활성화 함수

activation 인자 값

레이어에 `activation=` 인자로 전달

- `sigmoid` : 로지스틱 함수 (이항 분류의 출력, 한 범주의 확률을 0~1로 출력)
- `softmax` : 소프트맥스 함수 (다항 분류의 출력, 각 범주의 확률을 0~1로 출력)
- `relu` : ReLU (은닉층)
- `tahn` : 쌍곡탄젠트 (은닉층)

전체 함수 목록: https://www.tensorflow.org/api_docs/python/tf/keras/activations

활성화 레이어

활성화만 담당하는 레이어

`tf.keras.layers.Softmax`

`tf.keras.layers.ReLU`

다른 활성화 함수는 `tf.keras.layers.Activation`을 사용하여 추가

```
tf.keras.layers.Activation('sigmoid')
```

손실 함수

텐서플로에서 손실 함수의 사용법

1. model.compile에서 손실함수의 이름을 문자열로 전달

```
model.compile(loss='sparse_categorical_crossentropy')
```

2. `tf.keras.losses`의 손실 함수를 사용

```
model.compile(loss=tf.keras.losses.sparse_categorical_crossentropy)
```

3. `tf.keras.losses`의 손실 함수 클래스를 사용 (설정 가능)

```
loss_fn = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
model.compile(loss=loss_fn)
```

BinaryCrossentropy

이항 분류에서 출력으로 `sigmoid`를 사용할 때

문자열: `"binary_crossentropy"`

함수: `tf.keras.losses.binary_crossentropy`

클래스: `tf.keras.losses.BinaryCrossentropy`

CategoricalCrossentropy

다항 분류에서 출력으로 `softmax`를 사용할 때

`y_train`이 one-hot encoding 되어 있을 때

문자열: `"categorical_crossentropy"`

함수: `tf.keras.losses.categorical_crossentropy`

클래스: `tf.keras.losses.CategoricalCrossentropy`

One-Hot Encoding

다항 분류에서 답을 벡터로 표현하는 방법

벡터의 길이는 범주의 개수와 동일

정답은 1(one-hot), 나머지는 0(cold)으로 표현

예: 세 개의 범주 중, 0번째 범주가 답일 경우 `[1, 0, 0]`

SparseCategoricalCrossentropy

다항 분류에서 출력으로 `softmax`를 사용할 때

`y_train`이 범주의 인덱스로 되어 있을 때(one-hot encoding 안됨)

문자열: `"sparse_categorical_crossentropy"`

함수: `tf.keras.losses.sparse_categorical_crossentropy`

클래스: `tf.keras.losses.SparseCategoricalCrossentropy`

MeanSquaredError

회귀에서 사용. 오차 제곱의 평균.

문자열: `"mse"`

함수: `tf.keras.losses.MSE`

클래스: `tf.keras.losses.MeanSquaredError`

MeanAbsoluteError

회귀에서 사용. 오차 제곱의 절대값

문자열: `"mae"`

함수: `tf.keras.losses.MAE`

클래스: `tf.keras.losses.MeanAbsoluteError`

MNIST 손글씨 직접 입력

이미지 열기

```
import PIL.Image
img = PIL.Image.open('number.png')
```

이미지 전처리

```
import numpy as np
x = 1 - np.array(img.convert('L').resize((28, 28)), dtype='float32') / 255
```

- `img.convert('L')` : 흑백으로 변경
- `.resize((28, 28))` : 가로 28, 세로 28 크기로 변경
- `1 -` : 흰 바탕에 검은 글씨로 쓴 경우, 학습된 이미지는 배경이 0이므로 흑백 반전

차원 추가

```
x = np.expand_dims(x, 0)
```

`x`의 형태가 `(28, 28)`이므로 0번째 차원을 추가하여 `(1, 28, 28)` 형태로 변경한다

합성곱 신경망

임포트

```
import tensorflow as tf
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
```

모형 정의

```
model = tf.keras.Sequential()
```

`model.add`를 통해 레이어를 하나씩 추가할 수 있음

합성곱 레이어

첫 레이어이므로 `input_shape`을 지정 (안해도 됨)

```
model.add(
    Conv2D(
        filters=32,
        kernel_size=(3, 3),
        stride=1,
        padding='same',
        activation='relu',
        input_shape=(28,28,1)))
```

풀링 레이어

```
model.add(MaxPooling2D(pool_size=(2, 2)))
```

합성곱 레이어와 풀링 레이어 추가

`input_shape`을 지정하지 않아도 된다

```
model.add(Conv2D(32, (3, 3), padding='same', activation='relu'))
MaxPooling2D(pool_size=(2, 2))
```

