Homework 3

Due: March 13, 2018 at 7:00PM

Written Questions

Problem 1

(10 points)
Recall the update process for K-means. Let C_i denote the cluster of observations from sample S nearest to mean μ_i. Let t denote the iteration within the K-means update process. (*Note*: This is just an iterative restatement of the update rule from the slides):

$$\forall i \in \{1, ..., k\} : C_i^{t+1} = \{x \in S : i = \underset{j}{\operatorname{argmin}} ||x - \mu_j^t||\}$$

$$\forall i \in \{1, ..., k\} : \mu_i^{t+1} = \frac{1}{|C_i^{t+1}|} \sum_{x \in C_i^{t+1}} x$$

The loss for a K-means classifier is defined as follows:

$$\sum_{i=1}^{k} \sum_{x \in C_i} ||x - \mu_i||^2$$

Prove that the choices of μ_i^t minimizes loss given clusters $C_1^t, ..., C_k^t$.

Problem 2

(8 points)
Suppose we have a weighted coin that flips heads with probability p. We would like to predict the outcome of the next coin flip.

a. Suppose p is known. If $p \geq \frac{1}{2}$, we will predict heads for the next outcome as we know this to be the most likely possibility. If $p < \frac{1}{2}$, we will predict tails. What is the expected error of a single coin flip? *Note:* This method provides an optimal prediction for the next coin flip.

b. Suppose p is unknown. We instead opt to flip the coin, and use the result of that coin flip as a prediction for the following coin flip. What's the expected error of this prediction method? *Note:* This prediction method may be seen as analogous to a 1-NN model. We are making a prediction for an unknown label based upon the label of the nearest neighbor: our instanced coin flip.

c. Let E_a and E_b denote the expected error of our prediction methods in parts a and b respectively. Find c such that $E_b \leq cE_a \ \forall p \in [0, 1]$.

d. What does this analysis tell us about the best performance we can hope for in a 1-NN model (under the formal model discussed in lecture - see slide 9)?

Problem 3

(10 points)
For any hypothesis space H, to show that the VC Dimension of H is d, you should show each of the following:

- There exists a set C of size d that is shattered by H
- There exists no set C of size $d+1$ that is shattered by H

1. Compute the VC dimension for the following hypothesis spaces:

   a. The class of signed intervals, $H = \{h_{a,b,s} : a \leq b, s \in \{-1, 1\}\}$ where:

   $$h_{a,b,s} = \begin{cases} s & \text{if } x \in [a, b] \\ -s & \text{if } x \notin [a, b] \end{cases}$$

   b. The class of origin-centered spheres in R^d, $H = \{h_{a,s} : s \in \{-1, 1\}, a \in R\}$ where:

   $$h_{a,s} = \begin{cases} s & \text{if } x \text{ is within the origin centered sphere of radius } a \\ -s & \text{if } x \text{ is outside the origin centered sphere of radius } a \end{cases}$$

2. Consider two hypothesis spaces H_1, H_2 such that $H_1 \subset H_2$. Prove that the VC Dimension of H_2 is at least as large as the VC Dimension of H_1.

Problem 4

(12 points)
In the following question, we will bound the VC dimension for a binary K-means classifier on a 1-dimensional space.

a. Let K be the space of K-means classifiers with k positive means and k negative means.

- Prove that any set C of size $2k$ can be shattered by K.
- Argue that there is no set of points C of size $2k+1$ that may be shattered by K

b. Is a polynomial (in k, d, $1/\epsilon$, and $1/\delta$) amount of data enough to determine a good K means classifier? Why or why not?

Programming Assignment

Introduction

In this assignment, you will implement three classifiers, all of which take continuous features as input. You will revisit Decision Trees with continuous inputs and implement both a k-Nearest Neighbor (k-NN) & k-Means Classifier.

The Assignment

Decision Trees, Revisited

In this assignment, you will be working with Decision Trees once again! This time, you will be dealing with both continuous features and multiple class labels. Given your previous experience with Decision Trees from HW1, we have provided most of the code for this part of the assignment. We have left you the task of deciding how to split nodes and calculating gain functions. These aspects differ from HW1 in the following ways:

- The gain functions you implemented in HW1 may no longer work in the multiclass setting. You will need to generalize the `entropy` and `gini_index` function so that they can compute these values on a dataset with multiple classes.
- Each Node in the Decision Tree must split on both a feature and a particular value of that feature.

k-Means Classifier

k-Means is a *clustering* algorithm most often used for *unsupervised* machine learning. In unsupervised learning, the learner is given a dataset with no labels and attempts to learn some useful representation of the dataset. Examples of unsupervised learning are clustering and data compression.

