HW1_Q1.1-7

Spring semester - 2023

[120]:
```
import numpy as np
from sklearn.neighbors import KNeighborsClassifier as KNN
import pandas as pd
import matplotlib.pyplot as plt
```

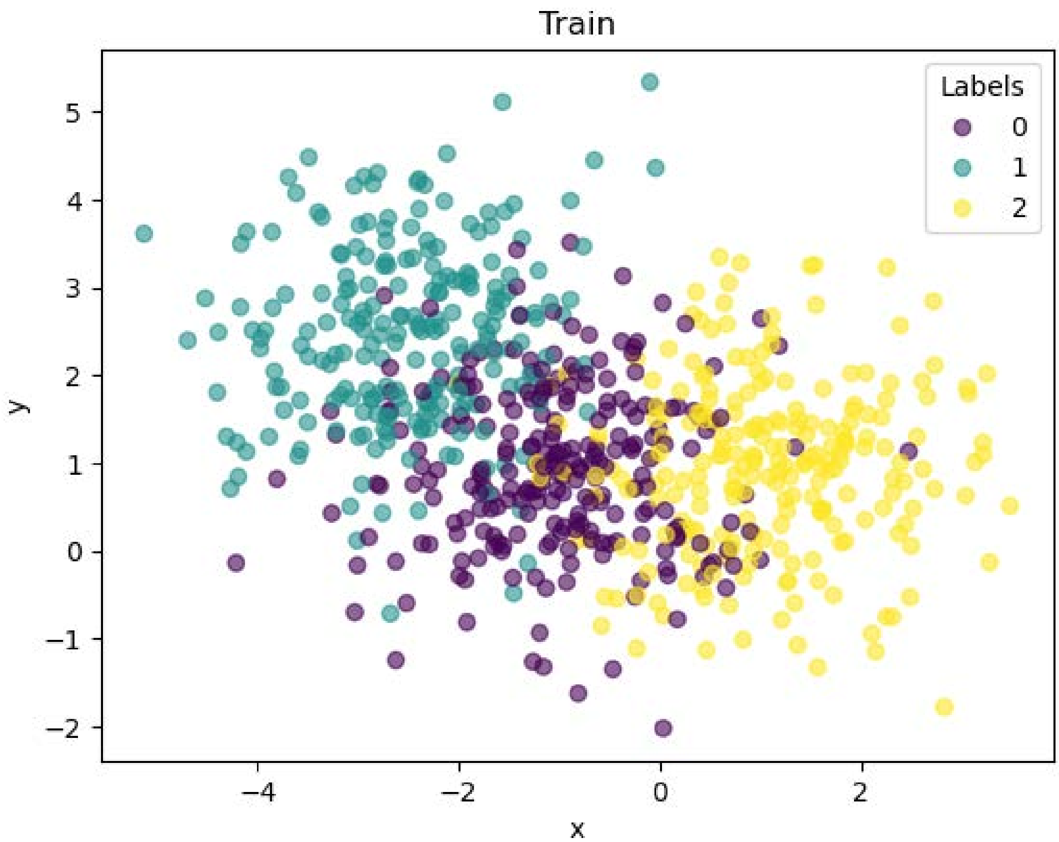
[121]:
```
import sys
print("py:", sys.version)
print("np:", np.version.version)
print("pd:", pd.__version__)
```

```
py: 3.8.8 (default, Apr 13 2021, 15:08:03) [MSC v.1916 64 bit (AMD64)]
np: 1.21.2
pd: 1.3.4
```

