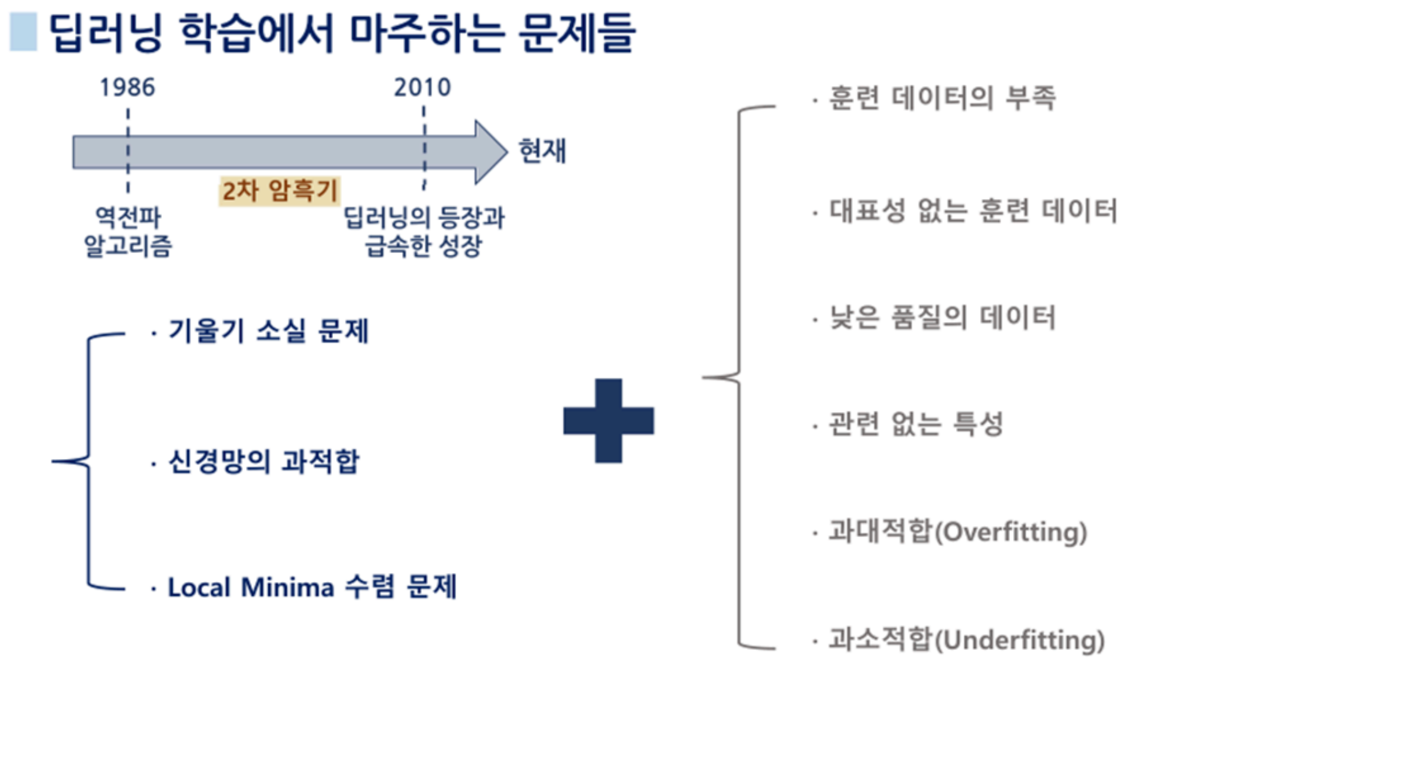
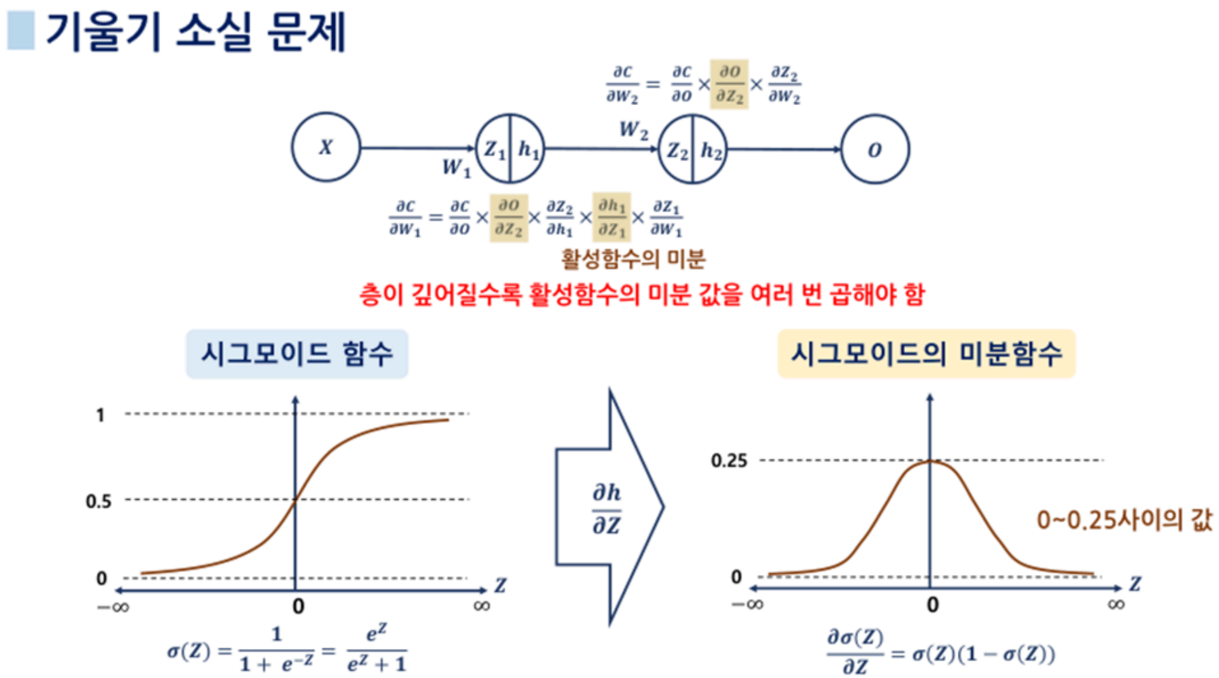
17강. 딥러닝 학습에서 마주하는 문제들

- 기울기 소실 문제
- RELU / 배치정규화
- 드롭아웃 / 데이터 증강 / 규제 / 조기종료
- 경사하강법 옵티마이저(Optimizer)

