신경망 - 1
Neural Networks

이건명
충북대학교 소프트웨어학과

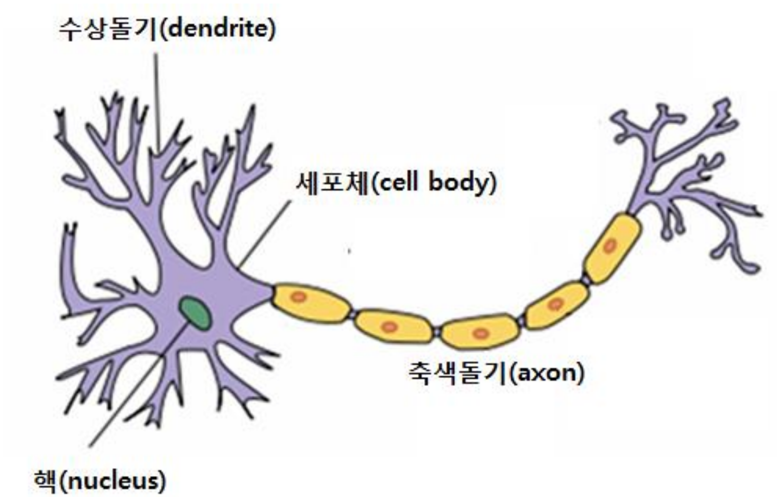
학습 내용

- 신경망을 구성하는 뉴런의 함수적 특성을 이해한다.
- 퍼셉트론 모델의 특성을 알아본다.
- 다층 퍼셉트론에 대해서 알아본다.
- 미분 방법에 대해서 복습한다.
- 다층 퍼셉트론의 학습 알고리즘인 오차 역전파 알고리즘에 대해 알아본다.

1. 신경망

❖ **신경망(neural network, artificial neural network)**

- 인간 두뇌에 대한 계산적 모델을 통해 인공지능을 구현하려는 분야
- **신경세포** (neuron)



신경세포 8.6×10^{10} 개

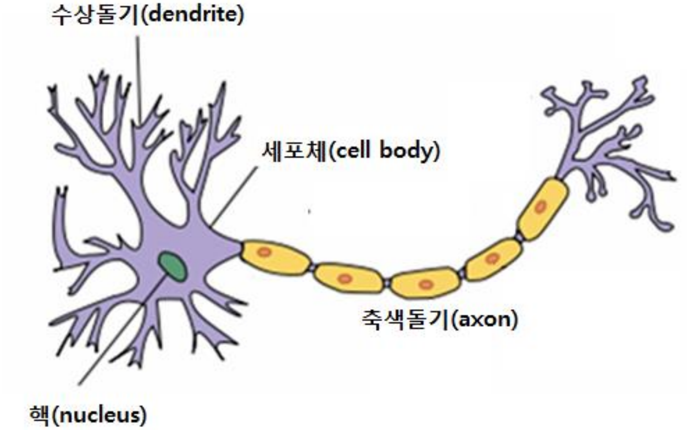
신경연접 1.5×10^{14} 개

- **수상돌기(**樹狀突起, dendrite) : 다른 신경세포의 축색돌기와 연결되어 **전기화학적 신호**를 받아들이는 부위
- **축색돌기**(軸索突起, axon) : 수신한 전기화학적 신호의 합성결과 값이 특정 임계값이 이상이면 신호를 내보는 부위.
- **신경연접**(神經連接, synapse) : 수상돌기와 축색돌기 연결 부위
  - 전달되는 신호의 증폭 또는 감쇄

2. 퍼셉트론

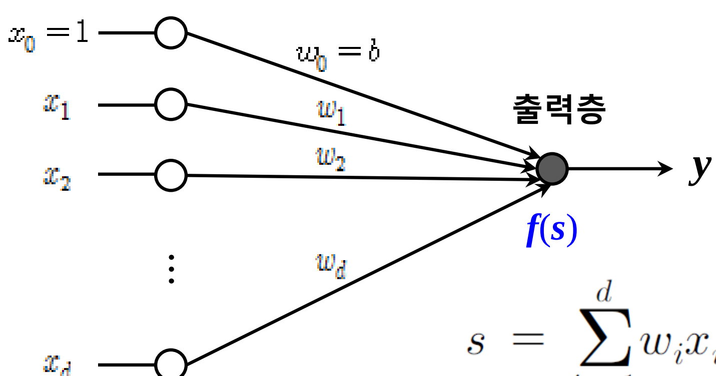
❖ 신경세포의 계산 모델

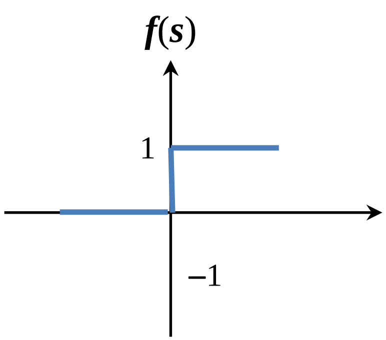
- 신경세포



McClulloch & Pitts, 1943
신경세포의 계산 모델

- **퍼셉트론**(Perceptron)
  - 로젠블랏(Rosenblatt, 1957)이 제안한 학습가능한 신경망 모델





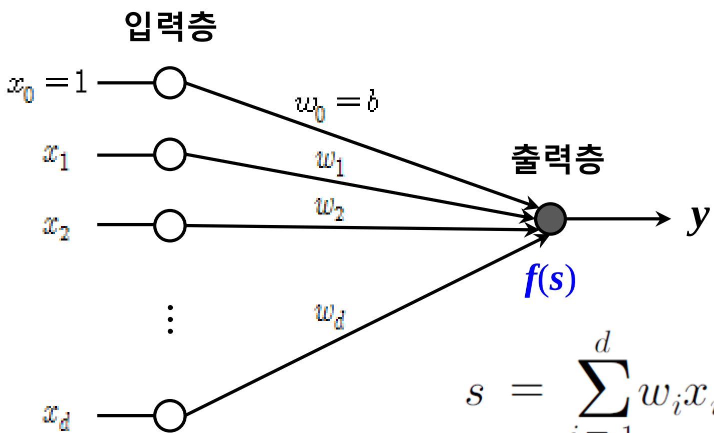
$$s = \sum_{i=1}^{d} w_i x_i + b = \sum_{i=0}^{d} w_i x_i$$