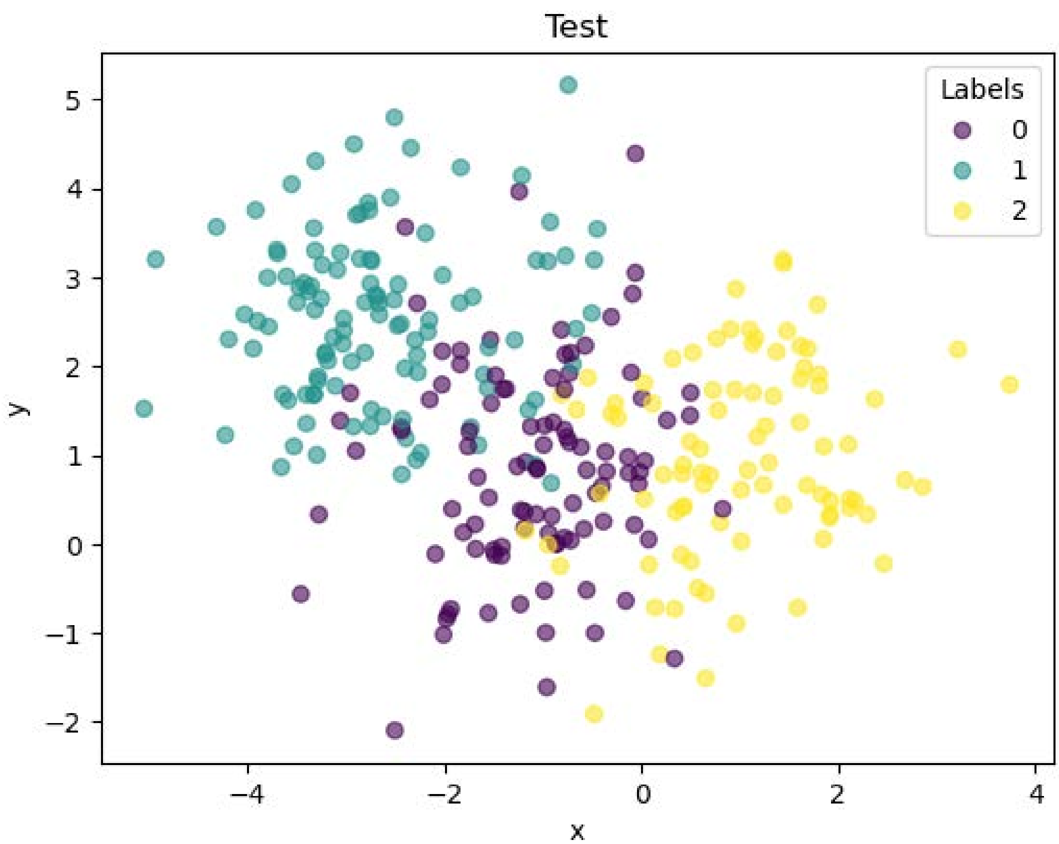
[122]:
```
# Question 1
# 1.

def sample(mu, sigma, p):
    rand = np.random.rand()
    for i in range(len(mu) + 1):
        if i*p < rand < (i+1)*p:
            return np.append(np.random.multivariate_normal(mean=mu[i],
 ↪cov=sigma[i]), i)

mu_0 = [-1,1]
mu_1 = [-2.5,2.5]
mu_2 = [1,1]
sigma = np.identity(2)

n_train = 700

train = pd.DataFrame((sample([mu_0, mu_1, mu_2], [sigma] * 3, 1/3) for _ in
 ↪range(n_train)), columns=('x', 'y', 'label'))
```

[123]:
```
# 2.

_, ax = plt.subplots()
```

```
scatter = ax.scatter(x=train['x'], y=train['y'], c=train['label'], alpha=0.6)
ax.legend(*scatter.legend_elements(), loc="upper right", title="Labels")
plt.gca().update(dict(title='Train', xlabel='x', ylabel='y'))
plt.show()
```



[124]:
```
# 3.

n_test = 300
test = pd.DataFrame((sample([mu_0, mu_1, mu_2], [sigma] * 3, 1/3) for _ in␣
↪range(n_test)), columns=('x', 'y', 'label'))

_, ax = plt.subplots()
scatter = ax.scatter(x=test['x'], y=test['y'], c=test['label'], alpha=0.6)
ax.legend(*scatter.legend_elements(), loc="upper right", title="Labels")
plt.gca().update(dict(title='Test', xlabel='x', ylabel='y'))
plt.show()
```



[125]:
```
# 4.

def CER(labels, predictions):     # Claccification Error Rate
    return np.sum(labels != predictions) / len(labels)

knn = KNN(n_neighbors=1) # euclidean distance is default
knn.fit(X=train[['x', 'y']], y=train['label'])

print("CER for train:", CER(train['label'].values, knn.predict(train[['x',␣
 ↪'y']])))
print("CER for test:", CER(test['label'].values, knn.predict(test[['x', 'y']])))
```

