CIS519: Applied Machine Learning **Fall 2018**

Homework 3

Handed Out: November 8^{th}, 2018 *Due: November* 19^{th}, 2018, *11:59 PM*

1 [Neural Networks - 50 points]

In this problem, you will implement both Feed-forward Neural Network and Convolutional Neural Network(CNN) on the CIFAR-10 image dataset. The goal of this problem is to help you understand how machine learning algorithms could apply to image classification task. In doing it you will better understand neural learning architectures (in particular, how the hyperparameters could affect model performance) and gain some hands-on experience with the deep learning framework Pytorch.

Dataset:

In this homework, you will only use a subset of the CIFAR-10 dataset to train a model and evaluate it. Overall, you will use 10,000 training images, and 2,000 testing images. Each image has a size of 32x32x3 in the RGB scale. The detailed dataset description can be found here: `https://www.cs.toronto.edu/~kriz/cifar.html`

Introduction to Pytorch:

We will use Pytorch, one of the most popular deep learning frameworks nowadays. In this part, you will need to read and understand our Pytorch tutorial before starting to use it. After fully understanding the tutorial code, you should be able to implement the simple feed-forward networks and convolutional neural networks using Pytorch. A useful suggestion is to plot the loss versus training epoch to check if the loss decreases during training. You can use the pytorch tutorial as your reference, and create a new python file to implement the following tasks.

Default Parameters:

As your default parameters, you should use:

1. SGD as the optimizer
2. 0.001 as the learning rate
3. 0.9 as the momentum
4. 64 as the batch size
5. 100 as the maximum number of epochs
6. Cross-entropy loss as the loss function

Experiment 1: Baseline Feed-Forward Neural Network [10 points]
In this part, you will implement a one-hidden layer neural network, and the architecture is shown in the table below. The original image size is 32x32x3, so you need to reshape the input image as a 3072x1 vector and feed into the network. The hidden layer has 1500 neurons followed by sigmoid activation function. The last layer outputs 10 probabilities for each class. Plot the training accuracy over epoch and report the test accuracy.

Layer	Hyperparameters
Fully Connected1	Out channels = 1500. Then use Sigmoid
Fully Connected2	Out channels = 10. Then use Sigmoid

Note you should complete experiment 1 by programming the class FeedForwardNN in the skeleton python file provided.

Experiment 2: Baseline Convolutional Neural Network [15 points]
The CNN architecture we would like you to use is shown in the table below. Plot the training accuracy over epoch and report the final test accuracy.

Layer	Hyperparameters
Convolution1	Kernel=(3x3), Out channels=7, stride=1, padding=0. Then use ReLU
Pool1	MaxPool operation. Kernel size=(2x2)
Convolution2	Kernel=(3x3), Out channels=16, stride=1, padding=0. Then use ReLU
Fully Connected1	Out channels=130. Then use ReLU
Fully Connected2	Out channels=72. Then use ReLU
Fully Connected3	Out channels=10. Then use Sigmoid

Note you should complete experiment 2 by programming the class ConvolutionalNN in the skeleton python file provided.

Experiment 3: Image Preprocessing [5 points]
In this part, you will explore how image pre-processing and data augmentation can play an important role in the performance of a convolutional neural network. First, instead of using the raw images, you should normalize images before training. Specifically, do the following:

Take each image and normalize pixel values in each of the RGB channel by subtracting its mean and dividing by the standard deviation. For example, if you are normalizing the red channel for an image, then for each of the red pixel values RP_i, you should compute:

$$\frac{RP_i - mean(RP_i)}{std(RP_i)}$$

Similarly, follow this guideline to normalize the blue and green channel of each image.

Note you should complete experiment 3 by programming the function normalize_image in the skeleton python file provided. You will then have to train your baseline convolutional neural network fron Experiment 2 using these normalized images.

Additional Optional Experiments [A. Extra Credit 5 points]: Data augmentation is also a useful technique used to improve the classification accuracy during training. Common data augmentation includes: randomly flipping the image horizontally and/or vertically, randomly cropping the images, and so on. You are encouraged to check and experiment with the Pytorch tutorial link below to explore the different ways you can preprocess the data. (`http://pytorch.org/docs/master/torchvision/transforms.html`). If you choose to do this extra credit, please include all of your work in the final pdf report and clearly label it as **A. Extra Credit**. You should include a mixture of discussions as well as graph plots of your model performance over test accuracy, training accuracy and loss following the data augmentation choices that you have made to the images in the report.

Experiment 4: Hyper-parameterization [10 points]
Hyper-parameter tuning is a very important procedure when training a neural network. In this part, you will change different hyper-parameters for both your baseline feed forward neural network from Experiment 1 and baseline convolutional neural network from Experiment 2 with some suggested values, such as

1. batch size [64, 128, 256],
2. learning rate [0.005, 0.003, 0.001],
3. convolutional kernel size [3x3, 5x5, 7x7],
4. number of neurons in the fully connected layers [your choices], and
5. number of fully connected layers [your choices].

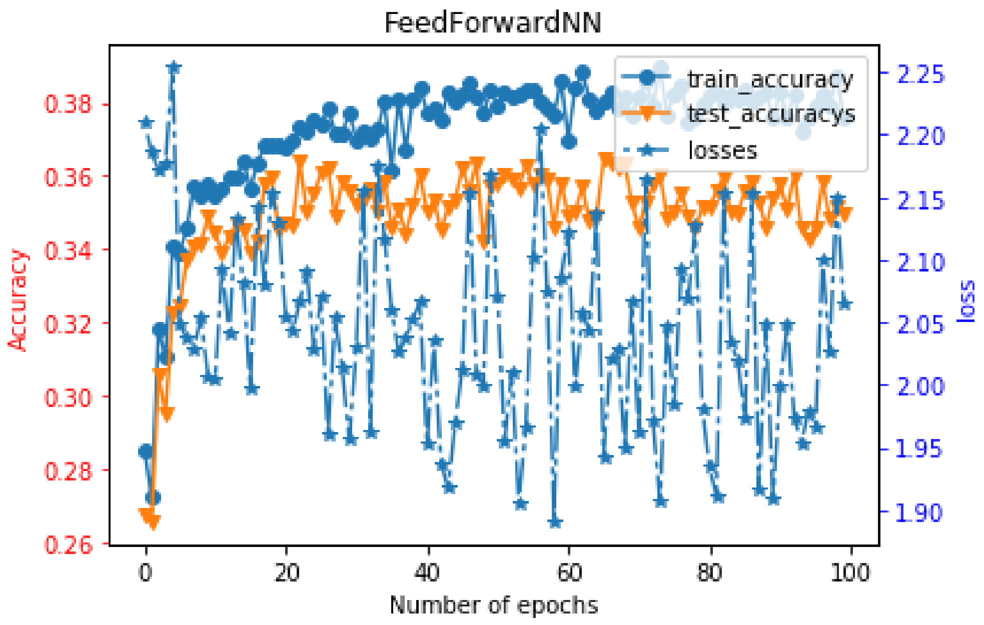
You do not need to try out all possible parameter combinations. The only requirement is that you alter the number of neurons in the fully connected layers or the number of fully connected layers to achieve a better test accuracy than the default settings. Implement the new hyper-parameterized neural network architecture in the HyperParamsFeedForwardNN and HyperParamsConvNN class as seen in the skeleton python file provided. Feel free to also change the max training iterations to be as large as you want in order to improve the model accuracy.