활성화 함수

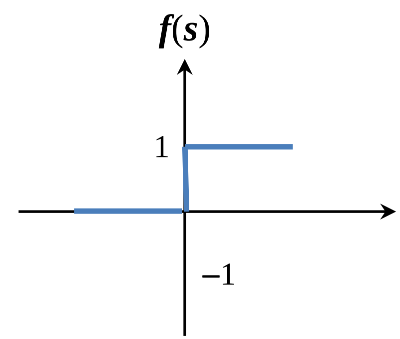
2. 퍼셉트론

❖ 신경세포의 계산 모델

- **퍼셉트론**(Perceptron)



$$s = \sum_{i=1}^{d} w_i x_i + b = \sum_{i=0}^{d} w_i x_i$$



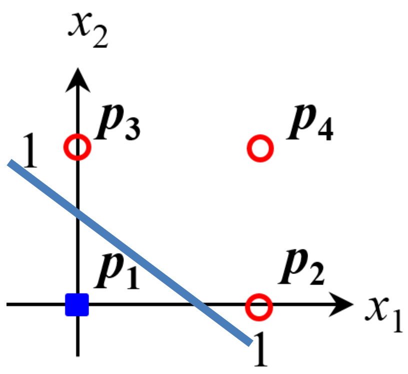
활성화 함수

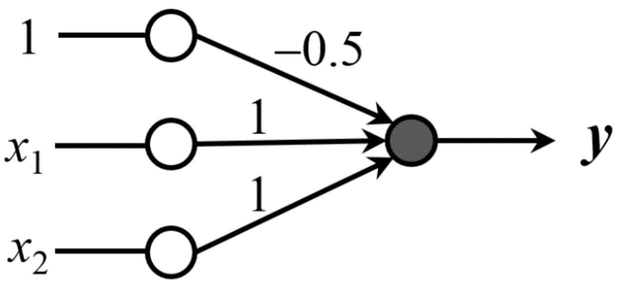
```
def Perceptron(inputs):
    sum = np.dot(inputs, weights[1:]) + weights[0]
    if sum > 0:
        activation = 1
    else:
        activation = 0
    return activation
```

퍼셉트론

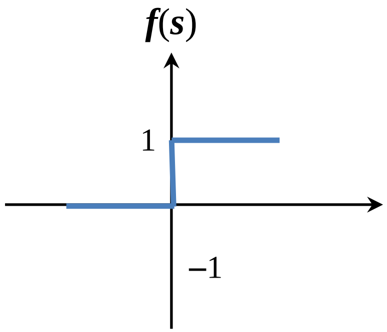
❖ **퍼셉트론**(Perceptron)

- **OR 연산을 수행하는 퍼셉트론**





$$y = f(s) = f(\sum_{i=1}^{2} w_i x_i + b) = f(\boldsymbol{w}^\top \boldsymbol{x}) = x_1 + x_2 - 0.5$$

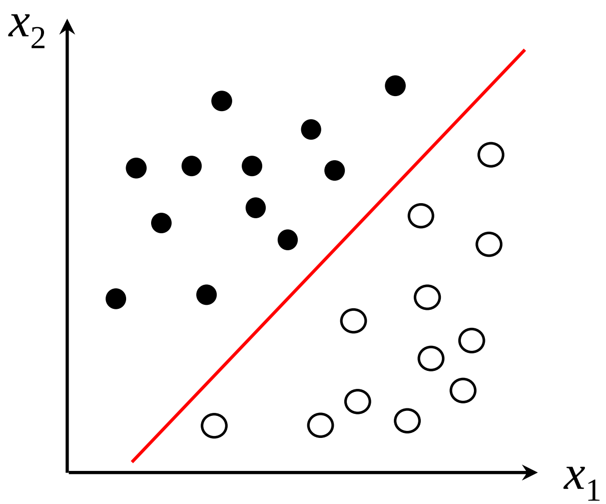


$p_1\ (0,0)$: $y = 0$
$p_2\ (1,0)$: $y = 1$
$p_3\ (0,1)$: $y = 1$
$p_4\ (1,1)$: $y = 1$

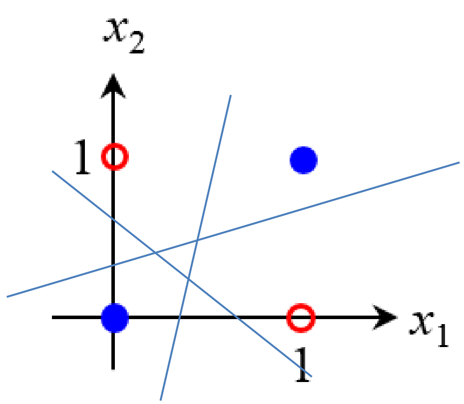
퍼셉트론

❖ **퍼셉트론**(Perceptron)

- **선형 분리가능 문제** (linearly separable problem)



