CS 178 Homework 1

Machine Learning, CS 178, Winter 2022

Due Date: 11:59pm Thursday Jan 13th, submit via Gradescope

This homework (and many subsequent ones) will involve data analysis and reporting on methods and results using Python code. You have to submit ***a single PDF file*** that contains everything to Gradescope, and associated each page of the PDF to each problem. This includes any text you wish to include to describe your results, the complete code snippets of how you attempted each problem, any figures that were generated, and scans of any work on paper that you wish to include. It is important that you include enough detail that we know how you solved the problem, since otherwise we will be unable to grade it.

I recommend that you use Jupyter/iPython notebooks to write your report. It will help you not only ensure all of the code for the solutions is included, but also provide an easy way to export your results to a PDF file [1]. I recommend liberal use of Markdown cells to create headers for each problem and sub-problem, explaining your implementation/answers, and including any mathematical equations. For parts of the homework you do on paper, scan it in such that it is legible (there are a number of free Android/iOS scanning apps, if you do not have access to a scanner), and include it as an image in the iPython notebook[2]. If you have any questions/concerns about using iPython, ask us on Piazza. If you decide not to use iPython notebooks, but go with Microsoft Word or Latex to create your PDF file, you have to make sure all of the answers can be generated from the code snippets included in the document.

Summary so far: (1) submit a single, standalone PDF report, with all code; (2) I recommed Jupyter notebooks.

Points: This homework adds up to a total at **100 points** points.

Problem 1: Python & Data Exploration	20 points
Problem 2: Linear Algebra	25 points
Problem 3: kNN Predictions	25 points
Problem 4: Conditional, Marginal, and Posterior Probabilities	25 points
Statement of Collaboration	5 points

[1] For example, by doing a *Print Preview* in Chrome and *printing* it to a PDF.

[2] Tips from Gradescope: `http://gradescope-static-assets.s3-us-west-2.amazonaws.com/help/submitting_hw_guide.pdf`

Problem 1: Python & Data Exploration (20 points)

In this problem, we will explore some basic statistics and visualizations of an example data set. First, download the zip file for Homework 1, which contains some course code (the `mltools` directory) and the "Fisher iris" data set, and load the latter into Python:

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

iris = np.genfromtxt("data/iris.txt",delimiter=None)  # load the text file
Y = iris[:,-1]              # target value is the last column
X = iris[:,0:-1]            # features are the other columns
```

The Iris data consist of four real-valued features used to predict which of three types of iris flower was measured (a three-class classification problem).

1. Use `X.shape` to get the number of features and the data points. Report both numbers, mentioning which number is which. *(5 points)*
2. Y takes value from 0, 1 and 2 in this data set, slicing the data points in 3 different groups. For each group, for each feature, plot a histogram(`plt.hist()`) of the data values (`plt.subplots(3, 4)`) *(5 points)*
3. Compute the mean, variance and standard deviation of the data points for each feature (`np.mean`, `np.var`, `np.std`) *(5 points)*
4. For each pair of features (1,2), (2,3), and (3,4), plot a scatterplot (see `plt.plot` or `plt.scatter`) of the feature values, colored according to their target value (class). (For example, plot all data points with $y = 0$ as blue, $y = 1$ as green, etc.) *(5 points)*

Problem 2: Basic Linear Algebra (25 points)

In this problem, you will revisit of some basic knowledge of linear algebra that is useful in machine learning. Please present all necessary details in your derivation.

1. Under what condition does a matrix have an inverse? *(5 points)*
2. Suppose we have two matrices,

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & -1 & 1 \\ 1 & 3 & 2 \end{bmatrix}, B = \begin{bmatrix} 0 & -8 & -2 \\ 1 & -4 & -2 \\ -4 & 4 & 1 \end{bmatrix},$$

compute the determinant of A and B. *(5 points)*

3. Compute the inverse of A and B. *(5 points)*
4. Compute the inverse of $A^\top$ and $B^\top$ based on what you have computed previously. *(5 points)*
5. Let $C = AB$. Compute the inverse of C based on what you have computed previously. *(5 points)*

Problem 3: kNN predictions (25 points)

In this problem, you will continue to use the Iris data and explore a KNN classifier using provided `knnClassify` python class. While doing the problem, please explore the implementation to become familiar with how it works.

First, we will shuffle and split the data into training and validation subsets:

```
iris = np.genfromtxt("data/iris.txt",delimiter=None)  # load the data
Y = iris[:,-1]
X = iris[:,0:-1]

# Note: indexing with ":" indicates all values (in this case, all rows);
#  indexing with a value ("0", "1", "-1", etc.) extracts only that value (here, columns);
#  indexing rows/columns with a range ("1:-1") extracts any row/column in that range.

import mltools as ml
# We'll use some data manipulation routines in the provided class code
# Make sure the "mltools" directory is in a directory on your Python path, e.g.,
#   export PYTHONPATH=$\$${PYTHONPATH}:/path/to/parent/dir
# or add it to your path inside Python:
#   import sys
#   sys.path.append('/path/to/parent/dir');

X,Y = ml.shuffleData(X,Y);                  # shuffle data randomly
# (This is a good idea in case your data are ordered in some pathological way,
#   as the Iris data are)

Xtr,Xva,Ytr,Yva = ml.splitData(X,Y, 0.8);  # split data into 80/20 train/validation
```

You may also find it useful to set the random number seed at the beginning (in general, for every assignment), e.g., `numpy.random.seed(0)`, to ensure consistent behavior each time.

Learner Objects Our learners (the parameterized functions that do the prediction) will be defined as python objects, derived from either an abstract classifier or abstract regressor class. The abstract base classes have a few useful functions, such as computing error rates or other measures of quality. More importantly, the learners will all follow a generic behavioral pattern, allowing us to train the function on a data set (i.e., set the parameters of the model to perform well on those data), and make predictions on a data set.

You can build now and *train* a kNN classifier on `Xtr,Ytr` and make predictions on some data `Xva` with it:

```
knn = ml.knn.knnClassify()    # create the object and train it
knn.train(Xtr, Ytr, K)      # where K is an integer, e.g. 1 for nearest neighbor
   prediction
YvaHat = knn.predict(Xva)   # get estimates of y for each data point in Xva

# Alternatively, the constructor provides a shortcut to "train":
knn = ml.knn.knnClassify( Xtr, Ytr, K );
YvaHat = predict( knn, Xva );
```

If your data are 2D, you can visualize a data set and a classifier's decision regions using e.g.,

```
ml.plotClassify2D( knn, Xtr, Ytr );  # make 2D classification plot with data (Xtr,Ytr)
```

This function plots the training data and colored points as per their labels, then calls `knn`'s `predict` function on a densely spaced grid of points in the 2D space, and uses this to produce the background color. Calling the function with `knn=None` will plot only the data.

1. Modify the code listed above to use only the first two features of X (e.g., let X be only the first two columns of `iris`, instead of the first four), and visualize (plot) the classification boundary for varying values of $K = [1, 5, 10, 50]$ using `plotClassify2D`. By comparing the plots, summarize the pros and cons of increasing the value of K in a kNN classifier. *(10 points)*

2. Again using only the first two features, compute the error rate (number of misclassifications) on both the training and validation data as a function of $K = [1, 2, 5, 10, 50, 100, 200]$. You can do this most easily with a for-loop:

   ```
   K=[1,2,5,10,50,100,200];
   for i,k in enumerate(K):
       learner = ml.knn.knnClassify(...# TODO: complete code to train model
       Yhat = learner.predict(... # TODO: predict results on training data
       errTrain[i] = ...       # TODO: count what fraction of predictions are wrong
       #TODO: repeat prediction / error evaluation for validation data
   plt.semilogx(...  #TODO: average and plot results on semi-log scale
   ```

   Plot the resulting error rate functions using a semi-log plot (`semilogx`), with training error in red and validation error in green. Based on these plots, what value of K would you recommend? *(10 points)*

3. Provide the same plots as the previous, but with all the features in the dataset. Are the plots very different? Is your recommendation different? *(5 points)*

Problem 4: Conditional, Marginal, and Posterior Probabilities (25 points)

We consider a dataset involving 10 different emails. The dataset contains binary-valued features about each email, including whether we know the author or not, whether the email is long or short, and whether it has

any of several key words, along with our final decision about whether to read it or simply file it away instead ($y = +1$ for "read", $y = -1$ for "discard").

x_1 know author?	x_2 is long?	x_3 has 'research'	x_4 has 'grade'	x_5 has 'lottery'	y $\Rightarrow$ read?
0	0	1	1	0	-1
1	1	0	1	0	-1
0	1	1	1	1	-1
1	1	1	1	0	-1
0	1	0	0	0	-1
1	0	1	1	1	1
0	0	1	0	0	1
1	0	0	0	0	1
1	0	1	1	0	1
1	1	1	1	1	-1

Although the problem relates to "Bayes Classifiers" for categorizing emails discussed in the course, at this point we don't care: all we need in this problem is to understand and compute conditional probabilities, marginal probabilities, and to apply Bayes rule. Include all steps you took to arrive at the answer.

1. Using the dataset above, estimate the class probabilities $p(y)$ for $y \in \{+1, -1\}$ and all the individual feature probabilities $p(x_i|y)$, for each class y and feature $x_i \in \{0, 1\}$ *(10 points)*

2. As follows, we define a joint distribution of our data as $p(\underline{\mathrm{x}}|y) \equiv \prod_{i=1}^{5} p(x_i|y)$ and $p(x, y) \equiv p(\underline{\mathrm{x}}|y)p(y)$, i.e., using the probabilities estimated in part 1. We want to use these probabilities to understand which combinations of features $\underline{\mathrm{x}}$ and labels y to expect in an email in order to decide to read it or not.

   For $\underline{\mathrm{x}} = (0\ 0\ 1\ 1\ 0)$, which of $p(\underline{\mathrm{x}}, y = 0)$ and $p(\underline{\mathrm{x}}, y = 1)$ is higher? What about for $\underline{\mathrm{x}} = (1\ 1\ 1\ 1\ 0)$? *(10 points)*

3. Compute the posterior probability that $y = +1$ given the observation $\underline{\mathrm{x}} = (0\ 0\ 1\ 1\ 0)$. *Hint: To this end, you will need to compute the marginal probability* $p(\underline{x}) = p(\underline{x}, y = 0) + p(\underline{x}, y = 1)$. *(5 points)*

Statement of Collaboration (5 points)

It is **mandatory** to include a *Statement of Collaboration* in each submission, with respect to the guidelines below. Include the names of everyone involved in the discussions (especially in-person ones), and what was discussed.

All students are required to follow the academic honesty guidelines posted on the course website. For programming assignments, in particular, I encourage the students to organize (perhaps using Piazza) to discuss the task descriptions, requirements, bugs in my code, and the relevant technical content *before* they

start working on it. However, you should not discuss the specific solutions, and, as a guiding principle, you are not allowed to take anything written or drawn away from these discussions (i.e. no photographs of the blackboard, written notes, referring to Piazza, etc.). Especially *after* you have started working on the assignment, try to restrict the discussion to Piazza as much as possible, so that there is no doubt as to the extent of your collaboration.