Artificial Neural Networks and Deep Architectures, DD2437

Short report on lab assignment 4

Deep neural network architectures with autoencoders

Jun Zhang, Victor Castillo and Balint Kovacs

February 28, 2019

1 Main objectives and scope of the assignment

The main purpose for this lab was to get familiar with some of the key ingredients of deep neural network (DNN) architectures. The focus was mostly on stacked autoencoders. After completing this assignment, we are able to

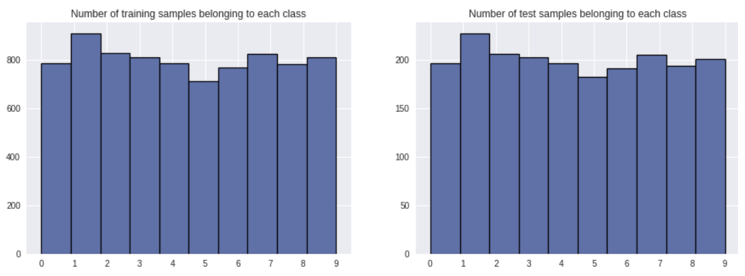
- explain key ideas underlying the learning process of autoencoders,
- apply basic algorithms for greedy training of autoencoder layers with the use of commonly available deep learning libraries,
- design multi-layer neural network architectures based on autoencoder layers for classification problems,
- study the functionality and test the performance of low-scale deep architectures developed using autoencoders.

2 Methods

For this assignment, we used TensorFlow and the Keras API, and matplotlib.

3 Results and discussion

By examining the train, and test samples, we have noticed that the classes in both the train and the test case are equally distributed, as can be seen in the histograms in Figure 1.



Figur 1: Distribution of classes.

3.1 Autoencoder for binary-type MNIST images

Considering the performance in storing the patterns, the more hidden layer we used resulted in the better performance. The difference in performance in the overcomplete case is less significant as we continue increasing the number of hidden nodes. One thing to note is that we had to use unusually high learning rates, in comparison to previous labs, for the best performing models (around 30). This can be due to the sigmoid function, which has small gradients in some regions, and the binary input.

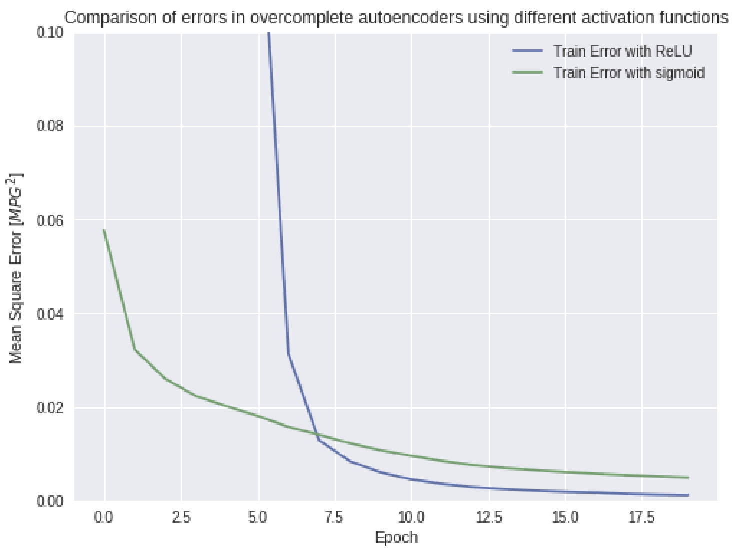
It appeared as if the network was predicting results that were too similar to particular images of the training data, or wher too blurry in some parts and for that reason, the momentum term (0.9) was used to prevent the network from getting stuck in a local minima and to continue optimizing.

When comparing the performance of the networks with ReLU and sigmoid activation functions, we have found that the ReLU slightly outperforms the sigmoid, as can be seen in Figure 2. The cause for this could be vanishing gradients in the case of sigmoid, which the ReLU doesn't suffer from, given the linearity of the function when the input is greater than zero.