Additional Optional Experiments [B. Extra Credit 5 points]: You can also try different activation functions and different optimizations, such as Adagrad, SGD and Adam. You are also encouraged to try more advanced CNN architectures, such as AlexNet, VGG, and ResNet. Again, if you choose to do this extra credit, please include all of your work in the final pdf report and clearly label it as **B. Extra Credit**. You should include a mixture of discussions as well as graph plots of your model performance over test accuracy, training accuracy and loss following your additional experiments in the report.

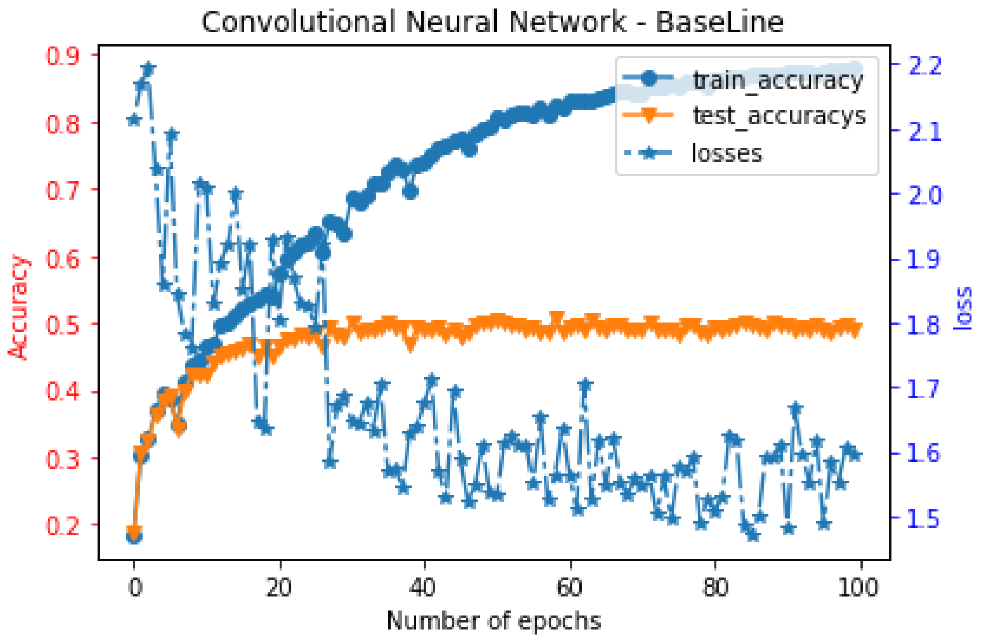
What to Report: [10 points]

1. **Experiment 1**: Plot the test accuracy, training accuracy and loss of your feed forward neural network over 100 epochs in your report.
2. **Experiment 2**: Plot the test accuracy, training accuracy and loss of your baseline convolutional neural network over 100 epochs in your report.
3. **Experiment 3**: Plot the test accuracy, training accuracy and loss of your baseline convolutional neural network over 100 epochs with the raw images and normalized images on the same graph in your report. Describe the graph by comparing and contrasting their performance over test accuracy, training accuracy and loss over 100 epochs.
4. **Experiment 4**: Plot the test accuracy, training accuracy and loss of your hyperparameterized feed forward neural network and hyperparameterized convolutional neural network over 100 epochs in your report. Provide a table that summarizes the different hyperparameters that you have chosen for both the feed forward neural network and convolutional neural network.
5. **Discussion**: Discuss how the model performance changes from experiment 1 to experiment 4. Do you see any improvements? If so, briefly discuss why? If not, briefly discuss why not? Discuss any other interesting patterns that you observe from the model performance.

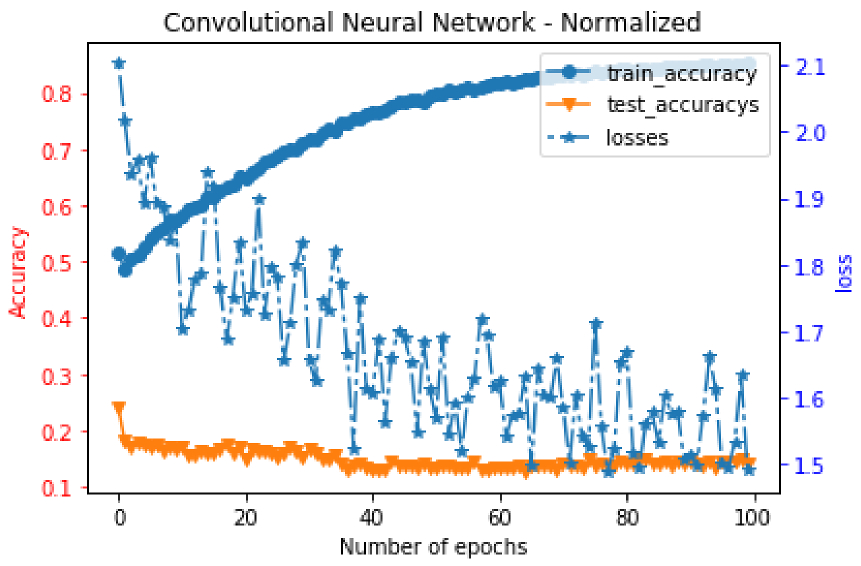
1. Experiment 1:

