Assignment 3

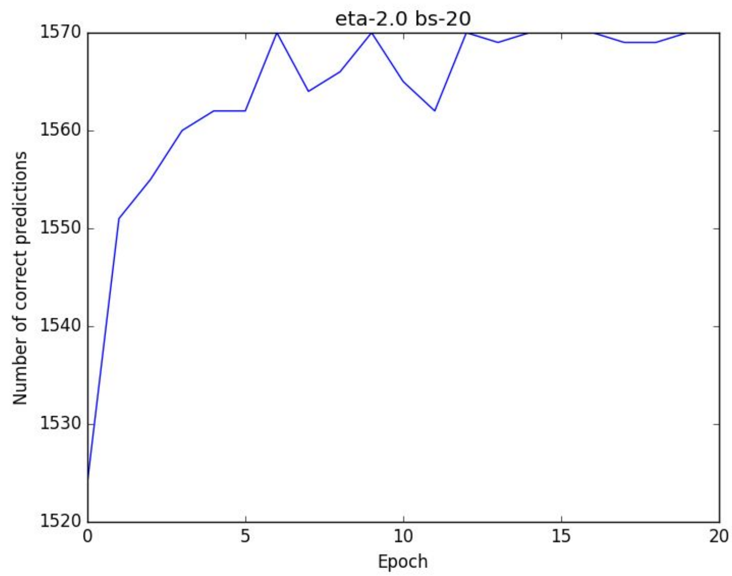
Kushagra Arora
2015049

PROGRAMMING QUESTIONS

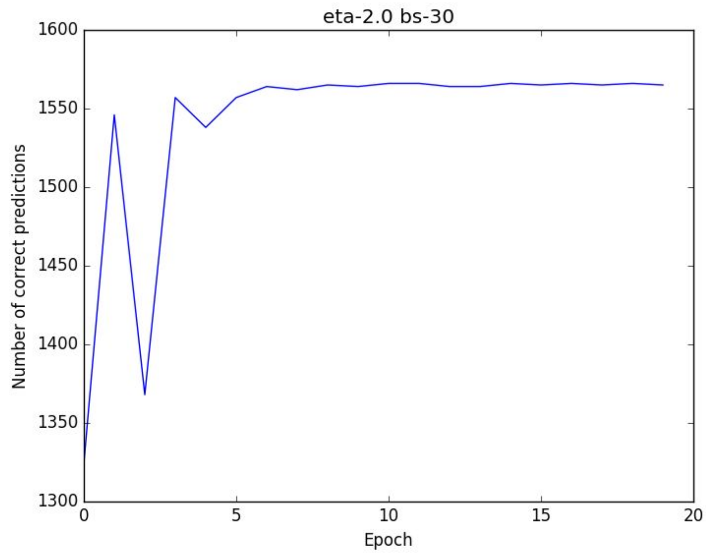
1. Implementing neural net

a. Number of data points(training + testing) : 14251
Number of classes : 2 (labels = 7, 9)
Grid Search over the following parameters:
Learning_rate = [2.0, 3.0, 5.0]
Mini_batch_size = [20, 30, 50]

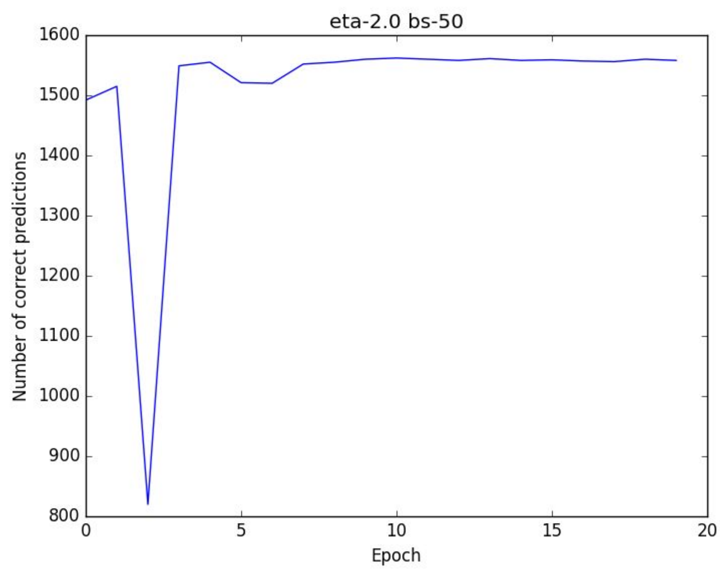
We get the following accuracy vs epoch graphs:



