4장-회귀 분석을 사용한 콜센터 볼륨 예측

선형 회귀

- 실제 데이터에 선형 회귀를 적용하고,
- 가우스 분포를 사용하여 예측하며,
- 회귀 모델의 예측 정확도 평가

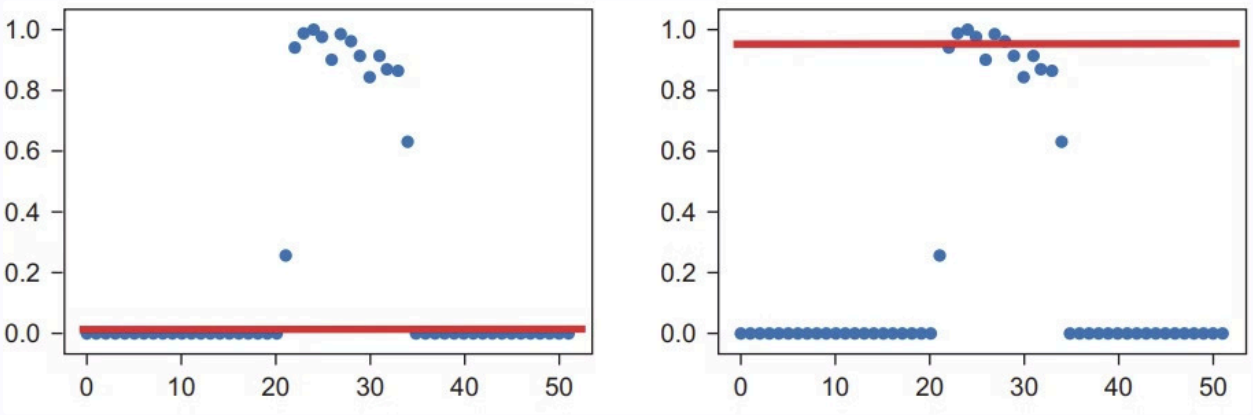
머신러닝 기반 예측 모델

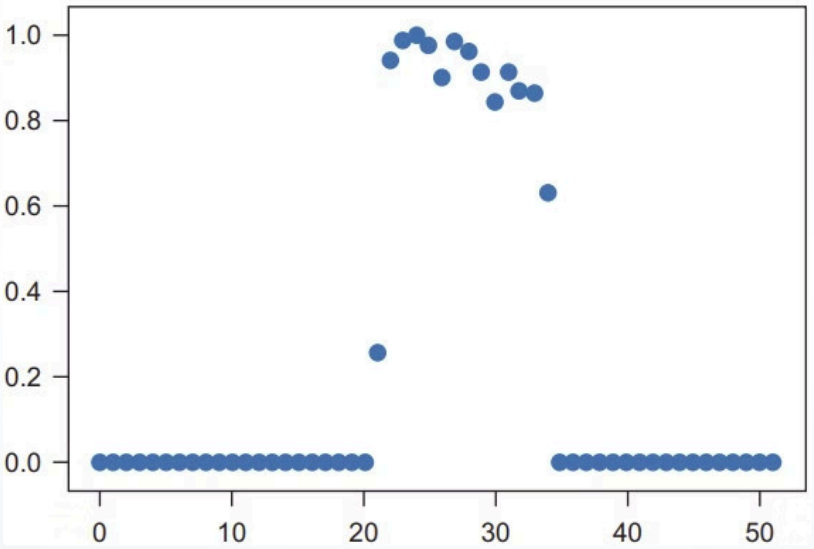
- 데이터 정제
- 모델 구축
- 모델 성능 평가 및 개선

안재목 교수 - 한림대 소프트웨어학부

실제 데이터의 복잡성

- 선형 회귀 모델을 사용한 두 개의 최적 적합선이나, 적합선이 실제 데이터와 크게 벗어남.
- 다항식 모델도 마찬가지로 예측이 크게 빗나감. 데이터가 x축을 따라 어느 위치에서 y값이 크게 증가하고 감소하기 때문.





실제 데이터에 대한 질문들

회귀의 유용성

- 회귀가 모든 데이터 포인트 집합을 예측하는 데 도움이 될 수 있을까?

다양한 회귀 모델

- 선형과 다항식 곡선 외에 어떤 회귀 모델이 존재할까?

데이터 준비와 모델 선택

실제 데이터는 전통적으로 회귀로 쉽게 적용 가능한 (x, y) 포인트 형태가 아님

실제 데이터에 가장 잘 맞는 모델을 어떻게 찾을 수 있을까?

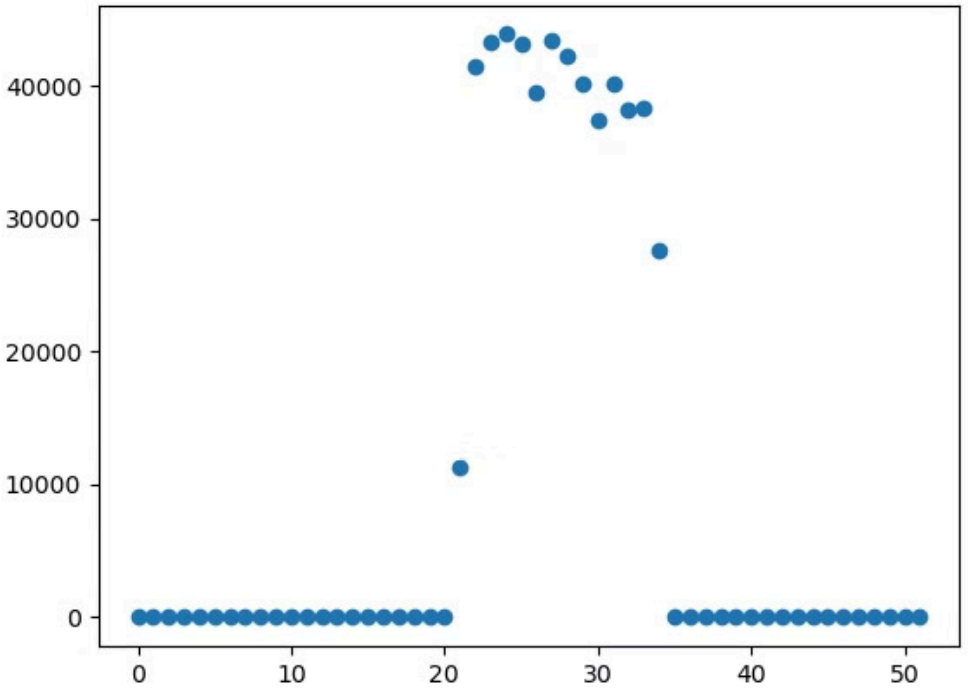
Kaggle 소개

- 오픈 기계 학습 플랫폼: **https://www.kaggle.com**
- 데이터셋, 문서, 공유 가능한 코드 제공
- TensorFlow 기반 노트북 및 코드 지원
- 다양한 시간 기반 데이터셋 제공
- 회귀 모델 (예: 부동산 가격, NYC 311 데이터셋)

부동산 가격 오픋 데이터셋 (**http://mng.bz/6Ady**)

뉴욕시(NYC) 311 오픈 데이터셋 (**http://mng.bz/1gPX**)

뉴욕시 Open Data 포털 **http://opendata.cityofnewyork.us**



311콜센터 통화량 예측

311 통화량 예측의 중요성

주어진 달의 서비스 통화량의 예측은 1년 동안의 통화량과 관련된 날짜와 시간을 살펴보고, 그 통화를 주간 단위로 집계하여 x 값이 주 번호(1-52, 또는 365일을 7일로 나눈 값)이고 y 값이 특정 주의 통화 수인 포인트 집합을 구성함.

311콜센터의 통화량 예측 과정

1. **데이터 정제 및 시각화**

   y축에 통화 수를, x축에 주 번호(1-52)를 플롯.

2. **추세 분석**

   추세를 검토하여 선, 곡선 또는 다른 것과 유사한지 확인.

3. **모델 선택 및 훈련**

   데이터 포인트(주 번호 및 통화 수)에 가장 잘 맞는 회귀 모델을 선택하고 훈련시킴.

