REPORT

Assignment #1_ Supervised Learning

과목명	머신러닝 02분반
교수명	김은우
학번	20206319
학과	소프트웨어학부 소프트웨어전공
이름	김가연

I used Kaggle for machine learning training. The environment of the computer is as follows.
The CPU type of this machine is 12th Gen Intel Core i9-12900K.
It has 16 cores, 24 logical processors, and clock speed is 3.20GHz.
The memory size is 32GB. The speed of the memory is 4400MHz.
The OS type is 64-bit operating system, x64 based processor.
My Kaggle's Jupiter notebook uses 3.5GB RAM.

All the environment setting codes for using the PyTorch library presented in the assignment were used, and the environment for the two classification methods was written and executed with the following code.

```
from torch.utils.data import DataLoader
from torchvision import transforms, datasets
import numpy as np
from sklearn import svm
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier
from scipy.stats import randint
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score

#CIFAR
CIFAR_transform_train = transforms.Compose([transforms.ToTensor()])
CIFAR_transform_test = transforms.Compose([transforms.ToTensor()])

trainset_CIFAR = datasets.CIFAR10(root='./data', train=True, download=True, transform=CIFAR_transform_train)
testset_CIFAR = datasets.CIFAR10(root='./data', train=False, download=True, transform=CIFAR_transform_test)

CIFAR_train = DataLoader(trainset_CIFAR, batch_size=32, shuffle=True, num_workers=2)
CIFAR_test = DataLoader(testset_CIFAR, batch_size=32, shuffle=False, num_workers=2)

CIFAR_train_images = []
CIFAR_train_labels = []
for batch in CIFAR_train:
    images, labels = batch
    images_flat = images.view(images.shape[0], -1)
    CIFAR_train_images.append(images_flat.numpy())
    CIFAR_train_labels.append(labels.numpy())

CIFAR_train_images = np.vstack(CIFAR_train_images)
CIFAR_train_labels = np.concatenate(CIFAR_train_labels)
```

```
CIFAR_test_images = []
CIFAR_test_labels = []
for batch in CIFAR_test:
    images, labels = batch
    images_flat = images.view(images.shape[0], -1)
    CIFAR_test_images.append(images_flat.numpy())
    CIFAR_test_labels.append(labels.numpy())

CIFAR_test_images = np.vstack(CIFAR_test_images)
CIFAR_test_labels = np.concatenate(CIFAR_test_labels)

# MNIST
mnist_train_transform = transforms.Compose([transforms.ToTensor()])
mnist_test_transform = transforms.Compose([transforms.ToTensor()])

trainset_mnist = datasets.MNIST(root='./data', train=True, download=True,
transform=mnist_train_transform)
testset_mnist = datasets.MNIST(root='./data', train=False, download=True,
transform=mnist_test_transform)

MNIST_train = DataLoader(trainset_mnist, batch_size=32, shuffle=True, num_workers=2)
MNIST_test = DataLoader(testset_mnist, batch_size=32, shuffle=False, num_workers=2)

MNIST_train_images = []
MNIST_train_labels = []
for batch in MNIST_train:
    images, labels = batch
    images_flat = images.view(images.shape[0], -1)
    MNIST_train_images.append(images_flat.numpy())
    MNIST_train_labels.append(labels.numpy())
MNIST_train_images = np.vstack(MNIST_train_images)
MNIST_train_labels = np.concatenate(MNIST_train_labels)

MNIST_test_images = []
MNIST_test_labels = []
for batch in MNIST_test:
    images, labels = batch
    images_flat = images.view(images.shape[0], -1)
    MNIST_test_images.append(images_flat.numpy())
    MNIST_test_labels.append(labels.numpy())
MNIST_test_images = np.vstack(MNIST_test_images)
MNIST_test_labels = np.concatenate(MNIST_test_labels)
```

This is code results.

```
Downloading https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz to ./data/cifar-10-python.tar.gz
Loading widget...
Extracting ./data/cifar-10-python.tar.gz to ./data
Files already downloaded and verified
Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz to ./data/MNIST/raw/train-images-idx3-ubyte.gz
Loading widget...
Extracting ./data/MNIST/raw/train-images-idx3-ubyte.gz to ./data/MNIST/raw

Downloading http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz to ./data/MNIST/raw/train-labels-idx1-ubyte.gz
Loading widget...
Extracting ./data/MNIST/raw/train-labels-idx1-ubyte.gz to ./data/MNIST/raw

Downloading http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz to ./data/MNIST/raw/t10k-images-idx3-ubyte.gz
Loading widget...
Extracting ./data/MNIST/raw/t10k-images-idx3-ubyte.gz to ./data/MNIST/raw

Downloading http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz
Downloading http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz to ./data/MNIST/raw/t10k-labels-idx1-ubyte.gz
Loading widget...
Extracting ./data/MNIST/raw/t10k-labels-idx1-ubyte.gz to ./data/MNIST/raw
```

