Neural Networks
Lecture Notes
Restricted Boltzmann machines

Learning Goals

After studying the lecture material you should:

1. know what is meant by unsupervised learning
2. know what the energy function of an RBM looks like
3. know what is meant by an energy-based model
4. understand the learning rules for partially observed (R)BMs
5. know what is meant by contrastive divergence
6. be able to write down the learning rule for RBMs based on CD-1
7. be able to explain what a DBN is

Notes

Unsupervised learning

Unsupervised learning refers to finding hidden structure in unlabelled data. In other words, it aims to learn about the (hidden) states in our environment that cause our sensory input. We now move on to describe learning in partially observed Boltzmann machines as this form the basis for one kind of model that can learn complex features in an unsupervised manner.

Learning in partially observed Boltzmann machines

Learning in partially observed Boltzmann machines can be seen as a form of unsupervised learning. When learning internal representations we have $\mathbf{x} = (\mathbf{v}, \mathbf{h})$ where $\mathbf{v}$ is *observed* and $\mathbf{h}$ is *hidden* (e.g. images and their causes). In that case, we only care about the *marginal distribution* w.r.t. $\mathbf{v}$:

$$P(\mathbf{v}) = \sum_{\mathbf{h}} P(\mathbf{v}, \mathbf{h}) = \frac{1}{Z} \sum_{\mathbf{h}} \exp\left(-E(\mathbf{v}, \mathbf{h})\right) . \tag{1}$$

That is, we use the hidden variables to model as accurately as possible the distribution over visible units.

As a trick, we introduce the *free energy*:

$$F(\mathbf{v} = -\log \sum_{\mathbf{h}} \exp\left(-E(\mathbf{v}, \mathbf{h})\right)$$

which allows us to write

$$
\begin{aligned}
P(\mathbf{v}) &= \frac{1}{Z}\sum_{\mathbf{h}}\exp\left(-E(\mathbf{v},\mathbf{h})\right) && (2)\\
&= \frac{1}{Z}\exp\left(\log\sum_{\mathbf{h}}\exp(-E(\mathbf{v},\mathbf{h})\right) && (3)\\
&= \frac{1}{Z'}\exp(-F(\mathbf{v})) && (4)\\
& && (5)
\end{aligned}
$$

This is equivalent to an EBM defined over $\mathbf{v}$ with free energy F instead of energy E! All previous results hold with F replacing E, which leads to:

$$\Delta\theta \propto \mathbb{E}_P\left[\frac{\partial F}{\partial\theta}\right] - \mathbb{E}_{\hat{P}}\left[\frac{\partial F}{\partial\theta}\right]. \tag{6}$$

If we can compute the partial derivative of F then we can perform learning as before. This partial derivative can be written as

$$\frac{\partial F(\mathbf{v})}{\partial\theta} = \sum_{\mathbf{h}} P(\mathbf{h}\mid\mathbf{v})\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial\theta} \tag{7}$$

which allows us to rewrite (6) as

$$
\begin{aligned}
\Delta\theta &\propto \mathbb{E}_P\left[\frac{\partial F}{\partial\theta}\right] - \mathbb{E}_{\hat{P}}\left[\frac{\partial F}{\partial\theta}\right] && (8)\\
&= \sum_{\mathbf{v}} P(\mathbf{v})\sum_{\mathbf{h}} P(\mathbf{h}\mid\mathbf{v})\frac{\partial E(\mathbf{x})}{\partial\theta} - \sum_{\mathbf{v}}\hat{P}(\mathbf{v})\sum_{\mathbf{h}} P(\mathbf{h}\mid\mathbf{v})\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial\theta} && (9)\\
&= \sum_{\mathbf{x}} P(\mathbf{x})\frac{\partial E(\mathbf{x})}{\partial\theta} - \sum_{\mathbf{v}}\hat{P}(\mathbf{v})\sum_{\mathbf{h}} P(\mathbf{h}\mid\mathbf{v})\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial\theta} && (10)
\end{aligned}
$$

where we used $\mathbf{x} = (\mathbf{v},\mathbf{h})$.

This form suggests the following learning algorithm. In each update step:

- As before, run Gibbs sampling until convergence and compute the expected value under the model distribution (*negative phase*).
- Next, run for each training example Gibbs sampling with clamped inputs $\mathbf{v}$ until convergence (i.e. don't update $\mathbf{v}$) and compute the expected value under the empirical distribution (*positive phase*).
- Use the difference in expectations to update network weights.

This completes our derivation of the learning rule for partially observed Boltzmann machines.

