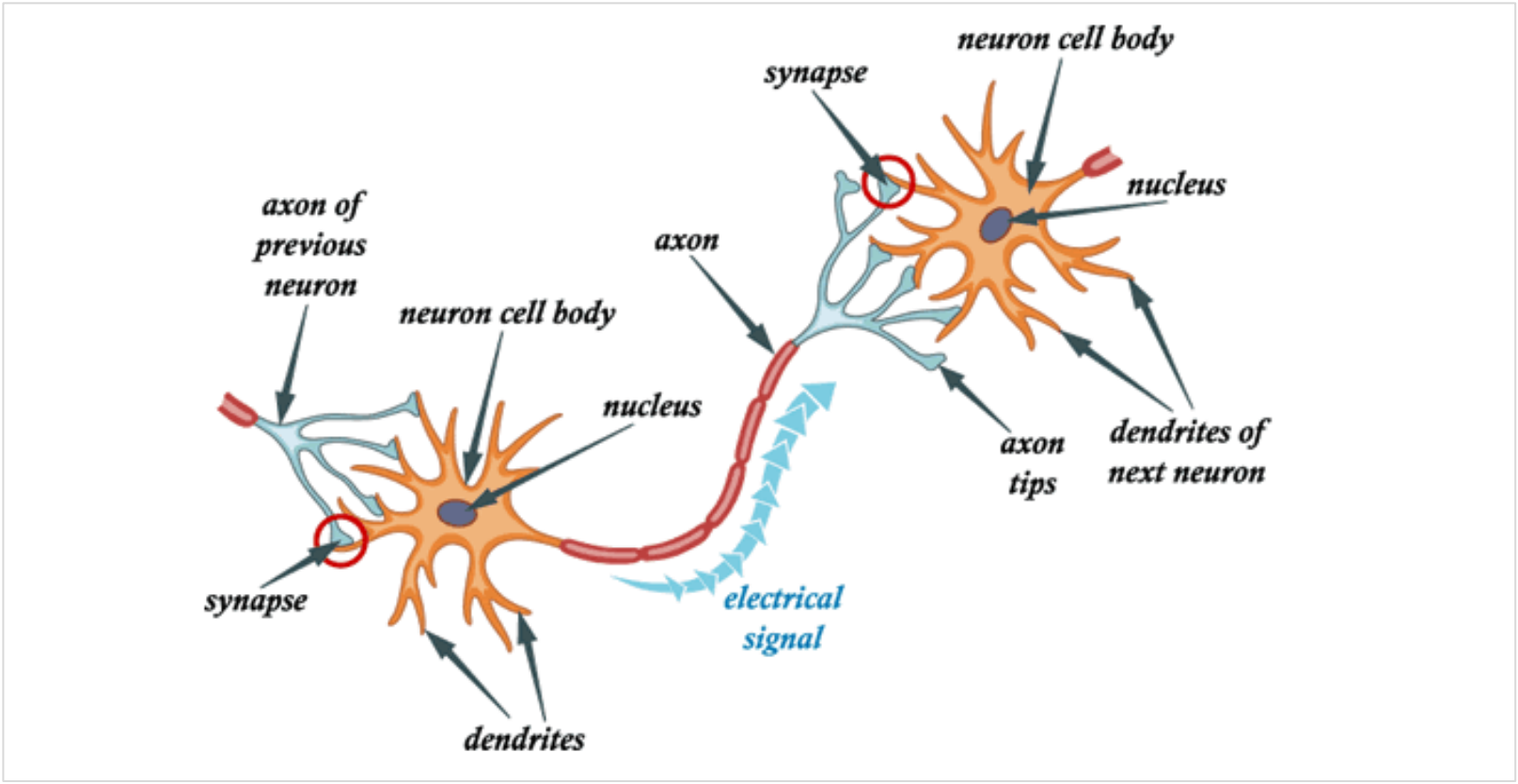
Machine Learning

Neural Network

김선녕(ksycafe@gmail.com)

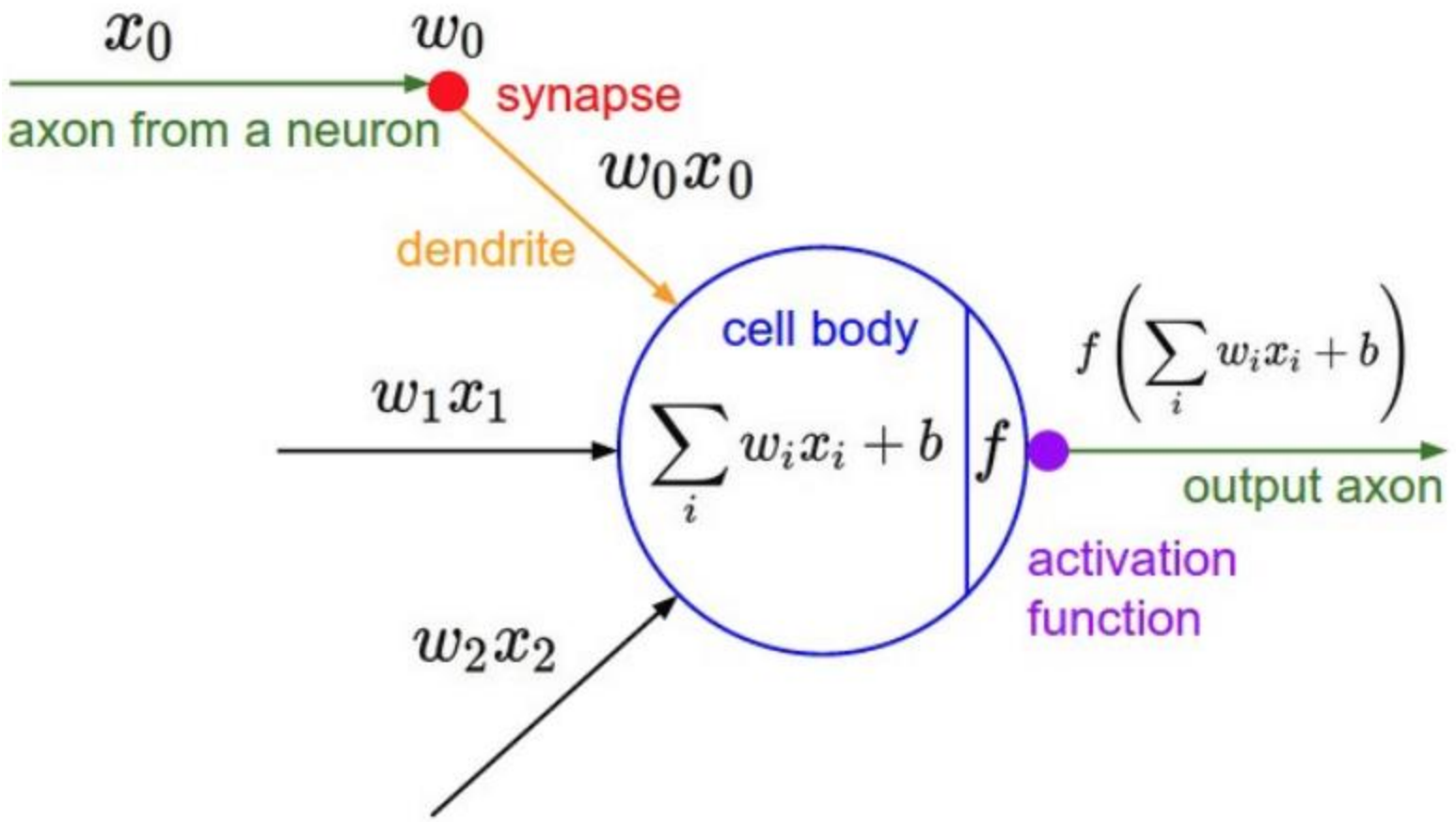
Biological neuron

- 인간의 뇌는 1000억 개가 넘는 뉴런이 100조 개 이상의 시냅스 ($synapse$) 를 통해 병렬적으로 연결되어 있다고 한다.
- 뉴런은 수상돌기($dendrite$)를 통해 다른 뉴런에서 입력 신호를 받아서 축색돌기($axon$)를 통해 다른 뉴런으로 신호를 내보낸다.
- 시냅스($synapse$) : 뉴런과 뉴런을 연결하는 역할



https://medium.com/autonomous-agents/mathematical-foundation-for-activation-functions-in-artificial-neural-networks-a51c9dd7c089

