Perceptrons - Training

Note for 717005@ Hallym University !

- Make a prediction with weights

In [1]:

```
def predict(X, w):
    bias = w[0]
    activation = bias + w[1]* X[0] + w[2]* X[1]
    if activation >= 0.0:
        return 1.0
    else:
        return 0.0
```

- Estimate Perceptron weights using stochastic gradient descent

In [2]:

```
def train_weights(train, l_rate, n_epoch):
    weights = [0.0 for i in range(len(train[0]))]
    for epoch in range(n_epoch):
        sum_error = 0.0
        for row in train:
            prediction = predict(row, weights)
            error = row[-1] - prediction
            sum_error += error**2
            weights[0] = weights[0] + l_rate * error
            for i in range(len(row)-1):
                weights[i + 1] = weights[i + 1] + l_rate * error * row[i]
        print('epoch={}, error={}'.format(epoch, sum_error))
    return weights
```

In [3]:

```
# test predictions
dataset = [[0, 0 ,0],
    [0, 1 ,0],
    [1, 0 ,0],
    [1, 1, 1]]
```

- Hyperparameters

In [4]:

```
l_rate = 0.1
n_epoch = 5
```

In [5]:

```
weights = train_weights(dataset, l_rate, n_epoch)
```

```
epoch=0, error=2.0
epoch=1, error=3.0
epoch=2, error=3.0
epoch=3, error=0.0
epoch=4, error=0.0
```

In [6]:

```
print(weights)
```

```
[-0.20000000000000004, 0.2, 0.1]
```

- Test !

Note : When we talk about supervised learning, we are usually talking about learning to predict a label associated with the data. The goal is for the model to generalize to new data.

In [7]:

```
predict([0, 0], weights)
```

Out[7]:

```
0.0
```

In [8]:

```
predict([1, 1], weights)
```

Out[8]:

```
1.0
```

In [9]:

```
predict([0.9, 0.5], weights)
```

Out[9]:

```
1.0
```