kNN Example-6 (k-Nearest Neighbor algorithm in plain Python)

လက်ရေး ဂဏန်းများ(hand-written digits)ကို ကိန်း(digit) အဖြစ်ပြောင်းပေးသည့် algorithm

ဤဥပမာသည် လက်ရေး ဂဏန်းများ(hand-written digits)ကို စကန်ဖတ်ထားသည့် images 1797 ခု ပါဝင်သည့် digits dataset ကို train ဒေတာအစု နှင့် test ဒေတာအစု ခွဲခြားပြီး kNN algorithm သည် ဖြင့် classification လုပ်သည့် ဥပမာ ဖြစ်သည်။

k-nn algorithm သည် classification လုပ်ရန် နှင့် regression လုပ်ရန် နှစ်မျိုးလုံးလုပ်ရန်အတွက် အသုံးပြုနိုင်သည့် supervised machine learning algorithm ဖြစ်သည်။ instance-based algorithm တစ်မျိုးဖြစ်သည်။ memory ပေါ်တွင် အဖြေကို ခန့်မှန်းခြင်း(estimating) ပြုလုပ်ရန် model တည်ဆောက်ထားခြင်းမျိုးမဟုတ်ဘဲ training examples များအားလုံးကို memory သိမ်းဆည်းထားပြီး prediction ပြုလုပ်ခြင်းဖြစ်သောကြောင့်လည်း instance-based algorithm ဟုခေါ်ဆိုခြင်းဖြစ်သည်။

k-nn algorithm အလုပ်လုပ်ပုံမှ test လုပ်မည့် input တစ်ခုကို memory ပေါ်ရှိ training example များနှင့် အနီးစပ်ဆုံးတူညီ(most similar)သည့် (euclidean_distance အနည်းဆုံးဖြစ်သည့်) အရာကို ရှာဖြေပြီး အဖြေထုတ်ပေးခြင်းဖြစ်သည်။ တစ်နည်းအားဖြင့် အနီးစပ်ဆုံးတူညီသူ၏ အမျိုးအစားဖြစ်ကြောင်း ခန့်မှန်းခြင်း ဖြစ်သည်။

test လုပ်မည့် input နှင့် training example တိုအကြား euclidean_distance အနည်းဆုံးဖြစ်ခြင်းသည် အနီးစပ်ဆုံးတူညီ(most similar)တူညီသည်။

Classification အတွက်ဆိုလျှင် a) unweighted: output the most common classification among the k-nearest neighbors
b) weighted: sum up the weights of the k-nearest neighbors for each classification value, output classification with highest weight

Regression အတွက်ဆိုလျှင်
a) unweighted: output the average of the values of the k-nearest neighbors
b) weighted: for all classification values, sum up classification value∗weight and divide the result trough the sum of all weights

Feature များ ၏ အရေးပါမှု ကွာခြားမှုသိပ်များသည့်အခါ unweighted k-nn algorithm ကို အသုံးပြုသည်။ Feature များ ၏ အရေးပါမှုမတူညီသည့်အခါ weighted k-nn algorithm ကို အသုံးပြုသည်။

The weighted k-nn version is a refined version of the algorithm in which the contribution of each neighbor is *weighted* according to its distance to the query point. Below, we implement the basic unweighted version of the k-nn algorithm for the digits dataset from sklearn.

Package များကို import လုပ်ခြင်း

matrix များကို တွက်ရန် numpy ကို import လုပ်သည်။
ဂရပ်ပုံများဆွဲတွက်ရန် matplotlib ကို import လုပ်သည်။
ဒေတာများ ရယူရန် sklearn.datasets မှ load_digits ကို import လုပ်သည်။
train ဒေတာအစု နှင့် test ဒေတာအစု ခွဲခြားရန် sklearn.model_selection မှ train_test_split import လုပ်သည်။

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
np.random.seed(123)

%matplotlib inline
```

Dataset

ဤ ဥပမာ တွင် digits dataset ကို အသုံးပြုသည်။ digits dataset ထဲတွင် လက်ရေး ဂဏန်းများ(hand-written digits)ကို စကန်ဖတ်ထားသည့် images 1797 ခု ပါဝင်သည်။digit တစ်ခုစီသည် pixel value များကို 64-dimensional vector ဖြင့် ဖော်ပြထားသည်။

In [2]:
```
# We will use the digits dataset as an example. It consists of the 1797 images of hand-written digits. Each digit is
# represented by a 64-dimensional vector of pixel values.

