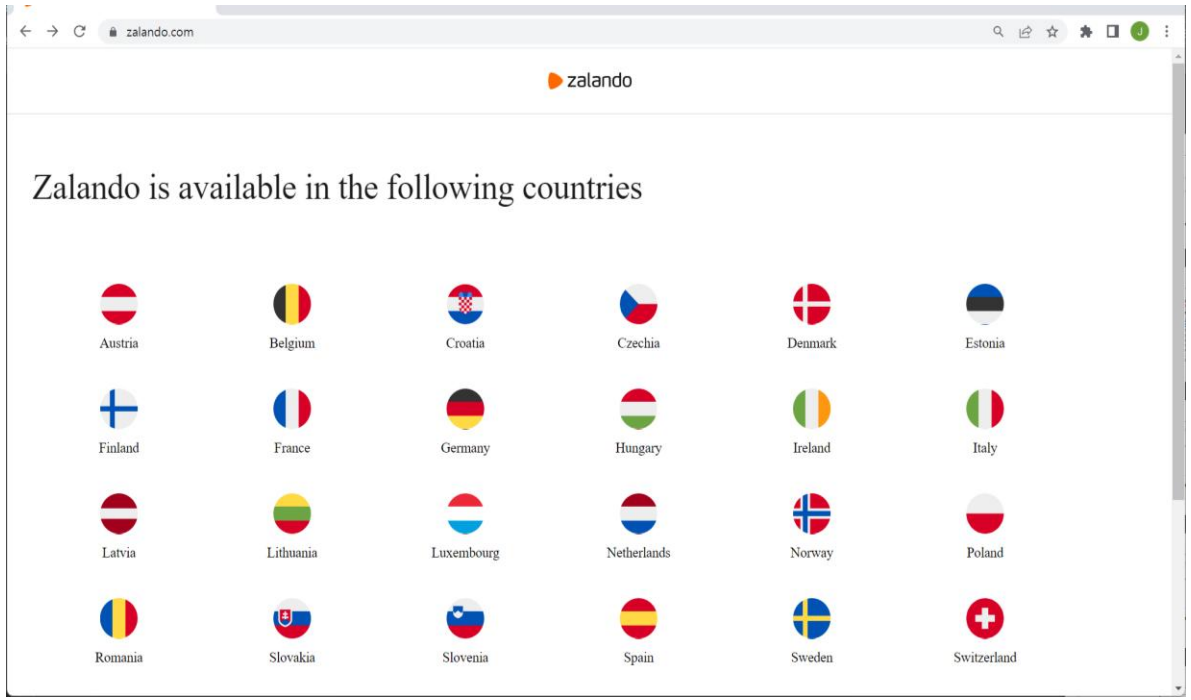
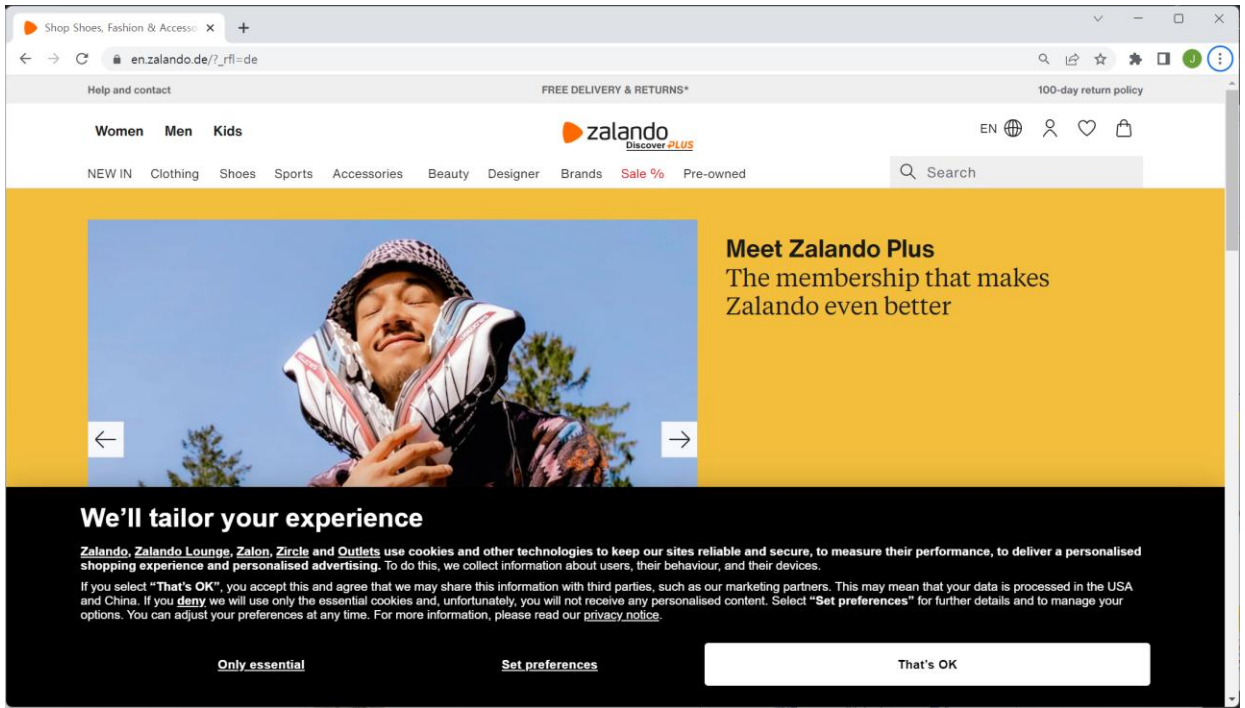
Deep Learning