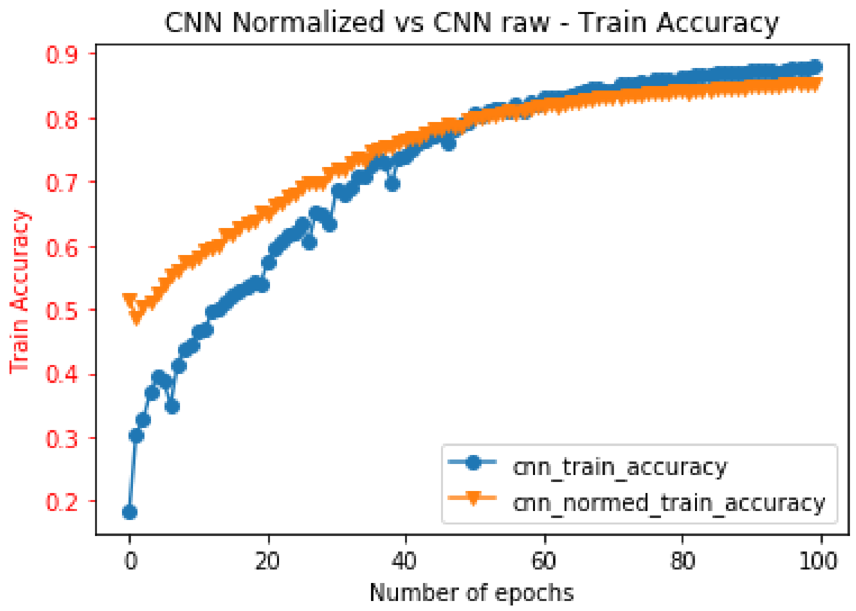


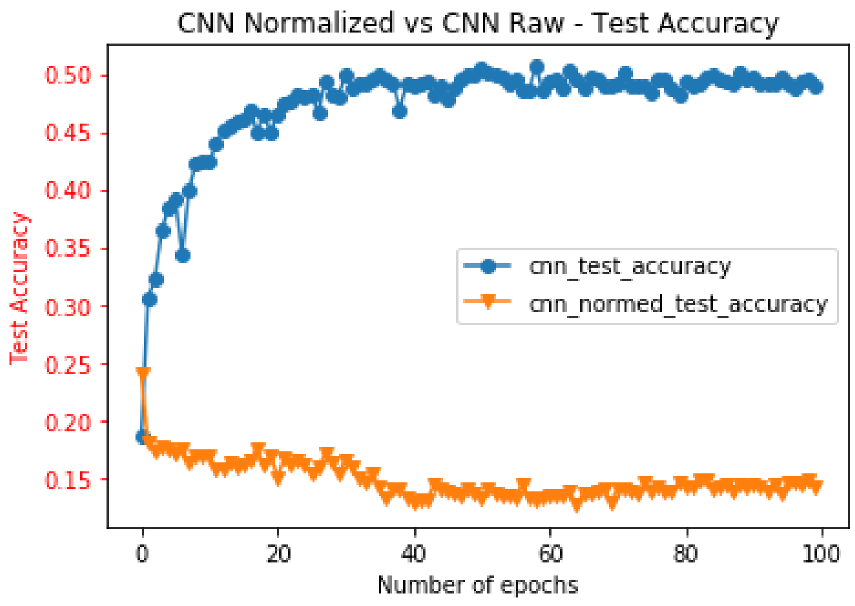
2. Experiment 2:

