Homework 3

CSCI 5525: Machine Learning

Due on Nov 7th 11:00am (before the class)

Please type in your info:

- **Name**:Anubhav Panda
- **Student ID**:5509727
- **Email**:panda047@umn.edu
- **Collaborators, and on which problems:**

Homework Policy. (1) You are encouraged to collaborate with your classmates on homework problems, but each person must write up the final solutions individually. You need to fill in above to specify which problems were a collaborative effort and with whom. (2) Regarding online resources, you should **not**:

- Google around for solutions to homework problems,
- Ask for help on online.
- Look up things/post on sites like Quora, StackExchange, etc.

Submission. Submit a PDF using this LaTeX template for written assignment part and submit .py files for all programming part. You should upload all the files on Canvas.

Written Assignment

Instruction. For each problem, you are required to write down a full mathematical proof to establish the claim.

Problem 1. VC Dimension.

(**5 points**) Prove this claim formally: If $\mathcal{F}$ is finite, then $\text{VCD}(\mathcal{F}) \leq \log(|\mathcal{F}|)$.

Your answer. lets assume the VC dimension of $\mathcal{F}$ is c.
so that the cardinality of $\mathcal{F}$ should be greater than or equal to 2^c
mathematically $2^c <= |\mathcal{F}|$
$=log(2^c) <= log(|\mathcal{F}|)$
$=c <= log(|\mathcal{F}|)$(proved)

Problem 2. Uniform convergence.

(10 points)

In this problem, we will prove a stronger generalization error bound for the binary classification that uses more information about the distribution. Let us say that P is a distribution over (X, Y) pairs where $X \in \mathcal{X}$ and $Y \in \{+1, -1\}$. Let $\mathcal{H} \subset \mathcal{X} \to \{+1, -1\}$ be a finite hypothesis class and let ℓ denote the zero-one loss $\ell(\hat{y}, y) = 1\{\hat{y} \neq y\}$. Let $R(h) = \mathbb{E}\,\ell(h(X), Y)$ denote the risk and let $h^* = \min_{h\in\mathcal{H}} R(h)$. Given n samples, let $\hat{h}_n$ denote the empirical risk minimizer. Here, we want to prove a sample complexity bound of the form:

$$R(\hat{h}_n) - R(h^*) \leq c_1\sqrt{\frac{R(h^*)\log(|\mathcal{H}|/\delta)}{n}} + c_2\frac{\log(|\mathcal{H}|/\delta)}{n} \tag{1}$$

for constants c_1, c_2. If $R(h^*)$ is small, this can be a much better bound than the usual excess risk bound. In particular, if $R(h^*) = 0$, this bound recovers the $1/n$-rate.

a) To prove the result, we will use Bernstein's inequality, which is a sharper concentration result.

Theorem 1 *(Bernstein's inequality). Let $X_1, ..., X_n$ be i.i.d real-valued random variables with mean zero, and such that $|X_i| \leq M$ for all i. Then, for all $t > 0$*

$$\mathbb{P}[\sum_{i=1}^{n} X_i \geq t] \leq \exp\left(-\frac{t^2/2}{\sum_{i=1}^{n}\mathbb{E}[X_i^2] + Mt/3}\right) \tag{2}$$

Using this inequality, show that with probability at least $1-\delta$

$$|\bar{X}| \leq \sqrt{\left(\frac{2\,\mathbb{E}\,X_1^2\log(2/\delta)}{n}\right)} + \frac{2M\log(2/\delta)}{3n} \tag{3}$$

where $\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$ and X_i's satisfy the conditions of Bernstein's inequality.

b) Using Eq. (3) and the union bound, show Eq. (1)

Your answer. lets assume $\delta/2 = \exp\left(-\frac{t^2/2}{\sum_{i=1}^{n}\mathbb{E}[X_i^2] + Mt/3}\right)$

$=\log(\delta/2) = \left(-\frac{t^2/2}{\sum_{i=1}^{n}\mathbb{E}[X_i^2] + Mt/3}\right)$