- **선형 분리불가 문제**(linearly inseparable problem)
  - **XOR**(exclusive OR) **문제**

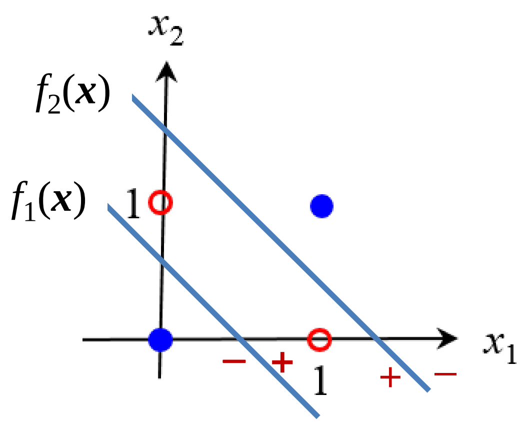


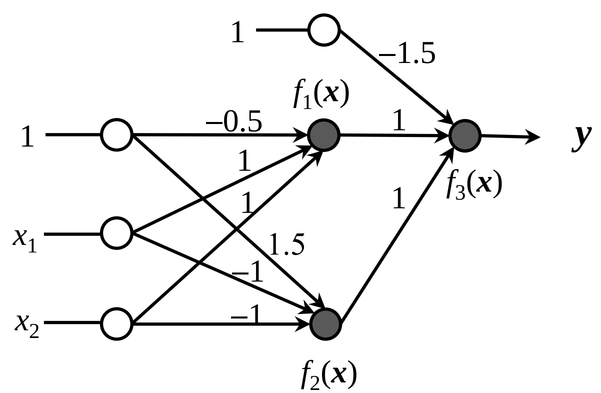
(0,0) : 0
(0,1) : 1
(1,0) : 1
(1,1) : 0

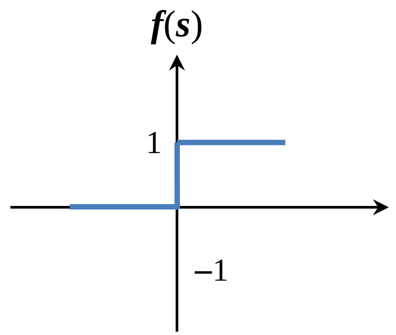
Minsky & Papert 1969

3. 다층 퍼셉트론

- ❖ **다층 퍼셉트론**(multilayer Perceptron, **MLP**)
  - 여러 개의 퍼셉트론을 층 구조로 구성한 신경망 모델



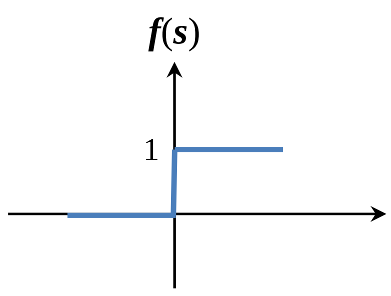




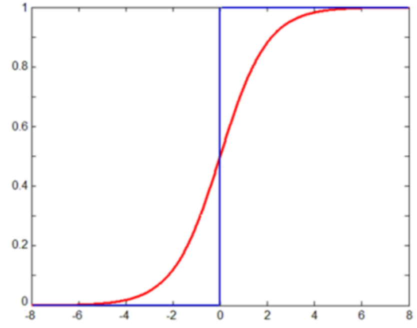
$$y = f(s) = f\left(\sum_{i=1}^{2} w_i x_i + b\right) = f(\boldsymbol{w}^\top \boldsymbol{x})$$

다층 퍼셉트론

- ❖ **다층 퍼셉트론**(multilayer Perceptron, **MLP**) – cont.
  - **다층 퍼셉트론의 학습**
    - 입력-출력 $(\boldsymbol{x}_i, y_i)$의 학습 데이터에 대해서, 기대출력 y_i와 다층 퍼셉트론의 출력 $f(\boldsymbol{x}_i)$의 차이, 즉 **오차**(error)가 최소가 되도록 **가중치 $\boldsymbol{w}$**를 **결정**하는 것
  - **학습 가능한 다층 퍼셉트론**
    - **오차 역전파(error backpropagation) 알고리즘**
      - 활성화 함수를 **계단 함수**에서 미분가능한 **시그모이드 함수**로 대체
      - **경사 하강법** 적용



계단 (step) 함수



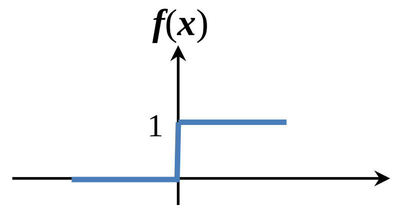
시그모이드(sigmoid) 함수

다층 퍼셉트론

❖ 활성화 함수(activation function)

▪ 계단 (step) 함수

```
def step(x):
    if x > 0:
        return 1
    else:
        return 0
```



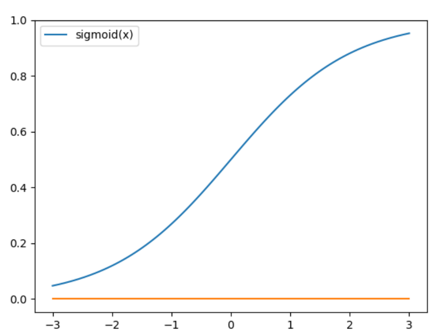
▪ 시그모이드(sigmoid) 함수

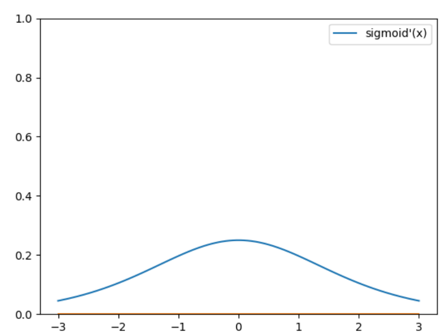
- **구간 (0,1)의 출력**

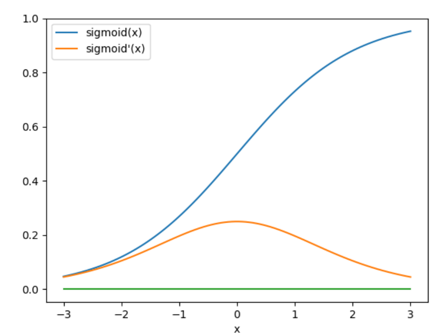
```
def sigmoid(x, a=1):
    return 1/(1+np.exp(-a*x))

def d_sigmoid(x, a=1):
    return a*sigmoid(x,a)*(1 - sigmoid(x,a))
```

