Artificial neural networks - Exercise session 4

Deep Learning: Stacked Autoencoders and Convolutional Neural Networks

Cristina Luna

May 8, 2018

1 Stacked Autoencoders

Autoencoders are an unsupervised learning technique for replying the input in the output but doing dimension reduction in the intermediate layers, so in some way they are able to identify the most representatives features of the input as PCA does, although no following the same rules.

In this project, we will analyze the performance of stacked autoencoders for handwritten digit recognition and the differences that they have with respect to MLPs, by comparing the average results obtained after 5 running for different parameter selections.

Experiment 1: In this test we investigated the results that the MLPs can obtain varying the number of layers and the number of neurons per layer but maintaining constant the maximum number of epochs in 1500. We used 1, 2, 3 and 4 layers, being the number of neurons for the first layer: 200,300,400,500,600 or 800. The number of neurons of the following layers were half of the previous layer, for example, if we had 200 neurons in the first layer, then, we had 100 in the second layer, 50 in the third one and 25 in the last layer if we were using 4 layers.

Experiment 2: For this second experiment, we tried to improve the default results obtained with stacked autoencoders by testing different number of epochs (100,400 and 800), the number of layers (in this case, test was performed just for 2 and 3 layers) and the amount of neurons per layer (starting with 200,300,400 or 500 in the first layer and halving them in the same way that we did with the MLP case).

From the experiment 1, the results obtained shows that for different number of layers, the best configuration is always the one that starts with 200 neurons except for 4 layers case, in which the best accuracy is reached for 800 neurons configuration.

The test accuracy obtained for 200 neurons case per layer was: 96,6%(1 layer); 96,68%(2 layers); 96,8%(3 layers) and 95,81 % (4 layers).

If we compare the results obtained in the experiment 2 with the previous mentioned, we come to the conclusion that we just need to use 2 layers for the stacked autoencoder for obtaining better results than a 3 layer MLP, although the training time is also higher than for MLP but thanks to the initialization of parameters that is performed in the stacked layers, when we apply tuning, the performance always increases. the effect of fine tuning is really important is when the pre-tuning accuracy is low. For the best configurations of stacked autoencoders, the post-tuning accuracy in test set was 99,7% for 200-100 neurons and 100-50-100 training epochs, and 99,72% for 500-250-125 neurons and 400-100-50-400 epochs.

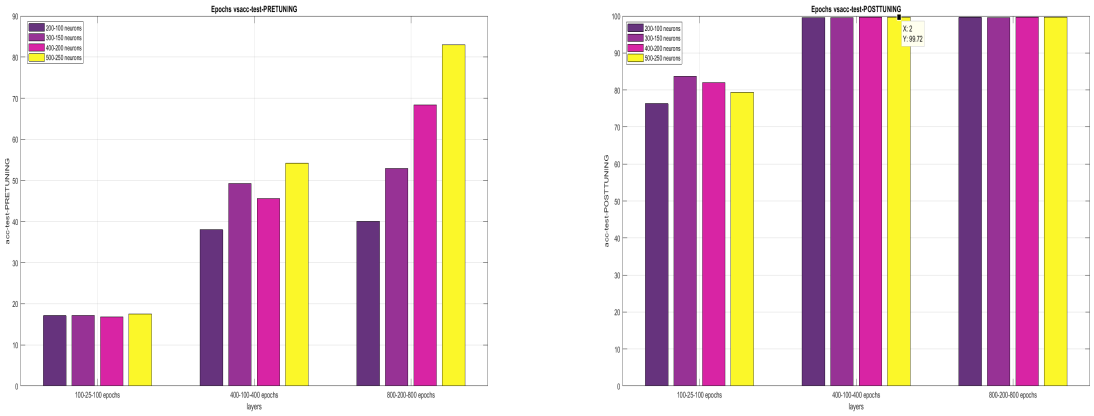

Figure 1: Comparative between pre-tunning and post-tunning accuracies for 3 layers' experiment

2 Convolutional Neural Networks

CNNs are models used in most cases for image recognition, the reason is simple, they are able to detect local spatial pattern independently of the position of them in the image.

The way in which they are able to detect some patterns is thanks to the training of their weights. These weights act as feature masks, so a group of weights conform a filter which is passed through all the image. The results of this convolution are the features maps that contain information about the matching between the input image and the 'pattern' that the filters save.

For the CNN of the exercise session, the weight dimensions of the second layer is 11x11x3x96, 11x11 is the size of the filter, 3 because of the number of channels of the image (RGB in this case) and 96 says the number of feature maps that will be generated, it means that these weights have information about 96 'different patterns or group of features' that may be found in the image.

As we can see from the previous example, CNNs generate a lot of new parameters in each convolution, which means that the complexity of the network could increase quickly if all the next layers have to use the full collection of parameters obtained. In order to address this problem, pooling layers are added, in that way we can reduce the redundant information of the features maps.

For our example, the dimension at the input of the layer 6 is 27x27x48. This value comes from three reductions applied in the layers 1 to 5 from the original image. The first reduction takes place in layer 'conv1' due to the stride size and the filters size: we obtain feature maps with the following size in each dimension (weight and height): (227(img size)-11(filter size))/4 (stride) +1 (last value for the last filter) = 55.

Another subsampling is done in the 'pool1' layer, by the max. pooling of 3x3 with stride of [2 2], which gives the following dimension: 55(feature map size) – 3(filter size))/2(stride) +1 (last value for the last filter) = 27. Thus, 27 will be the size of weight and high of the input feature maps in layer 6.

For obtaining the 3rd dimension, 96 feature maps are obtained from the value stablished by default in 'conv1' layer, therefore, as the next layers don't change the number of feature maps, this is the number of features maps that this layer receives as input.

After doing all the convolutions and crossing the whole network, the image is classified into 1 of the 1.000 final possible classes, so the number of neurons at the last layer has to be 1.000, which means that with respect to the initial dimension of the problem (227x227x3 = 154.587 parameters per image) we have reduced the dimension in a 99,3%.