Softmax Regression

상태	Done
작성자	김 김민준
최종 편집 일시	@2023년 12월 9일 오후 7:16
태그	

Softmax Regression

다중 클래스 분류: 세 개 이상의 선택지 중 하나를 고르는 문제

샘플 데이터에 대한 예측값으로 모든 가능한 정답지에 대해 정답일 확률의 합이 1이 되도록 구하는데 사용할 수있는 것이 이 소프트맥스 함수이다.

소프트맥스 함수는 class의 총 개수를 k라고 할 때 k차원의 벡터를 입력을 받아서 각 클래스에 대한 확률을 추정한다. 일단 수식을 보면

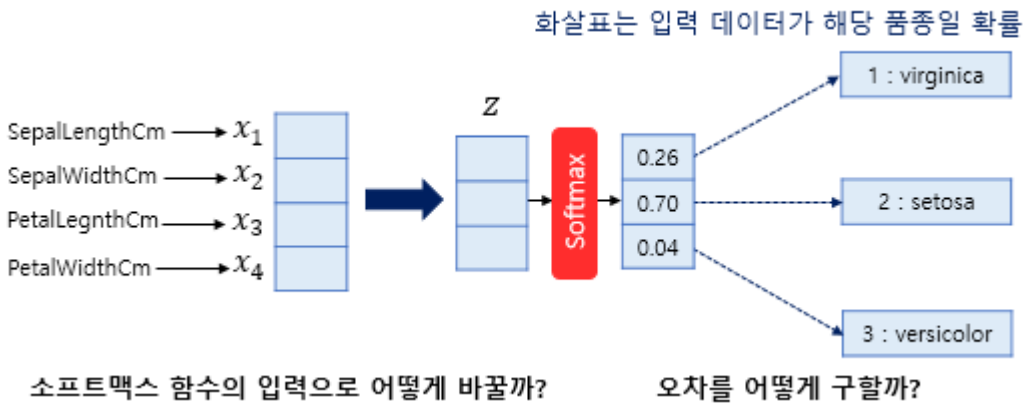
k차원의 벡터에서 i번째의 원소를 zi, i번째 클래스가 정답일 확률을 pi로 나타낸다면 소프트맥스 함수는 위의 수식과 같이 정의한다.

만약 3차원 벡터의 입력을 받으면 소프트맥스 함수는 아래와 같은 수식이 나온다.

$$softmax(z) = [\frac{e^{z_1}}{\sum_{j=1}^{3} e^{z_j}} \ \frac{e^{z_2}}{\sum_{j=1}^{3} e^{z_j}} \ \frac{e^{z_3}}{\sum_{j=1}^{3} e^{z_j}}] = [p_1, p_2, p_3] = \hat{y} = \text{예측값}$$

$$softmax(z) = [\frac{e^{z_1}}{\sum_{j=1}^{3} e^{z_j}} \ \frac{e^{z_2}}{\sum_{j=1}^{3} e^{z_j}} \ \frac{e^{z_3}}{\sum_{j=1}^{3} e^{z_j}}] = [p_1, p_2, p_3] = \hat{y} = \text{예측값}$$

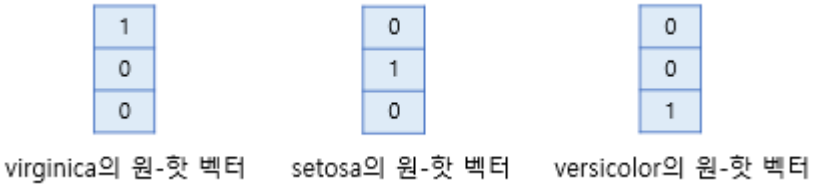
p1,p2,p3은 각각의 클래스가 정답일 확률인데 각각 0과 1사이의 값으로 나타나지고 총 합은 1이 된다.



