognition And Perception Series*, pages 282–317, 1986.

[7] J.R.R.A. Martins, P. Sturdza, and J.J. Alonso. The complex-step derivative approximation. *ACM Transactions on Mathematical Software (TOMS)*, 29(3):245–262, 2003.

[8] R.M. Neal. Connectionist learning of belief networks. *Artificial Intelligence*, 56(1):71–113, 1992.

[9] P. Smolensky. *Information processing in dynamical systems: foundations of harmony theory.* MIT Press Cambridge, MA, USA, 1986.

[10] R.S. Zemel. *A Minimum Description Length Framework for Unsupervised Learning.* PhD thesis, University of Toronto, 1993.