Mathematical Model



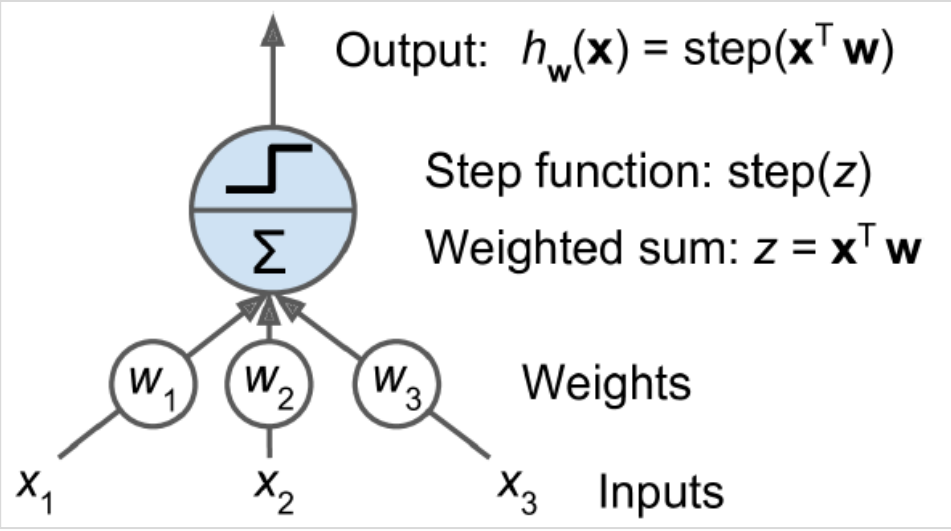
http://cs231n.github.io/neural-networks-1/

퍼셉트론(Perceptron)

- 1957년 프랭크 로젠블랫(Frank Rosenblatt)에 의해 고안된 알고리즘
- TLU(threshold logic unit) 혹은 LTU(linear threshold unit)이라고도 불린다
- 다수의 신호($Input: x_1, x_2 \cdots$)을 입력 받아서 계단함수를 적용하여 결과 신호($Output: y$)를 출력

$$w_1 x_1 + w_2 x_2 + \cdots + w_n x_n = \sum_{i=1}^{n} w_i x_i = w^T x$$

- 가중치($weight$) : 특징(feature)이 레이블(label)의 예측에 끼치는 영향도. 값이 클수록 예측에 미치는 영향이 크다. 학습이 진행되면서 값이 변동
- 임계값($threshold$) : 뉴런에서 보낸 신호의 총합이 정해진 한계치. θ로 표시

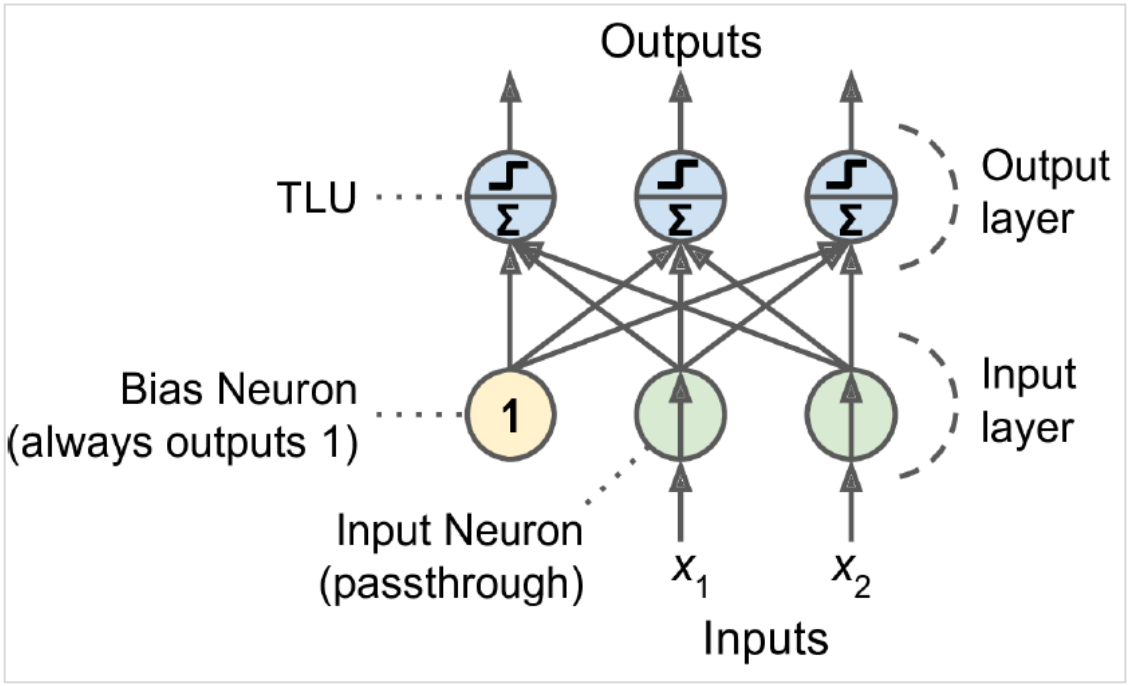


http://www.edubilla.com/inventor/frank-rosenblatt/

출력 계산

$$h_{w,b}(X) = \phi(WX + b)$$

- ϕ : 활성화 함수($activation\ function$). TLU인 경우 계단함수($step\ function$)



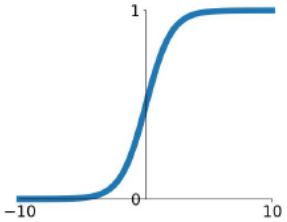
입력뉴런 두개, 편향 뉴런 한 개, 출력 뉴런 세개로 구성된 퍼셉트론 구조

활성화 함수

- 입력받은 신호를 적절한 처리를 하여 다음 뉴런(층)으로 출력하는 함수

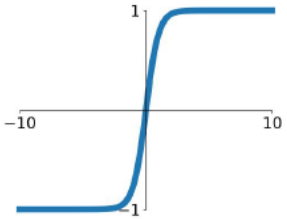
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



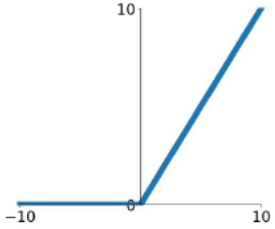
tanh

$$\tanh(x)$$



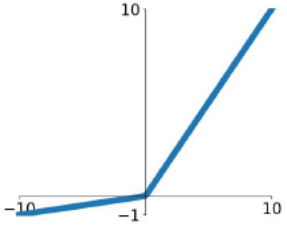
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

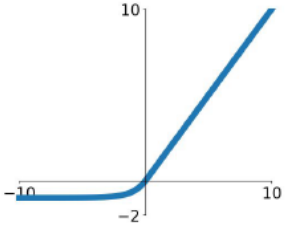


Maxout

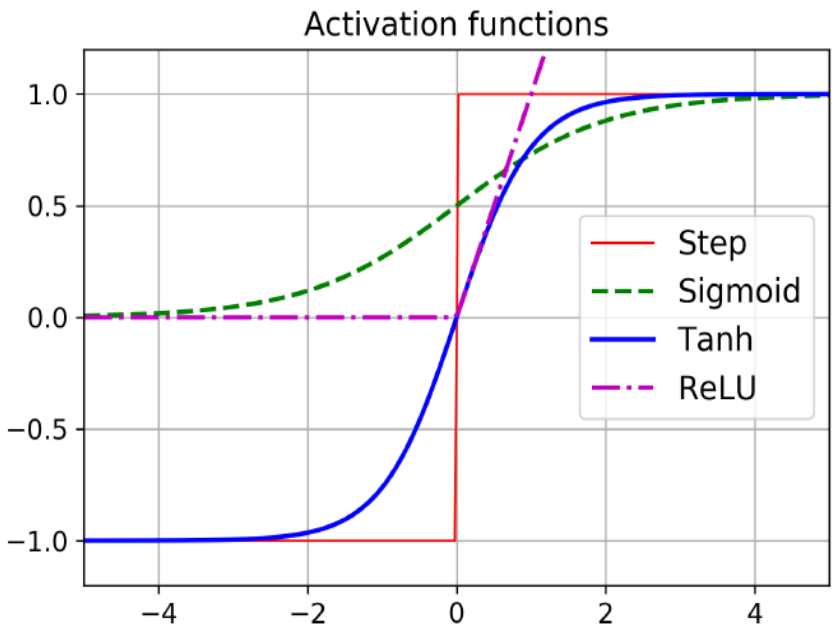
$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

