Restricted Boltzmann Machines

Justus H. Piater

Table of Contents

1. Preface

1.1. Generative Probabilistic Models

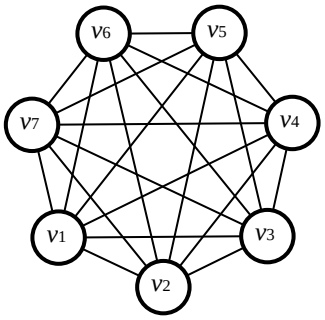
	Parametric	**Connectionist**
Provides explicit distribution/density function	yes	no
Capacity	low	high
Examples	Normal distribution; mixture of Gaussians	Restricted Boltzmann Machine; Variational Auto Encoder; Generative Adversarial Network

Here we are interested in the second column: We want to learn complex distributions from data in order to *sample* from them, even if we have to give up on an explicit distribution/density function.

2. Restricted Boltzmann Machines

2.1. Boltzmann Machine

Fully-connected MRF with bias term in the energy function



$$\begin{aligned} \mathbf{v} &\in \{0,1\}^D \\ P(\mathbf{v}) &= \frac{1}{Z}\exp(-E(\mathbf{v})) \\ E(\mathbf{v}) &= -\mathbf{v}^{\mathrm{T}} R \mathbf{v} - \mathbf{b}^{\mathrm{T}} \mathbf{v} \end{aligned}$$

Thus, the probability of unit i firing is constrained by a linear model (*logistic regression*) of the other units:

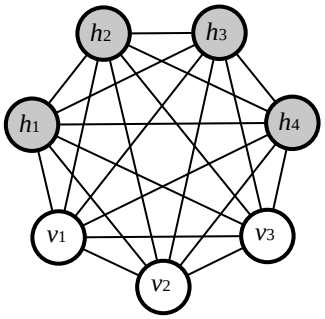
$$\begin{aligned} z_i &= \sum_j r_{ij} v_j + b_i \\ P(v_i = 1 \mid \mathbf{v}_{\setminus i}) &= \frac{\exp(z_i)}{\exp(0) + \exp(z_i)} \\ &= \frac{1}{1 + \exp(-z_i)} = \sigma(z_i) \end{aligned}$$

- Because of this firing rule, BM (and related energy-based models) are also known as ***stochastic neural networks***.
- $\tilde{P}(v_i = 1 \mid \mathbf{v}_{\setminus i}) = \frac{\tilde{P}(\mathbf{v})}{\tilde{P}(\mathbf{v}_{\setminus i})} = \exp(z_i)$. If $v_i = 1$ then we can omit this factor in z_i as above; if $v_i = 0$ then $z_i = 0$.
- Here and in the following, $\tilde{P}$ denotes an unnormalized distribution.

2.2. Boltzmann Machine With Hidden Units

Universal approximator

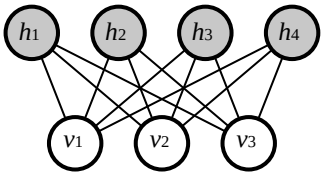
Visible units $\mathbf{v}$, hidden units $\mathbf{h}$:



$$E(\mathbf{v}, \mathbf{h}) = -\mathbf{v}^{\mathrm{T}} R \mathbf{v} - \mathbf{v}^{\mathrm{T}} W \mathbf{h} - \mathbf{h}^{\mathrm{T}} S \mathbf{h} - \mathbf{b}^{\mathrm{T}} \mathbf{v} - \mathbf{c}^{\mathrm{T}} \mathbf{h}$$

Difficult to train.

2.3. Restricted Boltzmann Machine



- *Bipartite* undirected graphical model consisting of one visible and one hidden layer.
- Commonly-used, stackable building block of deep probabilistic models.
- Evaluating $\boldsymbol{Z}$ is still intractable, but there are efficient training algorithms for RBM.

$$
\begin{aligned}
P(\mathbf{v}, \mathbf{h}) &= \frac{1}{Z}\exp(-E(\mathbf{v}, \mathbf{h})) \\
E(\mathbf{v}, \mathbf{h}) &= -\mathbf{v}^{\mathrm{T}} W \mathbf{h} - \mathbf{b}^{\mathrm{T}} \mathbf{v} - \mathbf{c}^{\mathrm{T}} \mathbf{h}
\end{aligned}
$$