Image Classification by MLP

Image classification 문제

- **목표: Fashion-MNIST data를 사용하여 각 의류 이미지의 Class를 분류한다.**
- **Fashion-MNIST data**
  - https://github.com/zalandoresearch/fashion-mnist
  - Fashion-MNIST is a dataset of Zalando's article images—consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image, associated with a label from 10 classes.





Data Description

Data : 28x28 grayscale images

Label

Label	Description
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

main.py

```
import tensorflow as tf
import matplotlib.pyplot as plt
# from ImageClassifier import ImageClassifier   ## 일단 주석 처리해 둠. ImageClassifier class만들고 나서 주석 해제


def run_classifier():
    fashion_mnist = tf.keras.datasets.fashion_mnist
    (train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()
    class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
                   'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']

    print("Train data shape")
    print(train_images.shape)
    print("Train data labels")
    print(train_labels)
    print("Test data shape")
    print(test_images.shape)
    print("Test data labels")
    print(test_labels)
```

main.py (cont'd)

```
##### def run_classifier() 내용 이어서.
 plt.figure()
 plt.imshow(train_images[0])
 plt.colorbar()
 plt.grid(False)
 plt.show()

 train_images = train_images / 255.0
 test_images = test_images / 255.0

 plt.figure(figsize=(10,10))
 for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i], cmap=plt.cm.binary)
    plt.xlabel(class_names[train_labels[i]])
 plt.show()
```

```
##### classifier train and predict – begin

ImageClassifier class와 함께
우리가 만들어야 할 부분

##### classifier train and predict – end
##### def run_classifier() 끝

if __name__ == "__main__":
    # execute only if run as a script
    run_classifier()
```

Network Design Process

- **1. 사용할 Network 구조를 선택한다.**
- **2. Input Layer를 정의한다.**
  - 입력 데이터 차원은 어떻게 정의할 것인가?
- **3. Output Layer를 정의한다.**
  - 출력 데이터 차원 (node 개수), Activation Function은 무엇을 사용할 것인가?
- **4. Objective Function을 정의한다.**
  - Model 학습은 어떤 함수의 값을 최소화 할 것인가?
- **5. Hidden Layer 층을 정의한다.**
  - Hidden Layer는 몇 층으로 할 것인가?
  - 각 Layer의 node 개수, Activation Function은 무엇을 사용할 것인가?
- **6. Optimization Algorithm을 선택한다.**
  - 예) Gradient Descent? ADAM?

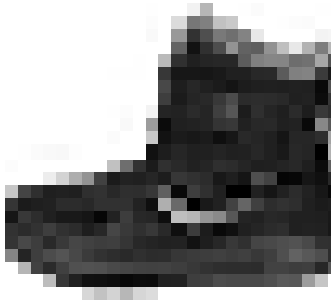
Network Design Process (Cont'd)

- **1. 사용할 Network 구조를 선택한다.**
  - 하려는 Task를 수행하기에 적합한 Network 구조를 선택하는 것이 좋다.
  - Image 를 사용한 분류이므로 Image에서 중요 feature를 추출하기 위한 popular한 구조 중 하나인 CNN (Convolutional Neural Network)를 사용할 수 도 있다.
  - 수업에서 배운 구조는 MLP 이므로 이번에는 우리가 알고 있는 MLP를 사용한다.
    - 분류 Task및 이미지의 형태가 상대적으로 단순하므로 MLP를 사용해도 어느 정도의 성능이 나온다고 기대하자.

