Machine Learning Coding Assignment

2021321139 조수연

Q1. Classification with Decision Trees

Feature Engineering : PCA를 이용한 차원 축소

- 기존(32x32x3 = 3072) -> 변경 이후(48)
  (분산의 설명 비율이 70% 넘기 전까지의 주성분 이용)

```
# vectorization/flat operation (to make the 2D input into 1D vector)
X_tr_flat = X_tr.reshape(X_tr.shape[0],-1)
X_ts_flat = X_ts.reshape(X_ts.shape[0],-1)
```

```
# PCA
pca = PCA()
pca.fit(X_tr_flat)
```

```
# 변환
X_pc_tr = pca.transform(X_tr_flat)
X_pc_ts = pca.transform(X_ts_flat)
```

```
n_pc = sum(pca.explained_variance_ratio_.cumsum() < 0.7) #48

X_pc_tr = X_pc_tr[:,:n_pc]
X_pc_ts = X_pc_ts[:,:n_pc]
```

```
print(X_tr_flat.shape)
print(X_pc_tr.shape)
```

Python

```
(41280, 3072)
(41280, 48)
```

REPORT 1-1 : Describe model details (hyperparameters) your have used (e.g., depth, 2. any specialty of this model and etc.)

1) Hyperparameter Search : Random Search
2) Hyperparameter Search

```python
DT_params = {
    'criterion' : ['gini', 'entropy'],
    'max_depth' : randint(40, 80),
    'min_samples_split' : randint(2, 10)
}
```

Python

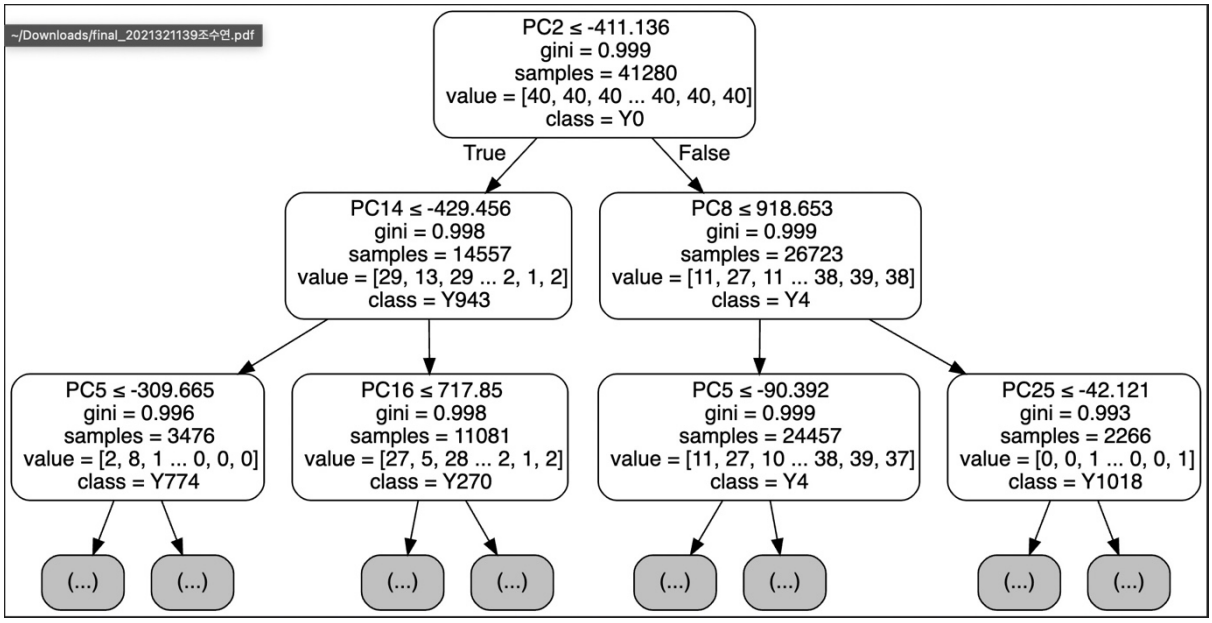
REPORT 1-2 : How to choose to use these hyperparameters? (e.g., using cross validation)

- 최적의 Hyperparameter 선정 : 5-fold Cross Validation

REPORT 1-3: Report both the training and testing accuracy in a plot (x: epoch, y: accuracy)

	Untuned	Tuned
Train Accuracy	1.0	0.7723
Validation Accuracy	0.1619	0.1466
Test Accuracy	0.1844	0.1675

REPORT 1-4: Visualize your tree using Graphviz



REPORT 1-5: Discuss any ideas to improve the accuracy (e.g., other depth)

- Hyperparameter Search에서 Random Search의 대상 parameter 범위를 넓힌다

Q2. Classification with Logistic Regression

Feature Engineering : PCA를 이용한 차원 축소

- 기존(32x32x3 = 3072) -> 변경 이후(48)
  (분산의 설명 비율이 70% 넘기 전까지의 주성분 이용)

REPORT 2-1: Describe model details (hyperparameters) your have used (e.g., different entropy operators and etc.)