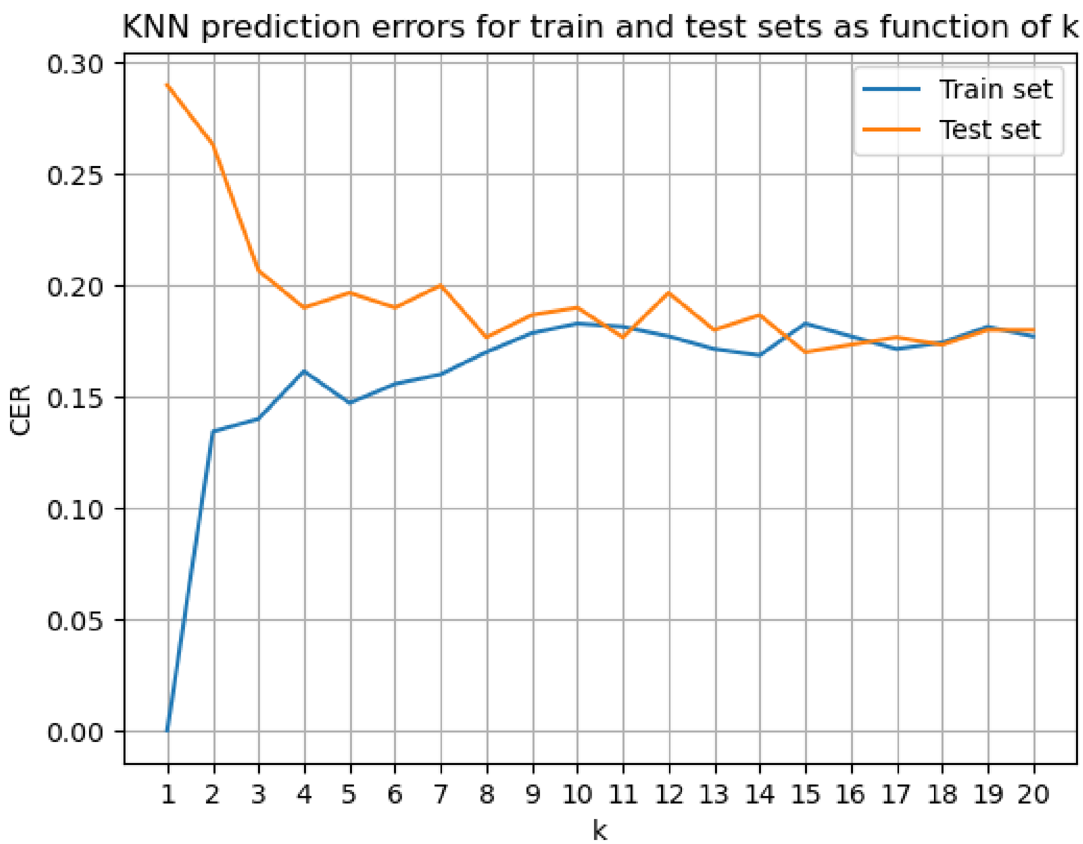
```
CER for train: 0.0
CER for test: 0.29
```

There is a gap, and zero error for train set! Because when k=1, the only point to base the prediction on is a point itself, so there is a serious overfitting for the train set.

[126]:
```
# 5.
errors = {'train': [], 'test': []}
ks = np.arange(1, 21)
```

```
for k in ks:
    knn = KNN(n_neighbors=k)
    knn.fit(X=train[['x', 'y']], y=train['label'])
    errors['train'].append(CER(train['label'].values, knn.predict(train[['x',␣
 ↪'y']])))
    errors['test'].append(CER(test['label'].values, knn.predict(test[['x',␣
 ↪'y']])))
```

[127]:
```
plt.plot(ks, errors['train'], label="Train set")
plt.plot(ks, errors['test'], label="Test set")
plt.gca().update(dict(title='KNN prediction errors for train and test sets as␣
 ↪function of k', xlabel='k', ylabel='CER'))
plt.xticks(ks)
plt.grid()
plt.legend()
plt.show()
```

