Restricted Boltzmann Machines

— Lecture Notes —

Laurenz Wiskott
Institut für Neuroinformatik
Ruhr-Universität Bochum, Germany, EU

12 January 2017

— Summary —

Restricted Boltzmann machines (RBMs) are a particular type of graphical models that form a bipartite graph, meaning that the nodes can be split into two groups with no edges between nodes of the same group. This allows efficient Gibbs sampling, since all nodes of one group can be sampled simultaneously. The network as a whole can be trained to model a probability distribution over a large set of random variables. We consider here only binary RBMs, but there are also ones with continuous values.

1 Binary Restricted Boltzmann Machines can model probability distributions over binary variables. They are mathematically formulated in terms of an energy function that is then translated into a probability for any given state, a method known from physics. This section shows how to calculate conditional and marginal probabilities in the network, how to train it from data, and how to sample from the trained network.

2 Applications shows (i) how an RBM can be used to model human walking or running patterns and (ii) how a stack of RBMs can perform digit recognition and generation.

Contents

1 Binary Restricted Boltzmann Machines

1.1 The model $P(X, H)$

© 2016, 2017 Laurenz Wiskott (ORCID http://orcid.org/0000-0001-6237-740X, homepage https://www.ini.rub.de/PEOPLE/wiskott/). This work (except for all figures from other sources, if present) is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License, see http://creativecommons.org/licenses/by-sa/4.0/. If figures are not included for copyright reasons, they are uni colored, but the word 'Figure', 'Image', or the like in the reference is often linked to a freely available copy.

Core text and formulas are set in dark red, one can repeat the lecture notes quickly by just reading these; ♦ marks important formulas or items worth remembering and learning for an exam; ◇ marks less important formulas or items that I would usually also present in a lecture; + marks sections that I would usually skip in a lecture.

More teaching material is available at https://www.ini.rub.de/PEOPLE/wiskott/Teaching/Material/.
1 Binary Restricted Boltzmann Machines

(This section is largely based on (Bengio, 2009; Fischer and Igel, 2010).)

1.1 The model $P(X, H)$

We now want to consider one particular type of Gibbs network in greater detail, the Restricted Boltzmann Machine (RBM). Its graphical structure consists of an input layer with nodes/variables $X = (X_1, ..., X_I)^T$ and an output (usually referred to as hidden) layer with nodes/variables $H = (H_1, ..., H_J)^T$, see Fig. 1.1. The nodes have undirected edges from all input to all hidden nodes but neither between input nodes nor between hidden nodes. For simplicity the variables are assumed to be binary with values 0 and 1 for the time being.

![Graphical structure of an RBM](image)

Figure 1.1: Graphical structure of an RBM. © CC BY-SA 4.0

The probability distribution of the whole model is assumed to be of the form

\[
P(X, H) := \frac{1}{Z} \exp(-E(X, H)) \]  

with

\[
E(X, H) := -X^Tb - c^TH - X^TWH .
\]  

$Z$ is a constant that obviously has the value

\[
Z = \sum_{X, H} \exp(-E(X, H))
\]
to make the probability normalized.

This way of writing a probability distribution may seem a bit strange. It comes from statistical physics, where \( E \) would be the energy (D: Energie) of a certain state \((X, H)\) and states with high energy are unlikely while states with low energy are more likely. \( Z \) is called the partition function (D: Zustandssumme).

The energy (1.2) of an RBM is a bilinear function (linear in \( X \) as well as \( H \)) and therefore particularly simple. It is often useful to write the corresponding probability distribution like

\[
\begin{equation}
\begin{aligned}
P(X, H) &= \frac{1}{Z} \exp (-E(X, H)) \\
&= \frac{1}{Z} \exp (X^T b + c^T H + X^T W H) \\
&= \frac{1}{Z} \exp \left( X^T b + \sum_j (c_j + X^T w_j) H_j \right) \\
&= \frac{1}{Z} \exp (X^T b) \prod_j \exp \left( (c_j + X^T w_j) H_j \right),
\end{aligned}
\tag{1.4}
\end{equation}
\]

where \( w_j \) indicates the \( j_{th} \) column vector of \( W \) and \( H_j \) the \( j_{th} \) component of \( H \).

