Machine Learning

Assignment 4 - Multi-Class Classification
Summer 2023

Introduction

In this assignment you will implement two multi-class classifiers: *K-Nearest Neighbors* and *Decision Trees*, and evaluate their classification performance on multiple datasets.

You may **not** use any functions from a ML library in your code. And as always your code should work on any dataset that has the same general form as the provided one.

Grading

Part 1 (Theory)	25pts
Part 2 (KNN)	30pts
Part 3 (DT)	30pts
Part 4 (Additional Dataset)	15pts
TOTAL	100pts

Datasets

Cartiotocography Dataset (CTG.csv) Download the file CTG.csv from Bblearn. This file contains 2126 instances of 21 feature pertaining to information obtained from Cardiotocography tests. Our task is to determine the fetal state class code given an observation. This code can be one of the 3 values and pertains to the LAST column of the dataset. The second to last column of the dataset can also be used for classification but for our purposes DISCARD it. That is, discard the column named **CLASS**, and use the column named **NSP** as our target column.

You can read more about the dataset here:

http://archive.ics.uci.edu/ml/datasets/Cardiotocography

Yale Faces Datasaet This dataset consists of 154 images (each of which is 243x320 pixels) taken from 14 people at 11 different viewing conditions (for our purposes, the first person was removed from the official dataset so person ID=2 is the first person).

The filename of each images encode class information:

$$\text{subject}< ID >.< condition >$$

Data obtained from: http://cvc.cs.yale.edu/cvc/projects/yalefaces/yalefaces.html

1 Theory

1. Consider the following set of training examples for an unknown target function: $(x_1, x_2) \rightarrow y$:

Y	x_1	x_2	Count
+	T	T	3
+	T	F	4
+	F	T	4
+	F	F	1
-	T	T	0
-	T	F	1
-	F	T	3
-	F	F	5

(a) What is the sample entropy for the class label overall, $H(Y)$ from this training data (using log base 2) (3pts)?

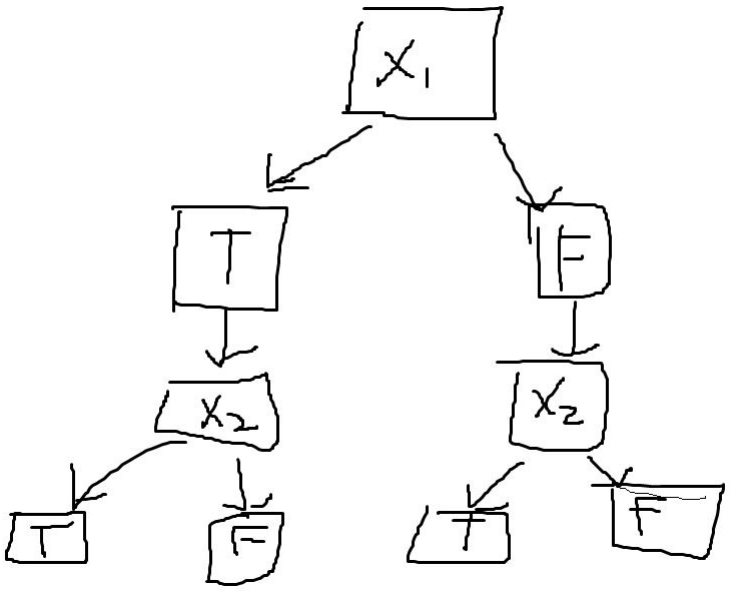
$$H(+,-) = (-\frac{12}{21} log_2 \frac{12}{21}) + (-\frac{9}{21} log_2 \frac{9}{21}) = 0.985$$

(b) What are the weighed average entropies for branching on variables x_1 and x_2 (6pts)?

$$H(x_1) = \frac{8}{21}(H_{x_{1(T)}}) + \frac{13}{21}(H_{x_{1(F)}}) = 0.8$$

$$H(x_2) = \frac{10}{21}(H_{x_{2(T)}}) + \frac{11}{21}(H_{x_{2(F)}}) = 0.94$$

(c) Draw the decision tree that would be learned by the ID3 algorithm without pruning from this training data. You may use software to draw this or draw it by hand. But either way the figure should be embedded in your PDF submission. (6pts)



(d) Computer the posteriors for the observation $x = [T, T]$ using:

i. Inference (5pts)

$$P(Y = +|x = [T, T]) = \frac{(\frac{3}{7})^2 * \frac{7}{16}}{\frac{63}{784}} = 0.44$$

ii. Naive Bayes (5pts)

$$\frac{7}{16} \cdot (\frac{3}{7})^2 = 0.11$$

2 K-Nearest Neighbors Classifier

Let's train and validate a *K-Nearest Neighbors Classifier*.

