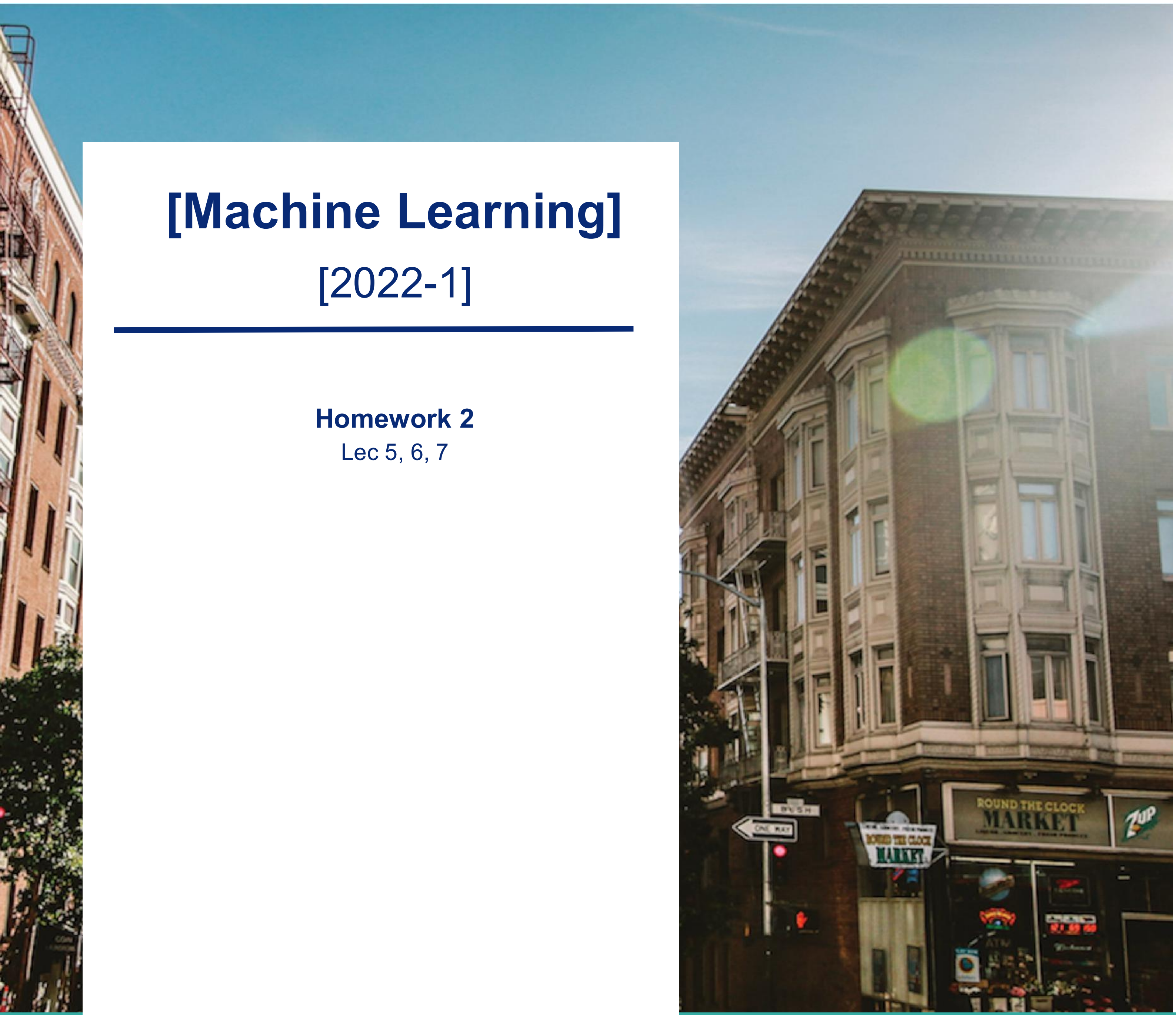


[Machine Learning]

[2022-1]

Homework 2

Lec 5, 6, 7

[Due Date] 2021.04.20

Student ID : 2017112138
Name : Yeo-joon Jung
Professor : Juntae Kim

1. Describe the logistic function(sigmoid) y = $\Phi(x)$ and show that $d\Phi(x)/dx = y(1 - y)$. (10 pts)

Your Answer
sigmoid function $\Phi(x) = 1 / (1+e^{-x})$ $d\Phi(x)/dx = \frac{(1+e^{-x}-1)}{(1+e^{-x})^2}$ $= \frac{1}{1+e^{-x}} - \frac{1}{(1+e^{-x})^2}$ $= \frac{1}{1+e^{-x}} * (1 - \frac{1}{1+e^{-x}})$ = y * (1 – y)

