INF 554: Machine Learning 1
Assignment
Neural Networks Learning for hand-written digit recognition

→ Approach for finding the best architecture with a good set of parameters.

We decided to train the neural network with 60,000 samples and to test it on 10,000 samples.

First of all, we trained and examined the basic neural network scheme with no hidden layers (this corresponds to a multi-linear classifier). This is useful to understand what are the results we can expect from our Neural Network:

[784, 10], lambda = 3.0, MaxFun = 50, Train Accuracy = 91.64, Test Accuracy = 91.64

We also reviewed state of the art results on the MNIST database and we found out that it is possible to get a 99.5% accuracy with a Neural Network (Convolutional Neural Networks have even better results with 99.8% accuracy).

Then, with lambda and MaxFun fixed, we tried to find the best architecture.
Here are the results that we got, with Lambda = 3.0 and MaxFun = 50 as initial parameters:

[784, 10], lambda = 3.0, MaxFun = 50, Train Accuracy = 91.64, Test Accuracy = 91.64
[784, 400, 10], lambda = 3.0, MaxFun = 50, Train Accuracy = 93.925, Test Accuracy = 93.87
[784, 50, 10, 10], lambda = 3.0, MaxFun = 50, Train Accuracy = 91.503, Test Accuracy = 91.47
[784, 50, 10, 10, 10], lambda = 3.0, MaxFun = 50, Train Accuracy = 81.075, Test Accuracy = 81.59
[784, 10], lambda = 3.0, MaxFun = 50, Train Accuracy = 91.771, Test Accuracy = 91.75
[784, 100, 10], lambda = 3.0, MaxFun = 50, Train Accuracy = 95.978, Test Accuracy = 95.51
[784, 113, 81, 10], lambda = 3.0, MaxFun = 50, Train Accuracy = 93.277, Test Accuracy = 93.33
[784, 113, 80, 20, 10], lambda = 3.0, MaxFun = 50, Train Accuracy = 84.073, Test Accuracy = 84.74
[784, 50, 10], lambda = 3.0, MaxFun = 50, Train Accuracy = 95.865, Test Accuracy = 95.42

Once we did that, the idea was to select the architecture that provided the best result, and to make Lambda and MaxFun vary. Technically in order to have better results we should have tried different values of lambda and Maxfun on all architectures. But the training of a network can take a lot of time to process. In a further development of the subject, it would be interesting to implement multi-processing in order to obtain more results.

We selected this architecture: [784, 100, 10]

Secondly, we examined variations of the regularization parameter, lambda. This parameter forces the Network to put some weights to zero which is something we want because it forces each neuron to focus only on a few inputs rather than all the inputs (just like human neurons). But this parameter must be chosen wisely. It must not become more important than the cross-entropy but it must be important enough to force weights to zero.

[784, 100, 10], lambda = 0.0, MaxFun = 50, Train Accuracy = 96.225, Test Accuracy = 95.74
[784, 100, 10], lambda = 0.5, MaxFun = 50, Train Accuracy = 96.485, Test Accuracy = 95.92

[784, 100, 10], lambda = 1.0, MaxFun = 50, Train Accuracy = 96.907, Test Accuracy = 96.38

[784, 100, 10], lambda = 1.5, MaxFun = 50, Train Accuracy = 96.065, Test Accuracy = 95.41

[784, 100, 10], lambda = 2.0, MaxFun = 50, Train Accuracy = 96.63, Test Accuracy = 95.78

[784, 100, 10], lambda = 2.5, MaxFun = 50, Train Accuracy = 96.593, Test Accuracy = 96.09

Again, we selected the lambda that provided the best result (best Test Accuracy).
We chose lambda = 1.0

Last of all, we examined variations of MaxFun. This parameter is the number of backwards propagations to train the model. Increasing Maxfun should always improve the training accuracy but it will not always have an impact on the test accuracy. If maxfun is too low, the network will not be trained enough but if maxfun is too high we could be overfitting our model to the training data.

[784, 100, 10], lambda = 1.0, MaxFun = 10, Train Accuracy = 77.378, Test Accuracy = 77.37

[784, 100, 10], lambda = 1.0, MaxFun = 20, Train Accuracy = 89.597, Test Accuracy = 90.12

[784, 100, 10], lambda = 1.0, MaxFun = 100, Train Accuracy = 99.36, Test Accuracy = 97.59

[784, 100, 10], lambda = 1.0, MaxFun = 500, Train Accuracy = 99.997, Test Accuracy = 98.34

Finally, the architecture that provides the best results is this one:

[784, 100, 10], lambda = 1.0, MaxFun = 500, Train Accuracy = 99.997, Test Accuracy = 98.34

To conclude we observe a great improvement of the accuracy regarding the one obtained with a multi-linear model. This improvement was paid with more processing time and a necessity to choose wisely the parameters of the model. Better results could be obtained with different activation function like the reLu function or the hyperbolic tangent.

→ Comments:

- With the 20 combinations that we examined, we never found any over-fitting, except for very high values of maxfun where we can notice some overfitting. Indeed, we can see that our model is becoming extremely accurate on the training (almost 100%) but the test accuracy doesn’t have an equal improvement. To force the model to generalize better we could introduce a dropout on the hidden layer.

- It was expected that the accuracy grows with MaxFun because MaxFun corresponds to the number of time a sample is tested in the neural network: the more it is tested, the more the network is trained, and the more it is accurate.

- Note on the random initialization: there exists a symmetry in the weights of the Neural Network. And if we initialize all weights to the same value, we are going to have the same values of gradient for each neuron of the same layer. Then we will never be able to break the symmetry and to have proper results. This is why we break the symmetry right away during the initialization.