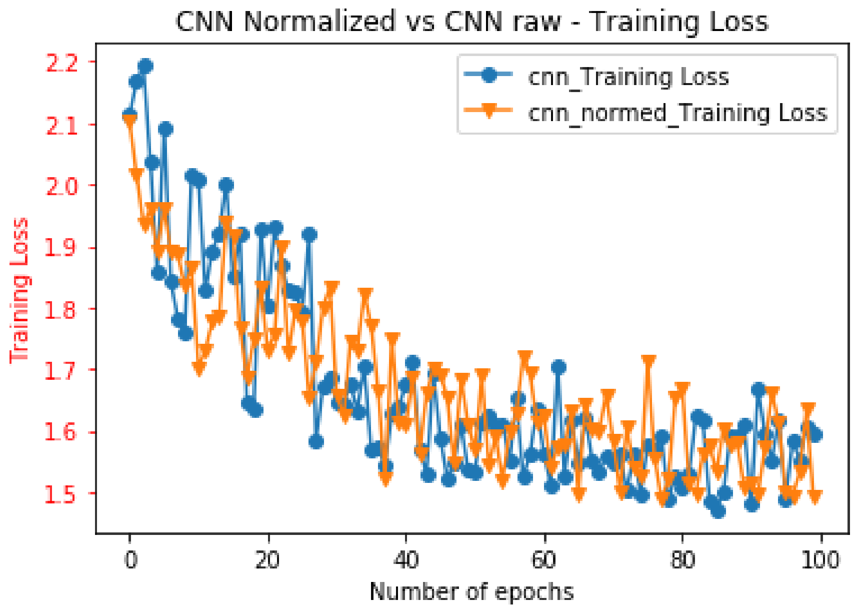


3. Experiment 3:



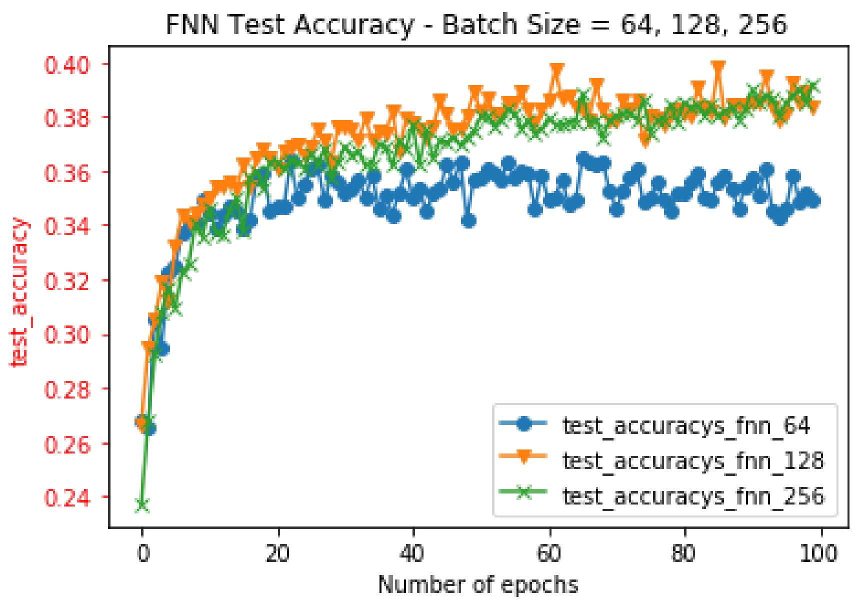


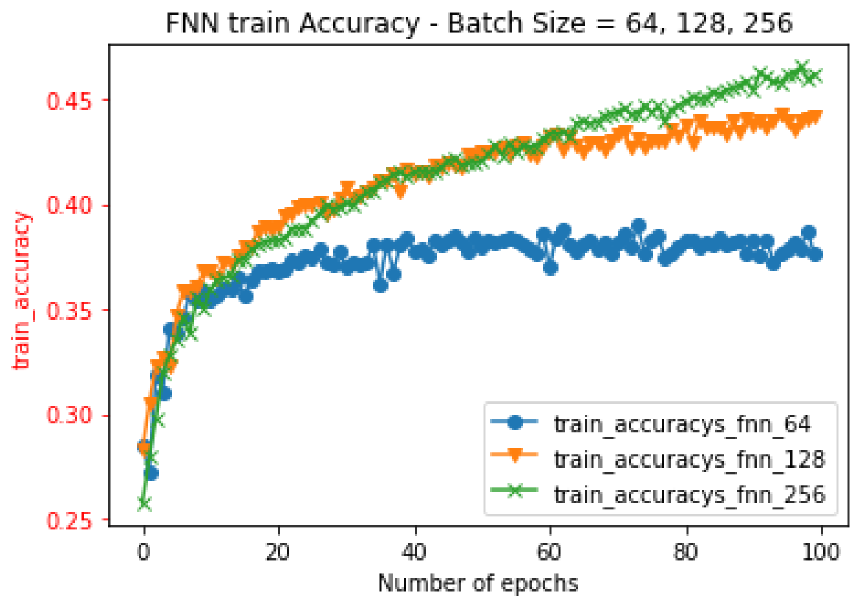


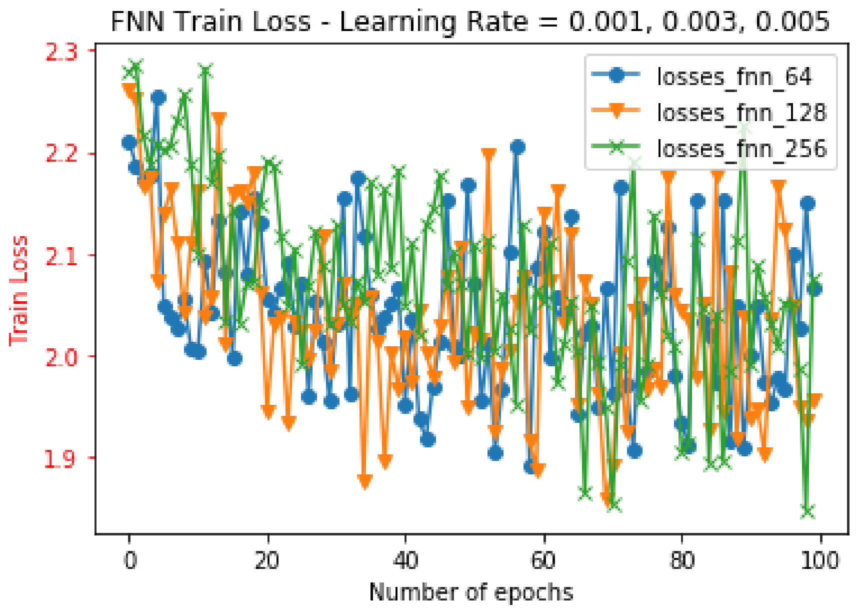


CNN Normalized vs CNN Raw:
For Train Accuracy for CNN, the Normalized version is better than raw version, however when epoch is less than 40, however, in epoch 40 to 100, the raw version is a slight better even the difference is subtle.
For Test Accuracy for CNN, the raw version is much better than the raw version, the test accuracy for normalized version is just a little bit higher than random guess, and it did not grow when the training sample grows. we can infer that the classifier may over-fit.

4. Experiment 4:
   Group1 : FNNs (Compare different batch sizes)
   Batch Size = 64, 128, 256; other parameters are as default.

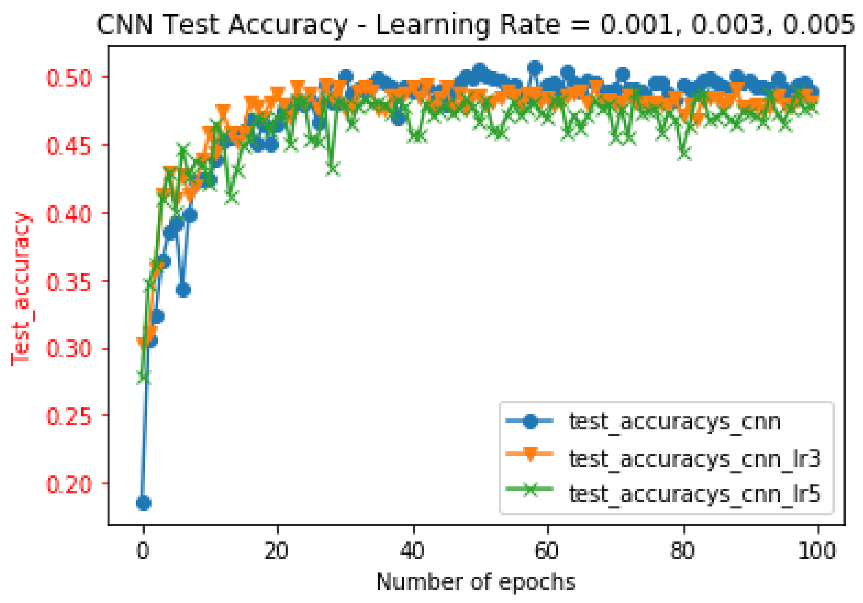


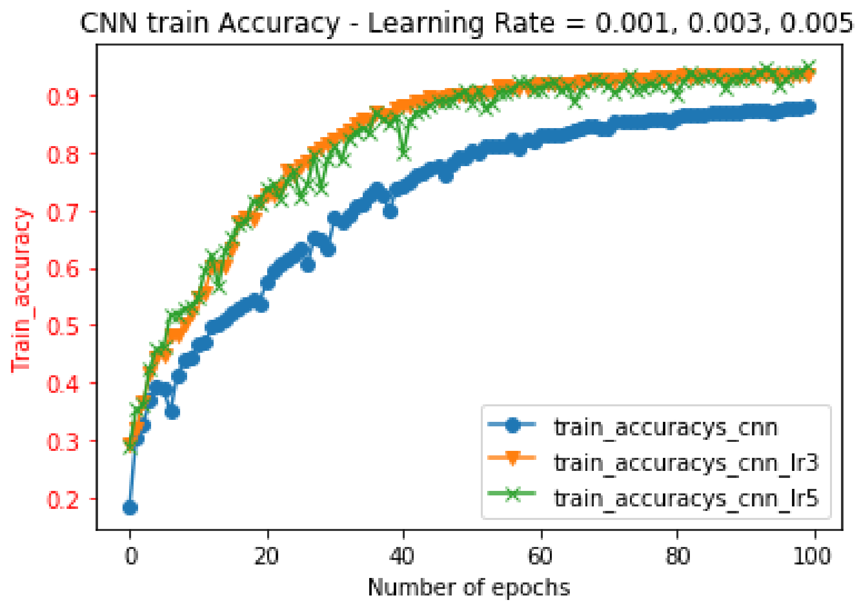


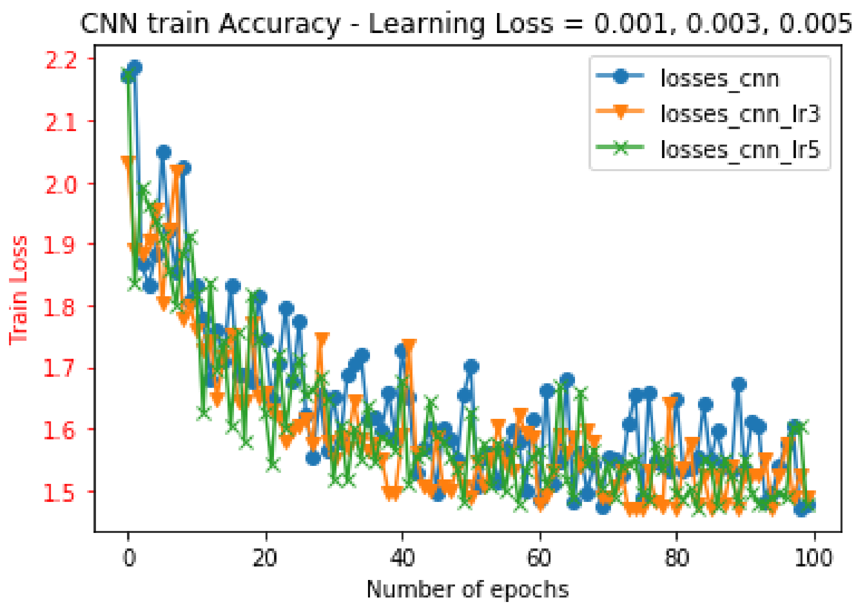


Tuning For Batch Size:
As for train accuracy, the greater the batch size is , the greater the train accuracy is.
As for test accuracy, it grows when the batch size grows from 64 to 128, however, it stops to grow when the batch size grows from 128 to 256.
As for training loss, the greater the batch size, the smaller the training loss at the end.

