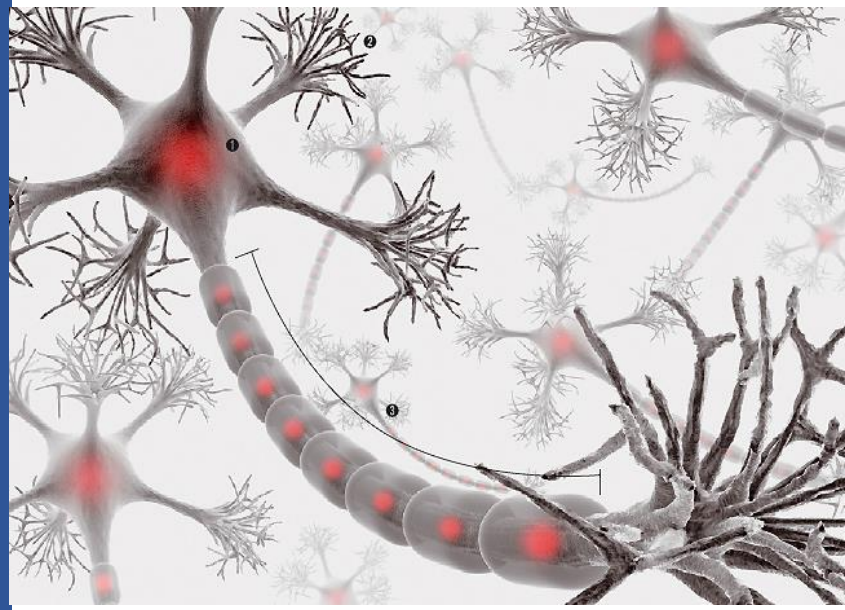
(DNN) Spiral Data Classification

학습 목표

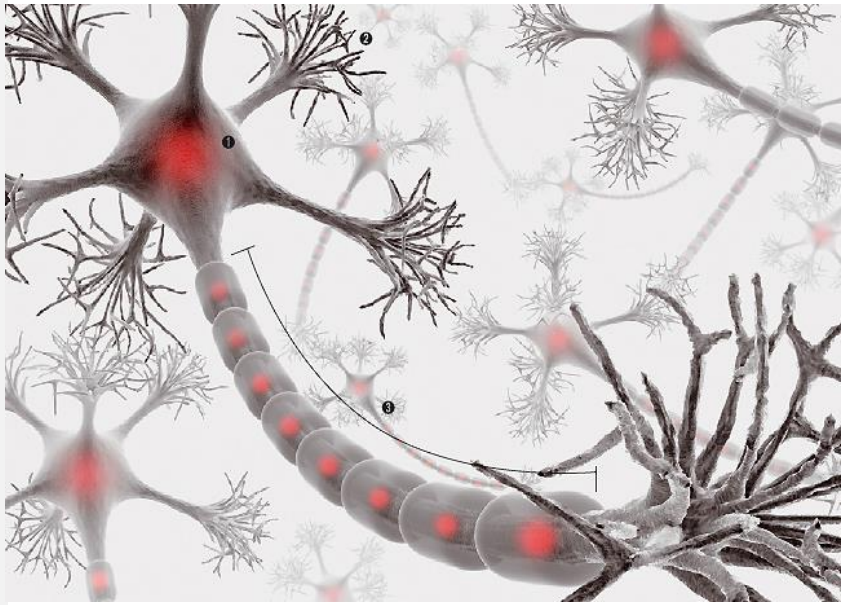
- Spiral 데이터를 분류하는 신경망 모델을 Keras로 만들어 본다.

