신경망 - 1
Neural Networks

이건명
충북대학교 대학원 산업인공지능학과

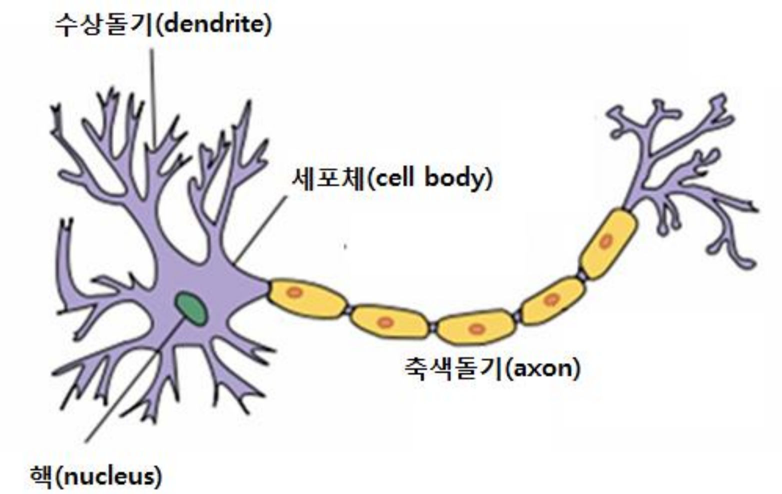
학습 내용

- 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

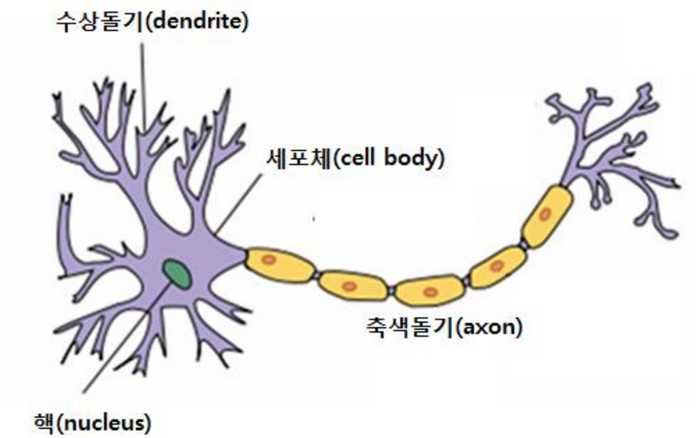
신경연접 1.5×10^{14} 개

- **수상돌기(樹狀突起, dendrite)** : 다른 신경세포의 축색돌기와 연결되어 전기화학적 신호를 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

