Apr 14 Computer Lab and Group Write up

The overall goal of this week's work is to get you on the computer and trying out the algorithms we've discussed so far (Monday, today).

You'll also summarize this work by writing up your solutions in a short document (an example will be given on the class website). You may use any word processing software to write this one up- whatever you're comfortable using. I do want you to use all the sections that I've put in my document.

- Document 1: $k-$nearest neighbors on the iris data set.

  You can look up information on Fisher's iris data set if you like to fill out the background for your paper. In summary, there are 150 points (50 for each flower), and each point has 4 observations. There is also a vector with the flower labels (these are the "targets", labeled as 1, 2, or 3).

  With your group, and using the "helper" functions described on the next page, (i) do any data preprocessing and/or visualization first, (ii) build the classifier, and (iii) report your results as a confusion matrix. In this case, if you follow along the template report, you'll do great.

- Document 2: $k-$nearest neighbors for regression.

  Once you've got your script files all written for the classifier, it will be straightforward to modify them for regression (we'll talk about that today (Wed)).

  Go through the same process as before, but now use regression. In performing the $k-$fold cross validation on how many neighbors to use, we'll need a different error measurement- the "coefficient of determination", or R^2 term.

  The way we define it is straightforward: Let vector $\mathbf{t}$ be your targets, and $\mathbf{y}$ be the model output. Then:

$$\begin{aligned} \bar{t} &= \texttt{mean}(t) \\ \text{SStot} &= \texttt{sum}((t-\bar{t})^2) \qquad \text{(Elementwise square)} \\ \text{SSres} &= \texttt{sum}((t-y)^2) \qquad \text{(Elementwise square)} \\ R^2 &= 1 - \text{SSres}/\text{SStot} \end{aligned}$$

DUE DATE: Wednesday, Apr 21.

Helper Functions for the k-NN HW

I've written some "helper functions" in Matlab to assist with building the k−nearest neighbor script. All of the Matlab functions are available for download as a single zip file on our class website. The Python versions already exist in `sklearn`, so here is a list of them, with some examples.

1. Build a confusion matrix. For Python, check to see how error (or score) is output.

   Matlab: `conf_matrix.m`   Python: `sklearn.metrics.confusion_matrix`

2. Takes data in and outputs the predicted classes using k−NN.

   Matlab: `fitknn.m`   Python: `sklearn.neighbors.kNeighborsClassifier`

   We won't be using too many options here. Be sure to look up how to extract the error on a test set.

3. Sample classification data. The data consists of 150 flowers, each with 4 measurements. Classify as to the type of flower (actually, iris). It is widely available on the net as well as available for us as a mat file and a built in file in Python.

   Matlab: `irisdata.mat`

   ```
   from sklearn import datasets
   iris = datasets.load_iris()
   X=iris.data
   targets=iris.target
   ```

4. Splits a given data set into k-folds, ready for cross validation. To the right is a full example using Python (`Kfold` is the command).

   Matlab: `KfoldCV.m`

   ```
   import numpy as np
   from sklearn.model_selection import KFold
   X = np.array([[1, 2], [3, 4], [1, 2], [3, 4]])
   y = np.array([1, 2, 3, 4])
   kf = KFold(n_splits=2,shuffle=True)
   kf.get_n_splits(X)

   for train_index, test_index in kf.split(X):
       print("TRAIN:", train_index, "TEST:", test_index)
       X_train, X_test = X[train_index], X[test_index]
       y_train, y_test = y[train_index], y[test_index]
   ```

5. StandardScaler.m: Scales the data by subtracting the mean and dividing by the standard deviation.

   Matlab: `StandardScaler.m`

   Python: StandardScaler is a class in `sklearn.preprocessing`. See the documentation.

6. Splits a given data set randomly into a training and testing sets (give it a percentage for test, like 0.3).

   Matlab: `TrainTestSplit.m`

   Example in Python:

   ```
   from sklearn.model_selection import train_test_split
   X, y = np.arange(10).reshape((5, 2)), range(5)
   X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
   ```

Pima Diabetes Study - Sample Write Up

Your Name(s) Here

April 13, 2021

Abstract

We analyzed the Pima Diabetes data using a k-nearest neighbor classifier. We scaled the data, visualized it using Principal Components Analysis, and then determined an optimal value for the number of nearest neighbors using a 5-fold cross validation study. Finally, we built our classifier and reported the results. We used 23 nearest neighbors, and found an accuracy of 73% using our technique.

1 Problem Discussion

In the problem discussion, give any background that might be relevant (if you're given any, of course). Give the reader an idea of what the problem is that you're trying to solve. Here's an example (from Kaggle).

This dataset is originally from a study done in 1988 [1]. The objective of the dataset is to diagnostically predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset. Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.

In the next section, we'll give an overview of the data, and look at any scaling protocols that may be necessary. In the section after that, we'll build an appropriate nearest neighbor classifier, then finally we'll take a look at the results.

