Homework 1: Perceptron and K-Nearest Neighbor Classification

EE 459/559 2020 Spring

Discussion is encouraged, but homework and codes shall be written on your own

Due: 11:59pm Feb 23th 2020

Late submissions will not accepted

1 Fitting classifiers by hand

1. (20 points) Consider two training points:

$$x^{(1)} = 1, y^{(1)} = -1$$
$$x^{(2)} = -2, y^{(2)} = 1.$$

Derive the Perceptron algorithm step by step until it converges by hand.

Initialization: step size $\alpha = 0.2$, bias $b = 1$, and $w = 1$.

The order at which the training points are used: Sample 1, Sample 2, Sample 1, Sample 1, Sample 1, Sample 1, Sample 1, Sample 2, Sample 2, Sample 1, Sample 2, Sample 1, Sample 2,... (may not necessarily use the entire sequence)

Stopping criterion: until all the samples are correctly classified.

2. (20 points) Write out the 3-nearest neighbor classifier for a feature $x \in \mathbb{R}$ based on the following training points:

$$x^{(1)} = 6, y^{(1)} = -1,$$
$$x^{(2)} = 4, y^{(2)} = -1,$$
$$x^{(3)} = 3, y^{(3)} = -1,$$
$$x^{(4)} = -3, y^{(4)} = 1,$$
$$x^{(5)} = 0, y^{(5)} = 1.$$

Hint: the classifier shall have the following form with the threshold c to be determined.

$$y = \begin{cases} -1, \text{ if } x \geq c \\ +1, \text{ if } x < c. \end{cases} \tag{1}$$

2 Programming questions

We recommend to use google's cloud based tensorflow platform: https://colab.research.google.com/ (You can also install python and tensorflow on your own computer. Anaconda is recommended to manage packages: https://www.anaconda.com/distribution/.)

In this task, we will perform a digit recognition task, where we are given an image of a hand-written digit, and the goal is to predict which number it represents.

We consider a simplified version of this problem, i.e., a binary classification problem between two numbers 3 and 5. This programming assignment shall be written using python. For the programming questions, an explanation of your results together with your codes shall be submitted.

Dataset: Download the MNIST dataset. See https://en.wikipedia.org/wiki/MNIST_database for an introduction of the dataset. The dataset consists of hand-written digits from 0 to 9.

Loss function: In this task, we use 0-1 loss.

When training the classifiers, use only the training data. The test data shall be used to evaluate how your classifiers perform on unseen data.

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
```

Figure 1: Import Package.

1. (30 points) Implement the Perceptron algorithm. The stepsize $\alpha = 0.1$, the initialization of w is an all-one vector. Repeat your experiments for the following two stopping criteria. Report the training error and test error.

   (a) Stopping criterion is that 95% of the training data are correctly classified.

   (b) Stopping criterion is that 80% of the training data are correctly classified.

   Hint: Follow the steps shown in the pictures to import packages, download data, and pre-process data.

```
#load data and preprocess data
mnist = tf.keras.datasets.mnist
(train_images, train_labels),(test_images,test_labels)=mnist.load_data()
#choose pictures of number 3 and number 5
(train_images, train_labels),(test_images,test_labels)=mnist.load_data()#include all numbers from 0 to 9
index_train=np.where((train_labels ==3) | (train_labels ==5))#index of numbers 3 and 5 in training data
index_test=np.where((test_labels ==3) | (test_labels ==5))#index of numbers 3 and 5 in test data
train_images_35=train_images[index_train]
train_images_35=train_images_35.reshape((len(train_images_35), train_images_35[1].size))
#label of number 3: -1; label of number 5: +1
train_labels_35=train_labels[index_train].astype('int')
test_images_35=test_images[index_test]
test_images_35=test_images_35.reshape((len(test_images_35), train_images_35[1].size))
test_labels_35=test_labels[index_test].astype('int')
#change labels from '3' and '5' to '-1' and '+1'
train_labels_35[np.where(train_labels_35==3)]= -1
train_labels_35[np.where(train_labels_35==5)]=1
test_labels_35[np.where(test_labels_35==3)]= -1
test_labels_35[np.where(test_labels_35==5)]=1
```

Figure 2: Load Data and Preprocess Data.

```
#show the first 25 training data
plt.figure(figsize=(10,10))
for i in range(25):
  plt.subplot(5,5,i+1)
  plt.xticks([])
  plt.yticks([])
  plt.imshow(train_images_35[i].reshape((28,28)),)
  plt.xlabel('number '+ str(train_labels_35[i]))
plt.show()
```

Figure 3: Show First 25 Training Data.

```
#append dummy feature 1 to feature vectors, and then normalize
train_images_35_w_dummy=np.insert(train_images_35,784,1,axis=1)/255
test_images_35_w_dummy=np.insert(test_images_35,784,1,axis=1)/255
#check the dimension, the feature vector of each sample shall be 785
print(train_images_35_w_dummy.shape)
print(test_images_35_w_dummy.shape)
```

Figure 4: Append Dummy Feature and Normalize Data.

2. (30 points) Implement the k-nearest neighbor algorithm on the same dataset, for $k = 1, 2, 3, 4, 5$, respectively. Report the training error and test error for each k. Discuss how the choice of k affects the training error and the test error.