COMP 4983: Lab Exercise #8 Mark: /40

[Due: Nov 4, 2022 @2359
Assignment Submission
Folders]

Instructions:

In this lab, you will

- compute the average leave-one-out cross-validation (LOOCV) estimate of prediction error on paper for k-Nearest Neighbors (k-NN) for a trivial dataset
- implement k-NN in Python without the use of library functions and verify your k-NN implementation using the KNeighborClassifier.fit() and KNeighborClassifier.predict() functions from sklearn.neighbors
- use k-NN to classify handwritten digits as well as determine the best value of k using 10-fold cross-validation

Part 1: LOOCV for k-NN (on paper)

In this part of the lab, you will compute the average LOOCV estimate of prediction error on paper for 3-NN using Euclidean distance as the distance metric on a training set consisting of the following five (5) training samples:

Sample	Input Vector	Output Value
x_1	(1, 1)	0
x_2	(2, 3)	0
x_3	(3, 2)	0
x_4	(3, 4)	1
x_5	(2, 5)	1

In each trial of cross-validation (as you compute the average LOOCV estimate of prediction error), clearly indicate the

- samples in the training set and the validation set
- k nearest neighbors for each sample in the validation set
- results of majority vote
- cross-validation estimate of prediction error

Part 2: *k*-NN Implementation

[20 marks] In this part of the lab, you will first implement *k*-NN for multiclass classification in Python without the use of the scikit-learn package. Replace ??? in kNN() as defined below with appropriate code (using Euclidean distance as the distance metric):

```
# k-Nearest Neighbors
# Inputs:
#  x: a test sample
#  Xtrain: array of input vectors of the training set
#  Ytrain: array of output values of the training set
#  k: number of nearest neighbors
# Outputs:
#  y_pred: predicted value for the test sample
def kNN(x, Xtrain, Ytrain, k):
  ???
return y_pred
```

Thereafter, you will verify your kNN() implementation using the KNeighborClassifier.fit() and KNeighborClassifier.predict() functions from sklearn.neighbors.

Steps:

1) Create a new Python script using the filename *kNN_lab8.py* and save it in your working directory.
2) Implement kNN() without the use of library functions.
3) Compute the average LOOCV estimate of prediction error for 3-NN using kNN() on the five (5) training samples provided in Part 1. You may use this output to verify your calculation from Part 1.
4) Compute the average LOOCV estimate of prediction error for 3-NN using the KNeighborClassifier.fit() and KNeighborClassifier.predict() functions from sklearn.neighbors on the five (5) training samples provided in Part 1 and verify your kNN() implementation.

Part 3: Classification of Handwritten Digits

[20 marks] In this part of the lab, you will

- apply the k-NN classifier from Part 2 to classify images of handwritten digits, 0-9, from the MNIST dataset
- determine the best value of k using 10-fold cross-validation on the training set
- evaluate the error rate (percentage of misclassifications) of the k-NN classifier on the test set

The MNIST dataset is one of the most well-studied dataset in machine learning. It consists of a training set of 60000 samples and a test set of 10000 samples. We will use 1200 samples from the MNIST dataset.

In the MNIST dataset, each sample is an 8 pixel by 8 pixel grayscale image of a handwritten digit, 0-9, reshaped into a row vector of 64 elements, where each pixel value is an integer ranging between 0 (white) and 16 (black).

Steps:

1) Download the handwritten digit dataset, *data_lab8.csv*, from BCIT Learning Hub (Content | Laboratory Material | Lab 8) and save it in your working directory. The dataset, *data_lab8.csv*, contains 1,201 rows (including a header row) and 65 columns. Each row of 65 columns contains the 64 pixels of a handwritten digit image, followed by the digit, 0-9, that this image corresponds to.
2) Add to your script, *kNN_lab8.py*, to read from *data_lab8.csv*.
3) Split the dataset into training and test sets, with the first 75% of the dataset for training and the remaining 25% for testing.
4) For each $k = [1,3,5,..,21]$, apply the k-NN classifier from Part 2 on the training set and evaluate the average cross-validation estimate of prediction error using 10-fold cross-validation. Plot the average cross-validation estimate of prediction error as a function of k. Include in your plot, a terse descriptive title, x-axis label, y-axis label and a legend. You may use **KNeighborClassifier.fit()** and **KNeighborClassifier.predict()** from **sklearn. neighbors**.
5) Determine the best value of k from Step (4).
6) Using the best value of k, evaluate and output the error rate (percentage of misclassifications) on the test set.

Deliverable:

All work submitted is subject to the standards of conduct as specified in BCIT Policy 5104. Late submissions will not be accepted.

[Nov 4, 2022 @2359] Ensure that your source code for Parts 2 and 3 are adequately commented and submit using the filename *kNN_lab8.py* to BCIT Learning Hub (Laboratory Submission | Lab 8).