1) Hyperparameter Search : Random Search
2) Hyperparameter Search

```python
LR_params = {
    'penalty' : ['none', 'l1', 'l2', 'elasticnet']
}
```

REPORT 2-2: Why did you choose to use these hyperparameters? (e.g., using cross validation)

- 최적의 Hyperparameter 선정 : 5-fold Cross Validation

REPORT 2-3: Report both the training and testing accuracy in a plot (x: epoch, y: accuracy)

	Untuned	Tuned
Train Accuracy	0.9388	0.9020
Validation Accuracy	0.6621	0.6693
Test Accuracy	0.6874	0.6988

REPORT 2-4: Discuss any ideas to improve the accuracy

- Threshold 값을 다양하게 시도해본다

Q3. Classification with Support Vector Mashine (SVM) Classifier

Feature Engineering : PCA를 이용한 차원 축소

- 기존(32x32x3 = 3072) -> 변경 이후(48)
  (분산의 설명 비율이 70% 넘기 전까지의 주성분 이용)

REPORT 3-1: Describe model details (hyperparameters) your have used (e.g., C, hinge or squared hinge and etc.)

1) Hyperparameter Search : Grid Search
2) Hyperparameter Search

```python
SVM_params = {
    'C' : randint(3, 10)
}
```

REPORT 3-2: Why did you choose to use these hyperparameters? (e.g., using cross validation)

- 최적의 Hyperparameter 선정 : 5-fold Cross Validation

REPORT 3-3: Report both the training and testing accuracy in a plot (x: epoch, y: accuracy)

	Untuned	Tuned
Train Accuracy	0.8999	0.9989
Validation Accuracy	0.6200	0.7610
Test Accuracy	0.6682	0.7901

REPORT 3-4: Discuss any ideas to improve the accuracy

- Kernel 값을 다양하게 시도해본다 (Guassian Kernel, Sigmoid Kernel)

Q4. Classification with neural network classifier - multi layer perceptron (MLP)

- Using pythorch

REPORT 4-1: Describe model details (hyperparameters) your have used (e.g., C, hinge or squared hinge and etc.)

```
mlp_model = mlp_classifier(in_dim = X_tr_mlp.shape[1], out_dim = 1032, hid_dim = 32).to(device)
```

```
loss_fn = nn.CrossEntropyLoss().to(device)
opt = torch.optim.Adam(mlp_model.parameters(), lr = 0.001)
```

REPORT 4-2: Why did you choose to use these hyperparameters? (e.g., using cross validation)

- layer의 개수와 layer 내의 hidden unit의 개수 변경
- Activation function : 성능이 가장 좋은 ReLU 사용
- Train_set : Validation_set = 80% : 20%로 나누고, validation loss 값이 가장 작은 것 선택

REPORT 4-3: Report both the training and testing accuracy in a plot (x: epoch, y: accuracy)

	MLP Accuracy
Train Accuracy	0.8821
Validation Accuracy	0.7602
Test Accuracy	0.7782

REPORT 4-4: Discuss any ideas to improve the accuracy

- Bath size 값을 바꿔서 시행해본다
- Droupout을 시행해본다

Q5. Classification with neural network classifier - deep convolutional neural network (CNN)

REPORT 5-1: Describe model details (hyperparameters) your have used (e.g., model architecture or loss etc.)

```
cnn_model = cnn_classifier(in_ch = 3, mid_ch = 32, num_class = 1032).to(device)
```

```
loss_fn = nn.CrossEntropyLoss().to(device)
opt = torch.optim.Adam(cnn_model.parameters(), lr = 0.001)
```

REPORT 5-2: Why did you choose to use these hyperparameters? (e.g., using cross validation)

- layer의 개수 : 3 (input_channel = 3, mid_channel = 63, out_channel = 1024)
- Kernel size : 3x3
- Train_set : Validation_set = 80% : 20%로 나누고, validation loss 값이 가장 작은 것 선택

REPORT 5-3: Report both the training and testing accuracy in a plot (x: epoch, y: accuracy)

	CNN Accuracy
Train Accuracy	0.9908
Validation Accuracy	0.9010
Test Accuracy	0.9103

REPORT 5-4: Discuss any ideas to improve the accuracy

- Fine tuning을 시행해본다