$=\log(2/\delta) = \left(\frac{t^2/2}{\sum_{i=1}^{n}\mathbb{E}[X_i^2] + Mt/3}\right)$

$=t^2/2 - log(2/\delta)(\sum_{i=1}^{n}\mathbb{E}[X_i^2] + Mt/3) = 0$

$t = \frac{m}{3}log(2/\delta) + \sqrt{\frac{m}{3}log(2/\delta))^2 + 2\sum_{i=1}^{n}\mathbb{E}[X_i^2]log(2/\delta)}$

($\frac{(-b+\sqrt{b^2-4ac})}{2a}$solution)

as we know $\sqrt{a+b} <= sqrt(a) + sqrt(b)$ and $a >= 0, b >= 0$

$=t <= 2\frac{m}{3}log(2/\delta) + \sqrt{2\sum_{i=1}^{n}\mathbb{E}[X_i^2]log(2/\delta)}$

as we have assumed $P[\sum_{i=1}^{n} X_i > t] <= \delta/2$ and $P[\sum_{i=1}^{n} X_i < -t] <= \delta/2$

adding both it can be written as $P[|\sum_{i=1}^{n} X_i| > t] <= \delta$

so $P[|\sum_{i=1}^{n} X_i| <= t] <= 1-\delta$ and from the above equation

$|\sum_{i=1}^{n} X_i| <= t$

$= |X_i| <= t/n$

$=|X_i| <= \frac{2\frac{m}{3}log(2/\delta)+\sqrt{2\sum_{i=1}^{n}\mathbb{E}[X_i^2]log(2/\delta)}}{n}$
$=|X_i| <= 2\frac{m}{3n}log(2/\delta) + \sqrt{2\sum_{i=1}^{n}\mathbb{E}[X_i^2]log(2/\delta)/n^2}$
$=|X_i| <= 2\frac{m}{3n}log(2/\delta) + \sqrt{2\,\mathbb{E}[X_i^2]log(2/\delta)/n}$(proved)
2b as we have seen before $R(\hat{h_n}) - R(h^*) <= \sqrt{\frac{(2.R(\hat{h_n}))log(\frac{2H}{\delta})}{n}} + \sqrt{\frac{(2.R(h^*))log(\frac{2H}{\delta})}{n}} + \frac{4mlog\frac{2H}{\delta}}{3n}$
lets assume $v_n = 2log(\frac{2H}{\delta})/n$
$\sqrt{v_nR(\hat{h_n})} = \sqrt{v_n(R(\hat{h_n}) - R(h^*) + R(h^*))}$
(AS we know)$\sqrt{a+b} <= \sqrt{a} + \sqrt{b}$
$=\sqrt{v_nR(\hat{h_n})} = \sqrt{v_n(R(\hat{h_n}) - R(h^*))} + \sqrt{v_nR(h^*))}$
and as we know $\sqrt{ab} < \frac{a+b}{2} = \frac{v_n}{2} + \frac{(R(\hat{h_n})-R(h^*))}{2} + \sqrt{v_nR(h^*))}$
replacing the above equation with the first equation $R(\hat{h_n}) - R(h^*) <= \frac{v_n}{2} + \frac{(R(\hat{h_n})-R(h^*))}{2} + \sqrt{v_nR(h^*))} + \sqrt{\frac{(2.R(h^*))log(\frac{2H}{\delta})}{n}} + \frac{4mlog\frac{2H}{\delta}}{3n}$
$=\frac{R(\hat{h_n})-R(h^*)}{2} <= \frac{v_n}{2} + \sqrt{v_nR(h^*))} + \sqrt{\frac{(2.R(h^*))log(\frac{2H}{\delta})}{n}} + \frac{4mlog\frac{2H}{\delta}}{3n}$
$=\frac{R(\hat{h_n})-R(h^*)}{2} <= \frac{v_n}{2} + \sqrt{v_nR(h^*))} + \sqrt{v_nR(h^*))} + \frac{4mv_n}{3}$
$=R(\hat{h_n}) - R(h^*) <= c_1\sqrt{v_nR(h^*))} + c_2\frac{v_n}{3}$
$=R(\hat{h_n}) - R(h^*) <= c_1\sqrt{\frac{log(\frac{H}{\delta})R(h^*))}{n}} + c_2\frac{log(\frac{H}{\delta})}{n}$(proved)

Problem 3. Model selection.