First, this is Decision Tree classification method code.

```
# Set hyperparameter grid
params_grid = {'min_samples_split': [2, 5, 10],
               'min_samples_leaf': [1, 2, 4],
               'max_leaf_nodes': [5, 10, None]}

# Train and evaluate decision tree on CIFAR-10
for depth in [3, 6, 9, 12]:
    print(f"Depth: {depth}")

    # Define decision tree with given depth
    dt = DecisionTreeClassifier(max_depth=depth)

    # Define grid search with decision tree and hyperparameter grid
    gs = GridSearchCV(dt, params_grid, cv=5, n_jobs=-1)

    # Train grid search on CIFAR-10 training set
    gs.fit(CIFAR_train_images, CIFAR_train_labels)

    # Retrieve decision tree with best hyperparameters
    dt = gs.best_estimator_

    # Predict CIFAR-10 test set labels and compute accuracy
    CIFAR_pred_labels = dt.predict(CIFAR_test_images)
    CIFAR_accuracy = accuracy_score(CIFAR_test_labels, CIFAR_pred_labels)

    # Predict CIFAR-10 training set labels and compute accuracy
    CIFAR_train_pred_labels = dt.predict(CIFAR_train_images)
    CIFAR_train_accuracy = accuracy_score(CIFAR_train_labels, CIFAR_train_pred_labels)

    print(f"CIFAR-10 Accuracy (test set): {CIFAR_accuracy:.4f}")
    print(f"CIFAR-10 Accuracy (training set): {CIFAR_train_accuracy:.4f}")
```

```
# Train and evaluate decision tree on MNIST
# Define decision tree with given depth
dt = DecisionTreeClassifier(max_depth=depth)

# Train decision tree on MNIST training set
dt.fit(MNIST_train_images, MNIST_train_labels)

# Predict MNIST test set labels and compute accuracy
MNIST_pred_labels = dt.predict(MNIST_test_images)
MNIST_accuracy = accuracy_score(MNIST_test_labels, MNIST_pred_labels)

# Predict MNIST training set labels and compute accuracy
MNIST_train_pred_labels = dt.predict(MNIST_train_images)
MNIST_train_accuracy = accuracy_score(MNIST_train_labels, MNIST_train_pred_labels)

print(f"MNIST Accuracy (test set): {MNIST_accuracy:.4f}")
print(f"MNIST Accuracy (training set): {MNIST_train_accuracy:.4f}")
```

As suggested in the assignment document, both CIFAR and MNIST were printed to see the accuracy for the training and test sets.
This is result and table.

```
Depth: 3
CIFAR-10 Accuracy (test set): 0.2394
CIFAR-10 Accuracy (training set): 0.2376
MNIST Accuracy (test set): 0.4953
MNIST Accuracy (training set): 0.4915
Depth: 6
CIFAR-10 Accuracy (test set): 0.2812
CIFAR-10 Accuracy (training set): 0.2959
MNIST Accuracy (test set): 0.7415
MNIST Accuracy (training set): 0.7382
Depth: 9
CIFAR-10 Accuracy (test set): 0.3043
CIFAR-10 Accuracy (training set): 0.3821
MNIST Accuracy (test set): 0.8504
MNIST Accuracy (training set): 0.8665
Depth: 12
CIFAR-10 Accuracy (test set): 0.3038
CIFAR-10 Accuracy (training set): 0.5188
MNIST Accuracy (test set): 0.8798
MNIST Accuracy (training set): 0.9492
```

depth		3	6	9	12
CIFAR-10	test	0.2394	0.2812	0.3043	0.3038
	training	0.2376	0.2959	0.3821	0.5188
MNIST	test	0.4953	0.7415	0.8504	0.8798
	training	0.4915	0.7382	0.8665	0.9492

In the case of CIFAR-10, the accuracy was high when the test set was depth 9 and the training set was depth 12.
In the case of MINIST, the accuracy was high when the test set was depth 12 and the training set was depth 12.

Second, this is SVM classification method code.

```
# Linear SVM
svm_linear = svm.SVC(kernel='linear')
svm_linear.fit(CIFAR_train_images, CIFAR_train_labels)

# Kernel SVM
svm_rbf = svm.SVC(kernel='rbf')
svm_rbf.fit(CIFAR_train_images, CIFAR_train_labels)
```

```
# Evaluate accuracy on CIFAR training set
linear_train_acc = accuracy_score(CIFAR_train_labels,
svm_linear.predict(CIFAR_train_images))
rbf_train_acc = accuracy_score(CIFAR_train_labels, svm_rbf.predict(CIFAR_train_images))

# Evaluate accuracy on CIFAR test set
linear_test_acc = accuracy_score(CIFAR_test_labels, svm_linear.predict(CIFAR_test_images))
rbf_test_acc = accuracy_score(CIFAR_test_labels, svm_rbf.predict(CIFAR_test_images))

# Print results in a table
print('SVM\t\tLinear\t\tKernel RBF')
print(f'Training accuracy\t{linear_train_acc:.4f}\t\t{rbf_train_acc:.4f}')
print(f'Test accuracy\t\t{linear_test_acc:.4f}\t\t{rbf_test_acc:.4f}')
```

As suggested in the assignment document, both CIFAR and MNIST were printed to see the accuracy for the training and test sets.
This is result and table.

```
SVM                 Linear          Kernel RBF
Training accuracy       0.5749          0.7028
Test accuracy           0.3754          0.5436
```

SVM	Linear	Kernel RBF
Training accuracy	0.5749	0.7028
Test accuracy	0.3754	0.5436

In the case of Training sets and Test sets, the accuracy was high when the Kernel RBF.
Among training accuracy and test accuracy, training accuracy came out higher in Linear and Kernel RBF.