CIS519: Applied Machine Learning	**Spring 2018**
Homework 3	
Handed Out: March 20^{th}, 2018	*Due: March* 30^{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 SGD as the optimizer with learning rate of 0.001 and momentum of 0.9, batch size of 64, and max training iterations of 100 during training. The loss function is cross-entropy loss.

Experiment 1: 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 1024 neurons followed by sigmoid activation function. The last layer outputs 10 probabilities for each class. Plot the training accuracy over epoch and report the final test accuracy.

Layer	Hyperparameters
Fully Connected1	Output channel = 1024. Followed by Sigmoid
Fully Connected2	Output channel = 10. Followed by Sigmoid

Experiment 2: Convolutional Neural Network [10 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 size = (5x5x6), stride = 1, padding = 0. Followed by ReLU
Pool1	MaxPool operation. Kernel size = (2x2)
Convolution2	Kernel size = (5x5x16), stride = 1, padding = 0. Followed by ReLU
Fully Connected1	Output channel = 120. Followed by ReLU
Fully Connected2	Output channel = 84. Followed by ReLU
Fully Connected3	Output channel = 10. Followed by Sigmoid

Experiment 3: Image Preprocessing [10 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:

For each image, you need to normalize pixel values in each channel by subtracting its mean and dividing by the standard deviation. (Note that the mean and standard deviation are calculated from pixel values in all RGB channels, separately for each image.) Once you normalized the image, train the network. Plot the training accuracy over epoch, and report the final test accuracy.

Additional Optional Experiments: Data augmentation is also a useful technique used to improve the classification accuracy during training. Common data augmentation includes: randomly flip horizontally and vertically, randomly cropped, and so on. You are encouraged to check the Pytorch tutorial. (http://pytorch.org/docs/master/torchvision/transforms.html)

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 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].

Note that you do not need to try out all possible parameter combinations. However, you are expected to find a better hyperparameter setting than the default setting. You can also change the max training iterations to be as large as you want in order to improve the model accuracy. Report the best test accuracy and the best hyperparameter settings.

Additional Optional Experiments: 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.

What to Report: [10 points]
1. Plots of training accuracy over epoch and test accuracy in each of the four experiments.
2. If your implementation is correct, you should be able to see some improvement of model performance from experiment 1 to experiment 4. Briefly, explain the reasons for such improvements at each stage.

2 [Boosting - 25 points]

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

i	x	y	Label
1	0	8	−
2	1	4	−
3	3	7	+
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 of the form (a) $f_1 \equiv [x > \theta_x]$ or (b) $f_2 \equiv [y > \theta_y]$, for some integers θ_x, θ_y (either one of the two forms, not a disjunction of the two). There should be 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. Write the new representation of the data in terms of the *rules of thumb*, f_1 and f_2, in the fourth and fifth columns of Table 1.

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

Table 1: Table for Boosting results

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.

What to submit: Fill out Table 1 as explained, show computation of α and $D_1(i)$, and give the final hypothesis, H_{final}.

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	-2	0	1
2	-2.4	-1.6	1
3	1.3	2.6	-1
4	-0.3	-2.5	1
5	3	0.2	-1
6	0	2	-1

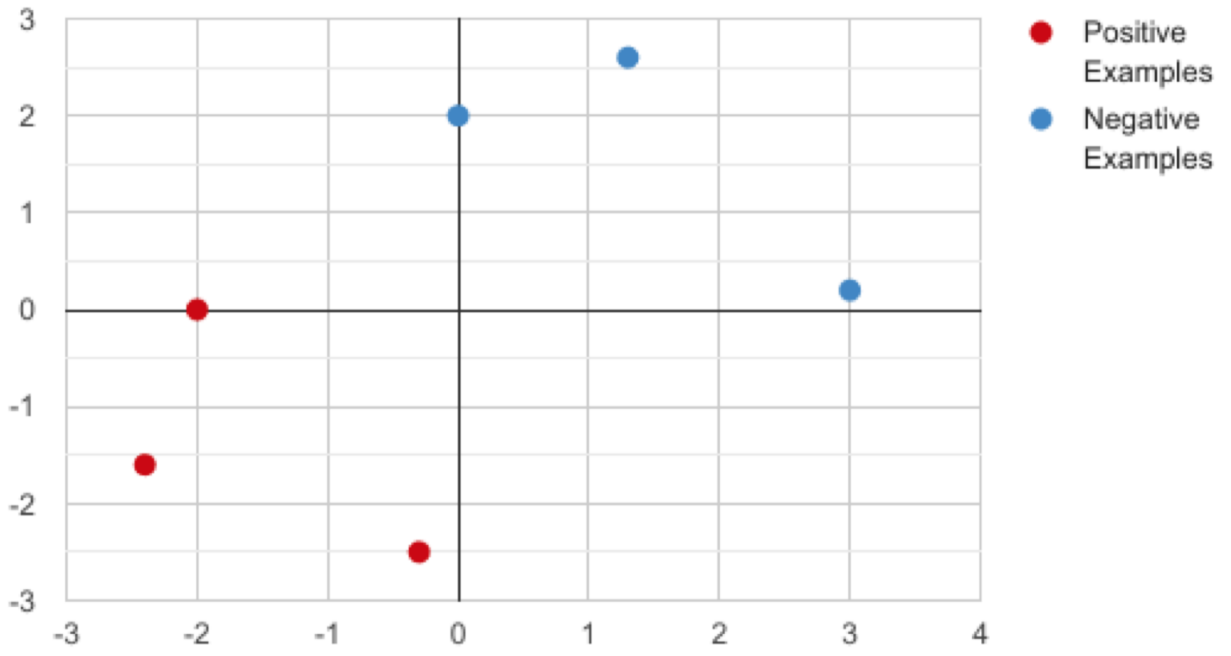

Figure 1: Training examples for SVM in question 1.(a)

(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 $\mathbf{w} =$ _____

   Define $\theta =$ _____

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

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

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

   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 $\mathbf{w} =$ _____

   Define $\theta =$ _____

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

(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 $I =$ _____

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 $\alpha = \{\alpha_1, \alpha_2, ..., \alpha_{|I|}\} =$ _____

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

   Objective function value = _____

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

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

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

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.