(**10 points**) In problem 2, we proved the $R(h^*)$ bound. The goal in this problem is to do this simultaneously while doing structural risk minimization. Specifically, given a family of hypothesis classes $\mathcal{H}_1 \subset \mathcal{H}_2 ... \subset \mathcal{H}_L$ of sizes $N_1 \leq N_2 \leq ... \leq N_L < \infty$, a loss function bounded on [0,1] and a sample of size n, design an algorithm that guarantees

$$R(\hat{h}) \leq \min_{i\in[\text{L}]}\min_{h^*\in\mathcal{H}_i}\left\{R(h^*) + c_1\sqrt{\frac{R(h^*)\log(LN_i/\delta)}{n}} + c_2\frac{\log(LN_i/\delta)}{n}\right\}$$

for $n \geq 2$. Your algorithm may use ERM (so need not be efficient) and your constants may vary.

You may find it useful to use the following *empirical Bernstein inequality.*

Theorem 2 *Let $X_1, ..., X_n$ be iid random variables from a distribution P supported on [0,1] and define the sample variance $V_n = \frac{1}{n(n-1)}\sum_{1\leq i<j\leq n}(X_i - X_j)^2$. Then for any $\delta \in (0,1)$ with probability at least 1 - δ*

$$\mathbb{E}\,X - \frac{1}{n}\sum_{i=1}^{n}X_i \leq \sqrt{\frac{2V_n\log(2/\delta)}{n}} + \frac{7\log(2/\delta)}{3(n-1)} \tag{4}$$

Your answer. From the Bernstein inequality we get $R(h) <= \hat{R}(h)+\sqrt{\frac{4\hat{R}(h)log\frac{2LN_i}{\delta}}{n}}+7\frac{log(\frac{2LN_i}{\delta})}{3(n-1)}$ (Equation 1)

from problem 2 we get $\hat{R}(h) <= R(h) + \sqrt{\frac{2R(h)log\frac{LN_i}{\delta}}{n}} + 2\frac{log(\frac{2LN_i}{\delta})}{3(n)}$ (Equation 2)
so the srm can be formed as

$R(\hat{h}) <= \min_{i\in[\text{L}]} \hat{R}(\hat{h}) + \sqrt{\frac{4\hat{R}(h)log\frac{2LN_i}{\delta}}{n}} + 7\frac{log(\frac{2LN_i}{\delta})}{3(n-1)}$

$R(\hat{h}) <= \min_{i\in[\text{L}]} \min_{h^*\in\mathcal{H}_i} \ R(h^*) + \sqrt{\frac{2R(h^*)log\frac{LN_i}{\delta}}{n}} + 2\frac{log(\frac{2LN_i}{\delta})}{3(n)} + \sqrt{\frac{4\hat{R}(h)log\frac{2LN_i}{\delta}}{n}} + 7\frac{log(\frac{2LN_i}{\delta})}{3(n-1)}$ (equation 3) (replacing the value of equation 2)

lets assume $\frac{log\frac{2LN_i}{\delta}}{n} = y_n$

$\sqrt{\hat{R}(h)y_n} = \sqrt{(\hat{R}(h) - R(h^*) + R(h^*))y_n}$

as we have seen before$\sqrt{a+b} < \sqrt{a} + \sqrt{b}$

$\sqrt{\hat{R}(h)y_n} <= \sqrt{(\hat{R}(h) - R(h^*))y_n} + \sqrt{R(h^*)y_n}$

as we have seen before$\sqrt{ab} <= (a+b)/2$

$\sqrt{\hat{R}(h)y_n} <= \frac{(\hat{R}(h)-R(h^*))}{2} + \frac{y_n}{2} + \sqrt{R(h^*)y_n}$

replacing equation 2 here

$\sqrt{\hat{R}(h)y_n} <= \frac{\sqrt{\frac{2R(h^*)log\frac{LN_i}{\delta}}{n}} + 2\frac{log(\frac{2LN_i}{\delta})}{3(n)}}{2} + \frac{y_n}{2} + \sqrt{R(h^*)y_n}$