Network Design Process (Cont'd)

- **2. Input Layer를 정의한다.**
  - 입력 데이터 차원은 어떻게 정의할 것인가?

분류하려는 이미지가 28x28 grayscale image 이므로 2 차원으로 표현된다.
따라서 입력 데이터의 shape는 28 by 28이 된다.

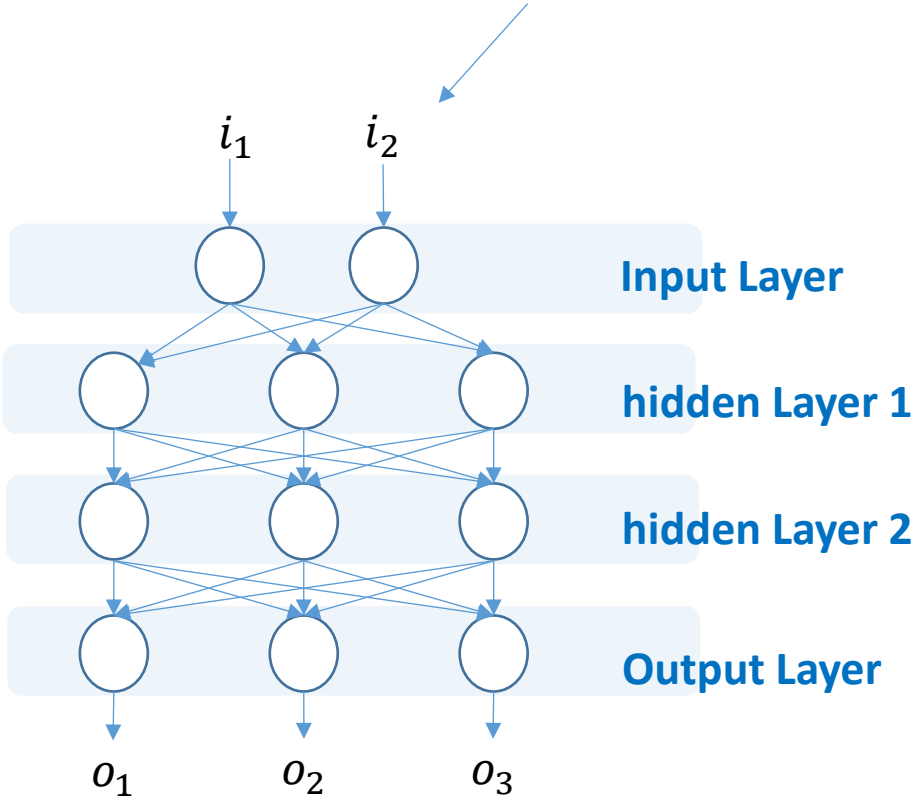


예)

```
input_layer = tf.keras.layers.Input(shape=[28, 28, ])
```

Network Design Process (Cont'd)

- **2. Input Layer를 정의한다. (Cont'd)**
  - 입력 데이터 차원은 어떻게 정의할 것인가?
  - 그런데 다음 layer는 Input layer의 node의 차원이 1차원이라 가정한다.
  - => **2 차원을 1 차원으로 변경**할 필요가 있다.



tf.keras.layers.Flatten(data_format=None, **kwargs)

입력으로 들어온 층을 1차원으로 변경.