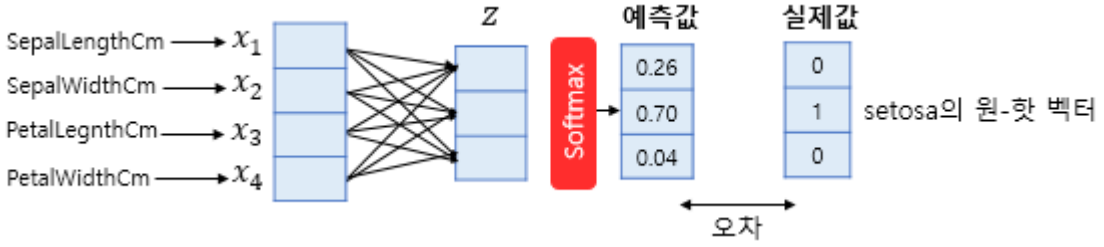
그림으로 그리면 이러한 형태로 나오는데 샘플 데이터를 1개씩 입력으로 받아 처리한다면 하나의 샘플 데이터는 4개의 독립변수 x를 가지는데 이것은 모델이 4차원 벡터의 입력을 받았음을 의미한다. 하지만 함수의 입력으로 사용되는 벡터는 벡터의 차원들이 분류를 하고자하는 클래스의 개수가 되어야 하므로 가중치 연산을 거쳐 3차원 벡터로 변환되어야 한다.

축소하는 방법은 소프트맥스 함수의 입력 벡터 z의 차원 수 만큼 결과값이 나오도록 가중치의 곱을 한다. 그림을 보면 화살표는 4 x 3=12 개이며 전부 다른 가중치를 가지며 학습 과정에서 점차적으로 오차를 최소화 하는 가중치로 값이 변하게 된다.

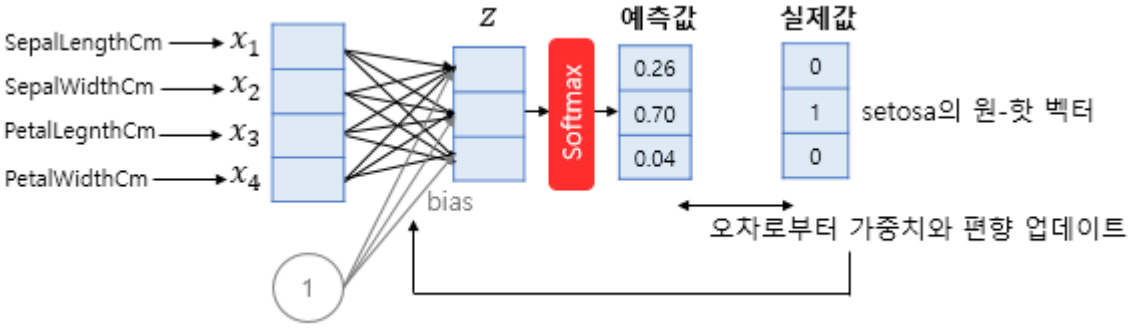
그 다음은 오차계산에 대한 것이다. 이 함수의 출력으로는 분류하고자 하는 클래스의 수 만큼 차원을 가지는 벡터로 각각의 원소는 0과 1 사이의 값을 가지는 데 각각의 값들은 특정한 클래스가 정답일 확률을 나타낸다. 각각의 예시로 각각의 원소들이 정답일 확률들이다. 이러한 예측 값과 비교를 할 수 있는 실제 값의 표현 방법이 있어야 하는데 소프트맥스 회귀에서는 실제값을 원 핫 벡터로 표현을 한다.



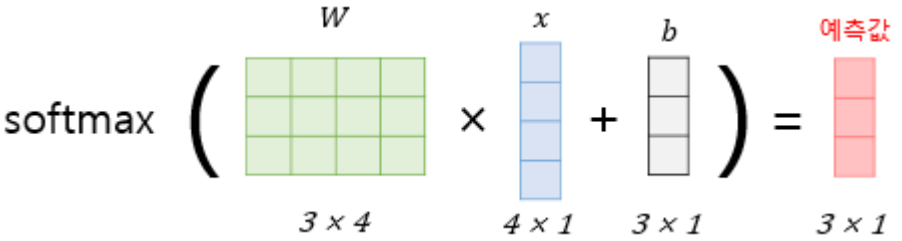
위 의 그림은 실제로 각각의 원소가 정답일 확률을 의미한다면 원 핫 인코딩을 수행을 하여서 실제 값을 원 핫 벡터로 수치화 한 것이다.



예를 들어 샘플 데이터의 실제 값이 두번째 원소라면 원-핫벡터는 [0 1 0]이다. 이 경우에는 예측값과 실제값의 오차가 0이 되는 경우는 소프트맥스 함수의 결과가 [0 1 0]이 되게 된다. 이 두 벡터의 오차를 계산하기 위해서 소프트맥스 회귀는 비용 함수로 크로스 엔트로피 함수를 사용한다.