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

$$\sigma'(x,a) = a\sigma(x,a)(1-\sigma(x,a))$$







다층 퍼셉트론

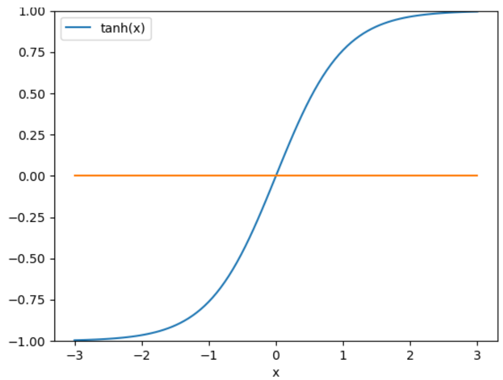
❖ **활성화 함수**(activation function) – cont.

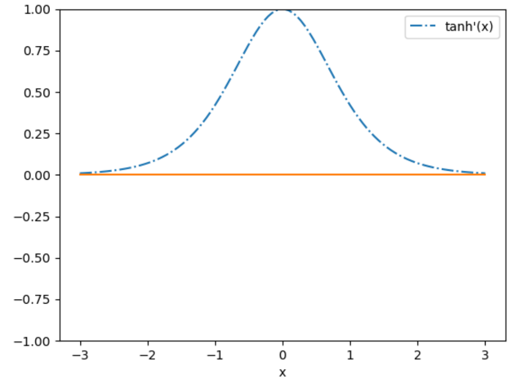
- **쌍곡탄젠트(tanh) 함수**
  - 구간 (-1, 1)의 출력

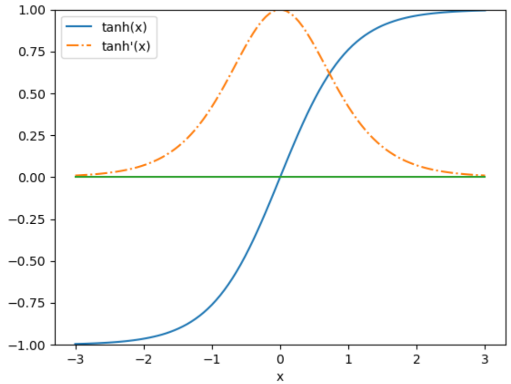
```
def tanh(x):
    return (np.exp(2*x)-1)/(np.exp(2*x)+1)

def d_tanh(x):
    return 1.0-tanh(x)*tanh(x)
```

$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1}$$

$$\tanh'(x) = 1 - \tanh^2(x)$$

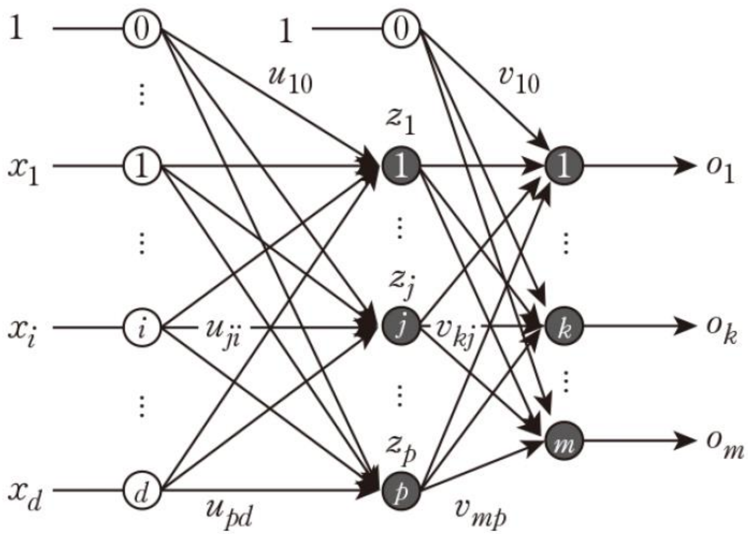






다층 퍼셉트론

❖ 다층 퍼셉트론 MLP의 동작



입력층 **은닉층 (hidden layer)** **출력층**

입력 $(x_1, x_2, \cdots, x_d)$

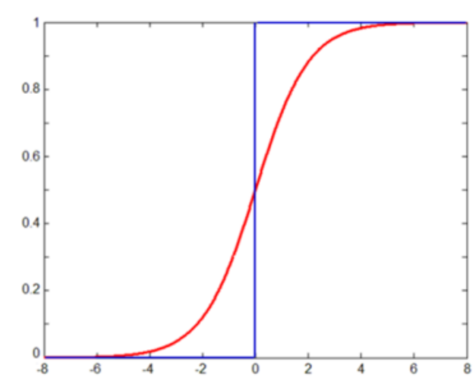
출력 $(y_1, y_2, \cdots, y_m)$

은닉층 $zsum_j = \sum_{i=1}^{d} u_{ji} x_i + u_{j0} \quad (1 \le j \le p)$

$$z_j = f(zsum_j)$$

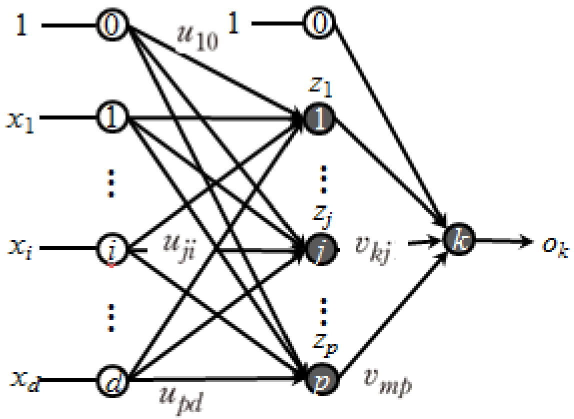
출력층 $osum_k = \sum_{j=1}^{p} v_{jk} z_k + v_{0k} \quad (1 \le k \le m)$

