[CSE302] Introduction to Machine Learning

< Assignment 2 >

(Deadline: 2022-05-10)

202123008 Jinmin Kim (김진민)

Phone: 010-6266-6099

Mail: rlawlsals@dgist.ac.kr

(1) For MNIST data set, train Logistic regression models and find the best model that can achieve the highest accuracy on the test data set.

** Code

```
import numpy as np
from sklearn.datasets import fetch_openml
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn import svm
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
```

: 사용한 모듈

```
# ######################################## Data import ######################################### #
print('** Data import ')
print('')

# MNIST 데이터 불러오기
mnist = fetch_openml('mnist_784')
print('MNIST data shape: ', mnist.data.shape, 'MNIST label shape: ', mnist.target.shape)

# Train, Test 데이터 정의
X_train, X_trash, y_train, y_trash = train_test_split(mnist.data, mnist.target, test_size=0.95)
X_train, X_test, y_train, y_test = train_test_split(X_train, y_train, test_size=0.2)
print('train data shape: ', X_train.shape, 'train label shape: ', y_train.shape)
print('test data shape: ', X_test.shape, 'test label shape: ', y_test.shape)

# Regularization
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

: MNIST 데이터셋을 불러온다.

- 불러온 데이터셋의 크기는 (70,000, 784), (70,000,)이다. 해당 데이터셋은 개수가 너무 많아 컴퓨팅 시간이 오래걸렸다. 따라서 Train 데이터로 2,800개, Test 데이터로 700개를 추출하여 사용하였다.
- 그 후 Train, Test 데이터의 Feature에 대한 Regularization을 진행하여 데이터 전처리를 진행하였다.

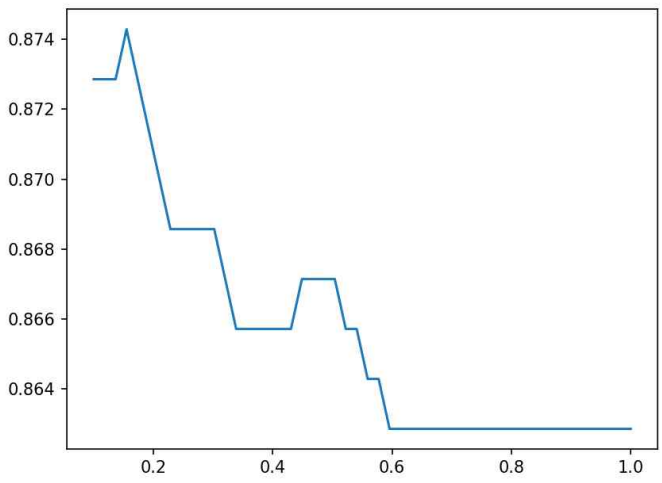
```
# ######################################## Problem 1 ######################################## #
print('** Problem 1 : Logistic regression models')
print('')

model1 = LogisticRegression(C=0.15, max_iter=100000)
model1.fit(X_train, y_train)
predict1 = model1.predict(X_test)

accuracy1 = accuracy_score(y_test, predict1)
mse1 = mean_squared_error(y_test, predict1)
rmse1 = np.sqrt(mse1)

