Machine Learning

Assignment # 3

Ahmad Tariq

15060004

# Question 1

## Data Description

We are given a dataset from the MNIST database, which comprises of handwritten digits. Each digit is stored in the form of a grey scale image. The dimensions of each picture are 28 x 28 pixels. The data is stored such that the pixel values are stored in a single row. The dataset comprises of 70,000 pictures out of which 60,000 are set apart for training a model and the other 10,000 are for testing. The train data is in the form of matrix of dimensions 785 x 60,000. Similarly the test data is of the dimensions 785 x 10,000. The labels corresponding to each input samples are given.

## Training the Neural Network

Using the backpropagation algorithm along with gradient descent as error optimization, we train the neural network. The steps are as follows:

* Weights are initialized randomly such that each weight belongs to a standard normal distribution.
* We feed forward each feature of the inputs one by one to the hidden layer by combining the weights and inputs as linear combinations. A sigmoid function is used as an activation function of the hidden layer. Similarly we feed forward the output of hidden layer combined with the weights of hidden layer to the output layer. The activation function of the output layer is the softmax function.
* After obtaining an output, we compare it with the target value and computer MSE. To minimize the error function, we use gradient descent.
* Back propagate the errors to the hidden layers to update the weights and similarly back propagating the hidden layer error to update the input layer weights.
* After updating the weights we continue back and forth in feed forward and back propagating until the maximum number of epochs is reached or the error reaches the minimum value.

## Testing

In testing we input the test data into the input layer of the trained neural network. The data is fed forward to the hidden layer with the combination of updated weights. The input is further fed forward to the output layer combined with the updated weights of hidden layer. The output is then checked as to which node has the highest value and the data is classified accordingly.

## Results

* As the dataset for training was very large, it took several minutes for the machine to actually output the trained model.
* The total MSE calculated at the end of the training the model was -9.80273111693293e-18.
* For some reasons, that are unknown to me, the 4th node of the output layer in the training phase constantly shows a value of one, which makes it impossible to test the data against the obtained model.
* I have tried debugging the code for several hours now and can’t seem to find the problem.

### D:

As the learning rate is increased, the error seems to change rapidly, can cause the algorithm not to converge to a local minima.

### 

# Question 2

## 1. Three advantages of using neural networks as auto encoder.

An auto encoder neural network is an unsupervised learning algorithm that applies backpropagation. It learns a representation for the data set in an encoded form. It is trained to copy its inputs to the outputs. Internally it has a hidden layer that describes a code to represent an input. Usually auto encoders are used for the reduction of the dimensions of the data.

There are following advantages of using neural networks as auto encoders

1. Typically, the neural network for an auto encoder consists of three layers, Input, Hidden and an Output layer. The main advantage is that the error of reconstructing data is minimized. Helps find the most efficient and compact representation of the input data.
2. Another advantage is dimensionality reduction. If you have 50 input nodes to an auto encoder with 20 hidden layer nodes, the encoder will replicate a 50 dimensional vector in the output layer and give a 20 dimensional vector that will encode your input.
3. In comparison to CNN, in some cases, the auto encoders yield biologically plausible filters and give a model based on the data instead of predefined filters. Auto encoders give us the filters that may fit the data better but there is an increase in computation time.

## 2. How CN works? Write in clear steps.

Convolutional Neural Networks are similar to the ordinary neural networks, they have layers which are made up of neurons, and they are interlinked with inputs and outputs with weights. The difference between them is that the CNNs exploit the advantage of the data being 2-Dimensional. The most familiar form of 2 Dimensional data are the images. CNN architecture comprises of one or more convolutional and sampling layers followed by some fully connected layers like an ordinary NN. CNN architecture is made by tying weights to local connections and then pooling them together resulting in translation variation feature. This results in having to calculate a lot less parameters unlike those in ordinary NNs. The architecture is explained as follows:

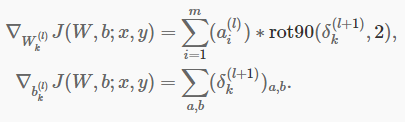
* The input to the convolutional layer is an m x m x r array representing an image. The m x n is the size of the image, whereas r denotes the number of channels in the image. For example for a colored image, r=3 for 3 channels (RGB).
* The convolutional layer has k filters of the size n x n x q, where n <=m and q can be equal to r. The filters lead to the locally connected structure, which is convolved with the image to produce k feature maps of m-n x 1.
* Each map is then subsampled with mean or max pooling over p x p contiguous regions where p can range from 2 to 5. The pooling actually down samples the feature map. Pooling layers follow a sequence of one or more convolutional layers and are intended to consolidate the features learned and expressed in the previous layers feature map. As such, pooling may be considered as a technique to compress or generalize feature representations and generally reduce the over fitting of the training data by the model.
* An additive bias is added either before or after subsampling.
* An activation function usually a sigmoid function is applied to each feature map of the image.
* The output from these layers is fed to a fully connected NN.
* Like the ordinary NN applying a back propagation algorithm, first we feed the inputs to the NN to get the output. Then work back using back calculating the errors. If the Lth layer is not a convolutional layer, the error calculations are exactly the same as for the ordinary neural networks



* If Lth layer is a convolutional layer, the error that is propagated backwards is given by:



* The up sample function has to propagate the error through the pooling layer by calculating the error w.r.t to each unit incoming to the pooling layer.
* Finally, to calculate the gradient w.r.t to the filter maps, we rely on the border handling convolution operation again and flip the error matrix δk (l) the same way we flip the filters in the convolutional layer.



* Where a (l) is the input to the L th layer, and a (1) is the input image. The operation (a (l) i) ∗δ (l+1) is the “valid” convolution between ii-th input in the L th layer and the error w.r.t. the K th filter.
* The process is repeated until the weights are optimized.

## 3. Comment on how regularized gradient descent affect model over fitting.

A training set is used to train a model to make predictions. If the model, though it fits the training data, does not provide us good and accurate predictions then we have an over fitting problem. An example of over fitting is when we use a high order polynomial to fit a regression line.

Regularization is a technique used to avoid over fitting of data. Regularization modifies the objective function that we minimize by adding additional terms that penalize large weights.

In gradient descent, regularization is provided by introducing an extra term to the already existing MSE function.



As the new term added is also a squared term, it does not affect the convexity of the existing MSE function. Now taking partial derivative to fine the local/global minima depending on the function, we get:



The interpretation is the same as the simple gradient descent, but the new term pushes the coefficients to zero with a higher pushing force for the high magnitude coefficients. This can be useful for avoiding over fitting problem. We can clearly see that from the above expression that there is no extra computational power required compared to the simple regression problem. The general idea is to trade off complexity of the model with it error on the data set.