2. Describe the two Impurity measures for Decision Tree Learning(including math formula) and their meaning. (10 pts)

Your Answer
1. Entropy   E = $-\sum_{i=1}^{k} pi \log_2 pi$ Entropy is numerical measure of Impurity. A high entropy means a high impurity. In math formula, the probability p refers to the ratio of specific data to the total data. The formula is to find out how many bits the data can represent and how much information is needed. When you say that there is a lot of space for necessary information, you mean that the entropy is large.   2. Gini Impurity   I = $\sum_{i=1}^{m} p_i(1 - p_i)$ Impurity refers to the degree to which data is not properly classified and mixed. So, The way to measure impurity is calculate (positive group possibility * negative group possibility).

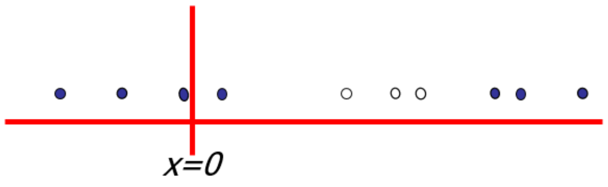
3. Describe the Naïve Bayesian classifier(including math formula) and its meaning. Also explain how you can deal with the continuous feature values. (10 pts)

Your Answer
Naïve Bayesian classifier math formula: $P(C_k\|x) = \frac{P(C_k)P(x\|C_k)}{P(x)}$ It can also be represented in this way (posterior: post probability, prior: pre-probability) $posterior = \frac{prior * likelihood}{evidence}$ We can deal with the continuous feature values by using Gaussian Naïve Bayes. Although the basic principle is the same, we only add the method of calculating P(B|A) as the probability density equation in the normal distribution, assuming that all attributes in the training set follow the Gaussian distribution.

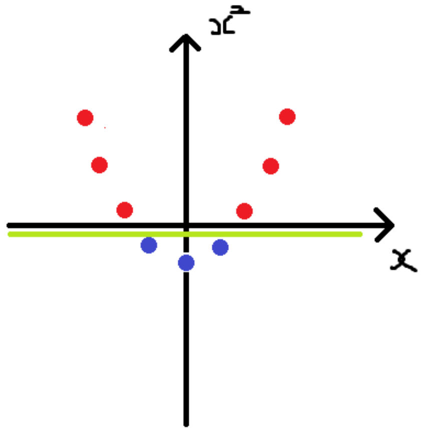
4. Describe why the K-Nearest Neighbors method is not appropriate for dataset with large number of features. (10 pts)

Your Answer
Because K-Nearest Neighbors method is very lazy learning. In fact, it only has learning data rather than learning. Usually in machine learning, there is a process of learning with data and making models, but KNN is an omitted classification method. Therefore, KNN takes a long time if there is dataset with large number of features.

5. Following 1-D dataset is not linearly separable (we cannot find decision boundary x = c. Show how we make them linearly separable by transform the dataset to higher dimensional space. (10 pts)



Your Answer
Use Kernel trick. If input space x is modified as x -> { x, x^2 }



And then we can get a linear boundary as above.

6. Apply Logistic Regression, Decision Tree, Naïve Bayesian Classifier, k-Nearest Neighbor, and SVM on Wine Dataset to predict the origin of wines. Describe the learned model, and compare the accuracies. (30 pts)

- Dataset

https://archive.ics.uci.edu/ml/datasets/Wine

The dataset is the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the 3 types of wines.

- Use downloaded raw data or scikit-learn library

https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_wine.html

```
from sklearn.datasets import load_wine

wine = load_wine()
X = wine.data
y = wine.target
```

- Check test accuracy (use 30% for test)

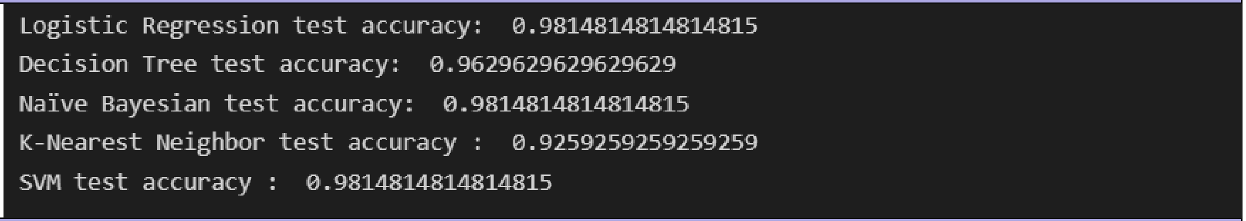
```
lr.score(X_test_std, y_test)
0.9814814814814815
```

