Towards Semi-Supervised Classification with Markov Random Fields

Xiaojin Zhu Zoubin Ghahramani

June 2002
CMU-CALD-02-106

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

We investigate the use of Boltzmann machines in semi-supervised classification. We treat the labeled / unlabeled dataset as a Markov random field, and derive a Boltzmann machine learning algorithm for it to learn the feature weights, label noise and labels for unlabeled data all at once. We present some Markov chain Monte Carlo methods needed for learning, and discuss the need to regularize model parameters. Preliminary experimental results are presented.

Keywords: I.2.6 [Artificial Intelligence]:Learning; I.5.1 [Pattern Recognition]:Models–Statistical, I.5.2 [Pattern Recognition]:Design Methodology–Classifier design and evaluation; Additional Key Words: semi-supervised learning, Boltzmann machine

1 Introduction

In many classification applications, labeled training data are scarce but unlabeled data are abundant. It is very useful if we can use unlabeled data to aid labeled data in learning a classifier. Semi-supervised learning deals with exactly this problem [See01]. We assume the dataset in question has the property that nearby (under some local distance metric, e.g. Euclidean) data points tend to have the same labels. Under this assumption the spatial distribution of data, revealed by large amount of unlabeled data, is correlated to classification. We can propagate labels from labeled data to the whole dataset through dense unlabeled data regions.

We formulate the problem as a Markov random field, with nodes being all data points, and edge weights a function of the local distance metric. We propose the use of Boltzmann machine to learn parameters that maximize the likelihood of labeled data, and to compute the labels on unlabeled data. The Boltzmann machine model allows for the learning of individual scaling factors of each dimension, and the learning of label noise. We derive the gradient ascent formula, and suggest several Markov chain Monte Carlo sampling schemes to facilitate learning. We present preliminary experimental results, and discuss the need to regularize the parameters.

2 Boltzmann machine model

2.1 Problem Setup

Let $(x_1, y_1) \dots (x_l, y_l)$ be labeled data, where $Y_L = \{y_1 \dots y_l\} \in \{1 \dots C\}$ are the class labels. Let $(x_{l+1}, y_{l+1}) \dots (x_{l+u}, y_{l+u})$ be unlabeled data where $Y_U = \{y_{l+1} \dots y_{l+u}\}$ are unobserved, usually $l \ll u$. Let $X = \{x_1 \dots x_{l+u}\} \in R^D$. The problem is to estimate Y_U from X and Y_L.

Intuitively, we want data points that are close to have similar labels. So we define our model as

$$P(Y|X) = \frac{1}{Z} \exp[\sum_{i=1}^{l+u} \sum_{j<i} \delta(y_i, y_j) A_{ij}] \tag{1}$$

where Z is the partition function. A_{ij} is a non-negative decreasing function of some distance metric over data points. For any give pair of points x_i, x_j, the label configuration Y in which they have the same label is thus $\exp(A_{ij})$ times more likely than a configuration where they have different labels. A_{ij} in the simplest case decays exponentially with regard to the Euclidean distance d_{ij} between x_i, x_j, with a single hyper-parameter σ. We will discuss a more complex model later. Other choices of distance are possible e.g. when x is discrete. The complete model for now is

$$P(Y|X) = \frac{1}{Z} \exp\left(\sum_{i=1}^{l+u} \sum_{j<i} \delta(y_i, y_j) \exp(-\frac{d_{ij}^2}{2\sigma^2}) \right) \tag{2}$$

Eq.(2) defines a fully connected Boltzmann machine [AHS85] over all data points. A_{ij} are the connection weights, which are tied together by σ. Note when $d_{ij} \approx 0$, $A_{ij} \approx 1$, and when $d_{ij} > 3\sigma$, $A_{ij} \approx 0$, i.e. σ acts like a threshold on interaction range. With the model, we can compute the likelihood of labeled data $P(Y_L|X, \sigma)$. We find the maximum a posteriori (MAP) estimate of σ:

$$\begin{aligned} \sigma_{MAP} &= \arg\max_{\sigma} P(\sigma|X, Y_L) = \arg\max_{\sigma} P(Y_L|X, \sigma)P(\sigma) & (3) \\ &= \arg\max_{\sigma} \sum_{Y_U} P(Y_L, Y_U|X, \sigma)P(\sigma) & (4) \end{aligned}$$

The MAP labeling of the unlabeled data can be inferred from

$$y_i^{MAP} = \arg\max_{y_i} P(y_i|Y_L, X, \sigma_{MAP}), y_i \in Y_U \tag{5}$$

which is in fact a by-product of estimating σ_{MAP}, as shown later.

This model should find the optimal parameter, and allow labels to propagate through high density regions as defined by unlabeled data.

2.2 Learning by Gradient Ascent

We use gradient ascent to find the σ_{MAP}. The gradient consists of two parts:

$$\frac{\partial}{\partial\sigma} \log P(\sigma|X, Y_L) = \frac{\partial}{\partial\sigma} \log P(Y_L|X, \sigma) + \frac{\partial}{\partial\sigma} \log P(\sigma) \tag{6}$$

First we consider the gradient on the log likelihood of labeled data. It can be shown that

$$\begin{aligned} & \frac{\partial}{\partial\sigma} \log P(Y_L|X, \sigma) & (7) \\ = \; & \frac{1}{P(Y_L|X, \sigma)} \frac{\partial}{\partial\sigma} P(Y_L|X, \sigma) & (8) \\ = \; & \frac{1}{P(Y_L|X, \sigma)} \sum_{Y_U} \frac{\partial}{\partial\sigma} P(Y_L, Y_U|X, \sigma) & (9) \\ = \; & \frac{1}{P(Y_L|X, \sigma)} \sum_{Y_U} P(Y_L, Y_U|X, \sigma) \frac{\partial}{\partial\sigma} \log P(Y_L, Y_U|X, \sigma) & (10) \\ = \; & \sum_{Y_U} P(Y_U|Y_L, X, \sigma) \frac{\partial}{\partial\sigma} \log P(Y_L, Y_U|X, \sigma) & (11) \end{aligned}$$

$$= \sum_{Y_U} P(Y_U|Y_L, X, \sigma) \frac{\partial}{\partial \sigma} [\sum_{i=1}^{l+u} \sum_{j<i} \delta(y_i, y_j) \exp(-\frac{d_{ij}^2}{2\sigma^2}) - \log Z] \quad (12)$$

$$= \sum_{Y_U} P(Y_U|Y_L, X, \sigma) \frac{\partial}{\partial \sigma} [\sum_{i=1}^{l+u} \sum_{j<i} \delta(y_i, y_j) \exp(-\frac{d_{ij}^2}{2\sigma^2})] - \frac{\partial}{\partial \sigma} \log Z \quad (13)$$

$$= < \frac{\partial}{\partial \sigma} [\sum_{i=1}^{l+u} \sum_{j<i} \delta(y_i, y_j) \exp(-\frac{d_{ij}^2}{2\sigma^2})] >_c - \frac{\partial}{\partial \sigma} \log Z \quad (14)$$

where $<>_c$ stands for the expectation under the *clamped phase* where Y_L is fixed and only Y_U can change. The last term

$$\frac{\partial}{\partial \sigma} \log Z \quad (15)$$

$$= \frac{1}{Z} \frac{\partial}{\partial \sigma} Z \quad (16)$$

$$= \frac{1}{Z} \sum_Y \frac{\partial}{\partial \sigma} \exp[\sum_{i=1}^{l+u} \sum_{j<i} \delta(y_i, y_j) \exp(-\frac{d_{ij}^2}{2\sigma^2})] \quad (17)$$

$$= \sum_Y \frac{\exp[\sum_{i=1}^{l+u} \sum_{j<i} \delta(y_i, y_j) \exp(-\frac{d_{ij}^2}{2\sigma^2})]}{Z} \frac{\partial}{\partial \sigma} [\sum_{i=1}^{l+u} \sum_{j<i} \delta(y_i, y_j) \exp(-\frac{d_{ij}^2}{2\sigma^2})] \quad (18)$$

$$= \sum_Y P(Y|X, \sigma) \frac{\partial}{\partial \sigma} [\sum_{i=1}^{l+u} \sum_{j<i} \delta(y_i, y_j) \exp(-\frac{d_{ij}^2}{2\sigma^2})] \quad (19)$$

$$= < \frac{\partial}{\partial \sigma} [\sum_{i=1}^{l+u} \sum_{j<i} \delta(y_i, y_j) \exp(-\frac{d_{ij}^2}{2\sigma^2})] >_u \quad (20)$$

$$(21)$$

where $<>_u$ is the expectation under the *unclamped phase* where all labels of Y are allowed to change freely. Put it back and we get

$$\frac{\partial}{\partial \sigma} \log P(Y_L|X, \sigma) \quad (22)$$

$$= < \sum_{i=1}^{l+u} \sum_{j<i} \delta(y_i, y_j) \exp(-\frac{d_{ij}^2}{2\sigma^2}) \frac{d_{ij}^2}{\sigma^3} >_c - < \sum_{i=1}^{l+u} \sum_{j<i} \delta(y_i, y_j) \exp(-\frac{d_{ij}^2}{2\sigma^2}) \frac{d_{ij}^2}{\sigma^3} >_u \quad (23)$$

$$= < \frac{\partial}{\partial \sigma} \log P(Y|X, \sigma) >_c - < \frac{\partial}{\partial \sigma} \log P(Y|X, \sigma) >_u \quad (24)$$

That is, the gradient for the log likelihood is the difference of the expected gradient of negative energy function, between clamped and unclamped phases. This is a standard Boltzmann machine learning problem.