주요 내용

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



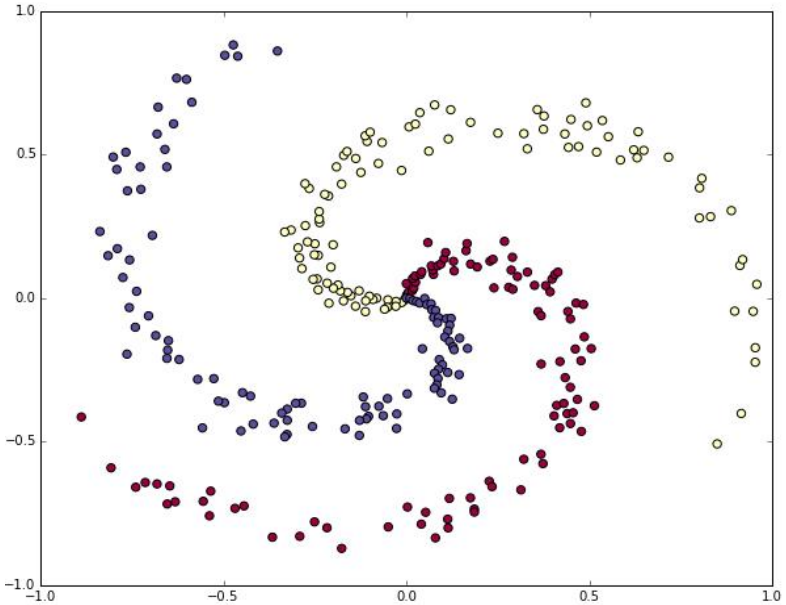
1 문제 정의



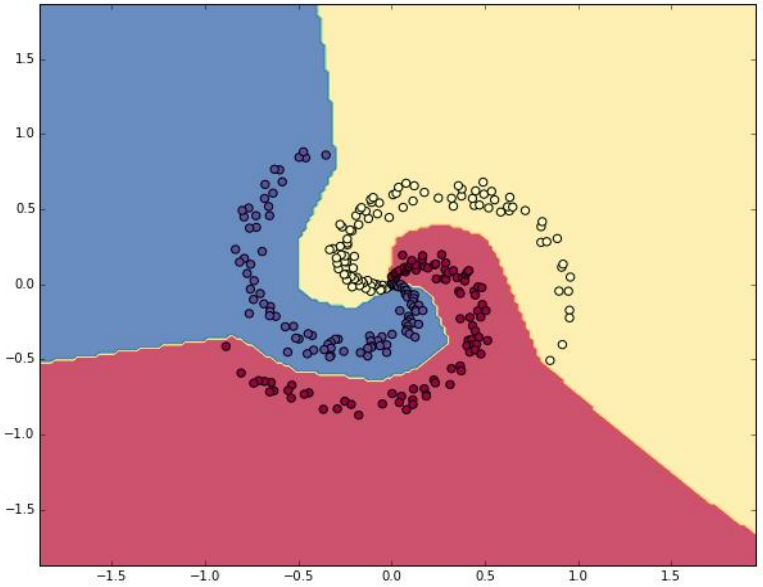
Classification 문제

2차원 나선형 데이터를 분류하는 신경망을 개발해보자.

