RBM Learning

Frank Lu

April 19, 2022

Contents

1 RBM learning Redux

In this section, we will introduce the Restricted Boltzmann Machine (RBM) in a machine learning context.

An RBM is a generative neural network model, with two layers of nodes – the hidden layer and the visible layer. A generative model aims to learn a probability distributions, and can also generate samples of their own.

In machine learning, they are quite useful for a whole bunch of reasons. * fix this!

Also is interesting from a quantum many-body physics point of view.

For instance, Deep belief networks are made from stacked RBMs.

In practice, however, RBMs have been supplanted by other generative models such as Autoencoders and GANs.

1.1 Structure and notation

RBMs contain a hidden and a layer of nodes, with hidden and visible units respectively. In graph theoretic terms, the two layers form a bipartite graph where there is an edge between every hidden unit and every visible unit. The word "Restricted" in the name refers to that there are no edges between nodes from the same layer. (In contrast, Boltzmann Machines do not have this restriction.)

Consider an RBM with N_H hidden units and N_V visible units. There are a corresponding set of weights for the hidden units and visible units. We can represent these as vectors, denoted as H_h for $h = 1, 2, \ldots, N_H$, and V_v for $v = 1, 2, \ldots, N_V$ respectively. There is a weight associated with each edge of the graph. This is represented as a matrix of weights, where W_{hv} is the weight between the hth hidden unit and the vth visible unit. Overall, the weights for an RBM is determined by $\{H_h, V_h, W_{hv}\}$.

Fig. 1 gives an example of an RBM as a diagram.

1.2 RBM as a probability distribution

We can consider the RBM as a function on binary strings of length N_V RBM : $\{0,1\}^{N_V} \to \mathbb{R}$

$$\text{RBM}(\mathbf{x}) = \sum_{\mathbf{h}} e^{-E(\mathbf{x},\mathbf{h})}$$

RBMs are an energy based model that uses an energy function in determining its output.

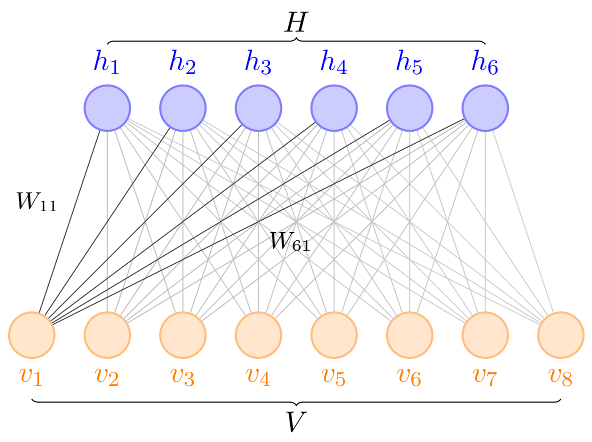

Figure 1: A diagram of an RBM with 6 hidden units and 8 visible units.

Here, $\mathbf{v}, \mathbf{h}$ are binary vectors.

$$E(\mathbf{v}, \mathbf{h}) = -\sum_{v=1}^{N_V} V_v \mathbf{v}_v - \sum_{h=1}^{N_H} H_h \mathbf{h}_h - \sum_{v=1}^{N_V} \sum_{h=1}^{N_H} \mathbf{h}_h W_{hv} \mathbf{v}_v \tag{1}$$

Alternatively, Eq. (1) can be written as follows when vectorised.

$$E(\mathbf{v}, \mathbf{h}) = -V^T \mathbf{v} - H^T \mathbf{h} - \mathbf{h}^T W \mathbf{v} \tag{2}$$

Probability distribution: $\Pr(\mathbf{x}) = \frac{1}{Z}\mathrm{RBM}(\mathbf{x})$

$Z = \sum_{\mathbf{x}} \mathrm{RBM}(X = \mathbf{x})$ (In the machine learning literature, it is common to use $\mathbf{a}$ and $\mathbf{b}$ for the hidden and visible unit weights.)

In order for $\mathrm{RBM}(\mathbf{x})$ to be a probability distribution, it needs to be normalised using the partition function Z.

Indeed, RBM 'learning' consists of adjusting the weights W, V, H to model a probability distribution.