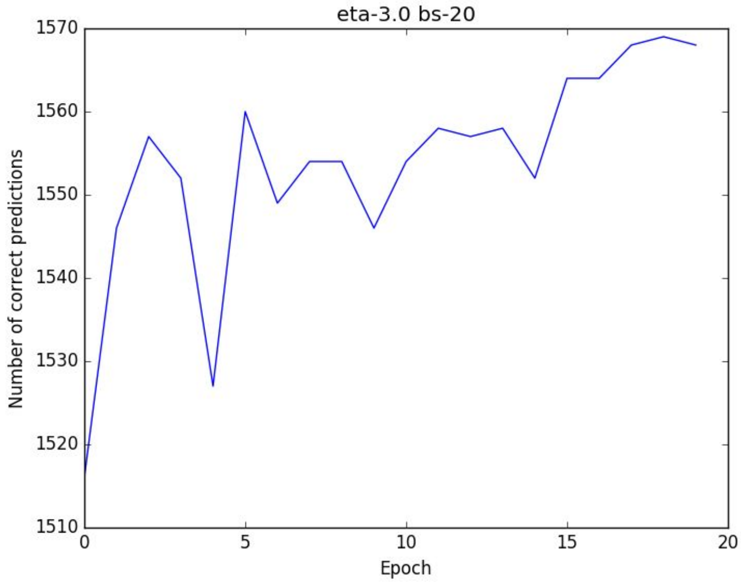
Max_accuracy = 98.61



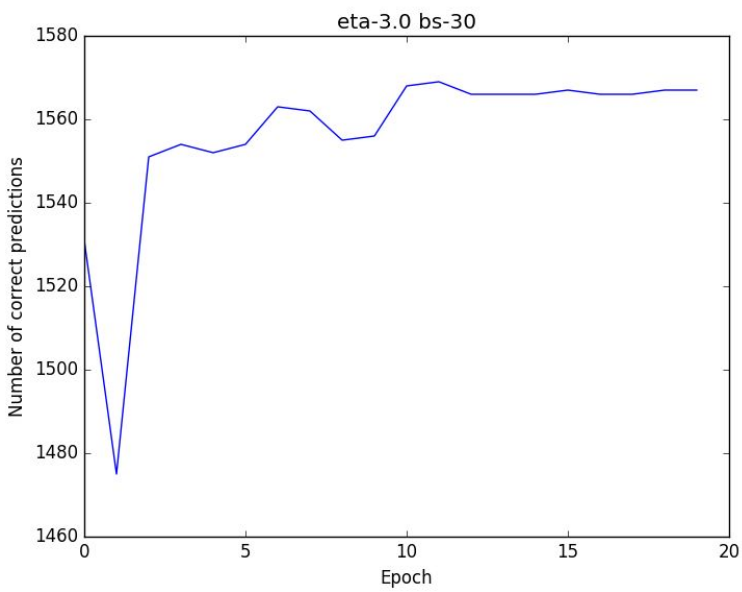
Max_accuracy = 98.36



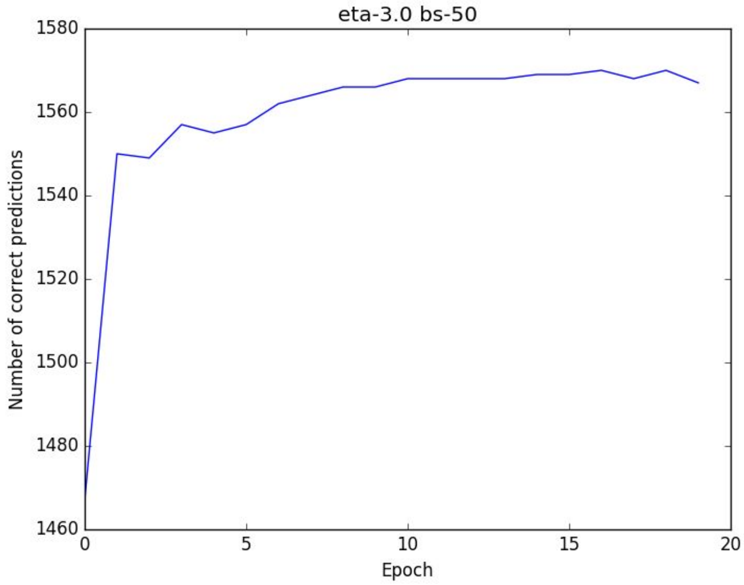
Max_accuracy = 98.11