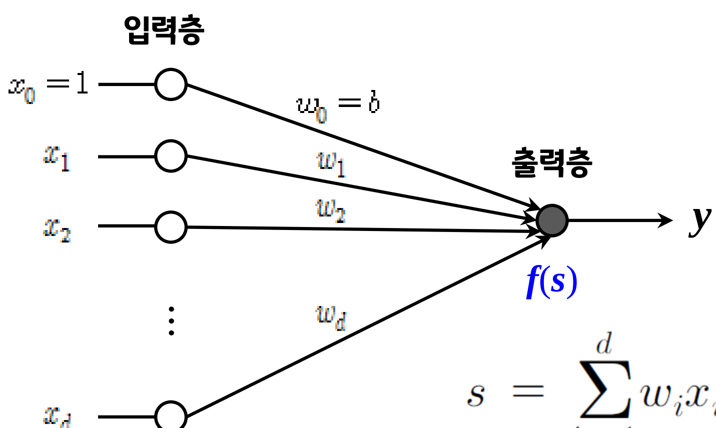
❖ 신경세포의 계산 모델

- **신경세포**

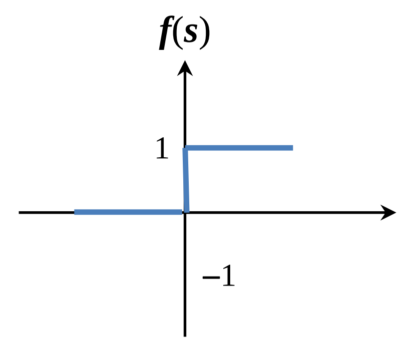


McClulloch & Pitts, 1943
신경세포의 계산 모델

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



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

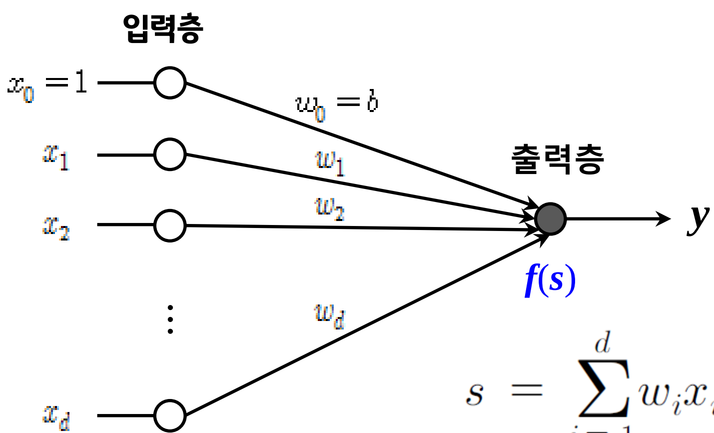


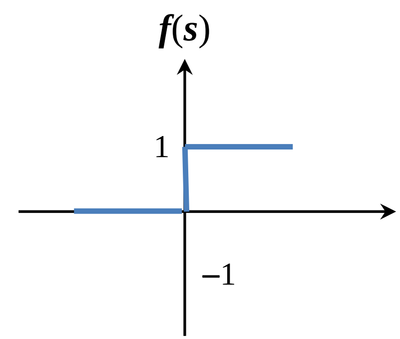
활성화 함수

2. 퍼셉트론

❖ 신경세포의 계산 모델

- **퍼셉트론(Perceptron)**





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

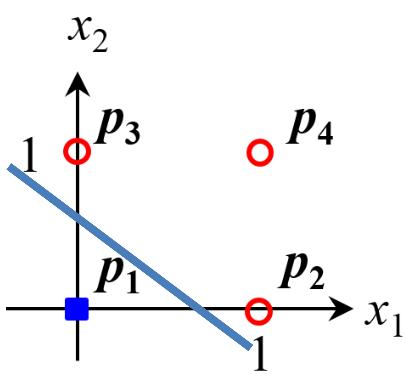
활성화 함수

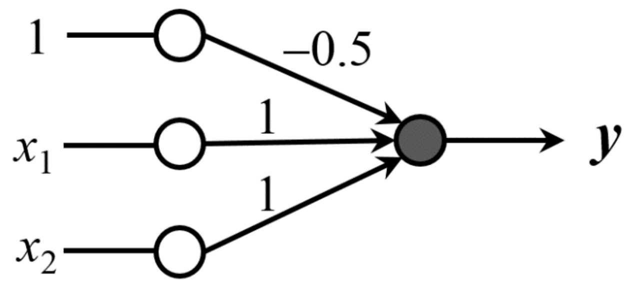
```
def Perceptron(inputs):
    sum = np.dot(inputs, weights[1:]) + weights[0]
    if sum > 0:
        activation = 1
    else:
        activation = 0
    return activation
```