1.3 Part 2: RBM learning

Suppose $p_{data}(\mathbf{x})$ is the target distribution we would like our RBM to learn. Then, training requires a list of m samples $\mathbb{X} = [\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(m)}]$ from $p_{data}(\mathbf{x})$.

Training an

1.3.1 Simple worked example

We will go through a simple example of a RBM learning a probability distribution. Consider the distribution of the sum of two dice rolls. Indeed, this is a binomial distribution.

Fig. 2 is a plot of the samples, drawn from 10000 trials.

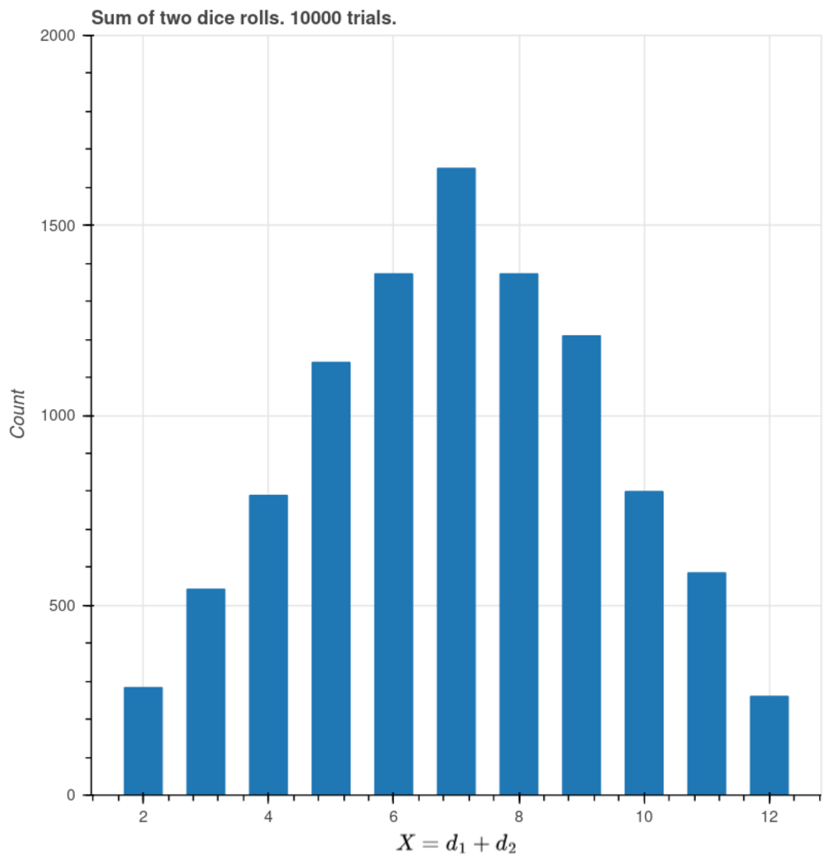

Figure 2: The sum of two dice rolls

2 Burning questions

- Why use the Sigmoid function?

Answer: It's differentiable at non 0 places. Gets infinitely better as it gets closer to 0.

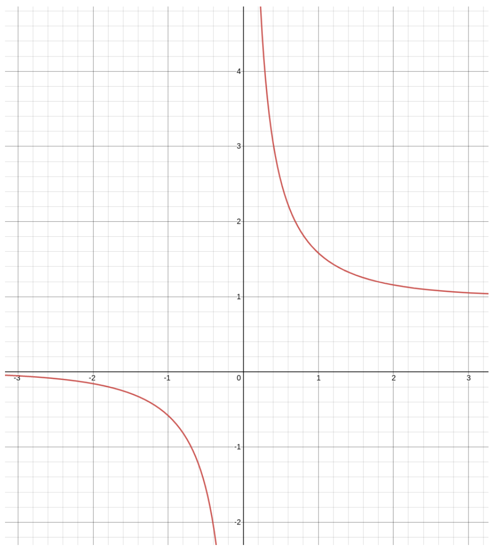

Figure 3: The Logistic Sigmoid function

- Multivariate calculus...

