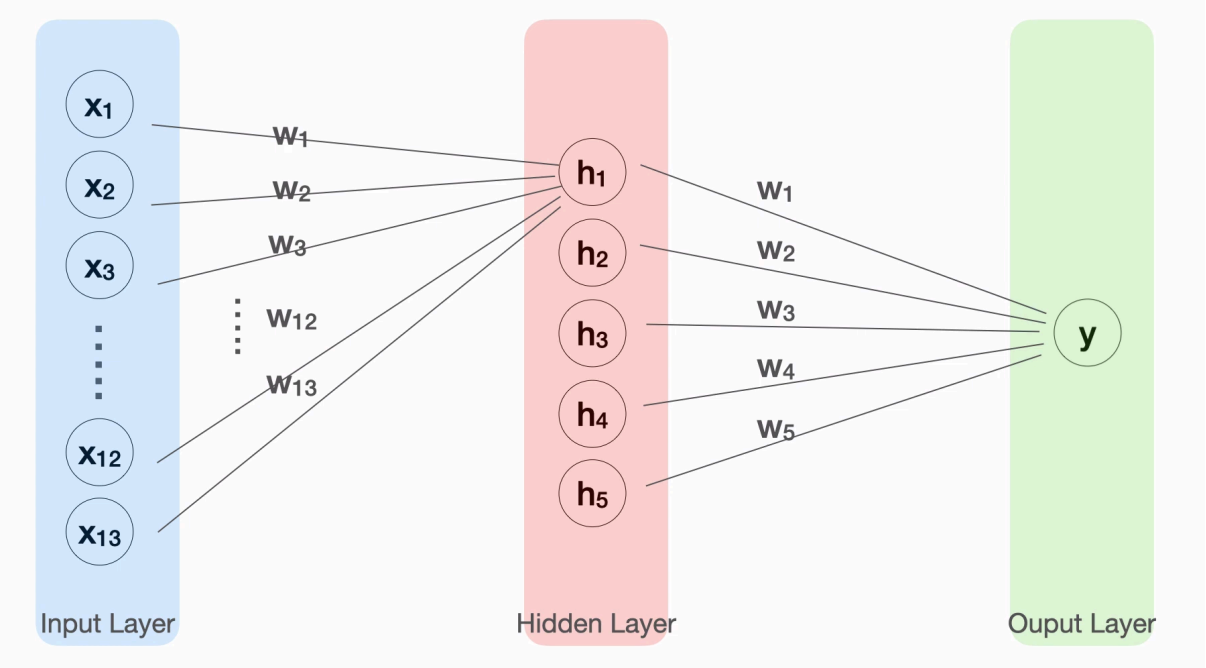
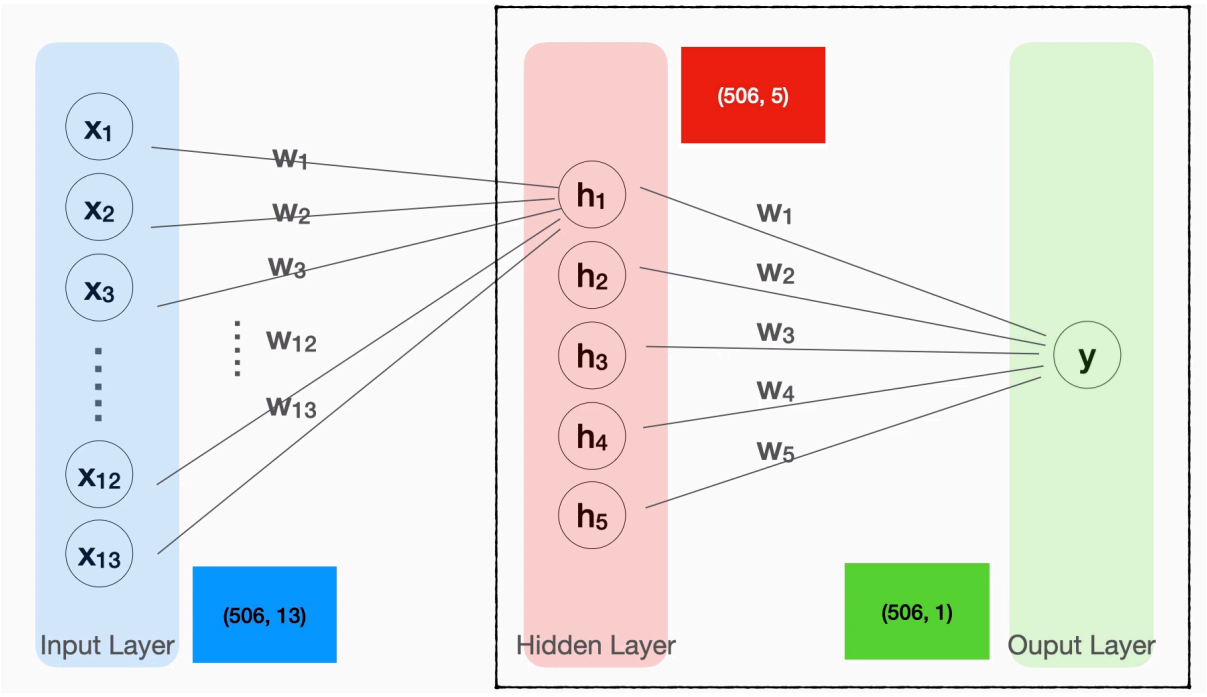
히든레이어

