**Assignment report**

2022250081 김재영

1. 개요

Phase encoding을 통해 확보된 데이터를 5개의 레이블로 분류된 데이터를 Multi-preceptron을 통해 학습시켜 test데이터가 주어지면 5개 중 하나의 결과를 출력하는 모델을 제작하였다.

2. 방식

Scikit-learn이나 tensorflow와 같은 시중에 배포된 라이브러리를 사용하지 않고, numpy, pandas, matplotlib라이브러리로만 구현했다. Sigmoid function은 은닉층에서, softmax fucntion은 출력층에서 사용하였다. 입력데이터 -> 은닉층 -> 출력층 방향으로 진행하는 Feedforward방식을 채택했으며 손실함수는 cross-entropy, 손실함수의 기울기를 계산해 가중치를 업데이트하기 위해 gradient-descend method를 사용하였다. 아래 사진은 직접 작성한 코드이다.

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

본격적으로 데이터를 입력받고 학습을 시작하기에 앞서 test data는 없고 train data만 제공받은 상태에서 정확도를 높이고자 10-fold cross validation방식을 사용했다. 또한 최적의 hyper-parameter를 계산한후, 다시 학습을 해 최적의 모델을 찾을 수 있도록 하였다.

# Load Data

import os

os.chdir(os.path.dirname(\_\_file\_\_))

print('Loading the dataset...')

dataset = pd.read\_csv("olfactory\_encoding\_train.csv")

# classify data

features = dataset.iloc[:, 1:].values  # numpy array type

labels = dataset.iloc[:, 0].values

features = features /255.0 # 0~1 normalization

#one - hot encoding

num\_classes = 5

labels\_one\_hot = np.zeros((labels.size, num\_classes))

labels\_one\_hot[np.arange(labels.size), labels] = 1

#10-fold validation

folds = 10

indices = np.arange(features.shape[0])

np.random.shuffle(indices)

fold\_size = len( indices) // folds

learning\_rates = [0.001, 0.01, 0.1]

lambdas = [0.001, 0.01, 0.1]

best\_hyperparams = None

best\_accuracy = 0

accuracies = {}

# 신경망 구조 정의

input\_size = features.shape[1]

hidden\_size = 128 #hidden layer개수

output\_size = num\_classes

for fold in range(folds):

    val\_indices = indices[fold \* fold\_size:(fold + 1) \* fold\_size]

    train\_indices = np.setdiff1d(indices, val\_indices)

    X\_train, X\_val = features[train\_indices], features[val\_indices]

    Y\_train, Y\_val = labels\_one\_hot[train\_indices], np.argmax(labels\_one\_hot[val\_indices], axis=1)

    # 가중치 및 편향 초기화

    np.random.seed(42)

    W1 = np.random.randn(input\_size, hidden\_size) \* 0.01

    b1 = np.zeros((1, hidden\_size))

    W2 = np.random.randn(hidden\_size, output\_size) \* 0.01

    b2 = np.zeros((1, output\_size))

    # 학습

    epochs = 50

    learning\_rate = 0.01

    lambda\_ = 0.001

    for epoch in range(epochs):

        # 순전파(feed-forward)

        A1, A2 = feedforward(X\_train, W1, b1, W2, b2)

        # 손실 계산

        loss = cross\_entropy(Y\_train, A2)

        # 경사 하강법으로 가중치 업데이트

        W1, b1, W2, b2 = Gradient(X\_train, Y\_train, A1, A2, W1, b1, W2, b2, learning\_rate, lambda\_)

        if epoch % 10 == 0:

            print(f"Fold {fold + 1}, Epoch {epoch}, Loss: {loss:.4f}")

    # validation

    fold\_accuracies = []

    \_, A2\_val = feedforward(X\_val, W1, b1, W2, b2)

    predictions = np.argmax(A2\_val, axis=1)

    accuracy = np.mean(predictions == Y\_val)

    fold\_accuracies.append(accuracy)

    #store results

    accuracy\_avg = np.mean(fold\_accuracies)

    accuracies[(learning\_rate, lambda\_)] = accuracy\_avg

    if accuracy\_avg > best\_accuracy:

        best\_accuracy = accuracy\_avg

        best\_params = (learning\_rate, lambda\_)

# 교차검증 결과 출력

print(f"10-Fold Cross-Validation Accuracy: {np.mean(fold\_accuracies):.2f}")

#plot results

best\_learning\_rate, best\_lambda = best\_params

print(f"Best Parameters: Learning Rate = {best\_learning\_rate}, Lambda = {best\_lambda}, accuracy={best\_accuracy:.4f}")

# Train with best hyperparameters and plot training loss

np.random.seed(42)

W1 = np.random.randn(input\_size, hidden\_size) \* 0.01

b1 = np.zeros((1, hidden\_size))

W2 = np.random.randn(hidden\_size, output\_size) \* 0.01

b2 = np.zeros((1, output\_size))

train\_losses = []

for epoch in range(epochs):

    A1, A2 = feedforward(X\_train, W1, b1, W2, b2)

    loss = cross\_entropy(Y\_train, A2)

    train\_losses.append(loss)

    W1, b1, W2, b2 = Gradient(X\_train, Y\_train, A1, A2, W1, b1, W2, b2, best\_learning\_rate, best\_lambda)

plt.figure(figsize=(10, 6))

plt.plot(range(epochs), train\_losses, label="Training Loss")

plt.xlabel("Epochs")

plt.ylabel("Loss")

plt.title("Training Loss over Epochs")

plt.legend()

plt.show()

# Evaluate on test set

\_, A2\_test = feedforward(features[indices[-fold\_size:]], W1, b1, W2, b2)

predictions = np.argmax(A2\_test, axis=1)

true\_labels = np.argmax(labels\_one\_hot[indices[-fold\_size:]], axis=1)

accuracy = np.mean(predictions == true\_labels)

print(f"Test set accuracy with best hyperparameters: {accuracy:.4f}")

# Output the best weights

print("Best weights:")

print("W1:", W1)

print("b1:", b1)

print("W2:", W2)

print("b2:", b2)

이에 따라 learning rate = 0.01, Lambda= 0.001, accuracy = 0.3267의 최적의 결과를 찾을 수 있었다.

아래 그래프는 50epochs를 실행할 동안의 loss의 변화를 나타낸 것이다.

스크린샷, 텍스트, 그래프, 라인이(가) 표시된 사진

자동 생성된 설명