The probability that the network assigns to a training image can be raised by adjusting the
and biases to lower the energy of that image and to raise the energy of other images, especial
that have low energies and therefore make a big contribution to the partition function. The de
of the log probability of a training vector with respect to a weight is surprisingly simple.

$$\frac{\partial \log p(\mathbf{v})}{\partial w_{ij}} = \langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model}$$

where the angle brackets are used to denote expectations under the distribution specified
subscript that follows. This leads to a very simple learning rule for performing stochastic s
ascent in the log probability of the training data:

$$\Delta w_{ij} = \epsilon(\langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model})$$

where ϵ is a learning rate.

3 Very rough notes

The family of gradient based leaning algorithms commonly used by RBMs is called *Contrastive Divergence.*

3.1 The RBM itself

Assume your training set of binary images. Then, you can feed your image $\succapprox$ into the RBM.

$$p(\mathbf{v}) = \frac{1}{Z}\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})} \tag{4}$$

Your **h** vectors are binary vectors, and the sum is taken over all possible values of

3.2 Contrastive Divergence

Certain models result in an *unnormalised probability distributions* (page 582, textbook), where the total probability is not 1.

3.2.1 Partition Function

To normalise such a distribution $\tilde{p}(\mathbf{x})$,

$$p(\mathbf{x}) = \frac{1}{Z}\tilde{p}(\mathbf{x}) \tag{3}$$

And, Z which is called the *partition function* is defined as follows for continuous variables.

$$Z = \int \tilde{p}(\mathbf{x})d\mathbf{x} \tag{4}$$

Or, for discrete variables

$$Z = \sum_{\mathbf{x}} \tilde{p}(\mathbf{x}) \tag{5}$$

Note that the argument to Z is often omitted in the literature.

Usually, it is intractable to compute Z directly, so an approximation is used.

!Gibbs distribution! – distribution of product of clique potentials.

4 Machine learning notes

Here, I will briefly give the background for machine learning in the context of RBMs.

Machine learning algorithms are used when it is difficult to directly write an algorithm to do some task. For example, computer vision, or speech recognition.

Mathematically, we want some function f given some input to produce an output, but f cannot be readily defined so we opt for a (not necessarily probablistic) but 'good enough' model. In this case, it is sensible to choose a family of functions (?) $f(x;\theta)$ parameterised by θ as our model. In a sense, a good model will have a good choice of f, in that f is able to approximate the distribution of our outputs, and a good choice of θ.

We will start with supervised learning, where we have a set $X = \{x_1, x_2, \ldots, x_k\}$ of k inputs and $Y = \{y_1, y_2, \ldots, y_k\}$ outputs. Our goal is to find $f : X \to \hat{Y}$, where $\hat{Y} = \{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_k\}$ and for all i, $f(x_i) = \hat{y}_i$. The hats on the Ys is to show that this just an approximation for actual y values. Of course, a good model will be one such that for all i, $\hat{y}_i \approx y_i$.

Linear regression is a good example of a machine learning algorithm.

4.1 Finding a good parameterisation

What is a good choice of θ? It only makes sense to ask this question if we can measure the performance of a particular choice of θ. To this end, let $C(f, \theta)$ be the cost function that produces a scalar output. So now, the best choice of θ is one that optimises C.

A simple and commonly used cost function is the mean squared error, $C(Y, \hat{Y}) = \sum_i (y_i - \hat{y}_i)^2$.

4.2 Mathematical fundamentals

From scratch spends a lot of time talking about calculating gradients to functions.

Suppose, $f : A \to A$, $g : (A, A) \to A$. Then, given the map $Z(x, y) = (f \circ g)(x, y)$, we want to find out how the gradient of the output with respect to the input. More precisely, $\frac{dZ}{dx}$ and $\frac{dZ}{dy}$. (This is done using the chain rule).

The backward pass is precisely the act of finding gradients with respect to the input variables.

5 Understanding: Learning for undirected models

5.1 Partition Functions

5.2 Maximum likelihood

“Negative phase is very hard”. So naive algorithm isn’t used.

5.3 Contrastive Divergence

5.4 Gibbs Sampling

6 Thinking

6.1 Why is it hard to draw from RBMs?

6.2 What are the common RBM learning algorithms?

- Contrastive Divergence
- Stochastic maximum likelihood
- Pseudo-likelihood