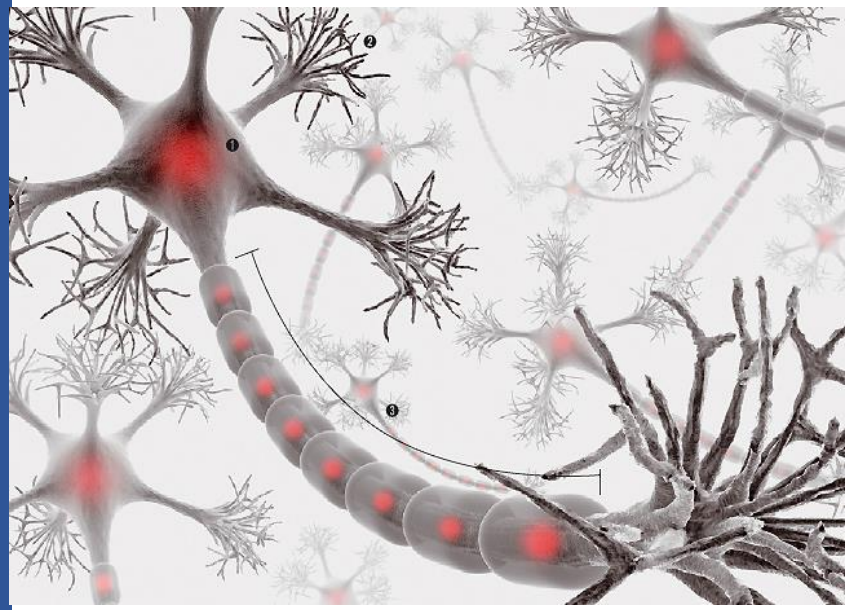
(DNN) Spiral Data Classification

학습 목표

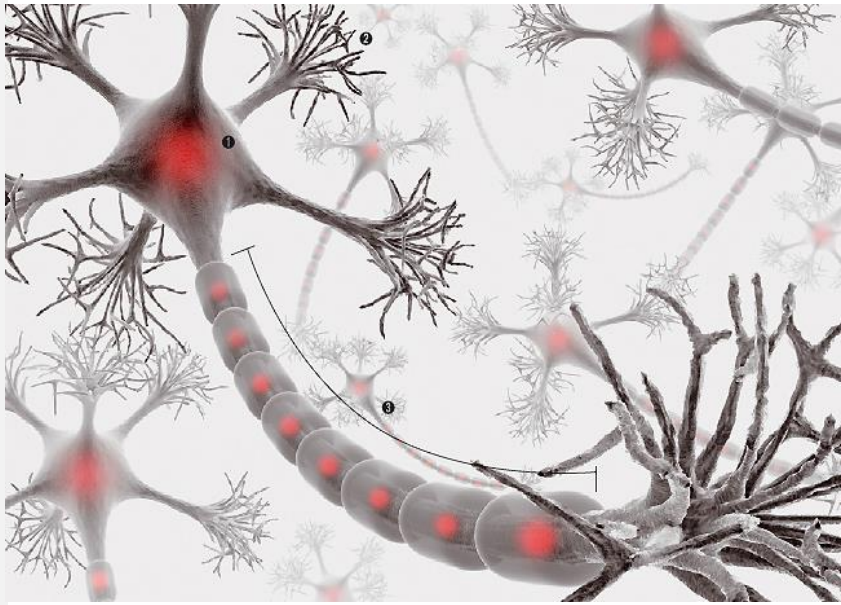
- Spiral 데이터를 분류하는 신경망 모델을 만들어 본다.