For the time being we assume a flat prior which contributes nothing to the gradient. Later we will discuss the use of different priors.

3 Computing the clamped and unclamped expectations

3.1 Enumeration

Clearly the expectations $<>_c$ and $<>_u$ are impossible to compute exactly for nontrivial datasets. However as a first step, we implemented the Boltzmann machine with an enumeration algorithm. That is, we enumerate all possible Y and compute the probability for each and every one of them. The advantage is that nothing is approximation, and we can have a precise understanding of the proposed framework. Obviously the computation is exponential to the number of data points, so we can only work with very small datasets. In particular, on a 1Ghz Linux machine, it takes 6 seconds to perform one step of hill climbing with a dataset of 16 points. We change the hill climbing step size dynamically: if several previous hill climbing steps were successful, we increase the step size; if overshoot happens, we return to the last good parameters and reduce step size. Hill climbing stops when the step size is reduced below a small threshold.

3.2 Gibbs sampling

Instead of enumerating every possible Y, we can use Markov Chain Monte Carlo (MCMC) methods to sample Y's [Nea93]. MCMC methods allow us to handle larger problems, at the cost of approximation. Note asymptotically we are still guaranteed to get the correct solution, which is different than other approximations like mean field [PA87] [JGJS99].

We start from Gibbs sampler for its simplicity. We need to sample from the clamped distribution $P(Y_U|Y_L, X, \sigma)$ and the unclamped distribution $P(Y|X, \sigma)$ respectively. To sample from $P(Y_U|Y_L, X, \sigma)$, we note

$$P(y_i = c|(Y_U)_{-i}, Y_L, X, \sigma) \sim \exp\left(\sum_{y_j=c}^{j\neq i} A_{ij}\right) \tag{25}$$

for $i = (l+1)\cdots(l+u), c = 1\cdots C$. We need to clamp Y_L. We start from an arbitrary Y_U, hold all but y_{l+1} fixed, sample y_{l+1} from $P(y_{l+1}|Y_{-(l+1)}, X, \sigma)$, and loop through $y_{l+2}, \cdots, y_{l+u}$ and repeat. We need appropriate burn-in time, and set an appropriate interval to take out a sample Y out of the chain. Similarly, to sample from the unclamped distribution $P(Y|X, \sigma)$, we use the same scheme, except that the labeled points are not fixed now and also participate in the sampling.

Given samples $Y_c^1, \cdots, Y_c^{N_c}$ from the clamped distribution and $Y_u^1, \cdots, Y_u^{N_u}$ from the unclamped distribution, the gradients (24) can be approximated by

$$\frac{1}{N_c}\sum_{n=1}^{N_c}\frac{\partial}{\partial\sigma}\log P(Y_c^n|X, \sigma) - \frac{1}{N_u}\sum_{n=1}^{N_u}\frac{\partial}{\partial\sigma}\log P(Y_u^n|X, \sigma) \tag{26}$$

The log likelihood of the observed labels can be estimated from unclamped samples:

$$\mathcal{L} \approx \log \frac{1}{N_u} \sum_{n=1}^{N_u} P(Y_u^n | X, \sigma) \delta((Y_u^n)_L, Y_L) \tag{27}$$

This estimated log likelihood is very crude and should only be trusted on small datasets.

3.3 Gibbs sampling with global Metropolis step

The Gibbs sampler has a hard time producing global changes. One kind of global change that is prominent in our model is the global permutation of labels. For example if the dataset consists of two tight clusters, one with class 1 and the other class 2, then the opposite labeling is usually about equally likely. But Gibbs sampler can take a very long time to swap the two classes. We can perform a global Metropolis step after several Gibbs steps. The proposal distribution is as follows: pick a permutation π of labels uniformly from all permutations, then propose to jump to the new state $\pi(Y) = \pi(y_1) \cdots \pi(y_{l+u})$. This proposal distribution is symmetric. We accept the proposed new state with the Boltzmann acceptance function, i.e. we accept it with probability

$$\frac{P(\pi(Y))}{P(Y) + P(\pi(Y))}$$

otherwise we stay at state Y. We take out a sample from the chains after every big step, which consists of several Gibbs steps and the Metropolis step defined above.

3.4 Swendsen-Wang sampling

The above scheme is not sufficient either: it cannot allow clusters to change labels independent of other clusters. As a result, the statistics we obtain tend to have large variance. The generalized Swendsen-Wang algorithm [SW87] allows large label changes at the level of clusters. We derive the algorithm for our model below.

Assume $A_{ij} \leq 1$ (true in our model). First we look at the unclamped phase. Let us slightly modify the representation of our model so that each factor is within $[0, 1]$. This of course does not change the model:

$$P(Y|X, \sigma) = \frac{1}{Z} \exp \left(\sum_{i=1}^{l+u} \sum_{j<i} (\delta(y_i, y_j) A_{ij} - 1) \right) \tag{28}$$

Following the notation in [Nea93], we write it as

$$P(Y|X,\sigma) = \frac{1}{Z}\prod_{i,j} W_{ij}(Y) \tag{29}$$

where

$$W_{ij}(Y) = \begin{cases} \exp(A_{ij}-1) & \text{if } y_i = y_j \\ \exp(-1) & \text{if } y_i \neq y_j \end{cases} \tag{30}$$

Next we introduce auxiliary variables $z_{ij} \in [0,1]$, with the joint distribution

$$\begin{aligned} P(Y,Z|X,\sigma) &= \frac{1}{Z}\prod_{i,j} \Xi(W_{ij}(Y) - z_{ij}) \qquad (31) \\ &= \frac{1}{Z}\prod_{i,j} \begin{cases} \Xi(\exp(A_{ij}-1) - z_{ij}) & \text{if } y_i = y_j \\ \Xi(\exp(-1) - z_{ij}) & \text{if } y_i \neq y_j \end{cases} \qquad (32) \end{aligned}$$

where $\Xi(w)$ is the 'hard sigmoid' function, i.e. 1 if $w \geq 0$, 0 if $w < 0$. We can now look at yet another binary random variable D_{ij}, which is determined by the value of z_{ij}

$$D_{ij} = \begin{cases} 1 & \text{if } z_{ij} \in (\exp(-1), \exp(A_{ij}-1)) \\ 0 & \text{otherwise} \end{cases} \tag{33}$$

The reason we introduce this extra level of sophistication is because we can do Gibbs sampling iteratively by first fixing Y and sample D_{ij}, and then fixing D_{ij} and sample Y. Both conditional distributions have nice forms, and the Y samples conform to $P(Y|X,\sigma)$. In particular, sampling D_{ij} given Y can be thought of as putting bonds between the nodes with the following probabilities:

$$P(D_{ij}=1|Y) = \begin{cases} 1-\exp(-A_{ij}) & \text{if } y_i = y_j \\ 0 & \text{if } y_i \neq y_j \end{cases} \tag{34}$$

Intuitively, a bond is possible only between nodes with the same labels; and the closer the nodes, the more likely a bond there is. On the other hand, to sample Y given the bonds D_{ij}, the conditional distribution is

$$P(Y|D_{ij}) \propto \prod_{ij} \begin{cases} 1 & \text{if } y_i = y_j \text{ or } D_{ij} = 0 \\ 0 & \text{if } y_i \neq y_j \text{ and } D_{ij} = 1 \end{cases} \tag{35}$$

Intuitively, the bonds divide Y into several connected components. And all we have to do is to uniformly randomly assign a label to each component as a whole.

To run a MCMC chain for the clamped distribution with Swendsen-Wang algorithm, all remain the same except when there are labeled data in a connected component, the component must be assigned the label of the labeled data.

