1. 파이썬 코드

```
import tensorflow as tf
import numpy as np

# [털, 날개] [0, 0] = 기타 / [1, 0] = 포유류 / [_, 1] = 조류
x_data = np.array([[0, 0], [1, 0], [1, 1], [0, 0], [0, 0], [0, 1], # 기타, 포유류, 조류, 기타, 기타, 조류
                   [1, 0], [0, 1], [0, 1], [1, 0], [1, 1], [0, 1], # 포유류, 조류, 조류, 포유류, 조류, 조류
                   [1, 0], [1, 1], [0, 1], [1, 1], [0, 0], [0, 1], # 포유류, 조류, 조류, 조류, 기타, 조류
                   [0, 0], [1, 0], [0, 0], [0, 1], [1, 0], [1, 1]]) # 기타, 포유류, 기타, 조류, 포유류, 조류

# 기타 [1, 0, 0], 포유류 [0, 1, 0], 조류 [0, 0, 1]
y_data = np.array([[1, 0, 0], [0, 1, 0], [0, 0, 1], [1, 0, 0], [1, 0, 0], [0, 0, 1], # 기타, 포유류, 조류, 기타, 기타, 조류
                   [0, 1, 0], [0, 0, 1], [0, 0, 1], [0, 1, 0], [0, 0, 1], [0, 0, 1], # 포유류, 조류, 조류, 포유류, 조류, 조류
                   [0, 1, 0], [0, 0, 1], [0, 0, 1], [0, 0, 1], [1, 0, 0], [0, 0, 1], # 포유류, 조류, 조류, 조류, 기타, 조류
                   [1, 0, 0], [0, 1, 0], [1, 0, 0], [0, 0, 1], [0, 1, 0], [0, 0, 1]]) # 기타, 포유류, 기타, 조류, 포유류, 조류

X = tf.placeholder(tf.float32)
Y = tf.placeholder(tf.float32)

# 가중치 편향 추가
W1 = tf.Variable(tf.random_uniform([2, 10], -1., 1.))
W2 = tf.Variable(tf.random_uniform([10, 20], -1., 1.))
W3 = tf.Variable(tf.random_uniform([20, 3], -1., 1.))

b1 = tf.Variable(tf.zeros([10]))
b2 = tf.Variable(tf.zeros([20]))
b3 = tf.Variable(tf.zeros([3]))

# 특징 입력값에 첫 번째 가중치와 편향, 활성화 함수 적용
L1 = tf.add(tf.matmul(X, W1), b1)
L1 = tf.nn.relu(L1)

# 두 번째 가중치와 편향, 활성화 함수 적용
L2 = tf.add(tf.matmul(L1, W2), b2)
L2 = tf.nn.relu(L2)

# 세 번째 가중치와 편향을 적용하여 최종 모델 생성
model = tf.add(tf.matmul(L2, W3), b3)
```

```
# 손실함수 : cross entropy / 최적화 함수 : AdamOptimizer >> Gradient Descent Optiomization 중 하나
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits_v2(labels=Y, logits=model))

optimizer = tf.train.AdamOptimizer(learning_rate = 0.01)
train_op = optimizer.minimize(cost)

# 텐서플로의 세션을 초기화합니다.
init = tf.global_variables_initializer()
sess = tf.Session()
sess.run(init)

# 앞서 구성한 특징과 레이블 데이터를 이용해 학습을 100번 진행합니다.
for step in range(100):
  sess.run(train_op, feed_dict={X: x_data, Y: y_data})

  # 학습 도중 10번에 한 번씩 손실값을 출력해 봅니다.
  if (step + 1 ) % 10 == 0:
    print(step + 1, sess.run(cost, feed_dict={X: x_data, Y: y_data}))

# 학습 결과 확인
prediction = tf.argmax(model, axis=1)
target = tf.argmax(Y, axis=1)
print('예측값:', sess.run(prediction, feed_dict={X: x_data}))
print('실제값:', sess.run(target, feed_dict={Y: y_data}))

# 정확도 출력
is_correct = tf.equal(prediction, target)
accuracy = tf.reduce_mean(tf.cast(is_correct, tf.float32))
print('정확도: %.2f' % sess.run(accuracy * 100, feed_dict={X: x_data, Y: y_data}))
```