1.2 Marginal \( P(X) \) over the hidden variables

The hidden variables have been introduced to increase the representational power of the model, they are not of direct interest. Thus we have to compute

\[
\begin{equation}
\begin{aligned}
P(X) &= \sum_h P(X, h) \\
&= \frac{1}{Z} \exp (X^T b) \prod_j \exp \left( (c_j + X^T w_j) h_j \right) \\
&= \frac{1}{Z} \exp (X^T b) \prod_{h_1} \sum_{h_2} \prod_j \exp \left( (c_j + X^T w_j) h_j \right) \\
&= \frac{1}{Z} \exp (X^T b) \prod_{h} \sum_{h_{j=1}} \exp \left( (c_j + X^T w_j) h_j \right) \\
&= \frac{1}{Z} \exp (X^T b) \prod_j \left( 1 + \exp (c_j + X^T w_j) \right) \quad \text{(since } h_j \in \{0, 1\}\).
\end{aligned}
\tag{1.8}
\end{equation}
\]

The second last step is analogous to \( a_1 b_1 c_1 + a_1 b_1 c_2 + a_2 b_2 c_1 + a_2 b_2 c_2 + a_2 b_1 c_2 + a_2 b_2 c_1 + a_2 b_2 c_2 = (a_1 + a_2)(b_1 + b_2)(c_1 + c_2) \) and saves a lot of computation. Thus, \( P(X) \) can be computed fairly efficiently.

In analogy to the energy defined above, one defines free energy (D: freie Energie) \( F(X) \) such that \( P(X) \) can be written as

\[
\begin{equation}
P(X) = \frac{1}{Z} \exp (-F(X))
\tag{1.13}
\end{equation}
\]

with

\[
\begin{equation}
Z = \sum_X \exp (-F(X)).
\tag{1.14}
\end{equation}
\]
It is easy to relate free energy and energy as follows.

\[
\begin{align*}
\triangle & \quad \frac{1}{Z} \exp (-F(X)) \overset{(1.13)}{=} P(X) \\
\triangle & \quad = \sum_h P(X, h) \\
\triangle & \quad \overset{(1.1)}{=} \sum_h \frac{1}{Z} \exp (-E(X, h)) \\
\triangle & \quad \iff F(X) = -\ln \left( \sum_h \exp (-E(X, h)) \right).
\end{align*}
\]

1.3 Conditional \(P(H|X)\) of the hidden given the visible variables and \textit{vice versa}

When we sample the hidden variables given the visible variables we need the conditional probability

\[
\begin{align*}
\triangle & \quad P(H|X) = \frac{P(X, H)}{P(X)} \quad (1.19) \\
\triangle & \quad \overset{(1.1, 1.12)}{=} \prod_j \frac{\exp \left( (c_j + X^T w_j) H_j \right)}{1 + \exp (c_j + X^T w_j)} \quad (1.20) \\
\triangle & \quad = \prod_j \frac{\exp \left( (c_j + X^T w_j) H_j \right)}{1 + \exp (c_j + X^T w_j)} \quad (1.21) \\
\triangle & \quad = \prod_j P(H_j|X) \quad \text{(see below)} \quad (1.22)
\end{align*}
\]

This confirms that the hidden variables are conditionally independent given the visible variables.

The last step can be seen as follows. If we marginalize over one \(H_j'\) variable and we indicate the \(H\)-vector without this one variable as \(H_{-j'}\), we get

\[
\begin{align*}
\triangle & \quad P(H_{-j'}|X) = P(H_{-j', h_{j'} = 0}|X) + P(H_{-j', h_{j'} = 1}|X) \\
\triangle & \quad \overset{(1.21)}{=} \left( \frac{\exp \left( (c_j + X^T w_j) \cdot 0 \right)}{1 + \exp (c_j + X^T w_j)} + \frac{\exp \left( (c_j + X^T w_j) \cdot 1 \right)}{1 + \exp (c_j + X^T w_j)} \right) \prod_{j \neq j'} \frac{\exp \left( (c_j + X^T w_j) H_j \right)}{1 + \exp (c_j + X^T w_j)}. \quad (1.24)
\end{align*}
\]

Applying this marginalization over all but one hidden variables yields \(P(H_j|X)\) as stated above. Interestingly,

\[
\begin{align*}
\triangle & \quad P(h_j = 1|X) \overset{(1.22)}{=} \frac{\exp \left( (c_j + X^T w_j) \right)}{1 + \exp (c_j + X^T w_j)} \quad (1.25) \\
\triangle & \quad =: \sigma(c_j + X^T w_j) \quad (1.26) \\
\triangle & \quad =: y_j(X) \quad (1.27)
\end{align*}
\]

where \(\sigma(m) := \frac{e^m}{1 + e^m}\) is a standard sigmoidal activation function and \(y_j\) is the activity of a unit known from artificial neurons. Thus, RBMs can be nicely related to artificial neural networks with bias terms \(c_j\) and synaptic weight vectors \(w_j\).
Due to the symmetry of the energy w.r.t. $X$ and $H$, analogous expressions can be derived for $P(X|H)$ and $P(x_i=1|H)$. Thus, the neural network can also be run backwards, but with bias terms $b_i$ and the row vectors of matrix $W$ as weight vectors, so that feedback weights are the same as the feedforward weights.