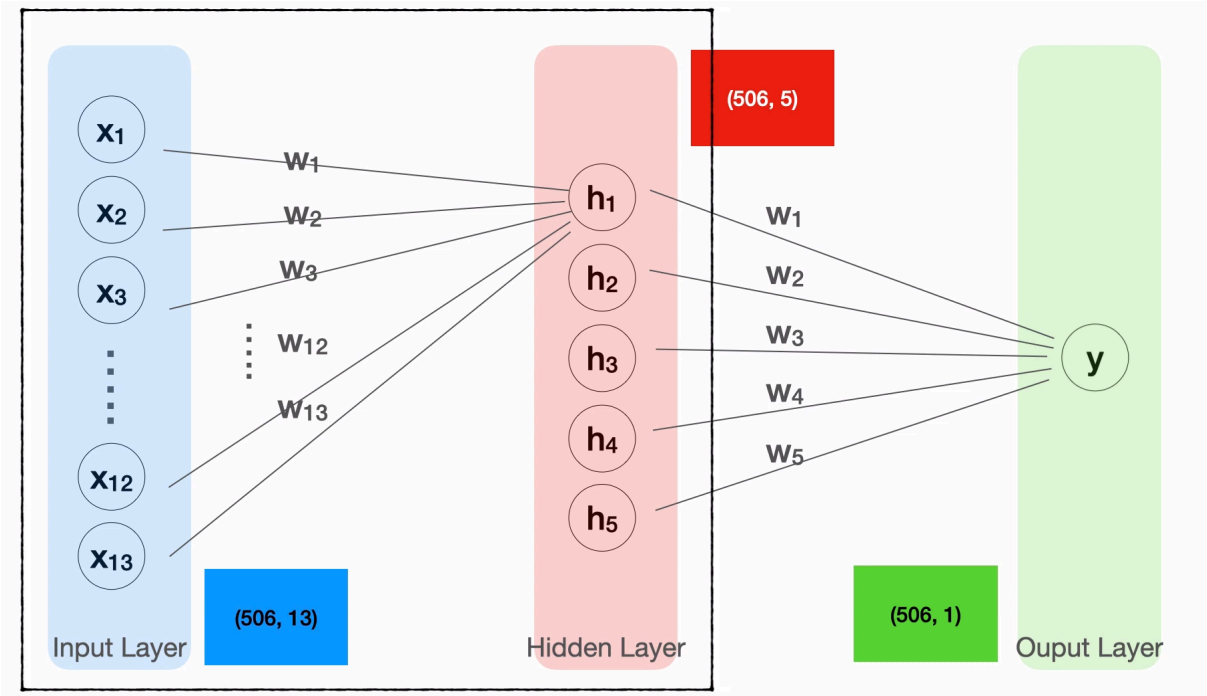
이번이 딥러닝을 완성하기 위한 마지막 남은 한조각입니다. 퍼셉트론 하나로 구성된 모델말고, 이제 퍼셉트론을 깊게 연결한 진짜 신경망, 딥러닝 모델을 만들어 봅시다. 여러분은 이미 깊은 신경망을 만들기 위한 모든 준비를 마쳤습니다. 신경망을 깊게 만드는 것은 의외로 단순합니다. 기존의 퍼셉트론을 여러개 사용하여 연결만 하면 됩니다. 입력과 결과 사이에 퍼셉트론을 추가해 주면 됩니다. 입력 부분을 input layer라고 하고 출력 부분을 output layer라고 하는데요. 그사이 추가한 부분을 hidden layer 라고 부릅니다.



