01. 머신러닝 실습

- SVM를 이용하여 BMI를 구한다. 비만도를 판정하는 예.
- 키와 몸무게 데이터를 입력하면 '저체중', '정상', '비만'이라고 판단하는 프로그램

In [17]:
```
import pandas as pd
import numpy as np
import tensorflow as tf
# 키, 몸무게, 레이블이 적힌 CSV 파일 읽어 들이기 --- (※1)
csv = pd.read_csv("bmi.csv")

print(csv.shape)
print(csv.info() )
csv.head()
```

```
(20000, 3)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20000 entries, 0 to 19999
Data columns (total 3 columns):
height    20000 non-null int64
weight    20000 non-null int64
label     20000 non-null object
dtypes: int64(2), object(1)
memory usage: 468.8+ KB
None
```

Out[17]:

	height	weight	label
0	142	62	fat
1	142	73	fat
2	177	61	normal
3	187	48	thin
4	153	60	fat

In [18]:
```
# 데이터 정규화 --- (※2)
csv["height"] = csv["height"] / 200
csv["weight"] = csv["weight"] / 100
csv.head()
```

Out[18]:

	height	weight	label
0	0.710	0.62	fat
1	0.710	0.73	fat
2	0.885	0.61	normal
3	0.935	0.48	thin
4	0.765	0.60	fat

In [19]:
```
# 레이블을 배열로 변환하기 --- (※3)
# - thin=(1,0,0) / normal=(0,1,0) / fat=(0,0,1)
bclass = {"thin": [1,0,0], "normal": [0,1,0], "fat": [0,0,1]}
csv["label_pat"] = csv["label"].apply(lambda x : np.array(bclass[x]))
csv.head()
```

Out[19]:

	height	weight	label	label_pat
0	0.710	0.62	fat	[0, 0, 1]
1	0.710	0.73	fat	[0, 0, 1]
2	0.885	0.61	normal	[0, 1, 0]
3	0.935	0.48	thin	[1, 0, 0]
4	0.765	0.60	fat	[0, 0, 1]

In [20]:
```
# 테스트를 위한 데이터 분류 --- (※4)
test_csv = csv[15000:20000]
test_pat = test_csv[["weight","height"]]
test_ans = list(test_csv["label_pat"])
```

In [21]:
```
# 데이터 플로우 그래프 구출하기 --- (※5)
# 플레이스홀더 선언하기
x  = tf.placeholder(tf.float32, [None, 2]) # 키와 몸무게 데이터 넣기
y_ = tf.placeholder(tf.float32, [None, 3]) # 정답 레이블 넣기

# 변수 선언하기 --- (※6)
W = tf.Variable(tf.zeros([2, 3])); # 가중치
b = tf.Variable(tf.zeros([3])); # 바이어스

# 소프트맥스 회귀 정의하기 --- (※7)
y = tf.nn.softmax(tf.matmul(x, W) + b)

```

In [22]:
```

# 모델 훈련하기 --- (※8)
cross_entropy = -tf.reduce_sum(y_ * tf.log(y))
optimizer = tf.train.GradientDescentOptimizer(0.01)
train = optimizer.minimize(cross_entropy)

# 정답률 구하기
predict = tf.equal(tf.argmax(y, 1), tf.argmax(y_,1))
accuracy = tf.reduce_mean(tf.cast(predict, tf.float32))

```

In [23]:
```
# 세션 시작하기
sess = tf.Session()
sess.run(tf.global_variables_initializer()) # 변수 초기화하기
# 학습시키기
for step in range(3501):
    i = (step * 100) % 14000
    rows = csv[1 + i : 1 + i + 100]
    x_pat = rows[["weight","height"]]
    y_ans = list(rows["label_pat"])
    fd = {x: x_pat, y_: y_ans}
    sess.run(train, feed_dict=fd)
    if step % 500 == 0:
        cre = sess.run(cross_entropy, feed_dict=fd)
        acc = sess.run(accuracy, feed_dict={x: test_pat, y_: test_ans})
        print("step=", step, "cre=", cre, "acc=", acc)

```

```
step= 0 cre= 108.66269 acc= 0.3242
step= 500 cre= 57.58866 acc= 0.8904
step= 1000 cre= 45.02092 acc= 0.898
step= 1500 cre= 41.654335 acc= 0.9566
step= 2000 cre= 34.66403 acc= 0.943
step= 2500 cre= 34.287025 acc= 0.9674
step= 3000 cre= 26.880764 acc= 0.9726
step= 3500 cre= 29.590666 acc= 0.9728
```

In [24]:
```
# 최종적인 정답률 구하기
acc = sess.run(accuracy, feed_dict={x: test_pat, y_: test_ans})
print("정답률 =", acc)
```

```
정답률 = 0.9728
```

In [ ]:
```

```

In [ ]:
```

```

In [ ]:
```

```