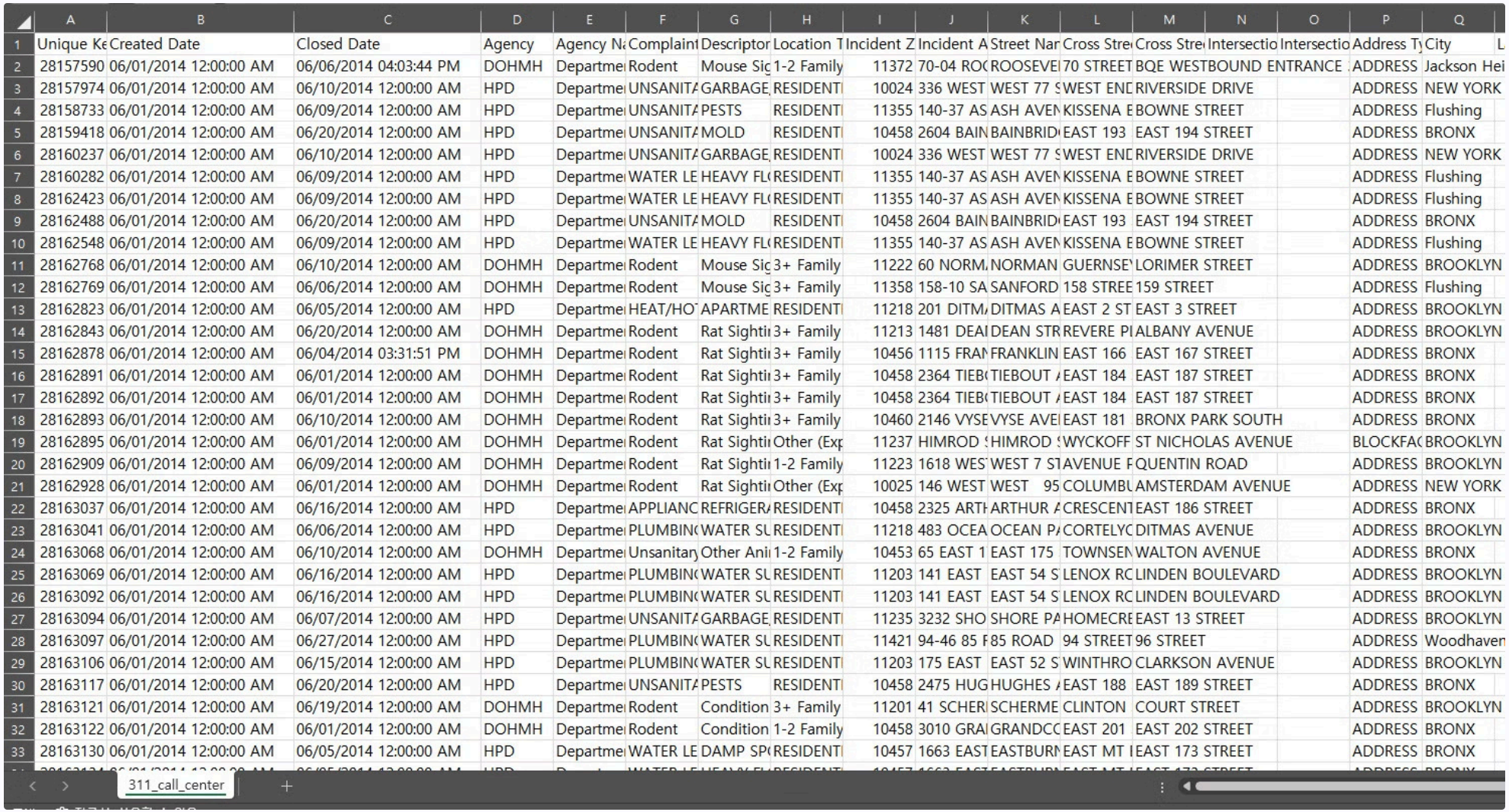
4. **모델 평가**

   오차를 계산하고 시각화하여 모델의 성능을 평가함.

5. **예측 수행**

   새로운 모델을 사용하여 311이 주어진 주, 계절 및 연도에 예상할 수 있는 통화 수를 예측함.

실제 데이터 (xls)



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	Unique Ke	Created Date	Closed Date	Agency	Agency Na	Complaint	Descriptor	Location T	Incident Z	Incident A	Street Nar	Cross Stre	Cross Stre	Intersectio	Intersectio	Address Ty	City
2	28157590	06/01/2014 12:00:00 AM	06/06/2014 04:03:44 PM	DOHMH	Departme	Rodent	Mouse Sig	1-2 Family	11372	70-04 RO(	ROOSEVEI	70 STREET	BQE WEST	BOUND ENTRANCE		ADDRESS	Jackson Hei
3	28157974	06/01/2014 12:00:00 AM	06/10/2014 12:00:00 AM	HPD	Departme	UNSANITA	GARBAGE	RESIDENTI	10024	336 WEST	WEST 77 S	WEST END	RIVERSIDE	DRIVE		ADDRESS	NEW YORK
4	28158733	06/01/2014 12:00:00 AM	06/09/2014 12:00:00 AM	HPD	Departme	UNSANITA	PESTS	RESIDENTI	11355	140-37 AS	ASH AVEN	KISSENA E	BOWNE	STREET		ADDRESS	Flushing
5	28159418	06/01/2014 12:00:00 AM	06/20/2014 12:00:00 AM	HPD	Departme	UNSANITA	MOLD	RESIDENTI	10458	2604 BAIN	BAINBRID(	EAST 193	EAST 194	STREET		ADDRESS	BRONX
6	28160237	06/01/2014 12:00:00 AM	06/10/2014 12:00:00 AM	HPD	Departme	UNSANITA	GARBAGE	RESIDENTI	10024	336 WEST	WEST 77 S	WEST END	RIVERSIDE	DRIVE		ADDRESS	NEW YORK
7	28160282	06/01/2014 12:00:00 AM	06/09/2014 12:00:00 AM	HPD	Departme	WATER LE	HEAVY FL(	RESIDENTI	11355	140-37 AS	ASH AVEN	KISSENA E	BOWNE	STREET		ADDRESS	Flushing
8	28162423	06/01/2014 12:00:00 AM	06/09/2014 12:00:00 AM	HPD	Departme	WATER LE	HEAVY FL(	RESIDENTI	11355	140-37 AS	ASH AVEN	KISSENA E	BOWNE	STREET		ADDRESS	Flushing
9	28162488	06/01/2014 12:00:00 AM	06/20/2014 12:00:00 AM	HPD	Departme	UNSANITA	MOLD	RESIDENTI	10458	2604 BAIN	BAINBRID(	EAST 193	EAST 194	STREET		ADDRESS	BRONX
10	28162548	06/01/2014 12:00:00 AM	06/09/2014 12:00:00 AM	HPD	Departme	WATER LE	HEAVY FL(	RESIDENTI	11355	140-37 AS	ASH AVEN	KISSENA E	BOWNE	STREET		ADDRESS	Flushing
11	28162768	06/01/2014 12:00:00 AM	06/10/2014 12:00:00 AM	DOHMH	Departme	Rodent	Mouse Sig	3+ Family	11222	60 NORM/	NORMAN	GUERNSE'	LORIMER	STREET		ADDRESS	BROOKLYN
12	28162769	06/01/2014 12:00:00 AM	06/06/2014 12:00:00 AM	DOHMH	Departme	Rodent	Mouse Sig	3+ Family	11358	158-10 SA	SANFORD	158 STREE	159 STREET			ADDRESS	Flushing
13	28162823	06/01/2014 12:00:00 AM	06/05/2014 12:00:00 AM	HPD	Departme	HEAT/HO	APARTME	RESIDENTI	11218	201 DITM/	DITMAS A	EAST 2 ST	EAST 3	STREET		ADDRESS	BROOKLYN
14	28162843	06/01/2014 12:00:00 AM	06/20/2014 12:00:00 AM	DOHMH	Departme	Rodent	Rat Sightii	3+ Family	11213	1481 DEAI	DEAN STR	REVERE PI	ALBANY	AVENUE		ADDRESS	BROOKLYN
15	28162878	06/01/2014 12:00:00 AM	06/04/2014 03:31:51 PM	DOHMH	Departme	Rodent	Rat Sightii	3+ Family	10456	1115 FRAI	FRANKLIN	EAST 166	EAST 167	STREET		ADDRESS	BRONX
16	28162891	06/01/2014 12:00:00 AM	06/01/2014 12:00:00 AM	DOHMH	Departme	Rodent	Rat Sightii	3+ Family	10458	2364 TIEB(	TIEBOUT /	EAST 184	EAST 187	STREET		ADDRESS	BRONX
17	28162892	06/01/2014 12:00:00 AM	06/01/2014 12:00:00 AM	DOHMH	Departme	Rodent	Rat Sightii	3+ Family	10458	2364 TIEB(	TIEBOUT /	EAST 184	EAST 187	STREET		ADDRESS	BRONX
18	28162893	06/01/2014 12:00:00 AM	06/10/2014 12:00:00 AM	DOHMH	Departme	Rodent	Rat Sightii	3+ Family	10460	2146 VYSE	VYSE AVE	EAST 181	BRONX PARK	SOUTH		ADDRESS	BRONX
19	28162895	06/01/2014 12:00:00 AM	06/01/2014 12:00:00 AM	DOHMH	Departme	Rodent	Rat Sightii	Other (Exp	11237	HIMROD S	HIMROD S	WYCKOFF	ST NICHOLAS	AVENUE		BLOCKFA(	BROOKLYN
20	28162909	06/01/2014 12:00:00 AM	06/09/2014 12:00:00 AM	DOHMH	Departme	Rodent	Rat Sightii	1-2 Family	11223	1618 WES	WEST 7 ST	AVENUE F	QUENTIN	ROAD		ADDRESS	BROOKLYN
21	28162928	06/01/2014 12:00:00 AM	06/01/2014 12:00:00 AM	DOHMH	Departme	Rodent	Rat Sightii	Other (Exp	10025	146 WEST	WEST 95	COLUMBU	AMSTERDAM	AVENUE		ADDRESS	NEW YORK
22	28163037	06/01/2014 12:00:00 AM	06/16/2014 12:00:00 AM	HPD	Departme	APPLIANC	REFRIGER/	RESIDENTI	10458	2325 ARTH	ARTHUR A	CRESCENT	EAST 186	STREET		ADDRESS	BRONX
23	28163041	06/01/2014 12:00:00 AM	06/06/2014 12:00:00 AM	HPD	Departme	PLUMBIN(	WATER SU	RESIDENTI	11218	483 OCEA	OCEAN PA	CORTELYC	DITMAS	AVENUE		ADDRESS	BROOKLYN
24	28163068	06/01/2014 12:00:00 AM	06/10/2014 12:00:00 AM	DOHMH	Departme	Unsanitary	Other Ani	1-2 Family	10453	65 EAST 1	EAST 175	TOWNSEN	WALTON	AVENUE		ADDRESS	BRONX
25	28163069	06/01/2014 12:00:00 AM	06/16/2014 12:00:00 AM	HPD	Departme	PLUMBIN(	WATER SU	RESIDENTI	11203	141 EAST	EAST 54 S	LENOX RC	LINDEN	BOULEVARD		ADDRESS	BROOKLYN
26	28163092	06/01/2014 12:00:00 AM	06/16/2014 12:00:00 AM	HPD	Departme	PLUMBIN(	WATER SU	RESIDENTI	11203	141 EAST	EAST 54 S	LENOX RC	LINDEN	BOULEVARD		ADDRESS	BROOKLYN
27	28163094	06/01/2014 12:00:00 AM	06/07/2014 12:00:00 AM	HPD	Departme	UNSANITA	GARBAGE	RESIDENTI	11235	3232 SHO	SHORE PA	HOMECRE	EAST 13	STREET		ADDRESS	BROOKLYN
28	28163097	06/01/2014 12:00:00 AM	06/27/2014 12:00:00 AM	HPD	Departme	PLUMBIN(	WATER SU	RESIDENTI	11421	94-46 85 F	85 ROAD	94 STREET	96	STREET		ADDRESS	Woodhaven
29	28163106	06/01/2014 12:00:00 AM	06/15/2014 12:00:00 AM	HPD	Departme	PLUMBIN(	WATER SU	RESIDENTI	11203	175 EAST	EAST 52 S	WINTHRO	CLARKSON	AVENUE		ADDRESS	BROOKLYN
30	28163117	06/01/2014 12:00:00 AM	06/20/2014 12:00:00 AM	HPD	Departme	UNSANITA	PESTS	RESIDENTI	10458	2475 HUG	HUGHES /	EAST 188	EAST 189	STREET		ADDRESS	BRONX
31	28163121	06/01/2014 12:00:00 AM	06/19/2014 12:00:00 AM	DOHMH	Departme	Rodent	Condition	3+ Family	11201	41 SCHER	SCHERME	CLINTON	COURT	STREET		ADDRESS	BROOKLYN
32	28163122	06/01/2014 12:00:00 AM	06/01/2014 12:00:00 AM	DOHMH	Departme	Rodent	Condition	1-2 Family	10458	3010 GRA	GRANDCC	EAST 201	EAST 202	STREET		ADDRESS	BRONX
33	28163130	06/01/2014 12:00:00 AM	06/05/2014 12:00:00 AM	HPD	Departme	WATER LE	DAMP SP(	RESIDENTI	10457	1663 EAST	EASTBURN	EAST MT I	EAST 173	STREET		ADDRESS	BRONX