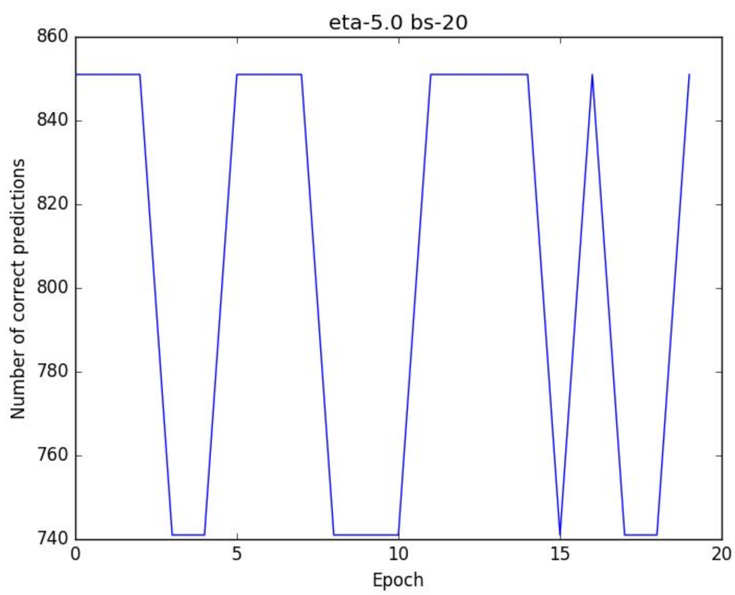
Max_accuracy = 98.55



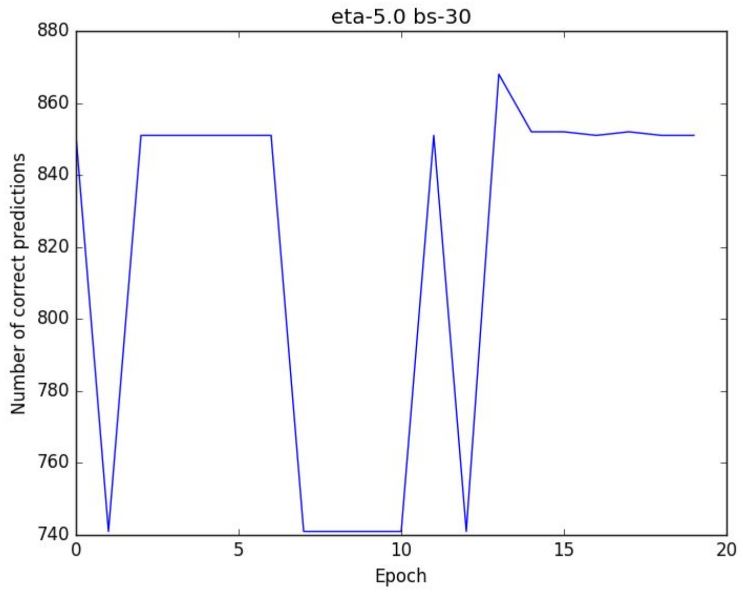
Max_accuracy = 98.55



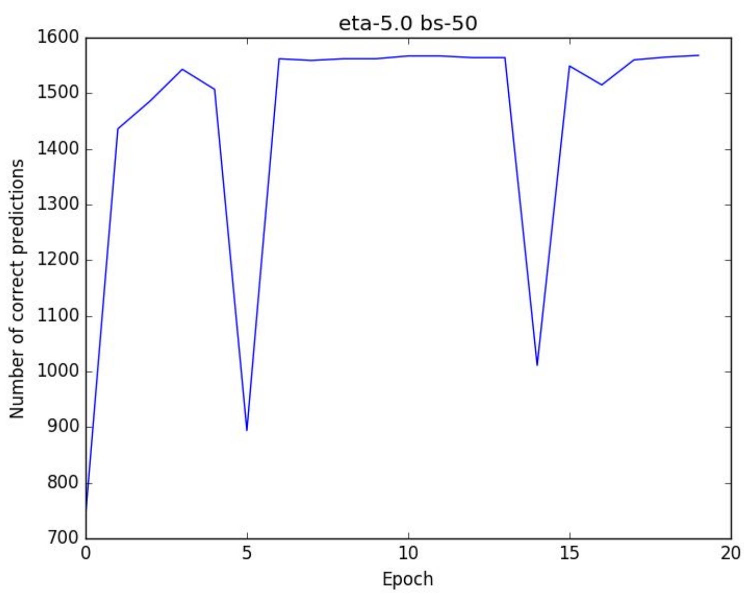
Max_accuracy = 98.61



Max_accuracy = 53.52