선형회귀나 로지스틱 회귀와 같이 편향의 대상이 되는 매개변수를 사용을 한다.



x를 특성의 수만큼 차원을 가진 입력 벡터이고 w는 가중치 행렬 b를 편향이라고 했을 때 소프트맥스 회귀에서 예측값을 구하는 과정을 연산으로 표현하면 위의 그림과 같다.

아래는 위에 잠깐 언급한 크로스 엔트로피 함수에 대한 내용이다.

여기서 y는 실제 값을 나타내며 k는 클래스의 개수이다. yj는 실제값 원 핫 벡터의 j번째 인덱스를 의미하고 pj는 샘플데이터가 j번째 클래스일 확률을 나타낸다.
c가 만약 실제값 원 핫 벡터에서 1을 가진 원소의 인덱스라고 하면 pc = 1은 pj가 yj를 정확하게 예측한 경우가 된다. 식을 대입해보면 -1log(1) = 0이 되기 때문에 pj가 yj를 정확하게 예측한 경우의 크로스 엔트로피 함수의 값은 0이 된다. 그렇기에 위의 함수의값이 최소화 되는 방향으로 학습을 해야한다. n개의 전체 데이터에 대한 평균을 구한다고 한다면 수식은 아래와 같다

다음으로는 코드로 구현한 소프트맥스 함수이다.

```
X = torch.tensor([[1.0, 2.0, 3.0], [4.0, 5.0, 6.0]])
X.sum(0, keepdims=True), X.sum(1, keepdims=True)
```

```
(tensor([[5., 7., 9.]]),
 tensor([[ 6.],
         [15.]]))
```

x 텐서의 열(axis = 0)별로 합을 계산한다. 결과를 차원을 유지한 채로 반환한다. 1행 3열의 텐서로 각 열의 합을 표현한다.

x 텐서의 열(axis = 1)별로 합을 계산한다. 결과를 차원을 유지한 채로 반환한다. 1행 3열의 텐서로 각 열의 합을 표현한다.

```
def softmax(X):
    X_exp = torch.exp(X)
    partition = X_exp.sum(1, keepdims=True)
    return X_exp / partition  # The broadcasting mechanism is
```

소프트맥스 함수의 구현

입력 텐서 x의 각 원소들을 지수함수로 변환을 하여서 x_exp 텐서에 저장을 한다. 각 원소의 지수값을 계산한다.

x_exp의 각 행(axis = 1)별로 합을 계산하여 partition이라는 텐서에 저장을 한다. 이 합은 소프트맥스 함수의 분모 역할을 한다. 결과를 차원을 유지하도록 설정을 한다.

각 원소를 해당 행(axis = 1)의 합으로 나누어준다.

```
X = torch.rand((2, 5))
X_prob = softmax(X)
X_prob, X_prob.sum(1)
```

```
(tensor([[0.1242, 0.2362, 0.1378, 0.2525, 0.2493],
         [0.1234, 0.1800, 0.2747, 0.1947, 0.2272]]),
 tensor([1., 1.]))
```

2행 5열의 랜덤한 값을 가진 텐서 x를 생성을 한다.

softmax함수를 사용해서 x에 대한 확률 분포를 얻는다.

소프트맥스함수 적용하여 출력. x_prob의 각 행별 합을 구하여 각 행의 합을 확인 각 행의 합은 1에 가까워져야하기에 각 행마다 거의 1에 가까운지를 확인.

```
class SoftmaxRegressionScratch(d2l.Classifier):
    def __init__(self, num_inputs, num_outputs, lr, sigma=0.0
        super().__init__()
        self.save_hyperparameters()
        self.W = torch.normal(0, sigma, size=(num_inputs, num
                              requires_grad=True)
        self.b = torch.zeros(num_outputs, requires_grad=True)

    def parameters(self):
        return [self.W, self.b]
```

Softmax 회귀 분류기 구현

d2l.classifier을 상속을 하는 분류 모델을 구현하는데 필요한 유틸리티 클래스이다.

메서드에서 모델을 초기화를 한다. num_inputs은 입력 특성의 수이고 num_ouputs은 클래스의 수이다. lr은 학습률을 의미를 하고 sigma는 초기화를 할 가중치의 표준편차이다.

상위 클래스의 생성자를 호출한다.

pytorch lighting에서 사용이 되는 함수로 현재 클래스의 하이퍼 파라미터들을 저장을 한다. 이것은 학습 중에 사용이 된 하이퍼 파라미터를 추적하고 기록한다.