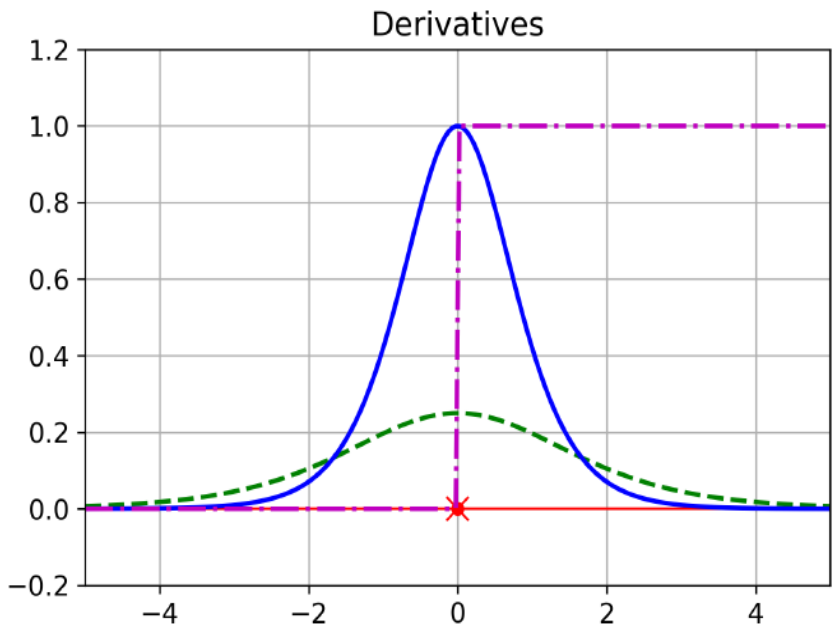
ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