digits = load_digits()
```

data set ကို ခေါ်ယူပြီးနောက်(load လုပ်ပြီးနောက်) data set ထဲတွင် ပါဝင်သည့် array ၄ ခုကို ကြည့်သည်။ array ၄ ခု မှာ data array, images array, target_names array, target array တို့ ဖြစ်သည်။

In [3]:
```
digits.images
```

Out[3]:
```
array([[[ 0.,  0.,  5., ...,  1.,  0.,  0.],
        [ 0.,  0., 13., ..., 15.,  5.,  0.],
        [ 0.,  3., 15., ..., 11.,  8.,  0.],
        ...,
        [ 0.,  4., 11., ..., 12.,  7.,  0.],
        [ 0.,  2., 14., ..., 12.,  0.,  0.],
        [ 0.,  0.,  6., ...,  0.,  0.,  0.]],

       [[ 0.,  0.,  0., ...,  5.,  0.,  0.],
        [ 0.,  0.,  0., ...,  9.,  0.,  0.],
        [ 0.,  0.,  3., ...,  6.,  0.,  0.],
        ...,
        [ 0.,  0.,  1., ...,  6.,  0.,  0.],
        [ 0.,  0.,  1., ...,  6.,  0.,  0.],
        [ 0.,  0.,  0., ..., 10.,  0.,  0.]],

       [[ 0.,  0.,  0., ..., 12.,  0.,  0.],
        [ 0.,  0.,  3., ..., 14.,  0.,  0.],
        [ 0.,  0.,  8., ..., 16.,  0.,  0.],
        ...,
        [ 0.,  9., 16., ...,  0.,  0.,  0.],
        [ 0.,  3., 13., ..., 11.,  5.,  0.],
        [ 0.,  0.,  0., ..., 16.,  9.,  0.]],

       ...,

       [[ 0.,  0.,  1., ...,  1.,  0.,  0.],
        [ 0.,  0., 13., ...,  2.,  1.,  0.],
        [ 0.,  0., 16., ..., 16.,  5.,  0.],
        ...,
        [ 0.,  0., 16., ..., 15.,  0.,  0.],
        [ 0.,  0., 15., ..., 16.,  0.,  0.],
        [ 0.,  0.,  2., ...,  6.,  0.,  0.]],

       [[ 0.,  0.,  2., ...,  0.,  0.,  0.],
        [ 0.,  0., 14., ..., 15.,  1.,  0.],
        [ 0.,  4., 16., ..., 16.,  7.,  0.],
        ...,
        [ 0.,  0.,  0., ..., 16.,  2.,  0.],
        [ 0.,  0.,  4., ..., 16.,  2.,  0.],
        [ 0.,  0.,  5., ..., 12.,  0.,  0.]],