Write a *function*, called *myKNN*, should return a vector of predictions for the validation data, and take in as parameters:

1. The observable training data as a matrix X_{train}.
2. The training data's enumerated target classes, as a column vector Y_{train}.
3. The observable validation data as a matrix X_{valid}.
4. An interger k where $k \geq 1$

We now want to use this function to train and evaluate a k-nearest neighbors classifier using the *cartiotograph* dataset. Write a *script* that:

1. Reads in the data, discarding the first two rows, and obtains the observable data matrix X using columns **LB** through **Tendency**. Discard the column **CLASS** and use the last column, **NSP** for your target values Y. Perform any pre-processing of the observable data that you find fit.
2. Shuffles the observations.
3. Selects the first 2/3 (round up) of the data for training and the remaining for validation.
4. Sends this data to your function to classify the validation data using the training data.
5. Computes the validation data's accuracy and generates a confusion matrix.

Try this out for at least 3 different values of k, computing the validation accuracy for each, and putting these findings in a table in your report:

k	validation accuracy
2	0.8997
9	0.9025
12	0.9040

Confusion matrix:

0	0	0
0	555	37
0	6	46

And finally, provide a confusion matrix for your best performing classifier. You must implement this yourself (you cannot use a python-related function to do this for you). Your confusion matrix will have the target classes on one axis, and the estimated classes on the other. In each cell just report the number of times that combination occurs.

In your report you will need:

1. Description of any pre-processing of the observable data that you did.
2. The completed table, shown above.
3. The confusion matrix for your best performing classifier.

3 Decision Trees

Let's train and test a *Decision Tree*.

Everyone taking this class should have implemented at least a **binary search tree** at some point, which can be the starting point for you decision tree impementation. If you're rusty on how to do this, here's a resource:

`https://www.tutorialspoint.com/python_data_structure/python_binary_tree.htm`

.

You should again implement a function, this time called, *myDT*, that returns the predictions for the validation set and takes in as parameters the training and validation datsets as mentioned in the prior section. However this time you don't need a parameter k now!

Now create a script that utilizes this function for the cartiograph dataset. This main script should:

1. Read in the data as specified in the previuos part.
2. Shuffle the observations
3. Select the first 2/3 (round up) of the data for training and the remaining for validation.
4. Pre-process the observable data, as appropriate. Since Decision Trees normally expect categorical features, you'll need to convert any continuous ones to categorical. For simplicity, convert these to *binary* features using the mean or median of the *training* data.
5. Send this data to your function to train your system using the training data and classify the validation data.
6. Compuate the validation data's accuracy and generate a confusion matrix.

Implementation Details

1. Seed the random number generator for reproducability.

In your report you will need:

1. Description of any additional pre-processing of the dataset you did.
2. The validation accuracy of your system.
   Answer: 0.863
3. Your confusion matrix.

0	0	0
0	543	43
0	12	38

4 Additional Dataset

Now let's see how your systems work on another multi-class dataset, the Yale Faces dataset!
Pre-process your data just like you did in HW1, however now you will also be populating your enumerated class targets based on the subject encoded in the file name.:

1. Read in the list of files
2. Create a 154×1600 data matrix X and a 154×1 target matrix Y such that for each image file
   (a) Read in the image as a 2D array (243x320 pixels)
   (b) Subsample/resize the image to become a 40x40 pixel image (for processing speed). I suggest you use your image processing library to do this for you.
   (c) *Flatten* the image to a 1D array (1x1600)
   (d) Concatenate this as a row of your data matrix and use the file name to encode a class ID and append that to your target matrix Y.

Once you've created your observable matrix X and its target column vector Y, create your training and validation sets and train and validate your *KNN* and *Decision Tree* systems, reporting the validation accuracy and creating confusion matrices. **HOWEVER**, since there's only a few observations per class (person), for each *person* shuffle their data and select 2/3 for training and 1/3 for validation. This will ensure that we get at least a few observations for each class in both our training and validation sets.

Submission

For your submission, upload to Blackboard a single zip file containing:

1. PDF Writeup
2. Source Code
3. readme.txt file

The readme.txt file should contain information on how to run your code to reproduce results for each part of the assignment.

The PDF document should contain the following:

1. Part 1: Answer to theory questions
2. Part 2:
   (a) Description of any pre-processing of the observable data that you did.
   (b) Table of validation accuracies for different values of k.
   (c) Confusion matrix.
3. Part 3:
   (a) Description of any pre-processing of the observable data that you did.
   (b) Validation accuracy and confusion matrix.
4. Part 4:
   (a) Description of any pre-processing of the observable data that you did.
   (b) Validation accuracies and confusion matrices for both your KNN and DT algorithms.