silw_HW2_KNN_stu_25

February 23, 2025

1 Homework 2: Implement the K Nearest Neighbors (KNN) Algorithm to detect breast cancer

2 Reference: Lecture 8

In Lecture 8, we implemented a KNN algorithm for classification which computed a `simple majority vote` of the nearest neighbors of each input sample. The breast cancer data set from Homework 1 will be used in this homework. You will apply KNN to classify new samples on the test set into two categories (0: malignant, 1: benign).

Task 1: Split the dataset

Task 2: Implement the KNN for regession

Task 3: Evaluate

Task 4: Explore the impact of different K

Load the diabetes dataset

[2]:
```
import numpy as np
from sklearn import datasets as ds
from sklearn.model_selection import train_test_split

# 1. Data preparation
breast_ds = ds.load_breast_cancer()
X = breast_ds.data #feature vectors
y = breast_ds.target #target values

print(breast_ds['DESCR'])
```

```
.. _breast_cancer_dataset:

Breast cancer wisconsin (diagnostic) dataset
--------------------------------------------

**Data Set Characteristics:**

:Number of Instances: 569

:Number of Attributes: 30 numeric, predictive attributes and the class
```

:Attribute Information:
    - radius (mean of distances from center to points on the perimeter)
    - texture (standard deviation of gray-scale values)
    - perimeter
    - area
    - smoothness (local variation in radius lengths)
    - compactness (perimeter^2 / area - 1.0)
    - concavity (severity of concave portions of the contour)
    - concave points (number of concave portions of the contour)
    - symmetry
    - fractal dimension ("coastline approximation" - 1)

    The mean, standard error, and "worst" or largest (mean of the three
    worst/largest values) of these features were computed for each image,
    resulting in 30 features. For instance, field 0 is Mean Radius, field
    10 is Radius SE, field 20 is Worst Radius.

    - class:
            - WDBC-Malignant
            - WDBC-Benign

:Summary Statistics:

	Min	Max
radius (mean):	6.981	28.11
texture (mean):	9.71	39.28
perimeter (mean):	43.79	188.5
area (mean):	143.5	2501.0
smoothness (mean):	0.053	0.163
compactness (mean):	0.019	0.345
concavity (mean):	0.0	0.427
concave points (mean):	0.0	0.201
symmetry (mean):	0.106	0.304
fractal dimension (mean):	0.05	0.097
radius (standard error):	0.112	2.873
texture (standard error):	0.36	4.885
perimeter (standard error):	0.757	21.98
area (standard error):	6.802	542.2
smoothness (standard error):	0.002	0.031
compactness (standard error):	0.002	0.135
concavity (standard error):	0.0	0.396
concave points (standard error):	0.0	0.053
symmetry (standard error):	0.008	0.079
fractal dimension (standard error):	0.001	0.03
radius (worst):	7.93	36.04

```
texture (worst):                        12.02  49.54
perimeter (worst):                      50.41  251.2
area (worst):                           185.2  4254.0
smoothness (worst):                     0.071  0.223
compactness (worst):                    0.027  1.058
concavity (worst):                      0.0    1.252
concave points (worst):                 0.0    0.291
symmetry (worst):                       0.156  0.664
fractal dimension (worst):              0.055  0.208
====================================== ====== ======

:Missing Attribute Values: None

:Class Distribution: 212 - Malignant, 357 - Benign

:Creator:  Dr. William H. Wolberg, W. Nick Street, Olvi L. Mangasarian

:Donor: Nick Street

:Date: November, 1995

This is a copy of UCI ML Breast Cancer Wisconsin (Diagnostic) datasets.
https://goo.gl/U2Uwz2

Features are computed from a digitized image of a fine needle
aspirate (FNA) of a breast mass.  They describe
characteristics of the cell nuclei present in the image.

Separating plane described above was obtained using
Multisurface Method-Tree (MSM-T) [K. P. Bennett, "Decision Tree
Construction Via Linear Programming." Proceedings of the 4th
Midwest Artificial Intelligence and Cognitive Science Society,
pp. 97-101, 1992], a classification method which uses linear
programming to construct a decision tree.  Relevant features
were selected using an exhaustive search in the space of 1-4
features and 1-3 separating planes.

The actual linear program used to obtain the separating plane
in the 3-dimensional space is that described in:
[K. P. Bennett and O. L. Mangasarian: "Robust Linear
Programming Discrimination of Two Linearly Inseparable Sets",
Optimization Methods and Software 1, 1992, 23-34].

This database is also available through the UW CS ftp server:

ftp ftp.cs.wisc.edu
cd math-prog/cpo-dataset/machine-learn/WDBC/
```

```
.. dropdown:: References

  - W.N. Street, W.H. Wolberg and O.L. Mangasarian. Nuclear feature extraction
    for breast tumor diagnosis. IS&T/SPIE 1993 International Symposium on
    Electronic Imaging: Science and Technology, volume 1905, pages 861-870,
    San Jose, CA, 1993.
  - O.L. Mangasarian, W.N. Street and W.H. Wolberg. Breast cancer diagnosis and
    prognosis via linear programming. Operations Research, 43(4), pages 570-577,
    July-August 1995.
  - W.H. Wolberg, W.N. Street, and O.L. Mangasarian. Machine learning techniques
    to diagnose breast cancer from fine-needle aspirates. Cancer Letters 77
(1994)
    163-171.
```