$\sqrt{\hat{R}(h)y_n} <= \frac{\sqrt{\frac{2R(h^*)log\frac{LN_i}{\delta}}{n}} +}{2} + \frac{y_n}{2} + \frac{log(\frac{2LN_i}{\delta})}{3(n)} + \sqrt{R(h^*)y_n}$

here i am assuming $\frac{log\frac{LN_i}{\delta}}{n}$ and $\frac{log\frac{2LN_i}{\delta}}{n}$ have really close value and replacing it with y_n

$\sqrt{\hat{R}(h)y_n} <= \frac{\sqrt{2R(h^*)y_n} +}{2} + \frac{y_n}{2} + \frac{y_n}{3} + \sqrt{R(h^*)y_n}$

replacing its value in equation 3

$R(\hat{h}) <= \min_{i\in[\text{L}]} \min_{h^*\in\mathcal{H}_i} \ R(h^*) + \sqrt{\frac{2R(h^*)log\frac{LN_i}{\delta}}{n}} + 2\frac{log(\frac{2LN_i}{\delta})}{3(n)} + \sqrt{\frac{4\hat{R}(h)log\frac{2LN_i}{\delta}}{n}} + 7\frac{log(\frac{2LN_i}{\delta})}{3(n-1)}$

$R(\hat{h}) <= \min_{i\in[\text{L}]} \min_{h^*\in\mathcal{H}_i} \ R(h^*) + \sqrt{2R(h^*)y_n} + 2\frac{y_n}{3} + 2\sqrt{\hat{R}(h)y_n} + 7\frac{y_n n}{3(n-1)}$

$R(\hat{h}) <= \min_{i\in[\text{L}]} \min_{h^*\in\mathcal{H}_i} \ R(h^*) + \sqrt{2R(h^*)y_n} + 2\frac{y_n}{3} + \sqrt{2R(h^*)y_n} + y_n + \frac{2y_n}{3} + \sqrt{R(h^*)y_n} + 7\frac{y_n n}{3(n-1)}$

$R(\hat{h}) <= \min_{i\in[\text{L}]} \min_{h^*\in\mathcal{H}_i} \ R(h^*) + c_1\sqrt{R(h^*)y_n} + c_2 y_n$

$R(\hat{h}) <= \min_{i\in[\text{L}]} \min_{h^*\in\mathcal{H}_i} \ R(h^*) + c_1\sqrt{2R(h^*)\frac{log\frac{2LN_i}{\delta}}{n}} + c_2\frac{log\frac{2LN_i}{\delta}}{n}$(proved)

Problem 4. Neural network.

(**10 points**) Consider a neural neural network with input $x \in R^{d_i,1}$ and the number of classes being d_o. The network can be given by $\hat{y} = \text{softmax}(W_2(ReLU(W_1x + b_1)) + b_2)$ where $W_1 \in R^{d_1,d_i}$, $b_1 \in R^{d_1,1}$, $W_2 \in R^{d_o,d_1}$, $b_2 \in R^{d_o,1}$ and Loss $L = \text{cross\_entropy}(y, \hat{y})$

Cross entropy loss between $y, \hat{y}$ is given by $-\sum_i^m y_i \log(\hat{y}_i)$ and softmax clamps x to [0,1] range using $\frac{e^{x_i}}{\sum_i e^{x_i}}$.