$$o_k = f(osum_k)$$



다층 퍼셉트론

❖ 다층 퍼셉트론(MLP)의 학습



입력 : $(x_1, x_2, \ldots, x_d)$

기대 출력 : y_k

MLP 출력 : o_k

- **학습 목표**
  - **기대 출력**과 **MLP 출력**이 최대한 **비슷**해지도록 **가중치**를 변경하는 것

$$E = \frac{1}{2}(o_k - y_k)^2$$

  - **경사 하강법**(gradient descent method) 사용

$$v_{jk}^{(t+1)} = v_{jk}^{(t)} - \eta \frac{\partial E}{\partial v_{jk}}$$

$$u_{ij}^{(t+1)} = u_{ij}^{(t)} - \eta \frac{\partial E}{\partial u_{ij}}$$

4. 미분

❖ **미분**(differentiation, 微分)

- 함수 $f(x)$의 변수 x에 대한 순간변화율
- x의 아주 미세한 변화에 대한 $f(x)$의 변화량

$$\boxed{f'(x) = \frac{df(x)}{dx} = \lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x}}$$

- 예) $f(x) = 2x^2 + 1$

$$f'(x) = \lim_{\Delta x \to 0} \frac{2(x + \Delta x)^2 + 1 - 2x^2 - 1}{\Delta x}$$

$$= \lim_{\Delta x \to 0} \frac{2x^2 + 4x\Delta x + (\Delta x)^2 + 1 - 2x^2 - 1}{\Delta x}$$

$$= \lim_{\Delta x \to 0} \frac{4x\Delta x + (\Delta x)^2}{\Delta x}$$

$$= \lim_{\Delta x \to 0} (4x + \Delta x)$$

$$= 4x$$

미분

- 상수 미분 $\frac{d}{dx}c = 0$
- 거듭제곱 미분 $\frac{d}{dx}x^n = n\,x^{n-1}$
- 지수함수의 미분 $\frac{de^{ax}}{dx} = ae^{ax}$
- 로그함수의 미분 $\frac{d\log x}{dx} = \frac{1}{x}$

- 상수배의 미분 $\frac{d}{dx}[cf(x)] = c\frac{d}{dx}f(x)$
- 합의 미분 $\frac{d}{dx}[f(x)+g(x)] = \frac{d}{dx}f(x) + \frac{d}{dx}g(x)$
- 곱의 미분 $\frac{d}{dx}[f(x)\,g(x)] = g(x)\,\frac{d}{dx}f(x) + f(x)\,\frac{d}{dx}g(x)$
- 분수식의 미분 $\frac{d}{dx}\left(\frac{f(x)}{g(x)}\right) = \frac{g(x)\,f'(x)+f(x)\,g'(x)}{(g(x))^2}$

미분

- **연쇄 법칙**(Chain Rule)

$$y = f\big(g(x)\big)$$

$$y = f(u), \qquad u = g(x)$$

$$\boxed{\frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx}}$$

$$x \xrightarrow{g} u \xrightarrow{f} y$$

$$\frac{\Delta y}{\Delta x} = \frac{\Delta y}{\Delta u}\frac{\Delta u}{\Delta x}$$

$$\frac{dy}{dx} = \lim_{\Delta x \to 0}\frac{\Delta y}{\Delta x} = \lim_{\Delta x \to 0}\left(\frac{\Delta y}{\Delta u}\frac{\Delta u}{\Delta x}\right) = \left[\lim_{\Delta x \to 0}\frac{\Delta y}{\Delta u}\right]\left[\lim_{\Delta x \to 0}\frac{\Delta u}{\Delta x}\right]$$

$\Delta x \to 0$이면, $\Delta u \to 0$ 이므로

$$= \left[\lim_{\Delta u \to 0}\frac{\Delta y}{\Delta u}\right]\left[\lim_{\Delta x \to 0}\frac{\Delta u}{\Delta x}\right]$$

$$= \frac{dy}{du}\frac{du}{dx}$$

미분

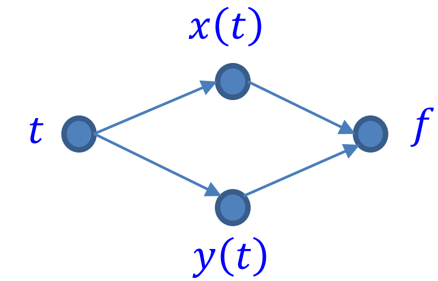
- **편미분**(partial differentiation)
  - **다변수 함수**에 대하여, 그 중 **하나의 변수**에 주목하고 나머지 변수의 값을 고정시켜 놓고 그 변수에 대해 하는 **미분**

  - 예. $f(x, y) = x^2 + xy + y^2$

$$\frac{\partial f(x, y)}{\partial x} = 2x + y$$

$$\frac{\partial f(x, y)}{\partial y} = x + 2y$$

미분



❖ 다변수 함수의 연쇄 법칙

- $f\big(x(t), y(t)\big)$

$$\frac{df(x(t),y(t))}{dt} = \frac{\partial f}{\partial x}\frac{dx}{dt} + \frac{\partial f}{\partial y}\frac{dy}{dt}$$

