CPSC 550: Machine Learning II **2008/9 Term 2**

Lecture 12 — Feb. 26, 2009

Lecturer: Nando de Freitas *Scribe: Chris Nell*

This lecture discusses the problem of maximum likelihood parameter estimation in restricted Boltzmann machines, given observed data. Much of the theory presented here was originally developed by Laurent Younes in the context of general Boltzmann machines (or Gibbsian fields); particularly relevant are sections 3.1 and 3.2 of his paper. [1]

12.1 Review of Boltzmann Machines

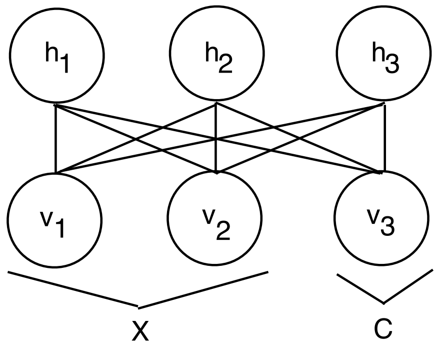

Figure 12.1. Graphical model of a two-layer restricted Boltzmann machine with 3 hidden and 3 visible units. Data corresponds to visible units X; class labels to visible units C.

Recall from the previous lecture the graphical structure of a restricted Boltzmann machine (RBM), as illustrated in Figure 12.1. Note that there are no intra-layer connections, but all inter-layer connections are present; this is unlike a general Boltzmann machine (BM) which can be fully connected. Where there are n_v visible and n_h hidden units, and where visible unit v_i is connected to hidden unit h_j with weight parameter ω_{ij}, the joint probability distribution of the RBM is given by:

$$P_\omega(h, v) = \frac{1}{Z(\omega)} \mathrm{e}^{+\sum_{i=1}^{n_v} \sum_{j=1}^{n_h} v_i \omega_{ij} h_j} \tag{12.1}$$

where $Z(\omega)$ is a normalization constant (partition function).

The parameter estimation problem in RBMs is the problem of learning values for the weight matrix ω, given observed (visible) data examples d. By learning these parameters, we also learn *features* of the data corresponding to each hidden node – the parameters ω_j^T characterize the data components relevant to node h_j. This insight implies an intuitive connection to self-taught learning. [2] Relevant general features can be learned through initial training on a large corpus of unlabeled data; subsequent training using labeled data will refine these features and learn their relationship to the labels.

Note that it can be useful to define an *energy* function, for RBMs this is:

$$E_\omega(v,h) = -\sum_{i=1}^{n_v}\sum_{j=1}^{n_h} v_i\omega_{ij}h_j \tag{12.2}$$

Using this, we rewrite Equation 12.1 as:

$$\begin{aligned} P_\omega(v,h) &= \frac{1}{Z(\omega)}\mathrm{e}^{-E_\omega(v,h)} && (12.3)\\ Z(\omega) &= \sum_v\sum_h \mathrm{e}^{-E_\omega(v,h)} && (12.4)\end{aligned}$$

where the summation in Equation 12.4 is over all possible values for the visible units $v = v_{1:n_v}$ and hidden units $h = h_{1:v_h}$.

Finally, recall the specific case of binary-valued units:

$$\begin{aligned} v_i \in \{0,1\} &\implies v \in \{0,1\}^{n_v}\\ h_j \in \{0,1\} &\implies h \in \{0,1\}^{n_h}\end{aligned}$$

In this case, we have seen that the conditional distributions are given by:

$$\begin{aligned} P_\omega(v|h) &= \prod_{i=1}^{n_v} \sigma(\omega_i h)^{v_i}[1-\sigma(\omega_i h)]^{1-v_i} && (12.5)\\ P_\omega(h|v) &= \prod_{j=1}^{n_h} \sigma(\omega_j^T v)^{h_j}[1-\sigma(\omega_j^T v)]^{1-h_j} && (12.6)\end{aligned}$$

where we denote by $\sigma(\cdot)$ the sigmoid function:

$$\sigma(x) = \frac{1}{1+\exp(-x)}$$

12.2 Parameter Estimation for (R)BMs

12.2.1 MLE for Boltzmann Machines

In order to derive a maximum likelihood parameter estimation algorithm for Boltzmann machines, we first determine the gradient of the log likelihood of the observed data d:

$$\nabla_\omega \log P_\omega(v = d) = \nabla_\omega \log \sum_h P_\omega(v = d, h) \tag{12.7}$$

Here, we marginalize h out of the joint to obtain the likelihood. Substituting Equation 12.3, we have:

$$= \nabla_\omega \log \frac{1}{Z(\omega)} \sum_h \mathrm{e}^{-E_\omega(v=d,h)} \tag{12.8}$$

$$= -\nabla_\omega \log Z(\omega) + \nabla_\omega \log \sum_h \mathrm{e}^{-E_\omega(v=d,h)} \tag{12.9}$$

Note that the first term in Equation 12.9 would disappear if we were working with a directed model, in which Z is not a function of the parameters ω. Unfortunately this is not the the case for BMs; rather, computing the partition function will entail significant computational effort.