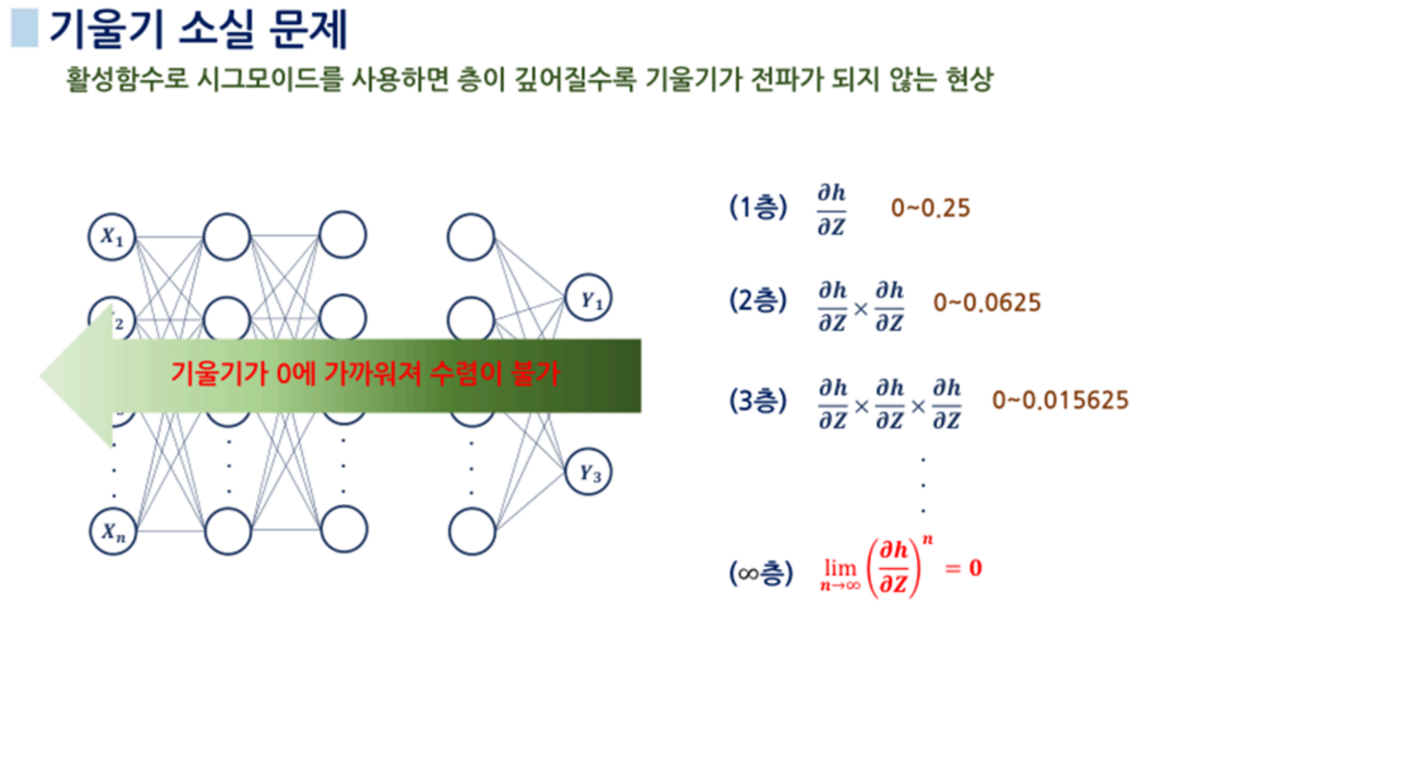


기울기 소실(Gradient Vanishing)와 기울기 폭주(Gradient Exploding)



기울기 소실 문제
$\frac{\partial C}{\partial W_2} = \frac{\partial C}{\partial O} \times \frac{\partial O}{\partial Z_2} \times \frac{\partial Z_2}{\partial W_2}$
X
W_1
Z_1 h_1
W_2
Z_2 h_2
O
$\frac{\partial C}{\partial W_1} = \frac{\partial C}{\partial O} \times \frac{\partial O}{\partial Z_2} \times \frac{\partial Z_2}{\partial h_1} \times \frac{\partial h_1}{\partial Z_1} \times \frac{\partial Z_1}{\partial W_1}$
활성함수의 미분
층이 깊어질수록 활성함수의 미분 값을 여러 번 곱해야 함
시그모이드 함수
시그모이드의 미분함수
1
0.5
0
$-\infty$
0
∞
Z
$\sigma(Z) = \frac{1}{1+e^{-Z}} = \frac{e^Z}{e^Z+1}$
$\frac{\partial h}{\partial Z}$
0.25
0~0.25사이의 값
$\frac{\partial \sigma(Z)}{\partial Z} = \sigma(Z)(1-\sigma(Z))$

순전파/역전파

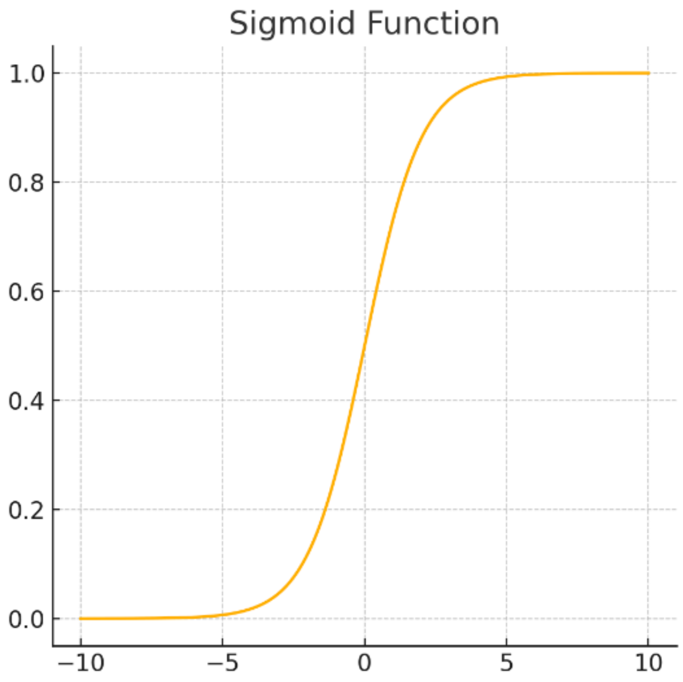


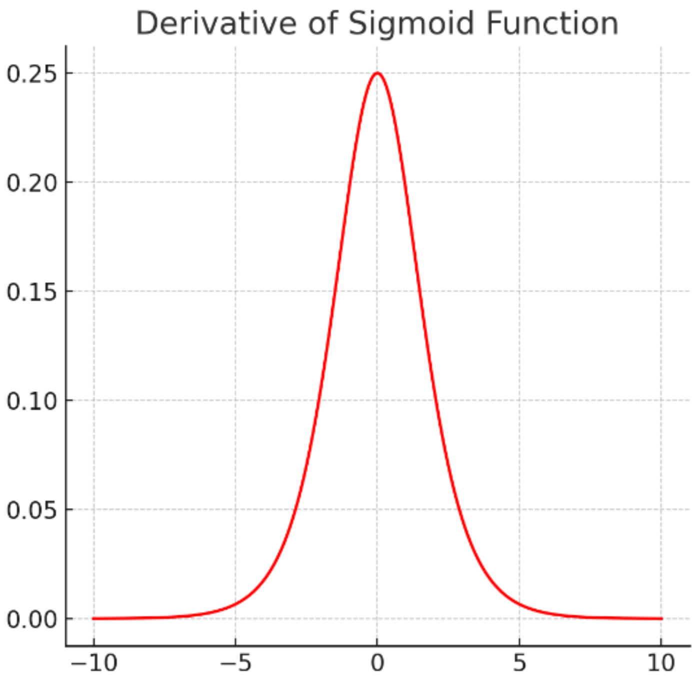
기울기 소실(Gradient Vanishing)과 폭주(Exploding)를 완화하는 방법

활성화 함수

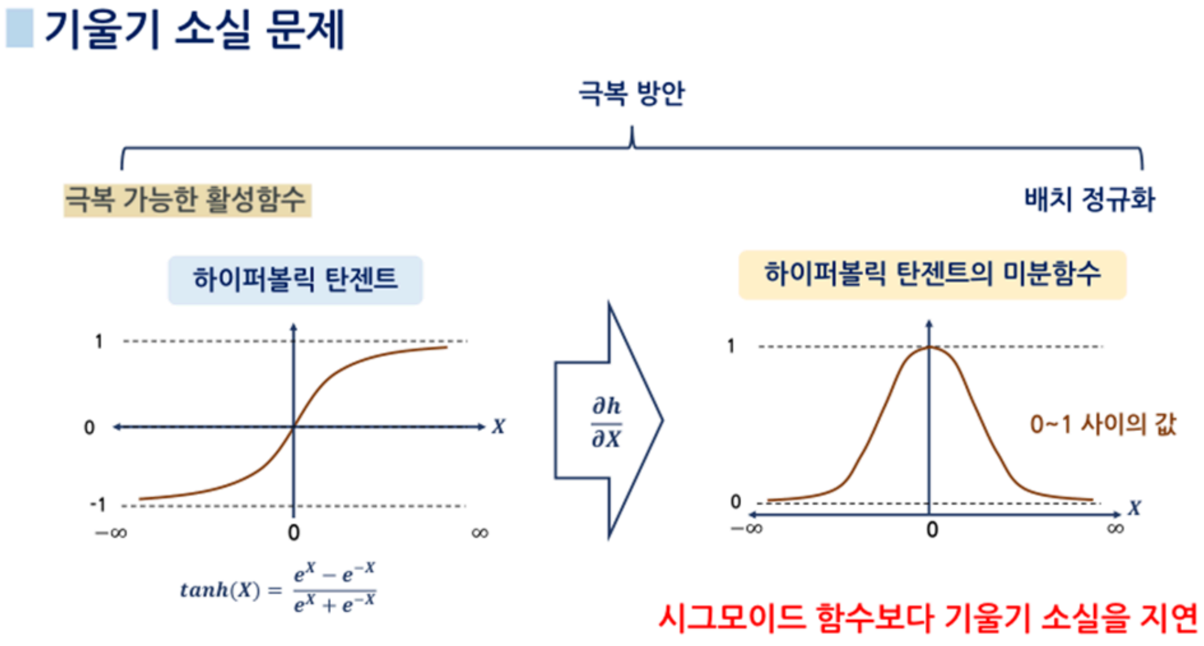
- Sigmoid나 tanh와 같은 활성화 함수는 입력 값이 매우 크거나 작을 때 출력의 변화가 매우 작음