Derive expressions for partial derivatives (be precise and clear) $\frac{\partial L}{\partial W_2}, \frac{\partial L}{\partial b_2}, \frac{\partial L}{\partial W_1}, \frac{\partial L}{\partial b_1}$

Your answer. the cross entropy loss can be defined as $L = \text{cross\_entropy}(y, \hat{y}_j)$

$=-\sum_j^m y_j \log \hat{y}_j$

lets assume $\hat{y} = softmax(z_j)$ and $softmax(z_j) = \frac{e^{z_j}}{\sum_i e^{z_j}}$

so $\frac{\partial L}{\partial z_j} = -\sum_j^m y_j \frac{\partial \log \hat{y}}{\partial z_j}$

as we mentioned before $\hat{y}_j = \frac{e^{z_j}}{\sum_j e^{z_j}}$

$\log \hat{y} = z_j - \log \sum_j e^{x_j}$

$=\frac{\partial \log \hat{y_j}}{\partial z_j} = 1 - \frac{1}{\sum_j e^{z_j}} \frac{\partial \sum_j e^{x_j}}{\partial z_j}$
$=1 - \frac{e^{z_j}}{\sum_j e^{z_j}}$
$=1 - \hat{y_j}$
so $\frac{\partial L}{\partial z_j} = -\sum_i^m y_j(1 - \hat{y_j})$
$= \hat{y_j} - y_j$ since $\sum_i^m y = 1$
as weknow $z_j = \sum_i W_{2i} O_i + b_2$
$\frac{\partial z_j}{\partial W_2} = O_i$
and $\frac{\partial L}{\partial W_2} = \frac{\partial L}{\partial z_j} * \frac{\partial z_j}{\partial W_2}$
$= O_i(\hat{y}_j - y_j)$
and $\frac{\partial z_j}{\partial b_2} = 1$
$\frac{\partial L}{\partial b_2} = \frac{\partial L}{\partial z_j} * \frac{\partial z_j}{\partial b_2}$
$=(\hat{y}_j - y_j)$
as we know $O_i = RELU(\sum_i W_{1_i} x_i + b_1)$ and its derivative will go to zero if $\sum_i W_{1_i} x_i + b_1 < 0$
And $\frac{\partial O_i}{\partial W_{1_i}}$
$=x_i$
$\frac{\partial z_j}{\partial O_i} = W_{2i}$
$\frac{\partial L}{\partial W_{1_i}} = \frac{\partial L}{\partial z_j} * \frac{\partial z_j}{\partial O_i} * \frac{\partial O_i}{\partial W_{1i}}$
$=(\hat{y}_j - y_j) W_{2i} x_i$ if $W_{1_i} x_i + b_1 > 0$ else 0
And $\frac{\partial O_i}{\partial b_1}$
$=1$
$\frac{\partial L}{\partial b_1} = \frac{\partial L}{\partial z_j} * \frac{\partial z_j}{\partial O_i} * \frac{\partial O_i}{\partial b_1}$
$=(\hat{y}_j - y_j) W_{2i}$ if $W_{1_i} x_i + b_1 > 0$ else 0

Programming Assignment

Instruction. For each problem, you are required to report descriptions and results in the PDF and submit code as python file (.py) (as per the question).

- **Python** version: Python 3.
- Please follow PEP 8 style of writing your Python code for better readability in case you are wondering how to name functions & variables, put comments and indent your code
- **Packages allowed**: pytorch, numpy, pandas, matplotlib
- **Submission**: Submit report, description and explanation of results in the main PDF and code in .py files.
- Please PROPERLY COMMENT your code in order to have utmost readability

Programming Common

This programming assignment focuses on implementing neural network for handwritten digits. This problem is about multi class classification and you will use MNIST dataset. You already have an idea of the dataset by now.

You will use pytorch to implement neural network. The implementation has to be for the CPU version only - No GPU's or MPI parallel programming is required for this assignment. Follow the installation instructions at https://pytorch.org in case you want to use your local machine, we recommend using conda environment.

Here is the pytorch tutorial. If you are new to pytorch, we recommend you to go through the tutorial here. https://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html

Problem 5. Neural Network Implementation.

(**25 Points**)

1. (**No Points**) You can directly download MNIST in pytorch using torchvision.datasets.MNIST. It will allow you to

```
 trainset = torchvision.datasets.MNIST(root='./data', train=True, download=True,
transform=transform)
testset = torchvision.datasets.MNIST(root='./data', train=False, download=True,
transform=transform)
```

2. (**8 Points**) First, implement a multi-layer fully connected network:
   - Input: 1-channel input, size 28x28
   - Keep batch size as 32.
   - Fully connected layer 1: Input with bias; output - 128
   - ReLu Layer
   - Fully connected layer 2: input - 128; output - 10
   - Softmax layer
   - Use cross entropy as loss function

- Use SGD as optimizer.