4 Model extension: allowing label noise

4.1 The extended model

In the basic model (section 2.1), labels are assumed to be noiseless. In this section we will extend the model to allow for label noise. Specifically we treat the labels of labeled data as noisy observations of their true (hidden) labels. Let $(x_1, o_1), \ldots, (x_l, o_l)$ be labeled data where $O_L = o_1, \ldots, o_l$ are observed labels. Let $(x_{l+1}, o_{l+1}), \ldots, (x_{l+u}, o_{l+u})$ be unlabeled data where $O_U = o_{l+1}, \ldots, o_{l+u}$ are unobserved. Let $y_1, \ldots, y_{l+u}$ be the true labels.

We construct a Markov random field in which the nodes are $o_1, \ldots, o_{l+u}, y_1, \ldots, y_{l+u}$. There are two sets of edges: The first set fully connects all y nodes and models data point proximity; The second set connects o_i to y_i for $i = 1 \ldots (l+u)$ and models label noise.

We define a new Boltzmann machine $p(O, Y|X, \Theta)$ as follows:

$$\frac{1}{\mathcal{Z}} \exp\left(\sum_{i=1}^{l+u} \sum_{j<i} \delta(y_i, y_j) A_{ij} + \sum_{i=1}^{l+u} \sum_{c=1}^{C} \delta(y_i, c)\beta_c + \sum_{i=1}^{l+u} \sum_{c=1}^{C} \sum_{c'=1}^{C} \delta(o_i, c)\delta(y_i, c') e^{\gamma_{cc'}} \right) \tag{36}$$

A_{ij} is parameterized by some hyper-parameter α. β_c defines class priors. $\gamma_{cc'}$ defines a cost matrix for mislabeling a class c' as c. The parameters of the Boltzmann machine are $\Theta = \{\alpha, \beta, \gamma\}$. As before we learn the MAP parameters

$$\Theta^* = \arg\max_{\Theta} p(\Theta|O_L, X) \tag{37}$$

The gradient of the log likelihood is

$$< \frac{\partial}{\partial \Theta} \log P(O, Y|X, \Theta) >_c - < \frac{\partial}{\partial \Theta} \log P(O, Y|X, \Theta) >_u \tag{38}$$

As an example, define A_{ij} so it decreases exponentially with the distance in each dimension, but at different speed:

$$A_{ij} = \exp\{\alpha_0 - \sum_{d=1}^{D} \frac{(x_i^d - x_j^d)^2}{e^{\alpha_d}}\} \tag{39}$$

$\alpha = \{\alpha_0, \ldots, \alpha_D\}$ are similar to the length scales in Gaussian process. We also assume all classes have the same confusion probability of being mislabeled as a different class. Under these assumptions, $p(O, Y|X, \Theta)$ can be written as

$$\frac{1}{\mathcal{Z}} \exp\left(\sum_{i=1}^{l+u} \sum_{j<i} \delta(y_i, y_j) \exp\{\alpha_0 - \sum_{d=1}^{D} \frac{(x_i^d - x_j^d)^2}{\exp(\alpha_d)}\} + \sum_{i=1}^{l+u} \sum_{c=1}^{C} \delta(y_i, c)\beta_c + \sum_{i=1}^{l+u} \delta(o_i, y_i) e^{\gamma} \right) \tag{40}$$

Note we use the parameterization e^{α_d} and e^{γ} instead of α_d and γ to implicitly force them to be positive, reflecting our belief that (a) closer distance means stronger interaction; (b) label noise are not so strong as to invert majority of the true labels. Simple calculation shows

$$\frac{\partial}{\partial \alpha_0} \log P(O, Y|X, \Theta) = \sum_{i=1}^{l+u} \sum_{j<i} \delta(y_i, y_j) A_{ij} \tag{41}$$

$$\frac{\partial}{\partial \alpha_d} \log P(O, Y|X, \Theta) = \sum_{i=1}^{l+u} \sum_{j<i} \delta(y_i, y_j) A_{ij} \frac{(x_i^d - x_j^d)^2}{\exp(\alpha_d)} \tag{42}$$

$$\frac{\partial}{\partial \beta_c} \log P(O, Y|X, \Theta) = \sum_{i=1}^{l+u} \delta(y_i, c) \;, c = 1, \ldots, C \tag{43}$$

$$\frac{\partial}{\partial \gamma} \log P(O, Y|X, \Theta) = \sum_{i=1}^{l+u} \delta(o_i, y_i) e^{\gamma} \tag{44}$$

4.2 Speed Up

We doubled the size of the Markov random field by introducing O. It seems computation will be much slower. But we show that in fact we only need to work with Y during computation, so the speed is close to the basic model. First let us define some shorthand notation:

$$F(Y|X, \Theta) = \sum_{i=1}^{l+u} \sum_{j<i} \delta(y_i, y_j) A_{ij} + \sum_{i=1}^{l+u} \sum_{c=1}^{C} \delta(y_i, c) \beta_c \tag{45}$$

$$G(O_L, Y|X, \Theta) = F(Y|X, \Theta) + \sum_{i=1}^{l} \sum_{c=1}^{C} \sum_{c'=1}^{C} \delta(o_i, c) \delta(y_i, c') e^{\gamma_{cc'}} \tag{46}$$

$$H(O, Y|X, \Theta) = G(O_L, Y|X, \Theta) + \sum_{i=l+1}^{l+u} \sum_{c=1}^{C} \sum_{c'=1}^{C} \delta(o_i, c) \delta(y_i, c') e^{\gamma_{cc'}} \tag{47}$$

In the unclamped phase, note

$$\begin{aligned} p(Y|X, \Theta) &= \sum_{O} P(O, Y|X, \Theta) && (48) \\ &= \sum_{O} \frac{1}{\mathcal{Z}} \exp[F(Y|X, \Theta)] \prod_{i=1}^{l+u} \exp[\delta(o_i, y_i) e^{\gamma}] && (49) \\ &= \frac{1}{\mathcal{Z}} \exp[F(Y|X, \Theta)] \prod_{i=1}^{l+u} \sum_{o_i=1}^{C} \exp[\delta(o_i, y_i) e^{\gamma}] && (50) \end{aligned}$$

$$= \frac{1}{\mathcal{Z}} \exp[F(Y|X,\Theta)] \prod_{i=1}^{l+u} (C - 1 + e^{e^\gamma}) \quad (51)$$

$$= \frac{1}{\mathcal{Z}_F} \exp[F(Y|X,\Theta)] \quad (52)$$

$$(53)$$

where

$$\mathcal{Z}_F = \sum_Y \exp[F(Y|X,\Theta)] \quad (54)$$

$P(Y|X,\Theta)$ has the same function form of $p(O,Y|X,\Theta)$, except without the last term in the exponent. This is not surprising given that O and Y only interact through the last term. $P(Y|X,\Theta)$ is sufficient to compute the expectations of gradient of α and β under the unclamped distribution. If we can do the same for γ, we would only need to sample from $P(Y|X,\Theta)$ and not the joint probability $P(O,Y|X,\Theta)$. Let's look at the expectation of gradient of γ under the unclamped distribution:

$$< \frac{\partial}{\partial\gamma} H(O,Y|X,\Theta) >_u \quad (55)$$

$$= \sum_O \sum_Y p(O,Y|X,\Theta) \frac{\partial}{\partial\gamma} H(O,Y|X,\Theta) \quad (56)$$

$$= \sum_Y p(Y|X,\Theta) \sum_O p(O|Y) \sum_{i=1}^{l+u} \delta(o_i,y_i) e^\gamma \quad (57)$$

$$= \sum_Y p(Y|X,\Theta) \sum_O \frac{\exp \sum_i \delta(o_i,y_i)e^\gamma}{\sum_{O'} \exp \sum_i \delta(o'_i,y_i)e^\gamma} \sum_{i=1}^{l+u} \delta(o_i,y_i) e^\gamma \quad (58)$$

There are

$$\mathcal{C}_{l+u}^k (C-1)^{(l+u-k)} \quad (59)$$

different O's that have exactly k states the same as any given Y, for $k = 0, \cdots, l+u$ and $\mathcal{C}_{l+u}^k = \frac{(l+u)!}{k!(l+u-k)!}$. Thus

$$(58) \quad (60)$$

$$= \sum_Y p(Y|X,\Theta) \sum_{k=0}^{l+u} \mathcal{C}_{l+u}^k (C-1)^{(l+u-k)} \frac{\exp(ke^\gamma)ke^\gamma}{\sum_{k'=0}^{l+u} \mathcal{C}_{l+u}^{k'} (C-1)^{(l+u-k')} \exp(k'e^\gamma)} \quad (61)$$

$$= \frac{\sum_{k=0}^{l+u} \mathcal{C}_{l+u}^k (C-1)^{(l+u-k)} (e^{e^\gamma})^k ke^\gamma}{\sum_{k'=0}^{l+u} \mathcal{C}_{l+u}^{k'} (C-1)^{(l+u-k')} (e^{e^\gamma})^{k'}} \quad (62)$$

$$= \frac{(l+u)e^{(\gamma+e^\gamma)}(C-1+e^{e^\gamma})^{(l+u-1)}}{(C-1+e^{e^\gamma})^{(l+u)}} \quad (63)$$

$$= \frac{(l+u)e^{(\gamma+e^\gamma)}}{(C-1+e^{e^\gamma})} \quad (64)$$

which is a function of γ only. Therefore everything we need during the unclamped phase can be obtained from $P(Y|X,\Theta)$.