311_call_center

데이터 정제

- **read 함수:** 날짜 필드가 인덱스 1(또는 0 인덱스 기준 두 번째 열)에 있고, 날짜가 'month/day/year hour:minutes:seconds AM/PM')과 같은 문자열로 포맷.
- **freq 딕션너리 변수:** 주당 및 연도당 통화량.
- **read 함수 인수:** filename은 파일 이름, date_idx는 날짜 열의 인덱스, date_parse는 날짜 형식, year는 훈련할 연도, bucket은 7일 또는 주간 빈 크기.

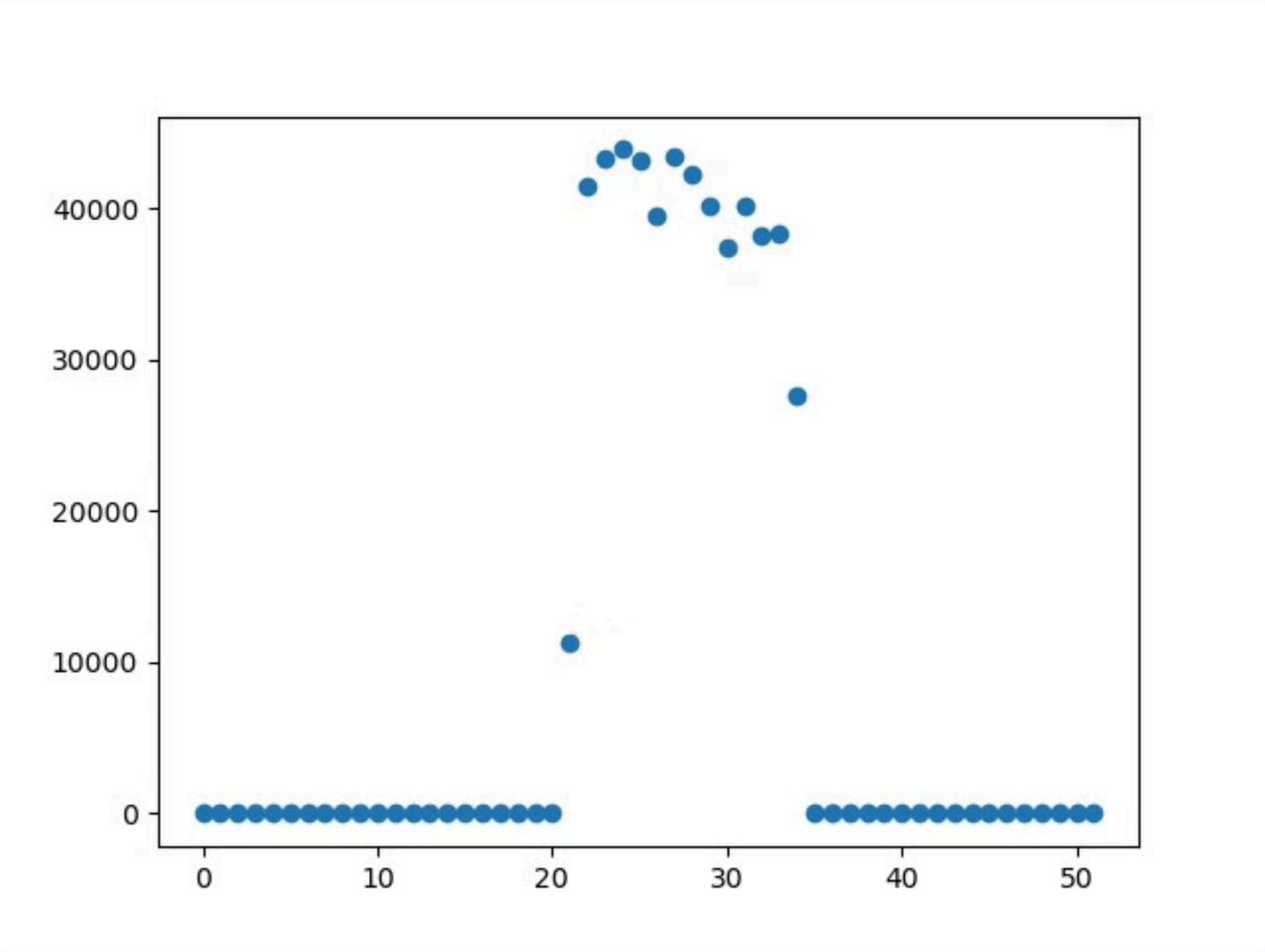
```
def read(filename, date_idx, date_parse, year=None, bucket=7):
    days_in_year = 365
    freq = {}
    if year != None:
        for period in range(0, int(days_in_year / bucket)):
            freq[period] = 0
    with open(filename, 'r') as csvfile:
        csvreader = csv.reader(csvfile)
        next(csvreader)
        for row in csvreader:
            if row[date_idx] == '':
                continue
            t = time.strptime(row[date_idx], date_parse)
            if year == None:
                if not t.tm_year in freq:
                    freq[t.tm_year] = {}
                    for period in range(0, int(days_in_year / bucket)):
                        freq[t.tm_year][period] = 0
                if t.tm_yday < (days_in_year - 1):
                    freq[t.tm_year][int(t.tm_yday / bucket)] += 1
            else:
                if t.tm_year == year and t.tm_yday < (days_in_year-1):
                    freq[int(t.tm_yday / bucket)] += 1
    return freq

freq = read('C:\\Users\\,\\311_call_center.csv', 1, '%m/%d/%Y %H:%M:%S %p', 2014)
```