Group2 : CNNs: (Compare different learning rates)
learning rate: 0.001, 0.003, 0.005; other parameters are as default.

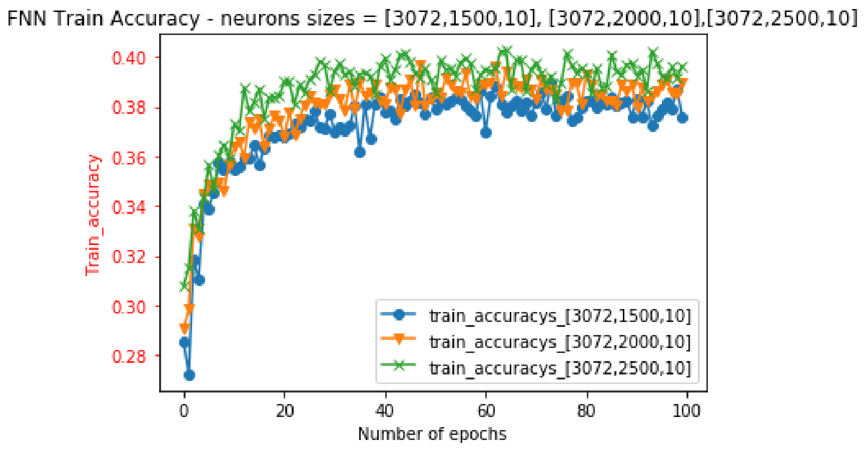


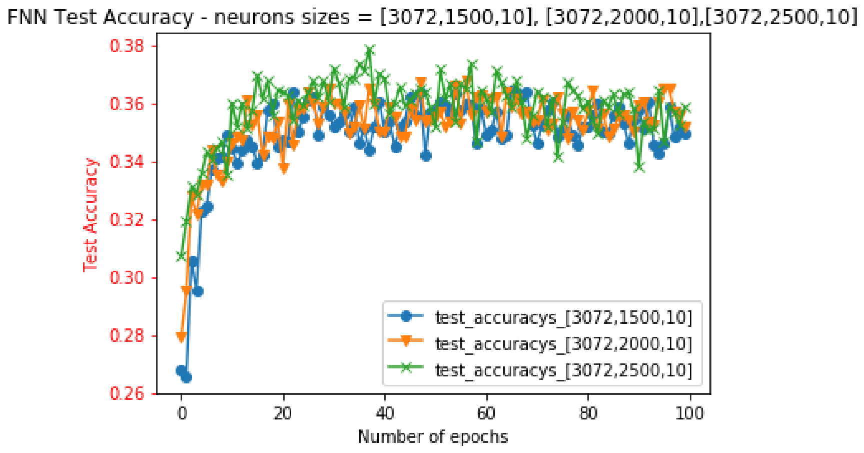


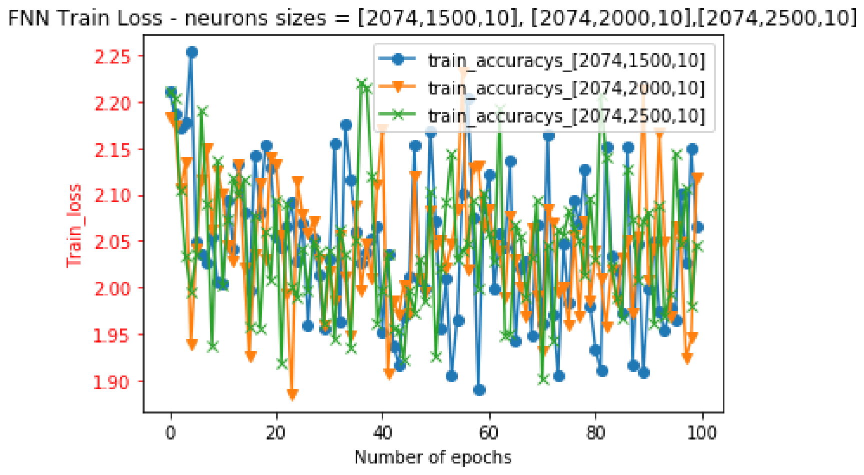


Tuning For Learning Rate:
As for Train Accuracy and Train Loss, different learning rates do not make much differences.
As for Test Accuracy, the CNN with lower learning rate performs better, however, the difference is very subtle.

Group3 : FNNs: (Compare different number of neurons in fc layers)
number of neurons = [3072, 1500, 10] ,[3072, 2000, 10], [3072, 2500, 10]; other parameters are as default