In the clamped phase, the value of $O_L = o_1, \ldots, o_l$ is fixed. It is easy to verify that

$$P(Y, O_U|O_L, X, \Theta) = \frac{1}{\mathcal{Z}'} \exp\{G(O_L, Y|X, \Theta) + \sum_{i=l+1}^{l+u} \delta(o_i, y_i) e^{\gamma}\} \tag{65}$$

By similar argument, we have the marginal distribution

$$P(Y|O_L, X, \Theta) = \frac{1}{\mathcal{Z}_G} \exp[G(O_L, Y|X, \Theta)] \tag{66}$$

from which we can compute the expectations of gradient of α and β under the clamped distribution. As for γ:

$$< \frac{\partial}{\partial\gamma} H(O, Y|X, \Theta) >_c \tag{67}$$

$$= \sum_Y \sum_{O_U} p(Y, O_U|O_L, X, \Theta) \frac{\partial}{\partial\gamma} H(O, Y|X, \Theta) \tag{68}$$

$$= \sum_Y p(Y|O_L, X, \Theta) \sum_{O_U} p(O_U|Y) \sum_{i=1}^{l+u} \delta(o_i, y_i) e^{\gamma} \tag{69}$$

$$= \sum_Y p(Y|O_L, X, \Theta) \sum_{O_U} \frac{\exp \sum_{i=l+1}^{l+u} \delta(o_i, y_i) e^{\gamma}}{\sum_{O'_U} \exp \sum_{i=l+1}^{l+u} \delta(o'_i, y_i) e^{\gamma}} [\sum_{i=1}^{l} \delta(o_i, y_i) + \sum_{i=l+1}^{l+u} \delta(o_i, y_i)] e^{\gamma} \tag{70}$$

Similarly, there are $\mathcal{C}_u^k (C-1)^{(u-k)}$ different O_U's that have exactly k states the same as any given Y_U, for $k = 0, \cdots, u$. Continue, we get

$$\sum_Y p(Y|O_L, X, \Theta) \sum_{k=0}^{u} \mathcal{C}_u^k (C-1)^{(u-k)} \frac{\exp(k e^{\gamma})}{\sum_{k'=0}^{u} \mathcal{C}_u^{k'} (C-1)^{(u-k')} \exp(k' e^{\gamma})}$$

$$\cdot [\sum_{i=1}^{l} \delta(o_i, y_i) + k] e^{\gamma} \tag{71}$$

$$= \sum_Y p(Y|O_L, X, \Theta) \{[\sum_{i=1}^{l} \delta(o_i, y_i)] e^{\gamma} \frac{\sum_{k=0}^{u} \mathcal{C}_u^k (C-1)^{(u-k)} \exp(k e^{\gamma})}{\sum_{k'=0}^{u} \mathcal{C}_u^{k'} (C-1)^{(u-k')} \exp(k' e^{\gamma})}$$

$$+ \frac{\sum_{k=0}^{u} \mathcal{C}_u^k (C-1)^{(u-k)} \exp(k e^{\gamma}) k e^{\gamma}}{\sum_{k'=0}^{u} \mathcal{C}_u^{k'} (C-1)^{(u-k')} \exp(k' e^{\gamma})}\} \tag{72}$$

$$= \sum_Y p(Y|O_L, X, \Theta) [\sum_{i=1}^{l} \delta(o_i, y_i)] e^{\gamma} + \frac{u e^{(\gamma + e^{\gamma})}}{C - 1 + e^{e^{\gamma}}} \tag{73}$$

which can be obtained from $P(Y|O_L, X, \Theta)$ too.

The log-likelihood $\mathcal{L}$ of the observed labels can be monitored, if desirable, by

$$\begin{aligned}
\mathcal{L} &= \log \sum_Y \sum_{O_U} P(O_L, O_U, Y|X, \Theta) && (74)\\
&= \log \sum_Y \sum_{O_U} \frac{1}{\mathcal{Z}} \exp(G(Y, O_L)) \exp(\sum_{i=l+1}^{l+u} \delta(o_i, y_i) e^{\gamma}) && (75)\\
&= \log \frac{1}{\mathcal{Z}} \sum_Y \exp(G(Y, O_L))(C - 1 + e^{e^{\gamma}})^u && (76)\\
&= \log \frac{\mathcal{Z}_G}{\mathcal{Z}} (C - 1 + e^{e^{\gamma}})^u && (77)
\end{aligned}$$

where the partition function

$$\begin{aligned}
\mathcal{Z} &= \sum_Y \sum_O \exp H(O, Y|X, \Theta) && (78)\\
&= \sum_Y \sum_O \exp(F(Y) + \sum_{i=1}^{l+u} \delta(o_i, y_i) e^{\gamma}) && (79)\\
&= \sum_Y \exp(F(Y))(C - 1 + e^{e^{\gamma}})^{(l+u)} && (80)\\
&= \mathcal{Z}_F (C - 1 + e^{e^{\gamma}})^{(l+u)} && (81)
\end{aligned}$$

therefore

$$\mathcal{L} = \log \frac{\mathcal{Z}_G}{\mathcal{Z}_F (C - 1 + e^{e^{\gamma}})^l} \tag{82}$$

5 Experiments with a flat prior over Θ

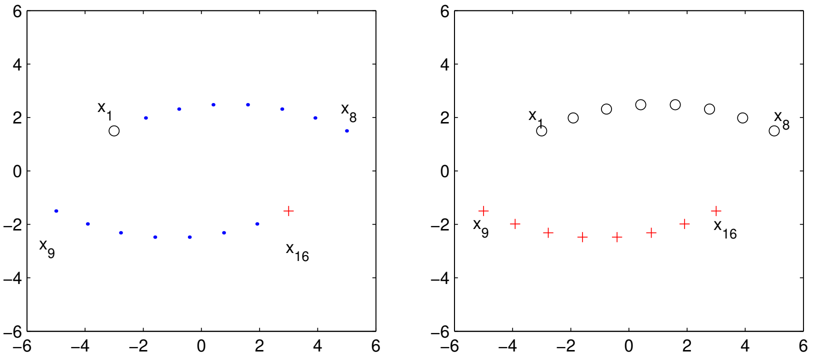
5.1 Boltzmann machines with fixed parameters

Before we present parameter learning, it is useful to show the behavior of Boltzmann machines with fixed parameters. The parameters used in this section are not arbitrary: they are selected from results of the best parameter learning trials. All computation in this section is done with enumeration, so they are exact. We present 4 synthetic datasets.

Synthetic Dataset 1: Double Arcs

This dataset has 16 points which form two arcs, as in Figure 1(a). On different ends of the arcs, x_1 and x_{16} are labeled as class 'o' and '+' respectively. We hope the class of x_1 will propagate all the way to x_8 through the upper arc, and similarly for x_{16} in the lower arc. The parameters are $\alpha_0 = 1.1$, $\alpha_1 = 0.8$, $\alpha_2 = -1.9$,

$\beta_1 = \beta_2 = 1$, $\gamma = 1.6$. The labels and posterior probabilities $P(y_i|O_L, X, \Theta^*)$ are shown in Figure 1(b) and Table 1. As expected, the Boltzmann machine thinks x_8 is more likely to be in class 'o' even though it is closer to x_{16} '+' by the original Euclidean distance. This result shows the ability of the Boltzmann machine to exploit high density regions defined by unlabeled data.