> - 예를 들어 Sigmoid의 도함수의 경우 아래와 같이 σ(x)가 0 또는 1에 가까워지면 도함수가 0에 가까워짐
> - 이러한 활성화 함수들은 입력 값이 크거나 작을 때 기울기를 거의 0으로 만들기 때문에, 역전파 동안 기울기가 점점 더 작아짐





$$W_i = W_{i-1} - \eta \frac{\delta Error}{\delta W}$$

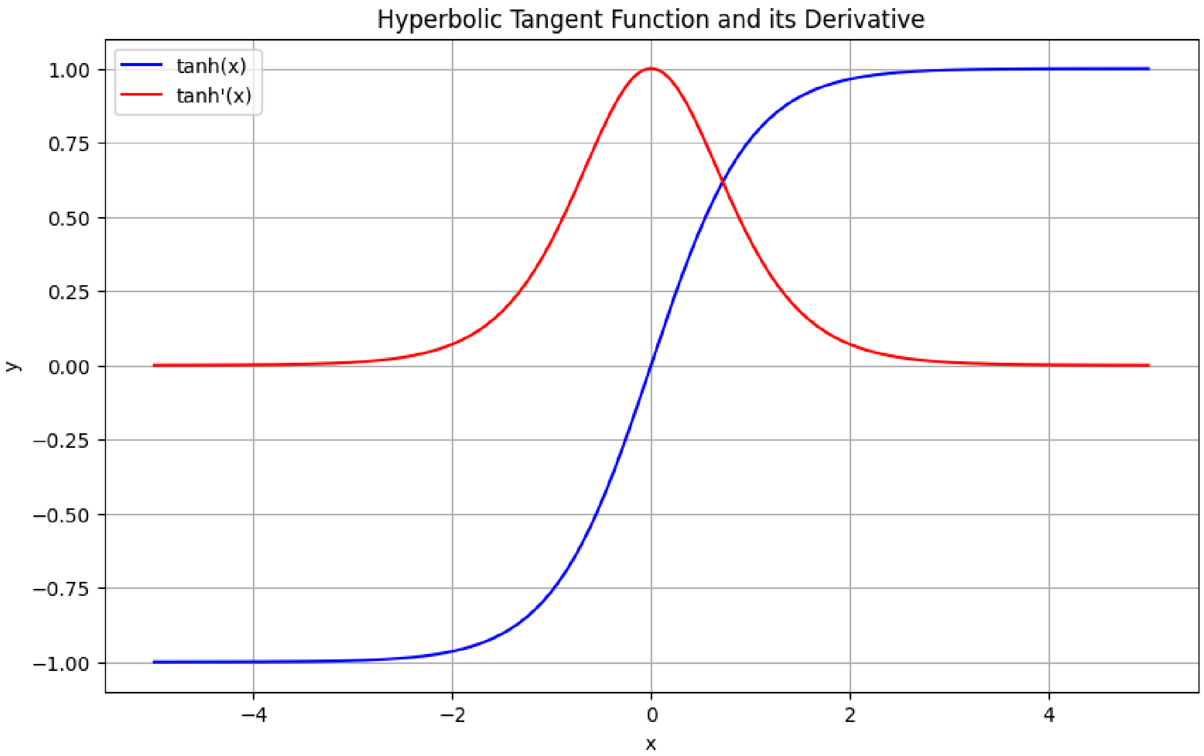


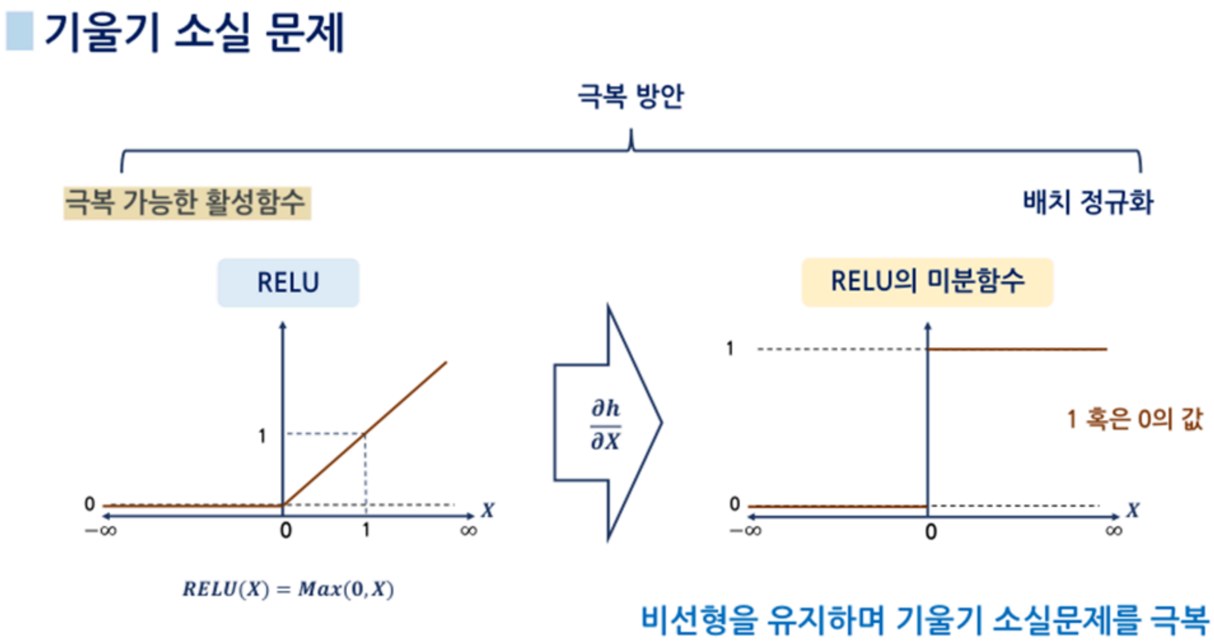
---

hyperbolic tangent function and its derivative

```
import numpy as np
import matplotlib.pyplot as plt

# Define the hyperbolic tangent function and its derivative
def tanh(x):
    return np.tanh(x)

def tanh_derivative(x):
    return 1 - np.tanh(x)**2

# Generate x values
x = np.linspace(-5, 5, 400)

# Calculate y values for tanh and its derivative
y_tanh = tanh(x)
```

```python
y_tanh_derivative = tanh_derivative(x)

# Create the plot
plt.figure(figsize=(10, 6))
plt.plot(x, y_tanh, label='tanh(x)', color='blue')
plt.plot(x, y_tanh_derivative, label="tanh'(x)", color='red')

# Add labels and title
plt.xlabel('x')
plt.ylabel('y')
plt.title('Hyperbolic Tangent Function and its Derivative')

# Add grid and legend
plt.grid(True)
plt.legend()

# Show the plot
plt.show()
```