퍼셉트론

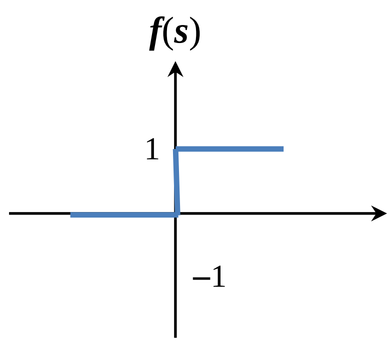
❖ **퍼셉트론(Perceptron)**

- **OR 연산**을 수행하는 퍼셉트론





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

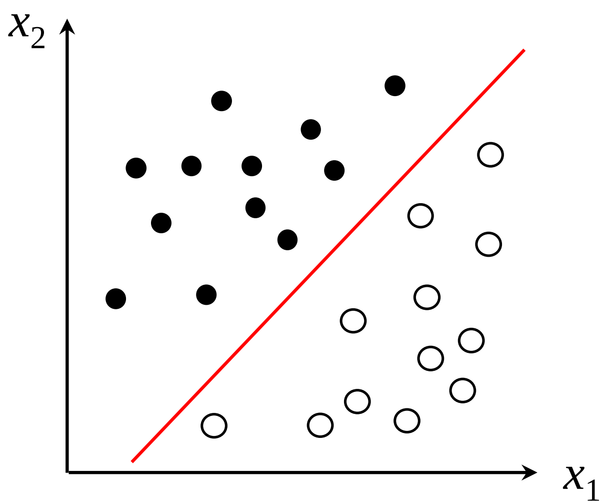


p_1 (0,0): $y = 0$
p_2 (1,0): $y = 1$
p_3 (0,1): $y = 1$
p_4 (1,1): $y = 1$

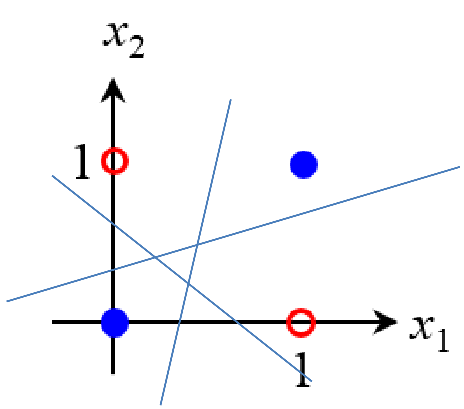
퍼셉트론

❖ **퍼셉트론(Perceptron)**

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



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



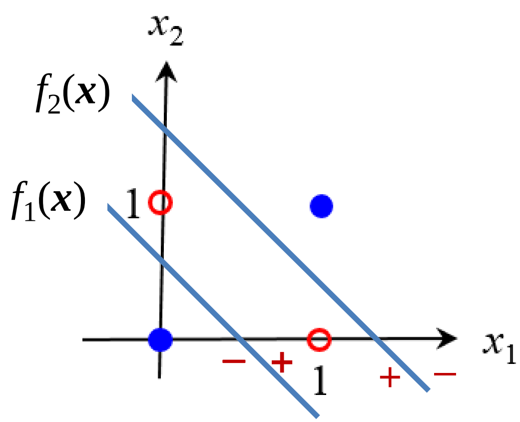
(0,0) : 0
(0,1) : 1
(1,0) : 1
(1,1) : 0

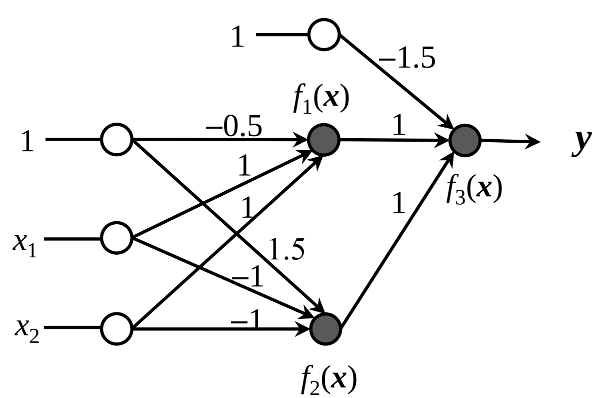
Minsky & Papert 1969

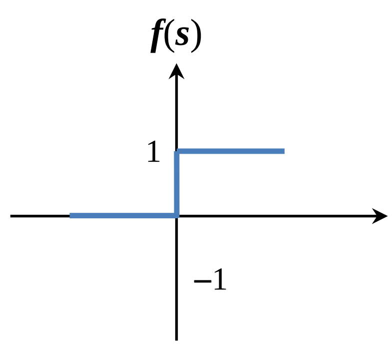
3. 다층 퍼셉트론

❖ **다층 퍼셉트론(multilayer Perceptron, MLP)**

- 여러 개의 퍼셉트론을 층 구조로 구성한 신경망 모델