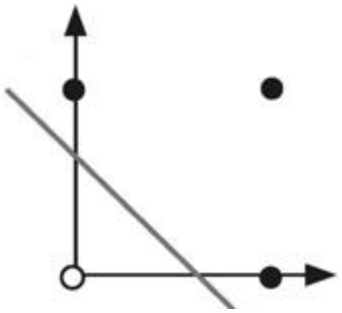
활성화 함수의 미분

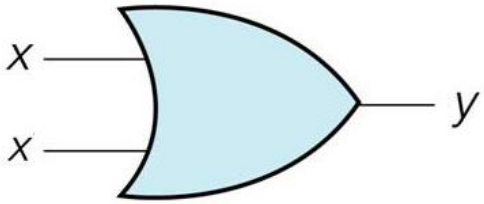

Activation functions
1.0
0.5
0.0
−0.5
−1.0
−4
−2
0
2
4
Step
Sigmoid
Tanh
ReLU


Derivatives
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−4
−2
0
2
4

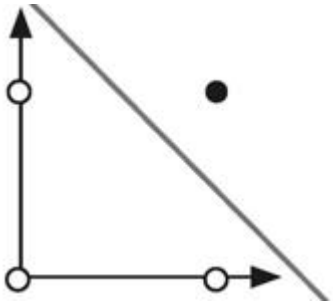
퍼셉트론 - XOR

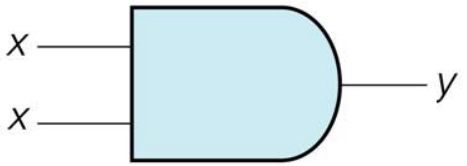
x	x	y
0	0	0
0	1	0
1	0	0
1	1	1



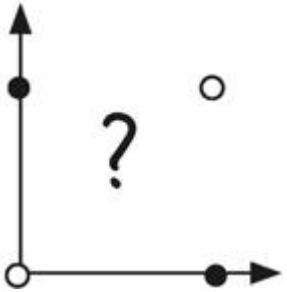


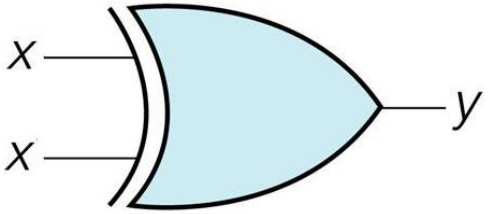
x	x	y
0	0	0
0	1	1
1	0	1
1	1	1



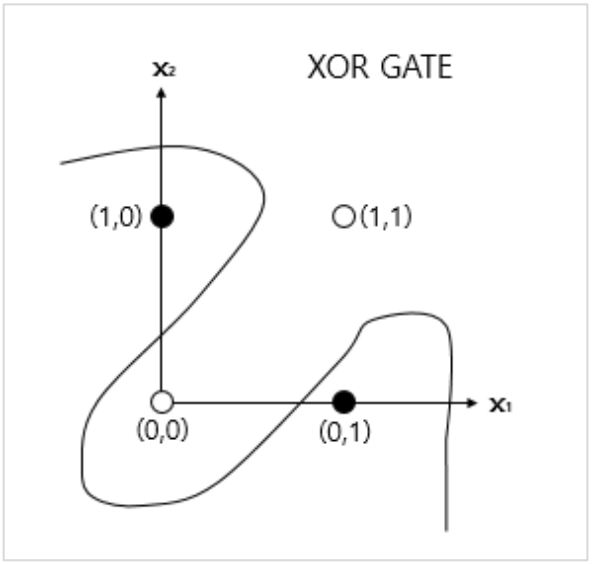


x	x	y
0	0	0
0	1	1
1	0	1
1	1	0

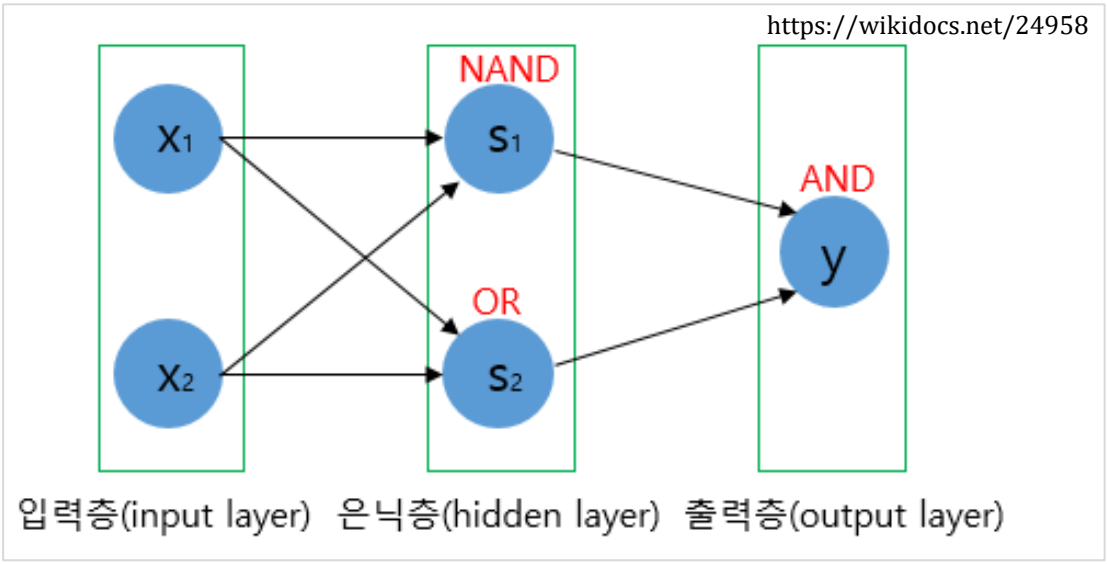




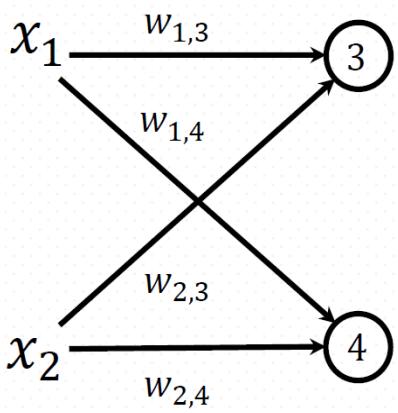
XOR



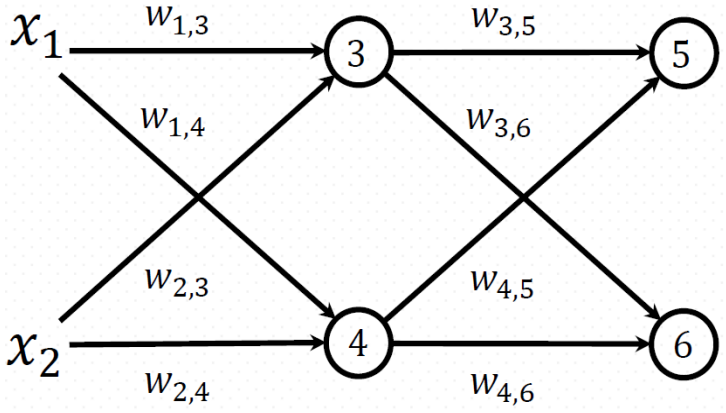
https://wikidocs.net/24958



Single Layer

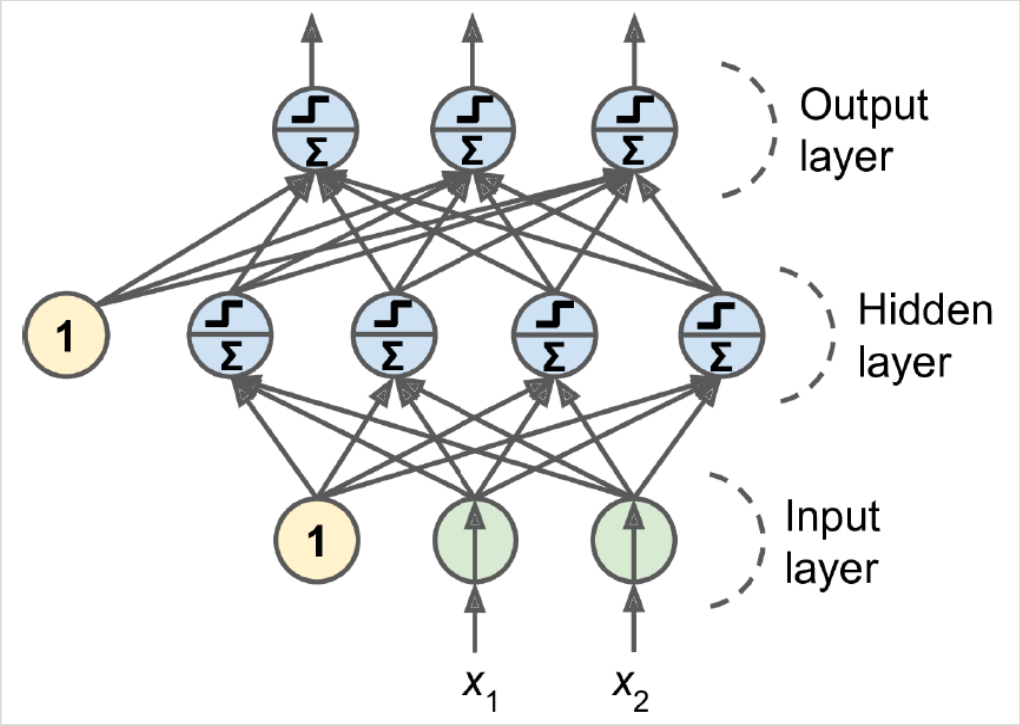


Multiple Layers



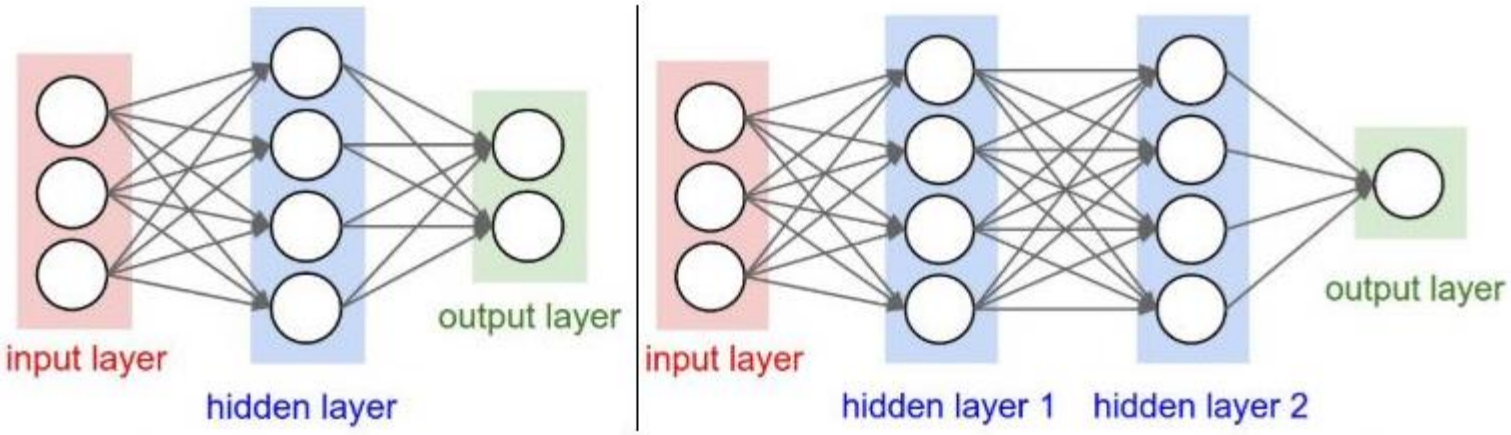
다층 퍼셉트론(MLP, Multilayer Perceptron)

- 입력층(input layer), 은닉층(hidden layer), 출력층(output layer)
- 출력층을 제외하고 모든 층은 편향 뉴런 포함
- 순전파 신경망(feedforward neural network) : 신호가 한방향으로 흐른다
- 심층 신경망(deep neural network, DNN) : 은닉층 여러 개인 망, 딥러닝



신경망 (Neural Networks) 구조

- 신경망의 크기를 측정하는 척도 : 뉴런의 수 혹은 parameter의 수
  - parameter : 뉴런과 뉴런의 연결된 부분의 weight or bias



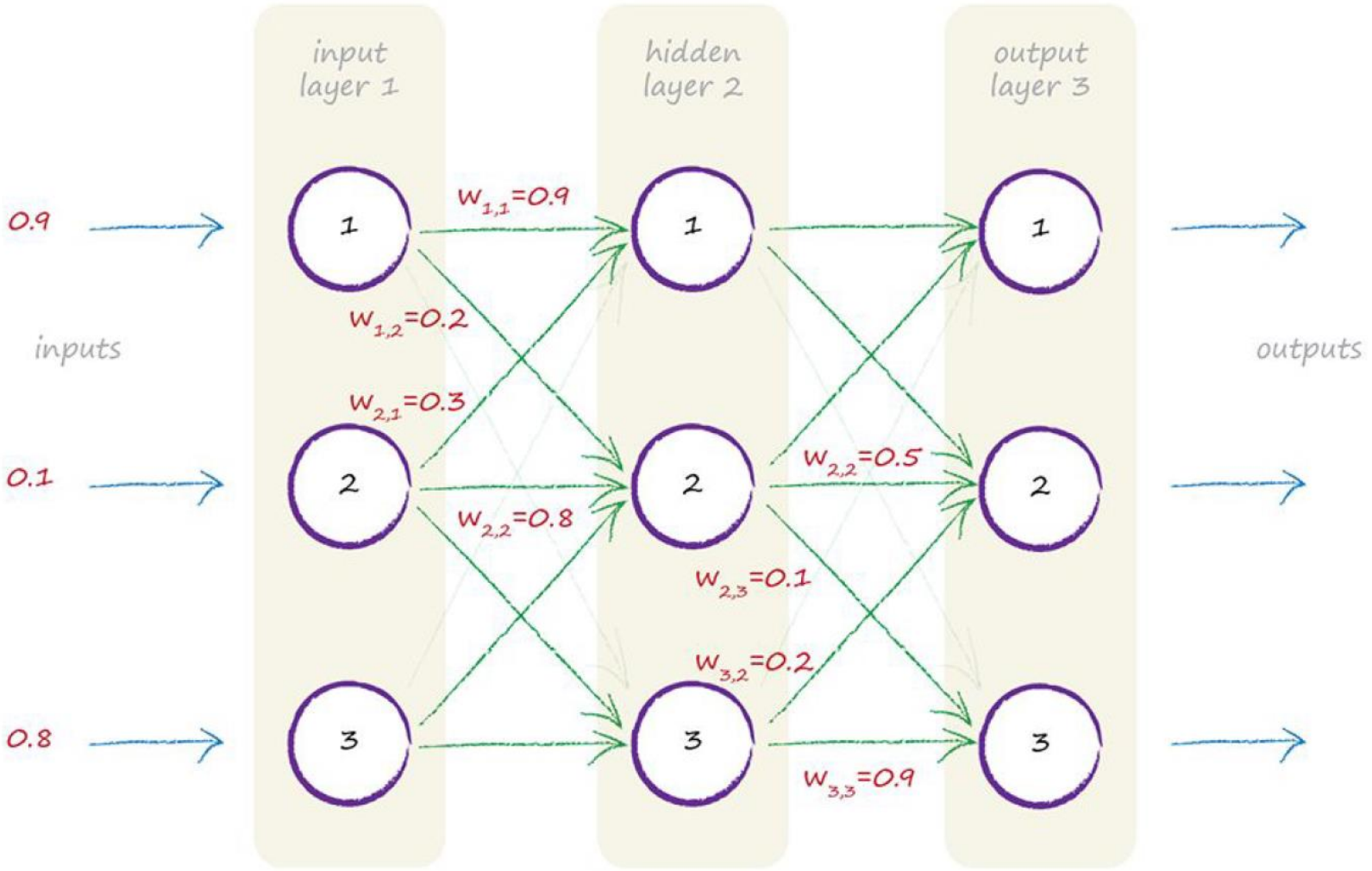
Left: A 2-layer Neural Network (one hidden layer of 4 neurons (or units) and one output layer with 2 neurons), and three inputs.
Right: A 3-layer neural network with three inputs, two hidden layers of 4 neurons each and one output layer. Notice that in both cases there are connections (synapses) between neurons across layers, but not within a layer.

