PORTFOLIO

20161865 박종근

2020.05.31

CONTENTS

1. 텐서플로 개요
2. 텐서플로 코딩
3. MNIST 이해
4. 원핫 인코딩과 드롭아웃
5. 퍼셉트론 및 Sigmoid, Relu
6. 선형회귀
7. 이진 분류 및 크로스 엔트로피

Part 1.

텐서플로 개요

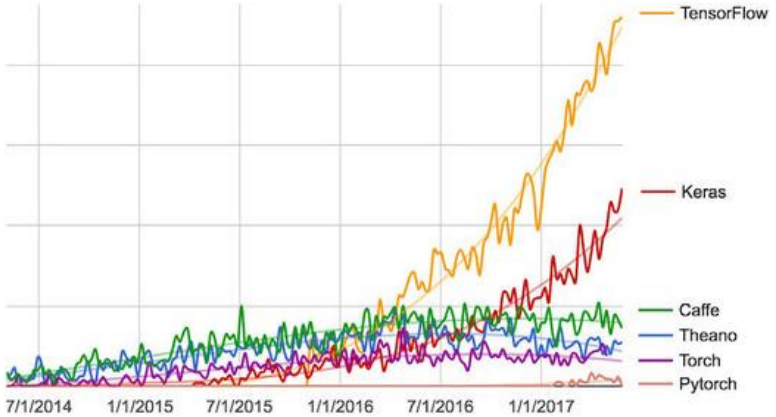
1.1 딥러닝 라이브러리 개요

딥러닝 라이브러리(플랫폼)

딥러닝 구현을 위한 클래스와 함수를 제공 한다.

다양한 라이브러리를 활용한다

EX) 텐서플로, 케라스, 파이토치



케라스

원래는 독자적인 고수준 라이브러리를 사용했음.

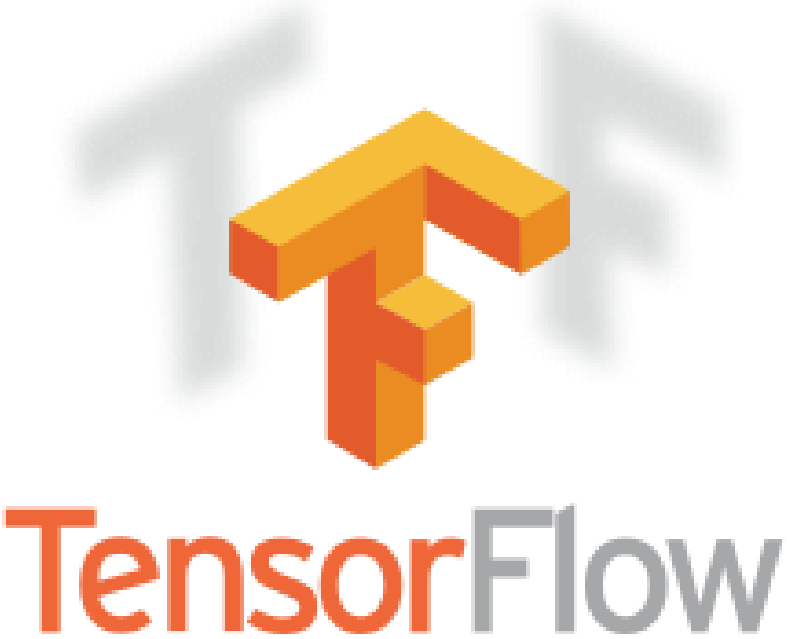
현재는 텐서플로의 고수준 API로도 사용

- 동일한 코드로 CPU 및 GPU 실행
- 사용하기 쉬운 API

1.1 딥러닝 라이브러리 개요

텐서플로

- 구글에서 만든 라이브러리(연구 및 프로덕션용 오픈소스)
- 딥러닝 프로그램을 쉽게 구현할 수 있도록 다양한 기능 제공
- 데스크톱, 모바일, 웹, 클라우드 개발용 API 제공
- Python, Java, Go 등 다양한 언어 지원
- 기본적으로 C++을 사용
- 파이썬을 최우선으로 지원(Python에서 개발 환경 편의)



1.2 텐서 플로

TensorFlow 계산 과정

- 모두 그래프라고 부르는 객체 내에 저장 및 실행
- 그래프를 계산하려면 외부 컴퓨터에 이 그래프 정보를 전달 및 결과 값 받아와야 함

Session

- 이 통신과정을 담당
- 생성, 사용, 종료 과정 필요

생성

Session 객체 생성

사용

Run 메서드에 그래프를 입력 및 출력, 값을 계산하여 반환

종료

Close 메서드 사용, with문을 사용하여 명시적으로 호출 불필요

```
x = tf.constant(3)
y = x**2

sess = tf.Session()
print(sess.run(x))
print(sess.run(y))
sess.close()
```

1.2 텐서 플로

TensorFlow 데이터 흐름 그래프

- 데이터 흐름 그래프로 이루어짐
- 텐서 형태의 데이터들이 딥러닝 모델을 구성하는 연산들의 그래프를 따라 흐르면서 연산 발생
- 딥러닝에서 데이터를 의미하는 Tensor와 DataFlow Graph를 따라 연산이 수행되는 형태를 합쳐 TensorFlow 단어 탄생

TensorFlow API 계층

- 딥 러닝 모델 구축 작업은 서로 다른 API 수준을 사용하여 해결
- 고급 API

Keras나 TF-Slim 과 같은 추상화 라이브러리를 제공하여 저수준 텐서플로 라이브러리에 대해 손쉽게 고수준 접근이 가능하게 해줌

- 중급 API
- 하위 수준 API

1.3 코랩을 이용한 기본 코딩

코랩

클라우드 기반의 무료 Jupyter 노트북 개발 환경

- 주피터 노트북을 지원하는 머신러닝, 딥러닝 클라우드 개발환경
- 파이썬, 판다스, 멧플롯리브 시각화 및 텐서플로우나 케라스 등 딥러닝 라이브러리 사용
- 구글 드라이브, 깃허브와 연계(깃허브 소스를 바로 코딩 가능)

코랩에서 버전 확인 하는 방법

%tensorflow_version 1.x

%tensorflow_version 2.x

```
# 텐서플로 1.0 버전 선택
try:
    # %tensorflow_version only exists in Colab
    %tensorflow_version 1.x
except Exception:
  pass

import tensorflow as tf
tf.__version__
```

```
TensorFlow is already loaded. Please restart the runtime to change versions.
'2.3.0'
```

1.3 코랩을 이용한 기본 코딩

조건 연산 tf.cond()

tf.cond(pred, true_fn=None, false_ fn=None, name=None)