Data Generation

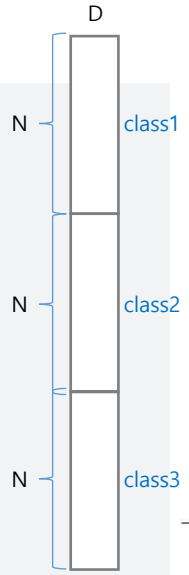


Classification



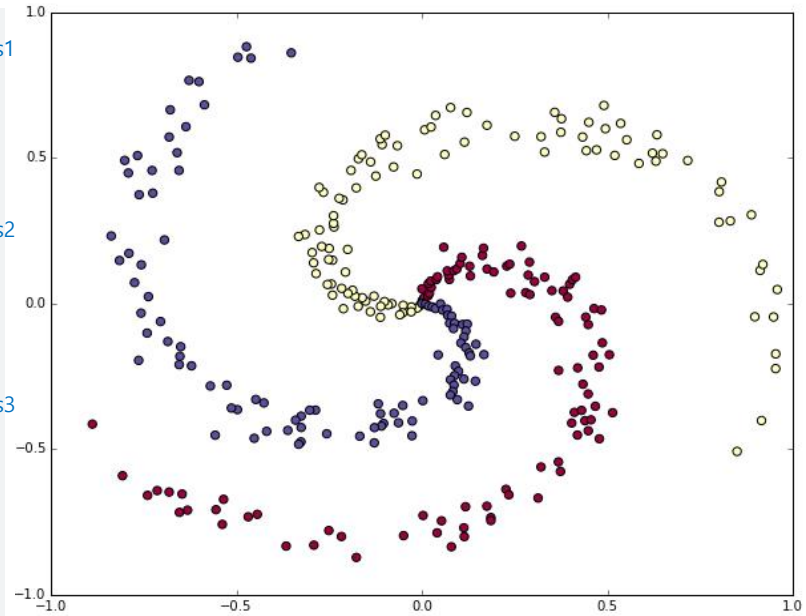
Hint : Spiral Dataset Generation

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

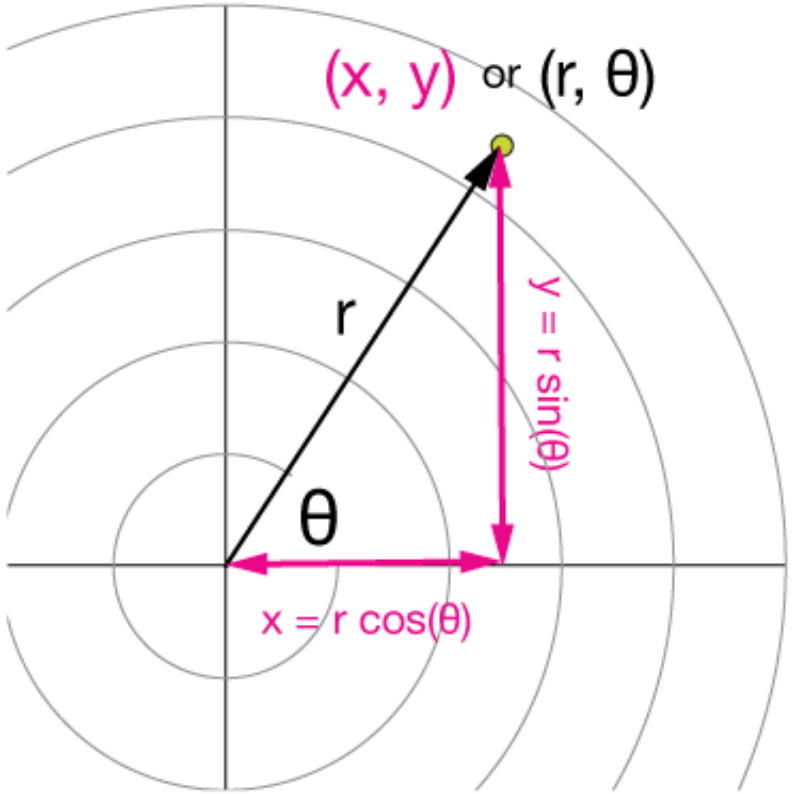


X (300,2) 입력 데이터　　**Y** (300) 레이블

- 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