Max_accuracy = 54.52



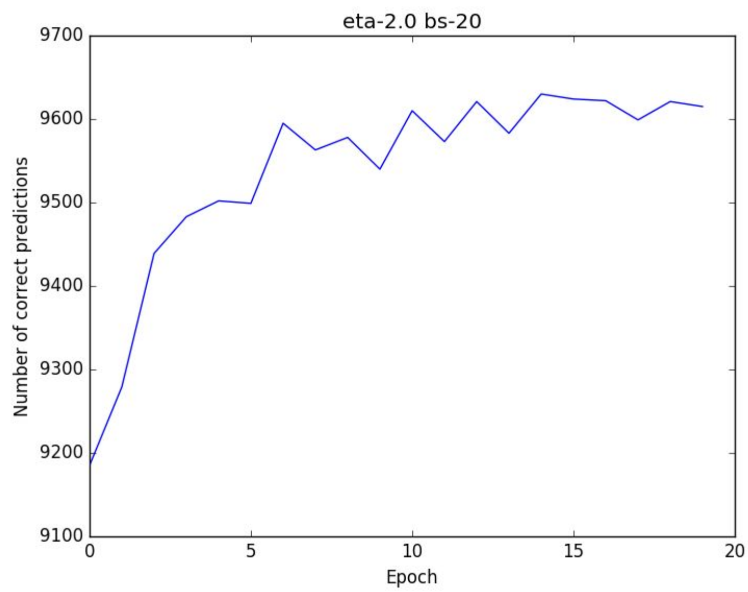
Max_accuracy = 98.49

We can see that, eta = 2.0 provides a smooth learning curve and shows maximum accuracy at batch_size = 20.