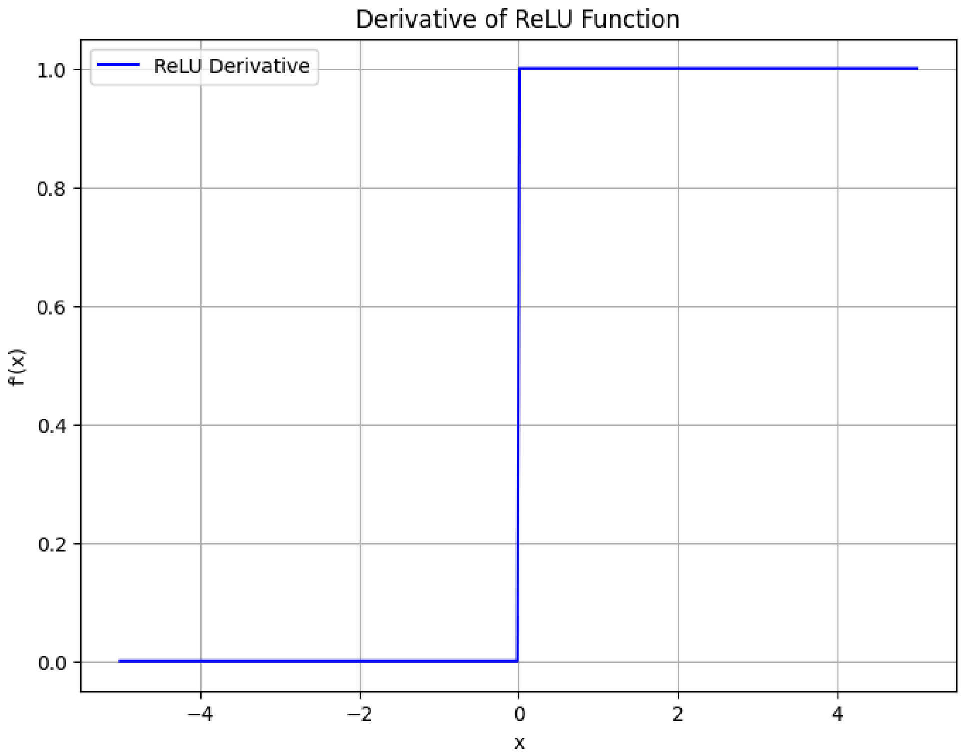




ReLU(Rectified Linear Unit) function and its derivative

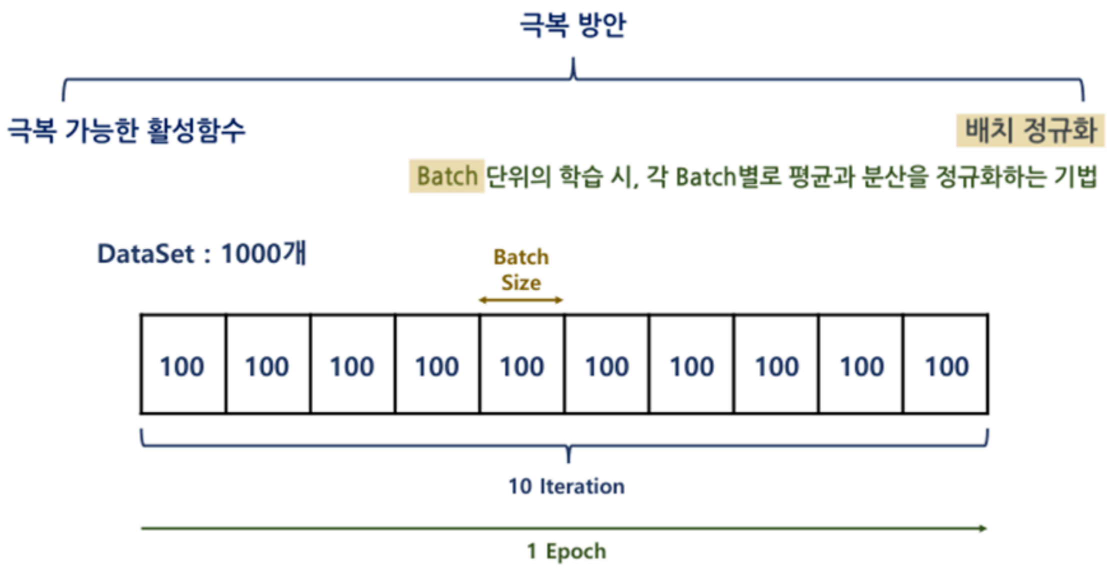
```
import numpy as np
import matplotlib.pyplot as plt

def relu_derivative(x):
  """
  Calculates the derivative of the ReLU function.

  Args:
    x: A numpy array of input values.

  Returns:
    A numpy array of the corresponding derivative values.
  """
  return np.where(x > 0, 1, 0)

```

```
# Generate x values
x = np.linspace(-5, 5, 400)
# Calculate ReLU derivative
y = relu_derivative(x)

# Create the plot
plt.figure(figsize=(8, 6))
plt.plot(x, y, label='ReLU Derivative', color='blue')
plt.xlabel('x')
plt.ylabel('f₩'(x)')
plt.title('Derivative of ReLU Function')
plt.grid(True)
plt.legend()
plt.show()
```



배치정규화

기울기 소실 문제



Epoch, Iteration, Batch size 개념

- Batch Size

> - 사람이 문제 풀이를 통해 학습해 나가는 과정에서 몇 개의 문제를 한 번에 쭉 풀고 채점할지를 결정하는 것과 같음
> - 총 100개의 문제가 있을 때, 20개씩 풀고 채점한다면 Batch 크기는 20
> - 전체 데이터가 3,000개이고 Batch 크기가 300이라면, 데이터를 300개씩 활용하여 모델을 점차 학습시켜 나감
> - 사람은 문제를 풀고 채점을 하면서 문제를 틀린 이유나 맞춘 원리를 학습함!!!!1

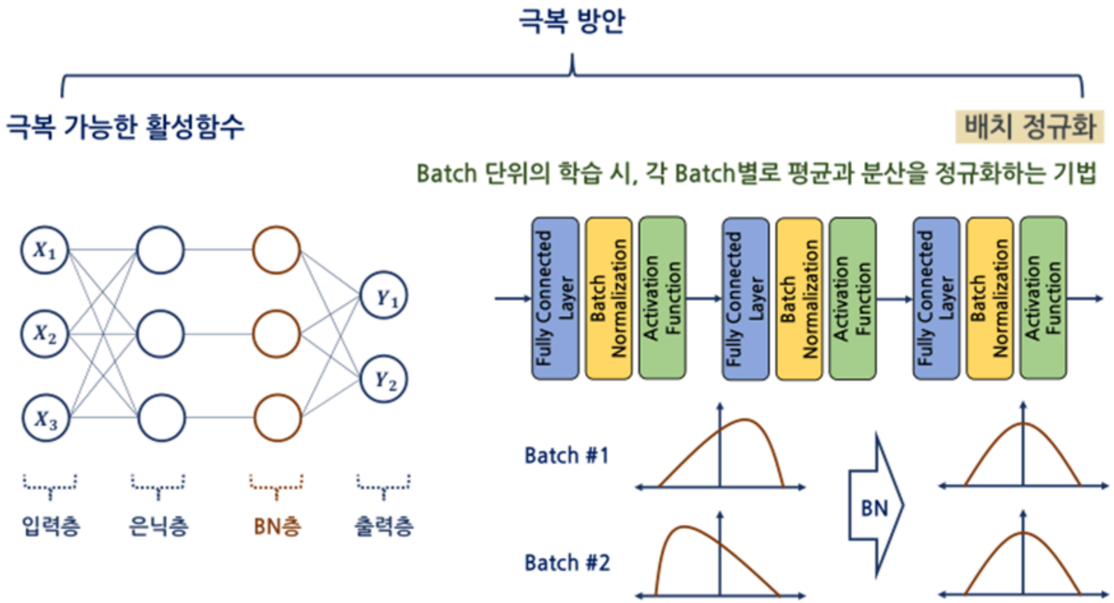
- Iteration - 전체 데이터에 대해 총 Batch의 수를 의미

> - Batch 크기가 300이고 전체 데이터 개수가 3,000이라면 전체 데이터셋을 학습시키기 위해서는 총 10개의 Batch가 필요함
> - 10번에 걸쳐 파라미터를 업데이트해야 되니까 즉, Iteration의 수는 10임

- Epoch - 전체 데이터셋을 학습한 횟수를 의미

> - 사람이 문제집으로 공부하는 상황에서 문제집에 있는 모든 문제를 처음부터 끝까지 풀고, 채점까지 마친 횟수를 의미
> - 전체 데이터셋을 1회 활용하여 모델을 학습했다면 Epoch는 1임

기울기 소실 문제



Batch normalization

```
import numpy as np
import matplotlib.pyplot as plt

def batch_normalization(x, gamma, beta, epsilon=1e-8):
    """
    Performs batch normalization on the input data.