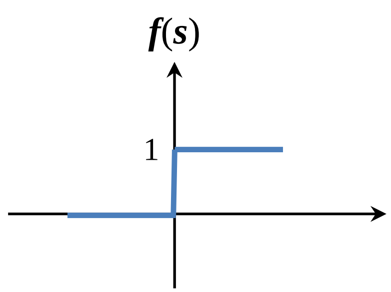
$$y = f(s) = f\left(\sum_{i=1}^{2} w_i x_i + b\right) = f(\boldsymbol{w}^\top \boldsymbol{x})$$

다층 퍼셉트론

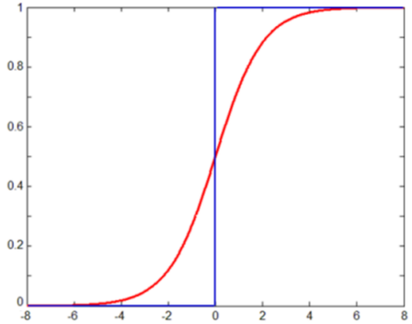
❖ **다층 퍼셉트론(multilayer Perceptron, MLP) – cont.**

- **다층 퍼셉트론의 학습**
  - 입력-출력 (x_i, y_i)의 학습 데이터에 대해서, 기대출력 y_i와 다층 퍼셉트론의 출력 $f(x_i)$의 차이, 즉 **오차**(error)가 **최소**가 되도록 **가중치** w를 **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



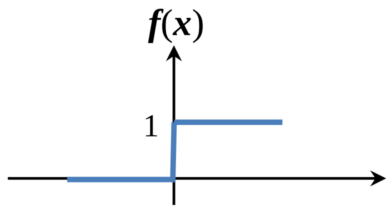
시그모이드(sigmoid) 함수

다층 퍼셉트론

❖ 활성화 함수(activation function)

▪ 계단 (step) 함수

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



▪ 시그모이드(sigmoid) 함수

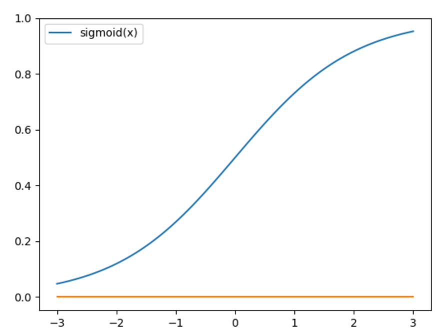
- 구간 (0,1)의 출력

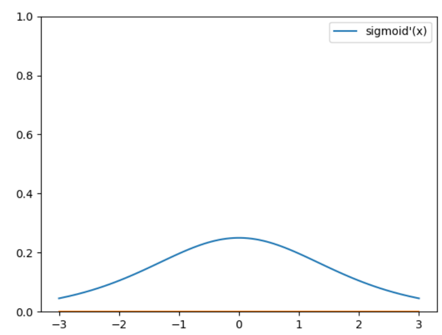
```
def sigmoid(x, a=1):
    return 1/(1+np.exp(-a*x))
```