print('accuracy :', accuracy1)
print('standard deviation :', rmse1)
```

: Problem 1 - Logistic regression 구현코드

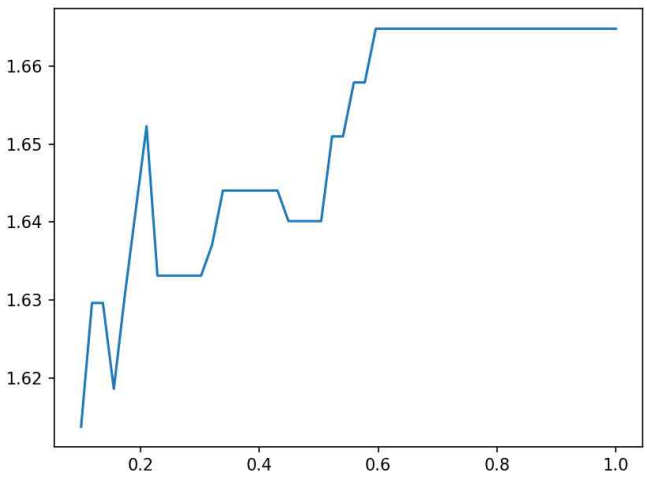
** The effect of parameters

Scikit-learn의 LogisticRegression 함수에서 조정할 hyper parameter는 Regularization term인 C이다.

- C parameter는 LogisticRegression의 Cost function에서 hyper parameter λ에 해당한다.
- C에 따른 모델의 Accuracy와 rmse는 다음 그림과 같이 추출되었다.



x축:C, y축:accuracy



x축:C, y축:rmse

** Discussion

- C에 따른 accuracy와 rmse 그래프를 통해서, 모델의 accuracy를 높이고 rmse를 낮게 하는 <u>가장 최적인 C는 0.15로 선정하였다</u>.

(2) For the same data set, train K-NN classifiers and find the best model that can achieve the highest accuracy on the test data set.

** Code

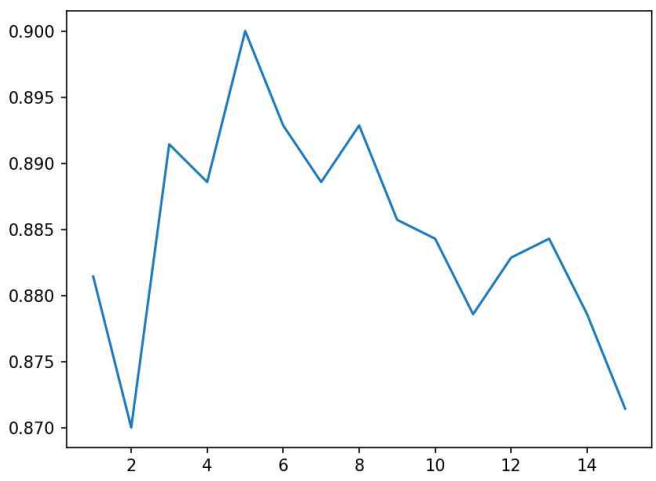
```
# ######################################## Problem 2 ######################################## #
print('** Problem 2 : K-NN classifiers')
print('')

model2 = KNeighborsClassifier(n_neighbors=5)
model2.fit(X_train, y_train)
predict2 = model2.predict(X_test)

accuracy2 = accuracy_score(y_test, predict2)
mse2 = mean_squared_error(y_test, predict2)
rmse2 = np.sqrt(mse2)

print('accuracy :', accuracy2)
print('standard deviation :', rmse2)
```

: Problem 2 - K-NN 구현코드

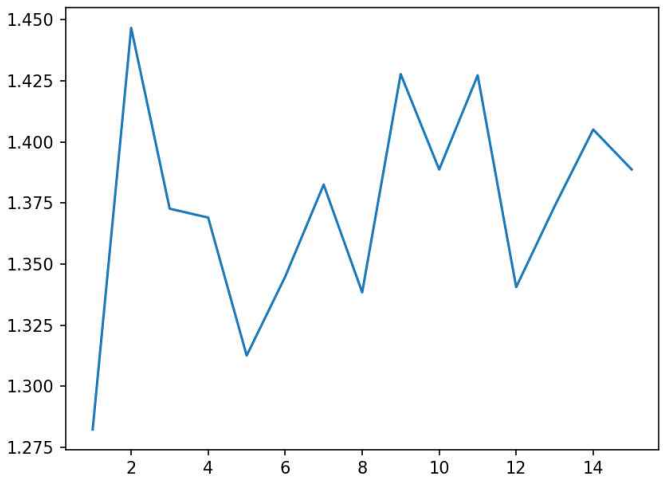
** The effect of parameters

K-Nearest Neighbors method 에서 조정하는 hyper parameter는 K이다.

- K는 몇 개의 이웃한 점을 사용할 것인가를 나타내는 것으로, K개의 이웃한 데이터를 이용하여 Decision boundary를 결정하게 된다.
- K에 따른 모델의 Accuracy와 rmse는 다음 그림과 같이 추출되었다.



x축:K, y축:accuracy



x축:K, y축:rmse

** Discussion

- K에 따른 accuracy와 rmse 그래프를 통해서, 모델의 accuracy를 높이고 rmse를 낮게 하는 **가장 최적인 K는 5로 선정하였다**.

(3) For the same data set, train SVM classifiers and find the best model that can achieve the highest accuracy on the test data set.

** Code