Subsitituting Equation 12.4 and taking derivatives:

$$\begin{aligned} &= \frac{1}{Z(\omega)} \sum_v \sum_h \nabla_\omega E_\omega(v, h) \mathrm{e}^{-E_\omega(v,h)} \\ &\quad - \frac{1}{\sum_h \mathrm{e}^{-E_\omega(v=d,h)}} \sum_h \nabla_\omega E_\omega(v = d, h) \mathrm{e}^{-E_\omega(v=d,h)} \end{aligned} \tag{12.10}$$

Note that v is free in the first term of Equation 12.10, but clamped to the data in the second term. Finally, again recalling Equations 12.3 and 12.4:

$$\begin{aligned} &= \sum_v \sum_h \nabla_\omega E_\omega(v, h) P_\omega(v, h) \\ &\quad - \sum_h \nabla_\omega E_\omega(v = d, h) P_\omega(v = d, h) \end{aligned} \tag{12.11}$$

Clearly the gradient is minimized when the two terms in this expression cancel. Intuitively, this occurs when the model generates samples (first term) which "look like" the data (second term). This leads to an algorithm for ML parameter estimation, but before we present it, we consider the special case of RBMs.

12.2.2 MLE for RBMs

While the preceding analysis applies to general BMs, for RBMs we can employ the energy expression of Equation 12.2. First, note:

$$\nabla_{\omega_{ij}} E_\omega(v,h) = -\nabla_{\omega_{ij}} \sum_{i'=1}^{n_v}\sum_{j'=1}^{n_h} v_{i'}\omega_{i'j'}h_{j'} = -v_i h_j \tag{12.12}$$

Substituting into Equation 12.11, we have: [1]

$$\begin{aligned}\nabla_{\omega_{ij}} \log P_\omega(v=d) = &-\sum_v \sum_h v_i h_j P_\omega(v,h) \\ &+ \sum_h d_i h_j P_\omega(v=d,h)\end{aligned} \tag{12.13}$$

If we expand the second term of Equation 12.13 exploiting the independence of hidden units in an RBM, we have:

$$\sum_h d_i h_j P_\omega(v=d,h) = \sum_{h_1}\dots\sum_{h_j}\dots\sum_{h_{n_h}} d_i h_j \prod_{j'} P_\omega(v=d,h_{j'}) \tag{12.14}$$

where we recall each $h_{j'}$ is summed over the values in its domain; for the binary case, this is $\{0,1\}$. Noting that the summations over $h_{j'} \neq h_j$ contribute a factor of 1, this expression simplifies to:

$$= \sum_{h_j} d_i h_j P_\omega(v=d,h_j) = d_i \mathbb{E}(h_j) \tag{12.15}$$

In the binary unit case, we can write this explicitly as:

$$= d_i P_\omega(h_j = 1 | v = d) = d_i \sigma(\omega_j^T d) \tag{12.16}$$

Importantly, we have shown an analytic solution to the second term of Equation 12.11 in the case of RBMs. Since the first term has no such analytic solution, we will have to rely on a sampling procedure (such as Gibbs sampling) to estimate its value.[2]

[1]Equation 12.13 is often written using expected value notation as $\langle d_i h_j \rangle_{\tilde{p}} - \langle v_i h_j \rangle_\infty$.

[2]Unfortunately, Gibbs sampling can become "stuck" for millions of iterations – the implications of this are discussed in Section 12.2.4.

Finally, for an independent set of T data observations, we generalize Equation 12.13, yielding:

$$\begin{aligned}\frac{1}{T}\nabla_{\omega_{ij}} \log P(v=d) =& \frac{1}{T}\sum_{t=1}^{T}\sum_{h} d_{it}h_j P_\omega(v_t=d_t,h_j) \\ & -\sum_{v}\sum_{h} v_i h_j P_\omega(v,h) \end{aligned} \tag{12.17}$$

12.2.3 Younes' Algorithm

The analysis of Section 12.2.1 motivates Younes' algorithm (Algorithm 1) for maximum likelihood parameter estimation for general Boltzmann machines. This algorithm can be proved to converge. [1]

Algorithm 1 Younes' Algorithm

1. Sample $\tilde{h}^{(k)} \sim \prod_j P_\omega(v=d,h_j)$, for example using MCMC/SMC/...

2. Sample $\left(\tilde{\tilde{v}}^{(k)}, \tilde{\tilde{h}}^{(k)}\right)$, for example using Gibbs:

$$P_\omega(v,h) \rightarrow \begin{cases} \tilde{\tilde{v}}^{(k)} \sim P_\omega(v, \tilde{\tilde{h}}^{(k-1)}) \\ \tilde{\tilde{h}}^{(k)} \sim P_\omega(\tilde{\tilde{v}}^{(k)}, h) \end{cases}$$

3. Update weights, where η_k specifies the learning rate:

$$\omega_{ij}^{(k)} = \omega_{ij}^{(k-1)} + \frac{\eta_k}{T}\sum_{t=1}^{T}\left(d_{it}\tilde{h}_{jt}^{(k)} - \tilde{\tilde{v}}_{it}^{(k)}\tilde{\tilde{h}}_{jt}^{(k)}\right)$$

4. Repeat from step 1, with $k \rightarrow k+1$.

For the special case of RBMs, note that the analytic term derived in Equation 12.15 means sampling in step 1 of Younes' Algorithm is no longer necessary; we can instead calculate $\tilde{h}^{(k)}$ exactly.