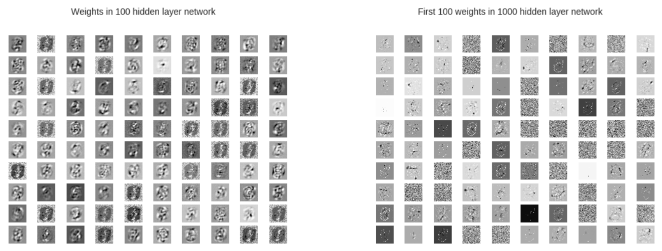
When we examined the sparsity of hidden layer representations, in the case of a 1000 hidden nodes, in average, 66% of them had non-zero values. One interesting aspect to notice was that when we used an 1500 node hidden layer and sigmoid activation, all the hidden layer values were non-zero, but with the same configuration, except using ReLU, the sparseness was 51%.

From Figure 3 it can be seen that in the case of undercomplete representation, the weights are more blurred and in the overcomplete case, they are specific on small spots.

The undercomplete network showed great capabilities in restoring noisy samples. This can be seen in Figure 4.



Figur 2: Comparing the performance of ReLU and sigmoid.

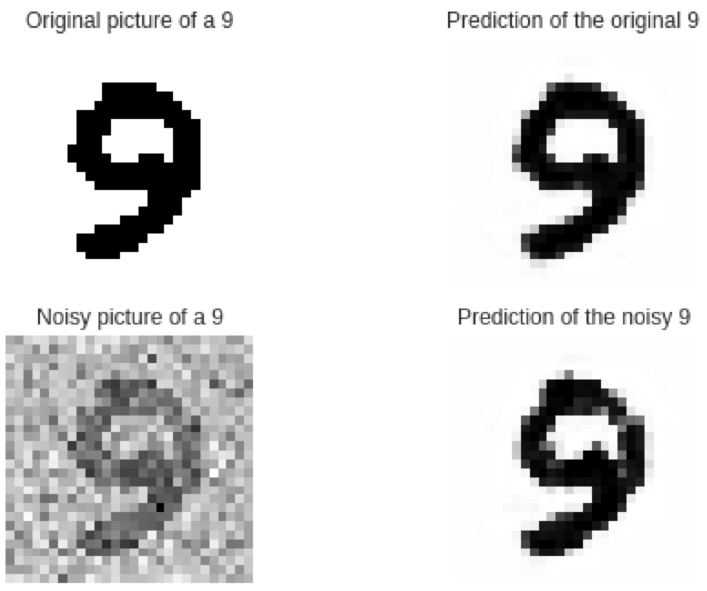

Weights in 100 hidden layer network
First 100 weights in 1000 hidden layer network

Figur 3: Weights of different architectures.

3.2 Stacked autoencoders for MNIST digit classification

In this part of the assignment, we used a layer with softmax activation function as a last layer to make class predictions. We used categorical entropy as our loss metric. In the case of the simple classifier with no hidden layers, the results were quite convincing, with around 90% accuracy. By adding hidden layers, we managed to increase this values by a couple percent. In the case of brier score, however, the 3 hidden layer model reached half of the score of the simple classifier.

Considering the size of the hidden layers, we used a 1000 nodes as the size of the first hidden layer, then added smaller layers (600 in the case of 2 hidden layer network, 800 and 500 in the case of 3 hidden layer network). We have not found much difference if we change the order of these layers.

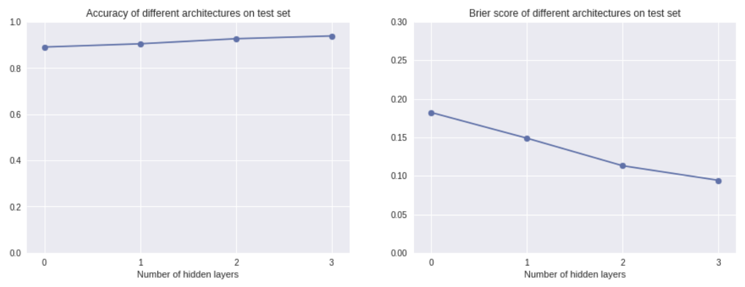


Figur 4: Noise restoring capabilities.

4 Final remarks

This exercise was a bit different than the others since we were encouraged to use existing libraries, on a typical machine learning dataset. Still, this formula can work as well, and it was a nice introduction to the world of deep learning.

The assignments in this course were really nice to understand the theory deeper and gain practical experience in the topic of neural networks.



Figur 5: Performance of models with different number of hidden layers.