데이터 정제 결과 확인

- 결과는 주당 통화 수가 있는 52주(0에서 인덱싱되므로 0-51) 히스토그램
- 출력에서 알 수 있듯이 데이터는 22주에서 35주 사이에 클러스터되어 있으며, 이는 2014년 5월 26일부터 8월 25일까지에 해당

freq {0: 0, 1: 0, 2: 0, 3: 0, 4: 0, 5: 0, 6: 0, 7: 0, 8: 0, 9: 0, 10: 0, 11: 0, 12: 0, 13: 0, 14: 0, 15: 0, 16: 0, 17: 0, 18: 0, 19: 0, 20: 0, 21: 10889, 22: 40240, 23: 42125, 24: 42673, 25: 41721, 26: 38446, 27: 41915, 28: 41008, 29: 39011, 30: 36069, 31: 38821, 32: 37050, 33: 36967, 34: 26834, 35: 0, 36: 0, 37: 0, 38: 0, 39: 0, 40: 0, 41: 0, 42: 0, 43: 0, 44: 0, 45: 0, 46: 0, 47: 0, 48: 0, 49: 0, 50: 0, 51: 0}

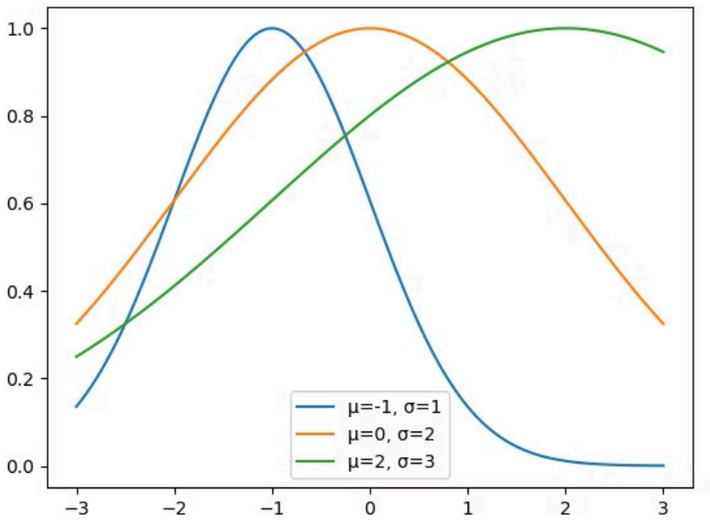
데이터 정제 시각화

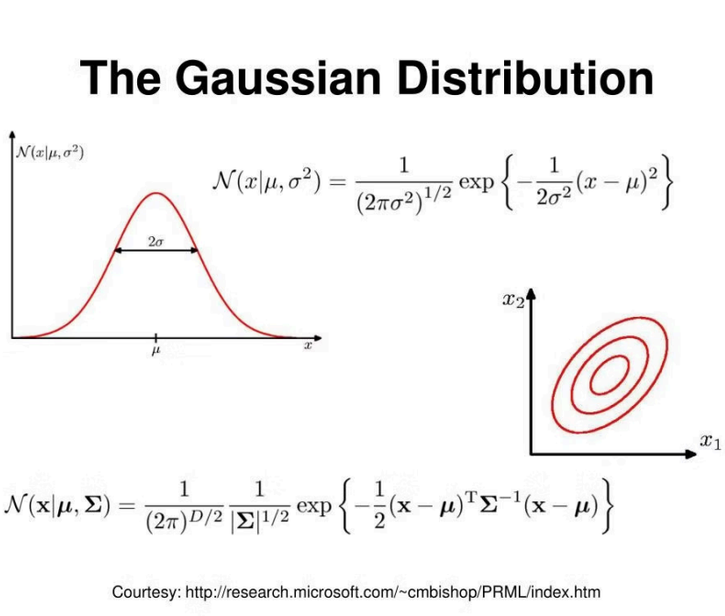



```
X_train = np.asarray(list(freq.keys()))
Y_train = np.asarray(list(freq.values()))
plt.scatter(X_train,Y_train)
```

가우시안 분포

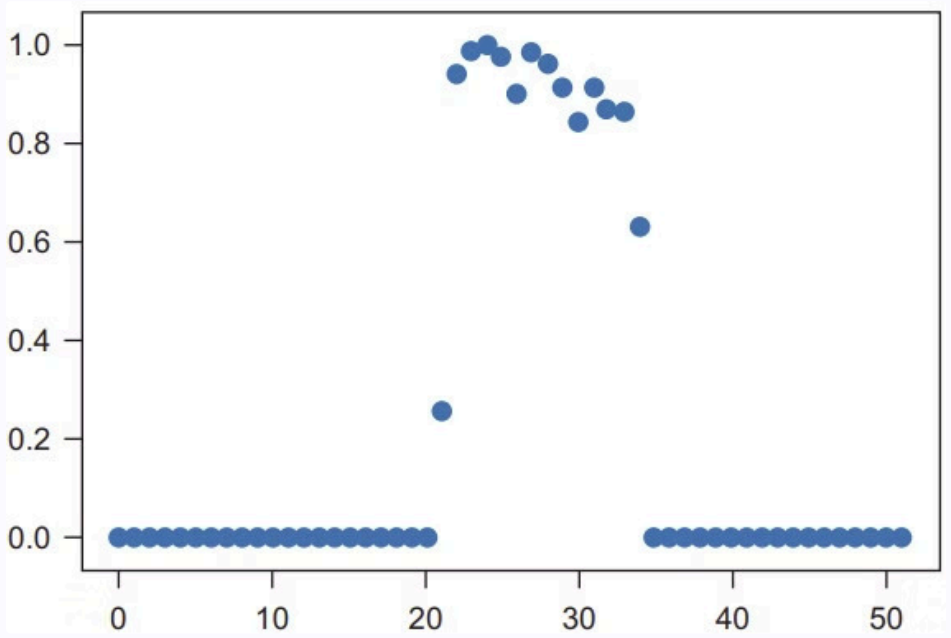
```
import numpy as np

import matplotlib.pyplot as plt

def gaussian(x, mu, sig):

    return np.exp(-np.power(x - mu, 2.) / (2 * np.power(sig, 2.)))

x_values = np.linspace(-3, 3, 120)

for mu, sig in [(-1, 1), (0, 2), (2, 3)]:

    plt.plot(x_values, gaussian(x_values, mu, sig),label=f'μ={mu}, σ={sig}')

plt.legend()

plt.show()
```





정규화

```
maxY = np.max(Y_train)
nY_train = Y_train / np.max(Y_train)
plt.scatter(X_train,nY_train)
```



가우시안 모델 구축

```
class Model:
    def __init__(self):
        self.mu = tf.Variable(1.0, dtype=tf.float32)
        self.sig = tf.Variable(1.0, dtype=tf.float32)
    def __call__(self, x):
        x_c = tf.cast(x, tf.float32)
        return tf.exp(-tf.pow(x_c - self.mu, 2.) / (2. * tf.pow(self.sig, 2.)))
```

- 모델 클래스에는 평균(mu)과 표준편차(sig) 매개변수
- 이 매개변수들은 tf.Variable로 정의되어 학습 가능함
- **call** 메서드에서는 입력 데이터 x에 대해 가우시안 분포 확률 밀도 함수 계산하여 가우시안 모델 출력