12.2.4 Contrastive Divergence

As a result of the need to run Gibbs sampling to convergence in step 2, Younes' algorithm can be prohibitively slow. Contrastive Divergence (CD) is a biased approximation to the ML algorithm in which, rather than running the sampler's

Markov chain to convergence, takes a constant number of steps. The simplest variant (called CD1 because only one step is taken) is represented graphically in Figure 12.2 and described as Algorithm 2.

Algorithm 2 Constrastive Divergence (CD1)

1. Sample:

$$\begin{aligned}\tilde{h}^{(k)} &\sim P_\omega(v = d, h)\\ \tilde{\tilde{v}}^{(k)} &\sim P_\omega(v, \tilde{h}^{(k)})\\ \tilde{\tilde{h}}^{(k)} &\sim P_\omega(\tilde{\tilde{v}}^{(k)}, h)\end{aligned}$$

2. Update weights, where η_k specifies the learning rate:

$$\omega_{ij}^{(k)} = \omega_{ij}^{(k-1)} + \frac{\eta_k}{T}\sum_{t=1}^{T}\left(d_{it}\tilde{h}_{jt}^{(k)} - \tilde{\tilde{v}}_{it}^{(k)}\tilde{\tilde{h}}_{jt}^{(k)}\right)$$

3. Repeat from step 1, with $k \to k + 1$.

In addition to being much faster than Younes' algorithm, CD can actually provide better discriminative performance. Intuitively, this is because CD only simulates in the neighborhood of the data, whereas running the full chain entails simulating over the full space. This is illustrated in Figure 12.3, for the task of distinguishing 7s and 8s.

Contrastive Divergence will be discussed further in the next lecture. For further information, the tutorial of Lecun is recommended. [3]

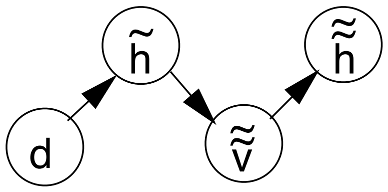

Figure 12.2. Graphical depiction of single-step contrastive divergence (CD1).

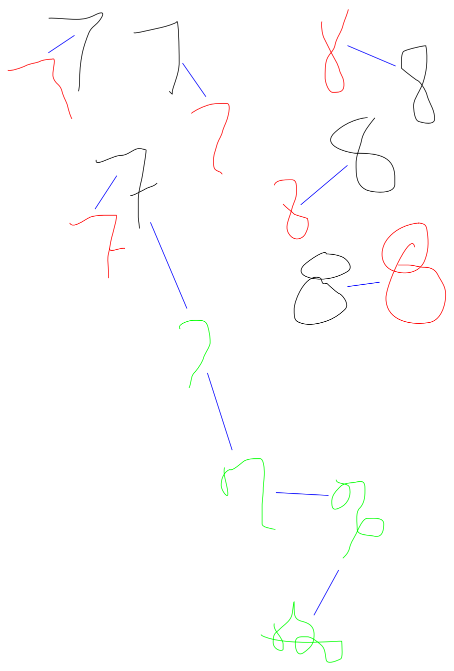

Figure 12.3. Intuitive distinction between Younes' ML algorithm and CD1. Training figures in black. ML simulates data for many steps as indicated by the green figures; these eventually become dissimilar to the data. CD1 simulates a single, similar datum for each training example, as illustrated in red.

Bibliography

[1] Younes, L.: Parameter estimation for imperfectly observed Gibbsian fields. Prob. Theory and Rel. fields 82, 625–645 (1989)

http://cis.jhu.edu/ younes/Preprints/younes.MRFpartial.pdf

[2] Raina, R., Battle, A., Lee, H., Packer, B., Ng, A.: Self-taught learning: transfer learning from unlabeled data, Proc. 24th ICML, 759–766, (2007)

http://www.stanford.edu/ hllee/icml07-selftaughtlearning.pdf

[3] LeCun, Y., Chopra, S., Hadsell, R., Marc'Aurelio, R., Huang, F.J.: A Tutorial on Energy-Based Learning. In: Bakir et al. (eds) "Predicting Strutured Data", MIT Press (2006)

http://yann.lecun.com/exdb/publis/pdf/lecun-06.pdf