- pred를 검사해 참이면 true_fc 반환
- pred를 검사해 거짓이면 false_fc 반환

```
x = tf. constant(1.)
bool = tf.constant(True)
res = tf.cond(bool, lambda: tf.add(x, 1.), lambda: tf.add(x, 10.))

print(res)
print(res.numpy())
```

```
tf.Tensor(2.0, shape=(), dtype=float32)
2.0
```

```
x = tf.constant(2)
y = tf.constant(5)
def f1(): return tf.multiply(x, 17)
def f2(): return tf.add(y, 23)
r = tf.cond(tf.less(x, y), f1, f2)

r.numpy()
```

```
34
```

1.3 코랩을 이용한 기본 코딩

1차원 배열 텐서

```
[ ]  t = tf.constant([1, 2, 3])
     t
```

```
<tf.Tensor: shape=(3,), dtype=int32, numpy=array([1, 2, 3], dtype=int32)>
```

```
[ ]  x = tf.constant([1, 2, 3])
     y = tf.constant([5, 6, 7])

     print((x+y).numpy())
```

```
[ 6  8 10]
```

```
[ ]  a = tf.constant([5], dtype=tf.float32)
     b = tf.constant([10], dtype=tf.float32)
     c = tf.constant([2], dtype=tf.float32)
     print(a.numpy())

     d = a * b + c

     print(d)
     print(d.numpy())
```

```
[5.]
tf.Tensor([52.], shape=(1,), dtype=float32)
[52.]
```

Part 2.

텐서플로 코딩

2.1 텐서플로 코딩

tf.matnul()

```
[ ]  import tensorflow as tf
     tf.__version__
```

```
'2.3.0'
```

```
[ ]  #텐서플로 2.0에서 즉시 실행은 기본으로 활성화
     tf.executing_eagerly()
```

```
True
```

```
[ ]  a = tf.constant([[1, 2],
                      [3, 4]])
     print(a)
     print(a.numpy())
```

```
tf.Tensor(
[[1 2]
 [3 4]], shape=(2, 2), dtype=int32)
[[1 2]
 [3 4]]
```

Numpy

```
     x = tf.constant([[0], [10], [20], [30]])
     y = tf.constant([0, 1, 2])

     print((x+y).numpy())
     # 0+0 0+1 0+2
     # 10+0 10+1 10+2
     # 20+0 20+1 20+2
     # 30+0 30+1 30+2
```

```
[ ]  import numpy as np

     print(np.arange(3))
     print(np.ones((3, 3)))
     print()

     x = tf.constant((np.arange(3)))
     y = tf.constant([5], dtype=tf. int64)
     print(x)
     print(y)
     print(x+y)
```

```
[0 1 2]
[[1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]]

tf.Tensor([0 1 2], shape=(3,), dtype=int64)
tf.Tensor([5], shape=(1,), dtype=int64)
tf.Tensor([5 6 7], shape=(3,), dtype=int64)
```

2.1 텐서플로 코딩

행렬, 원소와의 곱

```
[15] # 연산자 오버로딩 지원
     print(a)
     # 텐서로부터 numpy 값 얻기:
     print(a.numpy())
     print(b)
     print(b.numpy())
     print(a * b)
```

```
tf.Tensor(
[[1 2]
 [3 4]], shape=(2, 2), dtype=int32)
[[1 2]
 [3 4]]
tf.Tensor(
[[2 3]
 [4 5]], shape=(2, 2), dtype=int32)
[[2 3]
 [4 5]]
tf.Tensor(
[[ 2  6]
 [12 20]], shape=(2, 2), dtype=int32)
```

```
[14] # NumPy값 사용
     import numpy as np

     c = np.multiply(a, b)
     print(c)
```

```
[[ 2  6]
 [12 20]]
```

2.1 텐서플로 코딩

Shape과 reshgape

```
rank_three_tensor = tf.ones([3, 4, 5])
rank_three_tensor.shape
```

```
TensorShape([3, 4, 5])
```

```
[19] rank_three_tensor.numpy()
```

```
array([[[1., 1., 1., 1., 1.],
        [1., 1., 1., 1., 1.],
        [1., 1., 1., 1., 1.],
        [1., 1., 1., 1., 1.]],

       [[1., 1., 1., 1., 1.],
        [1., 1., 1., 1., 1.],
        [1., 1., 1., 1., 1.],
        [1., 1., 1., 1., 1.]],

       [[1., 1., 1., 1., 1.],
        [1., 1., 1., 1., 1.],
        [1., 1., 1., 1., 1.],
        [1., 1., 1., 1., 1.]]], dtype=float32)
```

```
#기존 내용을 6x10 행렬로 형태 변경
matrix = tf.reshape(rank_three_tensor, [6, 10])
matrix
```

```
<tf.Tensor: shape=(6, 10), dtype=float32, numpy=
array([[1., 1., 1., 1., 1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1., 1., 1., 1.]], dtype=float32)>
```

```
#기존 내용을 3x20 행렬로 형태 변경
#-1은 차원 크기를 계산하여 자동으로 결정하라는 의미
matrixB = tf.reshape(matrix, [3, -1])
matrixB
```

```
<tf.Tensor: shape=(3, 20), dtype=float32, numpy=
array([[1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
        1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
        1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.,
        1., 1., 1., 1.]], dtype=float32)>
```

2.1 텐서플로 코딩

Shape과 reshgape

```
#기존 내용을 4x3x5 텐서로 형태 변경
matrixAlt = tf.reshape(matrixB, [4, 3, -1])
matrixAlt
```

```
<tf.Tensor: shape=(4, 3, 5), dtype=float32, numpy=
array([[[1., 1., 1., 1., 1.],
        [1., 1., 1., 1., 1.],
        [1., 1., 1., 1., 1.]],

       [[1., 1., 1., 1., 1.],
        [1., 1., 1., 1., 1.],
        [1., 1., 1., 1., 1.]],

       [[1., 1., 1., 1., 1.],
        [1., 1., 1., 1., 1.],
        [1., 1., 1., 1., 1.]],

       [[1., 1., 1., 1., 1.],
        [1., 1., 1., 1., 1.],
        [1., 1., 1., 1., 1.]]], dtype=float32)>
```

2.1 텐서플로 코딩

tf.cast

tf.Tensor의 자료형을 다른 것으로 변경

```
[48] # 정수형 텐서를 실수형으로 변환.
     float_tensor = tf.cast(tf.constant([1, 2, 3]), dtype=tf.float32)
     float_tensor
```

```
<tf.Tensor: shape=(3,), dtype=float32, numpy=array([1., 2., 3.], dtype=float32)>
```

```
[49] float_tensor.dtype
```

```
tf.float32
```