Normal Distribution Formula

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right)$$

비용 함수 (cost) vs 손실 함수 (loss)

```
def cost_function(predicted_y, desired_y):
    loss_function = tf.square(predicted_y - desired_y)
    return tf.reduce_mean(loss_function)
```

비용 함수와 손실 함수의 개념 차이

비용 함수와 손실 함수는 모두 모델의 성능을 측정하는 지표이지만, 그 의미와 용도에서 차이가 있음

비용 함수

모델의 전체 오류를 측정

모든 데이터 포인트의 손실을 평균화

모델 학습 과정에서 최소화하려는 목적 함수

손실 함수

모델의 예측 오류를 측정

단일 데이터 포인트에 대한 오류를 측정

비용 함수를 구성하는 기본 요소

Parameters: θ_0, θ_1

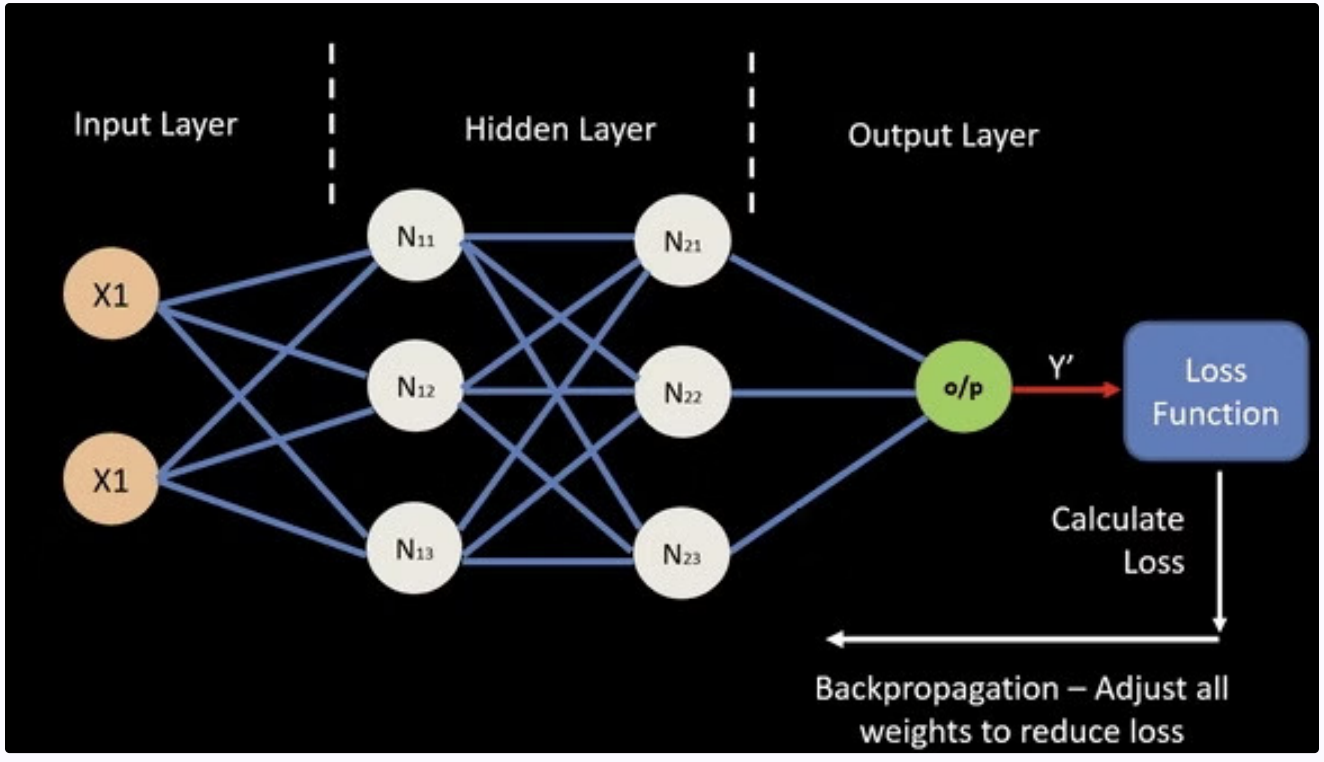
Cost Function: $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$

비용 함수와 손실 함수의 수학적 표현

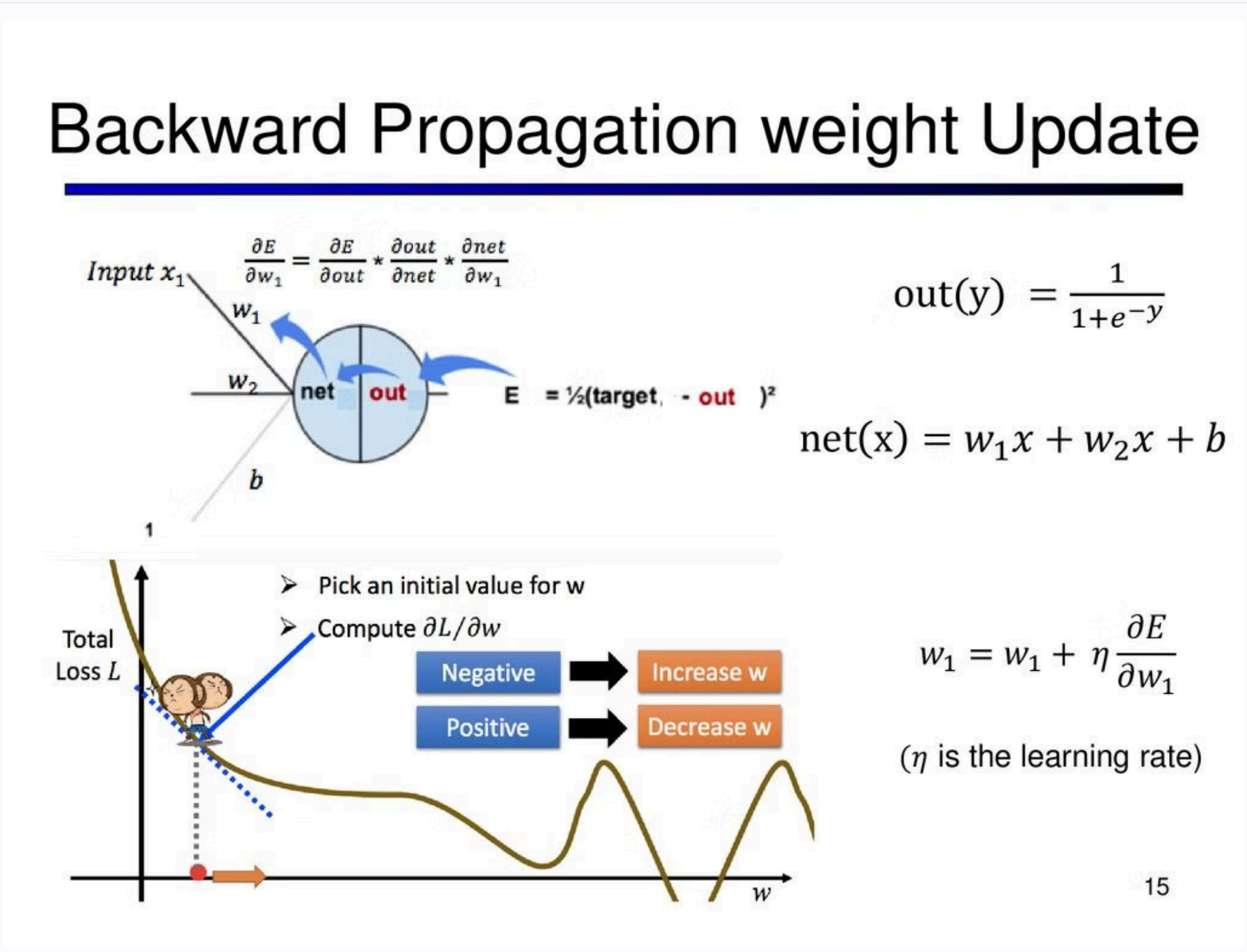
평균 제곱 오차 (MSE): J(w,b)=(1/2m) ∑ (h(xi)−yi)^2	제곱 오차 (SE): (h(xi)−yi)^2
교차 엔트로피 오차 (CEE, cross entropy error): Log Loss=−(1/m) ∑ [yi*log(pi)+(1−yi)*log(1−pi)]	로그 손실 (Log Loss): [yi*log(pi)+(1−yi)*log(1−pi)], pi는 i번째 데이터 포인트에 대한 모델의 예측 확률이며, 0과 1 사이의 실수 값을 가짐
평균 절대 오차 (MAE): MAE=(1/m) ∑ ∣yi−y^i∣	절대 오차 (AE): ∣yi−y^i∣

