모두의 딥러닝

목차

1. 나의 첫 딥러닝

2. 딥러닝을 위한 기초

3. 딥러닝의 동작 원리

4. 오차 수정하기: 경사 하강법

5. 참 거짓 판단 장치: 로지스틱 회귀

6. 퍼셉트론

7. 다층 퍼셉트론

8. 오차 역전파

9. 신경망에서 딥러닝으로

10. 모델 설계하기

11. 데이터 다루기

12. 다중 분류 문제 해결하기

13. 과적합 피하기

14. 베스트 모델 만들기

15. 선형 회귀 적용하기

16. 이미지 인식의 꽃, CNN 익히기

17. 딥러닝을 이용한 자연어 처리

18. 시퀀스 배열로 다루는 순환 신경명(RNN)

19. 세상에 없는 얼굴 GAN, 오토인코더

20. 전이 학습을 통해 딥러닝의 성능 극대화하기

심화학습 1. 오차 역전파의 계산법

심화학습 2. 파이썬 코드로 확인하는 신경망

1. 나의 첫 딥러닝

1-1. 딥러닝 실행을 위한 준비사항

① 인공지능이란?
인공지능 > 머신러닝 > 딥러닝

② 인공지능에 필요한 요소
-데이터, 컴퓨터(CPU, GPU), 프로그램

1-2. 딥러닝 작업환경 만들기

① 작업환경 구축 방법 2가지
-개인 PC에 프로그램 설치
-구글 코랩 활용하기(구글이 클라우드에 마련한 주피터 노트북 환경)

② 작업환경 구축 상세
-프로그램을 구동할 언어/통합 패키치 설치(파이썬, 아나콘다, 텐서플로, 케라스)

1-3. 미지의 일을 예측하는 원리

① 인공지능 예측 과정
-데이터 입력 → 모델 학습 → 미지의 값 예측
-머신러닝의 예측 성공률은 얼마나 정확한 경계선을 긋는지가 중요
-딥러닝은 머신러닝 방법 중에 가장 효과적인 학습 방법

1-4. 폐암 수술 환자의 생존율 예측하기

① 코딩 예제

In [1]:
```
# 딥러닝을 구동하는 데 필요한 케라스 함수를 불러옵니다.
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# 필요한 라이브러리를 불러옵니다.
import numpy as np
import tensorflow as tf

# 실행할 때마다 같은 결과를 출력하기 위해 설정하는 부분입니다.
np.random.seed(3)
tf.random.set_seed(3)

# 준비된 수술 환자 데이터를 불러들입니다.
Data_set = np.loadtxt("../dataset/ThoracicSurgery.csv", delimiter=",")

# 환자의 기록과 수술 결과를 X와 Y로 구분하여 저장합니다.
X = Data_set[:,0:17]
Y = Data_set[:,17]

# 딥러닝 구조를 결정합니다(모델을 설정하고 실행하는 부분입니다).
model = Sequential()
model.add(Dense(30, input_dim=17, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

# 딥러닝을 실행합니다.
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(X, Y, epochs=100, batch_size=10)
```

```
47/47 [==============================] - 0s 1ms/step - loss: 0.4166 - accuracy: 0.8553
Epoch 93/100
47/47 [==============================] - 0s 1ms/step - loss: 0.4517 - accuracy: 0.8383
Epoch 94/100
47/47 [==============================] - 0s 1ms/step - loss: 0.4427 - accuracy: 0.8383
Epoch 95/100
47/47 [==============================] - 0s 1ms/step - loss: 0.4044 - accuracy: 0.8447
Epoch 96/100
47/47 [==============================] - 0s 1ms/step - loss: 0.4986 - accuracy: 0.8298
Epoch 97/100
47/47 [==============================] - 0s 1ms/step - loss: 0.4665 - accuracy: 0.8255
Epoch 98/100
47/47 [==============================] - 0s 1ms/step - loss: 0.4450 - accuracy: 0.8234
Epoch 99/100
47/47 [==============================] - 0s 1ms/step - loss: 0.4509 - accuracy: 0.8489
Epoch 100/100
47/47 [==============================] - 0s 1ms/step - loss: 0.4190 - accuracy: 0.8468
```

Out[1]: <keras.callbacks.History at 0x270bf1a6be0>

In [ ]:

84.68%의 정확도

2. 딥러닝을 위한 기초

2-1. 일차함수, 기울기와 Y절편

① 일차함수

Y=aX+b

-y가 x에 관한 일차식으로 표현된 경우의 함수

2-2. 이차함수와 최솟값

② 이차함수

Y=aX^2

-포물선을 그리는 함수

2-3. 미분, 순간 변화율과 기울기

① 미분

-아주 잘게 나누는 것으로 순간 변화율과 기울기 도출 가능

2-4. 편미분

① 편미분

-모든 변수를 미분하는 것이 아니라 원하는 한가지 변수만 미분하고 그 외에는 모두 상수로 취급

2-5. 지수와 지수 함수

① 지수함수

Y=a^X

-변수가 지수자리에 있는 함수

2-6. 시그모이드 함수

① 시그모이드 함수

f(X) = 1/(1+e^-X)

-지수함수에서 밑의 값이 자연 상수 e인 함수

2-7. 로그와 로그 함수

① 로그함수

a^X=b

-a를 X만큼 거듭제곱한 결과를 나타내는 함수

3. 딥러닝의 동작 원리

3-1. 선형 회귀의 정의

① 선형회귀

-종속변수 Y와 한 개 이상의 독립변수 X와의 선형 상관 관계를 모델링하는 회귀분석 기법

-단순 선형회귀 : 하나의 X값으로 Y값을 설명하는 선형회귀

-다중 선형회귀 : 여러 개의 X값으로 Y값을 설명하는 선형회귀

3-2. 가장 훌륭한 예측선이란?

① 정확한 직선

Y=aX+b

-선형회귀는 정확한 직선을 찾아내는 과정이고, 이를 통해 최적의 a값과 b값을 찾아냄

3-3. 최소 제곱법

① 최소 제곱법

a = (X-X평균)(Y-Y평균)의 합 / (X-X평균)^2의 합

-주어진 데이터와의 오차를 최소화하는 직선을 구하는 방법

-최소 제곱법을 통해 최적화된 기울기를 구함

3-4. 코딩으로 확인하는 최소 제곱

① 코딩 예제

In [8]:
```
import numpy as np
x=[2,4,6,8]
y=[81,93,91,97]

mx = np.mean(x)
my = np.mean(y)

divisor = sum([(i-mx)**2 for i in x])

def top(x, mx, y, my) :
    d = 0
    for i in range(len(x)):
        d+=(x[i]-mx) * (y[i]-my)
    return d
dividend = top(x, mx, y, my)

a = dividend / divisor
b = my-(mx*a)

print(a)
print(b)
```

```
2.3
79.0
```

3-5. 평균제곱오차(MSE, Mean Square Error)

① MSE

-선형회귀를 통해 구한 직선이 얼마나 잘 그려졌는지 평가하기 위한 오차를 평가하는 알고리즘

3-6. 잘못 그은 선 바로잡기

$$MSE = \frac{1}{n}\sum_{i}^{n}(y_i - \widehat{y_i})^2$$

-선형회귀란 임의의 직선을 그어 이에 대한 평균 제곱 오차를 구하고 이 값을 가장 작게 만들어 주는 a와 b 값을 찾아가는 작업

3-7. 코딩으로 확인하는 평균제곱오차

① 코딩 예제

In [1]:
```
import numpy as np

#가상의 기울기 a와 y 절편 b
fake_a_b=[3,76]

# x 값과 y값
data = [[2, 81], [4, 93], [6, 91], [8, 97]]
x = [i[0] for i in data]
y = [i[1] for i in data]

# y=ax + b에 a,b 값 대입하여 결과를 출력하는 함수
def predict(x):
    return fake_a_b[0]*x + fake_a_b[1]

# MSE 함수
def mse(y, y_hat):
    return ((y - y_hat) ** 2).mean()

# MSE 함수를 각 y값에 대입하여 최종 값을 구하는 함수
def mse_val(y, predict_result):
    return mse(np.array(y), np.array(predict_result))

# 예측값이 들어갈 빈 리스트
predict_result = []

# 모든 x값을 한 번씩 대입하여 predict_result 리스트완성.
for i in range(len(x)):
    predict_result.append(predict(x[i]))
    print("공부시간=%.f, 실제점수=%.f, 예측점수=%.f" % (x[i], y[i], predict(x[i])))

# 최종 MSE 출력
print("MSE 최종값: " + str(mse_val(predict_result,y)))
```

```
공부시간=2, 실제점수=81, 예측점수=82
공부시간=4, 실제점수=93, 예측점수=88
공부시간=6, 실제점수=91, 예측점수=94
공부시간=8, 실제점수=97, 예측점수=100
MSE 최종값: 11.0
```