평균이 0이고 표준편차가 sifma인 정규분포를 따르는 가중치 행렬인 w를 초기화를 한다. 이 행렬은 입력 특성과 출력 클래스간의 가중치를 나타낸다. 역전파를 통하여 가중치가 학습이 되는 것이다.

편향 벡터를 초기화를 한다. 각 클래스의 편향을 나타내는데 이 값도 역전파를 통하여 학습이 된다.

이 매서드는 모델의 학습가능한 매개변수를 반환한다. self.w와 self.b를 반환을 하여서 모델 학습에 사용이 된다.

```
@d2l.add_to_class(SoftmaxRegressionScratch)
def forward(self, X):
    X = X.reshape((-1, self.W.shape[0]))
    return softmax(torch.matmul(X, self.W) + self.b)
```

해당 클래스에 새로운 매서드 또는 속성을 추가를 한다.

주어진 입력 x를 받아 모델을 통과시켜서 결과를 반환한다.

입력 . x를 행렬곱 연산을 하기에 적절하게 모양을 바꾼다.

입력 x와 모델의 가중치 및 편향을 이용을 해서 소프트맥스 함수를 적용한 결과를 반환한다.

다음으로는 크로스 엔트로피 함수를 구현한다.

```
y = torch.tensor([0, 2])
y_hat = torch.tensor([[0.1, 0.3, 0.6], [0.3, 0.2, 0.5]])
y_hat[[0, 1], y]
```

```
tensor([0.1000, 0.5000])
```

y는 인덱스로 사용될 텐서이다.

y_hat은 확률 값으로 이루어진 텐서이다. 각각의 행들은 각각의 샘플에 대한 클래스의 확률을 나타낸다,

y에 있는 값들을 인덱스로 사용을 하여서 y_hat에서 해당 위치의 값을 선택한다.

- `y_hat[0, 0]`: 첫 번째 행과 열의 값, 즉 `0.1`
- `y_hat[1, 2]`: 두 번째 행과 세 번째 열의 값, 즉 `0.5`

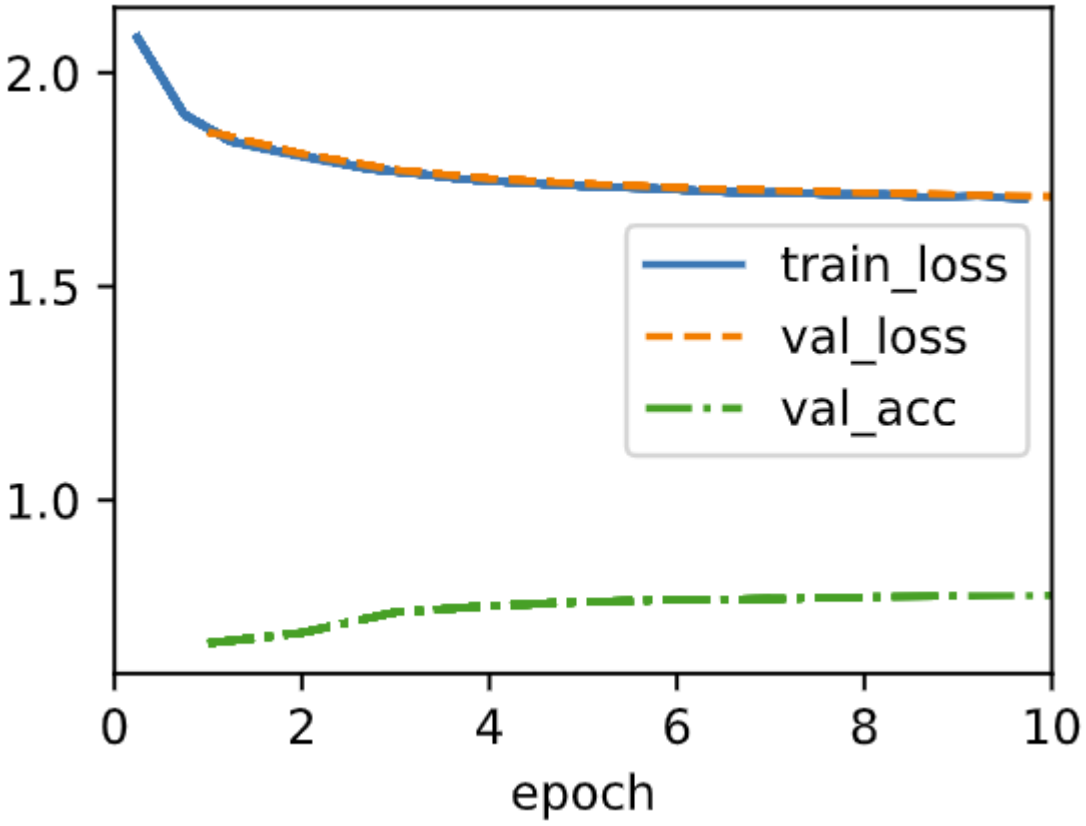
```
def cross_entropy(y_hat, y):
    return -torch.log(y_hat[list(range(len(y_hat))), y]).mean

cross_entropy(y_hat, y)
```

```
tensor(1.4979)
```