(a) Observed Labels o_i (b) Computed Labels y_i

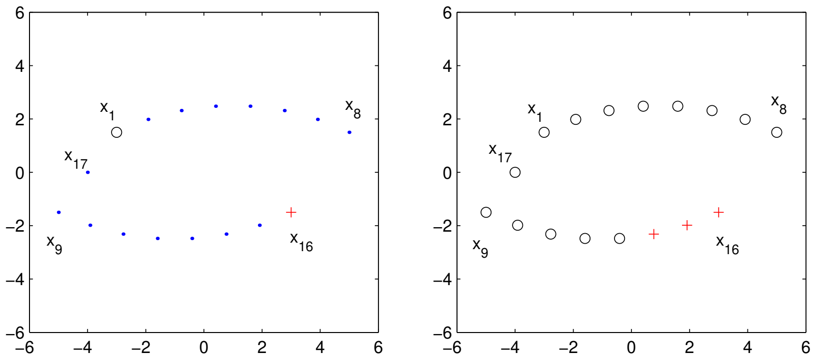
Figure 1: Double Arcs Dataset. Labels travel through high density regions. In the input data, x_1 and x_{16} are labeled with the 2 classes 'o'/'+' respectively, and 14 points are unlabeled. Under the parameters, the Boltzmann machine assigns the labels in (b). The probabilities of these labels are listed in Table 1.

Table 1: $P(y_i = \text{'o'}|O_L, X, \Theta^*)$ for the Double Arcs dataset.

y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8
0.995	0.591	0.537	0.525	0.519	0.513	0.505	0.501
y_9	y_{10}	y_{11}	y_{12}	y_{13}	y_{14}	y_{15}	y_{16}
0.499	0.495	0.487	0.481	0.475	0.463	0.409	0.005

Synthetic Dataset 2: Connected Arcs

We add another unlabeled data point x_{17} in the middle of x_1 and x_9, as in Figure 2(a). x_{17} serves as a bridge that connects the two arcs. We predict that o_1's influence will then reach the lower arc through x_{17}. We use parameters $\alpha_0 = 9.5$, $\alpha_1 = -2.0$, $\alpha_2 = 0.3$, $\beta_1 = 0.1$, $\beta_2 = -0.03$, $\gamma = 1.9$, and the results are shown in Figure 2(b) and Table 2.



(a) Observed Labels o_i (b) Computed Labels y_i

Figure 2: Connected Arcs Dataset. x_{17} connects the two arcs so the influence of o_1 is propagated to the lower arc. The probabilities of these labels are listed in Table 2.

Table 2: $P(y_i = \text{'o'}|O_L, X, \Theta^*)$ for the Connected Arcs dataset.

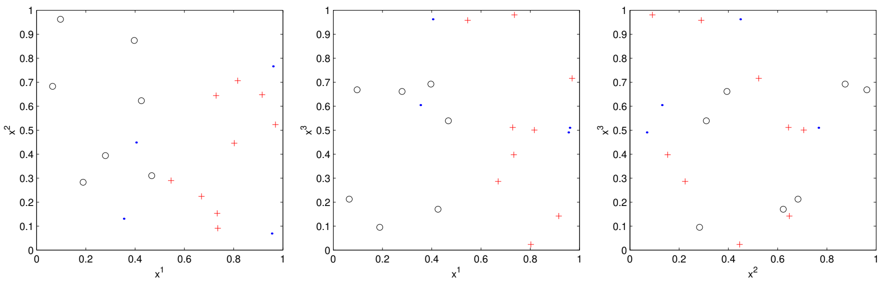
y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8
0.999	0.930	0.745	0.621	0.578	0.558	0.584	0.582
y_{17}							
0.923							
y_9	y_{10}	y_{11}	y_{12}	y_{13}	y_{14}	y_{15}	y_{16}
0.913	0.923	0.765	0.615	0.525	0.369	0.116	0.002

Synthetic Dataset 3: Cube with Irrelevant Dimensions

To demonstrate a Boltzmann machine's ability to ignore irrelevant dimensions, we create a synthetic dataset by uniformly sampling 20 points from the unit cube $[0, 1]^3$. We label 16 of them with class 'o' if the first dimension $x^1 < 0.5$, and class '+' otherwise. So by design the x^2 and x^3 dimensions are irrelevant. Figure 3 shows the projected data. We fix parameters at $\alpha_0 = 0.9$, $\alpha_1 = -4.8$, $\alpha_2 = 1.3$, $\alpha_3 = 1.0$, $\beta_1 = 1.0$, $\beta_2 = 1.0$, $\gamma = 2.2$. (the parameters are automatically learned. Note α_1 is much smaller than α_2 and α_3, which means that the machine pays much more attention to distinctions along the x^1 dimension). Table 3 gives the results under these parameters.

Synthetic Dataset 4: Noisy Labels

Yet another property of the extended Boltzmann machine is its ability to handle label noise. Figure 4(a) shows such a dataset. There are two clusters of data points, with x_1 and x_2 intentionally mislabeled. There is no unlabeled data.



x^1 vs. x^2 x^1 vs. x^3 x^2 vs. x^3

Figure 3: Cube with Irrelevant Dimensions. Only the first dimension x^1 is relevant to classification.

Table 3: Results of the Cube dataset. $P(y_i)$ stands for $P(y_i = \text{‘}o\text{'} | O_L, X, \Theta)$.

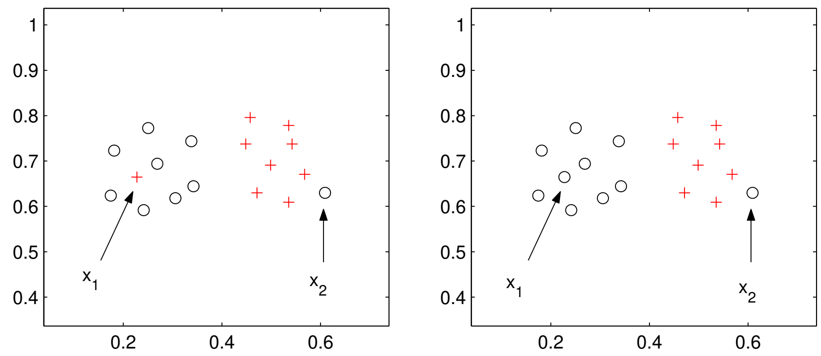
x_i^1	0.065	0.097	0.189	0.280	0.355	0.397	0.405	0.425	0.468	0.546
o_i	o	o	o	o	.	o	.	o	o	+
y_i	o	o	o	o	o	o	o	o	o	+
$P(y_i)$	1.000	1.000	1.000	1.000	0.998	1.000	0.999	1.000	1.000	0.001

x_i^1	0.670	0.729	0.733	0.736	0.802	0.817	0.916	0.956	0.961	0.970
o_i	+	+	+	+	+	+	+	.	.	+
y_i	+	+	+	+	+	+	+	+	+	+
$P(y_i)$	0	0	0	0	0	0	0	0.002	0.002	0

We show the result with parameters $\alpha_0 = 1.8$, $\alpha_1 = -6.9$, $\alpha_2 = 1.1$, $\beta_1 = 1.0$, $\beta_2 = 1.0$, and $\gamma = 1.0$. The Boltzmann machine believes that x_1 is mislabeled, and corrects it with high confidence as expected (see Figure 4(b) and Table 4). However it thinks x_2's observed label $o_2 =$ ‘o’ is probably correct. This behavior is somewhat counter-intuitive, but can be explained by the fact that there is no *a priori* notion of decision boundary or margin encoded in the Boltzmann machine. That is, although x_2 ‘looks’ like having a large margin to the imaginary vertical decision boundary between the two clusters, the machine has no such knowledge. What is important here is that the density of x_2's neighbors is lower than that of x_1's, and this allows x_2 to keep its label.

Table 4: The Noisy Labels dataset. $P(y_i)$ stands for $P(y_i = \text{‘}o\text{'} | O_L, X, \Theta)$.

	left cluster		right cluster	
	x_1	other	x_2	other
o_i	+	o	o	+
y_i	o	o	o	+
$P(y_i)$	0.999	> 0.998	0.808	< 0.001



(a) Observed Labels o_i (b) Computed Labels y_i

Figure 4: The Noisy Labels Dataset. x_1 and x_2 are intentionally mislabeled in (a). The trained Boltzmann machine (b) corrects x_1's, but not x_2's, label. The label probabilities are given in Table 4.