지금 우리는 input 과 output 사이에 하나의 층을 쌓아서 모델을 구성한 것입니다. 우리가 쌓은 layer는 5개의 노드를 가지고 있습니다. 결과를 만들기 위해서는 hidden layer의 모든 값들을 입력을 하는 하나의 퍼셉트론이 필요합니다. 그리고 hidden layer의 첫번째 결과를 만들기 위해서는 역시나 하나의 퍼셉트론이 필요합니다. 총 5개의 노드가 있으니 5개의 퍼셉트론이 필요합니다.
이번에는 데이터의 관점에서 이그림을 바라보겠습니다. 보스턴 집값 데이터를 상상해 보시면 좋겠습니다. 관측치는 총 506개였습니다. 독립변수로 13개의 컬럼을 가진 데이터를 입력을 하고 있었고, 종속변수로 1개의 컬럼을 정답으로 가지고 있었습니다. 같은 표현을 가지고 중간 레이어를 바라보면 hidden layer에서 나타날 데이터를 추론해 볼 수 있습니다. 5개의 컬럼을 가진 데이터이구요.
마지막 hidden layer에서 결과를 출력하는 부분만 떼어서 관찰해보면 5개의 입력을 받고 하나의 출력을 만드는 모델이라고 생각할 수 있습니다.



입력 쪽에서 hidden layer 를 만드는 부분을 떼어서 관찰해보면 13개의 입력을 받고 5개의 출력을 만든 모델이라고 할 수 있습니다.



이렇게 각각의 모델을 연속적으로 연결하여 하나의 거대한 신경망을 만드는 것이 딥러닝, 인공신경망입니다. 이해를 돕기위해 그림을 사용하여 설명을 드렸습니다. 그런데 이런 구조를 만들려면 상당히 복잡하지는 않을까? 그런 생각이 드실 것 같아요. tensorflow는 이런 복잡한 구조를 간단하게 완성할 수 있게 도와줍니다. 기존 코드에서 조금전에 만든 hidden layer를 추가하는 코드는 이렇습니다.

```
X = tf.keras.layers.Input(shape=[13])
H = tf.keras.layers.Dense(5, activation='swish')(X) <= 이곳에 추가
Y = tf.keras.layers.Dense(1)(H) <= 여기에 H 입력
model = tf.keras.models.Model(X, Y)
model.compile(loss='mse')
```

주의해야 할 것은 마지막 출력을 만들때 X를 넣으면 안되고, 마지막 hidden layer에서 만들어진 H를 넣어 주어야 합니다. 지금 추가된 코드는 방금 그림으로 보신 그 모델, 중간 결과를 5개 내놓는 레이어를 하나 추가하는 코드입니다. hidden layer의 활성화 함수로는 swish를 사용했다는 것에 주의해 주시면 됩니다. 다른 활성화 함수들을 사용 할 수도 있는데요. swish는 7년전(2017)에 발표된 성능이 좋은 활성화 함수이니 우리는 이것을 사용하겠습니다. 이 hidden layer는 모델을 구성하는 여러분이 자유롭게 추가 해주면 되는데요. hidden layer를 두층 더 쌓아 봅시다. hidden layer를3개 사용한 모델입니다.

```
X = tf.keras.layers.Input(shape=[13])
H = tf.keras.layers.Dense(5, activation='swish')(X) <= 이곳에 추가
H = tf.keras.layers.Dense(3, activation='swish')(X) <= 이곳에 추가
H = tf.keras.layers.Dense(3, activation='swish')(X) <= 이곳에 추가
Y = tf.keras.layers.Dense(1)(H) <= 여기에 H 입력
model = tf.keras.models.Model(X, Y)
model.compile(loss='mse')
```