2.4. Conditional Distributions in RBM

Thanks to the bipartite structure of RBM, these conditional probabilities factorize and can be computed exactly in closed form; this makes RBM relatively easy to train.

$$
\begin{aligned}
P(\mathbf{h} \mid \mathbf{v}) &= \frac{P(\mathbf{h}, \mathbf{v})}{P(\mathbf{v})} \\
&= \frac{1}{P(\mathbf{v})}\frac{1}{Z}\exp(\mathbf{v}^{\mathrm{T}} W \mathbf{h} + \mathbf{b}^{\mathrm{T}} \mathbf{v} + \mathbf{c}^{\mathrm{T}} \mathbf{h}) \\
&= \frac{1}{Z'}\exp(\mathbf{v}^{\mathrm{T}} W \mathbf{h} + \mathbf{c}^{\mathrm{T}} \mathbf{h}) \\
&= \frac{1}{Z'}\exp\left(\sum_j (\mathbf{v}^{\mathrm{T}} W)_j h_j + \sum_j c_j h_j\right) \\
&= \frac{1}{Z'}\prod_j \exp\left((\mathbf{v}^{\mathrm{T}} W)_j h_j + c_j h_j\right) \\
P(h_j = 1 \mid \mathbf{v}) &= \sigma\left((\mathbf{v}^{\mathrm{T}} W)_j + c_j\right) \\
P(\mathbf{h} \mid \mathbf{v}) &= \prod_j \sigma\left((2\mathbf{h} - 1) \odot (W^{\mathrm{T}} \mathbf{v} + \mathbf{c})\right)_j \\
P(\mathbf{v} \mid \mathbf{h}) &= \prod_i \sigma((2\mathbf{v} - 1) \odot (W\mathbf{h} + \mathbf{b}))_i
\end{aligned}
$$

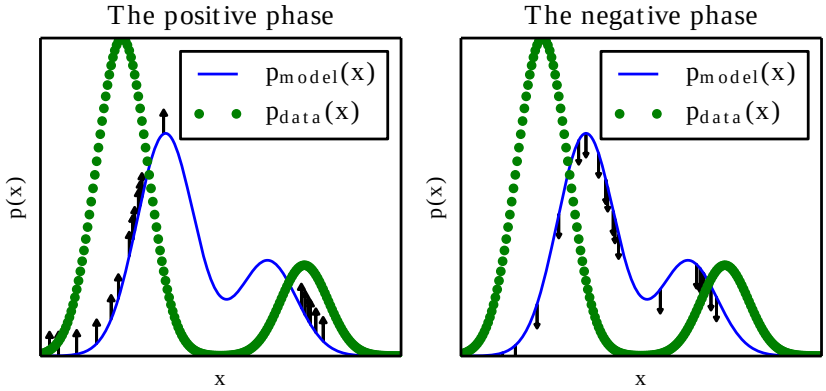
- Here, $\mathbf{v}$ is regarded as constant for the calculation of $P(\mathbf{h} \mid \mathbf{v})$.
- $P(h_j = 1 \mid \mathbf{v}) = \sigma(\cdot)$ follows analogously to the Boltzmann Machine (Section 2.1).
- The last two identities hold for Bernoulli distributions as discussed in the respective section of our chapter on Neural Networks; the $\odot$ operator denotes element-wise multiplication.
- $P(\mathbf{v} \mid \mathbf{h})$ follows an analogous derivation.

2.5. The Log-Likelihood Gradient

$$
\begin{aligned}
p(\mathbf{x};\boldsymbol{\theta}) &= \frac{1}{Z(\boldsymbol{\theta})}\tilde{p}(\mathbf{x};\boldsymbol{\theta}) \\
Z(\boldsymbol{\theta}) &= \sum_{\mathbf{x}} \tilde{p}(\mathbf{x};\boldsymbol{\theta}) \\
\nabla_{\boldsymbol{\theta}} \log p(\mathbf{x};\boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}} \log \tilde{p}(\mathbf{x};\boldsymbol{\theta}) - \nabla_{\boldsymbol{\theta}} \log Z(\boldsymbol{\theta}) \\
\nabla_{\boldsymbol{\theta}} \log Z &= \frac{1}{Z}\nabla_{\boldsymbol{\theta}} Z = \frac{1}{Z}\nabla_{\boldsymbol{\theta}} \sum_{\mathbf{x}} \tilde{p}(\mathbf{x}) = \frac{1}{Z}\sum_{\mathbf{x}} \nabla_{\boldsymbol{\theta}} \tilde{p}(\mathbf{x}) \\
&= \frac{1}{Z}\sum_{\mathbf{x}} \nabla_{\boldsymbol{\theta}} \exp(\log \tilde{p}(\mathbf{x})) \\
&= \frac{1}{Z}\sum_{\mathbf{x}} \exp(\log \tilde{p}(\mathbf{x}))\nabla_{\boldsymbol{\theta}} \log \tilde{p}(\mathbf{x}) \\
&= \frac{1}{Z}\sum_{\mathbf{x}} \tilde{p}(\mathbf{x})\nabla_{\boldsymbol{\theta}} \log \tilde{p}(\mathbf{x}) \\
&= \sum_{\mathbf{x}} p(\mathbf{x})\nabla_{\boldsymbol{\theta}} \log \tilde{p}(\mathbf{x}) \\
&= \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})} \nabla_{\boldsymbol{\theta}} \log \tilde{p}(\mathbf{x})
\end{aligned}
$$