5.2 Sensitivity to initial parameters during learning

In this section we show that our parameter learning is sensitive to initial parameters. The Boltzmann machine training algorithm is a hill climbing procedure, and the likelihood function has multiple local maxima in general. To see how sensitive the training is to initial parameters, we randomly generate random initial parameters as follows. Consider the true labels of two overlapping points y_i, y_j. Very roughly, the likelihood ratio of such overlapping points having the same label versus having different labels is $P(y_i = y_j)/P(y_i \neq y_j) = e^{e^{\alpha_0}}$. We arbitrarily specify the range of the likelihood ratio to be $[1.01, 1000]$, which translates to uniformly sample from

$$\alpha_0 \in [-4.61, 1.93]$$

For the scaling factors, we specify the range $e^{\alpha_d} \in [\min_{i,j}(x_i^d - x_j^d)^2/5, \max_{i,j}(x_i^d - x_j^d)^2 * 5]$, for $d = 1 \cdots D$, ranging from very sensitive to very flat in that dimension. That is, we uniformly sample from

$$\alpha_d \in [\log(\min_{i,j}(x_i^d - x_j^d)^2/5), \log(\max_{i,j}(x_i^d - x_j^d)^2 * 5)]$$

For class priors, we let them be between 0.1 and 0.9. In binary classification, we have roughly $P(y = c_1)/P(y = c_2) = e^{(\beta_1 - \beta_2)}$. We can fix β_1 and sample β_2:

$$\beta_1 = 0, \beta_2 \in [-2.20, 2.20]$$

For noisy labeling, roughly $P(y_i = o_i)/P(y_i \neq o_i) = e^{e^{\gamma}}$. Let $P(y_i = o_i)$ range from $[0.6, 0.999]$ so we sample from

$$\gamma \in [-0.90, 1.93]$$

We randomly generate 10 sets of initial parameters for each synthetic dataset above, and perform parameter learning starting from each set. Hill climbing stops when the log likelihood converges (or increases very slowly). The results are listed in Table 5 – 8. Each table lists the random initial parameters, the learned parameters, the log likelihood of O_L, and the 'y' column indicates whether the resulting labels Y are the same as the examples in the previous section.

Table 5: Hill climbing starting from 10 random initial parameter sets for the Double Arcs dataset. $\alpha_1 \in [-4.56, 6.21]$ and $\alpha_2 \in [-5.20, 4.81]$.

Trial	α_0	α_1	α_2	β_1	β_2	γ	$\mathcal{L}$	y
#1 starts	-4.008	-3.401	-3.105	0	-0.096	0.692		
ends	-4.008	-3.401	-3.105	-0.048	-0.048	0.691	-1.386	
#2 starts	0.876	-4.083	-1.549	0	-1.338	-0.285		
ends	0.876	-4.083	-1.549	-0.669	-0.669	-0.796	-1.386	
#3 starts	-1.986	4.661	-0.934	0	-1.202	0.774		
ends	-2.043	4.646	-0.975	-0.601	-0.601	0.514	-1.386	yes
#4 starts	-3.417	2.636	-2.509	0	-1.067	1.214		
ends	-3.418	2.636	-2.510	-0.534	-0.534	1.130	-1.386	yes
#5 starts	1.692	5.323	-2.202	0	-0.932	1.035		
ends	1.468	5.289	-2.668	-0.466	-0.466	0.487	-1.386	yes
#6 starts	-0.982	-1.416	-5.026	0	1.128	1.209		
ends	-0.982	-1.416	-5.026	0.564	0.564	1.112	-1.386	
#7 starts	-3.474	2.334	-4.476	0	-0.072	0.563		
ends	-3.474	2.334	-4.476	-0.036	-0.036	0.562	-1.386	
#8 starts	0.308	4.575	-3.976	0	-1.172	-0.235		
ends	0.227	4.520	-3.976	-0.586	-0.586	-0.653	-1.386	
#9 starts	0.057	-0.663	-2.147	0	0.845	-0.604		
ends	0.056	-0.666	-2.150	0.422	0.422	-0.794	-1.386	
#10 starts	-4.248	-2.203	0.746	0	1.438	1.785		
ends	-4.248	-2.203	0.746	0.719	0.719	1.764	-1.386	

6 The need for a non-flat prior

The experiments in the previous section not only shows that learning is sensitive to initial parameters, but also reveals problems with the optimization criterion. We learn Θ to maximize the log likelihood $\log P(O_L|X, \Theta)$ (under a flat prior over Θ). However log likelihood sometimes is not powerful enough to distinguish sensible vs. non-sensible parameters. Consider the Double Arcs dataset 1. A good Θ should propagate labels along the arcs. However, the Θ that maximizes the log likelihood of the two observed labels is the one that drives $A_{ij} \to 0$,

Table 6: Hill climbing starting from 10 random initial parameter sets for the Connected Arcs dataset. $\alpha_1 \in [-6.52, 6.21]$ and $\alpha_2 \in [-5.20, 4.81]$.

Trial	α_0	α_1	α_2	β_1	β_2	γ	$\mathcal{L}$	y
#1 starts	1.089	-4.198	0.699	0	-0.713	1.115		
ends	1.089	-4.198	0.699	-0.357	-0.357	1.073	-1.386	
#2 starts	-0.617	2.034	1.219	0	-0.071	-0.682		
ends	9.549	-1.978	0.273	0.046	-0.117	1.897	-1.287	yes
#3 starts	0.201	-4.729	3.720	0	0.066	0.025		
ends	0.201	-4.729	3.720	0.033	0.033	0.024	-1.386	
#4 starts	-2.371	-4.165	4.164	0	2.186	-0.469		
ends	-2.371	-4.170	4.164	1.093	1.093	-2.254	-1.386	
#5 starts	1.898	-0.646	-0.562	0	0.101	0.048		
ends	9.546	-1.978	0.274	0.132	-0.032	1.895	-1.287	yes
#6 starts	0.831	-6.289	2.494	0	1.587	1.623		
ends	0.831	-6.289	2.494	0.794	0.794	1.565	-1.386	
#7 starts	-1.222	-4.000	1.570	0	1.454	-0.338		
ends	-1.222	-4.001	1.569	0.727	0.727	-0.963	-1.386	
#8 starts	-4.012	-1.445	-2.363	0	-1.254	1.358		
ends	-4.012	-1.445	-2.363	-0.627	-0.627	1.274	-1.386	
#9 starts	-0.678	-6.137	-3.228	0	1.846	0.808		
ends	-0.678	-6.137	-3.228	0.923	0.923	0.031	-1.386	
#10 starts	0.654	0.601	-3.600	0	-1.375	-0.790		
ends	0.654	0.601	-3.601	-0.687	-0.687	-1.354	-1.386	yes

because this would 'disconnect' any connection between points. By doing so each point is independent of all other points, and somewhat counter-intuitively the log likelihood is maximized, since the two labeled points are from different classes. In general, imagine a dataset consists of C well separated classes, each being a tight cluster. We observe only one labeled point (and many unlabeled points) in each cluster. It is reasonable to expect the best Θ to be large enough to connect points within clusters, but small enough to not do so between clusters, so that labels can spread within clusters. But it is easy to prove that the Θ which maximizes log likelihood of labeled data is the one that disconnects all points, and results in no propagation.

The problem is more clearly demonstrated with the basic model Eq.(2). The only parameter is σ which controls propagation in all dimensions. When $\sigma \to 0$ all points are disconnected and independent of each other. We create several synthetic datasets with different structures, as shown in Figure 5. For each dataset, we plot its log likelihood vs. σ curve in Figure 6. These are computed with Swendsen-Wang sampling, with burn-in period of 100 samples, and 10000 clamped / 10000 unclamped samples.

Several datasets have a log likelihood plateau when σ is small. This is

Table 7: Hill climbing starting from 10 random initial parameter sets for the Cube dataset. $\alpha_1 \in [-13.70, 1.41]$, $\alpha_2 \in [-13.16, 1.38]$, and $\alpha_3 \in [-14.38, 1.52]$.