2.2 텐서플로 난수

균등분포 난수

tf.random.uniform([1],0,1)

```
[2]  rand = tf.random.uniform([1],0,1)
     print(rand)
```

```
tf.Tensor([0.46348584], shape=(1,), dtype=float32)
```

```
     rand = tf.random.uniform([5, 4], 0, 1)
     print(rand)
```

```
tf.Tensor(
[[0.03641546 0.1705457  0.8808149  0.7510241 ]
 [0.55611837 0.1768111  0.4458835  0.6222067 ]
 [0.51166415 0.4800651  0.3876667  0.58960736]
 [0.3413657  0.76402235 0.8994421  0.11557627]
 [0.2849313  0.53889287 0.61080146 0.7176615 ]], shape=(5, 4), dtype=float32)
```

```
[4]  rand = tf.random.uniform([1000], 0, 10)
     print(rand[:10])#10개까지만 보이도록
```

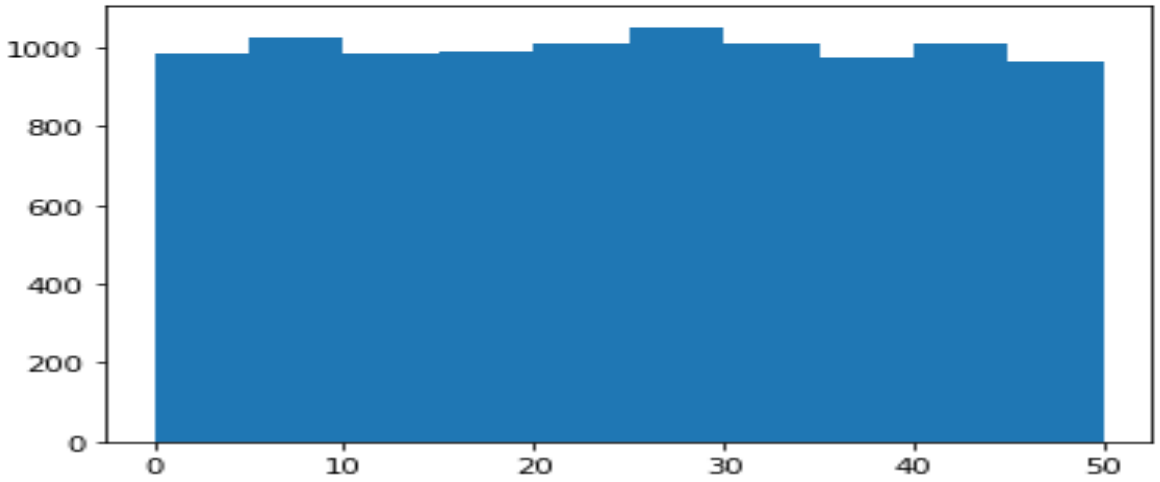
```
tf.Tensor(
[0.39684415 5.7802877  9.89257    8.745573   2.9729295  0.6878054
 6.1575556  6.4900446  0.04992723 8.040249  ], shape=(10,), dtype=float32)
```

2.2 텐서플로 난수

균등분포 1000개 그리기

```
import matplotlib.pyplot as plt
rand = tf.random.uniform([10000],0,50)
plt.hist(rand, bins=10)
```

```
(array([ 982., 1024.,  982.,  988., 1011., 1052., 1012.,  976., 1011.,
         962.]),
 array([3.8385391e-03, 5.0025668e+00, 1.0001295e+01, 1.5000024e+01,
        1.9998753e+01, 2.4997480e+01, 2.9996210e+01, 3.4994938e+01,
        3.9993668e+01, 4.4992393e+01, 4.9991123e+01], dtype=float32),
 <a list of 10 Patch objects>)
```



2.2 텐서플로 난수

정규분포 난수

- 크기, 평균, 표준편차

```
# 3, 9 랜덤한 수 여러개 얻기 (정규 분포)
rand = tf.random.normal([4],0, 1)
print(rand)
```

```
tf.Tensor([-1.3970348  -1.107464    0.8885755  -0.77111775], shape=(4,), dtype=float32)
```

```
# 3, 9 랜덤한 수 여러개 얻기 (정규 분포)
rand = tf.random.normal([2, 4],0,2)
print(rand)
```

```
tf.Tensor(
[[ 1.3843725  -0.5549141  -2.8101032   1.3291909 ]
 [ 1.4706075   3.6884804  -0.92601043 -0.34283492]], shape=(2, 4), dtype=float32)
```

2.2 텐서플로 난수

Shuffle

tf.random.shuffle(a)

```
import numpy as np
a = np.arange(10)
print(a)
tf.random.shuffle(a)
```

```
[0 1 2 3 4 5 6 7 8 9]
<tf.Tensor: shape=(10,), dtype=int64, numpy=array([2, 5, 1, 3, 6, 9, 4, 0, 7, 8])>
```

```
[ ]  import numpy as np
     a = np.arange(20).reshape(4, 5)
     a
```

```
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14],
       [15, 16, 17, 18, 19]])
```

```
[ ]  tf.random.shuffle(a)
```

```
<tf.Tensor: shape=(4, 5), dtype=int64, numpy=
array([[ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14],
       [15, 16, 17, 18, 19],
       [ 0,  1,  2,  3,  4]])>
```

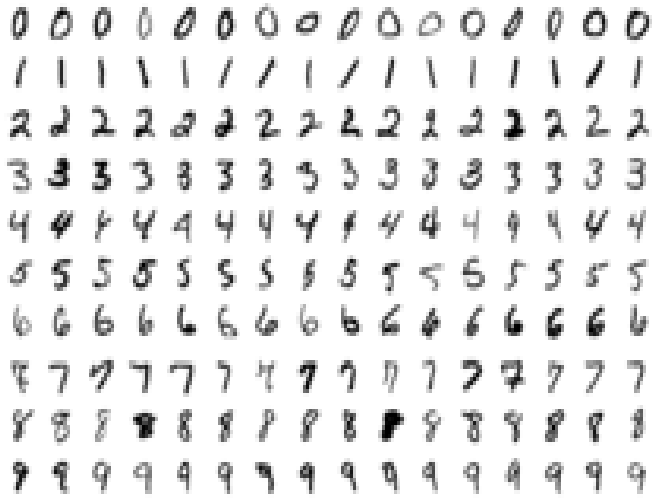
Part 3.