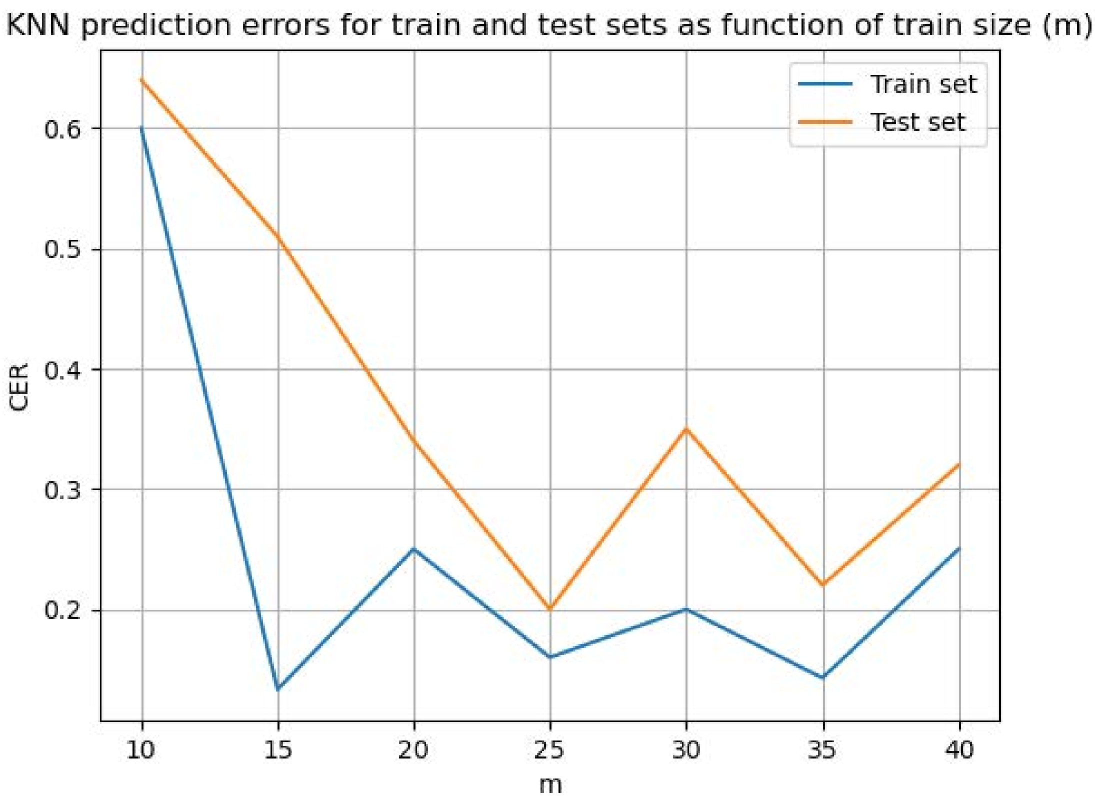


The test error decreases with k. As we saw with Iris dataset (tutorial 1), the error doesn't always decrease with k. In fact, where the dataset is of small size, KNN prediction with large k that is close to the dataset size will almost always be the label of the largest subset (subsets by labels), which will result in |largest set| / |all sets| accuracy, and this may be not the best accuracy we could achieve. In other words, KNN with large k is more sensitive to large subsets.

[132]:
```
# 6.

k = 10
ms = np.arange(10, 45, 5)
errors = {'train': [], 'test': []}

n_test = 100
test = pd.DataFrame((sample([mu_0, mu_1, mu_2], [sigma] * 3, 1/3) for _ in␣
 ↪range(n_test)), columns=('x', 'y', 'label'))

for m in ms:
    train = pd.DataFrame((sample([mu_0, mu_1, mu_2], [sigma] * 3, 1/3) for _ in␣
 ↪range(m)), columns=('x', 'y', 'label'))
    knn = KNN(n_neighbors=k)
    knn.fit(X=train[['x', 'y']], y=train['label'])
    errors['train'].append(CER(train['label'].values, knn.predict(train[['x',␣
 ↪'y']])))
    errors['test'].append(CER(test['label'].values, knn.predict(test[['x',␣
 ↪'y']])))
```

For samples of sizes 10, 15,..., 40 we expect the errors to decrease, as we have more train information with larger set.

[133]:
```
plt.plot(ms, errors['train'], label="Train set")
plt.plot(ms, errors['test'], label="Test set")
plt.gca().update(dict(title='KNN prediction errors for train and test sets as␣
 ↪function of train size (m)', xlabel='m', ylabel='CER'))
plt.xticks(ms)
plt.grid()
plt.legend()
plt.show()
```



7. There is a difference in plots at each of the trials, due to a small train sample - so the variance of trials differs. Nevertheless, at each trial there is a (non-monotonic) decrease in errors, as m goes larger, as expected.

8. נרצה גרסה ממושקלת של k-NN, בדומה ל- weighted regression rule מ-k-NN (*) שראינו בתרגול 1. נגדיר $w = \frac{1}{d}$, כאשר d - מרחק בין שתי נקודות.

כך נקבל משקל יותר גדול עבור נקודות יותר קרובות לנק' x מסוימת.
נסמן: $w(x,x_i)$ - משקל עבור השכן ה-i של נק' x. נרצה למצוא את ה-label שימקסם את הביטוי $\sum_{i=1}^{k} w(x,x_i) \cdot I(y_i = label)$. כלומר החיזוי יהיה:

$$y = \arg\max_{label} \sum_{i=1}^{k} w(x,x_i) \cdot I(y_i = label)$$

(ניתן גם לנרמל את הבעיה ע"י חילוק ב-$\sum_{i=1}^{k} w(x,x_i)$, כפי שראינו עבור (*), כדי להעריך את ההסתברות $P(y = label \mid x)$, הבעיה תשאר שקולה, בהנחה ש-$\sum_{i=1}^{k} w(x,x_i) > 0$.)