       [[ 0.,  0., 10., ...,  1.,  0.,  0.],
        [ 0.,  2., 16., ...,  1.,  0.,  0.],
        [ 0.,  0., 15., ..., 15.,  0.,  0.],
        ...,
        [ 0.,  4., 16., ..., 16.,  6.,  0.],
        [ 0.,  8., 16., ..., 16.,  8.,  0.],
        [ 0.,  1.,  8., ..., 12.,  1.,  0.]]])
```

In [4]:
```
digits.target_names
```

Out[4]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

data array ကို X အဖြစ် target array ကို y အဖြစ် သတ်မှတ်သည်။

In [5]:
```
X, y = digits.data, digits.target
```

In [6]:
```
X
```

Out[6]:
```
array([[ 0.,  0.,  5., ...,  0.,  0.,  0.],
       [ 0.,  0.,  0., ..., 10.,  0.,  0.],
       [ 0.,  0.,  0., ..., 16.,  9.,  0.],
       ...,
       [ 0.,  0.,  1., ...,  6.,  0.,  0.],
       [ 0.,  0.,  2., ..., 12.,  0.,  0.],
       [ 0.,  0., 10., ..., 12.,  1.,  0.]])
```

In [7]:
```
y
```

Out[7]:
```
array([0, 1, 2, ..., 8, 9, 8])
```

Train ဒေတာအစု နှင့် Test ဒေတာအစု ခွဲခြားခြင်း

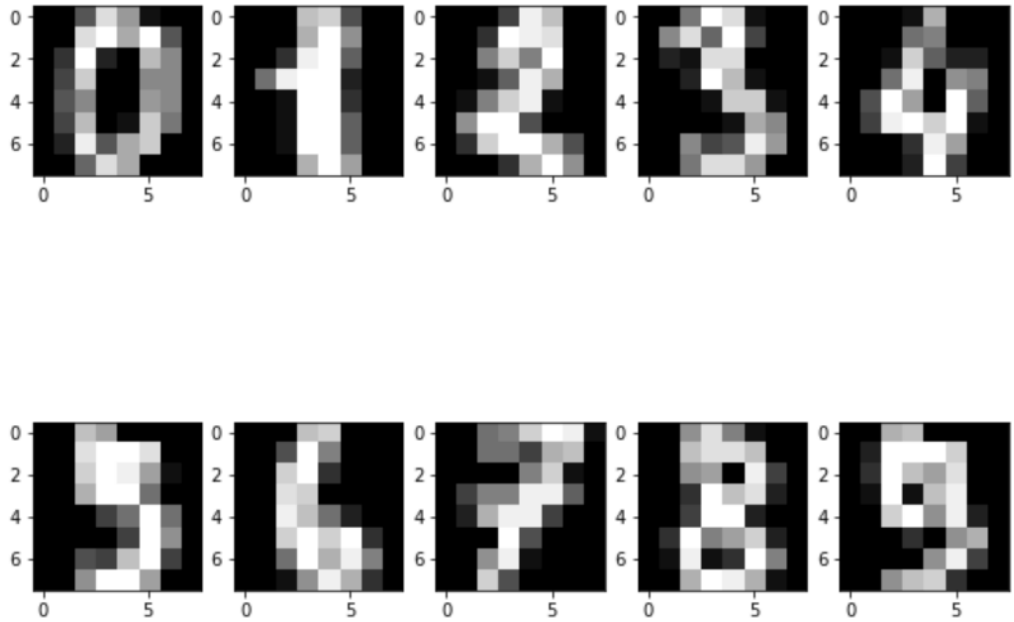
In [8]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y)
print(f'X_train shape: {X_train.shape}')
print(f'y_train shape: {y_train.shape}')
print(f'X_test shape: {X_test.shape}')
print(f'y_test shape: {y_test.shape}')
```

```
X_train shape: (1347, 64)
y_train shape: (1347,)
X_test shape: (450, 64)
y_test shape: (450,)
```

Digits dataset မှ လက်ရေး ဂဏန်းများ(hand-written digits)ကို ပုံဖော်ကြည့်ခြင်း

In [9]:
```
# Example digits
fig = plt.figure(figsize=(10,8))
for i in range(10):
    ax = fig.add_subplot(2, 5, i+1)
    plt.imshow(X[i].reshape((8,8)), cmap='gray')

plt.show()
```



k-Nearest Neighbor Class တည်ဆောက်ခြင်း

Euclidean_distance ကိုတွက်သည့် class တစ်ခုတည်ဆောက်သည်။ training data အများအားလုံးမှ တစ်ခုချင်းစီအတွက် training data တစ်ခုနှင့် new input example (matrix of input examples X) တို့အကြား အကွာအဝေး(euclidean distance)ကို တွက်သည်။

Euclidean_distance အနည်းဆုံးကို ရှာသည်။ ထို euclidean_distance အကွာအဝေးအနီးဖြစ်သည့် training data ပါဝင်သည့် အစု သို့မဟုတ် အဖွဲ့ သို့မဟုတ် အမျိုးအစား ထဲတွင် input test သည် ပါဝင်သည်။

