Using datasets to predict

Part 1

Background

These lab exercises use datasets containing information about three different varieties of iris flower: setosa, versicolor & virginica. One set of data (from which you've seen a sample in Lab 03), FisherIrisDatasetValues.csv & FisherIrisDataset-Labels.csv, created by Sir Ronald Aylmer Fisher from measurements of actual plants will be used as a basis for predicting the variety of new sample plants by comparing the measurements of the new sample with the verified data from the dataset. File FisherIrisDatasetValues.csv contains 120 flowers (i.e. rows) each containing four numeric measurements (i.e. columns): sepal length, sepal width, petal length and petal width. The flower type of each of these flowers is recorded in file FisherIris-DatasetLabels.csv.

The other set of data, OtherIrisSamplesValues.csv & OtherIrisSamplesLabels.csv, contain information on 30 more flowers and will be used as a test of the reliability of the prediction system that you will build in the first exercise.

Instructions

In order to compare the measurements of say, the petal length of the new sample, to those in the database, you need to calculate a similarity measure. A common way to do this is to calculate the absolute difference between the sample and the verified data for each of the measurements and compute a total difference measurement. Rather than just computing the raw total difference between the measurements, it is more accurate to compute each of the differences as a fraction of the range of values for each measurement. This distance measure is known as the Manhattan Distance (a.k.a. City-block distance). For example, if the length of the petal was two inches longer than one sample in the dataset, the contribution of two inches to the total difference might out-weigh a difference of one half inch in the petal width, whereas maybe the difference in the petal width is proportionally much more of a difference in the typical plant measurements. The entries from the dataset that have the smallest difference from the sample are then used to predict the iris type of the sample. This algorithm, called the kNN algorithm, or k-nearest neighbors, can be applied to many different types of problems from predicting your credit score to predicting the type of cancer in a patient. Use the algorithm below to develop a program, (MATLAB script), called Lab9Part1_KNN.m that will get four measurements of a single iris plant from the user, and from those measurements, predict the iris variety of that plant.

1. Read the data in files FisherIrisDatasetValues.csv & FisherIrisDatasetLabels.csv into a matrix called dataValues and a column vector called dataLabels, respectively:

```
dataValues =  impordata('FisherIrisDatasetValues.csv',',')
dataLabels=  impordata('FisherIrisDatasetLabels.csv',',')
```

   Now matrix dataValues contains the sepal length, sepal width, petal length and petal width measurements for 120 iris flowers and column vector dataLabels contains the corresponding flower type.

2. For clarity, assign each of the dataValues matrix columns to a column vector with meaningful names:

```
sepalLength = dataValues(:,1);
```

3. Determine and display (in a nice message) the range of values for each of the four measurements of the iris plants and assign each a variable name:

```
sepalLengthRange=max(sepalLength)-min(sepalLength);
```

4. Use the input MATLAB function to ask the user to input sepal length, sepal width, petal length and petal width values for their new iris sample and assign those values to single a vector called newSample.
5. Create 4 column vectors to hold the differences between the measurements from each of the elements of the dataset and the four measurements in vector newSample.

```
sepalLengthDifferences = abs((sepalLength - sepalLengthFornewSample)/ sepalLengthRange)
```

6. Add up the four column vectors to get a totalDifferences vector
7. Save a version of the totalDifferences vector sorted by closest (i.e. smallest) differences (ascending order). Use the sort MATLAB function for this purpose.
8. Do the same sort again, and use the feature of sort that allows you to save the indices of the original version of the vector. For example, if V is a vector, then [sortedV,indexV] = sort(V) would sort V and store it in vector sortedV and would create an index vector called indexV that contains the old location (i.e. in V) of each of the items in sortedV. Try the following in MATLAB before you proceed:

```
V = [ 8; 2; 5; 10; -1]
[sortedV,indexV] = sort(V)
%Now we can use indexV to easily rearrange any other vector having the same
%dimensions as V, in V's order. Try the following example.
Z = [ 1; 2; 3; 4; 5]
Z(indexV,:)
```

9. Use the indexV vector from step #8 to create a vector of the iris types column vector (i.e., dataLabels) sorted by closest differences.

```
dataLabelsSorted = dataLabels(indexFromStep8,:);
```

10. Ask the user to input an integer, k, that will be used to choose the k closest matches to your sample
11. Create another vector from the sortedIrisTypes that holds just the first k iris types and then use a for-loop to print a nicely labeled table of the top k differences and the associated iris type.
12. Calculate the majority type in the top k and display your prediction. PS: you need to find out the count of each type of flower which exits in the top k. The type associated with the largest count would be your final prediction.

Part 2

Assuming you have your prediction system working from part 1, make a copy of it called Lab9Part2_predictionAccuracy.m, and then modify it as needed so that your program will read in the file OtherIrisSampleValues.csv and use the measurements contained in it to predict the iris type for each sample. In other words, now the user doesn't input information for any samples; instead, you need to repeat your predictions for all samples in files OtherIrisSampleValues.csv & OtherIrisSampleLabels.csv into a matrix and a column vector, respectively. Note that file OtherIrisSampleLabels.csv already contains the answer as to

what type of iris the measurements are taken from, but the purpose of this exercise is to see how accurately the prediction system is that you built in part 1. Your program should use a loop to compute a predicted iris type for each of the 30 samples and then report what percentage of those samples you predicted accurately.

Part 3

Run your Lab9Part2_predictionAccuracy.m program with several different values of k and report what you find.

Based on Lab09 from CSCI 140.

- The problem will be the following:
  - You are given a file with a user ratings for X number of movies, no missing entires.
  - Your job is, given the ratings for X-1 movies for a user, what do you predict that they will rate the movie with no ratingl
  - To do this, you will find the user who is most similar to the user for the movies that do have ratings, then use their rating for the missing movie as the rating for the query user.
  - https://grouplens.org/datasets/movielens/
    - Use one of these datasets, choose columns and users to create a matrix with no empty entries.
  - https://www.kaggle.com/c/predict-movie-ratings