경사 하강 알고리즘 vs 역전파

1. 경사 하강 알고리즘의 목적 함수 = 비용 함수
2. 역전파는 비용함수를 계산한 후, 그 오차를 줄이기 위해 각 층의 가중치를 업데이트하는 알고리즘
3. 역전파를 통해 모델의 파라미터를 조정하여 비용 함수를 최소화하는 것이 경사 하강 알고리즘의 목표



가중치 업데이트



- ∂E/∂w는 특정 샘플에 대한 손실 함수의 기울기. 손실함수
- ∂L/∂w는 전체 데이터셋에 대한 총 손실 함수의 기울기. 비용함수

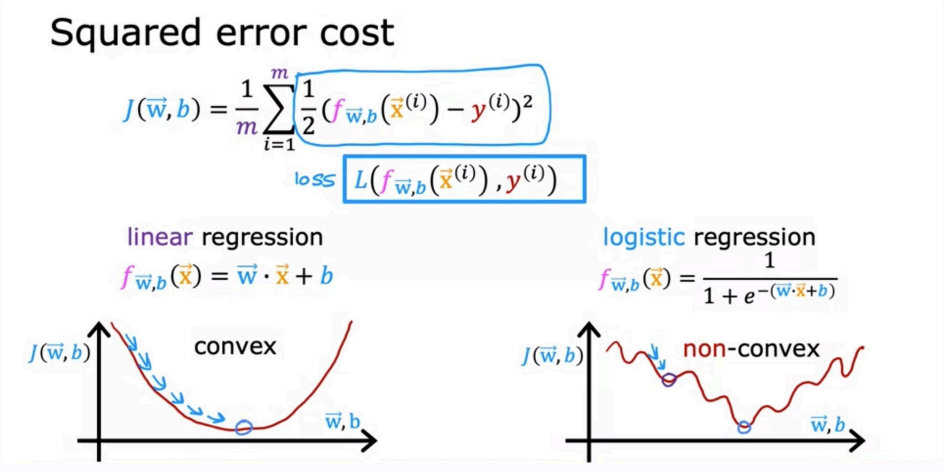
경사 하강 알고리즘 코드

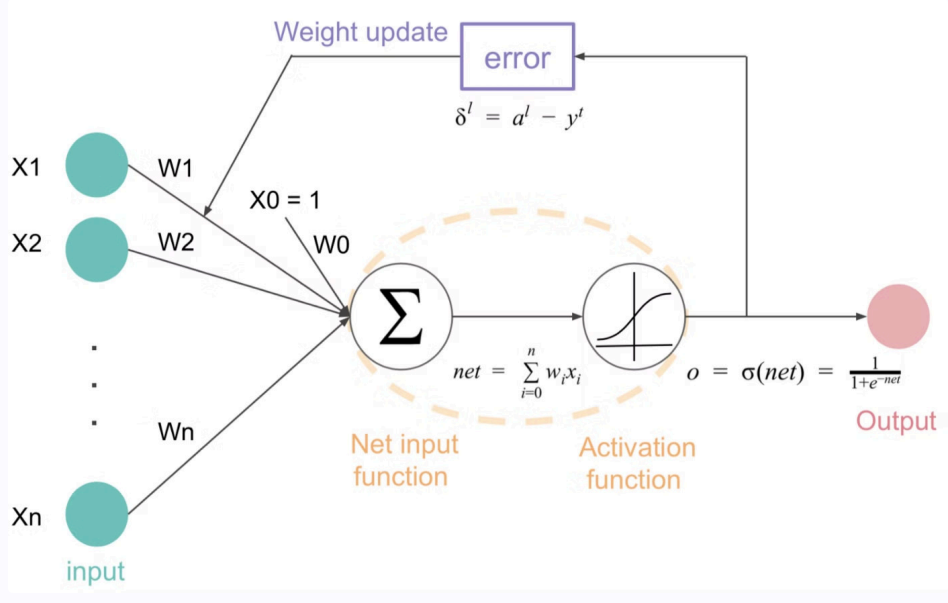
```
learning_rate = 1.5
optimizer = tf.keras.optimizers.SGD(learning_rate)

def train_step(model, inputs, outputs):
    with tf.GradientTape() as t:
        current_cost_function = cost_function(model(inputs), outputs)

    grads = t.gradient(current_cost_function, [model.mu, model.sig])
    optimizer.apply_gradients(zip(grads,[model.mu, model.sig]))
    return current_cost_function
```

- 학습률, 경사 하강 최적화 방법을 정의함. 여기서는 SGD(확률적 경사하강,stochastic gradient descent)사용
- train_step 함수는
  - 모델의 출력과 실제 출력 간의 비용 함수 계산
  - 비용 함수에 대한 모델 매개변수의 gradient를 계산
  - 계산된 gradient를 사용하여 모델 매개변수를 업데이트 함
- 이 함수는 현재 비용 함수 값을 반환함

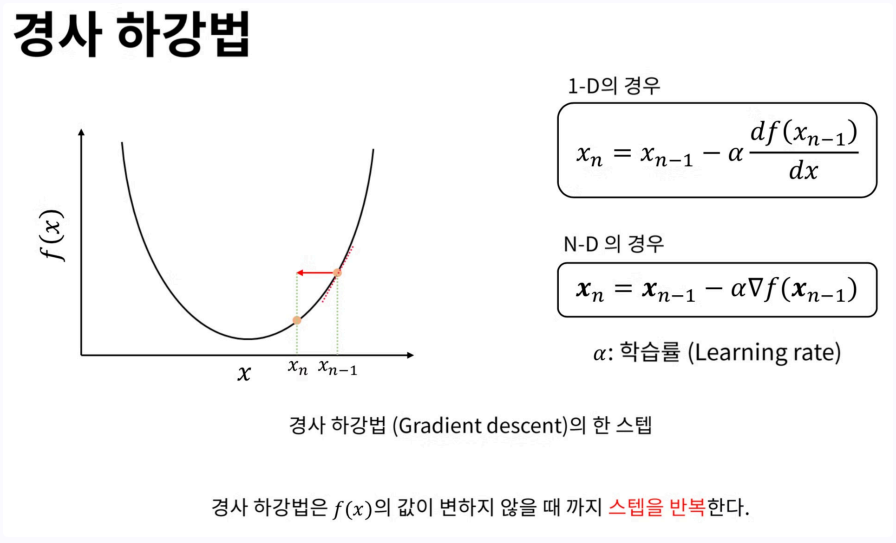
경사 하강 알고리즘 vs 학습률

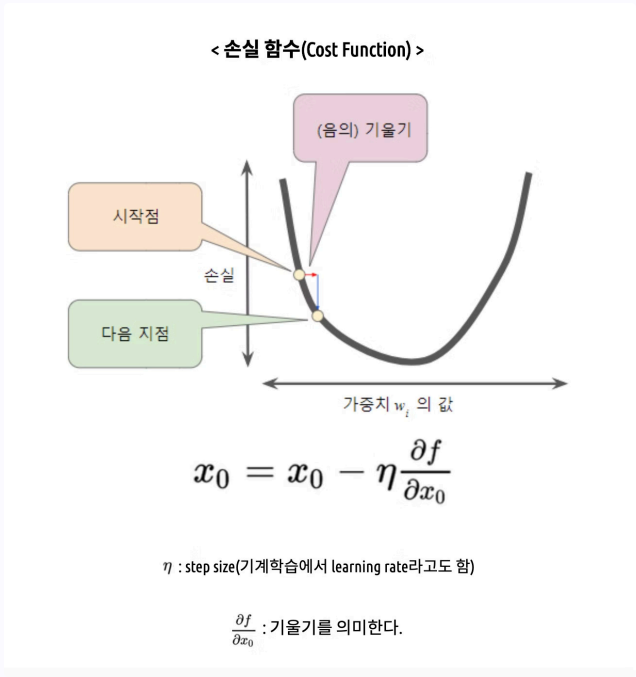
경사 하강 알고리즘은 비용 함수의 기울기를 사용하여 모델의 파라미터를 업데이트





학습률(에타)은 경사하강 알고리즘에서 내려가는 보폭을 결정, 가중치를 업데이트할 때





모델 학습 코드

```
model = Model()

training_epochs = 50
for epoch in range(training_epochs):
    for i in range(0, len(X_train)):
        _cost_function = train_step(model, X_train[i], nY_train[i])
    if epoch % 10 == 0:
        print("Current cost_function %f" % (_cost_function.numpy()))
```