שאלה 3

(א)

$$P_w(y_i=k\mid x_i)=\frac{e^{w_k^Tx_i}}{\sum_{j=1}^{k}e^{w_j^Tx_i}} \Rightarrow$$

עבור מקרה בינארי $y_i\in\{1,0\}$

$$\begin{cases} P(y_i=0\mid x_i)=\dfrac{e^{w_1^Tx_i}}{\sum_{j=1}^{2}e^{w_j^Tx_i}}=\dfrac{e^{w_1^Tx_i}}{e^{w_1^Tx_i}+e^{w_2^Tx_i}}=P_1=p \\ P(y_i=1\mid x_i)=\dfrac{e^{w_2^Tx_i}}{\sum_{j=1}^{2}e^{w_j^Tx_i}}=\dfrac{e^{w_2^Tx_i}+e^{w_1^Tx_i}}{e^{w_1^Tx_i}+e^{w_2^Tx_i}}-\dfrac{e^{w_1^Tx_i}}{\sum_{j=1}^{2}e^{w_j^Tx_i}}=P_2=1-p \end{cases}$$

מכאן ואם נגדיר $w_2=0, w_1=w$, מקבלים:

$$P_1=\frac{e^{w_1^Tx_i}}{e^{w_2^Tx_i}+e^{w_1^Tx_i}}\underset{\substack{w_2=0\\w_1=w}}{=}\frac{e^{w^Tx_i}}{1+e^{w^Tx_i}}=p$$

$$P_0=\frac{e^{w_2^Tx_i}}{\sum_{j=1}^{2}e^{w_j^Tx_i}}\underset{\substack{w_2=0\\w_1=w}}{=}\frac{1}{1+e^{w^Tx_i}}=1-P_1=1-p$$

ואשר מביא לנו בדיוק את המקרה הבינארי

(ב) ראינו כי לוג נראות מירבית עבור $y_i\in\{1,\dots,k\}$

$$\log\left(P_w(Y_i=y_i\mid x_i)\right)=\log\left(\frac{e^{w_{y_i}^Tx_i}}{\sum_{j=1}^{k}\exp(w_j^Tx_i)}\right)=w_{y_i}^Tx_i-\log\sum_{j=1}^{k}w_j^Tx_i$$

לכן לכלל המדגם

$$l_S(w)=\sum_{i=1}^{m}\left(w_{y_i}^Tx_i-\log\sum_{j=1}^{k}w_j^Tx_i\right)$$

$$\frac{\partial l_S(w)}{\partial(w_k)}=\sum_{i=1}^{m}I(y_i=k)\cdot x_i-\sum_{i=1}^{m}\left(\frac{e^{w_k^Tx_i}}{\sum_k e^{w_j^Tx_i}}\cdot x_i\right)\Rightarrow\sum_{i=1}^{m}I(y_i=k)\cdot x_i=\sum_{i=1}^{m}\frac{e^{w_k^Tx_i}}{\sum_k e^{w_j^Tx_i}}\cdot x_i$$

(ד) (1) בגלל ש $y_i\in R^3$, אנחנו עובדים עם המקרה הבינארי ונמצא 3 מישורים מפרידים בין נקודות במחלקה לשאר
לשאר הנקודות.

(2) אחד מן המימדים ב w_i משמש את ה bias , בד"כ נסמנו ב w_0.

(3) מניחים בפרט אני מניח שהכוונה שמשקל ה bias הוא w_1

, לכן נקבע $x_{i0}=1$ ונקבל:

$$P_{ik}=\frac{\exp(w_k^Tx_i)}{\sum_k\exp(w_j^Tx_i)}$$

$e^{w_1x_1}=e^{21.5}$, $e^{w_2x_1}=e^{-9.5}$, $e^{w_3x_1}=e^{-12}$ $\Rightarrow$ $P_{11}\cong1$, $P_{12}\cong0$, $P_{13}\cong0$ $\Rightarrow$ $\hat{y}_1=1$

$e^{w_1x_2}=e^{-11}$, $e^{w_2x_2}=e^{8}$, $e^{w_3x_2}=e^{3}$ $\Rightarrow$ $P_{21}\cong0$, $P_{22}\cong0.99$, $P_{23}\cong0$ $\Rightarrow$ $\hat{y}_1=2$

$e^{w_1x_3}=e^{-14}$, $e^{w_2x_3}=e^{2}$, $e^{w_3x_3}=e^{12}$ $\Rightarrow$ $P_{11}\cong0$, $P_{12}\cong0$, $P_{13}\cong0.99$ $\Rightarrow$ $\hat{y}_1=3$