- This derivation holds for any undirected graphical model with $p(\mathbf{x}) > 0$ everywhere.
- In the third equation, the first term is known as the ***positive phase*** of learning, and the second term as the ***negative phase***.
- All Z and $\tilde{p}$ here depend on $\boldsymbol{\theta}$, although it is often omitted to simplify notation.
- The final result implies that we do not need to compute the partition function Z because we do not need to compute $p(\mathbf{x})$; we just need to be able to sample from $p(\mathbf{x})$. We will do this using a ***Markov Chain Monte Carlo*** technique known as ***Gibbs sampling***.

2.6. Positive and Negative Phases



We draw instances from p_{data}, i.e., from the training set. If the model is different from the data distribution, it will tend to underestimate probabilities of the sample, resulting in the model to be adapted to increase these.

We draw instances from p_{model} using Gibbs sampling.. If the model is different from the data distribution, it will tend to overestimate probabilities of the sample, resulting in the model to be adapted to decrease these.

[Figure 18.1 from Goodfellow et al. 2016]

2.7. Naive MCMC Algorithm to Max. the Log Likelihood

- Repeat until convergence:

  a. **Positive Phase**

  i. Sample M instances $\mathbf{v}_1, \ldots, \mathbf{v}_M$ from the training set

  ii. $\mathbf{h}_m \sim P(\mathbf{h} \mid \mathbf{v}_m)$ for all $m = 1, \ldots, M$

  iii. $$\mathbf{g}_{\text{pos}} = \frac{1}{M} \sum_{m=1}^{M} \nabla_{\boldsymbol{\theta}} \log \tilde{P}(\mathbf{v}_m, \mathbf{h}_m; \boldsymbol{\theta})$$

  b. **Negative Phase**

  i. Generate a random sample of M instances $\tilde{\mathbf{v}}_1, \ldots, \tilde{\mathbf{v}}_M$

  ii. For $m = 1, \ldots, M$ (Gibbs sampling of M Markov chains):

  - Repeat K times: $\tilde{\mathbf{h}}_m \sim P(\mathbf{h} \mid \tilde{\mathbf{v}}_m)$ and $\tilde{\mathbf{v}}_m \sim P(\mathbf{v} \mid \tilde{\mathbf{h}}_m)$

  iii. $$\mathbf{g}_{\text{neg}} = \frac{1}{M} \sum_{m=1}^{M} \nabla_{\boldsymbol{\theta}} \log \tilde{P}(\tilde{\mathbf{v}}_m, \tilde{\mathbf{h}}_m; \boldsymbol{\theta})$$

  c. $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha(\mathbf{g}_{\text{pos}} - \mathbf{g}_{\text{neg}})$

At Step 1.a.ii, individual h_j are easily sampled as shown above.

2.8. Stochastic Maximum Likelihood

The naive algorithm starts new Markov chains at every top-level iteration. This SML algorithm (also known as ***Persistent Contrastive Divergence***, PCD) instead continues the Markov chains from the previous iteration. This allows the *burn-in* K to be drastically reduced, and works well if the negative phase moves relatively fast compared to the positive phase.

