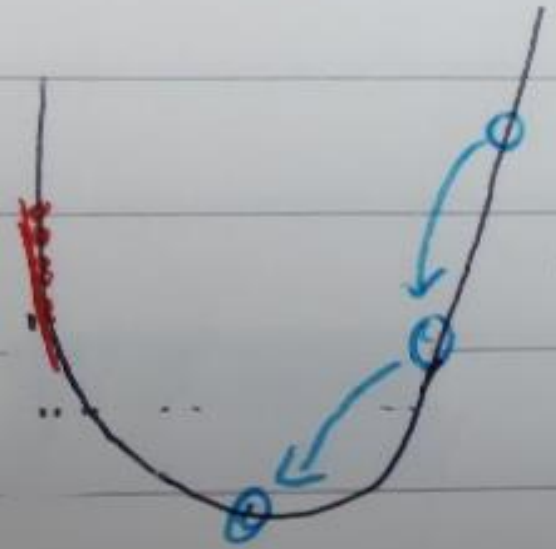
ML의 실용과 몇가지 팁

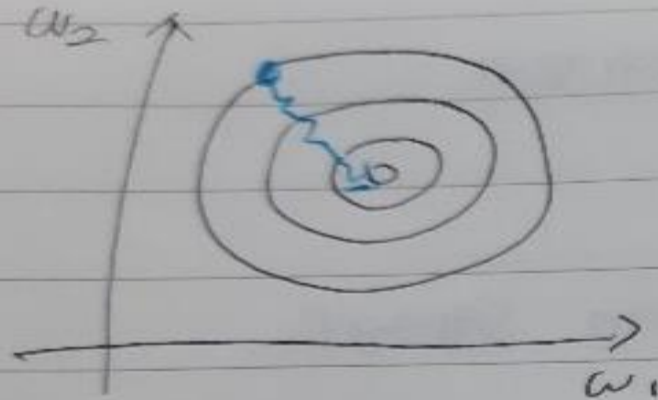


[1] running_rate

=> "α"

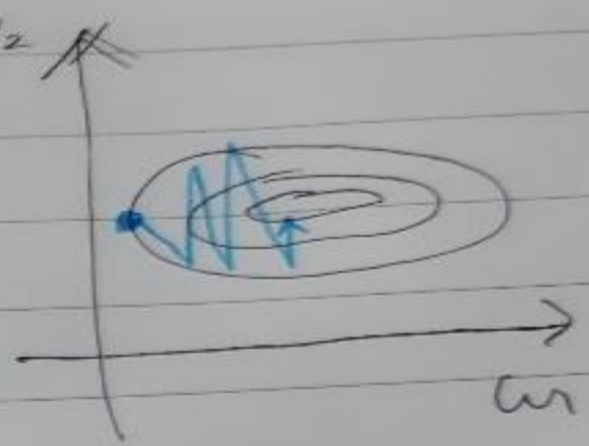
- 굉장히 큰 값을 줄경우. (overshotting)
  - => cost 값이 튕겨나감.
- 굉장히 작은 값 줄경우. (local minimum)
  - => 굉장히 오래 걸림.

[2] Preprocessing Data for gradient descent.



	X_1	X_2
1		9000
2		-5000

두 값 차이크면

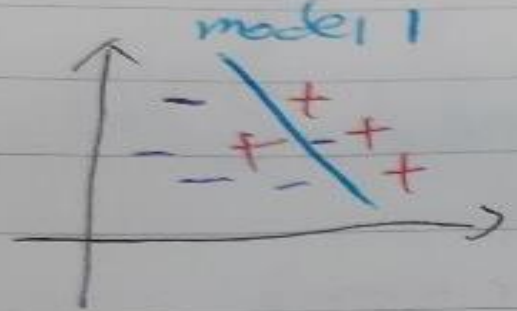


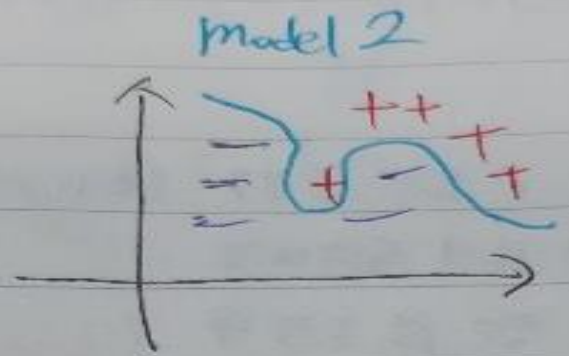
"Normalize" (1) (zero-Centered data, (2) normalized data)