MNIST 이해

3.1 MNIST

MNIST (Modified National Institute of Standards and Technology)

- 미국 국립 표준 기술원(NIST)
- 손으로 쓴 자릿수에 대한 데이터 집합
- “필기 숫자 이미지 " 와 정답인 “레이블 " 의 쌍으로 구성
- 숫자의 범위는 0 ~ 9, 총 10개의 패턴을 의미
- 필기 숫자 이미지
- Label
- 대표적인 두 가지의 데이터 가져오는 방법



tensorflow.keras.datasets.mnist

tensorflow.examples.tutorials.mnist

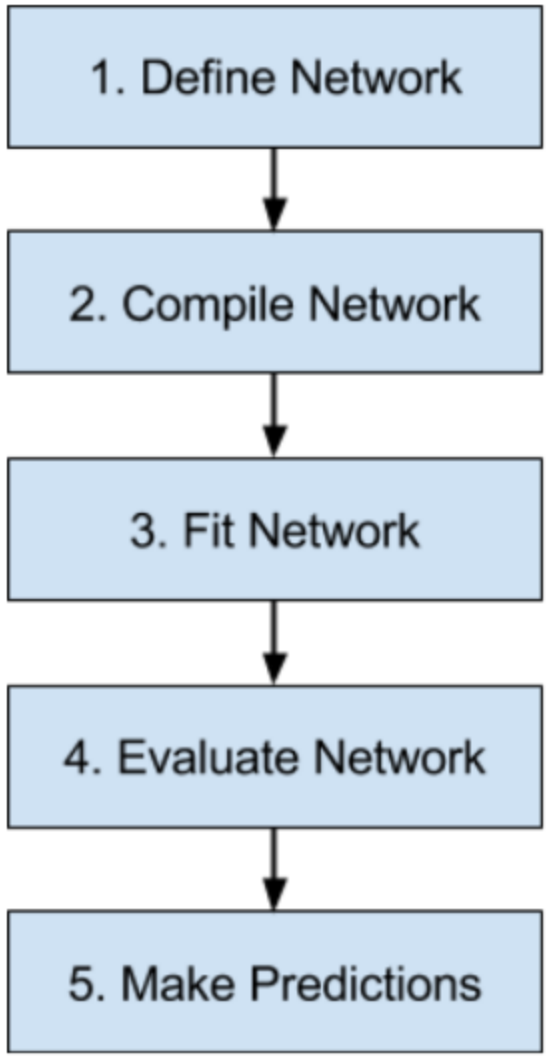
3.1 MNIST

케라스 딥러닝 (5개의 과정)

- 딥러닝 모델 만들기 (define)
- 주요 훈련방법 설정 (compile)

1. 최적화 방법(optimizers)
2. 손실 함수(losses)
3. 훈련 모니터링 지표(metrics)

- 훈련 (fit)
- 테스트 데이터 평가 (evaluate)
- 정답 예측 (predict)



3.1 MNIST

MNIST 손글씨 데이터 구조

```
[ ] x_train.shape

(60000, 28, 28)

[ ] y_train.shape

(60000,)

x_train[0]

  253, 198, 182, 247, 241,   0,   0,   0,   0,   0,   0,   0,   0,
    0,   0],
 [  0,   0,   0,   0,   0,   0,   0,   0,  80, 156, 107, 253, 253,
  205,  11,   0,  43, 154,   0,   0,   0,   0,   0,   0,   0,   0,
    0,   0],
 [  0,   0,   0,   0,   0,   0,   0,   0,   0,  14,   1, 154, 253,
   90,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
    0,   0],
 [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 139, 253,
  190,   2,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
    0,   0],
 [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  11, 190,
  253,  70,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
    0,   0],
 [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  35,
  241, 225, 160, 108,   1,   0,   0,   0,   0,   0,   0,   0,   0,
    0,   0],
```

```
y_train[0]

5
```

3.1 MNIST

행렬 내용 직접 출력

```
import sys
for x in x_train[0]:
  for i in x:
    sys.stdout.write('%3d' % i)
  sys.stdout.write('₩n')
```

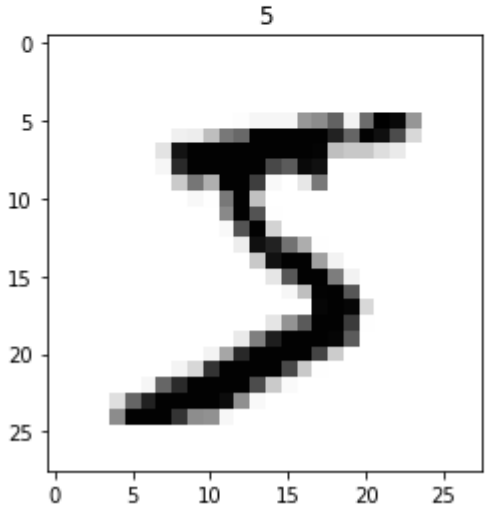
```
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  3 18 18 18126136175 26166255247127  0  0  0  0
  0  0  0  0  0  0  0  0 30 36 94154170253253253253253225172253242195 64  0  0  0  0
  0  0  0  0  0  0  0 49238253253253253253253253251 93 82 82 56 39  0  0  0  0  0
  0  0  0  0  0  0  0 18219253253253253253198182247241  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0 80156107253253205 11  0 43154  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0 14  1154253 90  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0139253190  2  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0 11190253 70  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0 35241225160108  1  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0 81240253253119 25  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0 45186253253150 27  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 16 93252253187  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0249253249 64  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 46130183253253207  2  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0 39148229253253253250182  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0 24114221253253253253201 78  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0 23 66213253253253253198 81  2  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0 18171219253253253253195 80  9  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0 55172226253253253253244133 11  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0136253253253212135132 16  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

3.1 MNIST

훈련 데이터 손글씨 보기

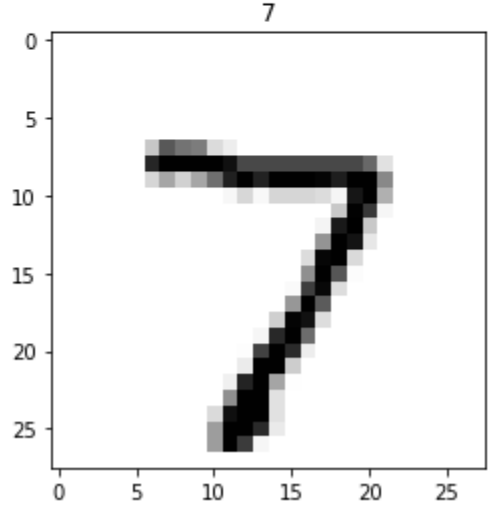
```
import matplotlib.pyplot as plt
n = 0
ttl = str(y_train[n])
plt.figure(figsize=(6, 4))
plt.title(ttl)
plt.imshow(x_train[n], cmap='Greys')
```

```
<matplotlib.image.AxesImage at 0x7f2c04e71240>
```




```
import matplotlib.pyplot as plt
n = 0
ttl = str(y_test[n])
plt.figure(figsize=(6,4))
plt.title(ttl)
plt.imshow(x_test[n], cmap='Greys')
```