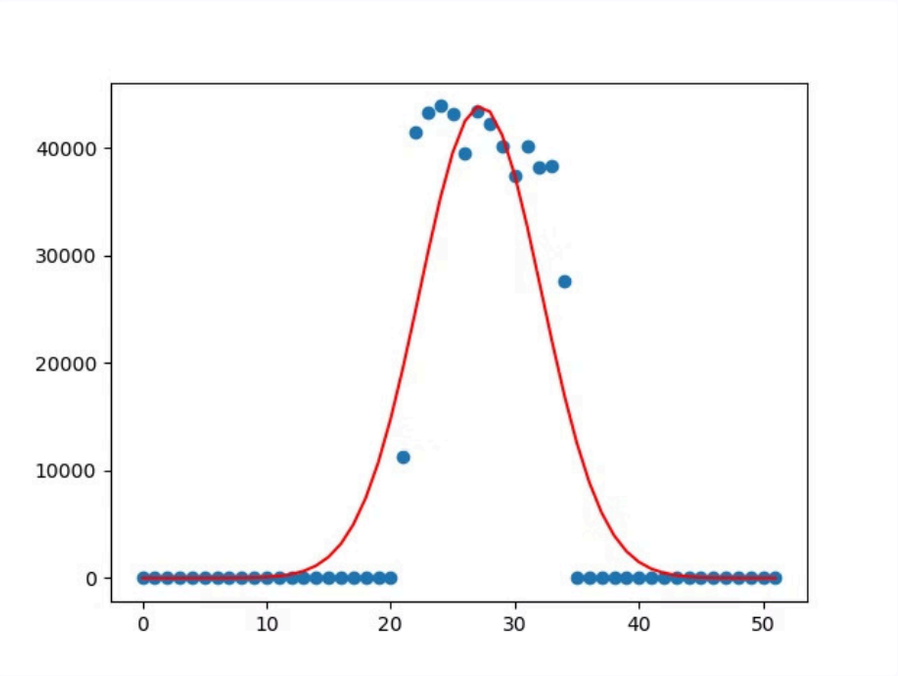
model = Model() 클래스의의 인스턴스 생성

훈련 횟수 설정: training_epochs = 50

두 번째 반복문은 각 에포크마다 `X_train` 데이터셋의 모든 샘플을 사용하여 모델을 학습하고, `train_step` 함수는 모델의 현재 가중치를 업데이트하고, `X_train[i]`와 `nY_train[i]`라는 입력 데이터를 사용해 비용 함수(cost function)의 값을 반환.

매 10번째 에포크마다 비용함수 값을 출력함

출력 시각화-가우시안 분포

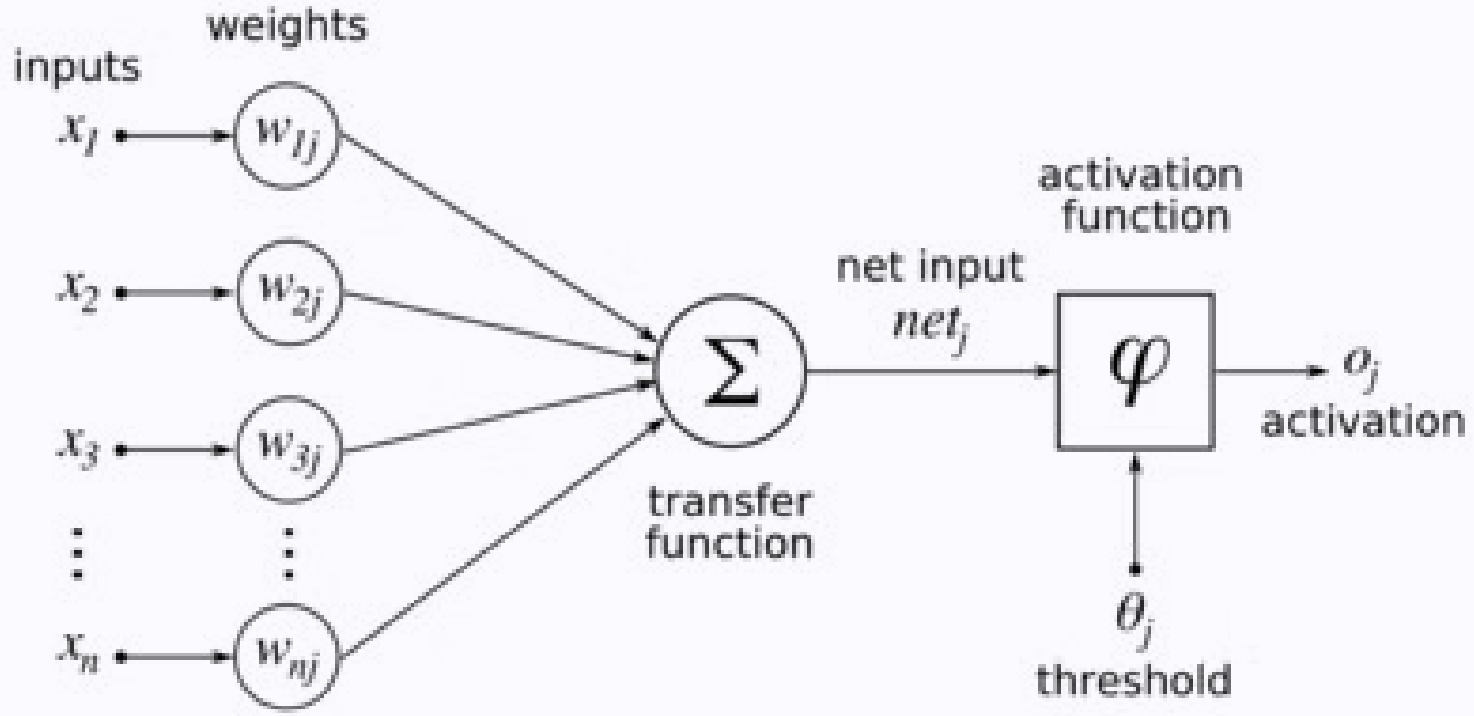



```
mu_val = model.mu
sig_val = model.sig
print(mu_val.numpy())
print(sig_val.numpy())
```

$$e^{\frac{(-(x-\mu)^2)}{2\sigma^2}}$$

```
plt.scatter(X_train, Y_train)
trY2 = maxY * (np.exp(-np.power(X_train - mu_val, 2.) / (2 * np.power(sig_val, 2.))))
plt.plot(X_train, trY2, 'r')
plt.show()
print("Prediction of week 35", trY2[33])
print("Actual week 35", Y_train[33])
```

