Restricted Boltzmann Machines

INM427 - Neural Computation
MSc in Data Science
Artur Garcez and Son N. Tran

In this tutorial we study Restricted Boltzmann Machines (RBMs). RBMs have a visible layer and a hidden layer. The RBM is a simpler form of the Boltzmann machine where there is no connection between units in the same layer and no recursive connection either.

1 Training RBMs

RBMs learn to represent the data distribution $P(v)$ which is defined as:

$$P(v) = \sum_h P(v,h) = \sum_h \exp(-E(v,h))/Z \tag{1}$$

where $E(v,h) = -\sum_{ij} v_j w_{ij} h_j - \sum_i a_i v_i - \sum_j b_j h_j$ is the energy function, and $Z = \sum_{v,h} \exp(-E(v,h))$ is the partition function. Training RBMs is difficult because the partition function is intractable, i.e. cannot be computed exactly for large data sets. Approximation algorithms such as Contrastive Divergence are then used for efficient learning.

One interesting property of RBMs is that we can use it for classification. For this, we add the labels to the visible layer. Let's say we have data x with label y. The visible layer can become $v = \{x, y\}$. Here, y is called a one-hot vector, i.e. a vector with value 1 in the position corresponding to the label and 0 otherwise. For example, the label 3 of 10 classes denoting e.g. the digits from 0 to 9, is represented as [0 0 1 0 0 0 0 0 0 0]. For classification, after training the RBM in the usual way we can predict the label which gives the highest propability $P(y|x)$. However, notice that the training of the RBM seeks to learn the data distribution $P(v) = P(x, y)$; it is not designed for classification. Thus, classification performance should improve if we train the RBM to represent $P(y|x)$ instead.

Normally, we can investigate what have been learned by an RBM by visualizing the feature detectors. Each feature detector is a weight vector connecting the visible layer to a hidden unit. The meaning of a feature detector is that if the input data is close to a feature detector it will be more likely to activate the corresponding hidden unit.

2 Sparsity

Letters, punctuation and spaces are a sparse coding for the English language (and many other languages). There are several ways to learn sparse features from RBMs by constraining the RBM such that only a small number of hidden neurons gets activated by each training data point. One way is to constrain

the learning by only allowing a small number of training examples to activate a hidden unit. In this way, the feature detectors will learn some common patterns in a group of data, as follows.

- 1. Open file `rbm_lab.m`
- 2. Run the script that will show several plots. Can you find an explanation for each plot? The problem being tackled here is known as image denoising.
- 3. Change `conf.gen=1;` to `conf.gen=0;`. Run the script again and compare the classification result on the test set with the previous one. Which one is better? Can you find out why?
- 4. Investigate why in comparison with the denoising result from the previous case (`conf.gen=1;`), the reconstruction from impaired images is not as good when `conf.gen=0;`.
- 5. Order the feature detectors according to the L_1 norm (a.k.a. taxicab norm; the distance derived from this norm is called the Manhattan distance); see `%RankfeaturewithL1norm`.
- 6. Set `conf.gen=1;` back again. Set the number of hidden unit to 100, and adjust the learning rate to reduce the validation error. After that increase the number of hidden units to 1000, and adjust the learning rate, also to reduce the validation error. Give your opinion on the relationship between the learning rate and the network capacity.
- 7. Fix the number of hidden units at 100. Change the generalization parameter to `conf.params(4)=0.01` and the sparsity parameters to `conf.lambda=50;`. Inspect what has changed in the visualization fo the feature detectors. Why are they different from the previous experiments? What happens if you increase `conf.lambda` to a (much) higher value?