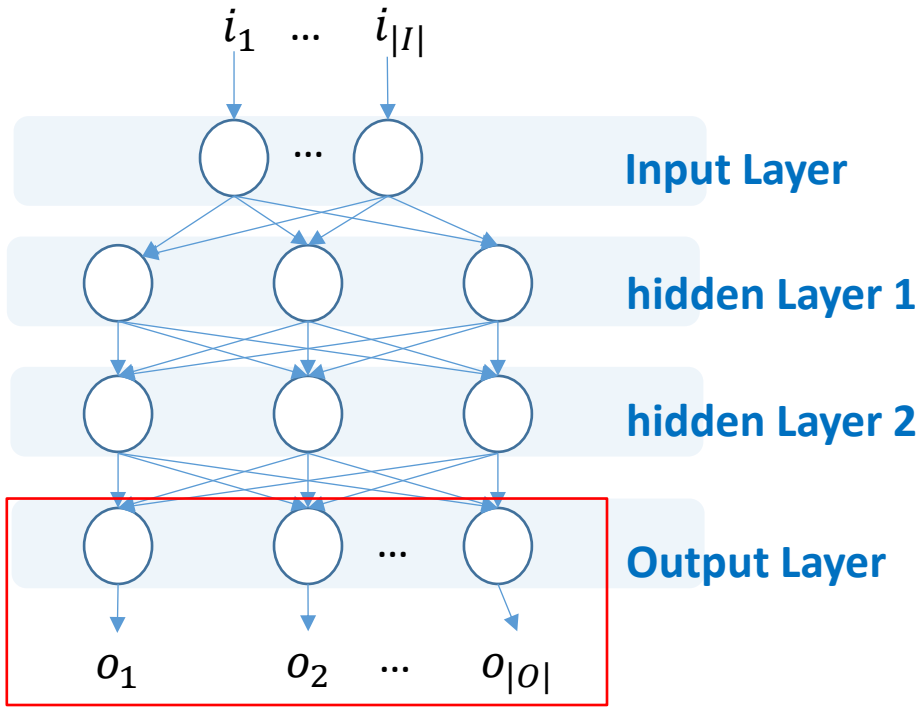
변경 방식: C에서 n-dimensional Array가 메모리에서 1차원으로 저장되는 것과 유사한 방식.

예)

```
input_layer = tf.keras.layers.Input(shape=[28, 28, ])
flatten_layer = tf.keras.layers.Flatten()(input_layer)
```

Network Design Process (Cont'd)

- **3. Output Layer를 정의한다.**
  - 출력 데이터 차원 (node 개수), Activation Function은 무엇을 사용할 것인가?



Label

Label	Description
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

Label (Class)가 3개 이상 존재하므로 multi-class classification problem 이다.
그렇다면 출력 데이터의 차원은 얼마가 좋을까?

Network Design Process (Cont'd)

- **3. Output Layer를 정의한다.**
  - 출력 데이터 차원 (node 개수), Activation Function은 무엇을 사용할 것인가?

Output Layer의 Activation 함수는 softmax를 사용한다.

예)

```
ac_func_softmax = tf.keras.activations.softmax
output_layer = tf.keras.layers.Dense(units=self.num_labels, activation=ac_func_softmax)(hidden_layer_2)
```

Softmax Function

$\vec{c}$ 가 다음과 같은 k차원의 real vector라 하자.

$\vec{c}$=<c_1, c_2, …, c_k>

그러면 $\vec{c}$ 의 i 차원 element의 값에 대한 softmax 값은 다음과 같이 정의 된다.

$$softmax(c_i) = \frac{e^{c_i}}{\sum_{j=1}^{k} e^{c_j}}$$

$$\sum_{i=1}^{k} softmax(c_i) = 1$$

Softmax Function은 Sigmoid Function (=Logistic Function)을 muli-dimension으로 generalization한 형태이다.

Softmax Function (Cont'd)

Why Softmax?

- 우리가 알고 싶은 것은 주어진 데이터가 각 category (class)에 속할 확률.
- 확률은 [0.0, 1.0] 의 범위.
- 데이터를 x, class 집합$C = \{c_1, c_2, \ldots, c_{|C|}\}$ 라하고, x가 class c_i에속할 확률을 $P_{x,i}$ 라 하면, $\sum_{c_i \in C} P_{x,i}$=1.0

Softmax Function is **often used as the last activation function of a neural network** to normalize the output of a network to a probability distribution over predicted output classes.

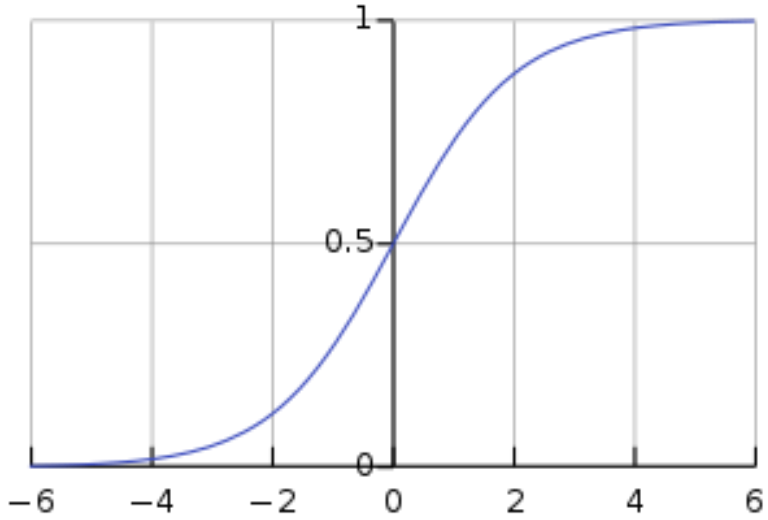
In probability theory, the output of the Softmax function can be **used to represent a categorical distribution** (A probability distribution over K different possible outcomes)