2. 입력 데이터

- x_data : [털, 날개] data set ([0, 0] = 기타 / [1, 0] = 포유류 / [0 or 1, 1] = 조류)

```
x_data = np.array([[0, 0], [1, 0], [1, 1], [0, 0], [0, 0], [0, 1],
                   [1, 0], [0, 1], [0, 1], [1, 0], [1, 1], [0, 1],
                   [1, 0], [1, 1], [0, 1], [1, 1], [0, 0], [0, 1],
                   [0, 0], [1, 0], [0, 0], [0, 1], [1, 0], [1, 1]])
```

x_data = [기타, 포유류, 조류, 기타, 기타, 조류,
포유류, 조류, 조류, 포유류, 조류, 조류,
포유류, 조류, 조류, 조류, 기타, 조류,
기타, 포유류, 기타, 조류, 포유류, 조류]

총 24 개의 데이터

- y_data : x_data 에 따른 data set ([1, 0, 0] = 기타 / [0, 1, 0] = 포유류 / [0, 0, 1] = 조류)

```
y_data = np.array([[1, 0, 0], [0, 1, 0], [0, 0, 1], [1, 0, 0], [1, 0, 0], [0, 0, 1],
                   [0, 1, 0], [0, 0, 1], [0, 0, 1], [0, 1, 0], [0, 0, 1], [0, 0, 1],
                   [0, 1, 0], [0, 0, 1], [0, 0, 1], [0, 0, 1], [1, 0, 0], [0, 0, 1],
                   [1, 0, 0], [0, 1, 0], [1, 0, 0], [0, 0, 1], [0, 1, 0], [0, 0, 1]])
```

3. 심층 심경망 구현

- 3 개의 은닉층, 각 은닉층의 노드 수는 10 개, 20 개, 10 개
  W1, b1 은 첫번째 은닉층의 가중치 및 편향
  W2, b2 은 첫번째 은닉층의 가중치 및 편향
  W3, b3 은 첫번째 은닉층의 가중치 및 편향

```
# 가중치 편향 추가
W1 = tf.Variable(tf.random_uniform([2, 10], -1., 1.))
W2 = tf.Variable(tf.random_uniform([10, 20], -1., 1.))
W3 = tf.Variable(tf.random_uniform([20, 3], -1., 1.))

b1 = tf.Variable(tf.zeros([10]))
b2 = tf.Variable(tf.zeros([20]))
b3 = tf.Variable(tf.zeros([3]))
```

- 가중치와 편향 생성 후 순차적으로 적용하여 최종 모델 생성

```
# 특징 입력값에 첫 번째 가중치와 편향, 활성화 함수 적용
L1 = tf.add(tf.matmul(X, W1), b1)
L1 = tf.nn.relu(L1)

# 두 번째 가중치와 편향, 활성화 함수 적용
L2 = tf.add(tf.matmul(L1, W2), b2)
L2 = tf.nn.relu(L2)

# 세 번째 가중치와 편향을 적용하여 최종 모델 생성
model = tf.add(tf.matmul(L2, W3), b3)
```

4. 결과

- 학습은 총 100 회 진행하였다.

- 손실값의 경우 학습을 여러 번 반복할수록 값이 줄어들었다.

학습횟수	손실값
10	0.250478
20	0.09889542
30	0.034931038
40	0.0132894935
50	0.0062113595
60	0.0036644957
70	0.0025712438
80	0.0019915586
90	0.0016332762
100	0.0013827725

- 예측값과 실제값

```
예측값: [0 1 2 0 0 2 1 2 2 1 2 2 1 2 2 2 0 2 0 1 0 2 1 2]
실제값: [0 1 2 0 0 2 1 2 2 1 2 2 1 2 2 2 0 2 0 1 0 2 1 2]
정확도: 100.00
```

0 은 기타, 1 은 포유류, 2 는 조류를 의미한다.

[기타, 포유류, 조류, 기타, 기타, 조류, 포유류, 조류, 조류, 포유류, 조류, 조류, 포유류, 조류, 조류, 조류, 기타, 조류, 기타, 포유류, 기타, 조류, 포유류, 조류]

총 24 개의 데이터를 입력하였고, 이를 숫자로 변환한 값이 실제값 data 이다.

학습을 충분히 진행하여 실제값과 예측값이 완전히 동일하여 정확도 100 의 결과를 얻었다.

다만, 학습횟수를 30 회정도로 줄여도 충분히 안정적으로 정확도 100 의 예측 결과를 얻을 수 있었다.