INF 582 - Assignment 1

Olivier Moindrot

January 2016

1 Initialization

1.1 Why random initialization?

A first way to initialize the weights would be to put them all to 0. However, such a choice would mean that each gradient descent - each training of the network - would start at the same point. Therefore we cannot be able to start different initializations and choose the best. An initialization to 0 could lead to the same local minimum over and over again.

Moreover, random initialization allows to break the initial symmetry between the different weights. With initialization to 0, all hidden units have the same effect on the result, so their gradient will be the same, and they will be updated identically. Only the last weight matrix will have different updates. This is a clear waste of computation time.

1.2 Glorot initialization

Here we can now initialize uniformly in $[+\epsilon, -\epsilon]$. But how to choose ϵ wisely? To avoid saturating the sigmoid function, we have to reduce ϵ according to the number of inputs in the previous layer. For instance imagine you have first a 784 input layer with numbers in $[0, 1]$. If each weight is randomly selected in $[-1, 1]$, then the total variance of the weighted sum will be in $\mathcal{O}(n)$. To have an ouput variance of 1, we have to make sure that $\epsilon = \mathcal{O}(\frac{1}{\sqrt{n}})$.

From Xavier Glorot paper, we can have the exact initialization for the sigmoid function, where input is the number of input units and output is the number of output units:

$$\epsilon = 4 * \frac{\sqrt{6}}{\sqrt{input + output}}$$

2 Activation function

In the assignment, we use the sigmoid function as the activation function. However the activation function in the last layer should be a softmax function to obtain probabilities that sum to 1:

$$softmax(x_i) = \frac{e^{x_i}}{\sum\limits_{j=1}^{n} e^{x_j}}$$

In the other layers, a better activation function is the rectified linear unit or ReLu:

$$relu(x) = max(0, x)$$

This function has better results in images, so we will use it in further experiments. Moreover, the computation time is much cheaper, and results in a 3-fold to 4-fold increase in time.

3 Results

3.1 Results with the assignment code

Here we use the sigmoid as the activation function for every layer, and perform gradient descent with categorical cross-entropy as the objective function, with regularization λ. The gradient descent algorithm is L-BFGS.
Hyper-parameters:

- size of training set: $60,000$ (maximum possible)
- size of test set: $10,000$ (maximum possible)
- $\lambda = 3.0$
- Architecture of the network: 4 hidden layers of size 256, 128, 32 and 16.
- Number of iterations of L-BFGS: 500

Training accuracy: 99.99%
Test accuracy: 98.28%

I also saved the images not well classified in the figure_5.png file.

4 Conclusion

In the files main3.py and kerasTest.py, I have also tried using the library keras to experiment with different parameters. Here I use the ReLu activation function, and another initialization (He normal initialization). I also use dropout for regularization. I haven't tried a lot of different parameters, but I obtain easily 98.73% accuracy on the test set with these parameters:

- size of training set: $60,000$ (maximum possible)
- size of test set: $10,000$ (maximum possible)

- $dropout = 0.2$
- Architecture of the network: 4 hidden layers of size 1024, 1024, 512 and 128.
- Number of iterations of RMS: 200