4. 신경망 학습

- **학습** : 훈련데이터로부터 가중치 매개변수의 최적값을 자동으로 획득하는 것
- **손실함수** : 신경망이 학습할 수 있도록 해주는 지표
- **학습의목표** : 손실함수의 결과값을 가장 작게 만드는 가중치 매개변수의 값을 찾는 것

4.1 데이터에서 학습한다

- 매개변수의 수가 많고 신경망의 층이 깊어지면 사람의 손으로 매개변수를 정하기 쉽지 않다.
- **퍼셉트론 수렴 정리** : 선형으로 분리 가능한 문제는 유한번의 학습을 통해 풀이가 가능하다는 정리 (비선형 문제의 경우 자동 학습 불가능함)

4.1.1 데이터 주도 학습

이미지를 인식하는 방식

- 이미지 인식 알고리즘을 직접 고안함
    - 이미지안에 숨은 규칙을 명확히 로직으로 풀어내기 쉽지 않은 방법 임
- 기계학습을 이용한 이미지 인식
    - 이미지에서 특징(feature)을 추출하고 그 특빙의 패턴을 기계학습 기술로 학습
    - 이미지의 특징은 보통 벡터로 추출하며 비전분야에서는 SIFT, SURF, HOG 등을 사용함 (사람이만듬)
    - 변환된 벡터를 지도학습 방법인 SVM, KNN 등으로 학습 할 수 있음 (자동화)
- 신경망(딥러닝)방식
    - 사람이 개입하지 않고, 데이터를 있는 그대로 학습 함

4.1.2 훈련데이터와 시험 데이터

- 기계학습 문제는 **훈련데이터**(training data)와 **시험데이터**(test data)로 나눠 학습과 실험을 하는것이 보통임
- 훈련데이터만 사용하여 최적의 매개변수를 찾고 시험데이터로 이 모델의 실력을 평가하는 방법
  (우리가 원하는것은 훈련데이터에 최적화된 모델이 아니라 범용성을 가진 모델인지를 시험하기 위함)
- **오버피팅**(overfitting) : 주어진 데이터 셋에만 지나치게 최적화된 상태를 뜻하는 용어

4.2 손실함수

- 손실함수는 신경망의 성능이 얼마나 **나쁜지**를 측정하는 지표
- 신경망이 훈련데이터를 얼마나 잘 처리하지 **못하느냐**를 나타냄

4.2.1 평균제곱 오차 (Mean Squared Error, MSE)

- 정답에 가까울 수록 평균제곱오차가 작은 값을 가짐
- y_k : 신경망의 출력. (신경망이 추정한 값)

- t_k : 정답 레이블. 정답에 해당하는 차원에만 값이 1이고 나머지는 0을 가짐
    - **원-핫 인코딩** : 한 원소만 1로 하고 나머지 원소를 0으로 나타내는 표기법
- k : 데이터의 차원의 수 (결과값이 가질 수 있는 분류의 갯수 정도로 이해함)

$$E = \frac{1}{2}\sum_k (y_k - t_k)^2$$

```
def mean_squared_error(y, t):
    return 0.5 * np.sum((y-t)**2)
```

4.2.2 교차 엔트로피 오차 (Cross Entropy Error, CEE)

- MSE와 함께 손실함수로 많이 사용 됨.
- 자연로그 함수를 사용하므로 llog 안의 값이 0이되어 마이너스 무한대의 값이 나올 수 있음
    - 코드로 구현시 로그 안에 아주 작은 값(delta)를 더해서 무한대 값이 나오지 안도록 함
    - 손실함수를 x축의 음의방향으로 평행이동한 것과 같은 효과로
      출력값의 상대적 오차를 구하는데 영향을 주지는 않음

$$E = -\sum_k t_k \log y_k$$

```
def cross_entropy_error(y, t):
    delta = 1e-7
    return -np.sum(t * np.log(y + delta))
```

4.2.3 미니배치 학습

- 데이터가 모두 N 개라면 교차엘트로피 오차는 다음과 같이 표시 할 수 있음
- t_{nk}는 n 번째 데이터의 k번째 값을 의미, y_{nk}는 신경망의 출력, t_{nk}는 정답레이블을 뜻함

$$E = -\frac{1}{N}\sum_n \sum_k t_{nk} \log y_{nk}$$

- 모든 데이터를 전부 학습하기에는 데이터 양이 많음.
- **미니배치**(mini-batch) : 전체 데이터 중에 일부만 임의로 골라서 학습을 수행하는 방법

4.2.4 (배치용) 교차 엔트로피 오차 구현하기

~~작성 필요함~~

4.2.5 왜 손실함수를 설정하는가?