```
# ##################################### Problem 3 ##################################### #
print('** Problem 3 : SVM classifiers')
print('')

model3 = svm.SVC(C=10.0, kernel='poly', gamma=0.1)
model3.fit(X_train, y_train)
predict3 = model3.predict(X_test)

accuracy3 = accuracy_score(y_test, predict3)
mse3 = mean_squared_error(y_test, predict3)
rmse3 = np.sqrt(mse3)

print('accuracy :', accuracy3)
print('standard deviation :', rmse3)
```

: Problem 3 - SVM 구현코드

** The effect of parameters

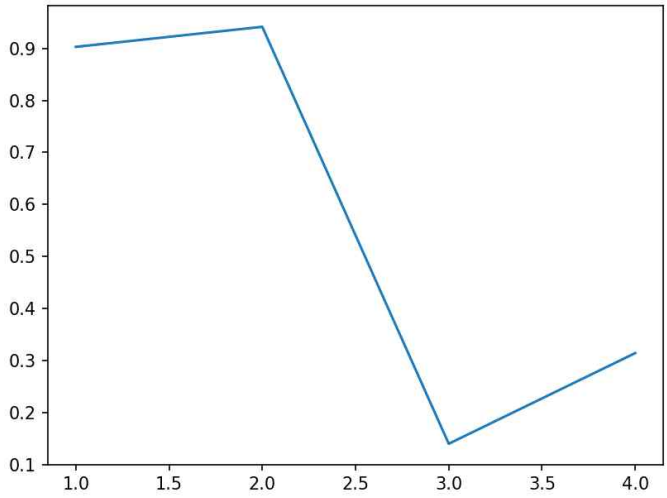
Support Vector Machine method에서 조정할 수 있는 hyper parameter는 C, kernel, gamma이다.

- C는 Regularization term에 있는 상수로써, slack variable의 가중치로 사용된다.
- kernel은 SVM의 trick으로, 사용자가 정의하여 사용한다.
- gamma는 일부 kernel의 coefficient이다.

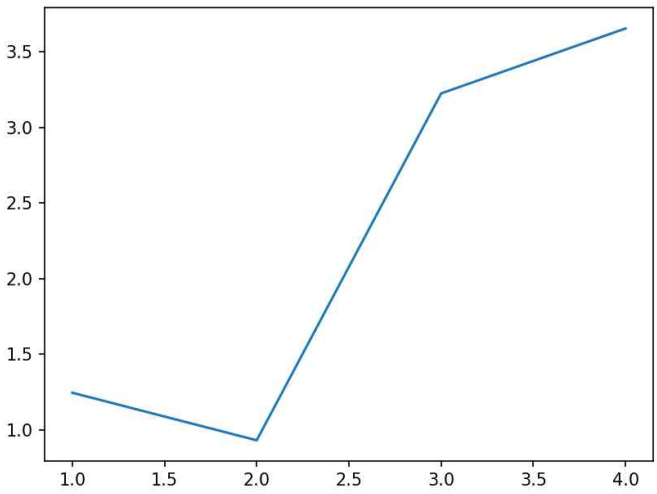
C를 작게 하면 Decision boundary를 곧게 그리는 대신 Margin을 크게 하고, C를 크게 하면 Margin을 작게 하는 대신 Decision boundary가 굴곡있게 그려진다.

우리의 데이터에 적합한 kernel을 찾기 위해 각 kernel을 한 번씩 사용해보았다.

사용한 Kernel은
Kernel 1: linear
Kernel 2: poly
Kernel 3: rbf
Kernel 4: sigmoid 이다.



x축:Kernels, y축:accuracy



x축:Kernels, y축:rmse

위 그래프를 통해, 우리의 데이터에 적합한 Kernel은 'poly'임을 알 수 있다.

** Discussion

- Kernel의 종류에 따른 accuracy와 rmse 그래프를 통해서, 모델의 accuracy를 높이고 rmse를 낮게 하는 **가장 최적인 Kernel은 'poly'로 선정하였다**.

(4) For the same data set, train Random forest classifiers and find the best model that can achieve the highest accuracy on the test data set.

** Code