2 The Data

The data consists of 8 possible variables and one outcome value for a total of 9 columns. There are 768 total records, so the original data matrix is 8×768. Before we visualize the data, the 8 variables are:

1. Number of pregnancies.
2. Plasma glucose concentration
3. Diastolic blood pressure
4. Triceps skin fold thickness
5. Two hour serum insulin
6. BMI
7. Diabetes Pedigree Function
8. Age

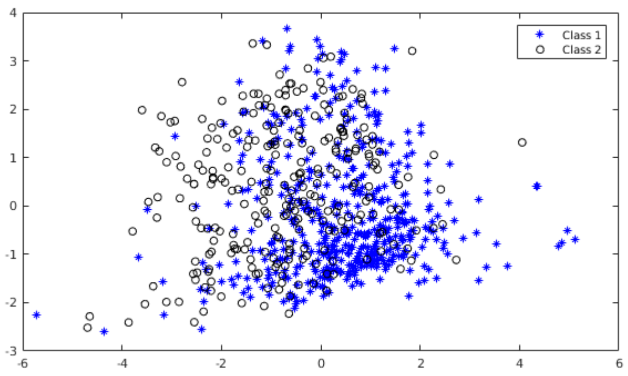
The mean and standard deviation of the 8 variables are given in the table below.

	Preg	Gluc	BP	Skin	Insulin	BMI	Ped	Age
mean	3.85	120.89	69.11	20.54	79.80	31.99	0.47	33.24
std	3.37	31.97	19.36	15.95	115.24	7.88	0.33	11.76

There was enough differences in the scales that we decided to perform a standard scaling across all variables to make each have zero mean and unit standard deviation.

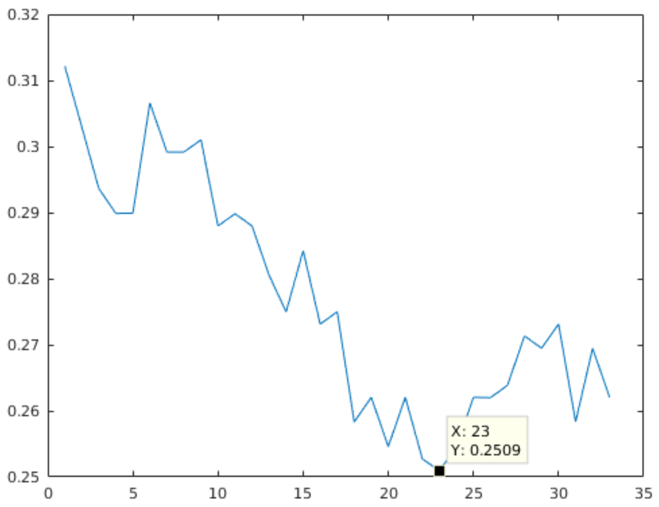
To visualize the two classes, we performed a Principal Components Analysis on the data, with the results shown graphically. The blue asterisks are class 1, and the black circles are class 2. Also, the first six singular values were somewhat close, from 0.11 to 0.19, with the next lowest at 0.08, suggesting that perhaps the rank is close to 6.

Before going into the classification stage, we reserved 30% of the data (233 points) chosen randomly, for the test set, and that left us with 535 points for training.



3 Construction of the Classifier

We performed a 5-fold cross validation study to determine the best number of neighbors. Repeating the process several times led us to determine an approximate number of neighbors, 23. The graph below shows the error rate for each choice of neighbors, from 3 to 35.



Finally, we used the full training set to classify the test set that was reserved initially.

4 Results

The results of the classification are summarized below in the confusion matrix. The predicted classes are shown in rows, the actual classes in the columns.

We did not have an equal class representation, as we see that 66% (152/230) of our sample population was in class 1. Of the actual class 1 samples, we were able to predict that 87.5% accuracy, and in class 2, 45% (78/230). We can see that predicting class 1 was better than class 2, perhaps because class 1 had more separation.
The overall accuracy was 73%.

	1	2	
1	133	43	176
2	19	35	54
	152	78	230

I'm including several references not cited just so you can see some options. The only references that should be included below are ones that are cited in the text.

References

[1] Smith, J.W., Everhart, J.E., Dickson, W.C., Knowler, W.C., & Johannes, R.S. (1988). Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. *In Proceedings of the Symposium on Computer Applications and Medical Care* (pp. 261–265). IEEE Computer Society Press.

[2] The Mathworks Inc. (n.d.). Image types in the toolbox. In Image processing toolbox. Retrieved at http://www.mathworks.com/access/helpdesk/help/toolbox/images/f14-13543.html.

[3] D. C. Lay, *Linear algebra and its applications.* Addison Wesley, New York, NY 2003.

5 Appendix: Code

```
Put the main part of your code here.

In Matlab, you don't need to include the 6 files that
I provided in the zip folder, but do provide your "driver"
or script file.
```