Deep Networks

INM427 - Neural Computation
MSc in Data Science

Dr. Artur Garcez and Son N. Tran

1 Introduction

Deep Learning has become a very useful tool for Data Science. In this tutorial, we study Deep Learning which combines two phases of learning: (1) unsupervised pre-training and (2) supervised fine tuning. As we have seen in previous lectures, neural networks with a single hidden layer can learn complex functions. Nevertheless, one main research question is: can perfomance be increased by increasing the number of hidden layers? In many applications, one hidden layer can achieve good performance and adding more hidden layers does not produce a significant improvement. A main problem is the "vanishing gradients" caused by back-propagating the errors through many layers. Recent work in Deep Learning has shown that the problem can be managed by applying unsupervised pre-training to each pair of layers, and using such pre-trained parameters as initialization for supervised back-propagation (called fine-tuning). Researchers finally have shown that multi-layer neural networks, now known as deep networks, can be useful at improving performance, and this new way of learning is now called Deep Learning. We have seen how Restricted Botlzmann Machines (RBMs) can be used for unsupervised learning. RBMs are very suitable for the pre-training phase of deep networks. We now use them to pre-train a deep network, which then will be fine-tuned by standard back-propagation.

2 Code

Download the file `train_nn.m` which has been used in a previous tutorial. Download also file `deepnet_lab.m` for training a Deep Network.

In line 14 of `deepnet_lab.m`: Define the structure of the network where `conf.hidNum` is a vector with the number of units in each hidden layer. For example $conf.hidNum = [20 \quad 40 \quad 30]$ defines a network with 1 input layer, 3 hidden layers and 1 output (softmax) layer, the hidden layers, from input to output (i.e. bottom-up), having 20, 40, and 30 units, respectively.

Lines 24 to 91 implement the pre-training phase. Here, for each layer (bottom-up) we create either an RBM or an Auto-Encoder (AE) and train it. Note that the top layer must be an RBM and it will be trained with softmax labels as well (as seen in class). If all layers are trained as RBMs then this model is called a Deep Belief Network.

Line 25 creates a `MOD` directory to store the intermediate features generated during the layer-wise pre-training. If the training of the model is stopped at layer i, the intermediate files are stored in `MOD` so that training can be resumed at the next layer i+1. To train the model from the begining again just delete the MOD folder which is located in the parent folder of the .m script.

Line 27 defines the method to pre-train each layer, e.g. `pre_train={'rbm','ae','rbm'}` means that it will train the input to the 1st hidden layer as an RBM, the 1st hidden layer to the 2nd hidden layer as an AE, and the 2nd hidden layer to the 3rd hidden layer (and that to the output) as an RBM (with a softmax layer).

In the case [20, 40, 30] and ["rbm, "ae", "rbm"] using the MNIST dataset, an RBM is trained from a 28x28 input to a 20-node hidden layer. The data is thus transformed onto feature vectors of size 20. Then, an autoencoder having 20 input, 40 hidden and 20 output nodes is trained. This is deterministic, i.e. does not use Gibbs sampling, and hence should be faster than training an RBM. At this point, the data is transformed onto feature vectors of size 40 (i.e. the hidden layer states). Finally, an RBM with 40 input and 30 hidden nodes is trained together with 10 extra softmax input units for the labels of the digits from 0 to 9.

Lines 28-30 define the learning rate, momentum and weight decay for every pair of layers in the pre-training (all bottom-up).

Lines 96 onwards implements the fine-tuning through standard back-propagation on the entire network, from the 28x28 input to the 10 softmax outputs.

3 Exercise

Work on the parameters of the pre-training and the fine-tuning to create different deep networks of your choice e.g. using the MNIST dataset but also other datasets provided earlier. Increase the number of epochs of the supervised fine-tuning to 1000. See if you can train a deep network with better test set performance than a shallow network trained on the same dataset or a deep network trained using backpropagation only without pre-training.