- Left
  - 4+2 = 6개의 뉴런. [3x4] + [4x2] = 20 개의 weights와 4+2 = 6개의 biases. 총 26개의 parameters
- Right
  - 4+4+1 = 9개의 뉴런. [3x4] + [4x4]+[4x1] = 32개의 weights와 4+4+1의 biases. 총 41개의 parameters

- 대략 10~20개의 층이 있는 신경망의 parameters의 수는?

http://cs231n.github.io/neural-networks-1/

3계층 신경망에 행렬곱 적용예



ref : Make your own neural network - Triq Rashid

입력값

$$I = \begin{pmatrix} 0.9 \\ 0.1 \\ 0.8 \end{pmatrix}$$

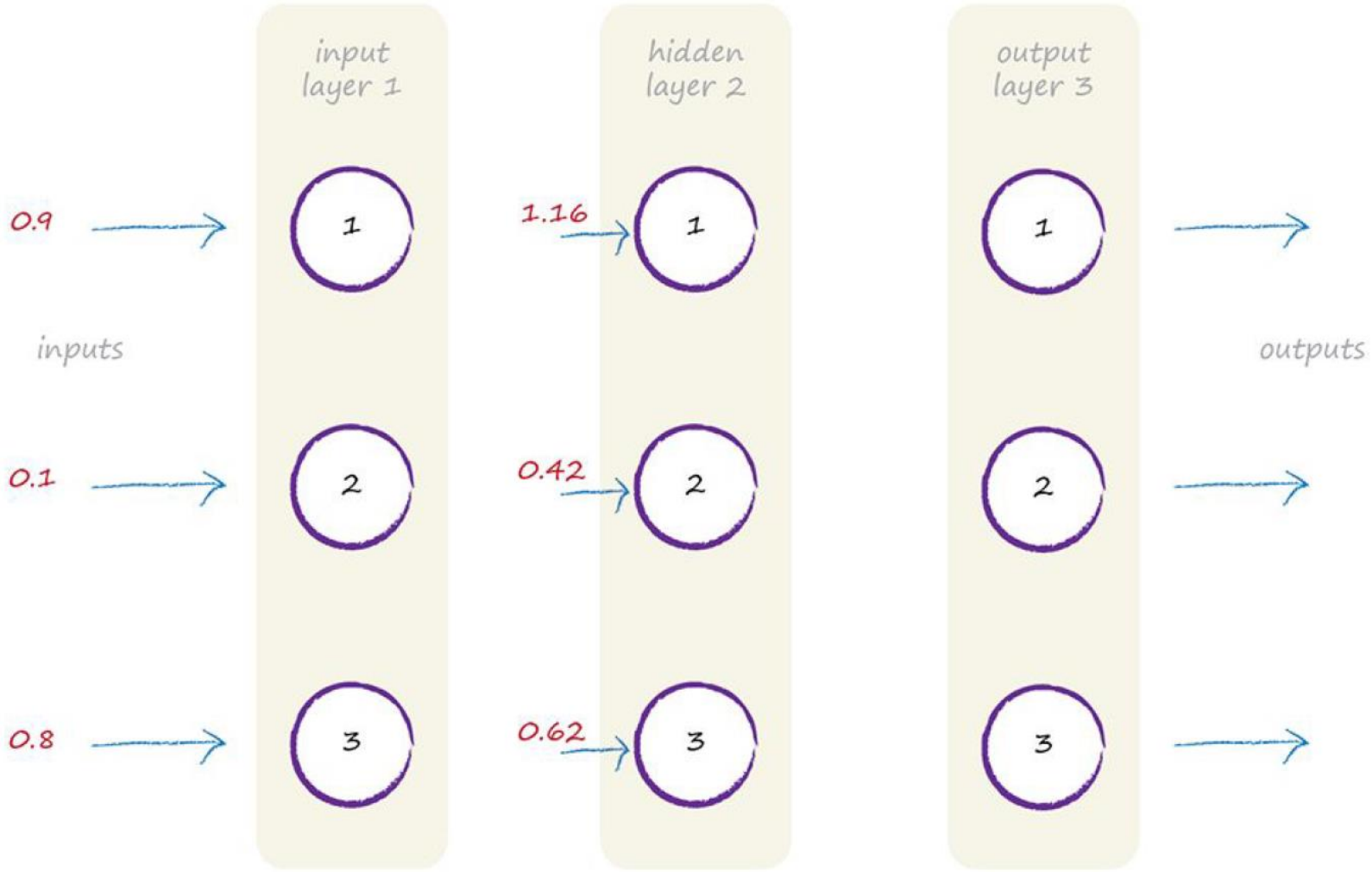
$$W_{i\_hidden} = \begin{pmatrix} 0.9 & 0.3 & 0.4 \\ 0.2 & 0.8 & 0.2 \\ 0.1 & 0.5 & 0.6 \end{pmatrix}$$

$$W_{o\_hidden} = \begin{pmatrix} 0.3 & 0.7 & 0.5 \\ 0.6 & 0.5 & 0.2 \\ 0.8 & 0.1 & 0.9 \end{pmatrix}$$

$$X_{hidden} = \begin{pmatrix} 0.9 & 0.3 & 0.4 \\ 0.2 & 0.8 & 0.2 \\ 0.1 & 0.5 & 0.6 \end{pmatrix} \begin{pmatrix} 0.9 \\ 0.1 \\ 0.8 \end{pmatrix} = \begin{pmatrix} 1.16 \\ 0.42 \\ 0.62 \end{pmatrix}$$

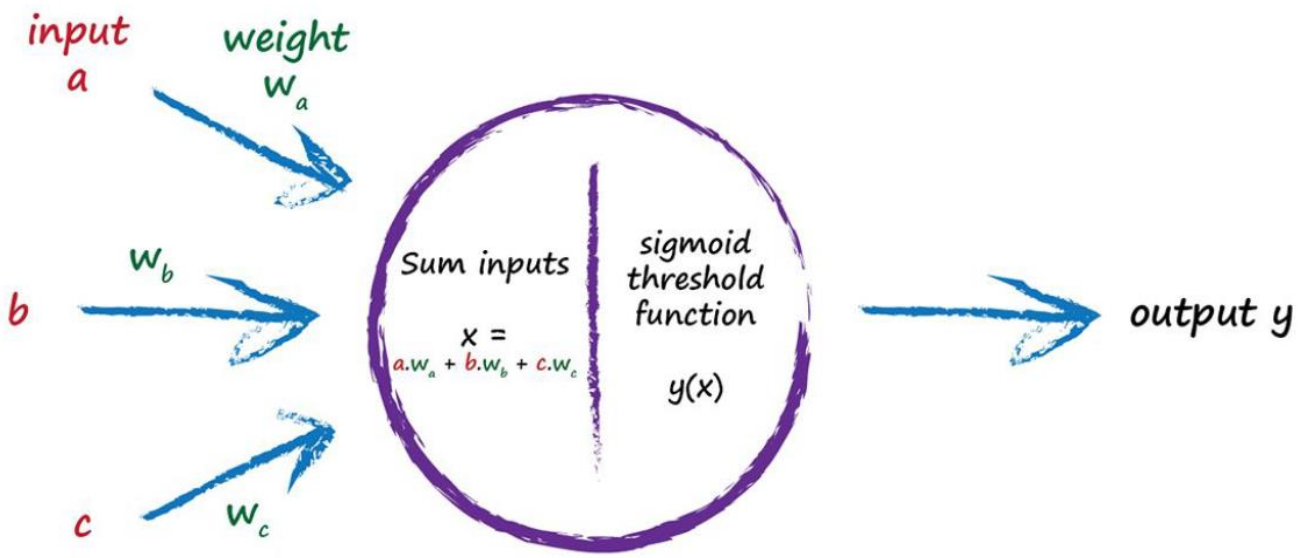
ref : Make your own neural network - Triq Rashid