- 신경망 학습에는 최적의 매개변수(가중치/편향)를 탐색할 때 손실함수의 값이 가능한 작아지는 매개변수값을

찾는다.

- 이 때, 매개변수의 변화에따른 미분값(기울기)을 구하고, 이 미분값을 단서로 매개변수를 서서히 갱신하는 과정을 반복한다.
- 가중치 매개변수의 **손실함수의 미분**이란 **가중치 매개변수의 값을 조금 변화시켰을때, 손실함수가 어떻게 변하는가**의 의미이다.
- 따라서 이 미분값이 0인 경우 손실함수가 가장 작은 값을 가질 것이며, 매개변수의 갱신을 멈춘다.

- 신경망을 학습할 때 **정확도**를 지표로 삼으면 대부분의 장소에서 **미분값이 0**이 된다.
  즉 학습이 진행되는 **방향을 찾을 수 없다.**
  - 정확도는 입력값 데이터 중 올바로 처리한 것의 결과를 나타내며, 매개변수의 변화에 따라 연속적으로 변화하지 않는다. 즉, 해석가능한 함수가 아니라 매개변수의 변화에 따른 **미분**값을 활용 할 수 없다.
  - 이는 계단함수를 활성화함수로 사용하지 않는 이유와도 같다. 계단함수는 대부분의 지점에서 기울기가 0이라 계단함수의 결과를 손실함소의 지표로 사용하는 것은 의미가 없다.

4.3 수치 미분

- **경사법** 에서는 기울기(경사) 값을 기준으로 나아갈 방향을 정한다.

4.3.1 미분

- 미분은 순간의 변화량을 의미하며 다음과 같이 수식으로 표현 할 수 있다.

$$\frac{df(x)}{dx} = \lim_{h\to 0} \frac{f(x+h) - f(x)}{h}$$

- 이 때, 수치계산중 오차를 줄이기 위해 (x-h) 일 때와 (x+h) 일때의 값을 이용하는 **중앙차분**을 사용 할 수 있다.

$$\frac{df(x)}{dx} = \lim_{h\to 0} \frac{f(x+h) - f(x-h)}{2h}$$

```
def numerical_diff(f, x):
    h = 1e-4 # 작은 값. 보통 0.0001정도면 적당 함
    return (f(x+h) - f(x-h)) / (2*h)
```

4.3.2 수치미분의 예

- 일 례로, $y = 0.01x^2 + 0.1x$ 의 해석적 해는 $\frac{df(x)}{dx} = 0.02x + 0.1$이다.
- x=5 일 때와 x=10일때의 해석적 해는 0.2와 0.3 이지만, 수치미분으로 구한 값은
  이 값과 정확히 일치하지 않고, 약간의 오차를 가지는 것을 확인 할 수 있다.

4.3.3 편미분

- 다음은 변수가 2개인 다변수 함수의 예 이다.

$$f(x_0, x_1) = x_0^2 + x_1^2$$

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

- 변수가 여럿인 함수에 대한 미분을 편미분 이라 한다.
- 다음은 $x_0 = 3, x_1 = 4$일 때 $\frac{\partial f}{\partial x_0}$ 를 구하는 코드의 예 이다. (x_1을 상수로 취급)

```
def function_tmp1(x0):
    return x0*x0 + 4.0**2

numerical_diff(function_tmp1, 3.0)
```

4.4 기울기

- **기울기**(gradient) : 모든 변수의 편미분을 벡터로 정리한 것
- 기울게 벡터가 가리키는 곳은 **각 장소에서 함수의 출력값을 가장 크게 줄이는 방향**을 의미 함

$$\nabla f = \left(\frac{\partial f}{\partial x_0}, \frac{\partial f}{\partial x_1} \right)$$

```
def numerical_gradient(f, x):
    h = 1e-4   #0.0001
    grad = np.zeros_line(x) #x와 형상이 같은 배열 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)
        # gradient 계산
        grad[idx] = (fxh1 - fxh2)/(2*h)
        x[idx] = tmp_val #값 복원

    return grad
```