AI 라이브러리 활용 기말고사 2024-2

1. CNN 알고리즘에 구체적으로 대하여 설명하시오. (20점)
먼저 Convolutional Neural Network는 합성곱 신경망이라고도 불리며 이미지, 영상, 음성 등의 데이터를 처리하기 위해 설계된 딥러닝 알고리즘이다. CNN의 핵심은 데이터를 처리하는 과정에서 입력 데이터의 공간적 구조를 보존하고 특징을 추출하는 것이다. 이 알고리즘은 주로 합성곱 층, 풀링 층, 활성화 함수, 완전 연결 층으로 구성되며, 각각 고유의 역할을 통해 데이터를 효과적으로 학습한다. 합성곱 층에서는 필터 또는 커널를 통해 입력 데이터의 패턴을 학습하며, 이는 가장 기본적인 단계로 이미지의 엣지, 텍스처 등 국소적인 특성을 감지한다. 이 과정에서 필터는 작은 크기의 행렬 형태로 적용되며, 필터 이동 간격과 패딩을 조정하여 출력 크기를 제어한다. 활성화 함수는 합성곱 연산 후 비선형성을 추가하는 역할을 하며, 주로 ReLU 함수를 사용하여 음수를 0으로 변환하고 연산을 단순화하면서도 비선형 특성을 유지한다. 풀링 층은 데이터의 크기를 줄이면서도 중요한 특징을 보존하기 위해 사용되며, 일반적으로 Max Pooling 또는 Average Pooling이 적용된다. 이를 통해 계산량이 감소하고 모델의 과적합을 방지할 수 있다. CNN의 마지막 단계는 완전 연결 층으로, 앞에서 추출한 특징을 기반으로 최종적인 분류나 예측을 수행한다. 이 과정에서 특징을 벡터 형태로 변환하고 각 클래스에 대한 확률값을 출력한다. CNN은 이미지 분류, 객체 검출, 얼굴 인식, 자율주행, 의료 영상 분석 등 다양한 응용 분야에서 뛰어난 성능을 보여준다. 또한, CNN은 필터 공유를 통해 학습 파라미터 수를 크게 줄이고, 공간적 구조를 효과적으로 활용할 수 있어 높은 효율성을 자랑한다. 그러나 대규모 데이터와 고성능 하드웨어가 필요하며, 최적의 구조를 설계하기 위해 많은 실험이 필요하다는 점은 단점이라고 할 수 있다.

2. 과적합 및 과적합을 피할 수 있는 방법을 제시하시오. (20점)
먼저 과적합(Overfitting)의 정의는 모델이 학습 데이터에 너무 특화되어 새로운 데이터에 대한 일반화 능력을 잃는 현상이라고 할 수 있다. 과적합은 학습 데이터의 노이즈나 세부적인 패턴까지 과도하게 학습한 결과로, 테스트 데이터나 실제 환경에서 성능 저하를 초래한다. 과적합은 주로 학습 데이터가 부족하거나 불균형할 때, 모델이 지나치게 복잡할 때, 또는 학습 과정이 너무 길어질 때 발생한다.
과적합을 방지하기 위해 다양한 방법을 사용할 수 있다. 첫째, 더 많은 데이터를 확보하거나 데이터 증강을 통해 기존 데이터를 변형하여 학습 데이터를 다양화한다. 둘째, L1 또는 L2 정규화와 같은 정규화 기법을 사용하여 모델의 복잡도를 줄이고 가중치에 제약을 부여한다. 셋째, Dropout을 활용하여 학습 과정에서 일부 뉴런의 출력을 랜덤하게 제외함으로써 네트워크의 의존성을 낮춘다. 넷째, 교차 검증을 통해 데이터를 여러 번 나누어 학습하고 평가함으로써 하이퍼파라미터를 최적화하고 과적합 여부를 확인한다. 다섯째, 학습 데이터를 검증 데이터와 테스트 데이터로 나누어 모델의 성능을 객관적으로 평가하며, 조기 종료를 통해 검증 손실이 증가하기 시작하는 시점에서 학습을 멈춘다. 여섯째, 배치 정규화를 사용하여 층별 입력 값을 정규화하고, 학습 속도를 높이며 과적합을 줄인다. 일곱째, 모델 구조를 단순화하여 필요 이상의 층이나 노드를 줄여 적절한 복잡도를 유지한다. 여덟째, 학습 데이터에 노이즈를 추가하거나 앙상블 기법을 사용하여 여러 모델의 예측을 결합함으로써 과적합의 영향을 완화한다. 해당 방법을 적절히 조합하면 과적합 문제를 해결할 수 있다.