```
<matplotlib.image.AxesImage at 0x7f2c04e1fd30>
```



3.1 MNIST

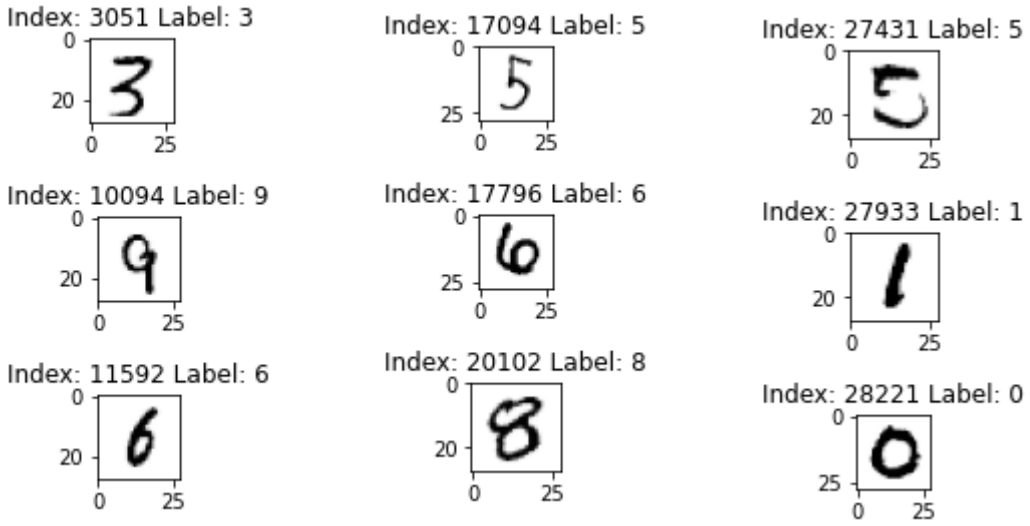
랜덤한 손글씨 그려 보기

```
from random import sample
norws, ncols = 4, 5
idx = sorted(sample(range(len(x_train)) , nrows * ncols))
#print(idx)

count = 0
plt.figure(figsize=(12, 10))

for n in idx:
    count += 1
    plt.subplot(nrows, ncols, count)
    tmp = "Index: " + str(n) + " Label: " +str(y_train[n])
    plt.title(tmp)
    plt.imshow(x_train[n], cmap='Greys')

