Training Binary Restricted Boltzmann Machines with Contrastive Divergence

David Andrzejewski
andrzeje@cs.wisc.edu

August 19, 2009

1 Setup

1.1 Definitions

Let $\mathbf{x}$ be the observed (visible) binary variables (dimension V), $\mathbf{y}$ are the latent (hidden) binary variables (dimension H), and W is the set of model parameters (an $H \times V$ matrix). Let the probability of a given ($\mathbf{x}$,$\mathbf{y}$) configuration we given by

$$P(\mathbf{x}, \mathbf{y}; W) = \frac{1}{Z(W)} e^{-E(\mathbf{x},\mathbf{y};W)}$$
$$E(\mathbf{x}, \mathbf{y}; W) = -\mathbf{y}^T W \mathbf{x}$$

1.2 Maximum Likelihood Training?

Given training data D, we would like to select W to maximize the log-likelihood $L(W, D)$.

$$L(W, D) = -\langle\langle E(\mathbf{x}; w)\rangle_{P(\mathbf{y}|\mathbf{x};W)}\rangle_0 - \log(Z(W))$$

If we try to take the gradient of this, we get

$$\frac{\partial L}{\partial W} = -\langle\frac{\partial E}{\partial W}\rangle_0 + \langle\frac{\partial E}{\partial W}\rangle_\infty$$

where the second term follows from multiple applications of the chain rule. Since our energy function E is so simple, the gradient with respect to each W entry w_{ij} is simple as well

$$\frac{\partial E}{\partial w_{ij}} = -y_i x_j$$

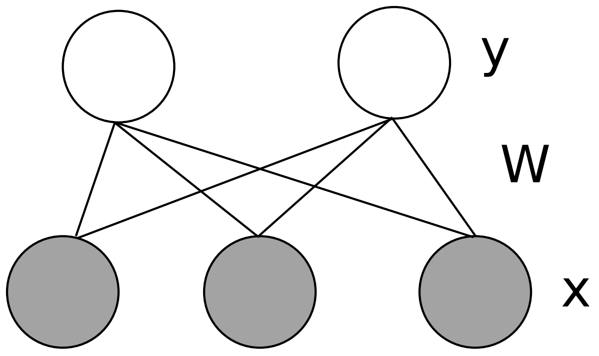

Figure 1: RBM graphical model.

However, since the second term is an expectation over the distribution of the model we cannot compute the gradient, making optimization of the log-likelihood infeasible. If we were able to do so, we could simply take steps of the form

$$w_{ij}^{t+1} = w_{ij}^{t} + \eta(\langle\langle\langle y_i x_j\rangle_{P(\mathbf{y}|\mathbf{x};W)}\rangle_0 - \langle\langle y_i x_j\rangle_{infty})$$

1.3 Contrastive Divergence Training

Maximum Likelihood traning can be shown to minimize the Kullback-Leibler divergence between the empirical training distribution and the distribution of the model

$$KL(P_0 \parallel P_\infty) = \sum_{(\mathbf{x},\mathbf{y})} P_0(\mathbf{x},\mathbf{y}) \log \frac{P_0(\mathbf{x},\mathbf{y})}{P(\mathbf{x},\mathbf{y};W)}.$$

In contrastive divergence learning, we instead seek to minimize the following related objective

$$CD_n = KL(P_0 \parallel P_\infty) - KL(P_n \parallel P_\infty)$$

where P_n is the distribution of a Markov chain with stationary distribution P_∞, started from P_0. This works pretty well, even for $n = 1$. The key benefit here is that the P_∞ terms cancel each other out (more or less). Furthermore, for the Restricted Boltzmann Machine it is very easy to sample from P_n due to the bipartite structure of the graphical model.

$$P(y_j = 1|\mathbf{x}, \mathbf{y}_{-j}; W) = \frac{1}{1 + \exp(-W_{j*}^T \mathbf{x})}$$

$$P(x_i = 1|\mathbf{x}_{-i}, \mathbf{y}; W) = \frac{1}{1 + \exp(-W_{*i}^T \mathbf{y})}$$

The contrastive divergence weight learning update is then

$$w_{ij}^{t+1} = w_{ij}^t + \eta(\langle\langle y_i x_j \rangle_{P(\mathbf{y}|\mathbf{x};W)}\rangle_0 - \langle y_i x_j \rangle_n). \tag{1}$$

Also note that the derivative $\frac{\partial E}{\partial w_{ij}}$ is bilinear, making it easy to calculate the expectations in Equation 1.

$$\langle\langle y_i x_j \rangle_{P(\mathbf{y}|\mathbf{x};W)}\rangle_0 = \frac{1}{n} \sum_{d=1}^{|D|} x_j^{(d)} P(y_i^{(d)} = 1|\mathbf{x}^{(d)}; W)$$

$$\langle y_i x_j \rangle_n = \frac{1}{n} \sum_{d=1}^{|D|} [x_j^{(d)}]_n P(y_i^{(d)} = 1|[\mathbf{x}^{(d)}]_n; W)$$

Here $[x_j^{(d)}]_n$ and $[\mathbf{x}^{(d)}]_n$ are the states of the visible units after n Gibbs sampling iterations.

1.3.1 Practical tweaks

In practice, there are a few modifications (observed in the code at http://www.cs.toronto.edu/~rsalakhu/code_AIS/rbm.m).

- Both hidden and visible units have *bias* terms, equivalent to appending all visible/hidden vectors with fixed 1 values.
- Weight *momentum* is used, where the current step is a weighted sum of the standard current step (Equation 1) and the step previously taken for this weight.
- Weight momentum *scheduling*, where the momentum parameter changes as training goes on.
- Weight *cost/penalty*, where larger weights are penalized.
- *Batch* learning, where the dataset is partitioned into subsets (the batches). A single weight step is then calculated and taken for each batch separately for each "epoch".
- Others?