Ezbakhe Hatim

**INF554: MACHINE LEARNING I**

**Assignment Report: Neural Networks Learning for hand-written digit recognition**

The goal of this project was to build a neural network and to apply it to the task of hand-written recognition. For that, we used the MNIST database that consists of hand-written digit images. The accuracy and effectiveness of the code depends on several parameters.

* **Number of hidden layers**

When maxfun remains at 50 iterations, but the number of hidden layers increases, the accuracy suddenly drops (unable to get better than 50%). In order to get it better, I had to keep increasing maxfun, but then the tests took too much time and I was unable to test it.

However, I did get some good accuracy values with 1 hidden layer, so that’s the parameter I used mostly.

With 4 layers and a training set of size 15 000, the accuracy is still bad (36.79). In order to improve it, maxfun should be increased which seems understandable: the NN is more sophisticated, thus needs more iterations for the gradient to get the right parameters (those who minimize the cost).

* **Training set size**

Of course, when training the model with a bigger training set, the accuracy gets better. The maximum accuracy is 94.56, obtained with a training set of size 10 000 (when the other parameters are fixed to the default parameters). The accuracy doesn’t increase when we go from 10 000 to 15 000. It is likely that some training data is actually less useful, as the model becomes ‘over-trained’.

Plus, the bigger the neural network is, the bigger the training set has to be. It is indeed because of the overfitting phenomenon. In my case, I used a training set of 2 000 instances for 1 hidden layer.

* **Number of units per layer**

I noticed a pattern while running some tests: as long as maxfun is big enough, the NN is getting more and more accurate when increasing the number of neurons per layer. There is one condition though, maxfun has to be high enough to ensure that the NN reaches its max potential.

* **Maxfun**

If maxfun is too small, the NN doesn’t reach its potential since it is not well trained, and the parameters found are getting further from those minimizing the cost function. For small NN, a good accuracy is obtained (~90%) with maxfun between 50 and 100. For bigger NN, maxfun has to be higher, which increases deeply the time of calculation.

* **Epsilon**

If epsilon is too big, maxfun has to be big too because then the learning takes more iterations (the parameters are initialized on a bigger range). Plus it can’t be equal to 0, otherwise theta\_grad and theta would be equal to 0.5.

* **Lambda**

It seems like the accuracy is a function f(lambda) with f’(lambda) decreasing, and its maximum is reached for lambda ~ 6.

The 3 parameters that took more to time to be tested were maxfun, the number of hidden layers and the training set size. This is normal since they’re the parameters that add a lot of computation when increasing.

Having tested all of these parameters, I tried to get the best accuracy possible. I was able to get 95.97 of accuracy (see parameters in the excel file).

* **Using a tanh instead of the sigmoid function**

The slope of tanh is probably not sharp enough, that’s why the accuracy values were not that interesting with this activation function. It could be better with tanh(4x).