Train using mini-batches of the given batch size. Plot loss and training accuracy for every epoch. At the end of the training, save your model with the name "mnist-fc". Load the saved model and report testing accuracy on the reloaded model. We should be able to reload your trained model and test.

epoch: An epoch tells you the number of times the learning algorithm sees the whole training data set. When it has seen all the training samples, you say 1 epoch is over and you start iterating in the next epoch.
ANS::The test accuracy of the model is 94 percent The training loss plot vs epoch is shown as below figure The training accuracy vs epoch figure is shown as the below figure2

Figure 1: loss vs epoch

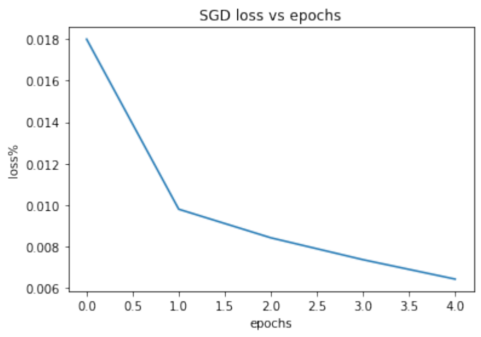
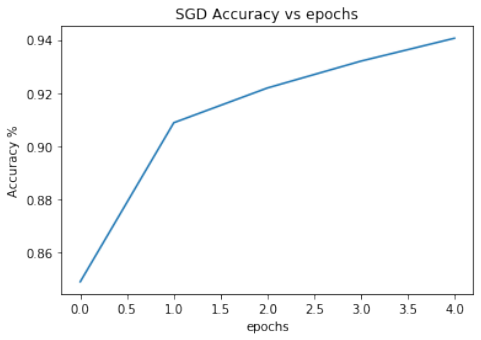


Figure 2: Accuracy vs epoch

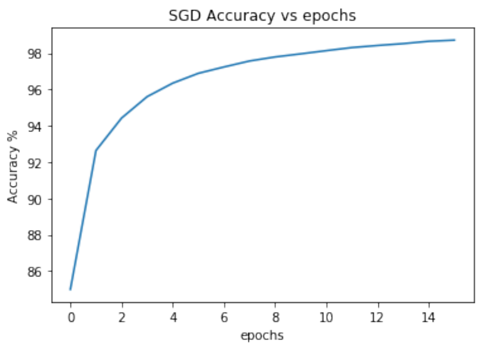


3. (**10 Points**) Implement a convolutional neural network with the following specifications.
   - Input: 1-channel input, size 28x28
   - Keep batch size as 32.
   - Convolution layer: Convolution kernel size is (3, 3) with stride as 1. Input channels - 1; Output channels - 20
   - Max-pool: 2x2 max pool
   - ReLu Layer
   - Flatten input for feed to fully connected layers
   - Fully connected layer 1: flattened input with bias; output - 128

- ReLu Layer
- Fully connected layer 2: input - 128; output - 10
- Use cross entropy as loss function
- Use SGD as optimizer.

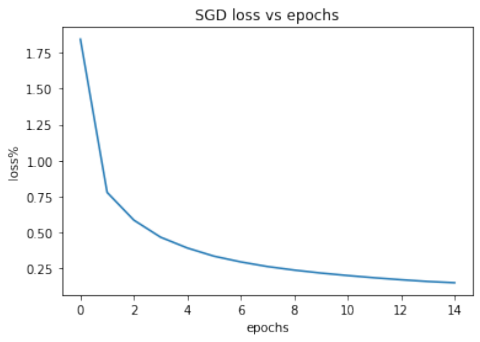
Train using mini-batches of the given batch size. Plot loss and training accuracy for every epoch. At the end of the training, save your model with the name ”mnist-cnn”. Load the saved model and report testing accuracy on the reloaded model. We should be able to reload your trained model and test.
ANS:The testing accuracy of the model is 98*percent* the training accuracy vs epochs figure is shown below