```

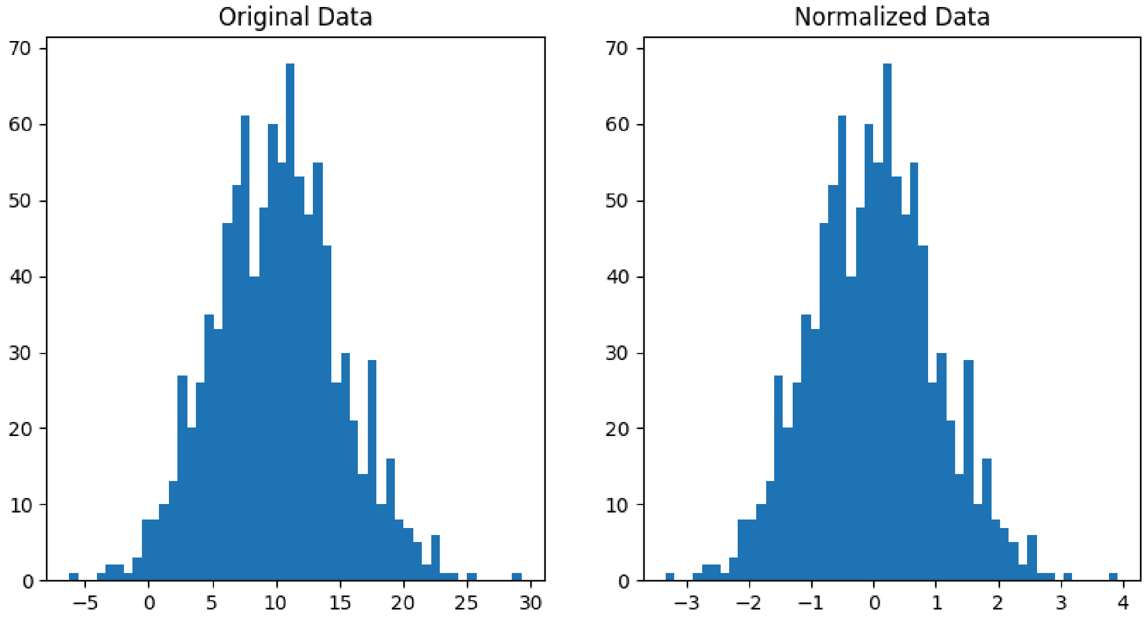
```
    Args:
      x: Input data (numpy array).
      gamma: Scale parameter (numpy array).
      beta: Shift parameter (numpy array).
      epsilon: Small constant for numerical stability.

    Returns:
      Normalized data (numpy array).
    """
    mean = np.mean(x)
    variance = np.var(x)
    x_norm = (x - mean) / np.sqrt(variance + epsilon)
    out = gamma * x_norm + beta
    return out

# Generate random data
np.random.seed(42)
x = np.random.normal(10, 5, 1000)

print(x)

# Set gamma and beta parameters
gamma = 1.0
beta = 0.0

# Apply batch normalization
x_normalized = batch_normalization(x, gamma, beta)

# Plot the original and normalized data
plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)
plt.hist(x, bins=50)
plt.title("Original Data")

plt.subplot(1, 2, 2)
plt.hist(x_normalized, bins=50)
plt.title("Normalized Data")

plt.show()
```

```
 15.15499761 14.6564006   5.80391238  8.45393812 11.65631716 14.87772564
  7.60412881  9.07170512  4.46832513  4.01896688 14.06262911 16.78120014
  9.63994939 15.01766449 11.80818013  6.77440123 11.80697803 17.69018283
  9.8208698  17.82321828 -3.09872552 14.10951252 10.43523534  8.50496325
 10.45880388  0.06215543  8.90164056 11.78556286 17.38947022  7.40864891
  5.95753199  7.49121478 14.57701059 11.64375555  7.35119898 12.56633717
 10.48538775 14.84322495  6.48973453  8.36168927  8.03945923  2.68242526
 11.48060139 11.30527636 10.02556728  8.82706433  2.92314629  7.89677339
  8.28642742  5.98861365  9.19357144 12.02025428 19.43092951 10.87288906
 11.28775195  9.62777042  0.40614392  9.86743062 10.30115105 22.31621056
  9.03819518 11.50773671  9.82644115  4.15660981 15.71411407 13.75966516
 13.95515974  5.45306273 17.01397155  2.99074469 12.93428547 20.95227813
  5.04731837  7.16851135 10.49825683  7.48262173  2.24668284 10.34281487
  4.68848143 12.36796215  5.40287883 17.74967203  6.08373354  8.38969242
 14.06758609  3.84567842 11.13729967 16.53571377  1.96258383 10.92316929
 11.29941397 13.90911436  3.81524645  3.39771693 12.60970783 11.48492337
 11.25246425 11.73224105  6.59987639 11.16126849 11.46536237  6.42824291
 19.32887256 12.3691646   4.04348251 13.28276804  5.12659165 13.93542302
 15.7929779   5.89658841 14.81688065 12.06390463 14.1103008  19.48396491
  8.77305942  6.23131918  5.55242785  5.92094858  9.61449145 11.70575987
 11.383454   14.13591625 10.06500946 17.26767039  8.67671583 23.60084583
 13.12833674  5.71421222  4.64553751 12.41236208  8.88268607 13.57000247
 12.36618812  9.63585544  5.76603141  2.42576388  7.76742524 14.28199397
 11.07046872  3.77130611 10.86590463 11.9265869   5.58071282 10.76862553
 10.29104359  4.28514851 11.7889368  12.80392263 15.41525622 15.26901026
  3.11165316  5.3108748  12.57517634 12.56892975 12.57523843 29.26365745
 12.85445255 15.6778282  14.77000882 13.25695626  8.42365378 13.7948461
  6.13587393  8.81590697  7.57318226 10.4093707  21.57329283  0.66367404
 13.43130095  1.93642064  7.64034067 15.44475298 10.3214001   4.61127611
  6.42348145 13.39798874  6.34816684 11.08229295 10.2278592   6.74199826
 20.71972045 13.16959511 -0.12571293 10.93227157  6.69106768 14.26216667
  6.03739631  9.42631779 12.52493639 14.32877597  3.99851796  8.32749382
  7.62527344  6.73335384 18.8272712  12.02490855  3.69558023 14.58930974
 20.61078099 15.1623263   2.40315017  7.57882964 16.33455575  6.46165267
 12.21909714 13.87317027  5.36534764  9.70237322 -6.2063367   4.87806179
  8.73715924  3.76108409 18.16205652  2.84929311  7.79977757 10.65370289
 17.20636645  2.82068924 15.81581876 10.05116531  5.09245674 12.31051737
 10.99529848  6.99891561 10.34901042  8.07343202 10.56758673 13.31065337
 17.93008408  3.81092251 20.66516687  0.239561    9.24107452 12.94158603
 11.40495934  6.8865024   8.95938875  7.53499533  7.05317622 14.24801049
 11.78507743  6.53545202 14.49799938 11.5364976  14.06431059 13.14814421
  5.85502495  7.1990948  13.73646803 13.05185133  9.89549203 10.58663692
 16.38832448  7.04214306 12.73548691  8.98903674  8.91159398 15.49388426
 14.12708174 14.06754818 16.52739404 10.10501921 13.40976486  8.44866622
 11.62083176  9.34928473 10.48497982 12.97578513  5.90889658 20.46193638
  4.96991309  3.92905694 15.79055437 13.95831347 13.12059909 13.14172755
```