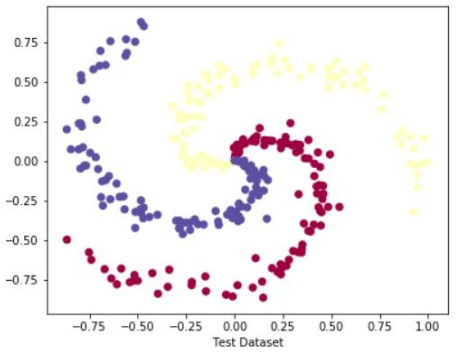
데이터 생성

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

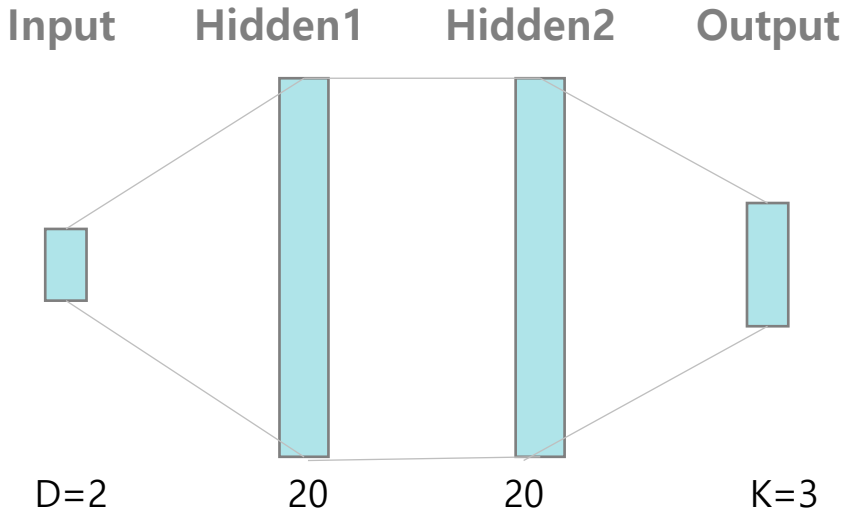
Hint : Network 구성

$x = (x_1, x_2)$

- 2차원 입력 벡터



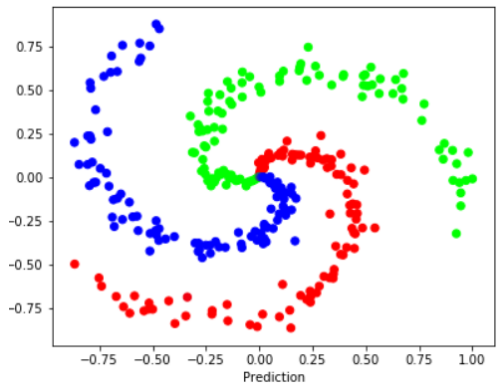
Fully Connected Network



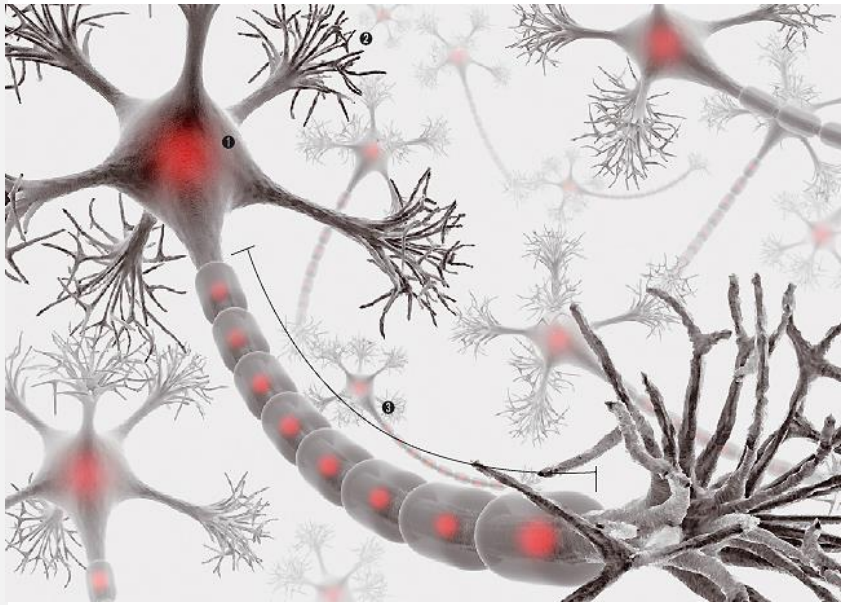