Boltzmann machines are beautiful constructs since they learn the hidden causes that explain the statistical structure of our input data via local interactions (think: the brain). However, Boltzmann learning is terribly expensive since in each (of many thousands of) gradient steps we need to run positive and negative phase Gibbs samplers (for many thousands of steps). The time needed to run until convergence grows exponentially with the number of units which makes it impractical for anything other than toy problems...

Restricted Boltzmann machines

Learning can be made more efficient by making use of restricted Boltzmann machines. These are Boltzmann machines that are described by a partite graph in which the visible nodes are fully connected to the hidden nodes but there are no connections between visible or between hidden

nodes. Hence, a restricted Boltzmann machine (RBM) is a special kind of Boltzmann machine with energy function:

$$E(\mathbf{v}, \mathbf{h}) = -\mathbf{b}^T\mathbf{v} - \mathbf{c}^T\mathbf{h} - \mathbf{h}^T\mathbf{W}\mathbf{v} \tag{11}$$

The RBM was introduced by Smolensky under the name Harmonium in 1986 but it became more popular after the development of a fast learning algorithm by Hinton around 2006 [HOT06].

Importantly, due to its factorization properties, the free energy in an RBM is given by:

$$F(\mathbf{v}) = -\mathbf{b}^T\mathbf{v} - \sum_i \log(1 + \exp(c_i + \mathbf{w}_i^T\mathbf{v})) \tag{12}$$

which allows analytical computation of the gradient of the free energy. The partial derivatives are given by

$$\frac{\partial F}{\partial b_j} = -v_j \tag{13}$$

$$\frac{\partial F}{\partial c_i} = -p(h_i = 1 \mid \mathbf{v}) \tag{14}$$

$$\frac{\partial F}{\partial w_{ij}} = -p(h_i = 1 \mid \mathbf{v})v_j \tag{15}$$

Since we can directly compute these partial derivatives, we can directly use (6):

$$\Delta\theta \propto \mathbb{E}_P\left[\frac{\partial F}{\partial \theta}\right] - \mathbb{E}_{\hat{P}}\left[\frac{\partial F}{\partial \theta}\right]. \tag{16}$$

Computing the expectation for the model distribution can also be done in an efficient manner using the *block Gibbs sampler*:

- start at a random sample for $\mathbf{v}$: $\mathbf{v}^0 \sim P(\mathbf{v})$
- sample $\mathbf{h}$ given $\mathbf{v}$: $\mathbf{h}^0 \sim P(\mathbf{h} \mid \mathbf{v}^0)$
- sample $\mathbf{v}$ given $\mathbf{h}$: $\mathbf{v}^1 \sim P(\mathbf{v} \mid \mathbf{h}^0)$
- sample $\mathbf{h}$ given $\mathbf{v}$: $\mathbf{h}^1 \sim P(\mathbf{h} \mid \mathbf{v}^1)$
- ...
- sample $\mathbf{v}$ given $\mathbf{h}$: $\mathbf{v}^k \sim P(\mathbf{v} \mid \mathbf{h}^{k-1})$

This is still expensive as it requires block Gibbs sampling and running over the training examples per update step. In 2006 Hinton et al. proposed a heuristic approximation called *contrastive divergence* which was a breakthrough for training of restricted Boltzmann machines. Contrastive divergence (CD) simply assumes that we start at the training examples and only take k block Gibbs sampling steps with k typically equal to 1. CD-1 learning yields the following update equation:

$$\Delta\theta \propto \frac{\partial F(\mathbf{v}^{(1)})}{\partial \theta} - \frac{\partial F(\mathbf{v}^{(0)})}{\partial \theta}. \tag{17}$$

Deep belief networks

RBM can also be used to implement a form of deep learning. We simply train and RBM. Then we stack another RBM on top such that the output of the first RBM becomes the input to the second RBM. This procedure is repeated to yield a *deep belief network* (DBN).[1] The successes obtained with this model rekindled interest in (deep) neural networks and let to the explosion of reseach on this topic in recent years.

[1] The resulting model can be fine-tuned using a wake-sleep algorithm.

Reading material

For more practical information on RBMs, consult the paper [Hin10].

References

[Hin10] Geoffrey E Hinton. A Practical Guide to Training Restricted Boltzmann Machines. Technical Report UTML TR 2010-003, 2010.

[HOT06] Geoffrey E Hinton, S Osindero, and Y Teh. A fast learning algorithm for deep belief nets. *Neural Comput.*, 18:1527–1554, 2006.