2.0.1 Task 1: Split the dataset into training (75%) and test sets (25%). 5 points.

[3]:
```
print(X.shape, y.shape)

# 2. Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,␣
 ↪random_state=42)

print("Trainig data shape: ", X_train.shape, y_train.shape) # 426 (75% of 569)␣
 ↪samples for training
print("Testing data shape: ", X_test.shape, y_test.shape) # 143 (25% of 569)␣
 ↪samples for testing
```

```
(569, 30) (569,)
Trainig data shape:  (426, 30) (426,)
Testing data shape:  (143, 30) (143,)
```

2.0.2 Task 2: Implement the KNN algorithm for classificaiton. 20 points.

- Find neighbors: for any input data sample, find its k nearest neighbors on the training set
- Prediction: the prediction function will predict the target value using the majority voting.

Task 2.1: Complete the following function to identify the K nearest neighbors of a given data sample.

- return both the indices and Euclidean distances of the K nearest neighbors for a given query/new sample.

[4]:
```
def findKNgbs(X_train, x_query, K = 5):
    '''find K nearest neighbors for a given data sample in X_train

        input:
            X_train: training set
```

```
            x_query: new data sample/a query
            K: the number of neighbors
        return:
            indK: the indices of of the K nearest neighbors
            disK: the distances of the K nearest neighbors to x_new
    '''

    # This Euclidean distance is between x_query and all the samples in X_train
    distance = np.linalg.norm(X_train - x_query, axis=1)
    indK = np.argsort(distance)[:K] # indices of the K nearest neighbors
    disK = distance[indK] # corresponding distances of the K nearest
    return indK, disK

x = X_test[0]
print('Input data sample:\n', x)
k_inx, k_dis = findKNgbs(X_train, x, K=5)
print('K nearest neighbors of x in the training set:\n', k_inx, '\nTheir␣
 ↪distances to the query:\n',k_dis)
```

```
Input data sample:
 [1.247e+01 1.860e+01 8.109e+02 4.819e+02 9.965e-02 1.058e-01 8.005e-02
 3.821e-02 1.925e-01 6.373e-02 3.961e-01 1.044e+00 2.497e+00 3.029e+01
 6.953e-03 1.911e-02 2.701e-02 1.037e-02 1.782e-02 3.586e-03 1.497e+01
 2.464e+01 9.605e+01 6.779e+02 1.426e-01 2.378e-01 2.671e-01 1.015e-01
 3.014e-01 8.750e-02]
K nearest neighbors of x in the training set:
 [297 325 118 311 127]
Their distances to the query:
 [13.23427258 37.46529809 37.65164384 45.20300917 46.0377704 ]
```

Task 2.2: Predict the target value for new data samples.

[5]:
```
def predict(X_in, X_train, y_train, K=5):
    '''predict the target vlues for input queries
        Input:
            X_in: new data samples/queries. n*4. contains multiple data samples
            X_train: the feature vectors of training samples
            y_train: the target values of the training samples
            K: the number of neighbors

        return:
            y_pred: the predictions of the input queries
    '''

    y_pred = np.zeros(X_in.shape[0])
    for i, x in enumerate(X_in):
        k_inx, _ = findKNgbs(X_train, x, K)
        most_common = np.bincount(y_train[k_inx]).argmax()
```

```
        y_pred[i] = most_common

    return y_pred

n = 10 # 3 test samples
K = 6 # k neighbors

X_in = X_test[0:n]
y_pred = predict(X_in, X_train, y_train, K)
print('Predictions:', y_pred)
print('True target values:', y_test[:n])
```

```
Predictions: [1. 0. 0. 1. 1. 0. 0. 0. 1. 1.]
True target values: [1 0 0 1 1 0 0 0 1 1]
```

2.0.3 Task 3: Evaluate the KNN method. 10 points.

- Calculate the error rate of the KNN on the test set
- Output the confusion matrix on the prediction of the test set