Hidden Layer 입력값



ref : Make your own neural network - Triq Rashid

sigmoid function



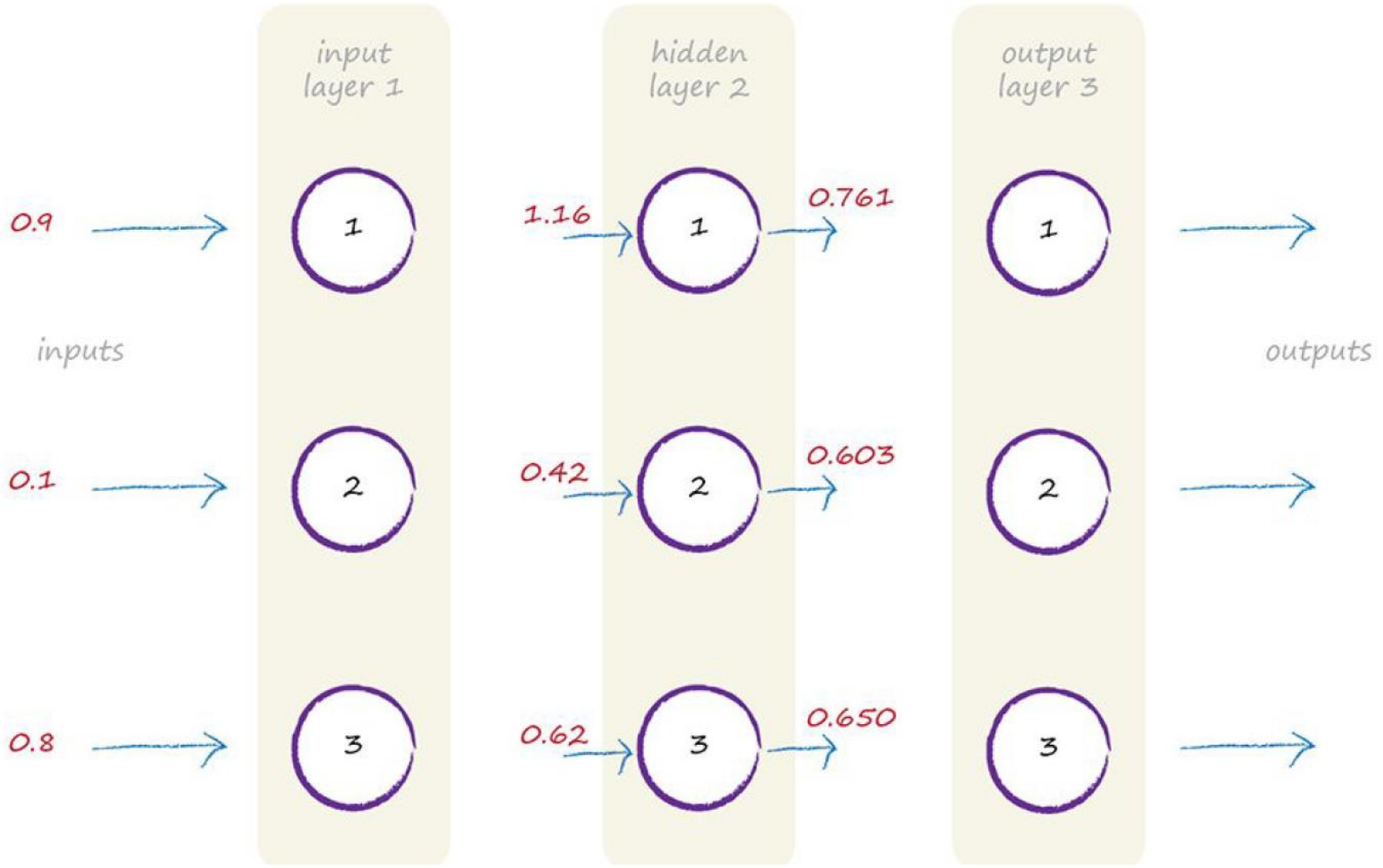
$$O_{hidden} = sigmoid(X_{hidden}) = \text{sigmoid}\begin{pmatrix}1.16\\0.42\\0.62\end{pmatrix} = \begin{pmatrix}0.761\\0.603\\0.650\end{pmatrix}$$

$$y = \frac{1}{1+e^{-x}} \quad , (x = 1.16 \text{ 대입하면 } e^{-x} = 0.3135, \qquad e = 2.71828 \ldots\ldots)$$