    plt.tight_layout()
    plt.show()
```



3.2 MNIST 데이터 딥러닝 모델 적용

딥러닝 과정

- **모델 구성(개발)**
- **모델 훈련 : train**

모델이 문제를 해결하도록 훈련

ex) 어린 아이가 부모에게 훈련 받는 것을 비유

학습 방법 및 모니터링 지표 설정

경사하강법(내리막 경사 따라 가기)

손실 함수(Loss Function)

모니터링 지표 metrics

- 예측 : inference, prediction

딥러닝 구현 순서

- 0 필요 모듈 임포트
- 훈련과 정답 데이터 지정
- 모델 구성
- 학습에 필요한 최적화 방법과 손실 함수 등 설정
- 생성된 모델로 훈련 데이터 학습
- 테스트 데이터로 성능 평가

3.2 MNIST 데이터 딥러닝 모델 적용

데이터셋

- 훈련용과 테스트용
- Train data set, Test data set
- x(입력, 문제), y(정답, 레이블)

모델

- 딥러닝 핵심 신경망, 여러 층 구성

완전연결층 Dense(), 1차원 배열로 평탄화 Flatten()

- 학습 방법의 여러 요소들
- 옵티마이저, 최적화 방법

손실 함수

딥러닝 훈련 Epochs

Part 4.

원핫 인코딩과
드롭아웃

4.1 One Hot Encoding

원 – 핫 인코딩

원 – 핫 인코딩은 단어 집합의 크기를 벡터의 차원으로 하고, 표현하고 싶은 단어의 인덱스에 1의 값을 부여, 다른 인덱스에는 0을 부여하는 단어의 벡터 표현 방식

이렇게 표현된 벡터를 원 – 핫 벡터라고 정의

한계

단어의 개수가 늘어날 수록, 벡터를 저장하기 위해 필요한 공간이 계속 늘어남

- > 벡터의 차원이 계속 늘어난다고 표현

단어의 유사도를 표현하지 못함

- > 검색 시스템 등에서 심각한 문제

이걸 해결하기 위하여 단어의 잠재 의미를 반영해 다차원 공간에 벡터화 하는 기법을 사용

(카운트 기반의 벡터화, 예측 기반의 벡터화)

4.2 Drop out

Drop out

2012년 토론토 대학의 힌튼 교수 및 그의 제자들이 개발

층에서 결과 값을 일정 비율로 제거하는 방법

- 오버피팅 문제를 해결하는 정규화 목적을 위해 필요

오버피팅 : 학습 데이터에 지나치게 집중해 실제 테스트에서는 결과가 더 나쁘게 나오는 현상

- H2와 h5를 0으로 지정하여 사용되지 않게 조정
- 훈련 단계보다 더 많은 유닛이 활성화 되기 때문에 균형을 맞추기 위해 층의 출력 값을 드롭아웃 비율만큼 줄이는 방법
- 일반적으로 훈련단계에서 적용

드롭아웃을 층에 적응하면 훈련하는 동안 층의 출력 특성을 랜덤하게 끔

Part 5.

퍼셉트론 및
Sigmoid, Relu

5.1 퍼셉트론

퍼셉트론

- 인공신경망의 한 종류
- 1957년 코넬 항공 연구소의 프랑크 로젠플라트에 의헤 고안
- 가장 간단한 형태의 피드포워드 네트워크 – 선형분류기 -

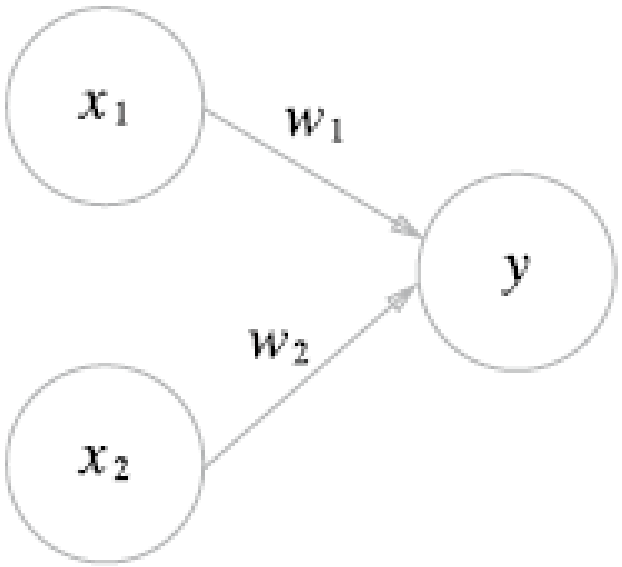
동작방식

각 노드의 가중치와 입력치를 곱한 것을 모두 합한 값이 활성함수에 의해 판단

그 값이 임계치보다 크면 뉴런이 활성화되고 결과값으로 1을 출력

뉴런이 활성화 되지 않으면 결과값으로 -1을 출력

5.1 퍼셉트론



- x_1과 x_2는 입력 신호, y는 출력 신호, w_1과 w_2는 가중치(weight)를 의미한다.
- 원을 **뉴런** 또는 **노드**라고 부른다.
- 입력 신호가 뉴런에 보내질 때는 각각 고유한 **가중치**가 곱해진다(w_1x_1, w_2x_2).
- 뉴런에서 전달 받은 신호의 총합이 **임계값** θ를 넘을 때만 1을 출력한다.

이것을 수식으로 나타내면, 아래와 같다.

$$y = \begin{cases} 0 & (w_1x_1 + w_2x_2 \leq \theta) \\ 1 & (w_1x_1 + w_2x_2 > \theta) \end{cases}$$

2.1 AND 게이트

x_1	x_2	y
0	0	0
1	0	0
0	1	0
1	1	1

5.1 퍼셉트론

2.2 NAND 게이트와 OR 게이트

NAND 게이트는 Not AND를 의미하며 AND 게이트의 출력을 반대로한 것과 같다.

$$(w_1, w_2, \theta) = (-0.5, -0.5, -0.7)$$

x_1	x_2	y
0	0	1
1	0	1
0	1	1
1	1	0

OR 게이트는 입력 신호 중 하나 이상이 1이면 출력이 1이 되는 논리 회로다.

$$(w_1, w_2, \theta) = (0.5, 0.5, 0.2)$$

x_1	x_2	y
0	0	0
1	0	1
0	1	1
1	1	1