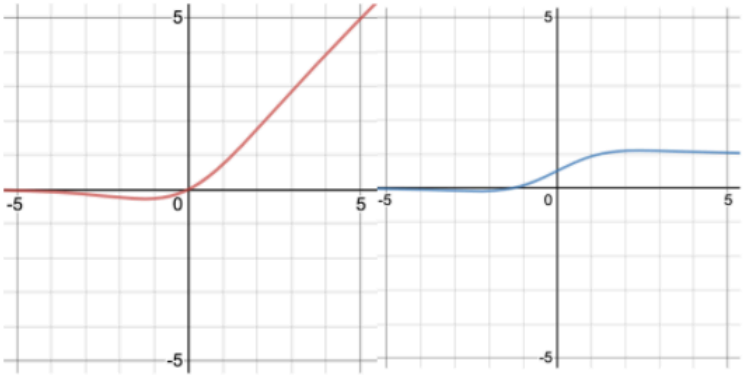
앞에 레이어 하나는 5개의 노드, 뒤에 두 개의 layer는 3개의 노드를 가진 모델이 되는 것입니다. 그리고 최종 1개의 출력을 만들게 되는 것이죠. 그전의 모델보다 훨씬 더 똑똑한 모델을 학습시킬 수 있게 됩니다. 이제 여러분은 딥러닝 모델을 만들 만반의 준비가 끝난 것입니다. 자 이제 딥러닝 모델을 만들어 가봅시다.

활성화 함수

Swish 활성화 함수를 통해 히든 레이어

Swish

$$f(x)=\frac{1}{1+e^{-x}}x \qquad f'(x)=f(x)+sigmoid(x)(1-f(x))$$



Swish ? 2017년에 발표된 성능이 좋은 활성화함수 중 하나. 특징 : ReLU를 대체하기 위해 구글이 고안한 함수. 시그모이드 함수에 X를 곱한 아주 간단한 형태를 보이지만, 깊은 레이어를 학습시킬 때 ReLU보다 더 뛰어난 성능을 보인다.

```python
# -*- coding: utf-8 -*-
import tensorflow as tf
import pandas as pd

파일경로 = './data/boston.csv'
보스턴 = pd.read_csv(파일경로)

독립 = 보스턴[['crim', 'zn', 'indus', 'chas', 'nox',
            'rm', 'age', 'dis', 'rad', 'tax',
            'ptratio', 'b', 'lstat']]
종속 = 보스턴[['medv']]
print(독립.shape, 종속.shape)

X = tf.keras.layers.Input(shape=[13])
H = tf.keras.layers.Dense(10, activation='swish')(X)
Y = tf.keras.layers.Dense(1)(H)
model = tf.keras.models.Model(X, Y)
model.compile(loss='mse')

model.summary()

model.fit(독립, 종속, epochs=1000, verbose=0)

model.fit(독립, 종속, epochs=10)

print(model.predict(독립[:5])) # 예측
print(종속[:5]) # 정답

파일경로 = './data/iris.csv'
아이리스 = pd.read_csv(파일경로)

아이리스 = pd.get_dummies(아이리스)

독립 = 아이리스[['꽃잎길이', '꽃잎폭', '꽃받침길이', '꽃받침폭']]
종속 = 아이리스[['품종_setosa', '품종_versicolor', '품종_virginica']]
print(독립.shape, 종속.shape)

X = tf.keras.layers.Input(shape=[4])
H = tf.keras.layers.Dense(8, activation="swish")(X)
H = tf.keras.layers.Dense(8, activation="swish")(H)
H = tf.keras.layers.Dense(8, activation="swish")(H)
Y = tf.keras.layers.Dense(3, activation='softmax')(H)
model = tf.keras.models.Model(X, Y)
model.compile(loss='categorical_crossentropy',
              metrics=['accuracy'])

model.summary()

model.fit(독립, 종속, epochs=100)

print(model.predict(독립[0:5]))
print(종속[0:5])
```