주요 내용

- 1. 문제 정의
- 2. 데이터 준비
- 3. 모델 정의 및 훈련, 검증



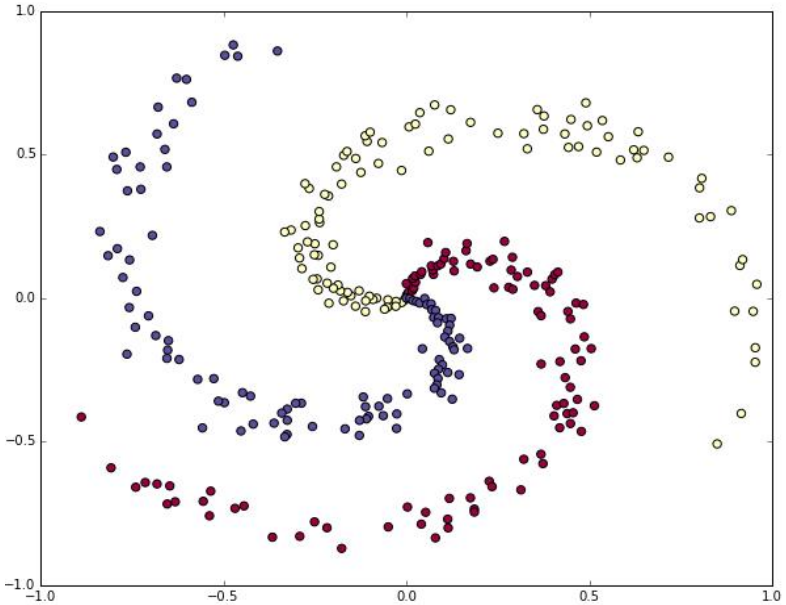
1 문제 정의



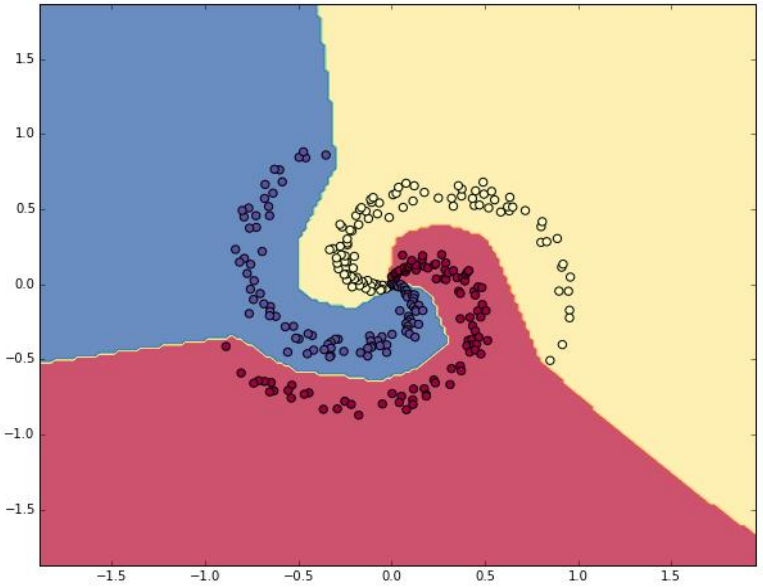
Classification 문제

2차원 나선형 데이터를 분류하는 신경망을 개발해보자.

Data Generation

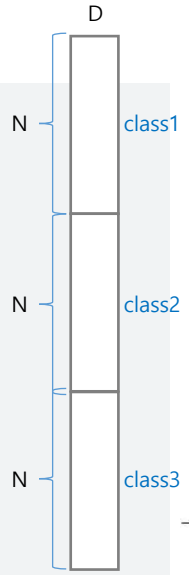


Classification



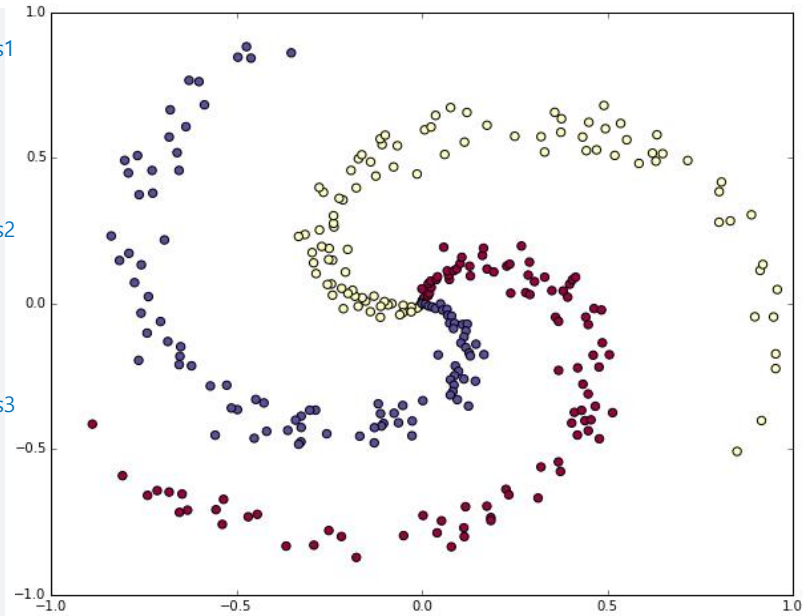
Hint : Spiral Dataset Generation

행렬 X의 모양




```
N = 100 # 클래스 별 포인트 개수
D = 2   # 차원
K = 3   # 클래스 개수
X = np.zeros((N*K,D)) # 데이터
y = np.zeros(N*K, dtype='uint8') # 레이블 (클래스)

for j in xrange(K):
  ix = range(N*j,N*(j+1)) # j번째 클래스
  r = np.linspace(0.0,1,N) # 반지름 [0,1]
  t = np.linspace(j*4,(j+1)*4,N) + np.random.randn(N)*0.2 # 각도 [0, 4]
  X[ix] = np.c_[r*np.sin(t), r*np.cos(t)]
  y[ix] = j

# lets visualize the data:
plt.scatter(X[:, 0], X[:, 1], c=y, s=40, cmap=plt.cm.Spectral)
plt.show()
```