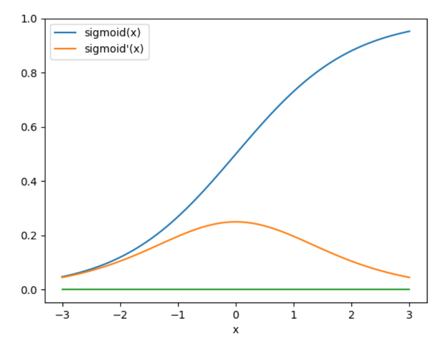
$$\sigma(x,a) = \frac{1}{1+e^{-ax}}$$

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

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







다층 퍼셉트론

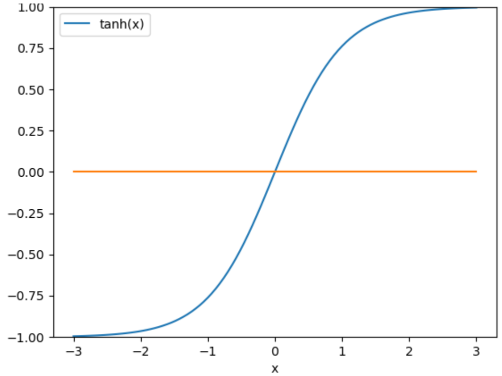
❖ **활성화 함수(activation function) – cont.**