[6]:
```
# calcualte the error rate.
err = np.mean(y_pred != y_test[:n])
print(round(err*100,3), '%')
```

```
0.0 %
```

[7]:
```
# output the confusion matrix
from sklearn.metrics import confusion_matrix
m = confusion_matrix(y_test[:n], y_pred)
print(m)
```

```
[[5 0]
 [0 5]]
```

2.0.4 Task 4: Explore the impact of different K. 10 points.

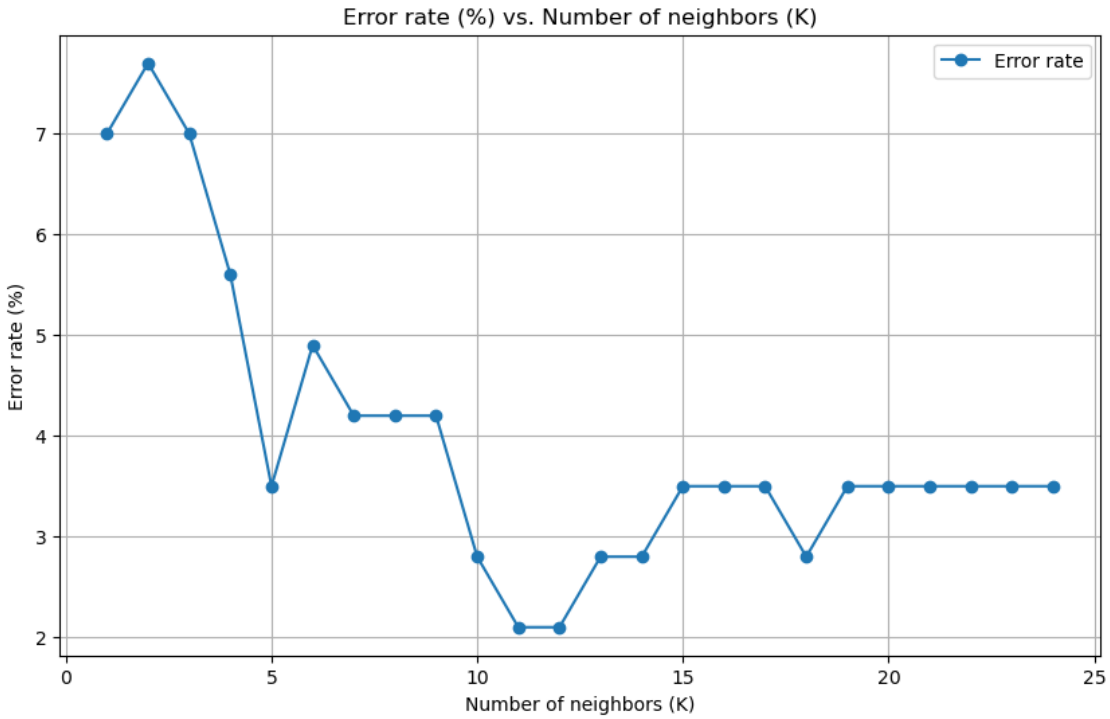
Task 4.1

```
- calculate the error rate for K from 1 to 25
- plot the error rates curve. Set the xlabel to 'K' and ylabel to 'Err'. https://matplotlib.org
```

[14]:
```
import matplotlib.pyplot as plt# evaluate the performance on the test set

#calculate the error rate for K from 1 to 25
errs= []
for k in range(1, 25):
    y_pred = predict(X_test, X_train, y_train, k)
    err = np.mean(y_pred != y_test) * 100
    errs.append(err)
```

```
#plot the error rates curve
plt.figure(figsize=(10, 6))
plt.plot(range(1, 25), errs, marker='o', linestyle='-')
plt.xlabel('Number of neighbors (K)')
plt.ylabel('Error rate (%)')
plt.title('Error rate (%) vs. Number of neighbors (K)')
plt.grid(True)
plt.legend(['Error rate'])
plt.show()
```



Task 4.2: Discuss your findings from the above curve, e.g., the trend, best K, and the impact of small and large Ks Response:

As we can see from the graph that the error rate at the beginning was high (7% for K = 1) for small value of K (for eg: K: 1-3). This is because the model becomes more sensitive to noise in the data when few neighbors are considered.

But when K starts growing, there was a significant drop in error rate. It even went lower to 2% for K = 11,12. We can say that this range of neighbors is the best balance for bias and variance. We need to test to confirm if this neighbor performs best for different subsets for our data.

Then there were slight up-down afterward until it became consistent (in between 3% and 4%). This could be say that increasing K further would not have much precision on model performance.