$$= \frac{1}{1+0.3135} = \mathbf{0.761}$$

ref : Make your own neural network - Triq Rashid

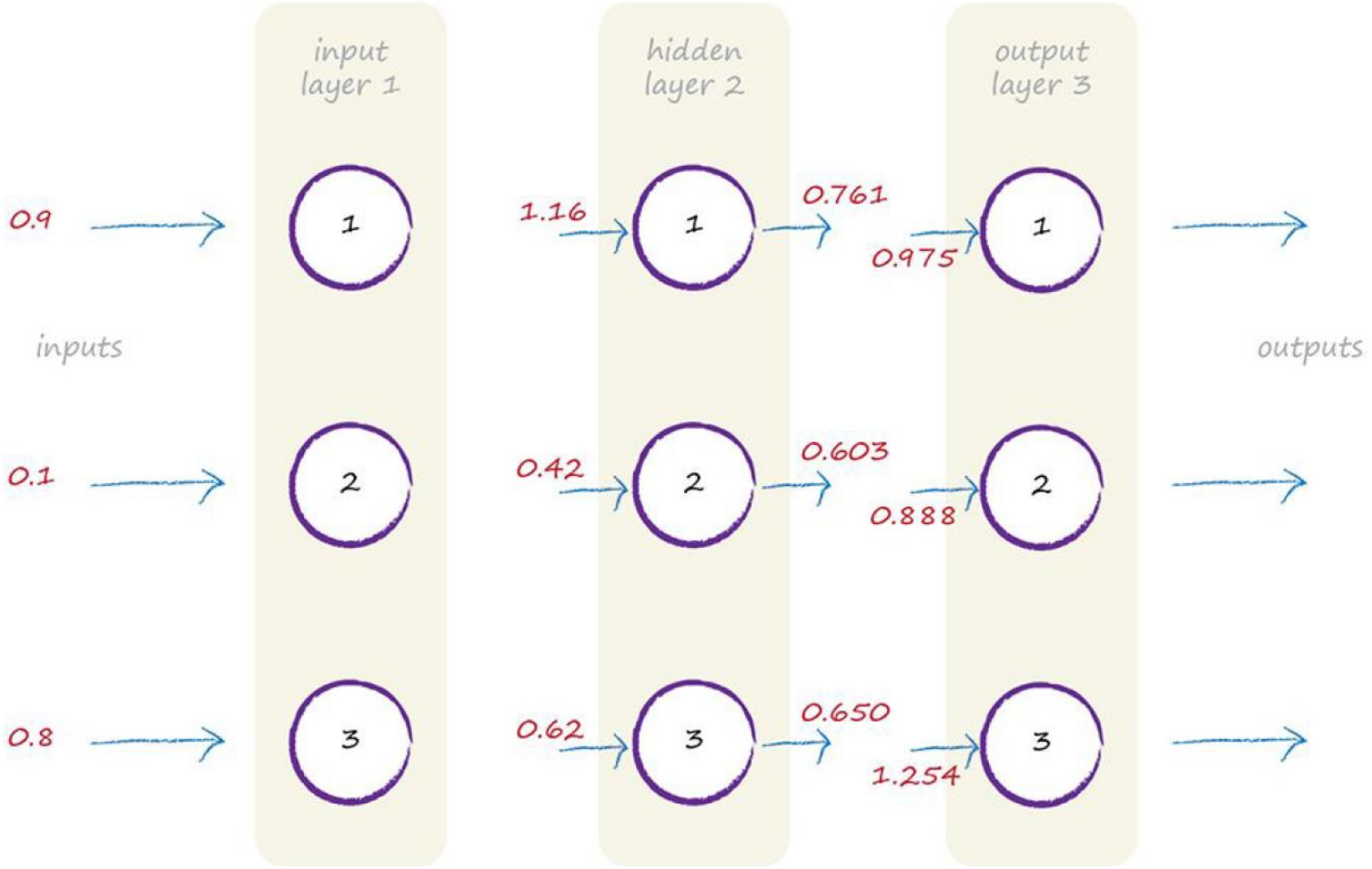
Hidden Layer결과값



ref : Make your own neural network - Triq Rashid

Ouput Layer입력

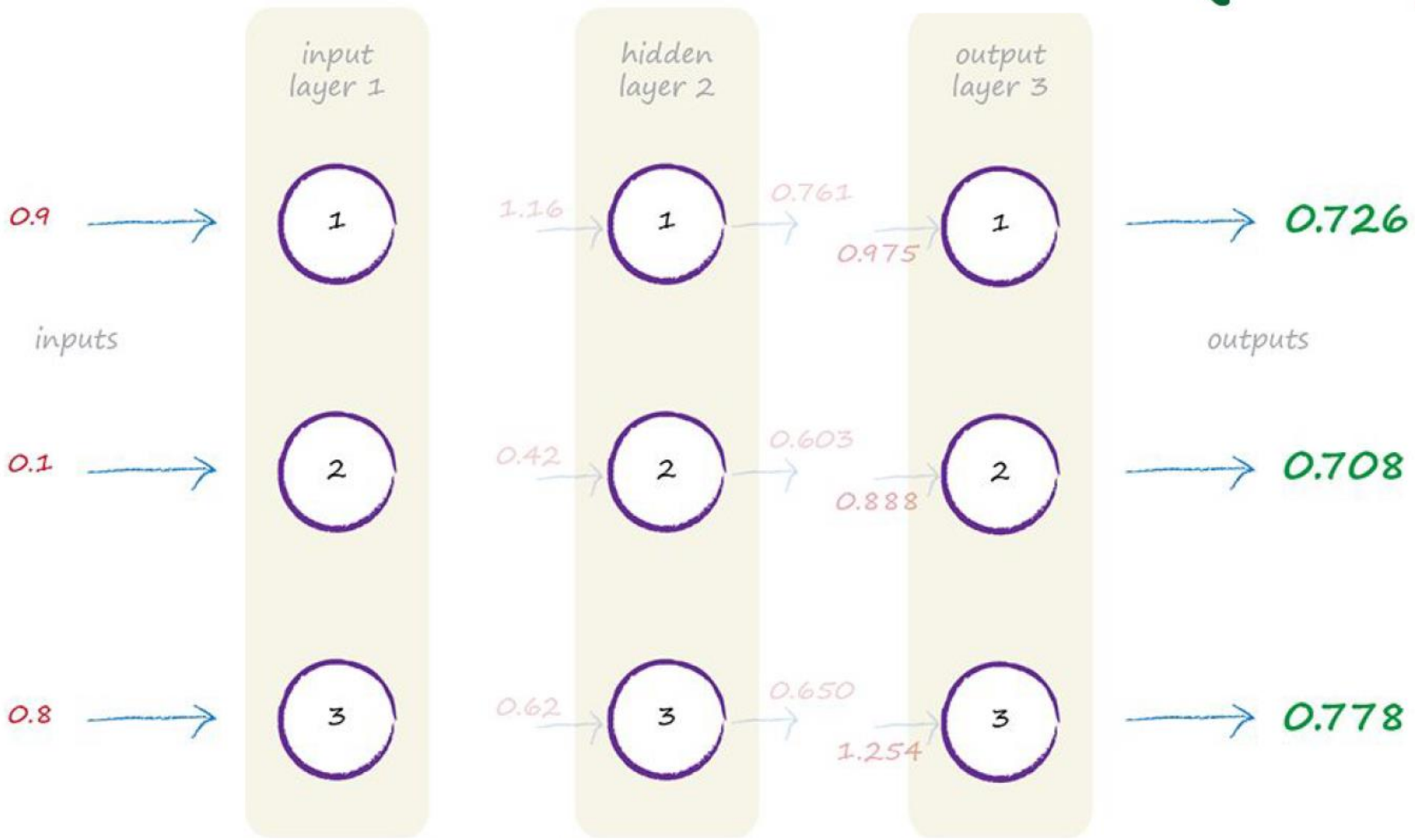
$$X_{output} = \begin{pmatrix} 0.3 & 0.7 & 0.5 \\ 0.6 & 0.5 & 0.2 \\ 0.8 & 0.1 & 0.9 \end{pmatrix} \cdot \begin{pmatrix} 0.761 \\ 0.603 \\ 0.650 \end{pmatrix} = \begin{pmatrix} 0.975 \\ 0.888 \\ 1.254 \end{pmatrix}$$



ref : Make your own neural network - Triq Rashid

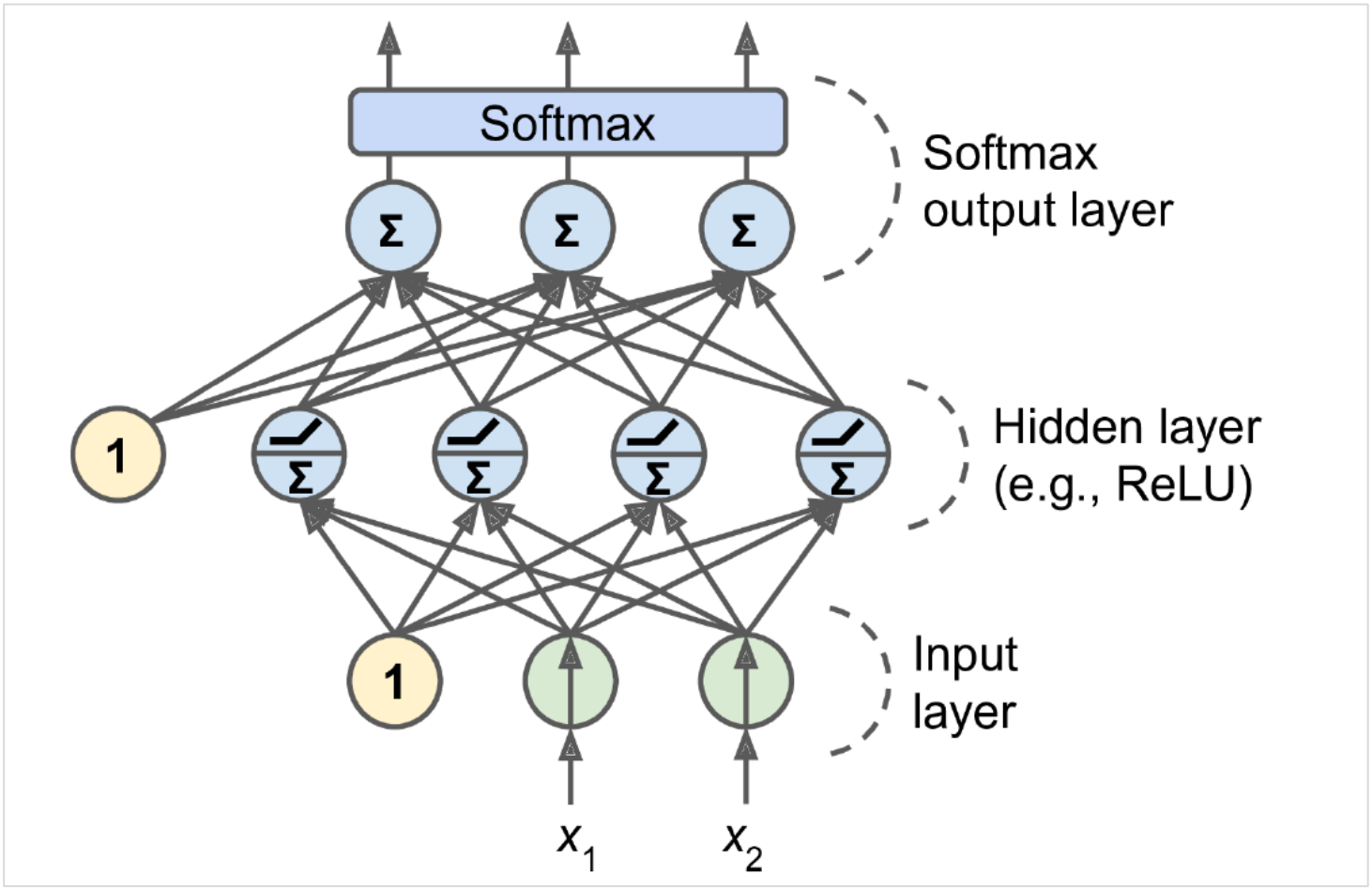
Output Layer결과값(최종결과값)

