DL_HW04_곽용하_Kwak Yongha_2014121047

(a)

1) tf.keras.layers.dense

- Role: Fully(Densely) connected layer, 즉 완전연결계층을 만들어주는 역할을 합니다.

- Arguments and their roles:

    Units: output space의 차원

    Activation function: input->hidden, hidden->output layer 사이에서의 activation function을 무엇으로 할지 설정한다.

    use_bias: bias 항을 사용할지 여부

    kernel_initializer: weight 값들에 대한 initializer

    bias_initializer: bias 값들에 대한 initializer

    kernel(or bias, activity)_regularizer: kernel weights(or bias, activation)에 대한 정규화

    kernel_constraint: kernel weight에 제약을 가하는 것

    bias_constraint: 마찬가지로 bias에 제약을 가하는 것

2) tf.keras.layers.Dropout

- Role: 정규화의 방식중 하나인 드롭아웃을 쉽게 구현해주는 역할을 한다. Input에 적용.

- Arguments and their roles:

    rate: 몇 퍼센트의 input을 dropout 시킬 것인지 설정 (0~1)

    noise-shape: the shape of the binary dropout mask

    seed: 임의의 seed. 정수값

(b)

```
# 기초 작업

# Importing the libraries

from numpy import loadtxt
```

```
import numpy as np
import tensorflow as tf
import os
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from tensorflow.keras.layers.experimental import preprocessing

# Change Working Directory
os.getcwd()
os.chdir("C:/Users/david/Downloads")

# load the dataset
dataset_train = loadtxt('mistletoeTrain.csv', delimiter=',')

# split into input (X) and output (y) variables
x_train = dataset_train[:,0:4]
y_train = dataset_train[:,4]
print(x_train.shape)
(20000, 4)
print(y_train.shape)
(20000, )

# normalization
normalizer = preprocessing.Normalization()
normalizer.adapt(np.array(x_train))
```

```
print(normalizer.mean.numpy())
model = tf.keras.models.Sequential([
  normalizer,
  tf.keras.layers.Dense(16, activation='softmax'),
  tf.keras.layers.Dense(32, activation='softmax'),
  tf.keras.layers.Dense(16, activation='softmax'),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(1, activation='sigmoid')
])
model.summary()
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
normalization (Normalization (None, 4)                 9
_________________________________________________________________
dense (Dense)                (None, 16)                80
_________________________________________________________________
dense_1 (Dense)              (None, 32)                544
_________________________________________________________________
dense_2 (Dense)              (None, 16)                528
_________________________________________________________________
dropout (Dropout)            (None, 16)                0
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 17
```

===

Total params: 1,178

Trainable params: 1,169

Non-trainable params: 9

(c)

```
# Compiling the model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

# Fitting the neural network to the training set
model.fit(x_train, y_train, epochs = 50)
```

Epoch 1/50

625/625 [==============================] - 2s 2ms/step - loss: 0.4860 - accuracy: 0.8984

Epoch 2/50

625/625 [==============================] - 1s 2ms/step - loss: 0.3392 - accuracy: 0.9051

Epoch 3/50

625/625 [==============================] - 1s 2ms/step - loss: 0.3197 - accuracy: 0.9051

Epoch 4/50

625/625 [==============================] - 1s 2ms/step - loss: 0.3130 - accuracy: 0.9051

Epoch 5/50

625/625 [==========] - 1s 2ms/step - loss: 0.3066 - accuracy: 0.9051

Epoch 6/50

625/625 [==========] - 1s 2ms/step - loss: 0.3008 - accuracy: 0.9051

Epoch 7/50

625/625 [==========] - 1s 2ms/step - loss: 0.2972 - accuracy: 0.9051

Epoch 8/50

625/625 [==========] - 1s 2ms/step - loss: 0.2948 - accuracy: 0.9051

Epoch 9/50

625/625 [==========] - 1s 2ms/step - loss: 0.2914 - accuracy: 0.9051

Epoch 10/50

625/625 [==========] - 1s 2ms/step - loss: 0.2893 - accuracy: 0.9051

Epoch 11/50

625/625 [==========] - 1s 2ms/step - loss: 0.2888 - accuracy: 0.9051

Epoch 12/50

625/625 [==========] - 1s 2ms/step - loss: 0.2867 - accuracy: 0.9051

Epoch 13/50

625/625 [==========] - 1s 2ms/step - loss: 0.2858 - accuracy: 0.9051

Epoch 14/50

625/625 [==========] - 1s 2ms/step - loss: 0.2854 - accuracy: 0.9051

Epoch 15/50

625/625 [==============================] - 1s 2ms/step - loss: 0.2846 - accuracy: 0.9051

Epoch 16/50

625/625 [==============================] - 1s 2ms/step - loss: 0.2834 - accuracy: 0.9051