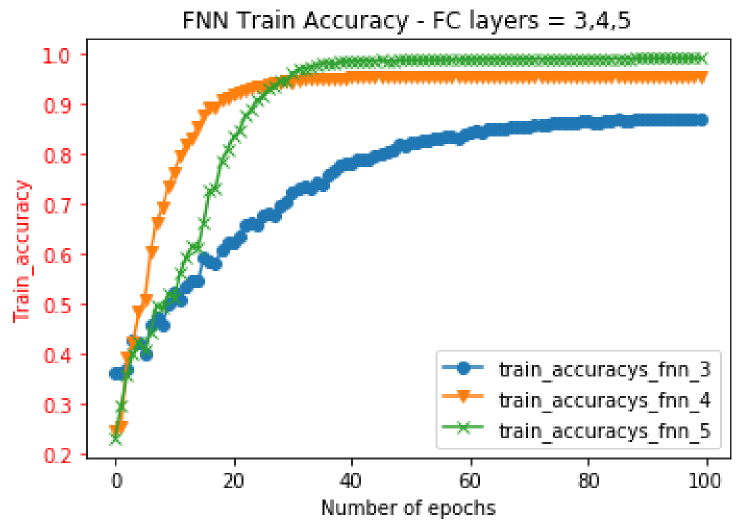


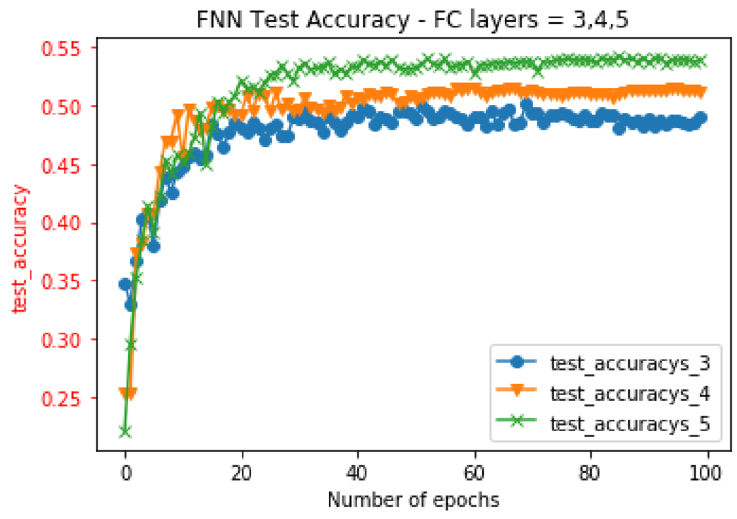


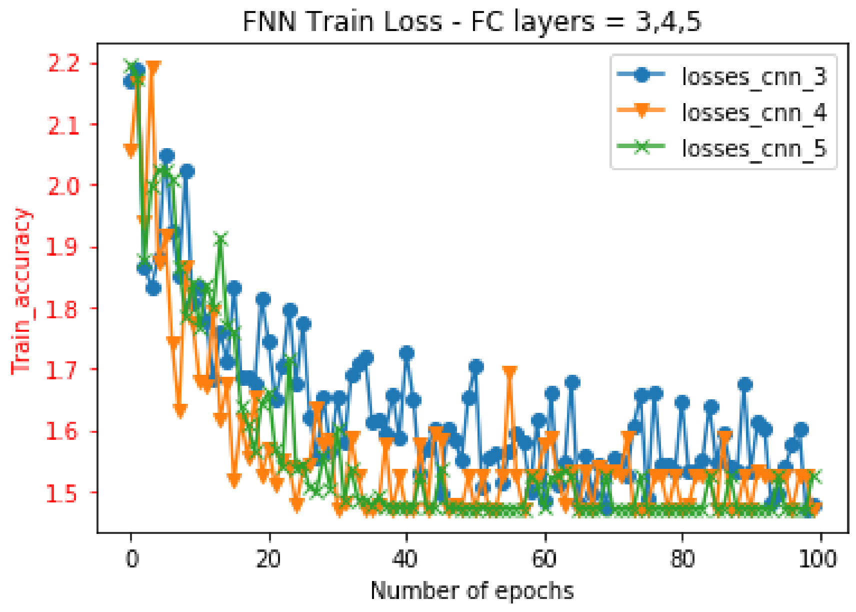


Tuning For Number of Neurons in FC Layers in FNNs:
As the the number of neurons in hidden layers increase from 1500 to 2000 then to 2500, the training accuracy increases slightly as well. As for Test Accuracy and Train Loss, the number of neurons in hidden layers do not make much difference.

Group4 : CNNs: (Compare different number of fully connected layers) number of layers in FC :3, 4, 5
number of neurons in FC layers : [2704, 1500, 10], [2704, 2000, 1500, 10],[2704, 2000,1500,500,10]; other parameters are as default.

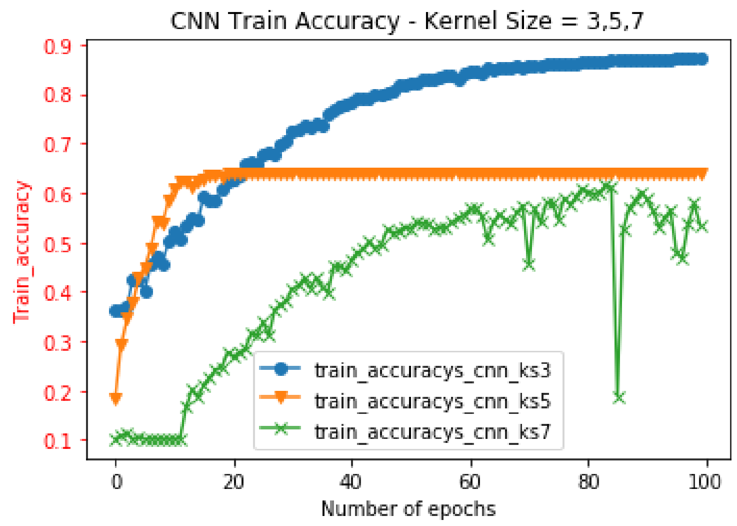


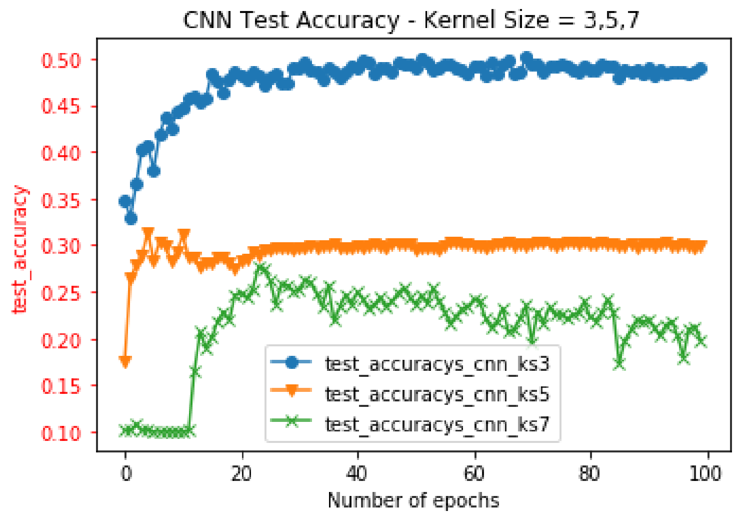


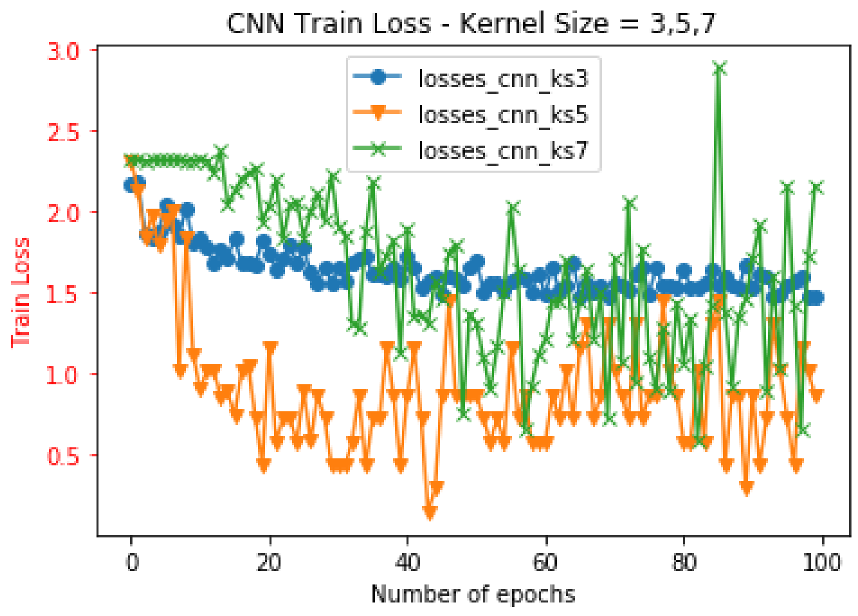


Tuning For Number of Fully Connected Layers in CNNs:
when we increase the number of fully connected layers in CNN, the final train accuracy and the final test accuracy performs better as well, while the Training loss perform worse.