- 2-hidden layers
- 20 neurons
- Relu

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

K개 Class에 속할 확률



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
```

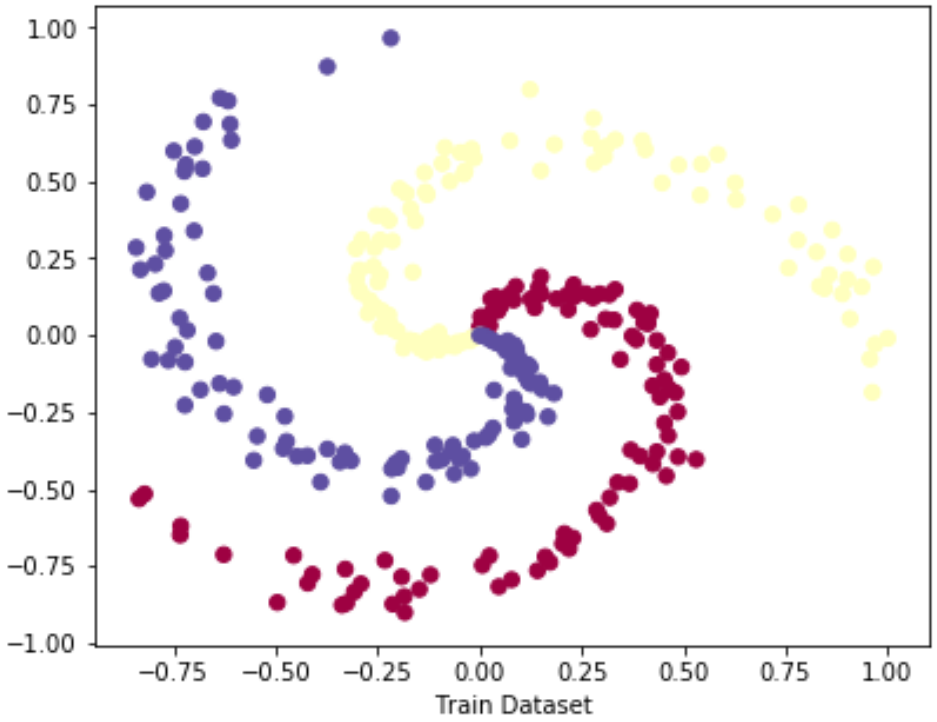
데이터셋 생성

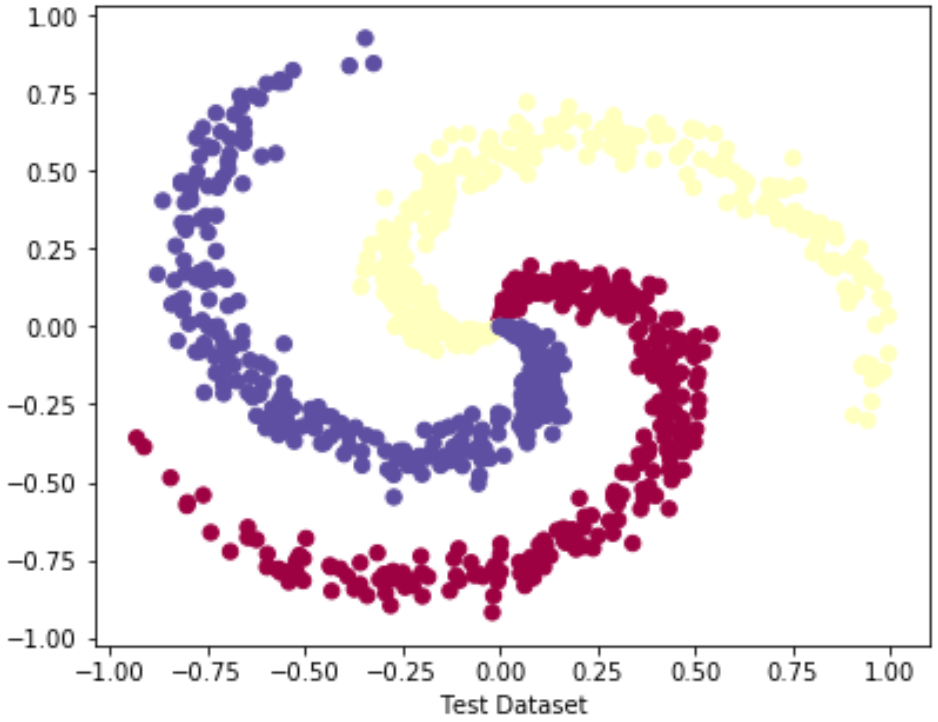
```
# generate the data
theta = 4 # class 별 시작 각도 : theta 배수