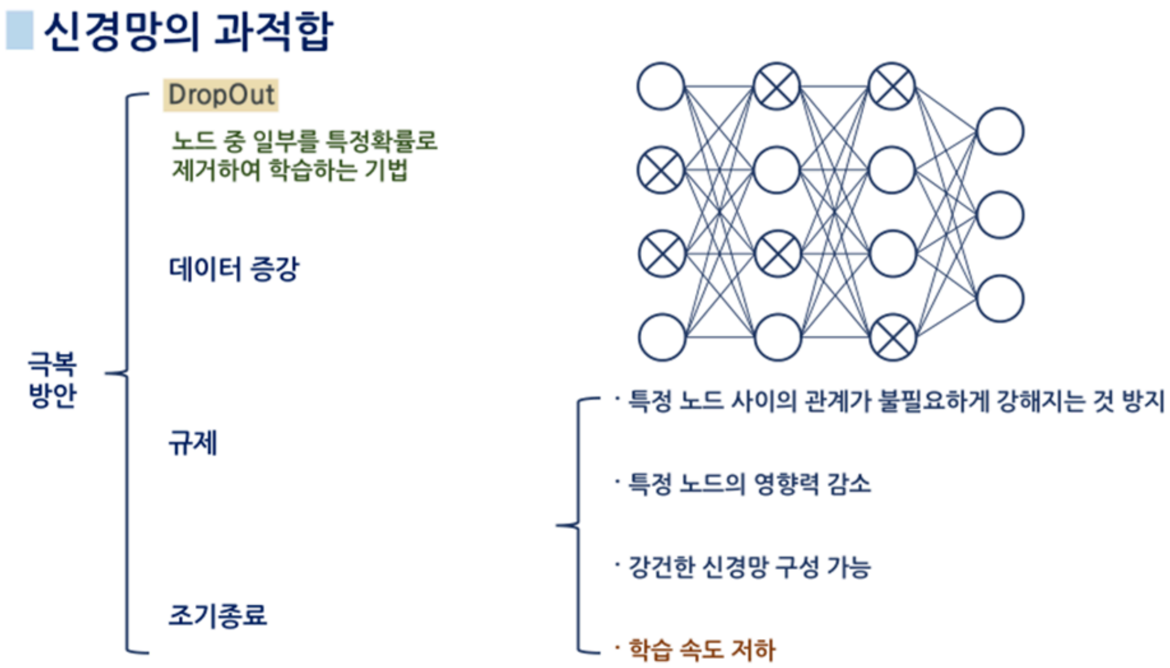
```
  9.93876614  5.51372814 10.37902279  6.61419144 14.87559867  9.26471309
  5.87251402  8.39307079 12.06465727  7.18137724  5.88889802 11.21843606
 11.22483286  7.46528412  7.64480847 11.16024969  2.75957829  2.96268113
  6.40777889  8.93276424 11.55453783 17.37678108 14.28829812  9.20030735
  9.90491896  4.98735318  9.90743432  8.55670681 11.6135928   5.86384528
 12.59673257 17.66369457  9.45619926 12.00855861 13.45071996  7.99389764
 11.12046241 10.062962   10.48838049  6.13495108 10.12255087 12.48999146
 17.25571804 14.79635413 20.76591229  6.16326219 14.36160318 10.91671003
 20.94901467  5.95850857  5.80139079  7.00303677 -0.61947862  7.37122489
  6.20433669 10.75196893 11.70877988 19.3808542  14.75211919  7.11548172
  5.50792664 12.45959586  3.39883396 19.15729383 15.8972006   7.65412174
  1.43432735 16.76936187  9.42730077 16.18908156  2.02786171  7.00312489
 10.0262185  10.23490297  7.74967264 13.11424966  4.66189785  9.28810257
 10.60147816 12.57219417 13.55807439  4.37678954  2.32942915 16.38838411
 11.66157006  6.25756732 17.75575988 10.57837317 15.89648592 10.33759241
 20.30373962 18.77670421  8.75517926 14.85785475 13.22687975 16.84315779
  5.1753827  13.4302573  15.29212243  1.20630257  4.08370744 -0.19616089
  8.65296583 13.58771128 17.51178526 10.3704739  18.14307773  3.09949271
  1.4830878   9.72226151 11.92032724  9.83652626 -0.3372105   9.5543998
  3.4776525  13.34836274 11.83299123  5.30060107  7.43066541  4.70393239
  9.68660451 14.7757116   5.07136977 12.52023258  7.34871191  6.03563584
  9.4648482   4.82378839  7.23175347  4.01061054 19.82362566 10.17631776
  6.50137246 11.06989955  9.43835975  8.895152   13.0708335  13.78753855
  7.34749426  7.1209088   8.62474151 -1.50960582  2.42404469 16.83437134
 18.22483857  8.7548198  12.88278482 11.55625077 25.39440404 15.59787456
  9.36041204  5.2222978   1.9677684  11.01731818  6.21824627  2.88873145
  6.76713558  4.59225998 18.43570818 14.40819878  9.96013679 17.39972069
 10.38684154  5.69357899 17.61562039 12.69455022  4.81376923  9.04830661
  5.62190873  3.08600135 14.63088774 19.5470832   3.00716213 12.81484618
  6.74678715  7.56437308  7.03803038  5.68004615 10.24260814  5.84524942
 11.35228413  9.74880945  8.80525977  5.46218169  7.11614335 13.77695613
 12.50458594  5.11222378 10.49666153 13.75693562  1.65297359 12.71680096
  6.68688121 12.85299334  6.18370422  0.9755895   1.86228781 10.24042473
 11.29861251  5.47841687 13.19296229  1.69239969  9.66960101  3.944919
  6.74081946 10.23699336  5.69793317  8.07722228 15.03146405  7.11554065
 14.17846056  4.35146573 12.64902089 17.2078431  -2.3582225   6.01552372
 12.88536064  8.98477307 11.85572937  6.98007407 10.43294894  9.22161382
 15.83891031 11.27210422 11.68801331  7.94061517  7.56196888  7.83720906
 11.97226071  7.8950776  11.44887428 20.37700399 14.35562352  8.36988234
 16.00606961  7.95962313 -0.19062268  4.95956845  0.64604039  8.24243258
 10.0920919  18.38218656 11.63463687  8.90449736 14.14702791 -1.05567655
 11.17807279 13.85432597  2.60706877 15.71877022 11.69248204  7.92356043
 13.16390933 21.35346429 10.90933128 11.24110293  7.7031955   5.75077815
 14.15167908  5.71958087 10.35783119  7.61171277 12.39489913 11.66831053
 15.18769972  7.44991801  8.65062532  5.10618142  7.7785337  11.88650247
 13.78494308  5.38917338 14.3480296  16.77818929 12.06717452 19.38397906
  6.131054    3.77672648  1.10639876 17.48022156 13.27182828  9.72207665
 11.39984313  4.37255476 22.2287599  10.64610591 10.54697397 13.62883312
 12.40504616 11.11942012  6.04762772 12.35734179 19.41012248 16.72710023
 17.96593313  7.44392162  5.0519759   9.3710654  10.27862456 15.47095759
  1.53767685 17.6477516   9.20996051  7.86559465  4.93947812  1.72571664
 14.11585292 10.36658984  3.5501955   3.52460614  8.3210765  18.34510763
  8.70204324  2.48428523  8.77128468  8.63638215 -3.48443321  9.72852567
  8.84532735 13.48103182 19.24478047 15.63282515  8.65555655  4.46737046
 22.86679902 10.29609217 10.06964646  9.87937456 10.9904238   9.27819794
  7.13168997  7.26570529  9.83623365  7.28287614  6.43577109 10.53215114
```

```
  8.72511391 17.51996494 -3.25484904 15.45753426 16.23042596 -0.36695116
  8.28656203  8.14279567  2.96244153  6.11091656  4.44712077 18.76135222
 14.67839197 16.35777547 13.60836032  4.35474114  7.37739867 12.44687281
  3.88936096 13.56499215  8.79837301  8.12589596 13.55479984 12.22131656
  8.19516917 15.79664902  4.59468336 13.07967803 12.96550629  8.4522678
 11.63066511  3.74443212 14.6201351   9.07548932  7.3863849  15.24504613
  6.47828155  2.95769352  2.21685413 13.03004976  3.59785324 18.77397091
 -0.40964704 18.48228184 11.05508734  9.51643444  7.27540457 11.99568057
  9.81182649 15.51650941 10.57113824 10.75150881  8.18193894  9.71527188
 11.53900884  1.44915804  3.25907289 13.71632047 10.85432719  9.08008332
 10.09216967 11.73790853  7.3012016   6.10847637 10.97922628  5.10813611
 12.04126378  1.48708198 15.14577819 12.36298741 11.28014867 14.91345492
 18.32737222 15.07185033  0.79562884  3.60211517  6.87590711 10.13045525
 12.5882951   6.37128093 10.93383382  6.22308534  6.94241099  2.96669452
  5.38383377  3.24157697  5.12063374 15.26820898  5.25300556 23.16191032
 12.4665895  10.92418062  5.7082111  13.5015494   7.12181087 10.61004907
 22.80042269  9.5197005  15.74636663  6.48411787  9.82505755 18.85400318
  6.86516471 19.06224279 13.53875968  7.18766612 13.1620387  14.86277225
 13.10904981  2.1488764   6.36431412  8.76240682  9.62783285 13.10336049
 10.888505    3.32327821 11.90098926 13.05292873 12.79895224 15.40390363
 14.16961077 12.2959004   9.64917144  1.69519533 12.1480911  11.03843844
 11.35789419  3.61625712  4.5947173  15.26576427  9.80222423 13.40750349
 10.14159188 10.1487807  14.69141903  7.41977636 10.48060388  7.68862356
  7.82751886  8.45413938 11.11066886  7.60625689 16.27878063  5.52696349
  9.06564178  7.80134471 17.23488942 10.98277388 15.1592227   2.57219813
 11.33525133 14.44815398 10.41141995 15.32740188  7.41355775 17.0467372
 21.49449062  8.1858072   7.77248739 17.26692239 17.89786073  7.38569986
  7.89906591  8.59107696  3.27774744  5.40674027  4.97929617  6.16101217
  9.82657556 11.17107366 17.75250246  5.0082298  14.92161199  8.93005578
  9.75268145 13.37409746  4.38638989 11.91204873 10.83226104 12.46225632
 11.44584322 22.2765007   6.81130008  7.34501522  6.88429737  7.2226144
  6.81306436 15.94508266 17.10252124  7.14626853  5.83822213 12.35707778
  7.23888478 13.16465909 11.0146151   2.42127943 17.73752601 18.97938837
  6.93605655  8.0614922  11.42932695 11.67228395 13.29272136 20.05102269
  9.11526386  6.00851378  3.10340386  6.3453498   9.83436514 18.97278932
  7.4119435  11.11893976  9.91788552 15.94196637 22.63466213  7.34565614
  7.55280279 15.22080439 13.40945745 19.23353663 12.91964093  8.20353955
 12.95327415 15.5435179  14.10241091 12.53637016 15.33337345 15.84647795
 16.91079496 13.24354944  9.1644096  10.73356843 16.03254483  5.91532165
 11.84336654  8.03330594 10.14372411 16.39225931 10.95549534 10.23218274
  3.2007193  13.73126783 13.22742091 20.81627362  8.46110883 11.09575164
 11.24691842 17.8872664   9.52352234 11.39510763 13.03948255 10.93304562
  7.76783193 10.97044996 15.36815875  4.8674235  10.66484837  6.49939593
 15.97523314  2.38406548  7.20539076 11.88605938 17.82762015  9.67124869
  7.22400237 19.40578535  2.7599305  -0.99402978 12.20007225  7.48972888
  4.89383591 13.54178224 11.21900357  7.17960685  3.59847801 14.36228664
 13.25100589  9.50412068 19.23318498  4.64957617  2.37237415  6.54045965
  9.77206992 11.21669725  8.79381971 11.76027698  3.74230288 17.21882302
  9.58924411 15.58647916 11.71362673 12.2837661  12.8488364  12.2385428
 13.2136138  16.64576265 10.98260585 13.54501879  9.55132153 17.20058608
  6.61803849 19.00470216  9.79921025  2.84612449 10.64052207  6.59474171
 14.20321774  6.7368801   7.76908283  0.55229635  7.7384684  -2.11939663
  2.08048588 13.80207328 13.92900079 12.12728781  5.16511928  9.76144322
  9.9819873   4.20817655 17.51699151 14.38681145  8.89517913 10.13442919
 11.04191404 -0.20867434  8.76411309  6.59007876  4.99189995  8.59449854
```

```
 18.98843263 13.20421431  7.14410505 12.86291391]
```