1.4 Maximizing the likelihood

Training the model is done by adapting the parameters $b$, $c$, and $W$ such that the likelihood of the data given the model is maximized, where the data is a set of $\tilde{x}$-vectors. The units in the hidden layer are only auxiliary variables and may assume any values (0 or 1).

A typical method to maximize a function like the likelihood is gradient ascent. To do so here we need an estimate of the gradient.

1.5 The gradient of the model

To simplify the writing a bit we first note that by definition the probability \((1.13)\) is always positive (never zero), and since the logarithm is a strictly monotonically increasing function, maximizing the likelihood is equivalent to maximizing the log-likelihood

\[
\ln(P(X)) \overset{(1.13)}{=} -F(X) - \ln(Z). \tag{1.28}
\]

If $\theta$ indicates any parameter, we can calculate the gradient in a rather general form

\[
\frac{\partial \ln(P(X))}{\partial \theta} \overset{(1.28)}{=} - \frac{\partial F(X)}{\partial \theta} - \frac{1}{Z} \frac{\partial Z}{\partial \theta} \tag{1.29}
\]

\[
\frac{\partial F(X)}{\partial \theta} \overset{(1.14)}{=} - \frac{\partial F(X)}{\partial \theta} + \frac{1}{Z} \sum_{x'} \exp(-F(x')) \cdot \frac{\partial F(x')}{\partial \theta} \tag{1.30}
\]

\[
\frac{\partial F(X)}{\partial \theta} \overset{(1.13)}{=} - \frac{\partial F(X)}{\partial \theta} + \sum_{x'} P(x') \cdot \frac{\partial F(x')}{\partial \theta} \tag{1.31}
\]

Averaging this gradient over all data points $\tilde{x}$ yields the learning rule

\[
\langle \Delta \theta \rangle = \frac{1}{M} \sum_{\tilde{x}} \frac{\partial \ln(P(\tilde{x}))}{\partial \theta} \tag{1.32}
\]

\[
\langle \frac{\partial F(\tilde{x})}{\partial \theta} \rangle + \langle \frac{\partial F(x)}{\partial \theta} \rangle \tag{1.33}
\]

\[
\langle \frac{\partial F(x)}{\partial \theta} \rangle \tag{1.34}
\]

with $M$ indicating the number of data points available. Since the second term in (1.31) does not depend on the training data, it can be treated as a constant w.r.t. the averaging (not w.r.t. $\theta$) and does not need to be explicitly averaged. Notice also that, apart from the different signs, both terms in (1.33) are expectation values $\langle \cdot \rangle$ of $\partial F/\partial \theta$, the first one over the training data, the second one over the full model distribution.

The first term can be calculated, but for the second term the question arises how to average over or even sample from the unknown $P(x)$. I will get back to this below. Finally, one would, of course multiply the rhs with a learning rate $\eta$, but I dropped that for notational convenience.
1.6 Derivatives w.r.t. the parameters

We now need to calculate the derivatives of the free energy $F$ w.r.t. to the parameters $\theta$ of the model, namely $b_i$, $c_j$, and $W_{ij}$. Generally we find

\[ \frac{\partial F}{\partial \theta} = \frac{\partial}{\partial \theta} \left( -\ln \left( \sum_h \exp(-E(X, h)) \right) \right) \]  \hspace{1cm} (1.35)

\[ = - \left( \sum_{h'} \exp(-E(X, h')) \right)^{-1} \left( \sum_h \frac{\partial}{\partial \theta} \left( \exp(-E(X, h)) \right) \right) \]  \hspace{1cm} (1.36)

\[ = - \left( \sum_{h'} \exp(-E(X, h')) \right)^{-1} \left( \sum_h - \exp(-E(X, h)) \cdot \frac{\partial E(X, h)}{\partial \theta} \right) \]  \hspace{1cm} (1.37)