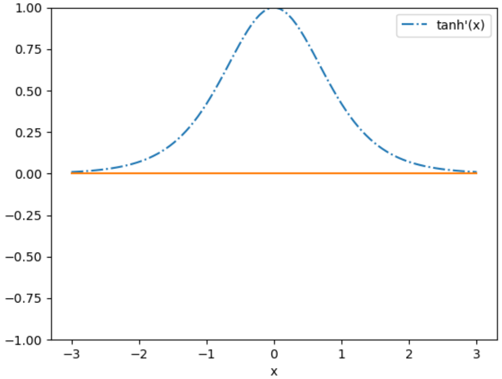
- **쌍곡탄젠트(tanh) 함수**
  - 구간 (-1, 1)의 출력

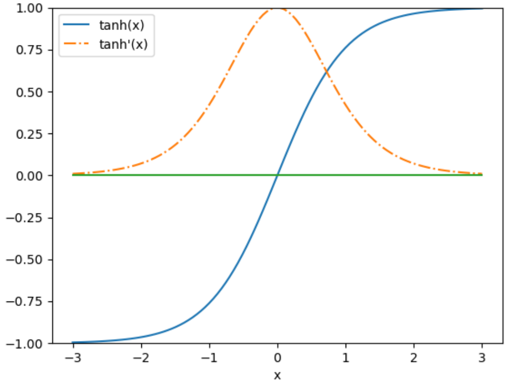
```
def tanh(x):
    return (np.exp(2*x)-1)/(np.exp(2*x)+1)

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

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

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

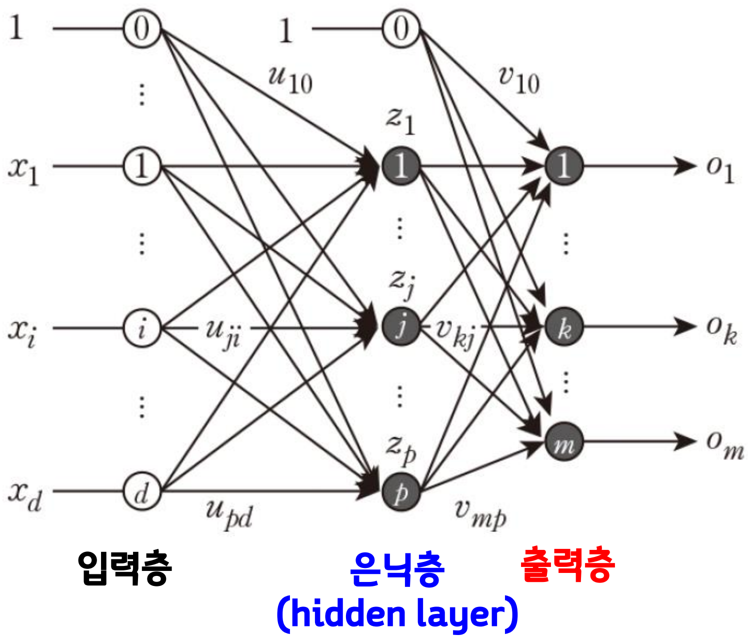






다층 퍼셉트론

❖ 다층 퍼셉트론 MLP의 동작



입력 $(x_1, x_2, \cdots, x_d)$

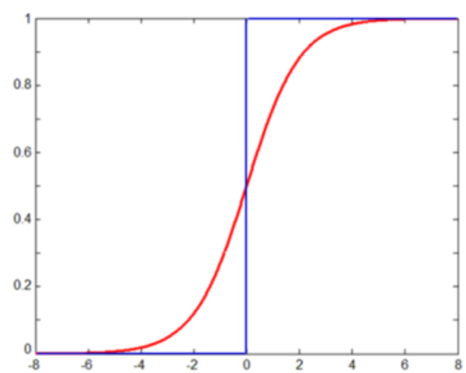
출력 $(y_1, y_2, \cdots, y_m)$

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

$$z_j = f(zsum_j)$$

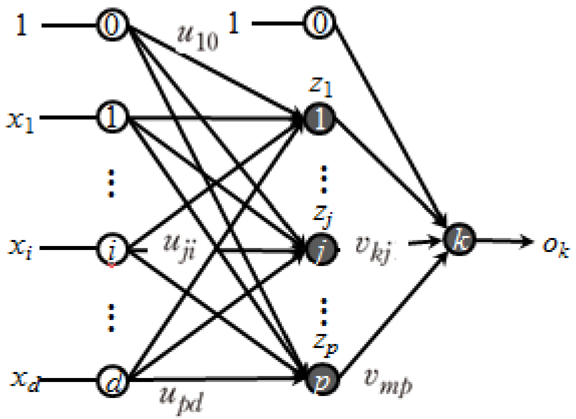
출력층 $osum_k = \sum_{j=1}^{p} v_{jk}z_k + v_{0k} \quad (1 \le k \le m)$

$$o_k = f(osum_k)$$



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 출력과 MLP 출력이 최대한 비슷해지도록 가중치를 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)

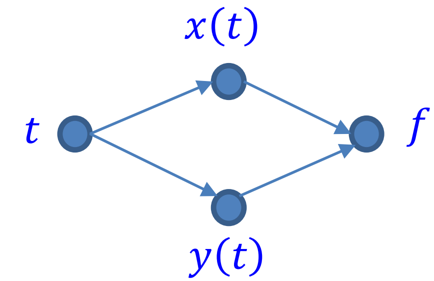
- **다변수 함수**에 대하여, 그 중 **하나의 변수**에 주목하고 나머지 변수의 값을 고정시켜 놓고 그 변수에 대해 하는 **미분**

- 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**

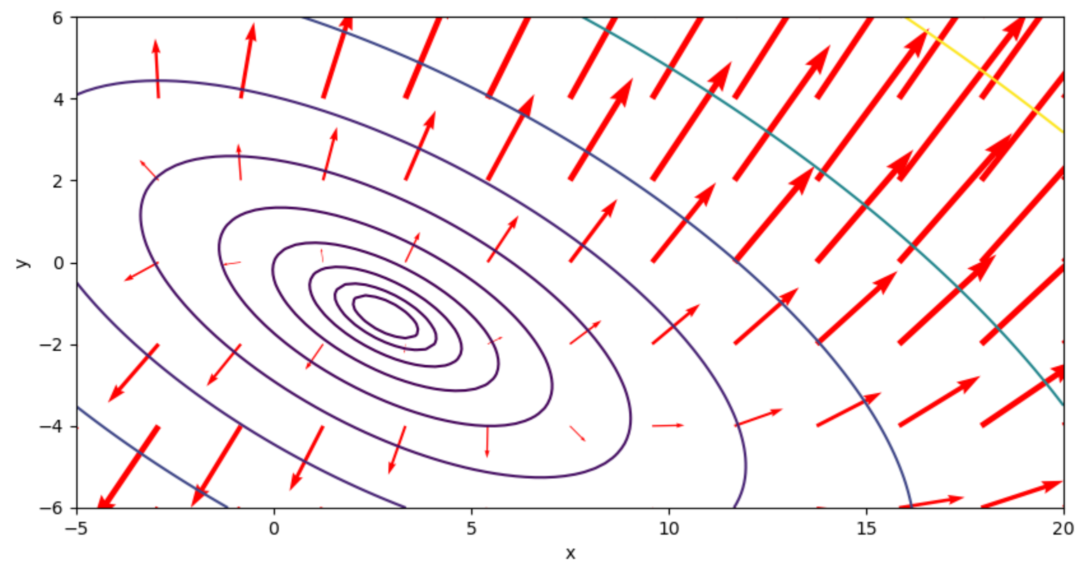
[실습] 그레디언트 그리기