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

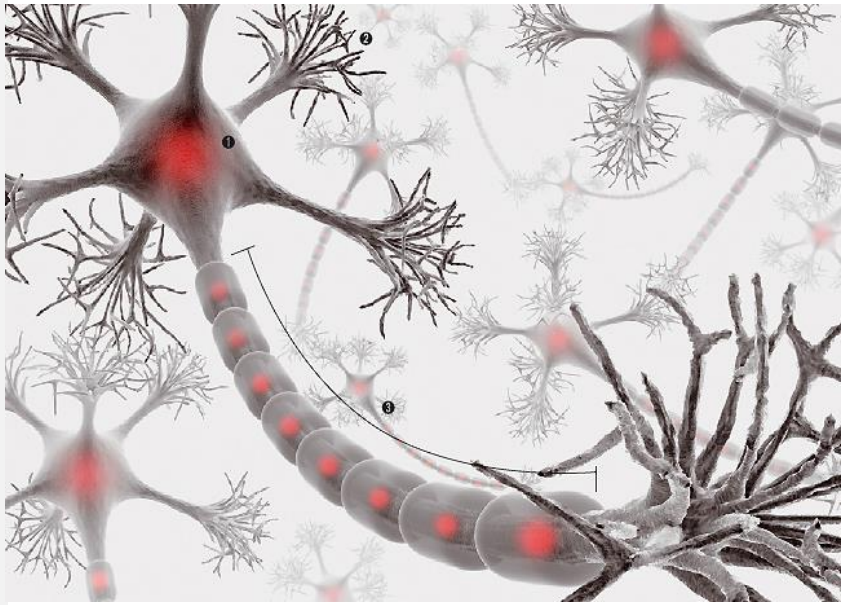
```
N = 200 # number of points per class
D = 2     # dimensionality
K = 3     # number of classes

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

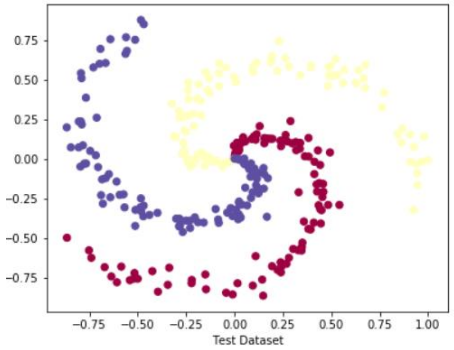


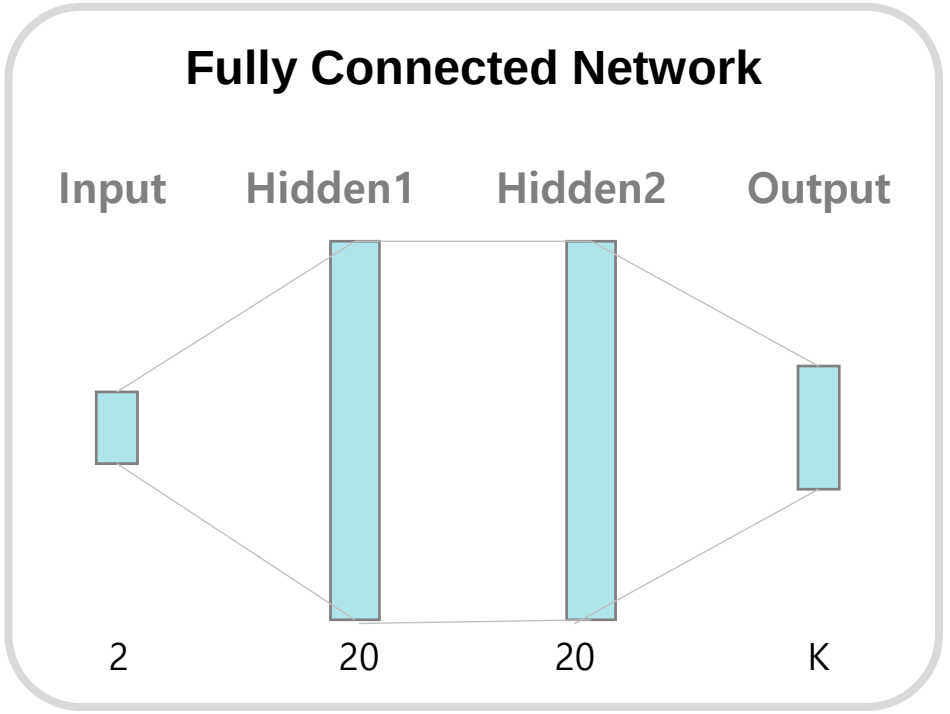
문제

Spiral Classification 코드를 Keras로 변경해보자!

$x = (x_1, x_2)$

- 2차원 입력 벡터

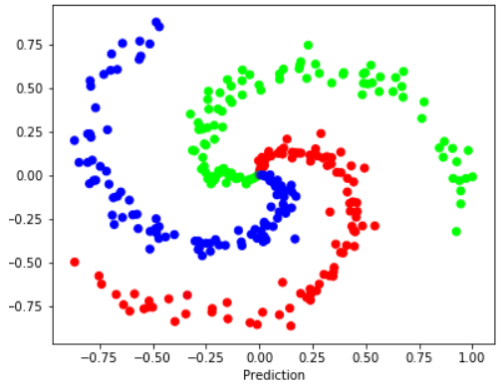




- 2-hidden layers
- 20 neurons
- Relu

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

K개 Class에 속할 확률



모델 정의 (문제)