Group5 : CNNs: (Compare different kernel sizes)
kernel sizes:3x3, 5x5, 7x7;
other parameters are as default.







Tuning For Kernel Sizes in CNN:
when we increase the number of kernel sizes in CNN, both the train accuracy and the test accuracy perform much worse. While the Training loss becomes more unstable.

5. Discussion :

From experiment 1(FNN) to experiment 2(CNN):
Both the test accuracy and the train accuracy improve. Because the data type is image, it exhibits locality(auto-correlation). And the CNN just look at part of the image at a time, make the assumption of locality, and hence more powerful.

From experiment 2(CNN with raw image) to experiment 3(CNN with normalized image):
As for train accuracy, the raw version is a slight better.
As for test accuracy, the raw version is much more better.
It may be caused by over-fitment.

In experiment 4(Tuning):
Group1 : FNNs (Compare different batch sizes)
Batch Size = 64, 128, 256; other parameters are as default.
Tuning For Batch Size:
As for train accuracy, the greater the batch size is , the greater the train accuracy is.
As for test accuracy, it grows when the batch size grows from 64 to 128, however, it stops to grow when the batch size grows from 128 to 256.
As for training loss, the greater the batch size, the smaller the training loss at the end.

Group2 : CNNs: (Compare different learning rates)
learning rate: 0.001, 0.003, 0.005; other parameters are as default.
Tuning For Learning Rate:
As for Train Accuracy and Train Loss, different learning rates do not make much differences.
As for Test Accuracy, the CNN with lower learning rate performs better, however, the difference is very subtle.

Group3 : FNNs: (Compare different number of neurons in fc layer)
number of neurons = [3072, 1500, 10] ,[3072, 2000, 10], [3072, 2500, 10];
other parameters are as default.
As the the number of neurons in hidden layers increase from 1500 to 2000 then to 2500, the training accuracy increases slightly as well. As for Test Accuracy and Train Loss, the number of neurons in hidden layers do not make much difference.

Group4 : CNNs: (Compare different number of fully connected layers) number of fc layers:3, 4, 5;
other parameters are as default.
Tuning For Number of Fully Connected Layers in CNNs:
when we increase the number of fully connected layers in CNN, the final train accuracy and the final test accuracy performs better as well, while the Training loss perform worse.

Group5 : CNNs: (Compare different kernel sizes)
kernel sizes:3x3, 5x5, 7x7;
other parameters are as default.
Tuning For Kernel Sizes in CNN:
when we increase the number of kernel sizes in CNN, both the train accuracy and the test accuracy perform much worse. While the Training loss becomes more unstable.

In summary, the most effective way to increase the performance is to increase the number of fully connected layers in the neural network in this particular problem.

2 [Boosting - 25 points]

Consider the following examples $(x, y) \in \mathbb{R}^2$ (i is the example index):

i	Label	Hypothesis 1				Hypothesis 2			
		D_0	$f_1 \equiv$ $[x > 2\ ]$	$f_2 \equiv$ $[y > 5\ ]$	$h_1 \equiv$ $[\ x > 2;\ ]$	D_1	$f_1 \equiv$ $[x > 10\ ]$	$f_2 \equiv$ $[y > 11\ ]$	$h_2 \equiv$ $[\ y > 11;\ ]$
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)
1	$-$	0.10	$-$	$+$	$-$	0.05	$-$	$-$	$-$
2	$-$	0.10	$-$	$-$	$-$	0.05	$-$	$-$	$-$
3	$+$	0.10	$+$	$+$	$+$	0.05	$-$	$-$	$-$
4	$-$	0.10	$-$	$-$	$-$	0.05	$-$	$-$	$-$
5	$-$	0.10	$-$	$+$	$-$	0.05	$-$	$+$	$+$
6	$-$	0.10	$+$	$+$	$+$	0.20	$-$	$-$	$-$
7	$+$	0.10	$+$	$+$	$+$	0.05	$+$	$-$	$-$
8	$-$	0.10	$-$	$-$	$-$	0.05	$-$	$-$	$-$
9	$+$	0.10	$-$	$+$	$-$	0.20	$-$	$+$	$+$
10	$+$	0.10	$+$	$+$	$+$	0.05	$-$	$-$	$-$

Table 1: Table for Boosting results

i	x	y	Label
1	0	8	$-$
2	1	4	$-$
3	3	6	$+$
4	-2	1	$-$
5	-1	13	$-$
6	9	11	$-$
7	12	7	$+$
8	-7	-1	$-$
9	-3	12	$+$
10	5	9	$+$

In this problem, you will use Boosting to learn a hidden Boolean function from this set of examples. We will use two rounds of AdaBoost to learn a hypothesis for this data set. In each round, AdaBoost chooses a weak learner that minimizes the error ϵ. As weak learners, use hypotheses from the following classes of functions: either functions of the form (a) $f_1 \equiv [x > \theta_x]$, or functions of the form (b) $f_2 \equiv [y > \theta_y]$, for some integers θ_x, θ_y. There is no need to try many values of θ_x, θ_y; appropriate values should be clear from the data.

1. [**7 points**] Start the first round with a uniform distribution D_0. Place the value for D_0 for each example in the third column of Table 1. Find the best weak learners, f_1 and f_2, by selecting an integer θ_x for f_1 and an integer θ_y for f_2 that minimizes their errors. Then, fill up the table based on the prediction of $f_1 \equiv [x > \theta_x]$ in column 4 and $f_2 \equiv [y > \theta_y]$ in column 5 for each of the example in the Table 1 above.

2. [**6 points**] Find the hypothesis given by the weak learner that minimizes the error ϵ for that distribution. Place this hypothesis as the heading to the sixth column of Table 1, and give its prediction for each example in that column.

3. [**7 points**] Now compute D_1 for each example, find the new best weak learners f_1 and f_2, and select hypothesis that minimizes error on this distribution, placing these values and predictions in the seventh to tenth columns of Table 1.