신경망 학습은 이 매개변수의 값을 정하는 작업을 컴퓨터가 자동으로 하도록 한다. 학습이란 적절한 매개변수 값을 정하는 작업 이며, 사람은 퍼셉트론의 구조를 고민하고 컴퓨터에 학습할 데이터를 주는 일을 한다.

5.2 Sigmoid, Relu

Sigmoid

Binary classificatio에 적절한 함수

일정 값을 기준으로 0인지 1인지 구분함으로써 분류하는 방식

딥러닝에서는 특정 임계치를 넘을 때만 활성화

처음은 input layer, 마지막은 output layer

가운데 보이지 않는 9개의 hidden layer

0<n<1 사이의 값만 다루므로 결국 chain rule을 이용해 계속 값을 곱해나간다고 했을 때 결국 값이 0에 수렴할 수 밖에 없다는 한계를 지님

나중에는 1보다 작아지지 않게 하기 위한 대안으로 Relu라는 함수 적용

5.2 Sigmoid, Relu

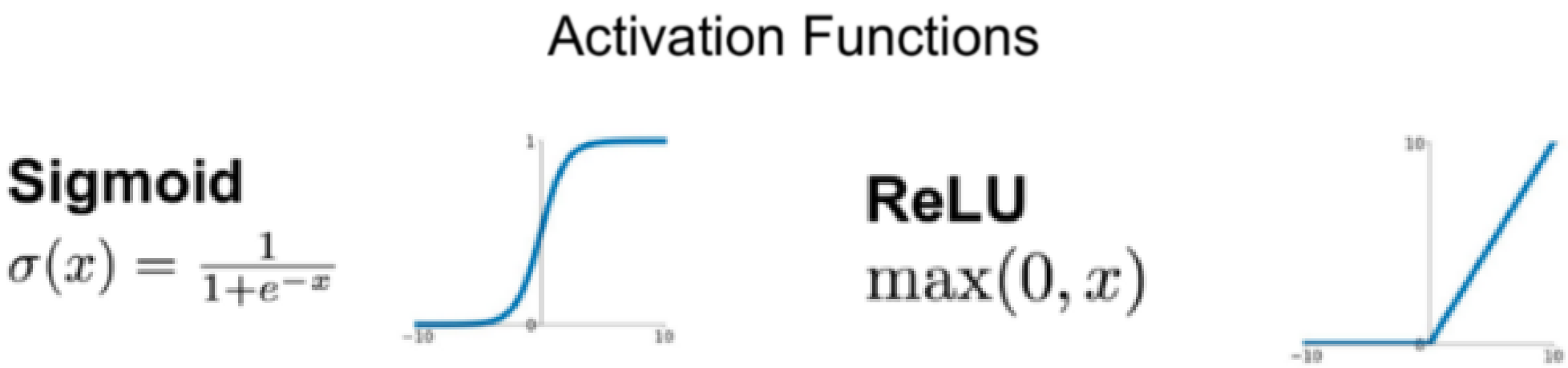
Relu

내부 hidden layer를 활성화 시키는 함수

Sigmoid를 사용하지 않고 Relu라는 활성화 함수를 사용하여 0보다 작은 값이 나온 경우 0을 반환, 0보다 큰 값이 나온 경우 그 값을 그대로 반환하는 함수

0보다 큰 값일 경우 1을 반환하는 sigmoid와 차이점을 지님

내부 hidden layer에는 Relu를 적용, 마지막 output layer에서만 sigmoid 함수를 적용하면 정확도가 상승



Part 6.

선형회귀

6.1 선형회귀와 로지스틱 회귀

선형 회귀

단순 선형 회귀 분석

입력은 특징이 하나이며, 출력은 하나의 값으로 나온다.

$$H(x) = Wx + b$$

Ex) 키로 몸무게를 추정

다중 선형 회귀 분석

입력은 특징이 여러 개이며, 출력은 하나의 값으로 나온다.

$$y = W_1x_1 + W_2x_2 + \ldots W_nx_n + b$$

Ex)역세권, 아파트 평수, 주소로 아파트값을 추정

로지스틱 회귀

이진 분류, 입력이 하나 또는 여러 개, 출력은 0 아니면 1값으로 나온다.

6.1 선형회귀와 로지스틱 회귀

선형 회귀의 해석

예측 변수에 대한 모든 값이 주어졌을 때, 수립한 선형 회귀 모델을 사용해 예측 변수 x가 응답 변수 y에 미치는 영향을 확인 할 수 있다.

회귀 결과를 해석할 때 주의해야 할 점은

일부 독립 변수가 응답 변수의 변화에 영향을 주지 않을 수 있다.

한계 효과가 큰 상황에서도 고유 영향은 적을 수 있다.

아주 복잡하게 서로 연계되어 있는 시스템을 분석하는 상황에서 고유 효과는 매우 중요한 역할을 한다.

이에 따라 예측 변수에 영향을 주기도 한다.

대다수 경우에는 다중 선형 회귀는 응답 변수와 예측 변수 사이의 관계를 밝히는 걸 실패한다.

6.1 선형회귀와 로지스틱 회귀

손실 함수

목적 함수, 비용 함수라고도 부름

실제 값과 예측 값에 대한 오차에 대한 식

- 예측 값의 오차를 줄이는 일에 최적화 된 식
- 평균 제곱 오차 등을 사용

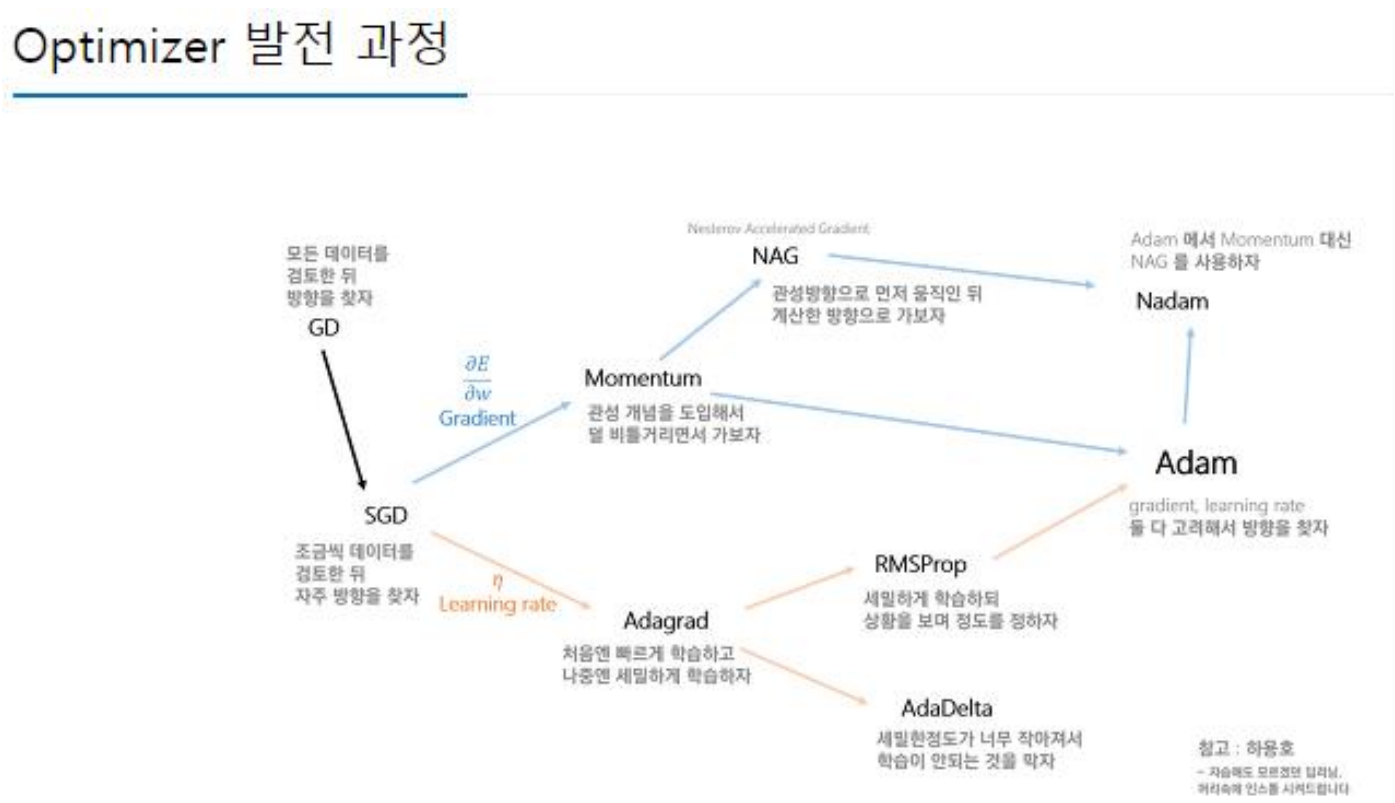
- **단순 선형 회귀의 손실 함수.**
  관측된 m개의 데이터 (x, y)에 대하여 단순 선형 회귀 모델을 다음과 같이 정의할 때,
  $h_\beta = \beta_0 + \beta_1 x$
  손실 함수 $J(\beta_0, \beta_1)$ 는 아래와 같이 정의할 수 있다.
  $$J(\beta_0, \beta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\beta(x^i) - y^i)^2$$

- **다변량 선형 회귀의 손실 함수.**
  n개의 독립 변수 X와 관측된 m개의 데이터(x, y)에 대하여 다변량 선형 회귀 모델을 다음과 같이 정의할 때,
  $h_\beta = \beta_0 + \beta_1 x_1 + \ldots + \beta_n x_n$
  손실 함수 $J(\beta)$는 아래와 같이 정의할 수 있다.
  $$J(\beta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\beta(x_1^i, x_2^i, \ldots, x_n^i) - y^i)^2$$

6.2 옵티마이저

옵티마이저

- 머신 러닝에서 학습
- 최적화 알고리즘
- 적절한 w와 b를 찾아내는 과정



$$w^{+} = w - \eta * \frac{\partial E}{\partial w}$$

learning rate : 한번에 얼마나 학습할지

gradient : 어떤 방향으로 학습할지

6.3 경사하강법

경사하강법

함수 값이 낮아지는 방향으로 독립 변수 값을 변형시켜 가면서 최종적으로는 최소 함수 값을 갖도록 하는 독립 변수 값을 찾는 방법