Network Design Process (Cont'd)

- Classifer 가 **Binary Classifier라면 output layer의 activation function은 softmax 대신 sigmoid를 사용**할 수 있다.

값 x가 [-INF, INF] 의 범위를 가지고 있다. 값 x가 양의 쪽으로 큰 값일수록 값 x가 Class C에 속할 확률이 크고, 음의 쪽으로 작은 값일수록 값 x 가 Class C에 속할 확률이 작다고 생각해 보자.

x값이 0이면?,
x 값이 2 면? 100이면?
혹은 x 값이 -2면? -100이면?



Network Design Process (Cont'd)

- **Softmax는 Sigmoid를 Multi-Class로 일반화한 버전**이다.
  - Binary Classifier라면 Class 개수가 2개이다.
  - 따라서 $\vec{c}$ 가 다음과 같은 2차원의 real vector라 하자.
    - $\vec{c}$=<c_1, c_2>
  - Binary Classifier는 하나의 class의 확률을 알면 다른 하나는 자동으로 알게 되므로 output dimension이 하나이다.
  - 따라서 Output의 값을 $\vec{c}$=<c_1, c_2> 에서 c_1 이라 하면, c_2 는 classifier의 출력이 없으므로 0 이라 할 수 있다.
    - $\vec{c}$=<c_1, 0>
  - $\vec{c}$=<c_1, 0> 일 때 $softmax(c_1)$ 의 값을 계산해 보라

Network Design Process (Cont'd)

- **4. Objective Function을 정의한다.**
  - Model 학습은 어떤 함수의 값을 최소화 할 것인가?
  - **Cross Entropy Loss**를 사용해 본다.
    - 이번 예에서는 Cross Entropy Loss인
    - Tf.keras.losses.CategoricalCrossentropy를 사용한다.

```
tf.keras.losses.CategoricalCrossentropy(from_logits=False )
```

from_logits=False (default 값이 False)로 되어 있으면, 입력으로 확률 값 [0.0, 1.0] 이 들어오는 것을 가정하고 Crossentropy를 계산함.
from_logits=True 라면, 입력값이 확률값이 아니라 가정하고 sigmoid or softmax함수를 사용하여 확률값으로 변경한 다음에 Crossentropy를 계산함.

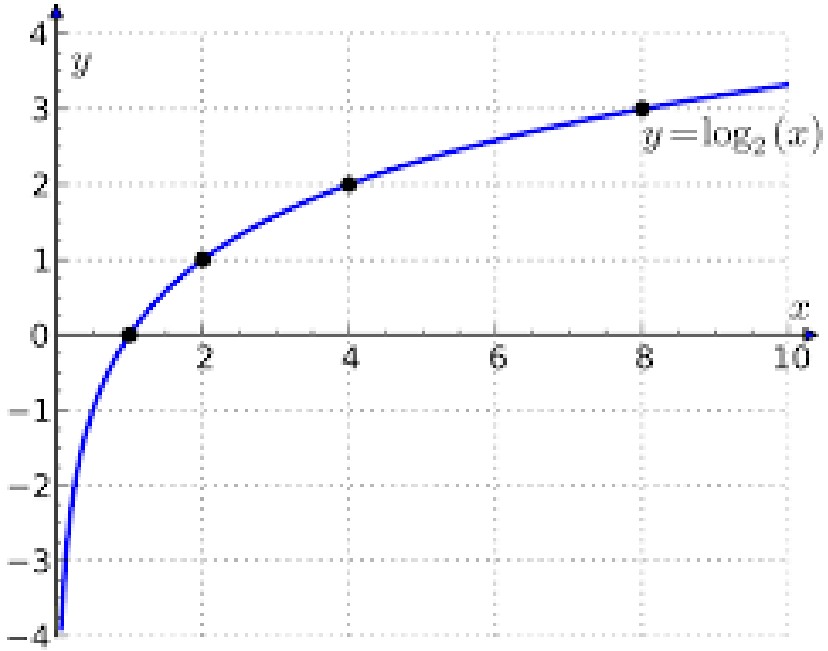
Cross-Entropy Loss (Multinomial (Categorical) Cross Entropy Loss)