드롭아웃



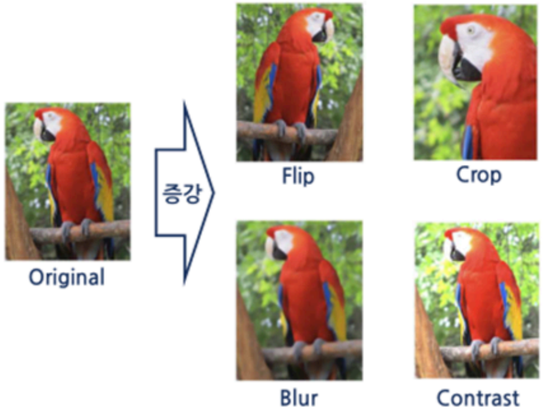
데이터 증강(Data Augmentation)이해

신경망의 과적합

극복 방안

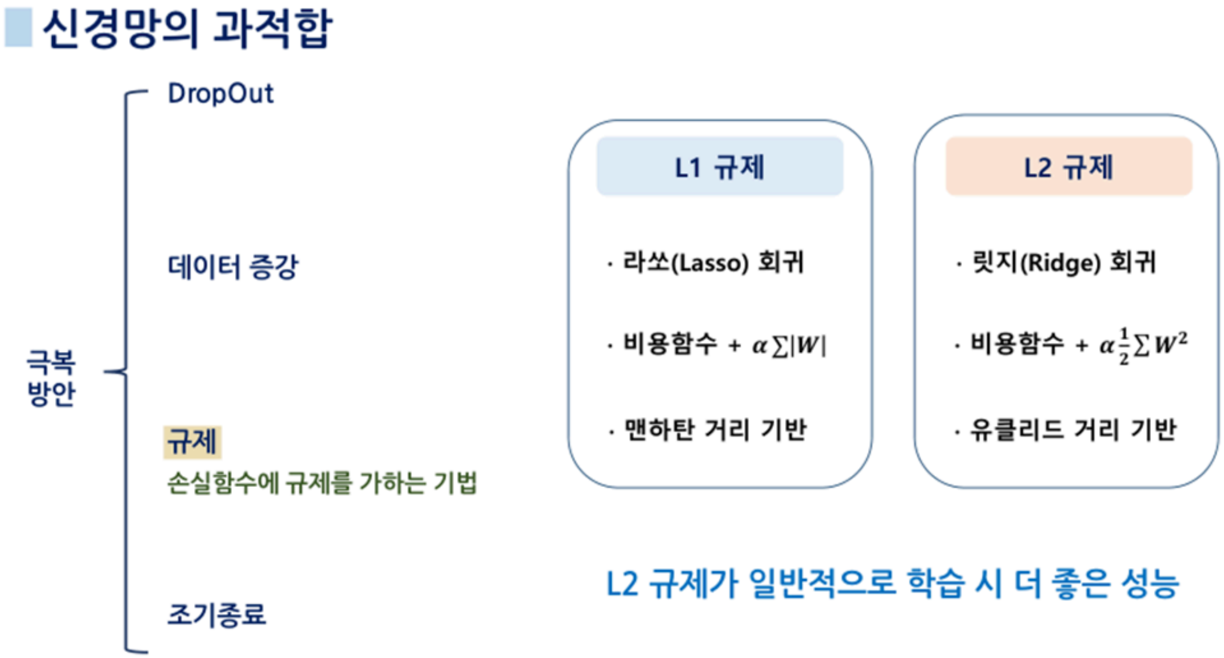
- DropOut
- 데이터 증강

  기존 데이터에서 새로운 데이터를 인위적으로 생성하여 데이터 양을 증가시키는 기법

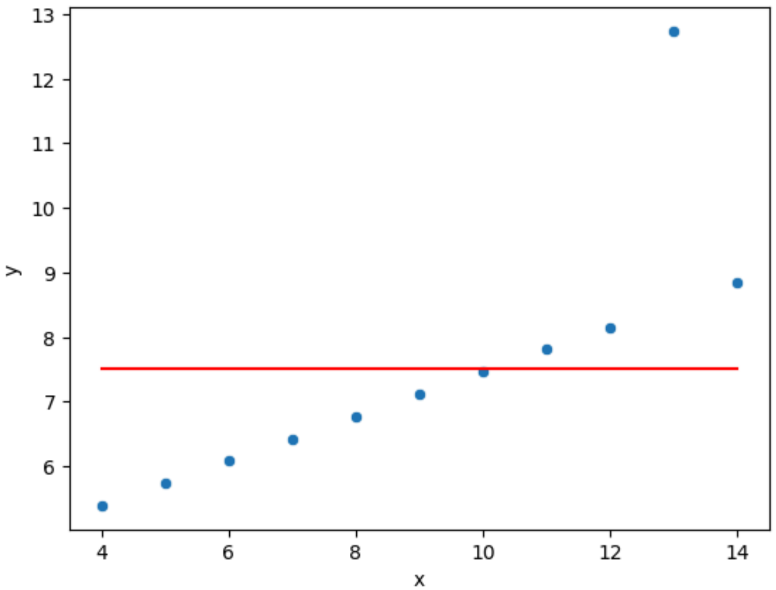


- 규제
- 조기종료

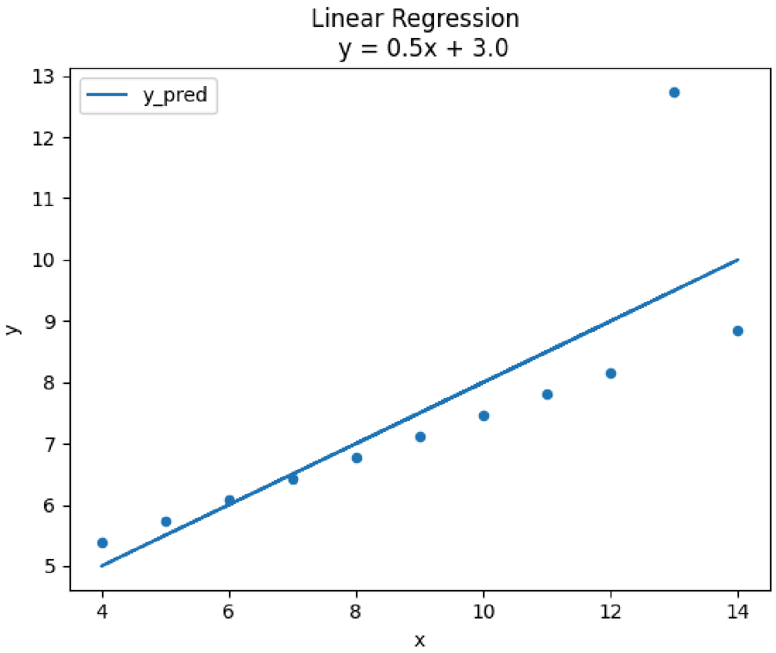
※ 출처 : What is image augmentation and how it can improve the performance of deep neural networks'
Link - [What is image augmentation - Albumentations Documentation]




```python
# 데이터 불러오기
import seaborn as sns

ans = sns.load_dataset('anscombe').query('dataset=="III"')
baseline = ans.y.mean() # 기준 모델
sns.lineplot(x='x', y=baseline, data=ans, color='red'); # 기준 모델 시각화
sns.scatterplot(x='x', y='y', data=ans);
```