color　marker size　color map

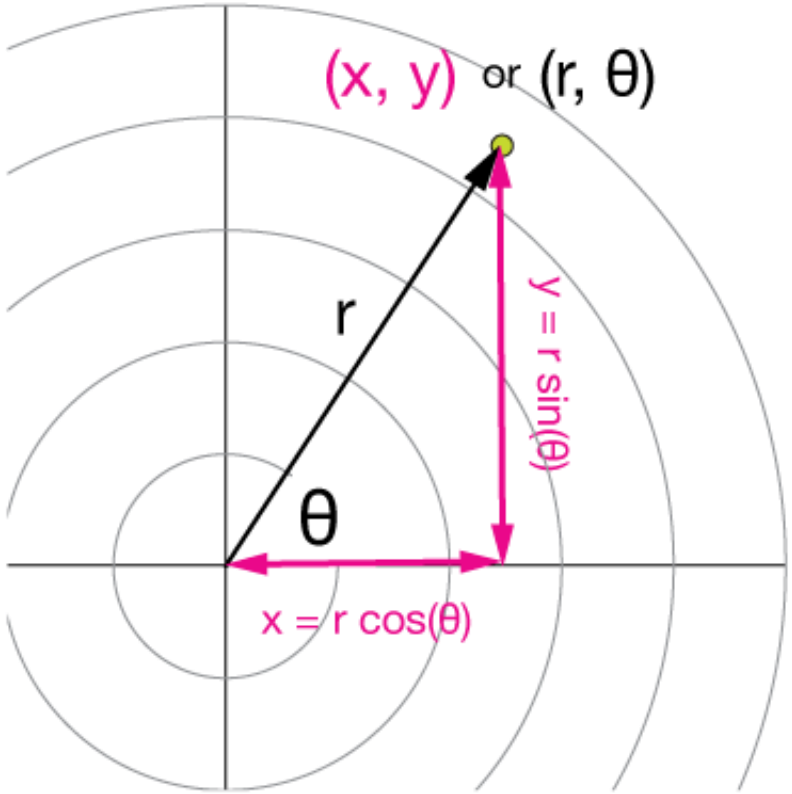


X (300,2) 입력 데이터　　**Y** (300) 레이블

- 반지름 r은 [0,1]에서 생성
- 각도 t는 [0, 4]에서 생성 + 노이즈 추가
- 각도가 4이어야 등간격으로 예쁘게 나옴
-
- 좌표가 r*sin(t), r*cos(t)라서 시계 방향으로 회전하는 나선형이 됨
- np.c_는 두 배열을 column으로 방향으로 합침

