4장 신경망 학습

학습 : 훈련 데이터로부터 가중치 매개변수의 최적값을 자동으로 획득하는 것

이미지에서 특징을 추출, 그 특징의 패턴을 기계학습 기술로 학습

⇒ 신경망은 주어진 데이터 그대로를 입력 데이터로 활용해 'end-to-end'로 학습할 수 있음

손실 함수 : 신경망이 최적의 매개변수 값을 탐색하는데에 사용하는 지표

- y_k : 신경망의 출력 (소프트맥스 함수의 출력 ⇒ 확률로도 해석 가능)
- t_k : 정답 레이블 (정답에 해당하는 인덱스 원소만 1, 나머지 0 - 원핫인코딩)

- 오차제곱합 : 각 원소의 출력값과 정답값의 차를 제곱한 후, 그 총합을 구함

  $E = (1/2)\sum_k (y_k - t_k)^2$
- 교차 엔트로피 오차 : **정답일 때의 출력이 전체 값을 결정함**
  - 정답에 해당하는 출력이 커질수록 0, 그 출력이 1일 때 0이 됨, 정답일 때의 출력이 작아질수록 오차는 커짐

  $E = -\sum_k t_k log y_k$

교차 엔트로피

- 정답 레이블이 원-핫 인코딩이 아닐때
- 배치용도 가능하게끔

```
def cross_entropy_error(y,t):
    if y.dim==1:
        t=t.reshape(1, t.shape)
        y=y.reshape(1, y.size)

    batch_size=y.shape[0]
    return -np.sum(np.log(y[np.arange(batch_size),t] + 1e-7))
```

`np.log(y[np.arange(batch_size),t] + 1e-7)`

1. 0부터 batch_size-1 까지 배열 생성
2. t에는 레이블이 저장되어있으므로,
3. 각 데이터의 정답 레이블에 해당하는 신경망의 출력을 추출!

'정확도'라는 지표를 놔두고 '손실 함수의 값'이라는 우회적인 방법을 택하는 이유?

- 미분 때문!
- 정확도는 미분 값이 대부분의 장소에서 0이 되어 매개변수를 갱신할 수 없기 때문
  - 대부분의 장소에서 0이 되는 이유?

    정확도가 지표라면 가중치 매개변수를 약간만 조정해서는 개선되지 않고 일정하게 유지됨

    개선되어도 연속적인 변화가 아닌 33%, 34%이런식으로 불연속적인 띄엄띄엄한 값으로 바뀜

    ⇒ 불연속적으로 갑자기 변화 like 계단 함수

    → 그에 비해 손실함수 값은 연속적으로 변화, 출력도 연속적, 곡선의 기울기도 연속적

미분 = 특정 순간의 변화량, 한 순간의 변화량

- h를 무한히 0으로 좁히는 것이 불가능함 → 개선으로 중앙 차분을 사용

```
def numerical_diff(f,x):
    h=1e-4
    return (f(x+h)-f(x-h)) / (2*h)
```

(f(x+h)-f(x)) / h 의 개선방향으로 중앙 차분이 나온 건데, 수치 미분이 아무리 근사치로 미분을 계산하는 방법이라해도, 얘도 똑같이 0으로 좁히는 것이 불가능한 것 아닐까?

편미분

- 변수 2개 이상일 때, 목표 변수 하나에 초점을 맞추고 다른 변수는 값을 고정
- 목표 변수를 제외한 나머지를 특정 값에 고정하기 위해 새로운 함수를 정의

```
def f_tmp1(x0):
    return x0*x0 + 4.0**2.0

numerical_diff(f_tmp1, 3.0)
```

→ 이런식으로

x0, x1의 편미분을 동시에 계산하고 싶다면??

⇒ 모든 변수의 편미분을 벡터로 정리한 것을 **기울기** 라고 함

```
def function_2(x):
    return x[0]**2+x[1]**2

def numerical_gradient(f,x):
    h=1e-4
    grad=np.zeros_like(x) # x와 형상이 같은 배열을 생성

    for idx in range(x,size):
        tmp_val=x[idx]
        # f(x+h) 계산
        x[idx]=tmp_val+h
        fxh1 = f(x)

        # f(x-h) 계산
        x[idx]=tmp_val-h
        fxh2=f(x)

        grad[idx]=(fxh1-fxh2)/(2*h)
        x[idx]=tmp_val # 값 복원

    return grad

    print(numerical_gradient(function_2,np.array([3.0,4.0])))
    #[6. 8.]
```

기울기가 가리키는 것 = **각 장소에서 함수의 출력 값을 가장 크게 줄이는 방향!!**

학습 단계에서 최적의 매개변수를 찾아냄

→ 최적의 매개변수 = 손실 함수가 최솟 값이 될 때의 배개변수 값

⇒ 기울기를 이용해 함수의 최솟값(가능한 한 작은 값)을 찾는 게 경사법

경사법

주의할 점 : 각 지점에서 함수의 값을 낮추는 방안을 제시하는 지표가 기울기인데, 기울기가 가리키는 곳에 정말 함수의 최솟값이 있는지, 나아갈 방향인지는 보장 불가함

- 실제로 복잡한 함수에서는 기울기가 가리키는 방향에 최솟값이 없는 경우가 대부분

기울어진 방향이 꼭 최솟값을 가리키진 않지만, 그 방향으로 가야 함수의 값을 줄일 수 있음!!

함수의 값을 점차 줄이는 것이 경사법 gradient method

학습률

- 한 번의 학습으로 얼마만큼 학습해야할지 = 매개변수 값을 얼마나 갱신하느냐를 정함

```
def gradient_descent(f, init_x, lr=0.01, step_num=100):
    x=init_x

    for i in range(step_num):
        grad=numerical_gradient(f,x)
        x-=lr*grad
    return x
```

→ 갱신 처리 step_num만큼 반복

신경망의 가중치 매개변수는 훈련 데이터와 학습 알고리즘에 의해 '자동'으로 획득되는 매개변수

BUT 학습률 같은 하이퍼파라미터는 사람이 직접 설정해야함

신경망에서의 기울기

- 신경망 학습에서도 기울기 구해야함 ⇒ 가중치 매개변수에 대한 손실 함수의 기울기