3. MNIST 데이터와 CNN 알고리즘을 이용하여 데이터를 0~9 까지 분리하고 에포크는 40회로 지정하고, 5회 이상 성능이 개선되지 않으면 학습을 자동 중단하는 코드를 작성하시오. (30점)

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.datasets import mnist
from tensorflow.keras.utils import to_categorical

(x_train, y_train), (x_test, y_test) = mnist.load_data()

x_train = x_train.reshape(-1, 28, 28, 1).astype("float32") / 255.0
x_test = x_test.reshape(-1, 28, 28, 1).astype("float32") / 255.0
y_train = to_categorical(y_train, 10)
y_test = to_categorical(y_test, 10)

model = Sequential([
    Conv2D(32, (3, 3), activation = 'relu', input_shape=(28, 28, 1)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation = 'relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation = 'relu'),
    Dropout(0.5),
    Dense(10, activation = 'softmax')
])

model.compile(optimizer = 'adam',
              loss = 'categorical_crossentropy',
              metrics = ['accuracy'])

early_stopping = EarlyStopping(monitor = 'val_loss', patience = 5, restore_best_weights = True)

history = model.fit(
    x_train, y_train,
    epochs = 40,
    batch_size = 128,
    validation_split = 0.2,
    callbacks = [early_stopping],
    verbose = 1
)

test_loss, test_acc = model.evaluate(x_test, y_test, verbose = 2)
print(f"\nTest Accuracy: {test_acc:.4f}")
```

4. 피마 인디언 데이터를 가지고 (30점)

1) Plasma 와 당뇨병의 발병 여부를 알아보고 그 확률을 구하시오.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score

# Load
file_path = 'pima-indians-diabetes.csv'
data = pd.read_csv(file_path)
data.columns = [
    "Pregnancies", "Glucose", "BloodPressure", "SkinThickness",
    "Insulin", "BMI", "DiabetesPedigreeFunction", "Age", "Outcome"
]

# 1) Plasma 와 당뇨병의 발병 여부를 알아보고 그 확률을 구하시오.
glucose_outcome_prob = data.groupby('Glucose')['Outcome'].mean()
probability_table = glucose_outcome_prob.reset_index()
probability_table.columns = ['Glucose Level', 'Diabetes Probability']
pd.set_option('display.max_rows', None)
glucose_prob_table
print(probability_table)
```

2) 상관관계를 나타내는 히트맵을 그리시오.

```
# 2) 상관관계를 나타내는 히트맵을 그리시오.
correlation_matrix = data.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Correlation Heatmap")
plt.show()
```

3) 당뇨병을 예측하시오.

```
# 3) 당뇨병을 예측하시오.
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report

X = data.drop("Outcome", axis=1)
y = data["Outcome"]

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

param_grid = {
    'C': [0.01, 0.1, 1, 10, 100],
    'solver': ['lbfgs', 'liblinear']
}

grid_search = GridSearchCV(LogisticRegression(max_iter=1000, random_state=42), param_grid, cv=5, scoring='accuracy')
grid_search.fit(X_scaled, y)

best_model = grid_search.best_estimator_

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42, stratify=y)
best_model.fit(X_train, y_train)

y_pred = best_model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
classification_report_output = classification_report(y_test, y_pred)

print("Model Accuracy:", accuracy)
print("\nClassification Report:\n", classification_report_output)
```