Standardization (3): $X_j = \frac{X_j - \mu_j}{\sigma_j}$ "

성능 ⇑

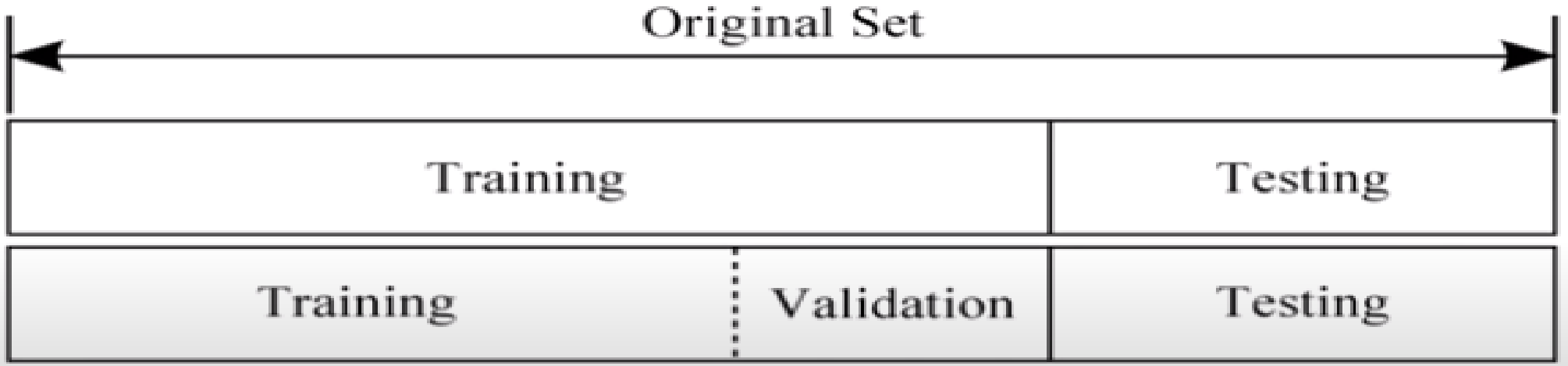
[3] Overfitting





⇒ 특정 data 에만 특화됨..

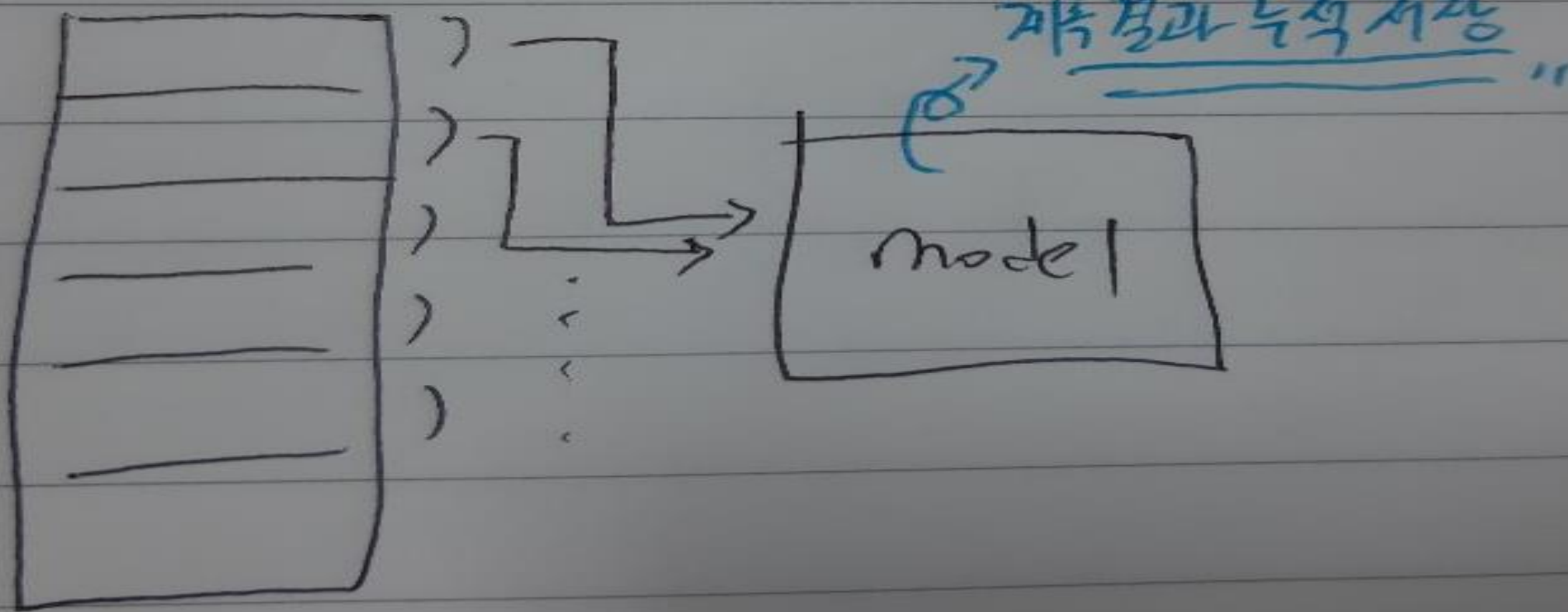
- More training data.
- Reduce the number of features.
- "Regularization"
  - weight 값들 너무 크지 않게 맞춰.
  - Cost + $\lambda \sum w^2$
    - ↳ regularization Strength.