$P = < p_1, p_2, \ldots, p_k >$ 가 데이터 x 가 k 개의 class 각각에 속할 **정답 확률(label vector)**이라 하고, $Q = < q_1, q_2, \ldots, q_k >$가 x 가 k 개의 class 각각에 속할 **예측 확률(predicted label vector)하 하면,** P 와 Q 사이의 Cross Entropy는 다음 식과 같다.

$$H_{cr}(P, Q) = -\sum_{i=1}^{k} p_i \cdot \log_2 q_i$$

정답 class를 잘 맞출 수록 $H_{cr}(P, Q)$ 의 값은 작아진다.

P = <1.0, 0.0>, Q = <0.5, 0.5> 일 경우와
P = <1.0, 0.0>, Q = <1.0, 0.0> 일 경우의
$H_{cr}(P, Q)$ 를 계산해 보라.



Binary Cross Entropy Loss

- **Cross-Entropy Loss의 Binary Classifier 특화 버전**

p_1 이 데이터 x 가 class 1에 속할 **정답 확률(label vector)**이라 하고, q_1 이 x 가 class1에 속할 **예측 확률(predicted label vector)이라 하면,** 정답과 예측 사이의 Binary Cross Entropy는 다음 식과 같다.

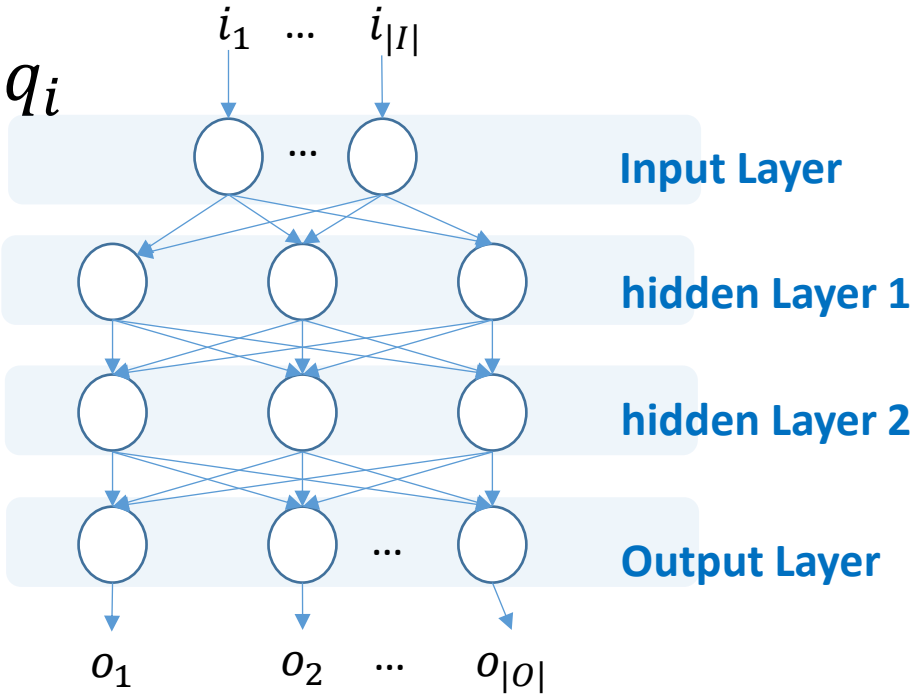
$$H_{cr}(P,Q) = -p_1 \cdot \log_2 q_1 - (1-p_1) \cdot \log_2(1-q_1)$$

위 식을 일반 Cross-Entropy의 식으로부터 유도해 보아라.

CategoricalCrossentropy() = Cross-Entropy Loss

$P = < p_1, p_2, \dots, p_k >$ 가 데이터 x 가 k 개의 class 각각에 속할 **정답 확률(label vector)**이라 하고, $Q = < q_1, q_2, \dots, q_k >$가 x 가 k 개의 class 각각에 속할 **예측 확률(predicted label vector)하 하면,** P 와 Q 사이의 Cross Entropy는 다음 식과 같다.

$$H_{cr}(P, Q) = -\sum_{i=1}^{k} p_i \cdot \log_2 q_i$$



tf.keras.losses.CategoricalCrossentropy(from_logits=False)

Neural Network의 출력이 주어진 입력이 속할 수 있는 10개의 class에 대한 확률분포이다.
Cross Entropy Loss를 사용하기 위해서는 정답도 element 10개인 확률분포(Vector로 주어져야 한다.)
*그러나, 우리가 가지고 있는 데이터는 0-9까지의 class index로 주어진다
=> class index로 주어진 **정답의 10차원 vector로 변환** 필요

to_onehotvec_label(index_labels, dim)

다음 함수는 0~9 까지의 정수로 표현된 “의류 그림의 클래스”를 10차원의 one-hot vector로 변경해 주는 함수.