\[ = \left( \sum_{h'} \frac{1}{Z} \exp(-E(X, h')) \right)^{-1} \left( \sum_h \frac{1}{Z} \exp(-E(X, h)) \cdot \frac{\partial E(X, h)}{\partial \theta} \right) \]  \hspace{1cm} (1.38)

\[ = \left( \sum_{h'} P(X, h') \right)^{-1} \left( \sum_h P(X, h) \cdot \frac{\partial E(X, h)}{\partial \theta} \right) \]  \hspace{1cm} (1.39)

\[ = \left( P(X) \right)^{-1} \left( \sum_h P(X, h) \cdot \frac{\partial E(X, h)}{\partial \theta} \right) \]  \hspace{1cm} (1.40)

\[ = \sum_h \frac{P(X, h)}{P(X)} \cdot \frac{\partial E(X, h)}{\partial \theta} \]  \hspace{1cm} (1.41)

\[ = \sum_h P(h|X) \cdot \frac{\partial E(X, h)}{\partial \theta} \]  \hspace{1cm} (1.42)

\[ = \sum_h \left( \prod_j P(h_j|X) \right) \cdot \frac{\partial E(X, h)}{\partial \theta}, \]  \hspace{1cm} (1.43)
and for the specific parameters one finds

\[ \frac{\partial F(X)}{\partial b_i} = \sum_h P(h|X) \cdot \frac{\partial E(X, h)}{\partial b_i} \]  
\[ = \sum_h P(h|X) \cdot (-X_i) \]  
\[ = -X_i \sum_h P(h|X) \]  
(1.42)  

\[ \frac{\partial F(X)}{\partial c_j} = \sum_h P(h|X) \cdot \frac{\partial E(X, h)}{\partial c_j} \]  
\[ = \sum_h P(h|X) \cdot (-h_j) \]  
(1.43)  

\[ = \sum_h \left( \prod_{j'} P(h_{j'}|X) \right) \cdot (-h_j) \]  
\[ = -\sum_h h_j P(h_j|X) \prod_{j' \neq j} P(h_{j'}|X) \]  
\[ = -\sum_{h_{-j}} P(h_j = 1|X) \prod_{j' \neq j} P(h_{j'}|X) \]  
\[ = -P(h_j = 1|X) \prod_{j' \neq j} \sum_{h_{j'}} P(h_{j'}|X) \]  
\[ = -P(h_j = 1|X) \prod_{j' \neq j} 1 \]  
\[ = -P(h_j = 1|X) \]  
(1.52)  

\[ = -\sigma(c_j + X^T w_j) \]  
(1.53)  

\[ \frac{\partial F(X)}{\partial w_{ij}} = \sum_h P(h|X) \cdot \frac{\partial E(X, h)}{\partial w_{ij}} \]  
\[ = \sum_h P(h|X) \cdot (-X_i h_j) \]  
(1.44)  

\[ = X_i \sum_h P(h|X) \cdot (-h_j) \]  
\[ = -X_i y_j(X). \]  
(1.54)
With these derivatives we get the learning rules

\[
\Delta b_i = \frac{1}{N} \sum_{\hat{x}} f(\hat{x}) - \langle f(\hat{x}) \rangle_{\hat{x}} \quad (1.62)
\]
\[
\Delta b = \langle \hat{x} \rangle_{\hat{x}} - \langle x \rangle_x \quad (1.63)
\]
\[
\Delta c = \langle y(\hat{x}) \rangle_{\hat{x}} - \langle y(x) \rangle_x \quad (1.64)
\]
\[
\Delta W = \langle \hat{x} y^T(\hat{x}) \rangle_{\hat{x}} - \langle x y^T(x) \rangle_x \quad (1.67)
\]

Now, it seems we have derived a good set of gradient ascent learning rules and are done. The first terms in the equations above are averages over the data vectors \( \hat{x} \) and easy to compute. However, averaging over the true distribution of input vectors \( x \) of the model is untractable, because there are usually too many of them. Imagine \( x \) would be a 100-dimensional binary vector, then the average would have \( 2^{100} \) terms, just too expensive. One thus has to resort to sampling methods, which are discussed in the next section.

### 1.7 Sampling over the model input distribution

Since there are generally too many states of \( X \), one cannot sum their probabilities over all of them. Instead one samples them and hopes that the sample is representative for the whole distribution. Thus, one makes the approximation

\[
\langle f(\hat{x}) \rangle_{\hat{x}} = \frac{1}{N} \sum_{\hat{x}} f(\hat{x}) \approx \sum_x P(x) \cdot f(x) = \langle f(x) \rangle_x \quad (1.68)
\]

where \( \hat{x} \) are the \( N \) sampled input vectors, \( x \) are all \( 2^T \) input vectors, and \( f(x) \) is any function one wants to average. If the sampling works well, the sampled vectors \( \hat{x} \) are dense in regions where \( P(x) \) is high and thus summing over all \( x \) weighted by \( P(x) \) can be replaced by just summing over the \( \hat{x} \).