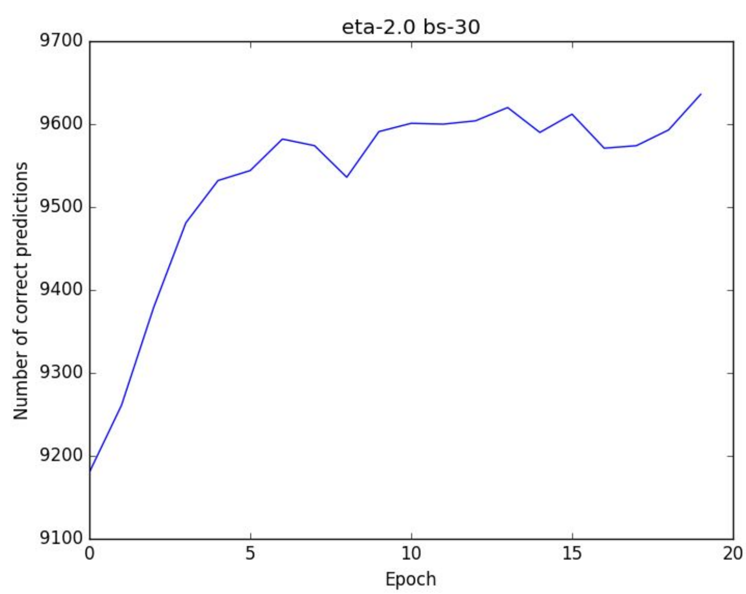
The maximum training accuracy = 98.61

This gives a testing accuracy of 98.10

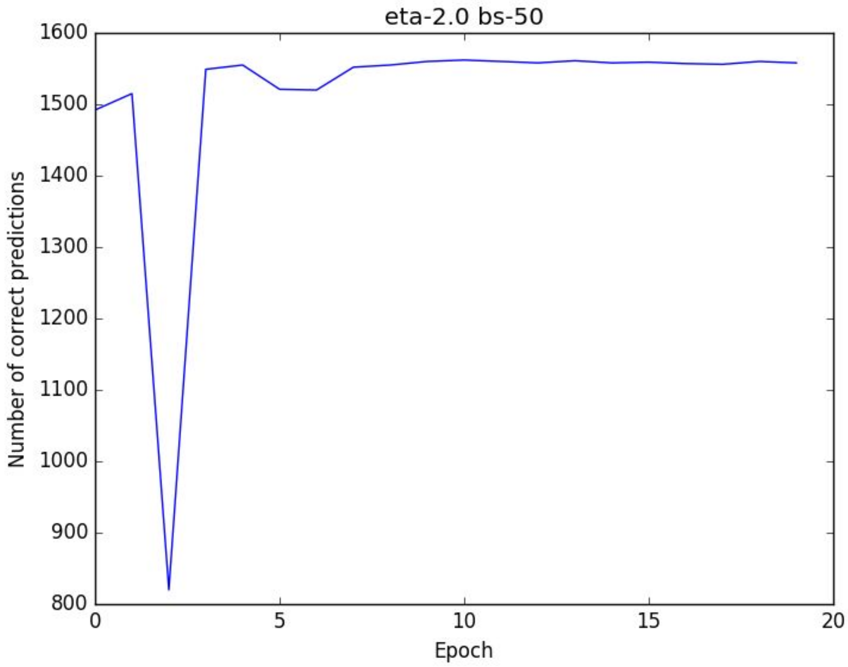
b. Number of data points(training + testing) : 60000 + 10000
Number of classes : 10 (labels = 0,1,2,3,4,5,6,7,8,9)
Grid Search over the following parameters:
Learning_rate = [2.0, 3.0, 5.0]
Mini_batch_size = [20, 30, 50]



