Assignment Set 5- Machine Learning

Martin Michaux i6220118

Question 1: Description of the network (e.g. number of layers, number of neurons per layer, etc.)¶

Our network contains 3 layers, the input, one hidden and the ouput neurons layer. The input is composed of 400 neurones each of them representing a pixel of different gray scale. The hidden layer is composed of 25 neurones and the output layer is composed of 10 neurones, each representing a number between 0 and 9.

Question 2: Impact of specific parameters such as λ, number of iterations, weight initialization, etc.

Default value: lambda = 1, max nbr of iterations = 150, weight initialization = random in the interval [-0.12,0.12] 1)

We will train, first, the lambda value as follow: We will run different experiment by changing lambda into 1,3,5 and then compare the accuracy of the training set and the cost

Lambda = 1: Training acc = 98.86, cost = 0.35099051

Lambda = 3: Training acc = 96.88, cost = 0.58443021

Lambda = 5: Training acc = 95.62, cost = 0.73577665

So, we can conlcude that the more the lambda increases, the less is the accuracy and the more the cost is. And also that lambda does not have a big impact on the accuracy

2) We now will train the number maximation of iterations as follow: We will run different experiment by changing the number maximum of iterations into 10,50,150,200,300 and then compare the accuracy of the training set and the cost

Number maximum of iterations = 10: Training acc = 74.06, cost = 1.84560172

Number maximum of iterations = 50: Training acc = 94.82000000000001, cost = 0.52767818

Number maximum of iterations = 150: Training acc = 98.76, cost = 0.0.35099051

Number maximum of iterations = 200: Training acc = 99.28, cost = 0.33840527

Number maximum of iterations = 300: Training acc = 99.4, cost = 0.32160587

So, we can clearly conclude that the higher the max nbr of iterations is, the higher is the accuracy and the smaller is the cost

3) We now will train the weight initialization of iterations as follow: We will run different experiment by changing the weight initialization of iterations into 0.01,0.12,0.5,2 and then compare the accuracy of the training set and the cost

Weight initialization = 0.01: Training acc = 98.88, cost = 0.35670011

Weight initialization = 0.12: Training acc = 98.86, cost = 0.35099051

Weight initialization = 0.5: Training acc = 98.9, cost = 0.3484599

Weight initialization = 2: Training acc = 98.46000000000001, cost = 0.41610894

So, for this experiment there doesn't seem to have a direct corolation between the Weight initialization and the accuracy howeverit seem to have an impact on the cost morehover it seem that if we reduce Max Iterations it has a bigger impact on the accuracy and the Cost

Question 3: How does the regularization affect the training of your ANN?

With regularization, the training accuracy = 98.86 and the cost = 0.35099051 with 150 iterations
Without regularization, the training accuracy = 98.44000000000001 and the cost = 0.3959034 with 91 iterations

So, when using regularization, it is more accurate due to the fact that it avoids the overfitting. The most important fac is that the number of iterations needed without regularization is smaller, due to the fact that the training overfits its trainning set and does not need more iterations

Question 4: Did you manage to improve the initial results (using values in debugweights.mat)? Which was your best result? How did you configure the system? How could you improve them even more?

To improve the initial results, I indeed used the Theta0 and Theta1 given in debugweights.mat, and I trained the network as usual and got an accuracy of 99.456! We could improve even more (a bit) if we would increase the maximum number of iterations.

Question 5: Imagine that you want to use a similar solution to classify 50x50 pixel grayscale images containing letters (consider an alphabet with 26 letters). Which changes would you need in the current code in order to implement this classification task?

What changes from the current network is that instead of an input neurones layer of 400 (20 x 20), we have instead 2500 (50 x 50). Also, the output neurones layer would contain 26 neurones instead of 9 because those would represent the alphabetic letters. Moreover, to decrease the complexity of our algorithm, we could increase the number of hidden layers or keep one hidden layer and increase the number of neurones in it (>25).

Question 6: Change the value of the variable show_examples in ex_nn, which information is provided? Did you get the expected information? Is anything unexpected there?

The point of that block of the code is to "display" the performance of our algorithm so that it computes each image and their corresponding value estimation. If we use the parameters given in debugweights.mat, the estimations are all right, which is normal since they are the previously learned weights. However, if we change the value with the method randInitializeWeights(), the prediction will be totally random and not accurate at all. But, if we use the train network we coded, depending on how well the network has been trained, the estimations will be mostly all right.

Question 7: How does your sigmoidGradient function work? Which is the return value for different values of z? How does it work if the input is a vector and if it is a matrix?

The sigmoidGradient is simply a method that computes the gradient of the sigmoid function evaluated at z. This means that, for each layer of our network, this method converts the model's output into a probability score. It has no problem to compute the output for a vector or a matrix since it makes the computation for each element.