**Sampling a single input vector** can be done by starting from a data vector or any random input vector and then alternating choosing values for \( H \) and \( X \) according to \( P(H|X) \) and \( P(X|H) \), see (1.25). Doing the alternation many times should lead to an input vector \( \hat{x} \) that is sampled approximately according to \( P(x) \). (There are some formal requirements for this to work, e.g. there must be a nonvanishing chained transition probability from the initial input vector to any other one.)

However, this is still expensive. One therefore often samples \( H \) and \( X \) only once to get a new input vector. Thus starting from each data vector \( \hat{x} \) one creates a sample vector by first choosing an output vector \( \hat{h} \) given \( \hat{x} \) according to \( P(H|\hat{x}) \) and then choosing an input vector \( \hat{x} \) given \( \hat{h} \) according to \( P(X|\hat{h}) \). One can also sample forth and back more than once. The final learning rules are then

\[
\Delta b = \langle \hat{x} \rangle_{\hat{x}} - \langle \hat{x} \rangle_{\hat{x}} \quad (1.69)
\]
\[
\Delta c = \langle y(\hat{x}) \rangle_{\hat{x}} - \langle y(\hat{x}) \rangle_{\hat{x}} \quad (1.70)
\]
\[
\Delta W = \langle \hat{x} y^T(\hat{x}) \rangle_{\hat{x}} - \langle \hat{x} y^T(\hat{x}) \rangle_{\hat{x}} \quad (1.71)
\]

This algorithm is referred to as contrastive divergence (CD). CD-1 indicates one sampling iteration, CD-5 would indicate five sampling iterations, for instance.
2 Applications

Section title: Restricted Boltzmann Machines
Image: (Ludwig Boltzmann, ca. 1900, Wikimedia, © public domain, URL)
2.1 Modeling Human Motion (→ slides)

This section is based on (Taylor et al., 2006).

---

Left: A Gaussian-Binary RBM was used to model the motion of joint angles as a human walks or runs. The visible layer represents 4 time steps \( (t-3) \) to \( t \) (only three are shown in the figure) with up to 62 Gaussian units each, representing the joint angles. The sublayers \( (t-3) \) to \( (t-1) \) are connected to the sublayer \( t \) as well as to the hidden layer, which has 200 binary units. The architecture thereby becomes a conditional RBM, which is not much more complicated to deal with than a standard RBM.

Right: After training, the network can be used to generate naturally looking motion patterns. First initialize the visible sublayers \( (t-3) \) to \( (t-1) \) with true joint angle values of either walking or running. Then sample the hidden and the visible sublayer \( t \). Finally the states of the visible sublayers \( (t-2) \) to \( t \) are copied to sublayers \( (t-3) \) to \( (t-1) \) and the process repeats, thereby generating arbitrarily long sequences, which can then be animated.

Figure: (Taylor et al., 2006, Figs. 2, 3, URL)
2.2 MNIST Digit Recognition (→ slides)

This section is based on (Hinton et al., 2006).

In this application several RBMs are stacked on top of each other, resulting in a so-called Deep Belief Network (DBN). Handwritten digits similar to those on the right are fed into the visible layer on the right of the network; label information telling which digit is being shown is provided on the left side. The network was trained on 60,000 images, first layer by layer and then as a whole network. When clamping the visible layer on the right with an image and sampling from the network including the visible layer on the left, digit recognition can be performed. In that mode it reached a state-of-the-art performance of 1.25% error rate. When clamping the visible layer on the left with a digit class label and sampling from the network including the visible layer on the right, naturally looking handwritten digits can be generated, shown on the right. The network can be interactively explored on web-page http://www.cs.toronto.edu/~hinton/adi/index.htm.

Figure: (Hinton et al., 2006, Fig. 8, URL)
Acknowledgment

I thank Jan Melchior for his valuable help.

References


Notes


2.3 Hinton, Osindero, & Teh, 2006, Neur. Comp., 18(7):1527–1554, Fig. 8, http://www.csri.utoronto.ca/~hinton/absps/ncfast.pdf

Copyright protection level: 2/2