크로스 엔트로피 손실을 계산하는 함수이다.

y를 이용을 하여서 y_hat 에서 해당하는 위치의 값을 선택한다. 실제 레이블에 해당하는 예측값을 가져오게 되고 선택된 값에 로그를 취한다. 그다음 부호를 바꾸어서 음수를 취하고 값들에 대한 평균을계산을 한다.

```
data = d2l.FashionMNIST(batch_size=256)
model = SoftmaxRegressionScratch(num_inputs=784, num_outputs=
trainer = d2l.Trainer(max_epochs=10)
trainer.fit(model, data)
```



소프트맥스 회귀 모델을 학습 하는 코드이다.

data를 d2l 라이브러리를 사용을 해서 fashionMNIST 데이터 셋을 불러와 배치크기가 256인 데이터 로더로 하여 생성을 한다.

model을 784개의 입력특성 10개의 출력 클래스, 학습률 0.1로 한다.

trainer를 모델을 학습하기 위한 객체로 하는데 최대 10번의 에포크 동안 학습을 수행을 한다.

fit 매서드를 사용을 해서 모델을 학습을 시킨다.

이제 분류를 해보자

```
X, y = next(iter(data.val_dataloader()))
preds = model(X).argmax(axis=1)
preds.shape
```

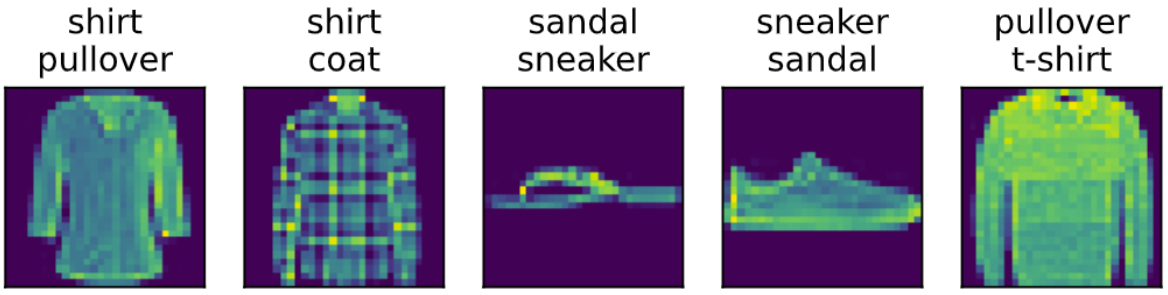
```
torch.Size([256])
```

데이터 셋으로 부터 batch 데이터를 가져온다.

predas를 입력 데이터인 x를 사용을 하여서 모델을 입력하고 예측 값을 계산을 한 값으로 한다.

shape를 통하여 해당 텐서의 형태를 반환을 하는데 예측된 클래 레이블의 개수가 반환이 된다.

```
wrong = preds.type(y.dtype) != y
X, y, preds = X[wrong], y[wrong], preds[wrong]
labels = [a+'\n'+b for a, b in zip(
    data.text_labels(y), data.text_labels(preds))]
data.visualize([X, y], labels=labels)
```



예측된 레이블과 실제 레이블을 비교하여서 잘못 예측이 된 샘플을 찾고 wrong에 저장을 한다.

그렇게 선택이 된 샘플들만을 선택을 하여서 X, y, preds에 저장을 한다.

각 샘플에 대한 실제 레이블과 예측된 레이블을 함께 표시할 레이블을 생성을한다.

잘못 예측이 된 샘플들을 시각화를 하여서 확인을 한다. X를 이미지로 y를 실제 레이블로 labels를 예측된 레이블로 보여준다.