In [10]:
```
class kNN():
    def __init__(self):
        pass

    def fit(self, X, y):
        self.data = X
        self.targets = y

    def euclidean_distance(self, X):
        """
        Computes the euclidean distance between the training data and
        a new input example or matrix of input examples X.

        Training data တစ်ခုနှင့် new input example ( matrix of input examples X) တို့အကြား
         အကွာအဝေး(euclidean distance )ကို တွက်သည်။
         """
        # input: single data point (data point တစ်ခုတည်းအတွက် euclidean distance တွက်ရန်)
        if X.ndim == 1:
            l2 = np.sqrt(np.sum((self.data - X)**2, axis=1))

        # input: matrix of data points (matrix  data point တစ်ခုတည်းအတွက် euclidean distance တွက်ရန်)
        if X.ndim == 2:
            n_samples, _ = X.shape
            l2 = [np.sqrt(np.sum((self.data - X[i])**2, axis=1)) for i in range(n_samples)]

        return np.array(l2)

    def predict(self, X, k=1):
        """
        Predicts the classification for an input example or matrix of input examples X
        """
        # step 1: compute distance between input and training data
        dists = self.euclidean_distance(X)

        # step 2: find the k nearest neighbors and their classifications
        if X.ndim == 1:
            if k == 1:
                nn = np.argmin(dists)
                return self.targets[nn]
            else:
                knn = np.argsort(dists)[:k]
                y_knn = self.targets[knn]
                max_vote = max(y_knn, key=list(y_knn).count)
                return max_vote

        if X.ndim == 2:
            knn = np.argsort(dists)[:, :k]
            y_knn = self.targets[knn]
            if k == 1:
                return y_knn.T
            else:
                n_samples, _ = X.shape
                max_votes = [max(y_knn[i], key=list(y_knn[i]).count) for i in range(n_samples)]
                return max_votes
```

Initializing and training the model

တည်ဆောက်ပြီးသည့် class ကိုသုံး၍ machine learning model တစ်ခုကို ပြုလုပ်သည်။ machine learning model ထဲသို့ input ဖြစ်သည့် X_train နှင့် target ဖြစ်သည့် y_train ကို ထည့်ပေးပြီး fit လုပ်သည်။

In [11]:
```
knn = kNN()
knn.fit(X_train, y_train)

print("Testing one datapoint, k=1")
print(f"Predicted label: {knn.predict(X_test[0], k=1)}")
print(f"True label: {y_test[0]}")
print()
print("Testing one datapoint, k=5")
print(f"Predicted label: {knn.predict(X_test[20], k=5)}")
print(f"True label: {y_test[20]}")
print()
print("Testing 10 datapoint, k=1")
print(f"Predicted labels: {knn.predict(X_test[5:15], k=1)}")
print(f"True labels: {y_test[5:15]}")
print()
print("Testing 10 datapoint, k=4")
print(f"Predicted labels: {knn.predict(X_test[5:15], k=4)}")
print(f"True labels: {y_test[5:15]}")
print()
```

```
Testing one datapoint, k=1
Predicted label: 3
True label: 3

Testing one datapoint, k=5
Predicted label: 9
True label: 9

Testing 10 datapoint, k=1
Predicted labels: [[3 1 0 7 4 0 0 5 1 6]]
True labels: [3 1 0 7 4 0 0 5 1 6]

Testing 10 datapoint, k=4
Predicted labels: [3, 1, 0, 7, 4, 0, 0, 5, 1, 6]
True labels: [3 1 0 7 4 0 0 5 1 6]
```

Accuracy on test set

တည်ဆောက်ထားသည့် machine learning model မှ ထုတ်ပေးသည့် ခန့်မှန်းချက် အဖြေ မည်မျှမှန်ကန်သည်ကို ဆန်းစစ်သည်။

In [12]:
```
# Compute accuracy on test set
y_p_test1 = knn.predict(X_test, k=1)
test_acc1= np.sum(y_p_test1[0] == y_test)/len(y_p_test1[0]) * 100
print(f"Test accuracy with k = 1: {format(test_acc1)}")

y_p_test5 = knn.predict(X_test, k=5)
test_acc5= np.sum(y_p_test5 == y_test)/len(y_p_test5) * 100
print(f"Test accuracy with k = 5: {format(test_acc5)}")
```

```
Test accuracy with k = 1: 97.77777777777777
Test accuracy with k = 5: 97.55555555555556
```

Ref: https://github.com/zotroneneis/machine_learning_basics (https://github.com/zotroneneis/machine_learning_basics)

မြန်မာဘာသာဖြင့် machine learning algorithm ရှင်းပြချက်များကို www.acmv.org/pyml.html တွင် ဖတ်နိုင်ပါသည်။