1. Generate a random sample of M instances $\tilde{\mathbf{v}}_1, \ldots, \tilde{\mathbf{v}}_M$

2. Repeat until convergence:

   a. **Positive Phase**

      i. Sample M instances $\mathbf{v}_1, \ldots, \mathbf{v}_M$ from the training set

      ii. $\mathbf{h}_m \sim P(\mathbf{h} \mid \mathbf{v}_m)$ for all $m = 1, \ldots, M$

      iii. $$\mathbf{g}_{\text{pos}} = \frac{1}{M} \sum_{m=1}^{M} \nabla_\theta \log \tilde{P}(\mathbf{v}_m, \mathbf{h}_m; \theta)$$

   b. **Negative Phase**

      i. For $m = 1, \ldots, M$ (Gibbs sampling of M Markov chains):

         - Repeat K times: $\tilde{\mathbf{h}}_m \sim P(\mathbf{h} \mid \tilde{\mathbf{v}}_m)$ and $\tilde{\mathbf{v}}_m \sim P(\mathbf{v} \mid \tilde{\mathbf{h}}_m)$

      ii. $$\mathbf{g}_{\text{neg}} = \frac{1}{M} \sum_{m=1}^{M} \nabla_\theta \log \tilde{P}(\tilde{\mathbf{v}}_m, \tilde{\mathbf{h}}_m; \theta)$$

   c. $\theta \leftarrow \theta + \alpha(\mathbf{g}_{\text{pos}} - \mathbf{g}_{\text{neg}})$

See also Hinton (2012).

2.9. Drawing Instances from an RBM

> ***Note***
>
> There is nothing new to be said, as this is the core operation of the *negative phase* of training.
>
> Technically, a single Markov chain should suffice. However, if the chain does not *mix* well (i.e., it fails to cover all modes of the distribution), it might help to run multiple Markov chains and select one at random each time an instance is requested. This is ***not*** a general solution however.
>
> There is nothing wrong with continuing to use the Markov chains used during training.

3. Exercises

3.1. Exercises

1. Calculate $\nabla_{\boldsymbol{\theta}} \log \tilde{P}(\mathbf{v}, \mathbf{h}; \boldsymbol{\theta})$ in an RBM!

   Restricted Boltzmann Machine (Section 2.3):

   $$\begin{aligned} \tilde{P}(\mathbf{v}, \mathbf{h}) &= \exp(\mathbf{v}^{\mathrm{T}} W \mathbf{h} + \mathbf{b}^{\mathrm{T}} \mathbf{v} + \mathbf{c}^{\mathrm{T}} \mathbf{h}) \\ \log \tilde{P}(\mathbf{v}, \mathbf{h}) &= \mathbf{v}^{\mathrm{T}} W \mathbf{h} + \mathbf{b}^{\mathrm{T}} \mathbf{v} + \mathbf{c}^{\mathrm{T}} \mathbf{h} \end{aligned}$$

   $$\begin{aligned} \frac{\partial}{\partial W_{ij}} \log \tilde{P}(\mathbf{v}, \mathbf{h}) &= v_i h_j & \nabla_W \log \tilde{P}(\mathbf{v}, \mathbf{h}) &= \mathbf{v}\mathbf{h}^{\mathrm{T}} \\ \frac{\partial}{\partial b_i} \log \tilde{P}(\mathbf{v}, \mathbf{h}) &= v_i & \nabla_b \log \tilde{P}(\mathbf{v}, \mathbf{h}) &= \mathbf{v} \\ \frac{\partial}{\partial c_j} \log \tilde{P}(\mathbf{v}, \mathbf{h}) &= h_j & \nabla_c \log \tilde{P}(\mathbf{v}, \mathbf{h}) &= \mathbf{h} \end{aligned}$$

2. Design an RBM with four visible units such that, when you draw samples from it, the first two and last two units are active together, respectively (i.e., if and only if the first unit is active, so is the second, and likewise for the third and fourth units). How many hidden units do you need? What are the parameters of the RBM?

   Warning

   It is easy to come up with simple distributions with a few distinct modes over a small number of visible nodes that turn out not to be representable by an RBM with a correspondingly small number of hidden units. If this is attempted, learning will fail to converge.

3. What will happen during training of a distribution with disjoint modes as in Ex. 2?

4. References

4.1. References

I. Goodfellow, Y. Bengio, A. Courville, *Deep Learning*[1], MIT Press 2016.

G. Hinton, "A Practical Guide to Training Restricted Boltzmann Machines[2]". *Neural Networks: Tricks of the Trade*, 2nd edition, pp. 599–619, 2012.

[1] https://www.deeplearningbook.org/
[2] https://www.cs.toronto.edu/~hinton/absps/guideTR.pdf