- 예. $f\big(x(t), y(t)\big) = x(t) + 2y(t)$
  $x(t) = 2t + 4, \quad y(t) = t^2$

$$\frac{\partial f}{\partial x} = 1\,, \ \frac{\partial f}{\partial y} = 2$$
$$\frac{dx}{dt} = 2\,, \frac{dy}{dt} = 2t$$

$$\frac{df(x(t),y(t))}{dt} = \frac{\partial f}{\partial x}\frac{dx}{dt} + \frac{\partial f}{\partial y}\frac{dy}{dt}$$
$$= 2 + 4t$$

- $g(x(t), y(t), z(t))$

$$\frac{dg(x(t),y(t),z(t))}{dt} = \frac{\partial g}{\partial x}\frac{dx}{dt} + \frac{\partial g}{\partial y}\frac{dy}{dt} + \frac{\partial g}{\partial z}\frac{dz}{dt}$$

- $h\big(x_1(t_1, t_2, \dots, t_m), x_2(t_1, t_2, \dots, t_m), \dots, x_n(t_1, t_2, \dots, t_m)\big)$

$$\frac{\partial h}{\partial t_i} = \frac{\partial h}{\partial x_1}\frac{\partial x_1}{\partial t_i} + \frac{\partial h}{\partial x_2}\frac{\partial x_2}{\partial t_i} + \cdots + \frac{\partial h}{\partial x_n}\frac{\partial x_n}{\partial t_i}$$

미분

- **그레디언트**(gradient)
  - 함수 $f(x, y, z)$의 각 변수 x, y, z에 대한 편미분을 성분으로 갖는 벡터

$$\nabla f(x, y, z) = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \\ \frac{\partial f}{\partial z} \end{bmatrix}$$