4. [**5 points**] Write down the final hypothesis produced by AdaBoost. Construct a distribution $D_0 = \{0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1\}$

   For hypothesis 1,

   let $f_1 \equiv [x > 2], \epsilon_{f1} = Pr_D[h_t \neq h_t(X_i)] = D_0 \cdot \{0, 0, 0, 0, 0, 1, 0, 0, 1, 0\} = 0.2$

   let $f_2 \equiv [y > 5], \epsilon_{f2} = Pr_D[h_t \neq h_t(X_i)] = D_0 \cdot \{1, 0, 0, 0, 1, 1, 0, 0, 0, 0\} = 0.3$

   thus $\epsilon_1 = min\{\epsilon_{f1}, \epsilon_{f2}\} = 0.2, h_1 \equiv [x > 2], \alpha_1 = \frac{1}{2} ln \frac{1-\epsilon_1}{\epsilon_1} = 0.693$

   $h_1(X_i) = \{-1, -1, 1, -1, -1, 1, 1, -1, -1, 1\},$

   $y_i = \{-1, -1, 1, -1, -1, -1, 1, -1, 1, 1\}$

   $-y_i h_1(X_i) = \{-1, -1, -1, -1, -1, 1, -1, -1, 1, -1\},\ Z_t = sum(D_0) = 1,$

   $D_1 = \frac{D_0}{Z_t} \cdot e^{-\alpha_1 y_i h_1(X_i)} = \{0.05, 0.05, 0.05, 0.05, 0.05, 0.2, 0.05, 0.05, 0.2, 0.05\}$

   For hypothesis 2,

   let $f_1 \equiv [x > 9], \epsilon_{f1} = D_1 \cdot \{0, 0, 1, 0, 0, 0, 0, 0, 1, 1\} = 0.375$

   let $f_2 \equiv [y > 11], \epsilon_{f2} = D_1 \cdot \{0, 0, 1, 0, 1, 0, 1, 0, 0, 1\} = 0.250$

   thus $\epsilon_2 = min\{\epsilon_{f1}, \epsilon_{f2}\} = 0.250, h_2 \equiv [y > 11], \alpha_2 = \frac{1}{2} ln \frac{1-\epsilon_2}{\epsilon_2} = 0.549$

   $h_1(X_i) = \{-1, -1, -1, -1, 1, -1, -1, -1, 1, -1\},$

   $y_i = \{-1, -1, 1, -1, -1, -1, 1, -1, 1, 1\}$

   $-y_i h_1(X_i) = \{-1, -1, 1, -1, 1, -1, 1, -1, -1, 1\},$

   $D_2 = \frac{D_1}{Z_t} \cdot e^{-\alpha_2 y_i h_1(X_i)} = \{0.03, 0.03, 0.09, 0.03, 0.09, 0.11, 0.09, 0.03, 0.11, 0.09\}/Z_t$

   $Final\ hypothesis : H_{final} = 0.693[x > 2] + 0.549[y > 11] = x > 2$

What to submit: Fill out Table 1 as explained, show computation of α and $D_1(i)$, and give the final hypothesis, H_{final}. This should all be included in the final pdf report along with your neural network experiments.

3 [SVM - 25 points]

We have a set of six labeled examples D in the two-dimensional space, $D = \{(\mathbf{x}^{(1)}, y^{(1)}), ..., (\mathbf{x}^{(6)}, y^{(6)})\}$, $\mathbf{x}^{(i)} \in \mathbb{R}^2$ and $y^{(i)} \in \{1, -1\}, i = 1, 2, ..., 6$ listed as follows:

i	$\mathbf{x}_1^{(i)}$	$\mathbf{x}_2^{(i)}$	$y^{(i)}$
1	0	-2	1
2	1.6	-2.4	1
3	-2.6	1.3	-1
4	2.5	-0.3	1
5	-0.2	3	-1
6	-2	0	-1

(a) [4 points] We want to find a linear classifier where examples $\mathbf{x}$ are positive if and only if $\mathbf{w} \cdot \mathbf{x} + \theta \geq 0$.

1. [1 points] Find an easy solution $(\mathbf{w}, \theta)$ that can separate the positive and negative examples given.

   Define $\mathbf{w} = [1, -1]$

   Define $\theta = 0$

2. [4 points] Recall the Hard SVM formulation:

$$\mathbf{min_w} \frac{1}{2}||\mathbf{w}||^2 \tag{1}$$

$$\text{s.t } y^{(i)}(\mathbf{w} \cdot \mathbf{x}^{(i)} + \theta) \geq 1, \forall (\mathbf{x}^{(i)}, y^{(i)}) \in D \tag{2}$$

   What would the solution be if you solve this optimization problem? (Note: you don't actually need to solve the optimization problem; we expect you to use a simple geometric argument to derive the same solution SVM optimization would result in).

   Define $\mathbf{w} = [\frac{1}{2}, -\frac{1}{2}]$

   Define $\theta = 0$

3. [5 points] Given your understanding of SVM optimization, how did you derive the SVM solution for the points in Figure 1?

svm_points.png

Figure 1: Training examples for SVM in question 1.(a)

The goal is to find the a separator with the maximum margin without misclassification. Then find two support vectors which is x^1 and x^6, than the perpendicular bisector for line x^1x^6 is the solution.

(b) [15 points] Recall the dual representation of SVM. There exists coefficients $\alpha_i > 0$ such that:

$$\mathbf{w}^* = \sum_{i \in I} \alpha_i y^{(i)} \mathbf{x}^{(i)} \tag{3}$$

where I is the set of indices of the support vectors.

1. [5 points] Identify support vectors from the six examples given.

   Define $I =$ $\underline{\quad 1, 6 \quad}$

2. [5 points] For the support vectors you have identified, find α_i such that the dual representation of $\mathbf{w}^*$ is equal to the primal one you found in (a)-2.

   Define $\alpha = \{\alpha_1, \alpha_2, ..., \alpha_{|I|}\} =$ $\underline{\quad \frac{1}{4}, \frac{1}{4} \quad}$

3. [5 points] Compute the value of the hard SVM objective function for the optimal solution you found.

   Objective function value $=$ $\underline{\quad \frac{1}{4} \quad}$

(c) [10 points] Recall the objective function for soft representation of SVM.

$$\mathbf{min}\ \frac{1}{2}||\mathbf{w}||^2 + C\sum_{j=1}^{m} \xi_i \tag{4}$$

$$\text{s.t } y^{(i)}(\mathbf{w} \cdot \mathbf{x}^{(i)} + \theta) \geq 1 - \xi_i, \xi_i \geq 0, \forall(\mathbf{x}^{(i)}, y^{(i)}) \in D \tag{5}$$

where m is the number of examples. Here C is an important parameter. For which **trivial** value of C, the solution to this optimization problem gives the hyperplane that **you have found in (a)-2**? Comment on the impact on the margin and support vectors when we use $C = \infty$, $C = 1$, and $C = 0$. Interpret what C controls.

Answer:
When $C \geq \frac{1}{4}$, the solution to this optimization problem gives the hyperplane that found in(a)-2.
C is the parameter that indicate how much it emphasize on error, i.e, how much we penalize the misclassification.

If $C = \infty$, it means no data violate the margin is possible, we only want $\mathbf{w}, \theta$ to separate the data.
If $C = 0$, we can set ξ_i to anything, then $\mathbf{w} = 0$, it basically ignores the data.
If $C = 1$, it allows some degree of violation.

Checklist:

1. Submit your report in a pdf format to Canvas. Make sure to include your experiments, answers to Boosting and SVM as well as any extra credits in your report.

2. Submit your report in a pdf format to Gradescope. Make sure to include your experiments, answers to Boosting and SVM as well as any extra credits in your report.

3. Submit your code to Canvas.

4. Comment out any code which is not an import statement or contained in a function definition, then submit it to Gradescope and pass the autograder.

Note on Grading for Homework 3
For the programming portion of the assignment, we will only be conducting simple test cases on Gradescope. If you pass all the test cases on Gradescope, you should automatically receive 20 points. The remaining 30 points including the write up of your report will be graded manually for correctness.