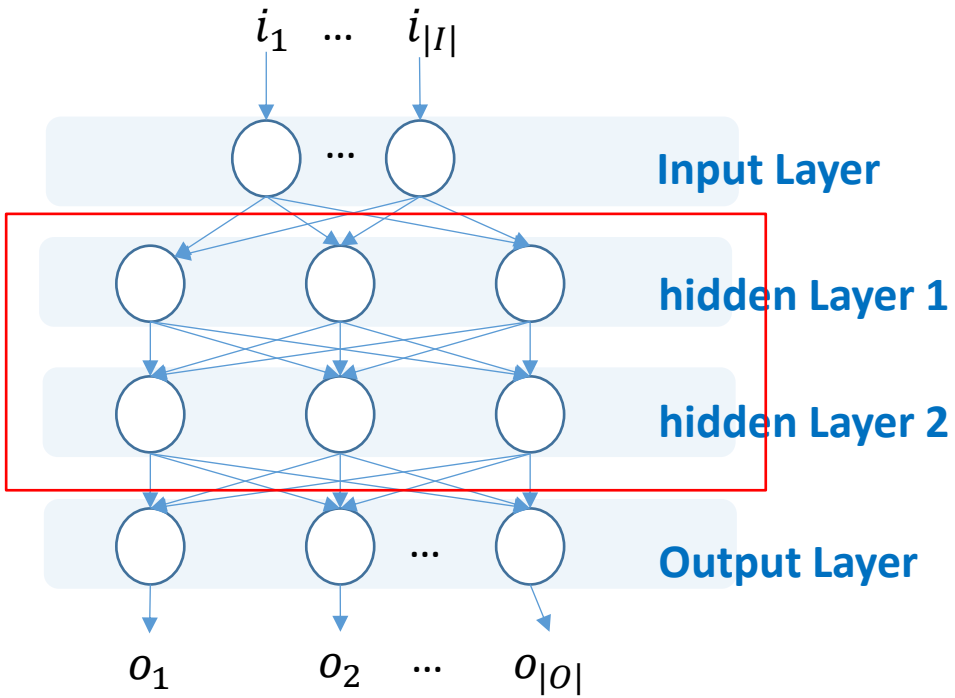
* ImageClassifier의 static method로 구현한 예

- 생각해 보기: 다음 코드의 생성 결과가 왜 One-hot vector 인가?
- staticmethod란?

```
@staticmethod
  def to_onehotvec_label(index_labels, dim):
      num_labels = len(index_labels)
      onehotvec_labels = np.zeros((num_labels, dim))
      for i, idx in enumerate(index_labels):
          onehotvec_labels[i][idx] = 1.0
      onehotvec_labels_tf = tf.convert_to_tensor(onehotvec_labels)
      return onehotvec_labels_tf
```

Network Design Process (Cont'd)

- **5. Hidden Layer 층을 정의한다.**
  - Hidden Layer는 몇 층으로 할 것인가?
  - 각 Layer의 node 개수, Activation Function은 무엇을 사용할 것인가?



이번 예에서는 Hidden Layer는 2 층, 각 Layer에 node 수 128개, Activation Function은 ReLU로 설정해 본다.
앞의 **Layer 수, node 수, Activation Function등의 설정은 일반적으로 cross-validation을 여러 번 해 보면서 최적의 값을 찾아야** 한다.

예)

```
ac_func = tf.keras.activations.relu
    hidden_layer_1 = tf.keras.layers.Dense(units=128, activation=
ac_func)(flatten_layer)
    hidden_layer_2 = tf.keras.layers.Dense(units=128, activation=
ac_func)(hidden_layer_1)
```

Network Design Process (Cont'd)

- **6. Optimization Algorithm을 선택한다.**
  - Adam Optimizer을 사용해 본다.