```
# Dense를 이용해서 3계층 신경망을 구축하시오.
# hidden 20, hidden 20, output K
# 첫번째 Dense 계층에 input_shape을 지정하시오
model = keras.Sequential(#your code)
```

참고 tf.keras.layers.Dense

```
tf.keras.layers.Dense(
    units, activation=None, use_bias=True, kernel_initializer='glorot_uniform',
    bias_initializer='zeros', kernel_regularizer=None, bias_regularizer=None,
    activity_regularizer=None, kernel_constraint=None, bias_constraint=None,
    **kwargs
)
```

- **units**: 뉴런 개수, Positive integer, dimensionality of the output space.
- **activation**: Activation function to use. If you don't specify anything, no activation is applied (ie. "linear" activation: a(x) = x).
- **use_bias**: Boolean, whether the layer uses a bias vector.
- **kernel_initializer**: Initializer for the kernel weights matrix.
- **bias_initializer**: Initializer for the bias vector.
- **kernel_regularizer**: Regularizer function applied to the kernel weights matrix.
- **bias_regularizer**: Regularizer function applied to the bias vector.
- **activity_regularizer**: Regularizer function applied to the output of the layer (its "activation")..
- **kernel_constraint**: Constraint function applied to the kernel weights matrix.
- **bias_constraint**: Constraint function applied to the bias vector.

https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dense

모델 컴파일 (문제)

훈련에 필요한 Optimizer, Loss, Metrics 설정
(단, Optimizer는 Adam, Loss는 Label을 One-Hot Encoding 해주는 Cross Entropy로 Metric은 accuracy로 설정)

