Apple and Oranges Dataset

In [1]:
```
dataset = [

    [0.88, 0.34, 0],
    [0.32, 0.95, 1],
    [0.95, 0.15, 0],
    [0.45, 0.75, 1],
    [0.55, 0.85, 1],
    [0.90, 0.30, 0],
    [0.75, 0.25, 0],
    [0.20, 0.80, 1],
    [0.35, 0.85, 1],
    [0.80, 0.22, 0],
    [0.65, 0.80, 0],
    [0.22, 0.90, 1],
    [0.14, 0.74, 1],
    [0.84, 0.22, 0],
    [0.47, 0.76, 0],
    [0.45, 0.79, 0],
    [0.18, 0.80, 1],
    [0.99, 0.10, 0],
    [0.30, 0.87, 1],
    [0.60, 0.89, 0]

]
```

Training and Testing Function for Perceptron Implementation

In [2]:
```
def predict(row, weights):

    activation = weights[0]

    for i in range(len(row) - 1):

        activation += weights[i + 1] * row[i]

    threshold = 0.0

    return 1.0 if activation >= threshold else 0.0
```

In [3]:
```
def test_data(weights):

    for row in dataset:

        prediction = predict(row, weights)

        if row[-1] == 0:
            exp = "apple"
        else:
            exp = "orange"

        if prediction == 0:
            pred = "apple"
        else:
            pred = "orange"

        print('expected = {0}'.format(exp).ljust(20) + 'predicted = {0}'.format(pred).center(20))
```

In [4]:
```
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)
            target = row[-1] - prediction
            weights[0] = weights[0] + l_rate * target

            for i in range(len(row) - 1):

                weights[i + 1] = weights[i + 1] + l_rate * target * row[i]

    return weights
```

Testing data with untrained weights

In [5]:
```
weights_before_training = [-0.25, 0.20653640140000007, 0.275]

test_data(weights_before_training)
```

```
expected = apple     predicted = orange
expected = orange    predicted = orange
expected = apple     predicted = apple
expected = orange    predicted = orange
expected = orange    predicted = orange
expected = apple     predicted = orange
expected = apple     predicted = apple
expected = orange    predicted = orange
expected = orange    predicted = orange
expected = apple     predicted = apple
expected = apple     predicted = orange
expected = orange    predicted = orange
expected = orange    predicted = apple
expected = apple     predicted = apple
expected = apple     predicted = orange
expected = apple     predicted = orange
expected = orange    predicted = orange
expected = apple     predicted = apple
expected = orange    predicted = orange
expected = apple     predicted = orange
```

Training of weights

In [6]:
```
l_rate = 0.25
n_epoch = 10
weights_after_training = train_weights(dataset, l_rate, n_epoch)
print("The updated bias and weights are = {0}".format(weights_after_training))
```

```
The updated bias and weights are = [0.0, -1.1324999999999998, 0.4200000000000001]
```

Testing of data with trained weights

In [7]:
```
test_data(weights_after_training)
```

```
expected = apple     predicted = apple
expected = orange    predicted = orange
expected = apple     predicted = apple
expected = orange    predicted = apple
expected = orange    predicted = apple
expected = apple     predicted = apple
expected = apple     predicted = apple
expected = orange    predicted = orange
expected = orange    predicted = apple
expected = apple     predicted = apple
expected = apple     predicted = apple
expected = orange    predicted = orange
expected = orange    predicted = orange
expected = apple     predicted = apple
expected = apple     predicted = apple
expected = apple     predicted = apple
expected = orange    predicted = orange
expected = apple     predicted = apple
expected = orange    predicted = orange
expected = apple     predicted = apple
```