기계학습 과제#3

4조(이희구, 유제우)

1) 기계 학습에서 학습이란 무엇인지를 정리하시오(2점).
(가중치, 손실함수가 무엇인지를 정리하고, 데이터, 가중치, 손실함수를 이용하여 학습이 무엇인지를 정리함.)

- **학습**이란 훈련 데이터로부터 가중치 매개변수의 최적값을 자동으로 획득 하는 것

- 기계 학습 모델은 **데이터**를 기반으로 학습한다. 데이터는 입력 변수와 그에 상응하는 정답(Label)으로 구성된다.

- **가중치** 매개변수는 모델이 학습 데이터로부터 지식을 습득하고 일반화하는데 중요한 역할을 한다. 가중치는 각 신호의 영향력을 조절하는 매개변수이다. 예를 들어, 고양이와 강아지를 구분하는 모델에서는 귀의 형태, 눈의 크기 등의 특성에 대한 가중치가 있을 것이다. 학습 알고리즘이 이러한 가중치를 조정하여 모델의 성능을 최적화한다.

- **손실함수**는 모델의 예측값과 실제 타깃값 사이의 차이를 측정하는 함수이다. 모델이 얼마나 정확하게 예측하는지를 평가하고 모델을 학습시킬 때 사용된다. 손실 함수의 목표는 최소화하는 것이며, 모델의 파라미터를 조정하면서 손실 함수를 최소화하려고 한다. 손실함수는 회귀문제에서 실제 값과 모델 예측 값 사이의 제곱 오차의 평균을 계산하는 평균 제곱 오차에 사용된다.

- **학습**이란 훈련 데이터로부터 가중치 매개변수의 최적값을 자동으로 획득 하는 것이다. 이 과정에서 모델은 데이터를 입력으로 받아 손실 함수를 통해 모델의 예측과 실제 타깃 간의 오차를 계산하고, 이 오차를 최소화하기 위해 가중치를 조정한다. 목표는 학습 데이터에 대한 최적의 일반화를 실현하기 위해 최적의 가중치와 편향을 찾아내는 것이다. 이를 통해 모델은 새로운 입력 데이터에 대한 예측을 수행할 수 있게 되며, 학습 데이터로부터 추출한 패턴과 지식을 활용하여 다양한 작업을 수행할 수 있다.

2) 확률적 경사 하강법의 소스 코드를 분석하시오(2점).

※ 먼저 저희는 확률적 경사 하강법의 코드를 분석하기 전에 확률적 경사 하강법 사용 이유에 대해 조사했습니다.

[확률적 경사 하강법을 사용하는 이유]

- 확률적 경사 하강법은 기계 학습에서 사용되는 최적화 알고리즘 중 하나이다.

- 확률적 경사 하강법은 전체 데이터셋을 사용하여 모델을 업데이트하는 것보다 빠르게 최적화를 수행할 수 있다. 또한, 전체 데이터셋을 한 번에 처리하는 것보다 각 데이터 포인트를 하나씩 처리하는 것이 계산적으로 효율적이다. 대용량의 데이터셋을 다룰 때 전체 데이터를 메모리에 올리는 것은 어려울 수 있다. 확률적 경사 하강법은 각 데이터 포인트를 하나씩 처리하므로, 메모리를 효율적으로 활용할 수 있다. 확률적 경사 하강법은 각 단계에서 무작위로 데이터 포인트를 선택하기 때문에 지역 최소값에 갇히는 상황을 피할 수 있다. 이는 전체 데이터셋을 기반으로 한 경사 하강법보다 더 넓은 영역에서 최적화를 수행할 수 있게 해준다. 새로운 데이터가 들어올 때마다 바로 모델을 업데이트할 수 있다. 이는 실시간으로 변화하는 환경에서 모델을 유지하기에 유용하다. 대규모 데이터셋을 처리할 때 전체 데이터셋을 한 번에 사용하는 것은 비효율적일 수 있다. 확률적 경사 하강법은 이러한 상황에서도 효과적으로 사용될 수 있다. 데이터 분포가 시간에 따라 변할 수 있는 경우, 확률적 경사 하강법을 사용하여 모델을 지속적으로 업데이트할 수 있다. 확률적 경사 하강법은 이러한 이유로 기계 학습 알고리즘에서 널리 사용되며, 특히 대용량 데이터셋이나 실시간 데이터 스트림에서 모델을 효과적으로 최적화하는데 적합하다.

[소스 코드]

```
n_epochs = 50
t0, t1 = 5, 50  # 학습 스케줄 하이퍼파라미터

def learning_schedule(t):
    return t0 / (t + t1)

theta = np.random.randn(2,1)  # 랜덤 초기화

for epoch in range(n_epochs):
    for i in range(m):
        random_index = np.random.randint(m)
        xi = X_b[random_index:random_index+1]
        yi = y[random_index:random_index+1]
        gradients = 2 * xi.T.dot(xi.dot(theta) - yi)
        eta = learning_schedule(epoch * m + i)
        theta = theta - eta * gradients
```

[소스 코드 분석]

n_epochs = 50: 전체 데이터셋을 몇 번 반복해서 학습할 것인지를 결정하는 변수이다. 여기서는 50번 반복한다.

t0, t1 = 5, 50: 학습 스케줄 하이퍼파라미터이다. learning_schedule 함수에서 사용된다.

def learning_schedule(t): 학습 스케줄을 정의한 함수이다. t는 현재까지의 반복 횟수를 나타낸다. t0 / (t + t1)의 형태로 학습률이 감소하게 된다. 초기에는 학습률이 크게 설정되고, 시간이 지남에 따라 조금씩 줄어들게 된다.

theta = np.random.randn(2,1): 랜덤 초기화된 가중치 벡터 theta를 생성한다. 여기서는 2x1 크기의 벡터로 초기화한다.

for epoch in range(n_epochs):: 주어진 반복 횟수(n_epochs)동안 반복한다.

for i in range(m):: 전체 데이터셋의 크기(m)만큼 반복한다.

random_index = np.random.randint(m): 무작위로 데이터 포인트를 선택한다.

xi = X_b[random_index:random_index+1]와 yi = y[random_index:random_index+1]: 선택한 데이터 포인트의 입력 변수와 타깃을 가져온다. 여기서 xi는 입력 변수를 포함한 벡터이며, yi는 해당 데이터 포인트의 타깃 값이다.

gradients = 2 * xi.T.dot(xi.dot(theta) - yi): 확률적 경사 하강법의 그래디언트를 계산한다. 이는 비용 함수의 편도함수를 계산하는 과정이다. 그래디언트는 현재 예측과 실제 값의 차이에 입력 변수를 곱한 것이다.

eta = learning_schedule(epoch * m + i): 학습률(eta)을 learning_schedule 함수를 통해 계산한다. 시간이 지남에 따라 학습률이 조절되어, 초기에는 크게 학습하고 나중에는 조금씩 학습한다.

theta = theta - eta * gradients: 경사 하강법 단계를 수행하여 가중치(theta)를 업데이트한다. 새로운 가중치는 이전 가중치에서 학습률과 그래디언트의 곱을 뺀 값이다.

이 과정은 모델을 여러 번 반복해서 업데이트하며, 확률적 경사 하강법을 통해 모델이 최적의 가중치를 찾아가게 된다.