Trial	α_0	α_1	α_2	α_3	β_1	β_2	γ	$\mathcal{L}$	y
#1 starts	-3.904	-11.605	-7.516	-4.240	0	-0.222	-0.779		
ends	-3.904	-11.605	-7.516	-4.240	-0.598	0.376	-0.565	-10.965	
#2 starts	-1.945	-3.772	-9.076	-7.442	0	-1.522	-0.377		
ends	-1.945	-3.772	-9.076	-7.442	-1.364	-0.158	-0.751	-10.965	
#3 starts	-0.261	-7.915	-0.784	-0.358	0	-0.110	-0.707		
ends	1.267	-4.960	1.765	-0.434	-0.068	-0.042	2.381	-4.089	yes
#4 starts	1.071	-3.417	0.363	-6.062	0	-1.689	-0.397		
ends	5.041	-4.560	0.605	-4.572	-0.883	-0.806	2.274	-4.156	
#5 starts	-4.606	-9.825	-3.696	-9.302	0	2.012	-0.792		
ends	-4.606	-9.825	-3.696	-9.302	0.081	1.931	-1.059	-10.965	
#6 starts	-0.217	-8.207	-2.767	-5.150	0	2.005	0.992		
ends	4.083	-5.665	1.461	-2.654	0.990	1.015	2.378	-4.074	yes
#7 starts	-0.954	-5.746	-10.055	-0.059	0	1.071	1.644		
ends	-0.953	-5.735	-10.061	-0.059	0.408	0.663	1.625	-10.965	
#8 starts	1.248	-8.199	-12.415	-9.598	0	-1.704	1.097		
ends	1.248	-8.199	-12.415	-9.598	-1.016	-0.688	0.711	-10.965	
#9 starts	0.300	-7.179	-1.424	-0.591	0	-1.062	0.539		
ends	1.294	-4.969	1.781	-0.503	-0.544	-0.518	2.391	-4.089	yes
#10 starts	-0.903	-12.067	-5.349	-0.430	0	-2.006	0.741		
ends	1.248	-4.952	1.315	-0.362	-1.016	-0.990	2.417	-4.100	yes

precisely the problem: for these datasets, as long as label points of different classes are not connected, the log likelihood is roughly flat (actually the left end is slightly higher). Gradient ascent will not work. We solve it by applying a prior on σ to encourage large σ values. We first limit the range of σ in $[0, \max_{ij}(d_{ij})]$, since a larger σ (larger than the diameter of the dataset) is not useful. Within this interval we define a prior as

$$p(\sigma) \propto e^{\lambda\sigma} \ , \sigma \in [0, \max_{ij}(d_{ij})] \tag{83}$$

where λ is a small positive number. The change to gradient Eq. (6) is straight forward:

$$\frac{\partial}{\partial\sigma} \log p(\sigma) = \lambda \tag{84}$$

This prior effective tilts the log likelihood curve counter-clockwise a little, so there will be a maximum and the right end of the original plateau. λ is empirically set for each dataset.

The vertical lines in Figure 6 shows the σ learned with gradient ascent after adding the above prior. The prior helps σ to settle in the large end of the

Table 8: Hill climbing starting from 10 random initial parameter sets for the Noisy Labels dataset. $\alpha_1 \in [-12.37, -0.06]$ and $\alpha_2 \in [-13.28, -1.57]$.

Trial	α_0	α_1	α_2	β_1	β_2	γ	$\mathcal{L}$	y
#1 starts	-2.295	-11.775	-13.150	0	1.746	1.357		
ends	-2.295	-11.775	-13.150	0.873	0.873	1.144	-12.477	
#2 starts	-3.220	-3.803	-4.835	0	2.191	-0.302		
ends	0.109	-6.087	1.231	1.095	1.095	2.665	-7.857	
#3 starts	-1.883	-11.808	-12.846	0	0.895	-0.647		
ends	-1.883	-11.808	-12.846	0.448	0.448	-0.863	-12.477	
#4 starts	-4.297	-0.709	-7.336	0	-1.243	0.801		
ends	0.227	0.945	-8.139	-0.448	-0.794	-1.948	-12.455	
#5 starts	-1.701	-9.829	-11.242	0	1.576	0.674		
ends	-1.701	-9.829	-11.242	0.788	0.788	0.124	-12.477	
#6 starts	1.428	-9.611	-4.180	0	0.905	1.676		
ends	5.608	-10.658	-1.016	0.452	0.452	2.484	-8.323	
#7 starts	1.438	-5.166	-9.656	0	-0.202	0.222		
ends	9.213	-6.687	-12.601	-0.101	-0.101	2.510	-11.785	
#8 starts	1.147	-11.931	-3.355	0	-1.224	0.168		
ends	5.653	-10.668	-1.001	-0.612	-0.612	2.487	-8.323	
#9 starts	-2.826	-12.141	-10.182	0	-1.612	0.537		
ends	-2.826	-12.141	-10.182	-0.806	-0.806	-0.105	-12.477	
#10 starts	-1.300	-2.661	-1.934	0	-1.306	-0.005		
ends	1.770	-6.885	0.205	-0.641	-0.665	0.972	-6.906	yes

(original) plateau. Figure 7 shows the label posterior under the learned σ for each dataset. In the figure, different symbols represent different classes. Each symbol is color coded to represent its class probabilities (confidence): vivid color means high confidence, and gray means low confidence (close to uniform class probabilities).

For the extended model, we will need more sophisticated prior over parameters. For instance, the parameters might 'conspire' by putting no penalty on noisy labels ($\gamma \to -\infty$), so that the model can explain any observed O_L with a Y in which all points have the same label. The model can then use large α's to maximize the likelihood of seeing the same labels. We will investigate this problem in the future.

7 Related work

The graph mincut algorithm [BC01] works on binary classification problems. It finds a minimum cut through the labeled/unlabeled data graph that separates the two classes. Since a cut removes an edge connecting two nodes y_i and y_j with different labels, a mincut is equivalent to the lowest energy state configuration, which is in turn the most likely state configuration of the Boltzmann machine $\arg\max_Y P(Y|X, w_{ij})$. Our algorithm finds the MAP state configuration (Eq. 5) instead, which is a weighted average of all configurations. Of course we can also find the approximate most likely configuration during sampling.

8 Summary

We formulated the problem of semi-supervised learning as Boltzmann machine learning. We parameterized feature weighting and label noise in the Boltzmann machines, learned parameters by gradient ascent on the likelihood of observed labels. We also presented several MCMC sampling schemes to facilitate learning.

The results on some small synthetic datasets are promising. However, two problems must be solved for this method to be useful. Firstly we need a better way to regularize parameters (e.g. priors) to avoid undesirable maxima in data likelihood. Secondly the computation is very heavy. The enumeration method is exponential in the number of data points. Even Swendsen-Wang and other MCMC methods are not practical on very large datasets. Besides, we might need techniques like coupling from the past [HN00] to ensure the convergence of the Markov chain.

We are investigating a related but different way to learn from both labeled and unlabeled data, by means of label propagation [ZG02]. We think it is a more promising direction.

Acknowledgement

We thank Ke Yang for helpful discussions. The first author is supported by a Microsoft Graduate Research fellowship.

References

[AHS85] David H. Ackley, Geoffrey E. Hinton, and Terrence J. Sejnowski. A learning algorithm for Boltzmann machines. *Cognitive Science*, 9:147–169, 1985.

[BC01] A. Blum and S. Chawla. Learning from labeled and unlabeled data using graph mincut. In *Proc. 18th International Conf. on Machine Learning*, 2001.

[HN00] Michael Harvey and Radford M. Neal. Inference for belief networks using coupling from the past. In *Uncertainty in Artificial Intelligence: Proceedings of the sixteenth conference*, pages 256–263, 2000.

[JGJS99] Michael I. Jordan, Zoubin Ghahramani, Tommi Jaakkola, and Lawrence K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999.

[Nea93] R. M. Neal. Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG-TR-93-1, University of Toronto, 1993.

[PA87] Carsten Peterson and James R. Anderson. A mean field theory learning algorithm for neural networks. *Complex Systems*, 1:995–1019, 1987.

[See01] Matthias Seeger. Learning with labeled and unlabeled data. Technical report, University of Edinburgh, 2001.

[SW87] R.H. Swendsen and J.S. Wang. Nonuniversal critical dynamics in monte carlo simulations. *Physical Review Letters*, 58(2):86–88, January 1987.

[ZG02] Xiaojin Zhu and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical Report CMU-CALD-02-107, Carnegie Mellon University, 2002.