https://cs231n.github.io/neural-networks-case-study/

Hint : Spiral Dataset Generation



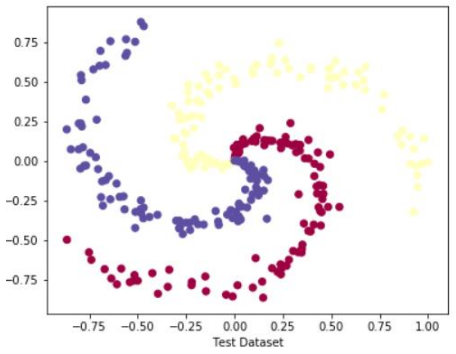
데이터 생성

- 반지름 r은 [0,1]에서 생성
- 각도 t는 [0, 4]에서 생성 + 노이즈 추가
- 각도가 4이어야 등간격으로 예쁘게 나옴

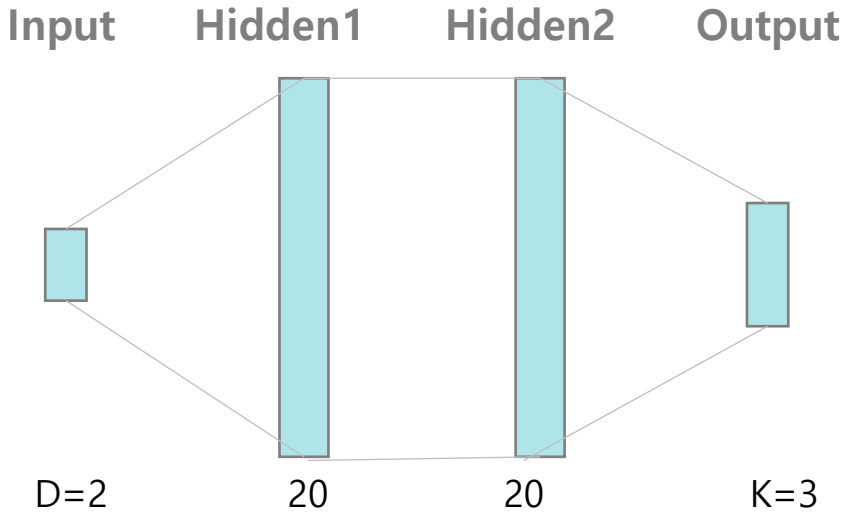
Hint : Network 구성

$x = (x_1, x_2)$

- 2차원 입력 벡터



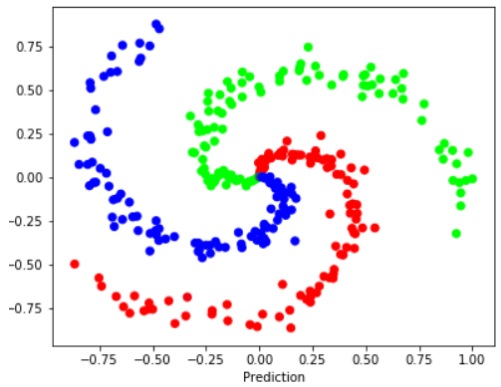
Fully Connected Network



- 2-hidden layers
- 20 neurons
- Relu

$y = (y_1, y_2, \ldots, y_K)$

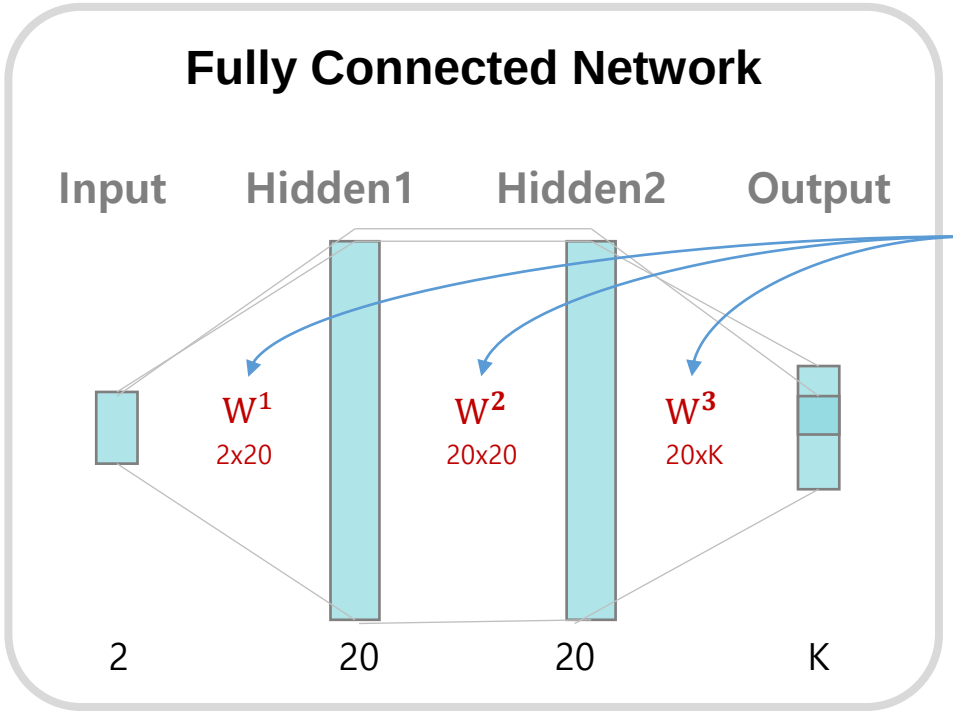
K개 Class에 속할 확률



Hint : Network 구성

$x = (x_1, x_2)$

- 2차원 입력 벡터

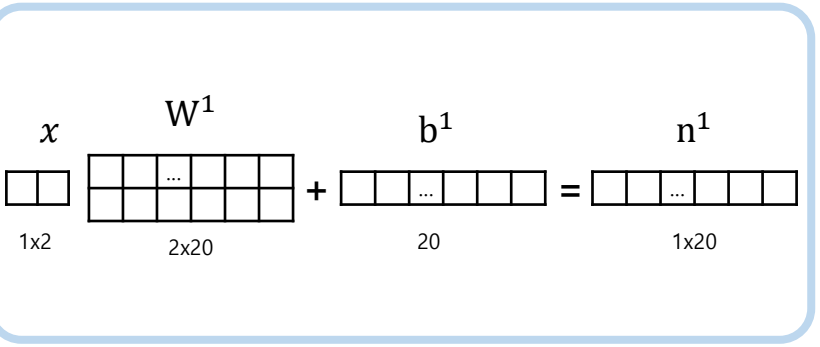


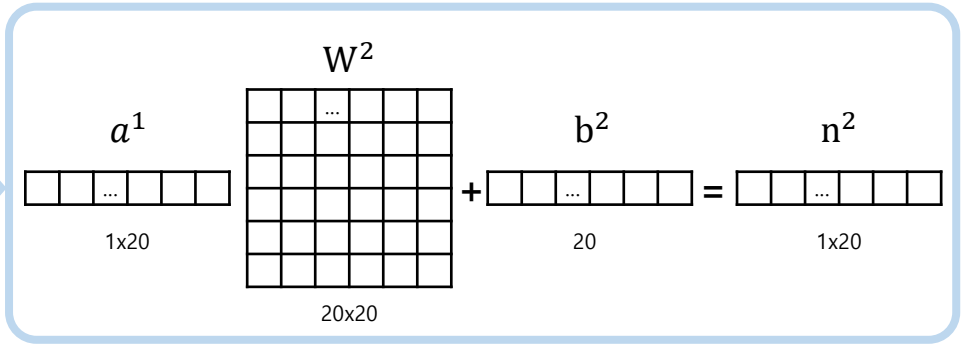
가중치 행렬의 크기는
(이전 Layer 뉴런 개수 x 현재 Layer 뉴런 개수)

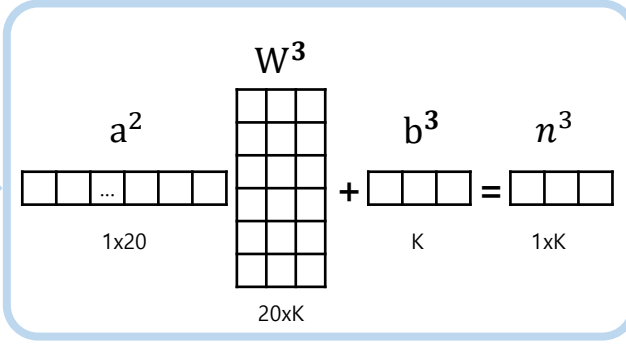
$y = (y_1, y_2, \ldots, y_K)$

K개 Class에 속할 확률

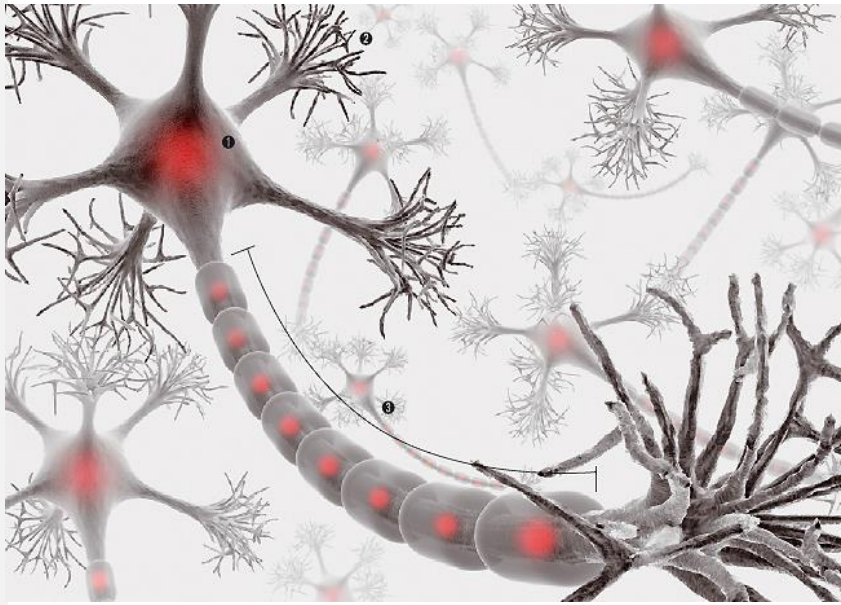
입력 샘플이 1개 때







2 데이터 준비



패키지 임포트

```
# tensorflow를 임포트합니다
import tensorflow as tf

# 헬퍼(helper) 라이브러리를 임포트합니다
import numpy as np
import matplotlib.pyplot as plt

print(tf.__version__)
```

```
2.0.0-dev20190524
```