예) 모델 정의 전체 예.

```
input_layer = tf.keras.layers.Input(shape=[self.img_shape_x, self.img_shape_y, ])
flatten_layer = tf.keras.layers.Flatten()(input_layer)

ac_func_relu = tf.keras.activations.relu
hidden_layer_1 = tf.keras.layers.Dense(units=128, activation=ac_func_relu)(flatten_layer)
hidden_layer_2 = tf.keras.layers.Dense(units=128, activation=ac_func_relu)(hidden_layer_1)

ac_func_softmax = tf.keras.activations.softmax
output_layer = tf.keras.layers.Dense(units=self.num_labels, activation=ac_func_softmax)(hidden_layer_2)
classifier_model = tf.keras.Model(inputs=input_layer, outputs=output_layer)

opt_alg = tf.keras.optimizers.Adam(learning_rate=0.001)
loss_cross_e = tf.keras.losses.CategoricalCrossentropy(from_logits=False)
classifier_model.compile(optimizer=opt_alg, loss=loss_cross_e, metrics=['accuracy'])
```

ImageClassifier.py

```
import tensorflow as tf
import numpy as np

class ImageClassifier_MLP:
    def __init__(self, img_shape_x, img_shape_y, num_labels):
        self.img_shape_x = img_shape_x
        self.img_shape_y = img_shape_y
        self.num_labels = num_labels
        self.classifier = None


    def fit(self, train_imgs, train_labels, num_epochs):
        self.classifier.fit(train_imgs, train_labels, epochs=num_epochs)

    def predict(self, test_imgs):
        predictions = self.classifier.predict(test_imgs)
        return predictions
```

ImageClassifier_MLP.py (Cont'd)

```
# class ImageClassifier_MLP 의 member method

def build_MLP_model(self):
    input_layer = tf.keras.layers.Input(shape=[self.img_shape_x, self.img_shape_y, ])
    flatten_layer = tf.keras.layers.Flatten()(input_layer)

    ac_func_relu = tf.keras.activations.relu
    hidden_layer_1 = tf.keras.layers.Dense(units=128, activation=ac_func_relu)(flatten_layer)
    hidden_layer_2 = tf.keras.layers.Dense(units=128, activation=ac_func_relu)(hidden_layer_1)

    ac_func_softmax = tf.keras.activations.softmax
    output_layer = tf.keras.layers.Dense(units=self.num_labels, activation=ac_func_softmax)(hidden_layer_2)
    classifier_model = tf.keras.Model(inputs=input_layer, outputs=output_layer)

    opt_alg = tf.keras.optimizers.Adam(learning_rate=0.001)
    loss_cross_e = tf.keras.losses.CategoricalCrossentropy(from_logits=False)
    classifier_model.compile(optimizer=opt_alg, loss=loss_cross_e, metrics=['accuracy'])

    self.classifier = classifier_model
```

ImageClassifier_MLP.py (Cont'd)

```
# class ImageClassifier_MLP 의 static method
 @staticmethod
   def to_onehotvec_label(index_labels, dim):
      num_labels = len(index_labels)
      onehotvec_labels = np.zeros((num_labels, dim))
      for i, idx in enumerate(index_labels):
         onehotvec_labels[i][idx] = 1.0
      onehotvec_labels_tf = tf.convert_to_tensor(onehotvec_labels)
      return onehotvec_labels_tf
```

Model Training

- fit() 함수와 data를 사용하여 모델을 Training해 보라.
  - Epoch 는 10 으로 한다.

Model Prediction

- main.py에서 ImageClassifier를 import하고,
  - from ImageClassifier import ImageClassifier
- Class ImageClassifier를 사용해서 model train 및 test(predict)를 구현해 보아라.
- Test(predict)할 때는 다음 코드를 참고한다.

```
predicted_labels = my_classifier.predict(test_imgs=test_images)
predicted_labels = tf.math.argmax(input=predicted_labels, axis=1)

plt.figure(figsize=(10, 10))
for i in range(25):
    plt.subplot(5, 5, i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(test_images[i], cmap=plt.cm.binary)
    plt.xlabel(class_names[predicted_labels[i]])
plt.show()
```

- tf.math.argmax가 어떤 동작을 하는지
- 특히 input parameter “axis”의 값이 무엇을 의미하는지,
- https://www.tensorflow.org/api_docs/python/tf/math/argmax
- 를 찾아보고 실제 debugger을 통해 결과를 확인해 보라.

결과

- 잘 되었다면 다음과 같은 분류 결과를 볼 수 있다.