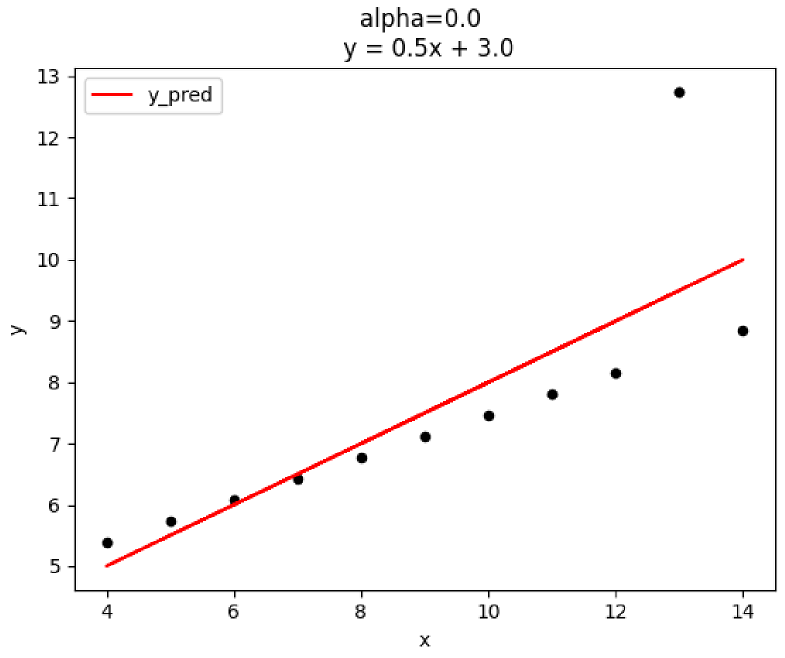


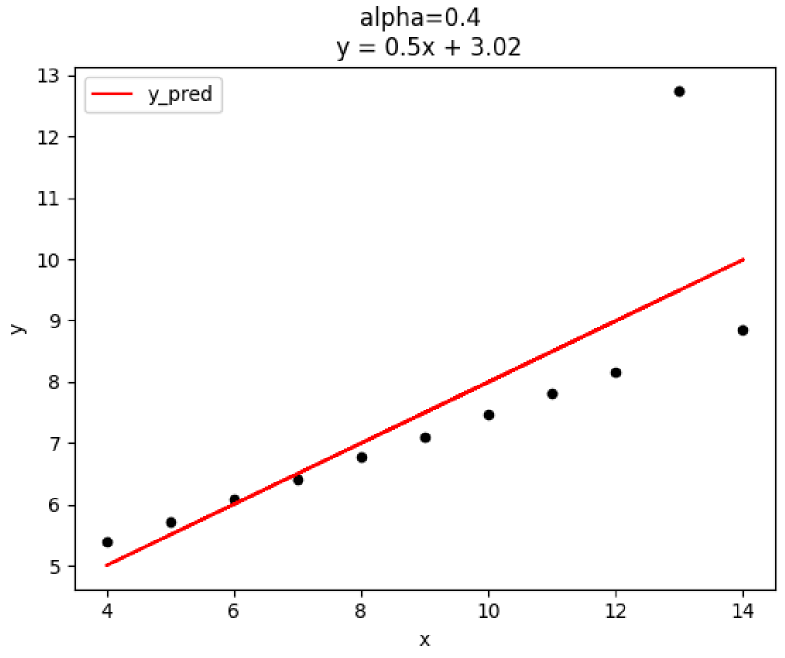

```
# 다중 선형 회귀(OLS)
from sklearn.linear_model import LinearRegression
%matplotlib inline

ax = ans.plot.scatter('x', 'y')

# OLS
ols = LinearRegression()
ols.fit(ans[['x']], ans['y'])

# 회귀계수와 intercept 확인
m = ols.coef_[0].round(2)
b = ols.intercept_.round(2)
title = f'Linear Regression \n y = {m}x + {b}'

# 훈련 데이터로 예측
ans['y_pred'] = ols.predict(ans[['x']])

ans.plot('x', 'y_pred', ax=ax, title=title);
```

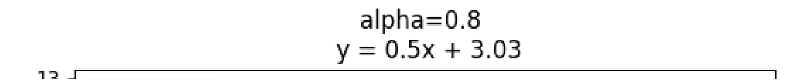


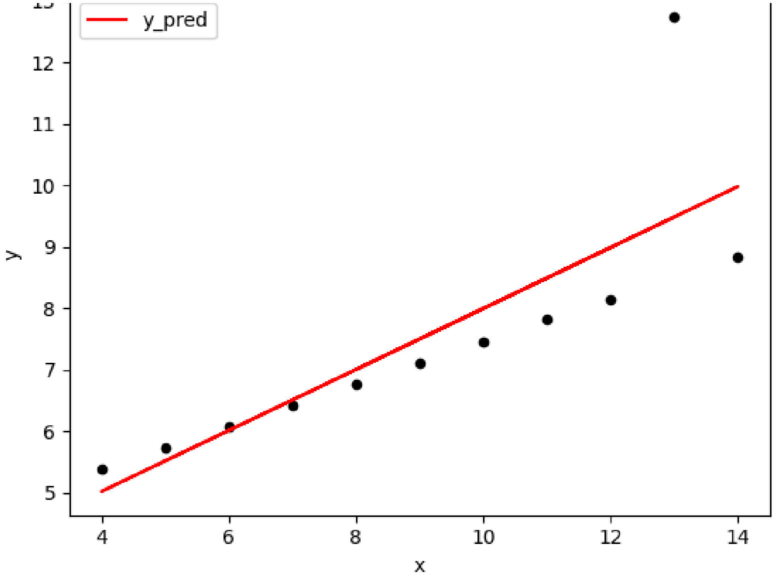

```python
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.linear_model import Ridge

# 데이터셋 load
df = sns.load_dataset('anscombe').query('dataset=="III"')

# λ 값을 변화 시키면서 회귀 계수의 변화를 확인
def ridge_ans(alpha):
    ridge = Ridge(alpha=alpha)
#     ridge = Ridge(alpha=alpha, normalize=True)
    ridge.fit(df[['x']], df['y'])

    df['y_pred'] = ridge.predict(df[['x']])

    #시각화 표현
    m = ridge.coef_[0].round(2)
    b = ridge.intercept_.round(2)
    title = f'alpha={alpha} ₩n y = {m}x + {b}'

    ax = df.plot.scatter('x', 'y', c='black')
    df.plot('x', 'y_pred', ax=ax, c='r', title=title)

    plt.show()

```

```
alphas = np.arange(0, 2.1, 0.4)
for alpha in alphas:
    ridge_ans(alpha=alpha)
```

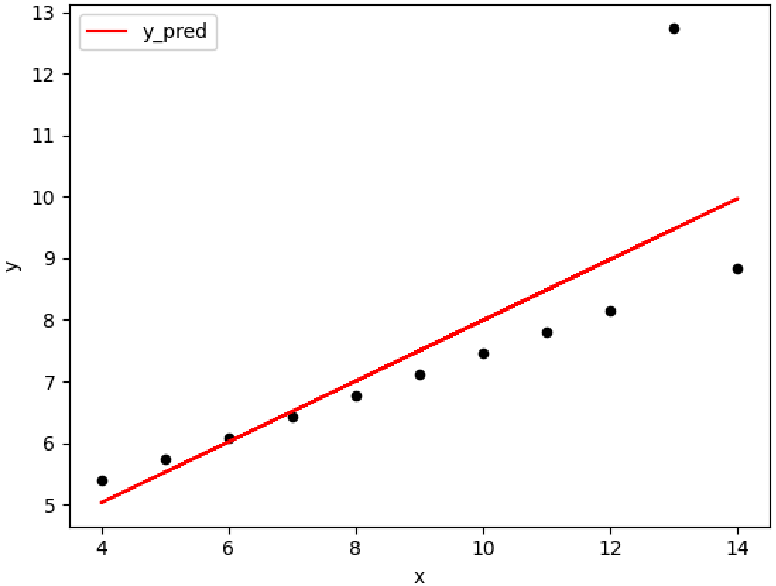

alpha=0.0
y = 0.5x + 3.0
y_pred
y
x


alpha=0.4
y = 0.5x + 3.02
y_pred
y
x

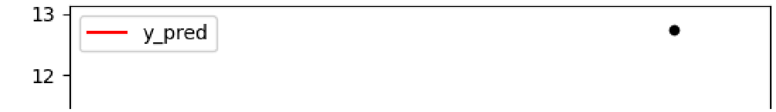

alpha=0.8
y = 0.5x + 3.03

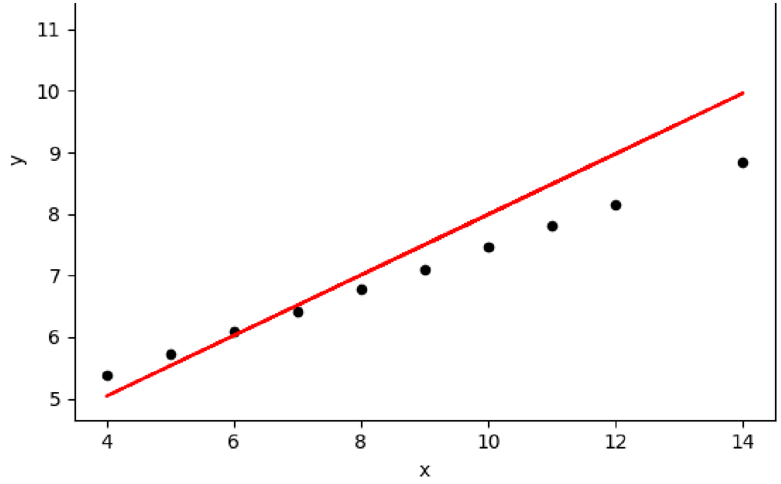


alpha=1.2000000000000002
y = 0.49x + 3.05



alpha=1.6
y = 0.49x + 3.07





alpha=2.0
y = 0.49x + 3.08