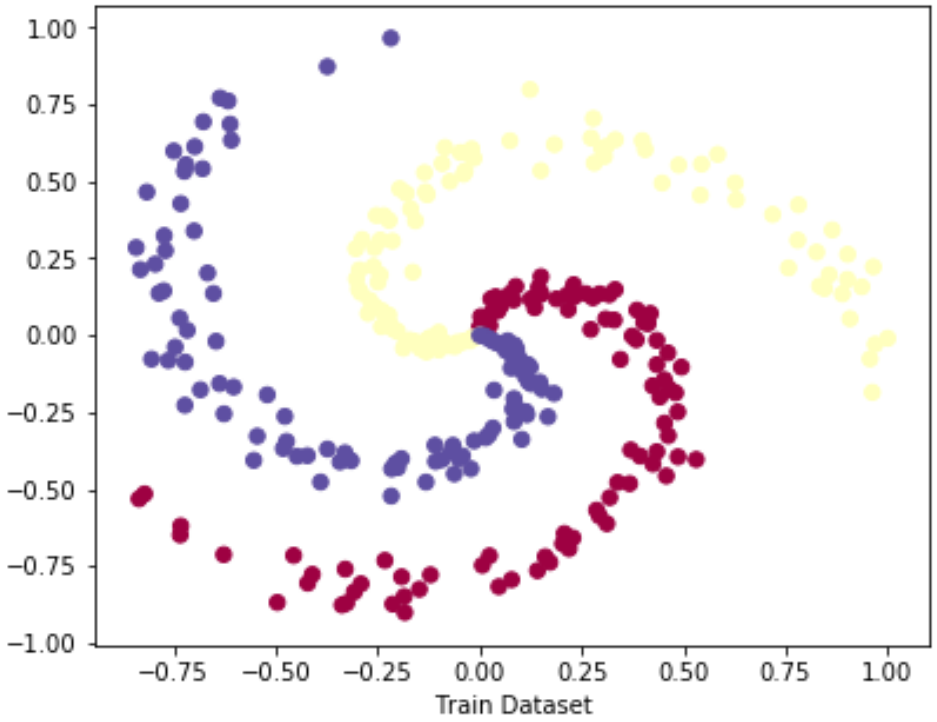
데이터셋 생성

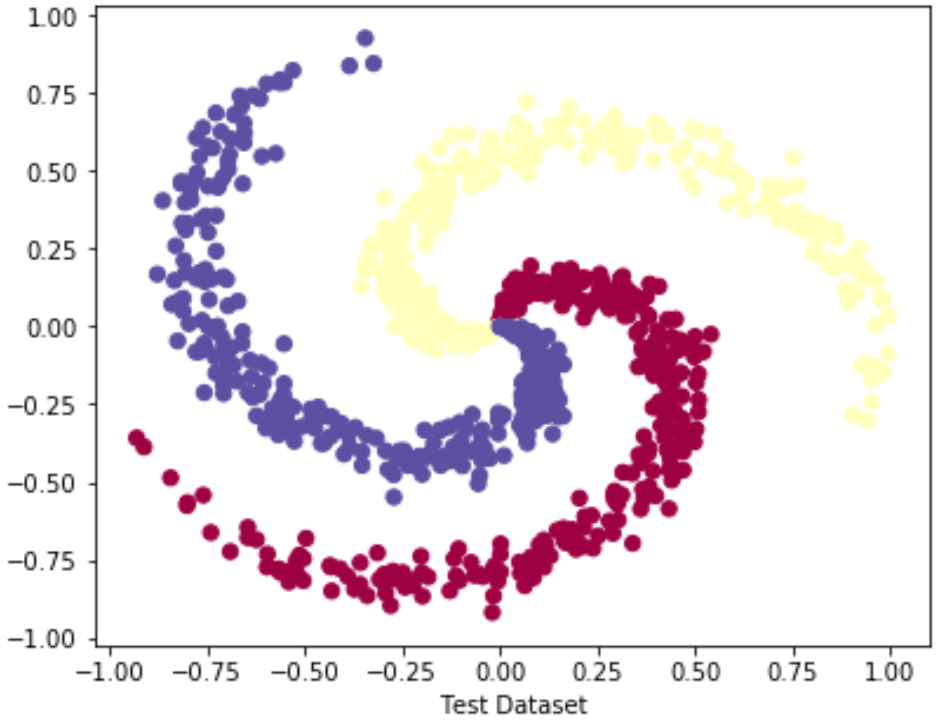
```
# generate the data
theta = 4 # class 별 시작 각도 : theta 배수

def generate_spiral_dataset(num_data, num_class, num_dim=2):
    input_data = np.zeros((num_data*num_class,num_dim)) # data matrix (each row = single example)
    output_data = np.zeros(num_data*num_class, dtype='uint8') # class labels

    for j in range(num_class):
      ix = range(num_data*j,num_data*(j+1)) # jth class data index

      r = np.linspace(0.0,1,num_data) # radius [0,1]
      t = np.linspace(j*theta,(j+1)*theta,num_data) + np.random.randn(num_data)*0.2  # theta [0, 4]

      input_data[ix] = np.c_[r*np.sin(t), r*np.cos(t)]        # inputs   (num_data*num_class, num_dim)
      output_data[ix] = j                                     # output  (num_data*num_class)

    return input_data, output_data
```