```
# ######################################## Problem 4 ######################################## #
print('** Problem 4 : Random forest classifiers')
print('')

model4 = RandomForestClassifier(n_estimators=74, random_state=0)
model4.fit(X_train, y_train)
predict4 = model4.predict(X_test)

accuracy4 = accuracy_score(y_test, predict4)
mse4 = mean_squared_error(y_test, predict4)
rmse4 = np.sqrt(mse4)

print('accuracy :', accuracy4)
print('standard deviation :', rmse4)
```

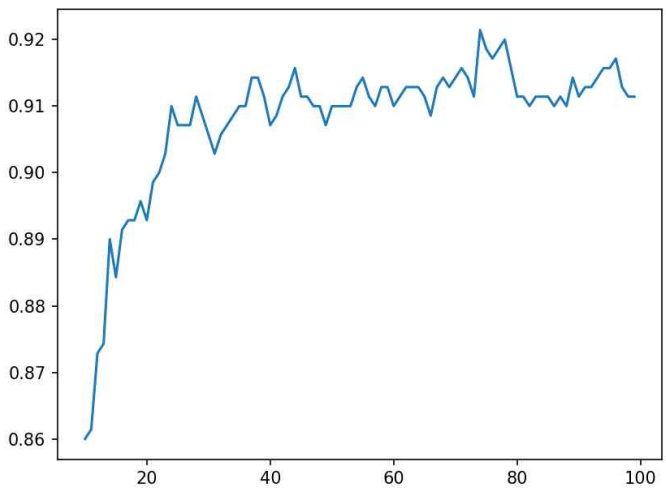
: Problem 4 - Random forest 구현코드

** The effect of parameters

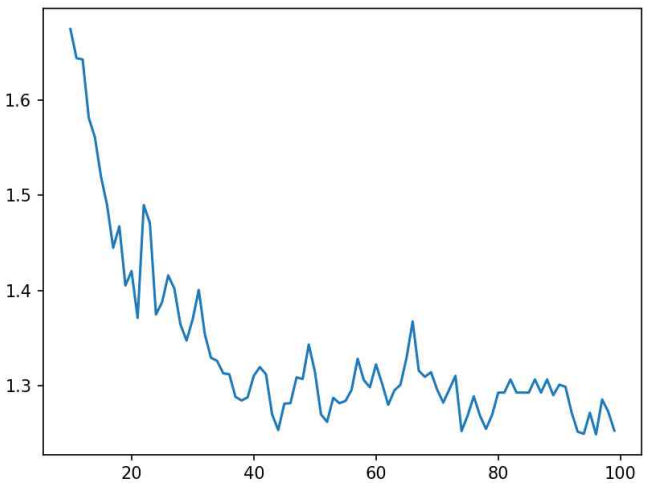
Random forest method에서 조정할 수 있는 hyper parameter는 forest의 tree 개수이다.

- Random forest 알고리즘은 앙상블 모델을 활용하여, 여러 개의 tree를 생성하고 각각의 tree의 prediction을 조합하여 최종 예측하는 알고리즘이다.
- tree의 개수가 많을수록 정확도를 개선되지만 기울기가 급변하는 지점이 존재하게 된다.

Tree의 개수에 대한 Accuracy와 rmse를 비교해보았다.



x축:number of tree, y축:accuracy



x축:number of tree, y축:rmse

- 위 그래프를 통해, 우리의 데이터 가장 적합한 tree의 개수는 74개임을 알 수 있다.

** Discussion

- Tree의 개수에 따른 accuracy와 rmse 그래프를 통해서, 모델의 accuracy를 높이고 rmse를 낮게 하는 **가장 최적인 tree의 개수는 74개로 선정하였다**.

Best results

- 최종적으로, 각 method에 대해 선정한 모델의 결과는 다음과 같다.

	Logistic regression	K-NN	SVM	Random Forest
Accuracy	0.88285714285714	0.89285714285714	0.93428571428571	0.93428571428571
rmse	1.34907375632320	1.22940171279715	1.03302606798805	0.99642217099838