시험 공부 모의 고사 단 한번의 기회

Online-learning.

Training Set이 100만개? 너무 많다.



Training/test dataset

```
# Training Data set
x_data = [[1, 2, 1],
          [1, 3, 2],
          [1, 3, 4],
          [1, 5, 5],
          [1, 7, 5],
          [1, 2, 5],
          [1, 6, 6],
          [1, 7, 7]]
y_data = [[0, 0, 1],
          [0, 0, 1],
          [0, 0, 1],
          [0, 1, 0],
          [0, 1, 0],
          [0, 1, 0],
          [1, 0, 0],
          [1, 0, 0]]

# Evaluation our model using this test dataset
x_test = [[2, 1, 1],
          [3, 1, 2],
          [3, 3, 4]]
y_test = [[0, 0, 1],
          [0, 0, 1],
          [0, 0, 1]]
```

```
# Launch graph
with tf.Session() as sess:
    # Initialize TensorFlow variables
    sess.run(tf.global_variables_initializer())

    for step in range(201):
        cost_val, W_val, _ = sess.run([cost, W, optimizer], feed_dict={X: x_data, Y: y_data})
        print(step, cost_val, W_val)

    # predict
    print("Prediction:", sess.run(prediction, feed_dict={X: x_test}))
    # Calculate the accuracy
    print("Accuracy: ", sess.run(accuracy, feed_dict={X: x_test, Y: y_test}))
```

Training data set과 Test data set을 분리하고
Training data를 학습시킨 뒤
마지막에 test data로 검증

Normalization

```python
def min_max_scaler(data):
    numerator = data - np.min(data, 0)
    denominator = np.max(data, 0) - np.min(data, 0)
    # noise term prevents the zero division
    return numerator / (denominator + 1e-7)
```

```python
xy = np.array(
    [
        [828.659973, 833.450012, 908100, 828.349976, 831.659973],
        [823.02002, 828.070007, 1828100, 821.655029, 828.070007],
        [819.929993, 824.400024, 1438100, 818.97998, 824.159973],
        [816, 820.958984, 1008100, 815.48999, 819.23999],
        [819.359985, 823, 1188100, 818.469971, 818.97998],
        [819, 823, 1198100, 816, 820.450012],
        [811.700012, 815.25, 1098100, 809.780029, 813.669983],
        [809.51001, 816.659973, 1398100, 804.539978, 809.559998],
    ]
)

# very important. It does not work without it.
xy = min_max_scaler(xy)
print(xy)

'''
[[0.99999999 0.99999999 0.         1.         1.        ]
 [0.70548491 0.70439552 1.         0.71881782 0.83755791]
 [0.54412549 0.50274824 0.57608696 0.606468   0.6606331 ]
 [0.33890353 0.31368023 0.10869565 0.45989134 0.43800918]
 [0.51436    0.42582389 0.30434783 0.58504805 0.42624401]
 [0.49556179 0.42582389 0.31521739 0.48131134 0.49276137]
 [0.11436064 0.         0.20652174 0.22007776 0.18597238]
 [0.         0.07747099 0.5326087  0.         0.        ]]
'''
```