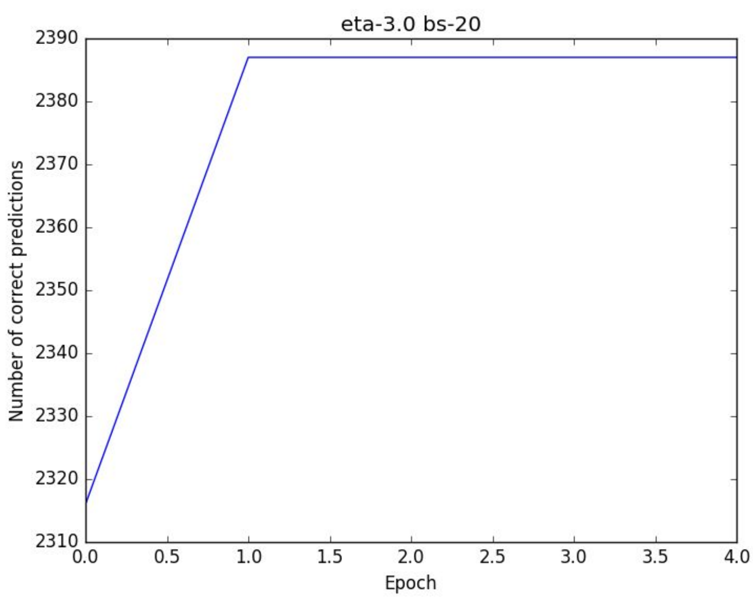
Max_accuracy = 96.28



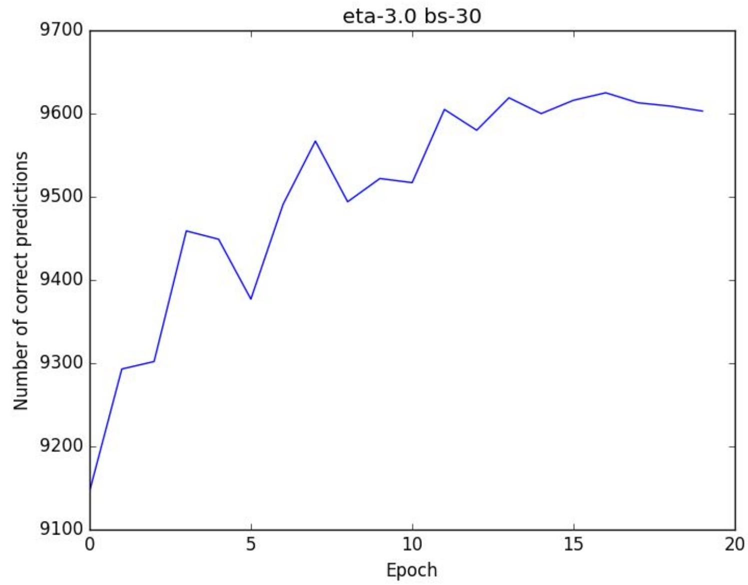
Max_accuracy = 96.34



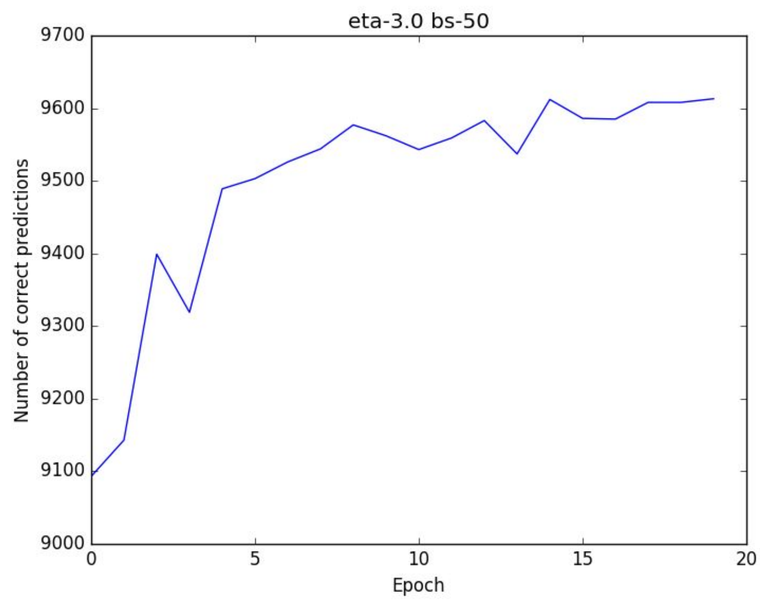
Max_accuracy = 95.55



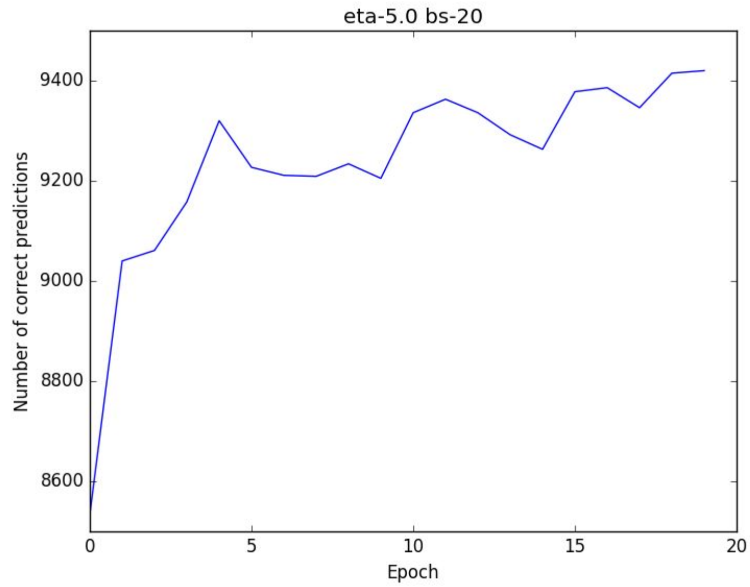
Max_accuracy = 95.55



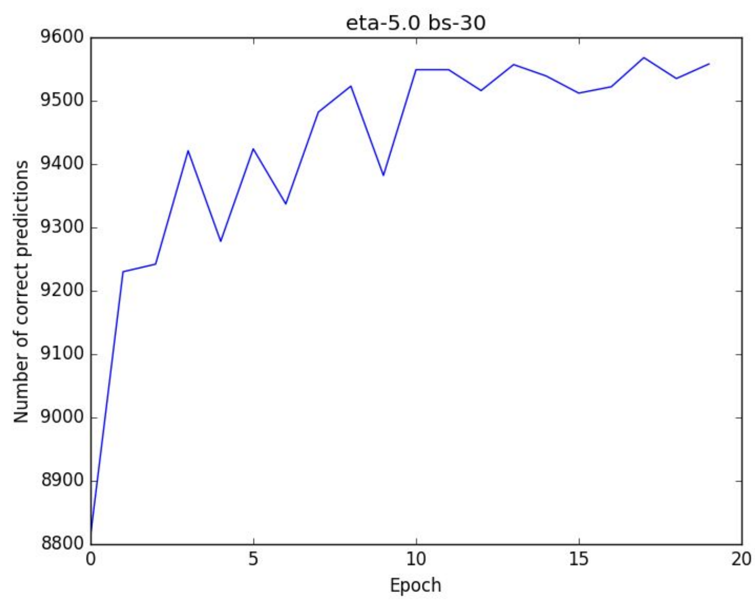
Max_accuracy = 96.23



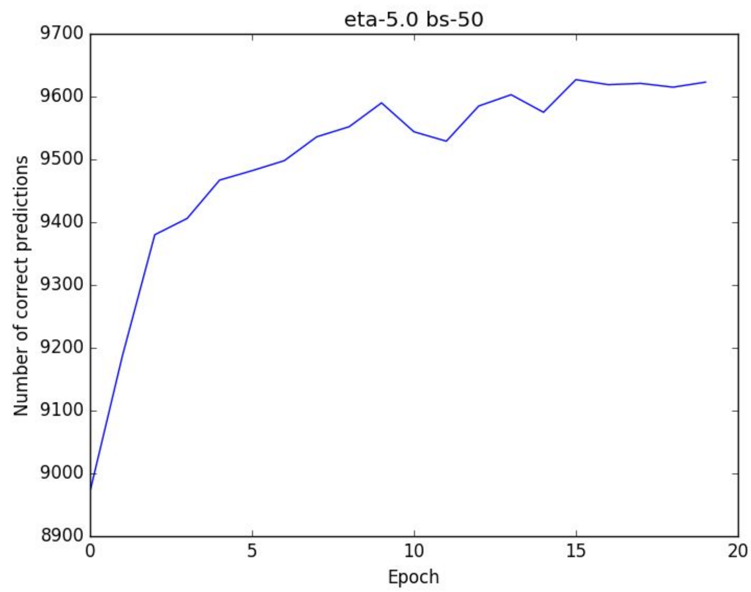
Max_accuracy = 96.11



Max_accuracy = 94.18



Max_accuracy = 95.66



Max_accuracy = 96.25

We see that again, a learning rate of 2.0 on the batch size of 30 gives us the best results.

Accuracy obtained on test_set = 96.20 %

c. Implementing ReLu and maxout activations on the above data sets.
   ReLu
   Accuracy on subset = 72.87 %
   Accuracy on bigset = 29.67 %

   I am not sure, where the problem is. It shows an overflow error even though I have tried scaling as well as normalizing the data set.

