2019-2 Machine Learning Mid Term Exam

Isack T. Nicholaus

October 2019

Question 6(a):

Mean Subtraction
I performed mean subtraction by compute the mean of each image and subtract it from each pixels of the image. In python, this can easily be done by using numpy package and call mean function on image in numpy array to get the mean value hence subtract it from the given image.

Normalization
For the minimum and maximum value of pixel picture can have is 0 and 255, then I simple divide each pixel of image by 255 to make all values between 0 and 1 inclusive.

Question 6(b):

Xavier/Glorot Initialization This technique extracts samples in the [-limit, limit] range with uniform distribution. Where limit = sqrt (6 / (fanin + fanout)) Where fanin corresponds to the number of input units of the weight tensor . fanout corresponds to the number of output units of the weight tensor.

In this experiment I applied Xavier initialization by adding it as one of the parameter to the convolution layer function as defined in tensorflow api. In tensorflow this can be done by adding $kernelinitializer =' glorotuniform'$

He Initialization

He is similar to Xavier initialization, with the factor multiplied by two.
It draws samples from a truncated normal distribution centered on 0 with

$$stddev = \left(\frac{2}{fanin}\right)^2$$

where fanin is the number of input units in the weight tensor.

In this experiment I applied He initialization by adding it as one of the parameter to the convolution layer function as defined in tensorflow api. In tensorflow this can be done by adding $kernelinitializer =' Henormal'$

Initialization	Test Accuracy
Xavier	0.9890
He	0.9903

.
Reason for the results
According to the author He initialization has slightly better result due to the weights are initialized keeping in the size of the previous layer which helps in attaining a global minimum of the cost function faster and more efficiently.The weights are still random but differ in range depending on the size of the previous layer of neurons. This provides a controlled initialization hence the faster and more efficient gradient descent.

Question 6(c):

Network Configuration:

Config. No	No. of Layers	Test Accuracy
1	3	0.9884
2	4	0.9905
3	5	0.9945
4	7	0.9956

The experiment was performed by stacking more layers on top of the input layer(which has 784 nodes for each input instance has a dimension of 28x28), each layer must at least have one node (neuron), by stacking them together allowed the output of the previous layer to be the input of the next layer until the desired output layer is reached and the output layer has the number of nodes which is equivalent to the number of classes in the dataset which was 10 in this experiment.

Reasons why results
More deeper networks work well because the deeper layers of the network tends to learn high level feature representations hence easily prediction for the network.

Question 6(d):

Optimizer	Test Accuracy
Gradient Descent	0.9890
Adagrad	0.9906
RMSprop	0.9884
Adam	0.9923

.
Reason for the results
Adam perform slightly better according to the authors because Adam is invariant to diagonal rescaling of the gradients, and is well suited for problems that are large in terms of data and/or parameters. RMSProp with momentum generates its parameter updates using a momentum on the rescaled gradient. When compared with adagrad, adam out performed adagrad due to the fact that adagrad does not consider bias correction which lead to infinitely large bias and large parameter updates.

Question 6(e):

This experiment was performed to check the effect of different activation function to the model performance, and the following functions where employed to the model with other hyper parameters remain constant but only change the activation function;

Activation Function	Test Accuracy
ReLU	0.9891
SELU	0.9902
LeakyReLU	0.9887

.
Reason for the results
SELU performed slightly better because it tends to find a activation function which fits the model well and use it.

Question 6(f):

Applied Regularization techniques:

Technique	Test Accuracy
L2	0.9887
Dropout	0.9888

.
Reason for the results
Ridge (L2) introduces only a penalty (large penalty for outliers and return larger error for those points) by changing the coefficient which can also lead to under-fitting if λ is not properly selected while dropout does not only introduces regularization but it allowing multiple versions of the networks to be tested during testing which led to robustness in both training as well as testing time.
.
NOTE:
For all experiments the baseline had the following properties;

-	Value or Number
Convolution Layers	2
Dense	2
max pooling	(2,2)
kernel size	(5,5)
kernel initializer	Xavier
activation	Lrelu
Optimizer	Gradient Descent