Code

```
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC


wine = load_wine()
X = wine.data
y = wine.target
```

```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3,
random_state=1,stratify=y)

sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)

lr = LogisticRegression(C=100.0, random_state=1)
lr.fit(X_train_std,y_train)
acc = lr.score(X_test_std,y_test)
print("Logistic Regression test accuracy: ",acc)

tree = DecisionTreeClassifier(criterion='gini',random_state=1)
tree.fit(X_train,y_train)

acc = tree.score(X_test,y_test)
print("Decision Tree test accuracy: ", acc)

gnb = GaussianNB()
gnb.fit(X_train,y_train)

acc = gnb.score(X_test,y_test)
print("Naïve Bayesian test accuracy: ", acc)

knn = KNeighborsClassifier(n_neighbors=5, p=2, metric='minkowski')
knn.fit(X_train_std,y_train)

acc = knn.score(X_test_std,y_test)
print("K-Nearest Neighbor test accuracy : ", acc)

svm = SVC(kernel='rbf',random_state=1,gamma=0.2)
svm.fit(X_train_std,y_train)

acc = svm.score(X_test_std,y_test)
print("SVM test accuracy : ", acc)
```

Result(Captured images)



```
Logistic Regression test accuracy:  0.9814814814814815
Decision Tree test accuracy:  0.9629629629629629
Naïve Bayesian test accuracy:  0.9814814814814815
K-Nearest Neighbor test accuracy :  0.9259259259259259
SVM test accuracy :  0.9814814814814815
```

Description

load_wine()을 통해 wine 데이터를 가져온다.

wine.data 를 X 에 저장하고 wine.target 을 y 에 저장한다.

train_test_split(test_size=0.3)을 이용해서 (use 30% for test) 조건을 만족시킨다. 이제 StandardScaler()를 사용해서 표준화해서 X_train_std 와 X_test_std 를 얻는다.
이렇게 얻은 값을 LogisticRegression(), DecisionTreeClassifier(), GaussianNB(), KNeighborsClassifier(), SVC() 함수들을 이용해서 각각의 정확도를 측정한다.

7. Apply Multilayer Perceptron on Olivetti Faces Dataset to identify persons from images. Describe the learned model. (20 pts)

- Dataset

https://scikit-learn.org/stable/modules/generated/sklearn.datasets.fetch_olivetti_faces.html

The Olivetti Faces dataset contains a set of face images taken at AT&T Laboratories Cambridge. The sklearn.datasets.fetch_olivetti_faces function is the data fetching function that downloads the data.
There are 10 different images of each of 40 distinct persons. For some persons, the images were taken at different times, varying the lighting, facial expressions (open / closed eyes, smiling / not smiling) and facial details (glasses / no glasses).
The 64x64 pixels image is quantized to 256 grey levels and stored as unsigned 8-bit integers; the loader will convert these to floating point values on the interval [0, 1].
The target for this database is an integer from 0 to 39 indicating the identity of the person pictured.

- Check test accuracy (use 20% for test)

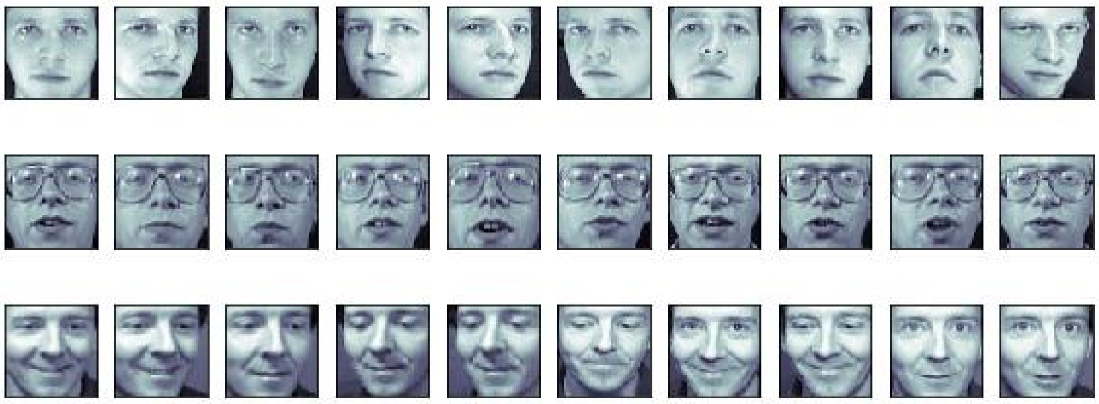
```
print("Training set score: %f" % mlp.score(X_train, y_train))
print("Test set score: %f" % mlp.score(X_test, y_test))
```

```
Training set score: 1.000000
Test set score: 0.975000
```