완전 연결 레이어

합성곱 레이어와 완전 연결 레이어는 형태가 다르므로 `Flatten` 레이어 추가

```
model.add(Flatten())
```

완전 연결 레이어로 된 은닉층 추가

```
model.add(Dense(64, activation='relu'))
```

최종 출력

```
model.add(Dense(10, activation='softmax'))
```

모형 요약

```
model.summary()
```

Dropout

Dropout

```
from tensorflow.keras.layers import Dropout
```

Dropout 레이어 추가

```
...
model.add(Flatten())
model.add(Dropout(rate=0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(rate=0.5))
model.add(Dense(10, activation='softmax'))
```

컬러 이미지

CIFAR10

```
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.cifar10.load_data()
```

이미지 보기

```
import matplotlib.pyplot as plt
plt.imshow(x_train[0])
```

```
y_train[0]
```

CIFAR10 범주 번호

- 0: airplane
- 1: automobile
- 2: bird
- 3: cat
- 4: deer
- 5: dog
- 6: frog
- 7: horse
- 8: ship
- 9: truck

모형 정의

```
model = tf.keras.Sequential([
    Conv2D(
        32, (3, 3), padding='same', activation='relu',
        input_shape=(32,32,3)),
    ...
```

`input_shape=(32, 32, 3)` 으로 설정

그림 크기는 32x32

컬러 이미지 이므로 채널 3개의 차원이 추가

하이퍼파라미터 튜닝

keras-tuner

하이퍼파라미터 튜닝을 위한 라이브러리

설치:

```
!pip install -q -U keras-tuner
```

임포트:

```
import kerastuner as kt
```

모형 수립 함수

```
def model_builder(hp):
  # 은닉층의 크기 범위
  units = hp.Int('units', min_value=32, max_value=512, step=32)

  model = tf.keras.Sequential()
  model.add(Flatten(input_shape=(28, 28)))
  model.add(Dense(units=units, activation='relu'))
  model.add(Dense(10, activation='softmax'))

  # 학습률 범위
  learning_rate = hp.Choice('learning_rate', values=[1e-2, 1e-3, 1e-4])
  optim_fn = tf.keras.optimizers.Adam(learning_rate=learning_rate)

  model.compile(optimizer=optim_fn,
                loss='sparse_categorical_crossentropy',
                metrics=['accuracy'])

  return model
```

튜닝 방법 설정

```
tuner = kt.Hyperband(model_builder,
                     objective='val_accuracy',
                     max_epochs=2,
                     factor=3,
                     directory='my_dir',
                     project_name='intro_to_kt')
```

- `Hyperband` : MAB 방식의 하이퍼파라미터 최적화 알고리즘
- `objective = 'val_accuracy'` : 검증 데이터에서 정확도를 기준으로 결정
- `max_epochs` : 하이퍼파라미터 당 최대 에포크
- `factor` : 한 번의 비교에서 줄일 후보의 비율 (`factor=3` 이면 1/3으로 줄임)

요약

```
tuner.search_space_summary()
```

화면 지우기 콜백

```
import IPython

class ClearTrainingOutput(tf.keras.callbacks.Callback):
  def on_train_end(*args, **kwargs):
    IPython.display.clear_output(wait = True)
```

노트북에서 화면을 지우기위한 코드

콜백(callback): 프로그래밍에서 다른 함수의 인자로 전달되어, 특정한 처리 이후에 수행되는 함수

데이터 분할

```
from sklearn.model_selection import train_test_split
x_tr, x_val, y_tr, y_val = train_test_split(x_train, y_train, test_size=0.1, random_state=42)
```

하이퍼파라미터 튜닝

`model.fit`과 사용법이 같음

```
tuner.search(
    x_tr,
    y_tr,
    epochs=2,
    validation_data=(x_val, y_val),
    callbacks=[ClearTrainingOutput()])
```

최적 파라미터

```
best_hps = tuner.get_best_hyperparameters()[0]
```

은닉층의 크기

```
best_hps.get('units')
```

학습률

```
best_hps.get('learning_rate')
```

최적 파라미터로 학습

최적 모형 만들기

```
model = tuner.hypermodel.build(best_hps)
```

또는 하이퍼파라미터 튜닝 중에 학습된 모형을 불러오기

```
model = tuner.get_best_models()[0]
```

추가 학습

```
model.fit(x_tr, y_tr, epochs = 10, validation_data = (x_val, y_val))
```