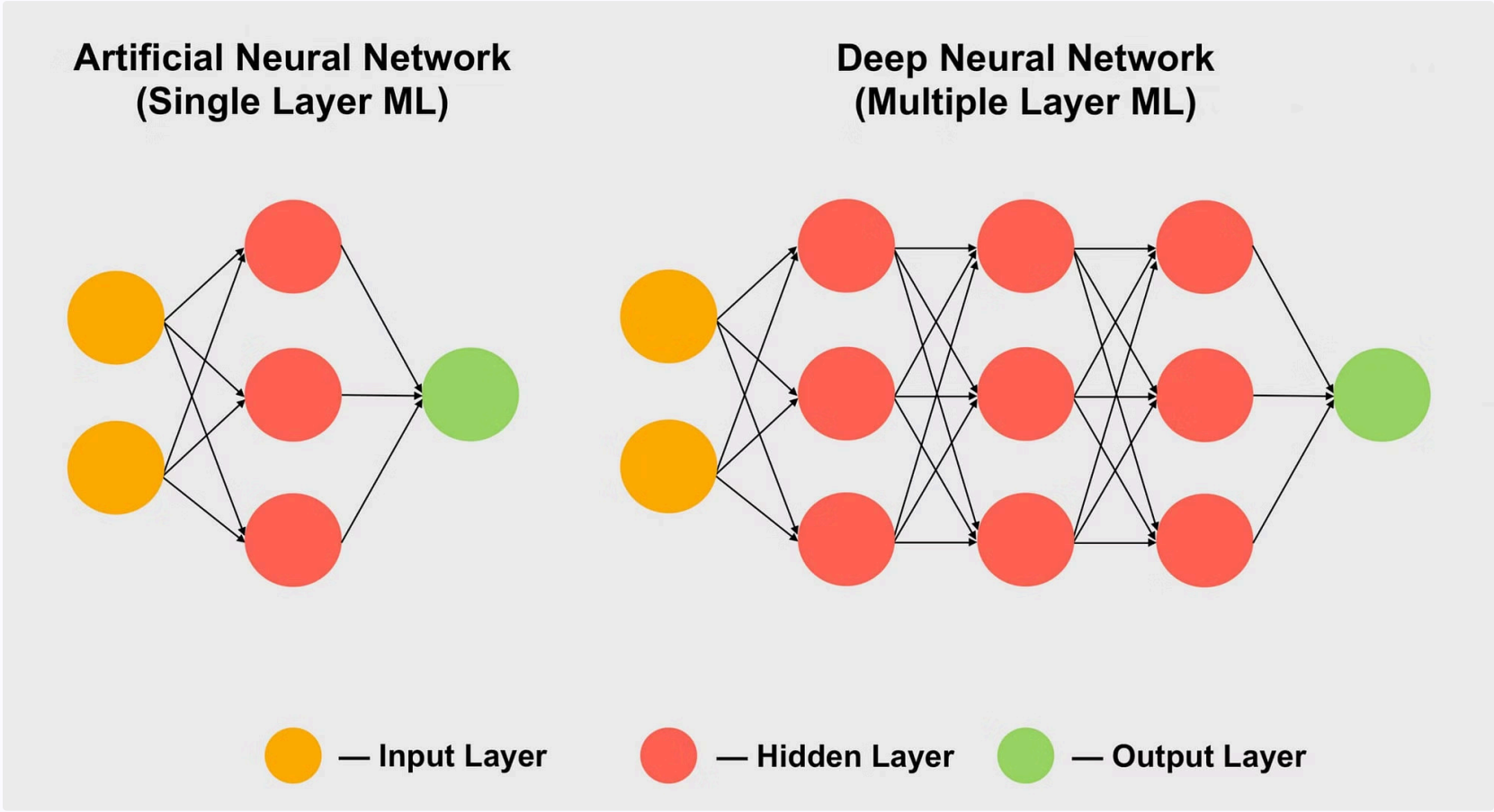
단일 뉴런


weights
inputs
x_1
x_2
x_3
x_n
w_{1j}
w_{2j}
w_{3j}
w_{nj}
Σ
transfer function
net input
net_j
activation function
φ
o_j
activation
θ_j
threshold

딥러닝 모델 코드

```
model = models.Sequential([
    layers.Input(shape=(1,)),
    layers.Dense(32, activation='relu'),
    layers.Dense(16, activation='relu'),
    layers.Dense(8, activation='relu'),
    layers.Dense(1)
])
#정규화
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train.reshape(-1, 1))#1D->2D
Y_train = scaler.fit_transform(Y_train.reshape(-1, 1))#1D->2D

X_train = tf.convert_to_tensor(X_train,dtype=tf.float32)
Y_train = tf.convert_to_tensor(Y_train,dtype=tf.float32)
```

모델 아키텍처


Artificial Neural Network
(Single Layer ML)
Deep Neural Network
(Multiple Layer ML)
— Input Layer
— Hidden Layer
— Output Layer

모델 훈련과 배치 사이즈

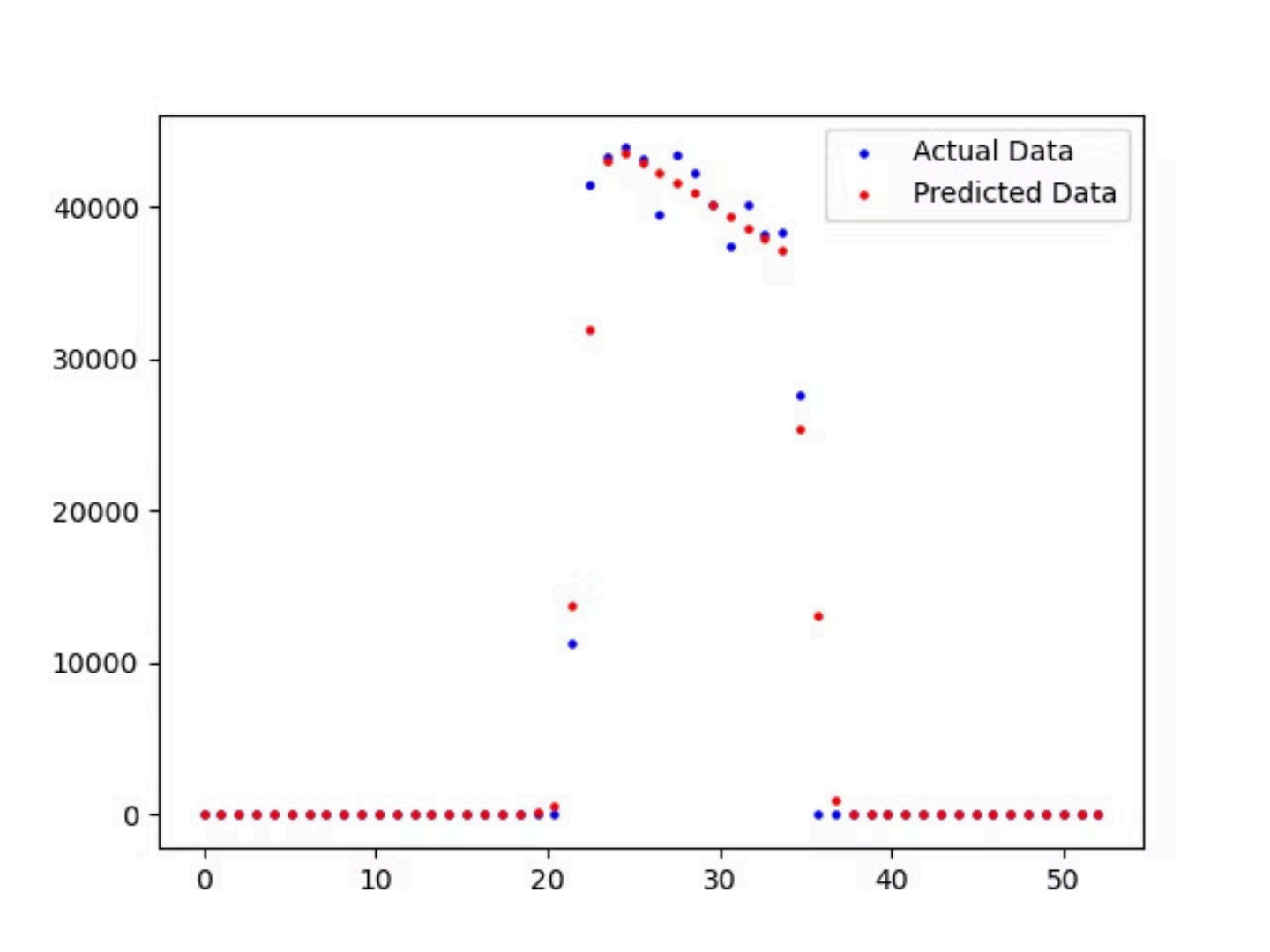
```
learning_rate = 0.1
training_epochs = 1000
optimizer = SGD(learning_rate=learning_rate)
model.compile(optimizer=optimizer, loss='mse', metrics=['accuracy'])
model.summary()
history = model.fit(X_train, Y_train, epochs=training_epochs, batch_size=13)
```

- 배치 사이즈는 한 번에 업데이트 되는 데이터 포인트 수를 결정함. 예를 들어, 전체 학습 데이터셋을 여러 개의 작은 그룹(배치)으로 나누어 한 번에 하나의 배치를 사용하여 모델의 가중치를 업데이트 함
- 배치 사이즈가 너무 작으면 모델이 각 배치에서 많은 업데이트를 수행하게 되므로 훈련 시간이 길어지고 메모리 사용량이 증가함.
- 배치 사이즈가 13라면 모델은 13개의 데이터 포인트를 한 번에 처리하여 가중치를 업데이트 함
- **작은 배치 사이즈**: 더 많은 업데이트를 수행하므로 모델이 더 자주 학습하지만, 노이즈가 많아 학습이 불안정할 수 있음.
- **큰 배치 사이즈**: 더 정확한 그래디언트(gradient) 추정이 가능해 안정적인 학습이 이루어지지만, 학습 속도가 느려질 수 있음.

딥러닝 출력 시각화 코드

```
learning_rate = 0.1
training_epochs = 1000
optimizer = SGD(learning_rate=learning_rate)
model.compile(optimizer=optimizer, loss='mse', metrics=['accuracy'])
model.summary()
history = model.fit(X_train, Y_train, epochs=training_epochs, batch_size=13)

y_pred = model.predict(X_train)
y_pred=scaler.inverse_transform(y_pred) #원래의 크기로 복구
y_pred[y_pred <0] =0 #음수값이면 0
X_train = X_train.numpy()*52 #원래의 크기로 복구
Y_train = scaler.inverse_transform(Y_train)

plt.scatter(X_train, Y_train, color='blue', s=5,label='Actual Data')
plt.scatter(X_train, y_pred, color='red', s=5,label='Predicted Data')
plt.legend()
plt.show()
```

딥러닝 출력 시각화


Actual Data
Predicted Data
40000
30000
20000
10000
0
0
10
20
30
40
50