데이터셋 생성

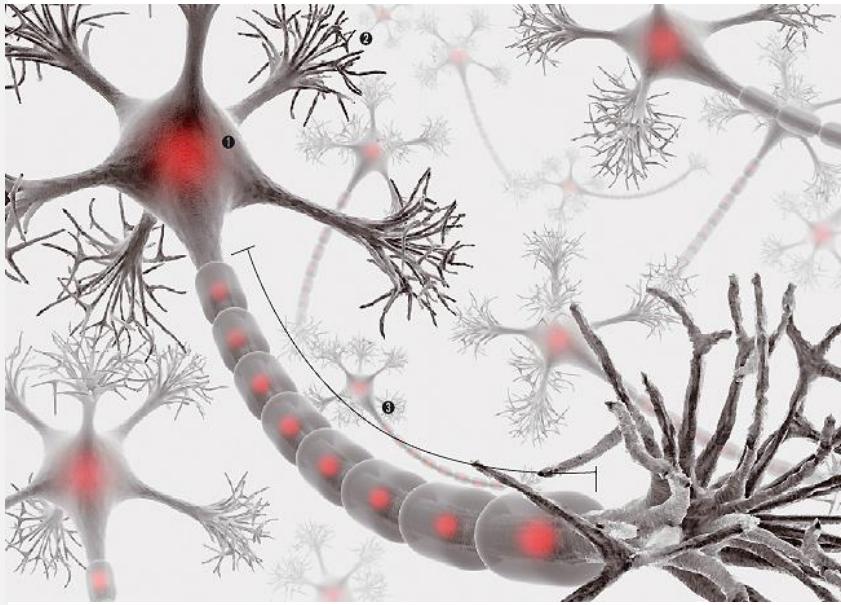
```
N = 200 # number of points per class
D = 2     # dimensionality
K = 3     # number of classes

train_x, train_y = generate_spiral_dataset(N, K, D)
test_x, test_y = generate_spiral_dataset(N, K, D)

# lets visualize the data:
plt.figure(figsize=(14,5))
plt.subplot(1,2,1)
plt.scatter(train_x[:, 0], train_x[:, 1], c=train_y, s=40, cmap=plt.cm.Spectral)
plt.xlabel("Train Dataset")

plt.subplot(1,2,2)
plt.scatter(test_x[:, 0], test_x[:, 1], c=test_y, s=40, cmap=plt.cm.Spectral)
plt.xlabel("Test Dataset")
plt.show()
```

데이터셋 생성





3 모델 정의 및 훈련, 검증



모델 정의 (문제)

```
class Model(tf.Module):
    def __init__(self):
        # create variables
        initializer = tf.initializers.GlorotUniform()
        W0 = # your code
        W1 = # your code
        W2 = # your code

        b0 = # your code # bias는 0으로 초기화, tf.zeros로 초기화
        b1 = # your code
        b2 = # your code

        self.weights = [W0, W1, W2]
        self.biases = [b0, b1, b2]
        self.activations = [tf.nn.relu, tf.nn.relu, None]

    def __call__(self, input):
        x = input
        for W, b, activation in zip(self.weights, self.biases, self.activations):
            # affine transformation
            x = tf.matmul(x, W) + b
            # activation
            if activation is not None:
                x = activation(x)
        return x
```

모델 훈련 (문제)

```
model = Model()
optimizer = tf.optimizers.Adam() # create optimizer

# run training
batch_size = 32
for training_step in range(10000):
    # get a random subset of the training data
    indices = np.random.randint(low=0, high=len(train_x), size=batch_size)
    input_batch = tf.Variable(train_x[indices], dtype=tf.float32, name='input')
    output_batch = tf.Variable(train_y[indices], dtype=tf.uint8, name='output')
    output_batch = tf.one_hot(output_batch, K) # one-hot encoding

    with tf.GradientTape() as tape:
        output_pred = model(input_batch)
        #  tf.nn.softmax_cross_entropy_with_logits와  tf.reduce_mean을 사용해서 Loss를 계산하시오
        loss = # your code

        grads  = tape.gradient(loss, model.trainable_variables)
        optimizer.apply_gradients(zip(grads, model.trainable_variables))

    if training_step % 1000 == 0:
        print('{0:04d} loss: {1:.3f}'.format(training_step, loss.numpy()))
```

One-Hot Encoding이란

One-hot encoding 이란?

클래스를 나타내는 숫자

cat → 1

mat → 2

on → 3

set → 4

the → 5

One-hot encoding

cat	1	0	0	0	0
mat	0	1	0	0	0
on	0	0	1	0	0
set	0	0	0	1	0
the	0	0	0	0	1

차원 : 클래스 수

- 클래스를 나타나내는 원소는 1로 표기하고 나머지는 0으로 표기하는 벡터 표기법
- 분류 문제에서 모델이 Categorical Distribution를 예측할 때, 타깃은 One-Hot Vector로 만들어서 Cross Entropy Loss를 계산한다.