```
model.compile(optimizer= #your code,
              loss= #your code,
              metrics= #your code)
```

참고 tf.keras.losses.SparseCategoricalCrossentropy

```
tf.keras.losses.SparseCategoricalCrossentropy(
    from_logits=False, reduction=losses_utils.ReductionV2.AUTO,
    name='sparse_categorical_crossentropy'
)
```

from_logits: Model의 출력 값이 Softmax를 거치기 전이면 True, Softmax를 실행했다면 False

SparseCategoricalCrossentropy vs. CategoricalCrossentropy

- SparseCategoricalCrossentropy는 label을 One-Hot vector로 자동 변환해 줌
- CategoricalCrossentropy는 label이 이미 One-Hot vector로 준비된 경우에 사용

https://www.tensorflow.org/api_docs/python/tf/keras/losses/SparseCategoricalCrossentropy

모델 훈련 (문제)

모델을 훈련하기 위해 training set, label, epoch 등을 지정

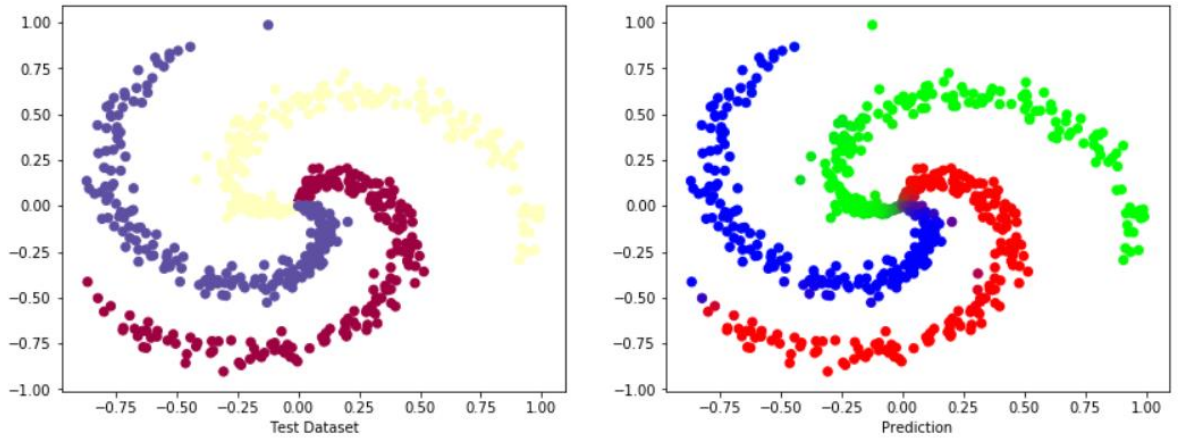
```
# epoch는 100 이상
model.fit(#your code)
```

```
Epoch 1/100
19/19 [==============================] - 0s 1ms/step - loss: 1.0984 - accuracy: 0.3017
Epoch 2/100
19/19 [==============================] - 0s 1ms/step - loss: 1.0649 - accuracy: 0.4033
Epoch 3/100
19/19 [==============================] - 0s 1ms/step - loss: 1.0334 - accuracy: 0.5117
...
Epoch 98/100
19/19 [==============================] - 0s 1ms/step - loss: 0.1024 - accuracy: 0.9867
Epoch 99/100
19/19 [==============================] - 0s 1ms/step - loss: 0.1005 - accuracy: 0.9917
Epoch 100/100
19/19 [==============================] - 0s 2ms/step - loss: 0.0991 - accuracy: 0.9883
```

https://www.tensorflow.org/api_docs/python/tf/keras/Model

모델 정의 (테스트)

```
test_output_pred = model.predict(test_x)

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, s=40, cmap=plt.cm.Spectral)
plt.xlabel("Prediction")
plt.show()
```



Thank you!