  - 함수 $f(x, y, z)$의 값이 가장 커지는 방향과 크기를 나타내는 벡터

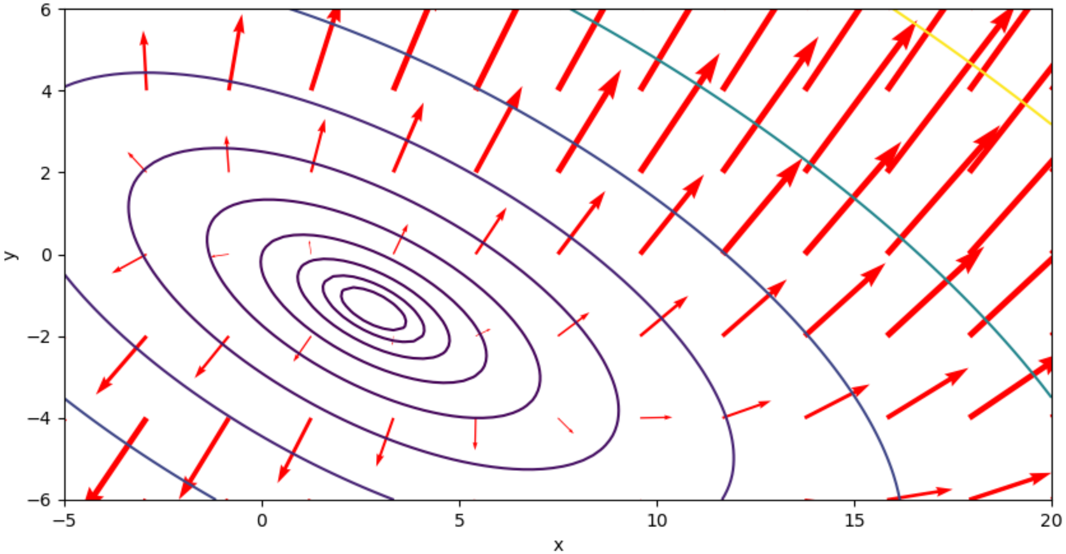
[실습] 그레디언트 그리기

```
import numpy as np
import matplotlib.pyplot  as plt

def f(x,y):
    return 2*x**2 +4*x*y + 5*y**2 - 6*x + 2*y + 10

def dx(x,y):
    return 4*x + 4*y - 6

def dy(x,y):
    return 4*x + 10*y + 2

xi = np.linspace(-5, 20, 100)
yi = np.linspace(-6, 6, 100)
X, Y = np.meshgrid(xi, yi)
Z = f(X,Y)

xj = np.linspace(-5, 20, 13)
yj = np.linspace(-6, 6, 7)
X1, Y1  = np.meshgrid(xj, yj)
Dx = dx(X1, Y1)
Dy = dy(X1, Y1)

plt.figure(figsize=(10,5))
plt.contour(X, Y, Z, levels=np.logspace(0,3,10))
plt.quiver(X1, Y1, Dx, Dy, color='red', scale=500, minshaft=4)
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```

$$f(x,y) = 2x^2 + 4xy + 5y^2 - 6x + 2y + 10$$

$$\frac{\partial f(x,y)}{\partial x} = 4x + 4y - 6$$

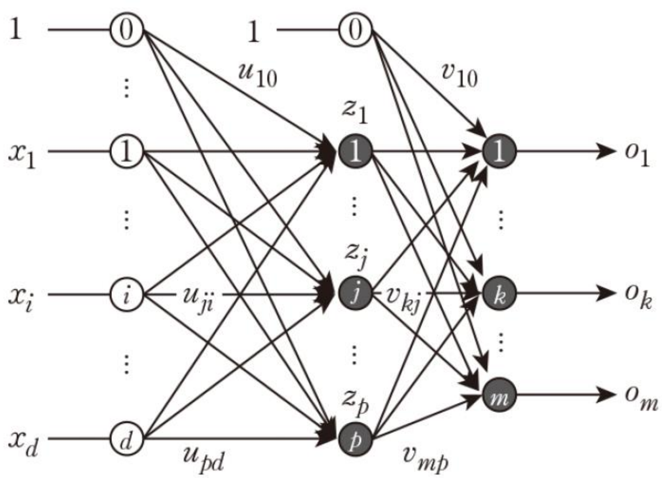
$$\frac{\partial f(x,y)}{\partial x} = 4x + 10y + 2$$



5. 다층 퍼셉트론의 학습

❖ 다층 퍼셉트론 MLP의 학습

- **오차 역전파 알고리즘**(Error back propagation algorithm, Backprop algorithm)



입력 $(x_1, x_2, \cdots, x_d)$

출력 $(y_1, y_2, \cdots, y_m)$

$$osum_k = \sum_{j=1}^{p} v_{kj} z_j + v_{k0} \quad (1 \le k \le m)$$

$$o_k = f(osum_k)$$

$$zsum_j = \sum_{i=1}^{d} u_{ji} x_i + u_{j0} \quad (1 \le j \le p)$$

$$z_j = f(zsum_j)$$

$$E = \frac{1}{2} \sum_{k=1}^{m} (o_k - y_k)^2$$ **오차함수**

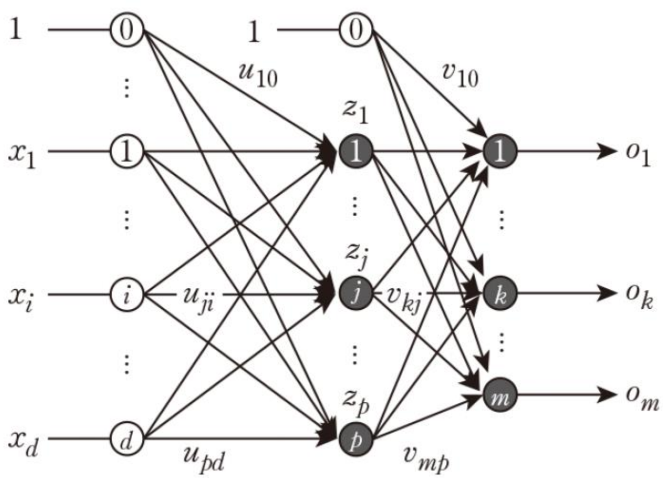
$$\boldsymbol{v}^{(t+1)} = \boldsymbol{v}^{(t)} - \eta \frac{\partial E}{\partial \boldsymbol{v}}$$

$$\boldsymbol{u}^{(t+1)} = \boldsymbol{u}^{(t)} - \eta \frac{\partial E}{\partial \boldsymbol{u}}$$

다층 퍼셉트론의 학습

❖ 다층 퍼셉트론 MLP의 학습

- **오차 역전파 알고리즘**(Error back propagation algorithm, Backprop algorithm)



입력 $(x_1, x_2, \cdots, x_d)$

출력 $(y_1, y_2, \cdots, y_m)$

$$\boxed{E = \frac{1}{2}\sum_{k=1}^{m}(o_k - y_k)^2}$$ 오차함수

$$\boldsymbol{v}^{(t+1)} = \boldsymbol{v}^{(t)} - \eta\frac{\partial E}{\partial \boldsymbol{v}}$$

$$\boldsymbol{u}^{(t+1)} = \boldsymbol{u}^{(t)} - \eta\frac{\partial E}{\partial \boldsymbol{u}}$$

$$\frac{\partial E}{\partial v_{kj}} = \frac{\partial E}{\partial o_k}\frac{\partial o_k}{\partial v_{kj}} = (o_k - t_k)f'(osum_k)z_j = \delta_k z_j$$

$$osum_k = \sum_{j=1}^{p} v_{kj}z_j + v_{k0} \quad (1 \le k \le m)$$

$$o_k = f(osum_k)$$

$$\frac{\partial E}{\partial u_{ji}} = \frac{\partial E}{\partial z_j}\frac{\partial z_j}{\partial u_{ji}} = \sum_{k=1}^{m}\frac{\partial E}{\partial o_k}\frac{\partial o_k}{\partial z_j}f'(zsum_j)x_i$$

$$= \sum_{k=1}^{m}(o_k - t_k)f'(osum_k)v_{kj}f'(zsum_j)x_i$$

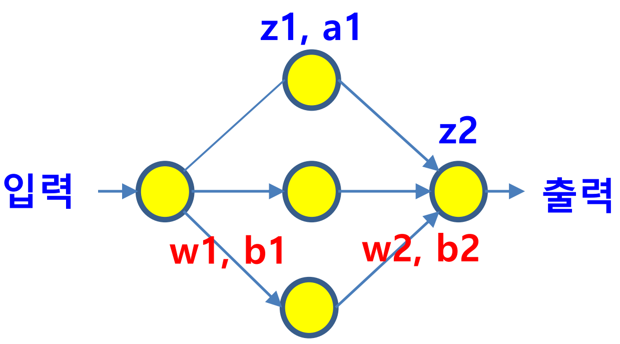
$$= \sum_{k=1}^{m}\delta_k v_{kj}f'(zsum_j)x_i$$

$$zsum_j = \sum_{i=1}^{d} u_{ji}x_i + u_{j0} \quad (1 \le j \le p)$$

$$z_j = f(zsum_j)$$

[실습] MLP 학습