Cross Entropy Loss 계산 시 One-Hot Encoding하는 이유

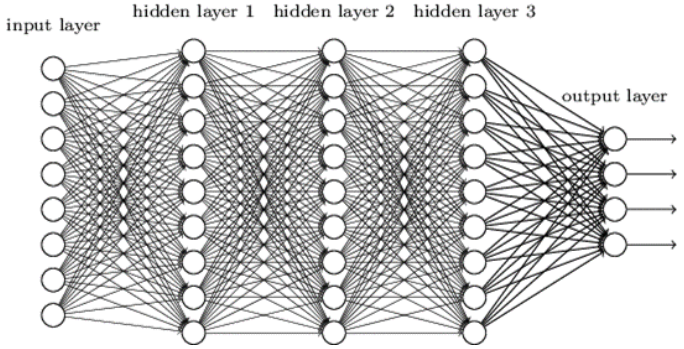
훈련 집합 (Train Set)

Input

Target

2

Network



Output

3.7	다람쥐
12	고양이
4.5	강아지
...	...

scores/logits

Categorical Distribution

0.1	다람쥐
0.7	고양이
0.2	강아지
...	...

probability

$\mu = (\mu_1, \mu_2, \ldots, \mu_K)^T$

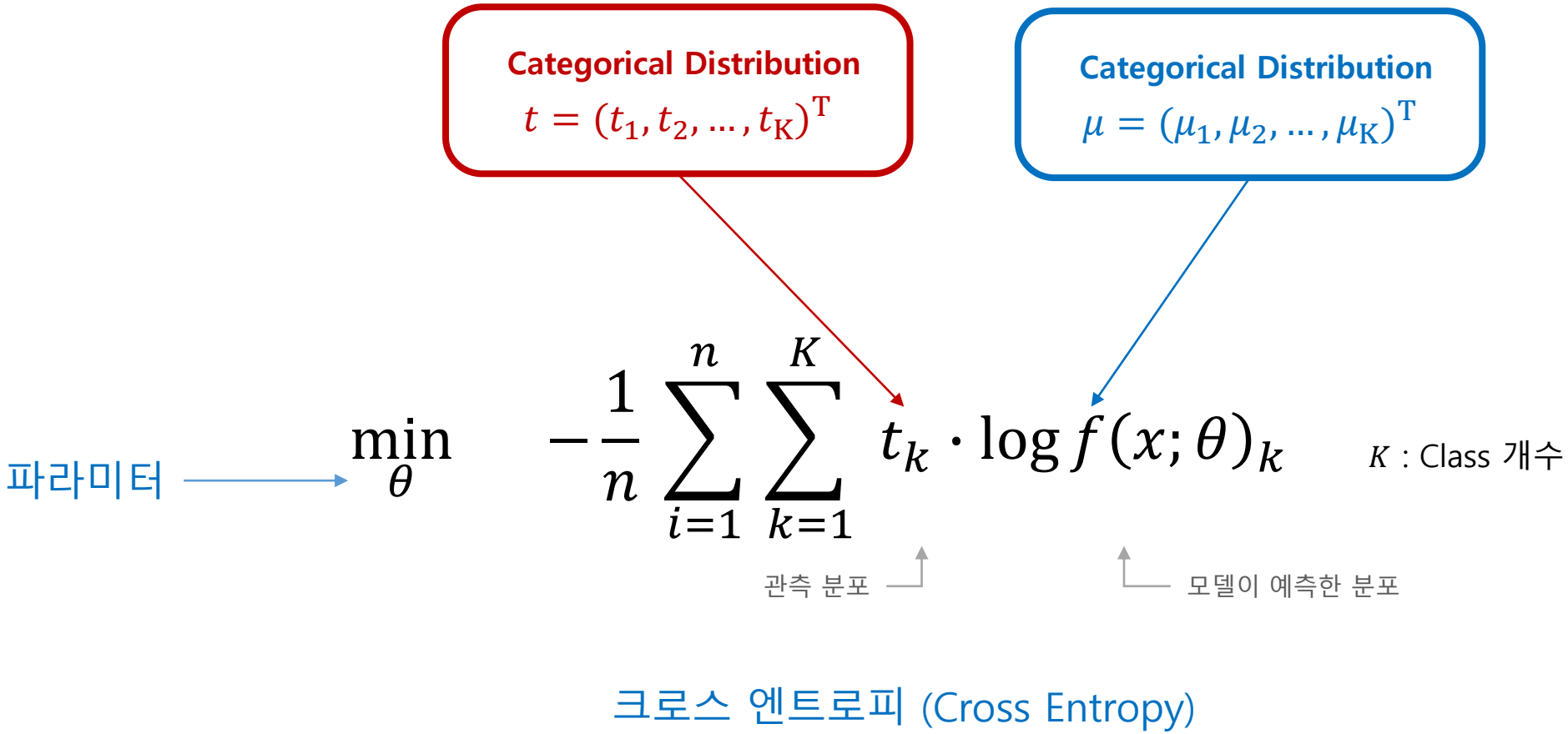
One-hot encoding

Categorical Distribution

$t = (t_1, t_2, \ldots, t_K)^T$

Cross Entropy Loss 계산 시 One-Hot Encoding하는 이유

예측 분포가 관측 분포와 같아지도록 두 분포의 차이를 최소화 하는 Loss가 Cross Entropy Loss이다.



크로스 엔트로피 (Cross Entropy)

참고 tf.nn.softmax_cross_entropy_with_logits

```
tf.nn.softmax_cross_entropy_with_logits(
    labels, logits, axis=-1, name=None
)
```