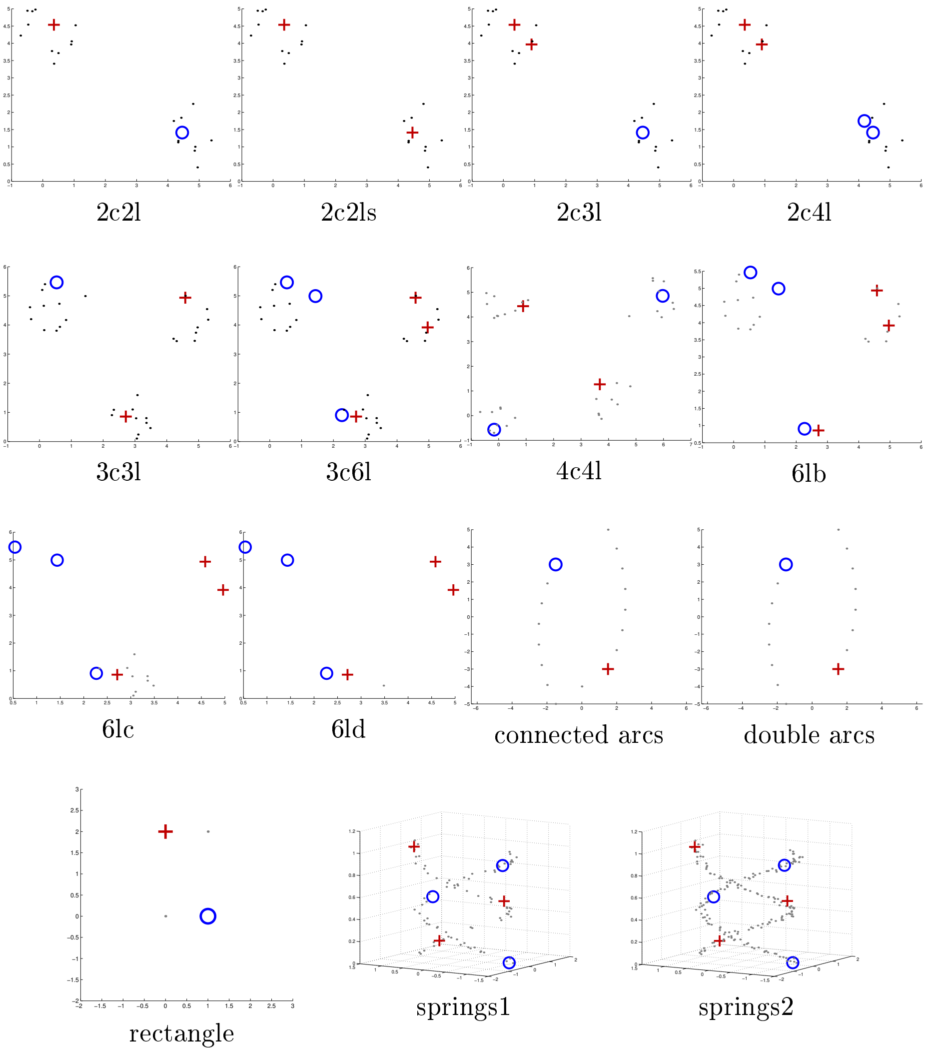


Figure 5: Several synthetic datasets. Large symbols are labeled data, dots are unlabeled data.

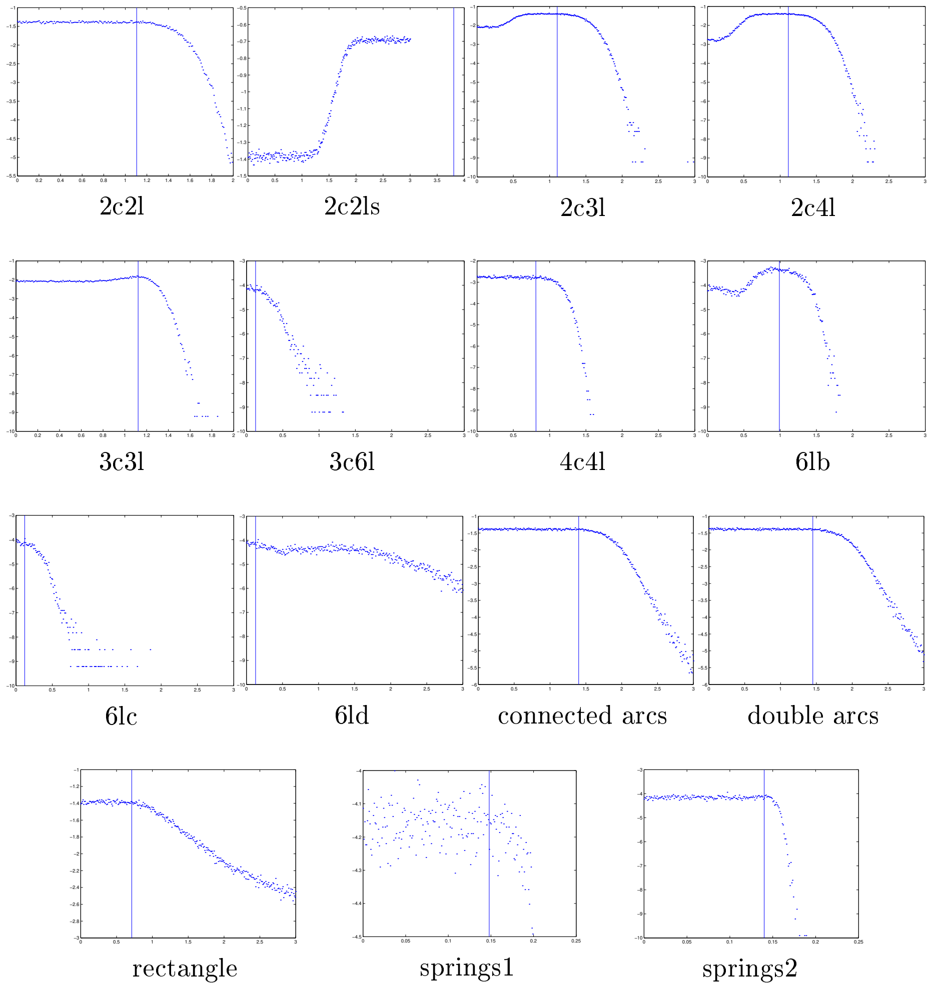


Figure 6: Log likelihood of labeled data (y-axis) vs. σ (x-axis) for the synthetic datasets. The vertical line marks the σ learned with gradient ascent after we add the prior Eq. (83).

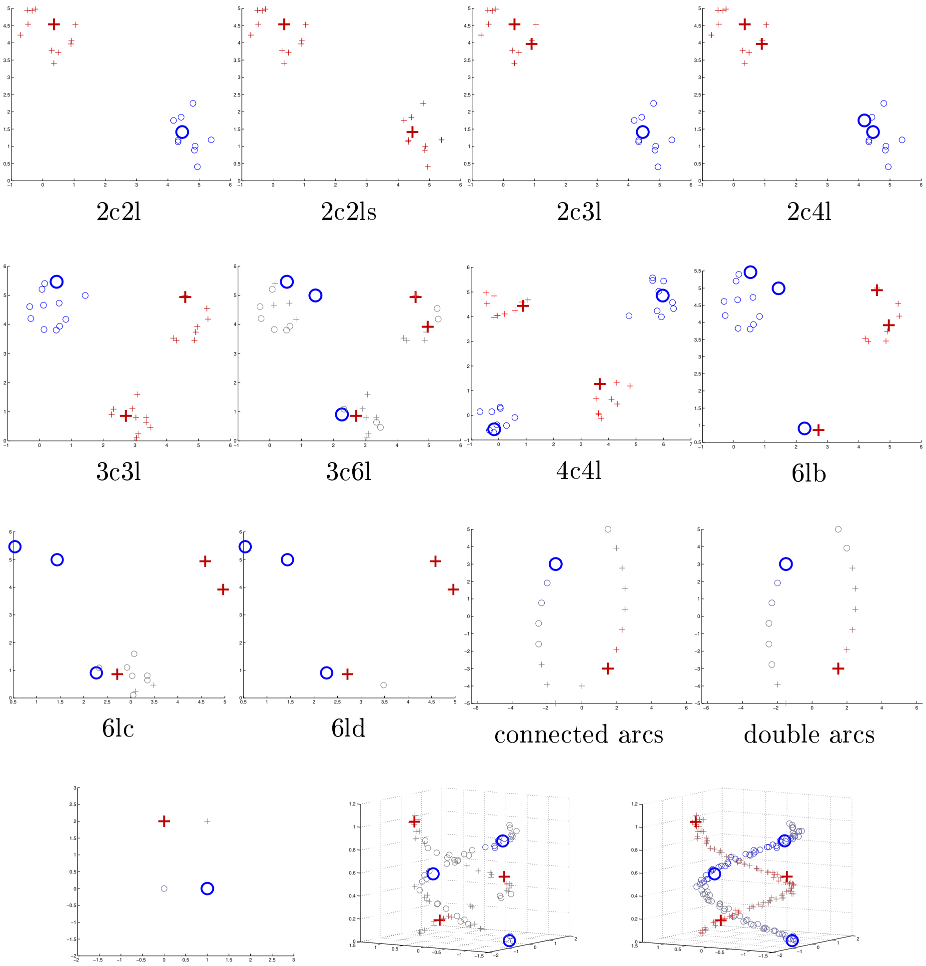


Figure 7: The labels of the synthetic datasets, under the learned σ with prior Eq. (83).