$$O_{output} = signmoid \begin{pmatrix} 0.975 \\ 0.888 \\ 1.254 \end{pmatrix}$$



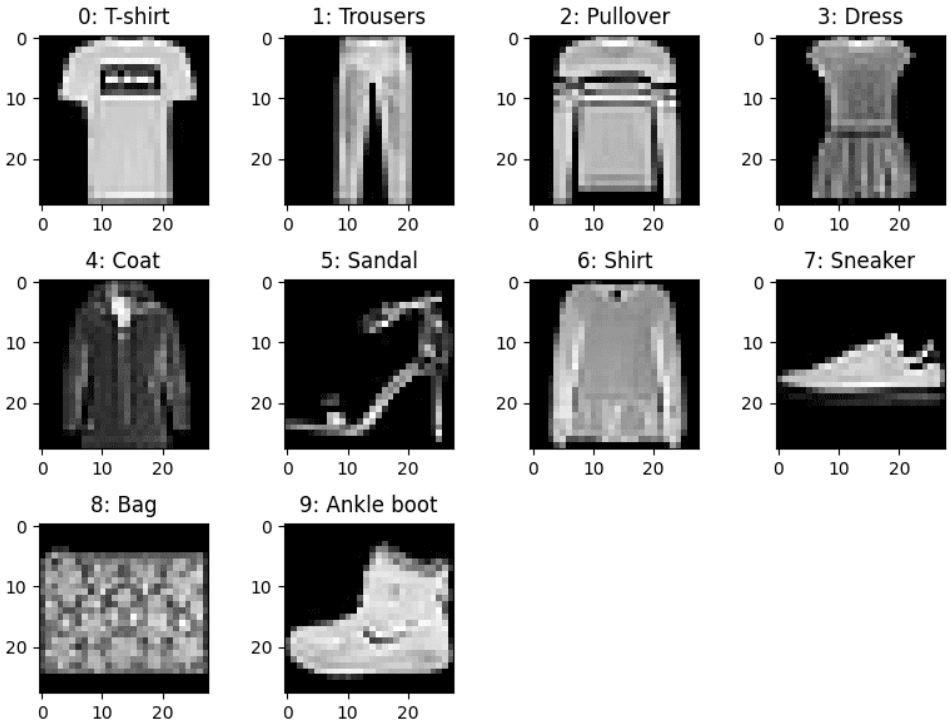
ref : Make your own neural network - Triq Rashid

분류를 위한 다층 퍼셉트론



텐서플로 - 케라스


```
y_train      # [4, 0, 7, ..., 3, 0, 5]
class_names = ["T-shirt/top", "Trouser", "Pullover", "Dress", "Coat",
               "Sandal", "Shirt", "Sneaker", "Bag", "Ankle boot"]
class_names[y_train[0]]    # Coat
# 검증 세트는 5,000개의 이미지
X_valid.shape    # (5000, 28, 28)
# 테스트 세트는 10,000개의 이미지
X_test.shape    #  (10000, 28, 28)
```



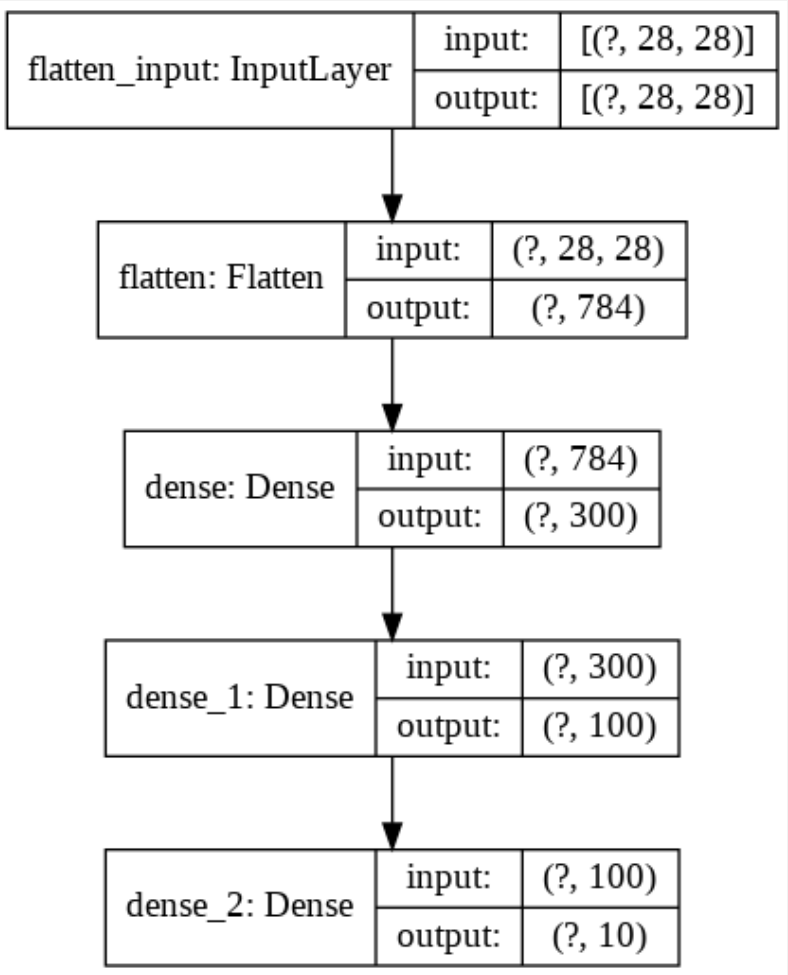
```
# 두 개의 은닉층으로 이루어진 분류용 다층 퍼셉트론
model = keras.models.Sequential()
model.add(keras.layers.Flatten(input_shape=[28, 28]))
model.add(keras.layers.Dense(300, activation="relu"))
model.add(keras.layers.Dense(100, activation="relu"))
model.add(keras.layers.Dense(10, activation="softmax"))

# 방법 2 - Sequential 모델에 층의 리스트를 전달
model = keras.models.Sequential([
    keras.layers.Flatten(input_shape=[28, 28]),
    keras.layers.Dense(300, activation="relu"),
    keras.layers.Dense(100, activation="relu"),
    keras.layers.Dense(10, activation="softmax")
])
```

summary()


```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)            (None, 784)               0
_________________________________________________________________
dense (Dense)                (None, 300)               235500
_________________________________________________________________
dense_1 (Dense)              (None, 100)               30100
_________________________________________________________________
dense_2 (Dense)              (None, 10)                1010
=================================================================
Total params: 266,610
Trainable params: 266,610
Non-trainable params: 0
_________________________________________________________________
```



```
# 손실함수와 최적화 지정
model.compile(loss="sparse_categorical_crossentropy",
              optimizer="sgd",
              metrics=["accuracy"])

# 모델 훈련 : epochs = 30
history = model.fit(X_train, y_train, epochs=30,
                    validation_data=(X_valid, y_valid))

# fit()메서드가 반환하는 훈련 파라미터
history.params

# 수행된epoch 리스트
print(history.epoch)

# 훈련세트와 검증 세트에 대한 손실과 측정한 지표를 담은 딕셔너리
history.history.keys()
```

```
model.evaluate(X_test, y_test)
```

313/313 [==============================] - 0s 1ms/step - **loss: 0.3382 - accuracy: 0.8822**
[0.3381877839565277, 0.8822000026702881]

```
# 예측 : 테스트 세트의 3개 샘플 사용
X_new = X_test[:3]
y_proba = model.predict(X_new)
y_proba.round(2)
```

array([[0. , 0. , 0. , 0. , 0. , 0.01, 0. , 0.03, 0. , **0.96**],
[0. , 0. , **0.99**, 0. , 0.01, 0. , 0. , 0. , 0. , 0.],
[0. , **1**. , 0. , 0. , 0. , 0. , 0. , 0. , 0. , 0.]],
dtype=float32)

```
y_pred = model.predict_classes(X_new)
y_pred  # array([9, 2, 1])
np.array(class_names)[y_pred]
```

array(['Ankle boot', 'Pullover', 'Trouser'], dtype='<U11')

```
y_new = y_test[:3]
y_new    # array([9, 2, 1], dtype=uint8)
```

```
plt.figure(figsize=(7.2, 2.4))
for index, image in enumerate(X_new):
    plt.subplot(1, 3, index + 1)
    plt.imshow(image, cmap="binary", interpolation="nearest")
    plt.axis('off')
    plt.title(class_names[y_test[index]], fontsize=12)
plt.subplots_adjust(wspace=0.2, hspace=0.5)
plt.show()
```