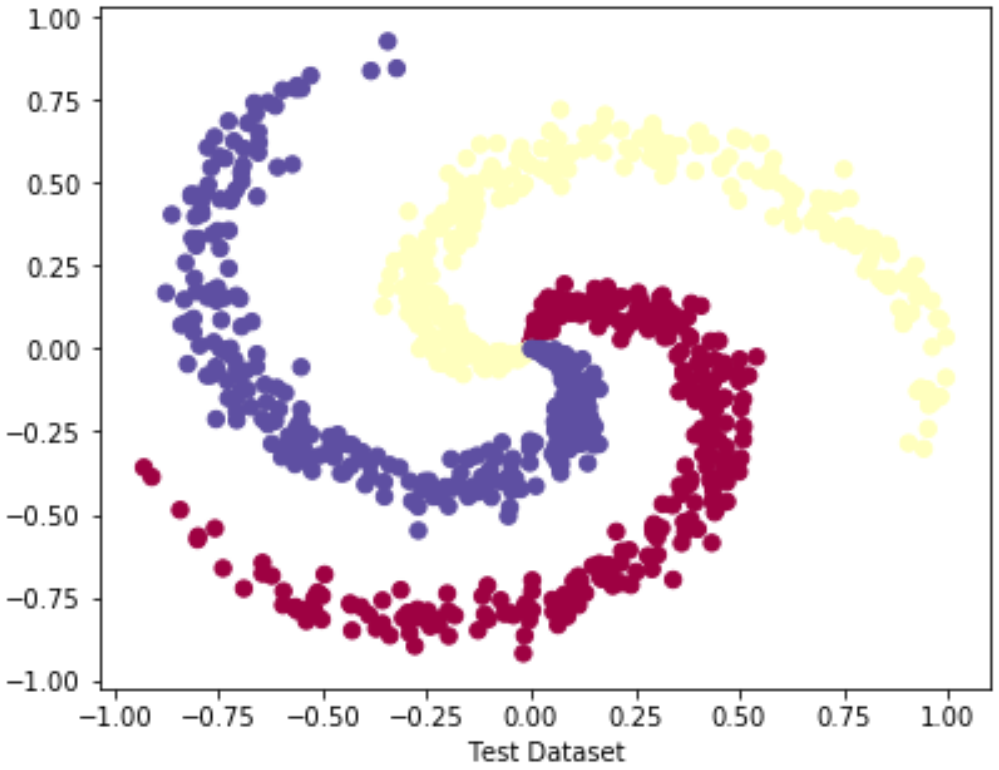
- **Labels :** 레이블, Each vector along the class dimension should hold a valid probability distribution e.g. for the case in which labels are of shape [batch_size, num_classes], each row of labels[i] must be a valid probability distribution.
- **Logits :** Softmax 적용 전 상태, Per-label activations, typically a linear output. These activation energies are interpreted as unnormalized log probabilities.
- **Axis :** The class dimension. Defaulted to -1 which is the last dimension.
- **Name :** A name for the operation (optional).

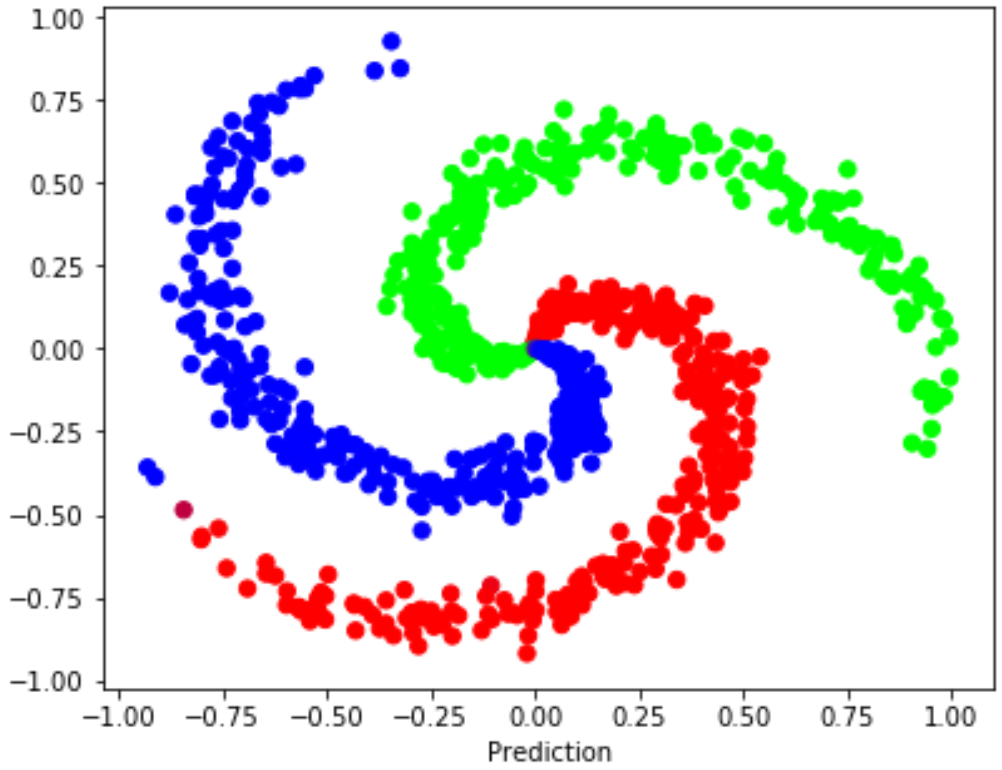
https://www.tensorflow.org/api_docs/python/tf/nn/softmax_cross_entropy_with_logits

테스트

```
test_input = tf.Variable(test_x, dtype=tf.float32, name='input')
test_output = tf.Variable(test_y, dtype=tf.float32, name='Output')

test_output_pred = model(test_input)
test_output_pred = tf.nn.softmax(test_output_pred, axis=-1)

plt.figure(figsize=(14,5))
plt.subplot(1,2,1)
plt.scatter(test_x[:, 0], test_x[:, 1], c=test_y, s=40, cmap=plt.cm.Spectral)
plt.xlabel("Test Dataset")

plt.subplot(1,2,2)
plt.scatter(test_x[:, 0], test_x[:, 1], c=test_output_pred.numpy(), s=40, cmap=plt.cm.Spectral)
plt.xlabel("Prediction")
plt.show()
```

테스트 (정답)





Thank you!