Contents

1 Burning questions

- Why use the Sigmoid function?

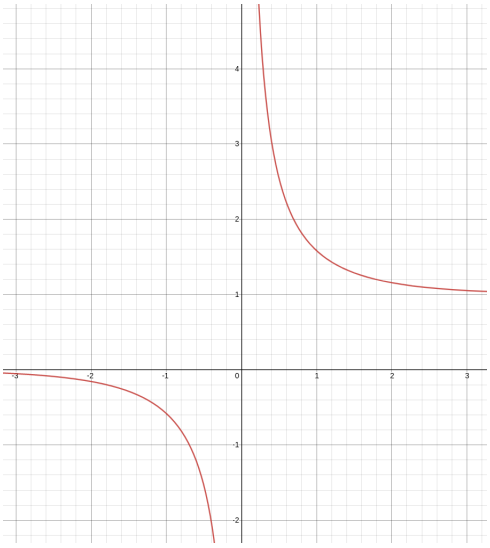

Figure 1: The Logistic Sigmoid function

- 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.

2 Very rough notes

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

2.1 The RBM itself

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

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

Your $\mathbf{h}$ vectors are binary vectors, and the sum is taken over all possible values of

2.2 Contrastive Divergence

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

2.2.1 Partition Function

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

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

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

$$Z = \int \tilde{p}(\mathbf{x}) d\mathbf{x} \tag{2}$$

Or, for discrete variables

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

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.

3 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.

3.1 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, $C(Y, \hat{Y}) = \sum_i (y_i - \hat{y}_i)^2$

3.2 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.

4 Understanding: Learning for undirected models

4.1 Partition Functions

4.2 Maximum likelihood

"Negative phase is very hard". So naive algorithm isn't used.

4.2.1 Contrastive Divergence