However, in this assignment, you will be using k-Means in order to build a classification model, a *supervised* learning task (one in which the learner is given labeled data). One way to create a classifer using k-Means is to do the following. Given a dataset $S = \{(X_1, y_1) \dots (X_n, y_n)\}$

- For each unique label y from the dataset S, run k-Means on $X_y = \{X_i \in S : y_i == y\}$. That is, run k-Means on the set of points that have the same label to determine k-clusters for that label.
- Store the k clusters returned for each label.
- To predict the label, y_{n+1} of a new datapoint X_{n+1}, find the cluster center closest to X_{n+1} and predict the label corresponding to that cluster center's label.

k-NN Classifier

Finally, you will be implementing a k-NN classifier. Please see the slides if you are unfamillar with k-NN classifiers.

Stencil Code

You have been provided with the following stencil files:

- `main.py`: contains a main function to read data, run classifiers and print/visualize results. This script takes paths to required datasets as arguments.

  To run your `main.py` program on a department machine, simply navigate to the directory where

`main.py` resides, and run the command

```
python main.py <Relative Path to the dataset file>
```

- `models.py`: contains three different *classes* of classifiers (Nearest Neighbors, k-means, and Decision Tree) that you will need to fill in.
- `kmeans.py`: contains helper functions for k-means clustering via iterative improvement that you will need to fill in.
- `kneighbors.py`: contains helper functions for k-nearest neighbors method that you will need to complete.
- `dtree.py`: contains helper functions for constructing decision tree.

Data

The datasets for this assignment are located in the `/course/cs1420/data/hw3` folder. To copy the files into your department machine, run the command

```
cp -p /course/cs1420/data/hw3/* <DEST_DIRECTORY>
```

where `<DEST DIRECTORY>` is the directory where you would like to copy the data files. If you are working locally, you will need to use the `scp` command to copy the files to your computer over `ssh`:

```
scp <login>@ssh.cs.brown.edu:/course/cs1420/data/hw3/* <DEST DIRECTORY>
```

Hand-written digits dataset

In the `digits.csv` file, each row is an observation of a 8 x 8 hand-written digit (0 - 9), containing a label in the first column and 8 x 8 = 64 features (pixel values) in the rest of columns.

Iris dataset

The `fishiris.csv` file consists of 3 different types of irises' (Setosa, Versicolour, and Virginica) petal and sepal information. The rows being the samples and the columns being: Label, Sepal Length, Sepal Width, Petal Length and Petal Width.

Functions

1. `models.py`

   In this file, you will implement five functions, two or three for each model. They are:

   - `KNeighborsClassifier`
     - **train()**: fit the training data (inputs and labels) to model
       Note: KNN is a lazy algorithm, because it doesn't learn a discriminative function from the training data but "memorizes" the training dataset instead. This function has been filled in.
     - **predict()**: predict label of inputs

  - **accuracy()**: compute accuracy of predictions against true labels
- `KmeansClassifier`
  - **train()**: for each unique label class in the training data, take all the data in that label class, and run k-means on it. That is, learn a set of k cluster centroids (representatives) for each label class that are robust (because they are estimated using a lot of data) and diverse. Store each label class and its k cluster centroids as a key-value pair in our dictionary-type *model attribute*
  - **predict()**: predict label of inputs using the label of closest centroid
  - **accuracy()**: compute accuracy of predictions against true labels
- `DecisionTree`: The functions for this class have already been implemented for you since you wrote similar functions for HW1.

2. `kneighbors.py`:

   In this file, you will implement three functions, they are:

   - **euclidean_distance()**: calculate distance between two data instances.
     Note: To make predictions, we need to measure the similarity between two data instances. Given that all measurements on the two datasets are numeric, we can directly use the Euclidean distance measure.
   - **get_neighbors_indices()**: use the similarity measure (here we're using Euclidean distance measure) to collect the k most similar instances for a given unseen instance.
   - **get_response()**: get the majority voted response from neighbors.