Figure 3: Accuracy vs epoch



the training loss vs epochs plot is shown below

Figure 4: loss vs epoch



4. (**4 Points**) For the convolutional network implemented above, vary the batch sizes as [32, 64, 96, 128] and plot the convergence run time vs batch sizes.
ANS:: time vs batchsize plot figure 5.As shown in the figure with increase in batch size the computation time decreases in the model having learning rate 0.01

5. (**3 Points**) ”torch optim” provides many optimizers. Try the following optimizers in your last implementation - ”SGD”, ”ADAM”, ”ADAGRAD”. (SGD is already covered in the class, for ADAM and ADAGRAD refer to https://arxiv.org/abs/1412.6980and http://www.jmlr.org/papers/volume12/duchi11a/duchi11a.pdf respectively). Plot loss vs

Figure 5: time vs batchsize plot

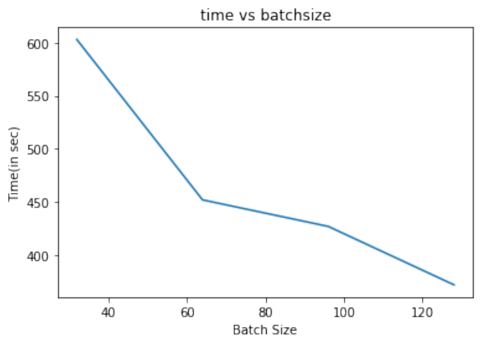
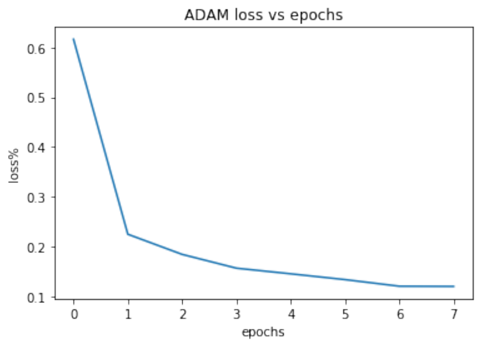
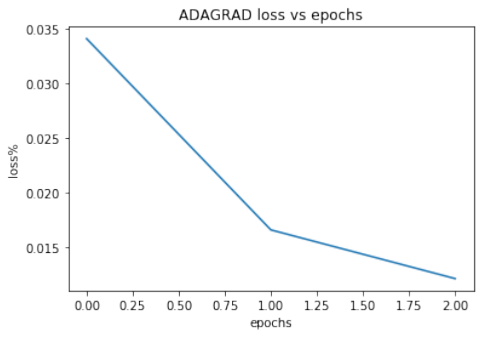


Figure 6: Adam optimizer loss vs epochs plot



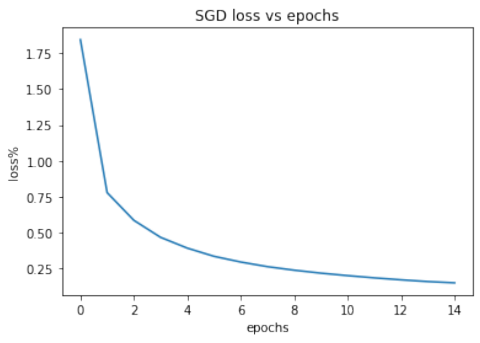
epochs for each optimizer. Briefly describe. Adam optimizer loss vs epochs plot Adagrad optimizer loss vs epochs plot SGD optimizer loss vs epochs plot

Figure 7: Adagrad optimizer loss vs epochs plot



Submission: Submit all plots requested and explanation in latex PDF. Submit your program in a file named (hw3_mnistfc.py and hw3_mnistcnn.py).

Figure 8: SGD optimizer loss vs epochs plot



As we can see in the figures in adagrad optimizer it converges within less epochs while in sgd it takes more number of epochs compared to adam.in each of the operations the batch size is remained constant as 32 and the learning rate is 0.01