미분계수가 0인 지점을 찾는 방식이 아닌 경사하강법을 사용하는 이유

- 주로 실제 분석에서 보게 되는 함수들은 닫힌 형태가 아니거나 함수의 형태가 복잡해 미분계수와 그 근을 계산하기 어려움
- 실제 미분계수를 계산하는 과정을 컴퓨터로 구현하는 것에 비해 경사하강법은 컴퓨터로 비교적 쉽게 구현 가능

6.3 경사하강법

경사하강법

함수의 기울기를 이용해 x의 값을 어디로 옮겼을 때 함수가 최소값을 찾는지 알아보는 방법

기울기가 양수라는 것은 x값이 커질 수록 함수 값이 커진다는 것을 의미하고, 반대로 기울기가 음수라면

x값이 커질 수록 함수의 값이 작아진다는 것을 의미한다고 볼 수 있다.

또, 기울기의 값이 크다는 것은 가파르다는 것을 의미하기도 하지만, 또 한편으로는

x의 위치가 최소값/최댓값에 해당되는 x좌표로부터 멀리 떨어져있는 것을 의미하기도 한다.

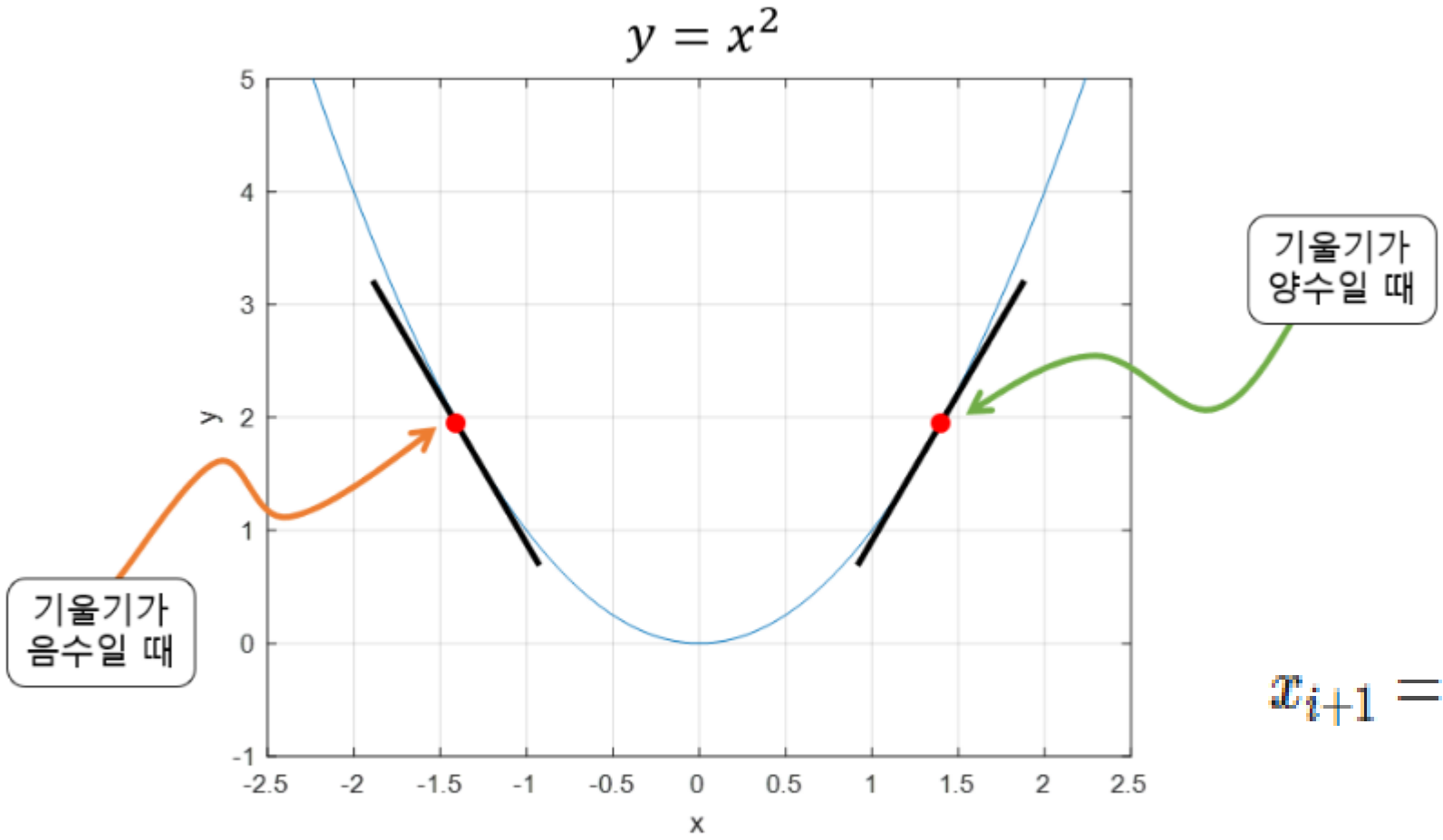
이를 이용해 특정 포인트 x에서 x가 커질 수록 함수값이 커지는 중이라면 (즉, 기울기의 부호는 양수) 음의 방향으로 x를 옮겨야 할 것이고,

반대로 특정 포인트x에서x가 커질 수록 함수값이 작아지는 중이라면 (즉, 기울기의 부호는 음수) 양의 방향으로 x를 옮기면 된다.

$$x_{i+1} = x_i - \text{이동 거리} \times \text{기울기의 부호}$$

6.3 경사하강법

경사하강법



$$x_{i+1} = x_i - \alpha \frac{df}{dx}(x_i)$$

그림 1. 기울기가 양수일 때와 음수일 때의 비교

6.3 오차역전파

순전파

입력층에서 출력층으로 계산해 최종 오차를 계산하는 방법

역전파

오차 결과 값을 통해서 다시 역으로 input 방향으로 오차가 적어지도록 다시 보내며 가중치를 다시 수정하는 방법

1986년 제프리 힌튼이 적용

- 엄청난 처리 속도의 증가

Part 7.

이진 분류 및 크로스 엔트로피

7.1 이진(이항) 분류

두 가지로 분류하는 방법

- PASS / FAIL
- SPAM / HAM
- 긍정 positive과 부정 negative (리뷰 텍스트를 기반으로 영화 리뷰)
- 로지스틱 회귀라고도 부름

결과 기술 방식

4 개의 결과를 기준

일반 레이블 방식 [0, 1, 0, 1]

One Hot Encoding 방식

[[1, 0], [0, 1], [1, 0], [0, 1]]

7.2 크로스 엔트로피

Entropy

정보이론이 말하는 핵심 아이디어는 Shannon이 창안했는데, 이를 압축하여 표현하면

“새롭고 불확실한 정보 또는 드물게 발생하는 사건 일수록 정보량아 많다.” 이다.

낯선 곳에서 배워야할 것과 습득해야할 정보가 더 많다.

이 정보의 양을 측정하고자 하는 것이 Shannon의 Entropy이다.

불확실할 수록 더 많은 정보가 있다는 점에서 착안하여 측정하는 방법도 생겼다.

$P(x)$는 x가 발현될 확률. $I(x)$는 x의 정보량이라고 하면,

- **불확실성이 클수록 정보량이 크다**: $P(x_1) > P(x_2) => I(x_1) < I(x_2)$.
- **두 개의 별건의 정보량은 각 정보량의 합과 같다**: $I(x_1, x_2) = I(x_1) + I(x_2)$. 여기서 두 개의 독립적인 사건 x_1, x_2의 발생 확률은 $P(x_1) * P(x_2)$인데, 정보량은 **합산**이기 때문에 이를 만족시키는 것은 log를 씌우는 것입니다. 즉, $I(x) = log\frac{1}{P(x)}$ 이 됩니다.
- **정보량은 bit로 표현된다**: $I(x) = log_2\frac{1}{P(x)}$.

$$H(X) = \sum_{i=1}^{k} log_2\frac{1}{P_i} * P_i$$

7.2 크로스 엔트로피

크로스 엔트로피

섀년 Entropy와 형태는 거의 동일하지만, 정보량 부분에 *Qi* (예측확률)가 사용되고, 확률 부분에 *Pi* (실제확률)로 사용되는 것이 유일한 차이점이다.

Cross Entropy라고 부르는 이유는, 아마도 두 *Qi*와 *Pi*가 곱이 이루어 지기 때문에 Cross라는 이름이 붙은게 아닐까라는 추측이다.

$$H(P,Q) = \sum_{i=1}^{k} log_2 \frac{1}{Q_i} * P_i$$

참조

https://velog.io/@vanang7

구글 설문조사

위키피디아

https://excelsior-cjh.tistory.com/169

https://gomguard.tistory.com/187

https://angeloyeo.github.io/2020/08/16/gradient_descent.html (공돌이의 수학정리노트)

감사합니다

20161865 박종근