```
import numpy as np
import matplotlib.pyplot  as plt

class MLP:
    def __init__(self, hidden_node=3):
        self.input_node = 1;  self.hidden_node = hidden_node;  self.output_node = 1
        self.w1 = np.random.rand(self.hidden_node, self.input_node)
        self.b1 = np.random.rand(self.hidden_node, 1)
        self.w2 = np.random.rand(self.output_node, self.hidden_node)
        self.b2 = np.random.rand(self.output_node, 1)

    def sigmoid(self, x):
        return 1/(1+np.exp(-x))

    def d_sigmoid(self, x):
        return self.sigmoid(x)*(1-self.sigmoid(x))

    def train(self, train_x, train_y, alpha=0.1, max_iter=500):
        np.random.seed(0)
        input_node = self.input_node;  hidden_node = self.hidden_node
        output_node = self.output_node;  alpha = alpha;    max_iter = max_iter
        for iter in range(1, max_iter):
            for i in range(n_train):
                z1 = np.dot(self.w1, train_x[i].reshape(1,1))+self.b1;   a1 = self.sigmoid(z1);
                z2 = np.dot(self.w2, a1)+self.b2;    y_hat = z2;   y_hat_list[i] = y_hat
                e = 0.5*(train_y[i] - y_hat)**2;  dy = -(train_y[i] - y_hat)
                dz2 = 1;  dw2 = a1.T
                delta_w2 = dy*dz2*dw2;   delta_b2 = dy*dz;
                da1 = self.w2.T;   dz1 = self.d_sigmoid(z1);   dw1 = train_x[i].T
                delta_w1 = dy*dz2*da1*dz1*dw1;  delta_b1 = dy*dz2*da1*dz1
                self.w2 -= alpha*delta_w2;  self.b2 -= alpha*delta_b2
                self.w1 -= alpha*delta_w1;  self.b1 -= alpha *delta_b1
```

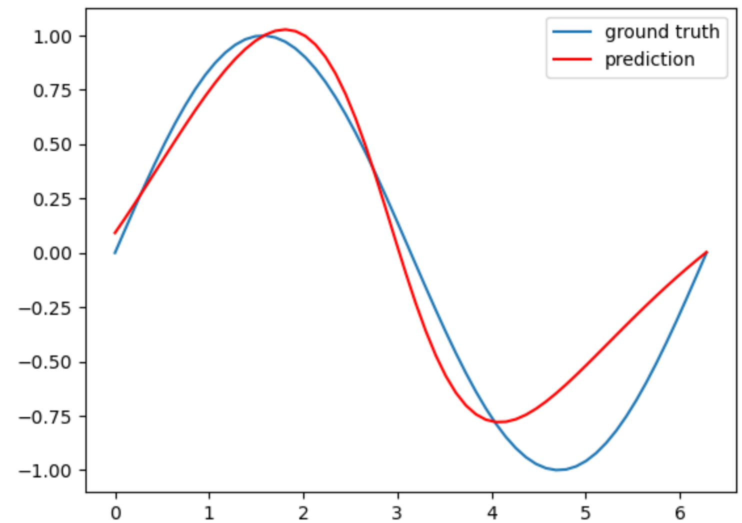


$$E = \frac{1}{2}(y_i - \hat{y})^2$$

$$w_1 \leftarrow w_1 \text{-} \alpha \Delta w_1$$

$$w_2 \leftarrow w_2 \text{-} \alpha \Delta w_2$$

```
    def predict(self, test_x):
        for i in range(n_test):
            z1 = np.dot(self.w1, test_x[i].reshape(1,1))+self.b1
            a1 = self.sigmoid(z1)
            z2 = np.dot(self.w2, a1)+self.b2
            y_hat = z2
            y_hat_list[i] = y_hat
        return y_hat_list

n_train = 20
train_x = np.linspace(0, np.pi*2, n_train)
train_y = np.sin(train_x)

n_test = 60
test_x = np.linspace(0, np.pi*2, n_test)
test_y = np.sin(test_x)
y_hat_list = np.zeros(n_test)

mlp = MLP(hidden_node=4)
mlp.train(train_x, train_y, max_iter=600)
plt.plot(test_x, test_y, label='ground truth')

y_hat_list = mlp.predict(test_x)
plt.plot(test_x, y_hat_list, '-r', label='prediction')
plt.legend( )
plt.show( )
```



[실습] sklearn의 MLP

```
import pandas as pd
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import classification_report,confusion_matrix

wine = load_wine()
data = pd.DataFrame(data=wine['data'], columns=wine['feature_names'])
print(data.head())

X = wine.data
y = wine.target
X_train, X_test, y_train, y_test = train_test_split(X, y)

scaler = StandardScaler()
scaler.fit(X_train)
StandardScaler(copy=True, with_mean=True, with_std=True)

X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)

mlp = MLPClassifier(hidden_layer_sizes=(13,13,13),max_iter=500)
mlp.fit(X_train,y_train)
predictions = mlp.predict(X_test)
print(confusion_matrix(y_test, predictions))
print(classification_report(y_test, predictions))
```

Classes	3
Samples per class	[59,71,48]
Samples total	178
Dimensionality	13
Features	real, positive

```
   alcohol  malic_acid   ash  ...   hue  od280/od315_of_diluted_wines
proline
0    14.23        1.71  2.43  ...  1.04                          3.92  1065.0
1    13.20        1.78  2.14  ...  1.05                          3.40  1050.0
2    13.16        2.36  2.67  ...  1.03                          3.17  1185.0
3    14.37        1.95  2.50  ...  0.86                          3.45  1480.0
4    13.24        2.59  2.87  ...  1.04                          2.93   735.0

[5 rows x 13 columns]
```

```
[[11  0  0]
 [ 1 15  1]
 [ 0  0 17]]
              precision    recall  f1-score   support

           0       0.92      1.00      0.96        11
           1       1.00      0.88      0.94        17
           2       0.94      1.00      0.97        17

    accuracy                           0.96        45
   macro avg       0.95      0.96      0.96        45
weighted avg       0.96      0.96      0.95        45
```

https://scikit-learn.org/

Quiz

1. 퍼셉트론은 선형불가 문제에 적용될 수 없다. (O,X)
2. 다층 퍼셉트론에서는 미분 가능한 활성화 함수를 사용하여 경사하강법으로 학습을 한다. (O,X)
3. 신경망에서 기본적인 학습은 가중치를 조정하는 것을 의미한다.(O,X)
4. 오차 역전파 알고리즘은 오차 함수에 대해 경사하강법을 적용하는 것으로 볼 수 있다. (O,X)