Epoch 17/50

625/625 [==============================] - 1s 2ms/step - loss: 0.2832 - accuracy: 0.9051

Epoch 18/50

625/625 [==============================] - 1s 2ms/step - loss: 0.2829 - accuracy: 0.9051

Epoch 19/50

625/625 [==============================] - 1s 2ms/step - loss: 0.2822 - accuracy: 0.9051

Epoch 20/50

625/625 [==============================] - 1s 2ms/step - loss: 0.2809 - accuracy: 0.9051

Epoch 21/50

625/625 [==============================] - 1s 2ms/step - loss: 0.2808 - accuracy: 0.9051

Epoch 22/50

625/625 [==============================] - 1s 2ms/step - loss: 0.2808 - accuracy: 0.9051

Epoch 23/50

625/625 [==============================] - 1s 2ms/step - loss: 0.2802 - accuracy: 0.9051

Epoch 24/50

625/625 [==============================] - 1s 2ms/step - loss: 0.2799 - accuracy: 0.9051

Epoch 25/50

625/625 [==============================] - 1s 2ms/step - loss: 0.2788 - accuracy: 0.9051

Epoch 26/50

625/625 [==============================] - 1s 2ms/step - loss: 0.2787 - accuracy: 0.9051

Epoch 27/50

625/625 [==============================] - 1s 2ms/step - loss: 0.2790 - accuracy: 0.9051

Epoch 28/50

625/625 [==============================] - 1s 2ms/step - loss: 0.2781 - accuracy: 0.9051

Epoch 29/50

625/625 [==============================] - 1s 2ms/step - loss: 0.2778 - accuracy: 0.9051

Epoch 30/50

625/625 [==============================] - 1s 2ms/step - loss: 0.2779 - accuracy: 0.9051

Epoch 31/50

625/625 [==============================] - 1s 2ms/step - loss: 0.2774 - accuracy: 0.9051

Epoch 32/50

625/625 [==============================] - 1s 2ms/step - loss: 0.2771 - accuracy: 0.9051

Epoch 33/50

625/625 [==============================] - 1s 2ms/step - loss: 0.2768 - accuracy: 0.9051

Epoch 34/50

625/625 [==============================] - 1s 2ms/step - loss: 0.2764 - accuracy: 0.9051

Epoch 35/50

625/625 [==============================] - 1s 2ms/step - loss: 0.2761 - accuracy: 0.9051

Epoch 36/50

625/625 [==============================] - 1s 2ms/step - loss: 0.2753 - accuracy: 0.9051

Epoch 37/50

625/625 [==============================] - 1s 2ms/step - loss: 0.2764 - accuracy: 0.9051

Epoch 38/50

625/625 [==============================] - 1s 2ms/step - loss: 0.2755 - accuracy: 0.9051

Epoch 39/50

625/625 [==============================] - 1s 2ms/step - loss: 0.2755 - accuracy: 0.9051

Epoch 40/50

625/625 [==============================] - 1s 2ms/step - loss: 0.2754 - accuracy: 0.9051

Epoch 41/50

625/625 [==============================] - 1s 2ms/step - loss: 0.2755 - accuracy: 0.9051

Epoch 42/50

625/625 [==============================] - 1s 2ms/step - loss: 0.2750 - accuracy: 0.9051

Epoch 43/50

625/625 [==============================] - 1s 2ms/step - loss: 0.2754 - accuracy: 0.9051

Epoch 44/50

625/625 [==============================] - 1s 2ms/step - loss: 0.2751 - accuracy: 0.9051

Epoch 45/50

625/625 [==============================] - 1s 2ms/step - loss: 0.2743 - accuracy: 0.9051

Epoch 46/50

625/625 [==============================] - 1s 2ms/step - loss: 0.2745 - accuracy: 0.9051

Epoch 47/50

625/625 [==============================] - 1s 2ms/step - loss: 0.2746 - accuracy: 0.9051

Epoch 48/50

625/625 [==============================] - 1s 2ms/step - loss: 0.2744 - accuracy: 0.9051

Epoch 49/50

625/625 [==============================] - 1s 2ms/step - loss: 0.2739 - accuracy: 0.9051

Epoch 50/50

625/625 [==============================] - 1s 2ms/step - loss: 0.2748 - accuracy: 0.9051

Accuracy는 0.9051이다.

(d)

dataset_test = loadtxt('mistletoeTest.csv', delimiter=',')

```
x_test = dataset_train[:,0:4]

y_test = dataset_train[:,4]

model.evaluate(x_test,y_test, verbose=True)
```

```
170/170 [==============================] - 1s 2ms/step - loss: 0.4532 - accuracy: 0.8205

Out[20]: [0.45324593782424927, 0.8204750418663025]
```

Accuracy는 0.8205이다. 이때 test data에서는 dropout을 하지 않는다.