   Maxout - Not implemented

2. Comparing with sklearn's MLPClassifier

a. Using the following classifier:
   MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
   beta_2=0.999, early_stopping=False, epsilon=1e-08,
   hidden_layer_sizes=(100, 50), learning_rate='constant',
   learning_rate_init=0.1, max_iter=50, momentum=0.9,
   nesterovs_momentum=True, power_t=0.5, random_state=1, shuffle=True,
   solver='sgd', tol=0.0001, validation_fraction=0.1, verbose=10,
   warm_start=False)

We obtain the following results:
Training accuracy = 100 %
Testing accuracy = 99.10 %

The learning converges in 18 iterations

b. For the complete data set, the following results were observed:
Training accuracy = 100 %
Testing accuracy = 98.23 %

The learning converges in 31 iterations

We can say that the better accuracy is due to better handling of precision errors. And also, I have run the code for 20 epochs. Running it for more epochs for a smaller learning rate will increase the accuracy.

3. Bonus : Did not attempt

THEORY PROBLEMS

1.

1. Given model : neural net with linear activation of arbitrary depth.

Let the number of layers be n.

Since, we have linear activation

$a_1 = w_1 x + b_1$

[where w_i is the weight matrix for layer i
b_i " " bias vector " " i]

$\Rightarrow a_2 = w_2 a_1 + b_2$

$= w_2(w_1 x + b_1) + b_2$

$= w_2 w_1 x + (w_2 b_1 + b_2)$

$\Rightarrow a_2$ is of the form $wx + b$

similary $a_n = wx + b$ for some w, b

$\Rightarrow$ neural net behaves like a linear classifier.

We can see that this similar to using SVM with linear kernel.

Since XOR is not linearly separable.

Our, neural net will not be able to classify.

2.

2.

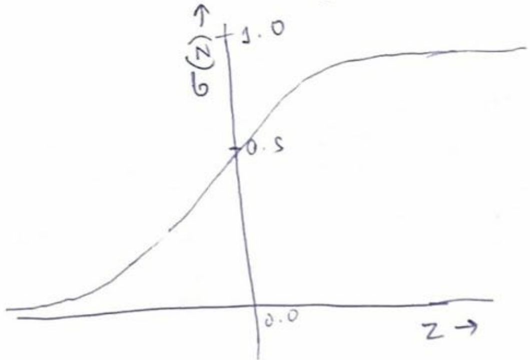
$$\sigma(z) = \frac{1}{1+e^{-z}}$$

We know that $x_i \in [0, 1000]$

$\Rightarrow \bar{z} = w\bar{x} + \bar{b}$

will have a higher magnitude since $\bar{z} \propto \bar{x}$

The graph of $\sigma(z)$ is as follows



$\therefore \ \forall z$ with high magnitude

$\sigma(z) \rightarrow 1$

$\Rightarrow \sigma'(z) = \sigma(z)(\sigma(z) - 1)$

approaches zero.

We know that $\delta^L = \nabla_a C_x \odot \sigma'(z)$

$\delta^l = (w^{l+1})^T \delta^{l-1} \odot \sigma'(z)$

Since $\sigma'(z)$ approaches zero it stagnates learning.

3.

Another problem this poses is the overflow problem for high z, e^z is very large and we might not be able to handle it.

Using ReLu function in the hidden layer
ReLu being a linear function
does not stagnate learning
Thus it definitely betters the problem.
However we see $f'(z)$ for ReLu

$$= \begin{cases} 1 & \forall z > 0 \\ 0 & \text{else} \end{cases}$$

$\Rightarrow$ ReLu will not be able to handle the big range of [0,100] since gives the same derivative for the complete range.

Pre-processing required:

To overcome sigmoid problem:

① We can scale the data to a smaller range → like [0,1]

To overcome ReLu problem:

② We can normalize the data about its mean to a range like [-1,1]

$$\Rightarrow \nabla_a C_a = \frac{a_x^L - y a_x^L - y + y a_x^L}{a_x^L (1 - a_x^L)}$$

$$= \frac{a_x^L - y}{\sigma(z)[1 - \sigma(z)]}$$

$$= \frac{a_x^L - y}{\sigma'(z)} \qquad [\because \sigma'(z) = \sigma(z)(1 - \sigma(z))]$$

$$\Rightarrow \delta^L = \nabla_a C_a \odot \sigma'(z)$$

$$= \frac{a_x^L - y}{\cancel{\sigma'(z)}} \cdot \cancel{\sigma'(z)}$$

$$= a_x^L - y$$

Hence, cross-entropy fastens the learning process