Machine Learning

Lab 6 - Nearest Neighbors
Spring 2025

Introduction

In this lab you will implement a *Nearest Neighbors* algorithm, 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

- +1pt Can run script properly.
- +2pt Parses datasets correctly.
- +1pt Generates requested info.
- +1pt Better than random results for Part 1
- +2pt Good results for Part 1 ($Accuracy \approx 90\%$)
- +1pt Better than random results for Part 2
- +2pt Good results for Part 2 ($Accuracy \approx 66\%$)

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 Nearest Neighbor

Create a script that performs classification using *k-nearest neighbors* on the Cartiotocography dataset (target predicting NSP, discard CLASS) . This script should:

1. Read in the data as specified in the previous 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.
5. Use the training data to classify each validation sample. **At the top of your script have a parameter K that specifies how many nearest neighbors to use**. Feel free to play with pre-processing data and choice of similarity or distance function.
6. Compuate the validation data's accuracy and generate a confusion matrix.

Implementation Details

1. Seed the random number generator for reproducability.

Your script should output:

1. The class priors
2. The validation accuracy of your system.
3. Your confusion matrix (either as a matrix, output to the screen, or a figure).

2 Additional Dataset

Now let's see how your systems work on another multi-class dataset, the Yale Faces dataset!
Recall that we already used this dataset in previous labs. However now you will also be populating your enumerated class targets based on the subject encoded in the file name. Creating your observable and target matrices is as follows:

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* system, 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.

Your script should output:

1. The class priors
2. The validation accuracy of your system.
3. Your confusion matrix (either as a matrix, output to the screen, or a figure).