MNIST Dataset Module not found error

- Anaconda에서 pip install tensorflow
- 다음 pip install –upgrade tensorflow==1.14

MNIST Dataset

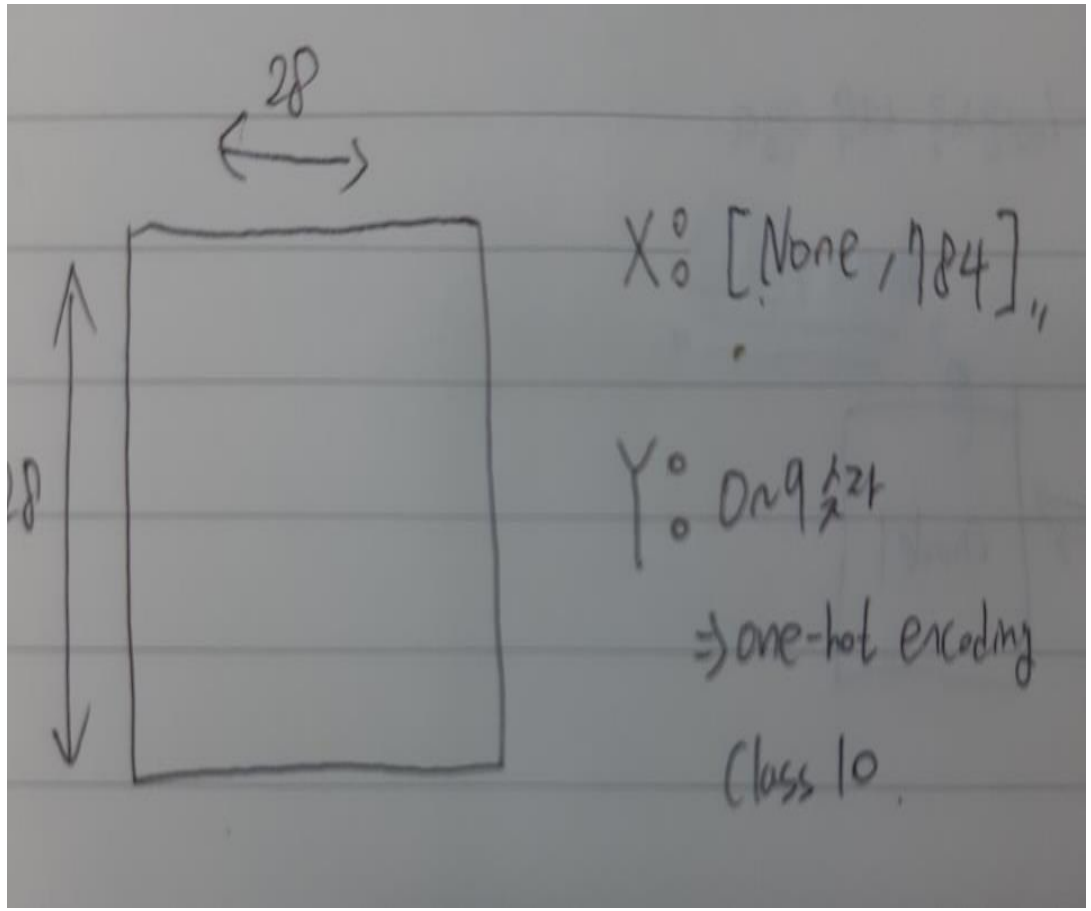
```
nb_classes = 10

# MNIST data image of shape 28 * 28 = 784
X = tf.placeholder(tf.float32, [None, 784])
# 0 - 9 digits recognition = 10 classes
Y = tf.placeholder(tf.float32, [None, nb_classes])
# X = 784, y = class
W = tf.Variable(tf.random_normal([784, nb_classes]))
b = tf.Variable(tf.random_normal([nb_classes]))

# Hypothesis (using softmax)
hypothesis = tf.nn.softmax(tf.matmul(X, W) + b)

cost = tf.reduce_mean(-tf.reduce_sum(Y * tf.log(hypothesis), axis=1))
train = tf.train.GradientDescentOptimizer(learning_rate=0.1).minimize(cost)

# Test model
is_correct = tf.equal(tf.argmax(hypothesis, 1), tf.argmax(Y, 1))
# Calculate accuracy
accuracy = tf.reduce_mean(tf.cast(is_correct, tf.float32))
```