Plotting several images (person 0, 1, 2)

```
from sklearn import datasets
from matplotlib import pyplot as plt

face = datasets.fetch_olivetti_faces()
X = face.data
y = face.target

%matplotlib inline

fig = plt.figure(figsize=(10, 4))
```

```
for i in range(30):
    ax = fig.add_subplot(3, 10, i + 1, xticks=[], yticks=[])
    ax.imshow(face.images[i], cmap=plt.cm.bone)
```

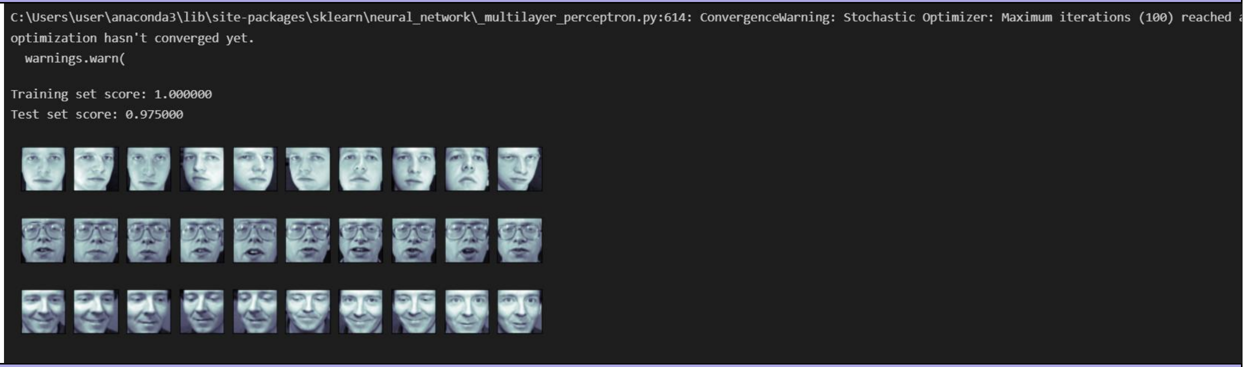


Code

```
from sklearn.datasets import fetch_olivetti_faces
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPClassifier
from matplotlib import pyplot as plt

face = fetch_olivetti_faces()
X = face.data
y = face.target

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2,
random_state=1,stratify=y)

sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)

mlp =
MLPClassifier(hidden_layer_sizes=(64,64),max_iter=100,alpha=1,solver='sgd',verbos
e=0,tol=1e-6,random_state=0,learning_rate_init=0.1)
mlp.fit(X_train_std,y_train)

print("Training set score: %f" % mlp.score(X_train_std, y_train))
print("Test set score: %f" % mlp.score(X_test_std, y_test))

%matplotlib inline
```

```
fig = plt.figure(figsize=(10, 4))
for i in range(30):
    ax = fig.add_subplot(3, 10, i + 1, xticks=[], yticks=[])
    ax.imshow(face.images[i], cmap=plt.cm.bone)
```

Result(Captured images)



```
C:\Users\user\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:614: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (100) reached a
optimization hasn't converged yet.
  warnings.warn(

Training set score: 1.000000
Test set score: 0.975000
```

Description

from sklearn.datasets import fetch_olivetti_faces
face = fetch_olivetti_faces()를 통해 Olivetti faces 데이터를 저장한다.
X = face.data
y = face.target
을 통해 각각 X 와 y 에 저장한다.
train_test_split(test_size=0.2)을 이용해서 (use 20% for test) 조건을 만족시킨다. StandardScaler()를 사용해서 표준화해서 X_train_std 와 X_test_std 를 얻는다. hidden layer 의 크기를 (64,64)로 설정하고 최대 반복을 100 회로 설정하고 random_state=0 으로 learning rate 를 0.1 로 설정한다. 표준화를 통해 얻은 값을 위에 값으로 설정한 Multilayer Perceptron 으로 학습시키고 정확도를 측정한다.

Note

1. Summit the file to e-class as pdf.
2. Specify your file name as “hw2_<StudentID>_<Name>.pdf”