   Note: Once we have located the most similar neighbors for a test instance, the next task is to devise a predicted response based on those neighbors. We can do so by allowing each neighbor to vote for their class, and take the majority vote as the prediction.

3. `kmeans.py`:

   In this file, you will implement four functions, they are:

   - **init_centroids()**: pick **k** random data points as cluster centers called centroids.
   - **assign_step()**: assign each data instance to its nearest cluster centroid using Euclidean distance measure.
   - **update_step()**: find the new cluster centroids by taking the average of its assigned data points.
   - **kmeans()**: run the k-means algorithm: initialize centroids, then repeat the assignment step and update step until the percentage centroids change between two iterartions is below a threshold or the maximum iteration time is met.

   Note: You might also want to create a separate function that calculates the Euclidean distance between two data points in the `kmeans.py` file. Please feel free to do so.

4. `dtree.py`:

   - **entropy()**: calculate the entropy for a multi-classes dataset.
   - **gini_index()**: calculate the gini index for multi-classes dataset.
   - **split_recurs()**: construct the decision tree for dealing with continuous input data.

Note: The logic of constructing the tree is the same as what you have done in HW1. Things should be noticed: the datasets are multi-classes and the attributes now have continues values rather than only be True or False.

We heavily encourage you to familiarize yourself with each of the functions beforehand. While there are many functions that need to be implemented for this assignment, most of them are fairly simple, especially functions within `model.py`. Think about how the helper functions fit together within each of the classifiers.

Project Report

The next section outlines some guiding questions that you should answer in your report. Please leave any code that you use in your final handin but make sure that it is **not** run by default when your program is run. We ask that your final report be a PDF file named `report.pdf`, which must be handed in along with your code. You may use any program to create the PDF file, but we highly recommend using LaTeX. We have provided an example report available on our course website to get you started.

Guiding Questions

- What does the classification plot look like when you apply KNN to the Iris data? How does the complexity of dicision boundries change with different k? Feel free to explore the helper function **plot_KNN()** in the `main.py` file.
- What do your k-means clusters look like when you apply k-means to the digits dataset? Explore the helper function **plot_Kmeans()** in the `main.py` file and attach the pyplot figure obtained after you run the `main` program.
- Tune the parameters and see if the accuracies of the models improves. Graph or table out accuracy with different parameters for k-nearest neighbors, k-means classifiers and decision trees based on given two datasets.

Grading Breakdown

Your grade for this homework will be calculated as follows:

Written Questions	40%
Nearest Neighbors Classifier	18%
Decision Trees with ranges	12%
K-means Classifier via iterative improvement	18%
Report	12%
Total	100%

Handing in

Written Questions

Answers to the written questions should be printed, stapled and labeled with the date, homework number and your unique course ID. You should place your handin in the CS1420 handin box located on the second floor of the CIT.

Programming Assignment

To hand in the programming component of this assignment, first ensure that your code runs on *Python 3* using our course `virtualenv`. You can activate the `virtualenv` on a department machine by running the following command in a Terminal:

```
source /course/cs1420/cs142_env/bin/activate
```

Once the `virtualenv` is activated, run your program and ensure that there are no errors. We will be using this `virtualenv` to grade all programming assignments in this course so we recommend testing your code on a department machine each time before you hand in. Note that handing in code that does not run may result in a significant loss of credit.

To hand in the coding portion of the assignment, run `cs142_handin hw3` from the directory containing all of your source code and your report in a file named `report.pdf`.

Anonymous Grading

You need to be graded anonymously, so do not write your name anywhere on your handin. Instead, you should use the course ID that you generated when filling out the collaboration policy form. If you do not have a course ID, you should email the HTAs as soon as possible.

Obligatory Note on Academic Integrity

Plagiarism—don't do it.

As outlined in the Brown Academic Code, attempting to pass off another's work as your own can result in failing the assignment, failing this course, or even dismissal or expulsion from Brown. More than that, you will be missing out on the goal of your education, which is the cultivation of your own mind, thoughts, and abilities. Please review this course's collaboration policy and, if you have any questions, please contact a member of the course staff.

Feedback

Many aspects of this course have been radically redesigned compared to previous iterations of the course. To continually improve this class and future iterations of the class, we are offering an online **anonymous feedback form**. If you have comments, we would love to hear your constructive feedback on this assignment and on any other aspects of the course.