Training epoch/batch

- One epoch = 전체 training data set을 학습 시킨 것
- 데이터가 굉장히 많은 경우 1 epoch를 batch size만큼 나눠 학습시킨다.
- 전체 데이터가 1000개이고, batch size=100이라면
  - 1epoch는 각 데이터의 size가 100인 batch가 들어간 10개의 iteration으로 나누어진다.
  - Epoch = 전체 데이터 셋에 대한 학습 횟수

```python
# parameters
num_epochs = 15
batch_size = 100
num_iterations = int(mnist.train.num_examples / batch_size)

with tf.Session() as sess:
    # Initialize TensorFlow variables
    sess.run(tf.global_variables_initializer())
    # Training cycle
    for epoch in range(num_epochs):
        avg_cost = 0

        for i in range(num_iterations):
            batch_xs, batch_ys = mnist.train.next_batch(batch_size)
            _, cost_val = sess.run([train, cost], feed_dict={X: batch_xs, Y: batch_ys})
            avg_cost += cost_val / num_iterations

        print("Epoch: {:04d}, Cost: {:.9f}".format(epoch + 1, avg_cost))

    print("Learning finished")
```

Epoch : 전체 데이터를 15번 학습하겠다.
Iteration : 전체 데이터 수를 batch_size로 나눔
batch_size(100)개씩 학습
=전체 데이터 중 100개씩 반복해서 학습

성능 평가

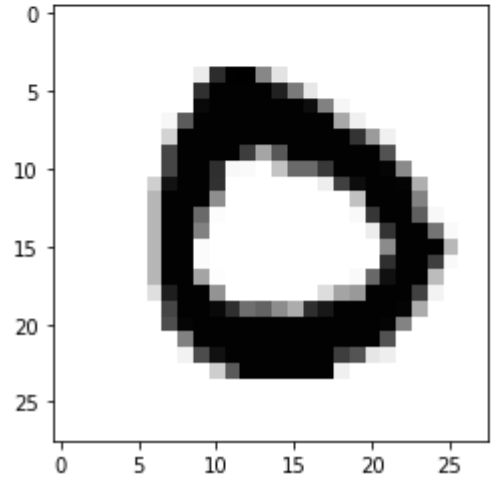
```
# Test the model using test sets
print(
    "Accuracy: ",
    accuracy.eval(
        session=sess, feed_dict={X: mnist.test.images, Y: mnist.test.labels}
    ),
)
```

성능 평가 시 사용하지 않은 데이터(test_data)를 사용함

```
# Get one and predict
r = random.randint(0, mnist.test.num_examples - 1)
print("Label: ", sess.run(tf.argmax(mnist.test.labels[r : r + 1], 1)))
print(
    "Prediction: ",
    sess.run(tf.argmax(hypothesis, 1), feed_dict={X: mnist.test.images[r : r + 1]}),
)

plt.imshow(
    mnist.test.images[r : r + 1].reshape(28, 28),
    cmap="Greys",
    interpolation="nearest",
)
plt.show()
```

```
Epoch: 0001, Cost: 2.834967095
Epoch: 0002, Cost: 1.121203029
Epoch: 0003, Cost: 0.896188526
Epoch: 0004, Cost: 0.783963787
Epoch: 0005, Cost: 0.712632324
Epoch: 0006, Cost: 0.661977553
Epoch: 0007, Cost: 0.623557729
Epoch: 0008, Cost: 0.592761933
Epoch: 0009, Cost: 0.567250710
Epoch: 0010, Cost: 0.546370934
Epoch: 0011, Cost: 0.527895901
Epoch: 0012, Cost: 0.512149333
Epoch: 0013, Cost: 0.498080472
Epoch: 0014, Cost: 0.486140881
Epoch: 0015, Cost: 0.475165493
Learning finished
Accuracy:  0.8885
Label:  [0]
Prediction:  [0]
```



Test에서 랜덤으로 정답에서 하나 가져오고 Prediction한 결과와 비교