```
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()
```

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

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

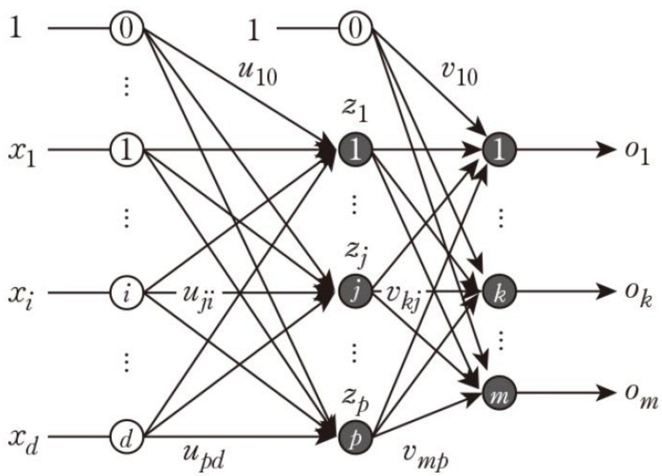
$$\frac{\partial f(x,y)}{\partial x} = 4x + 10y + 2$$



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 $(y_1, y_2, \cdots, y_m)$

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

$$o_k = f(osum_k)$$

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

$$z_j = f(zsum_j)$$

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

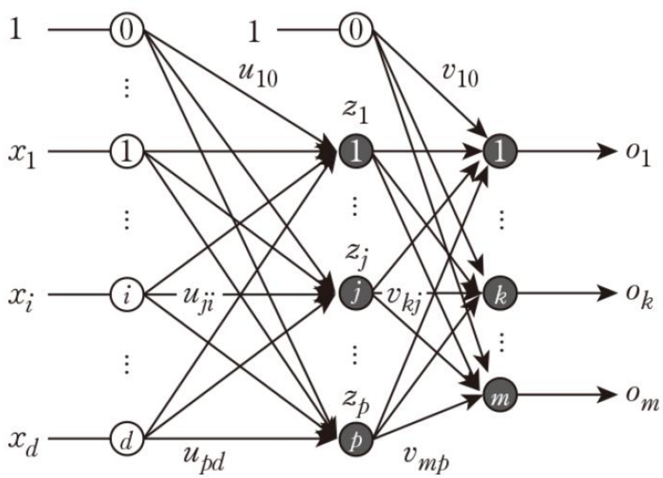
$$\boldsymbol{v}^{(t+1)} = \boldsymbol{v}^{(t)} - \eta \frac{\partial E}{\partial \boldsymbol{v}}$$

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

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

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

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

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

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

$$o_k = f(osum_k)$$

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

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

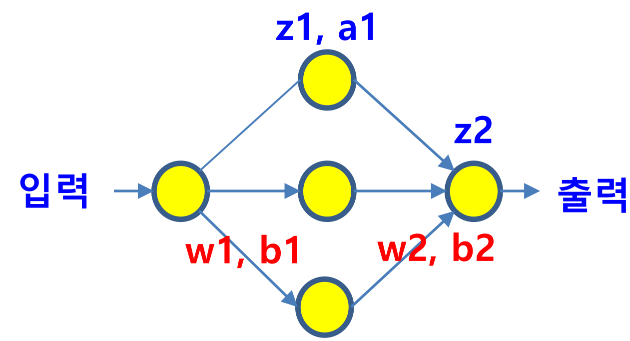
$$= \sum_{k=1}^{m}(o_k - t_k)f'(osum_k)v_{kj}f'(zsum_j)x_i$$

$$z_j = f(zsum_j)$$

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

[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
```

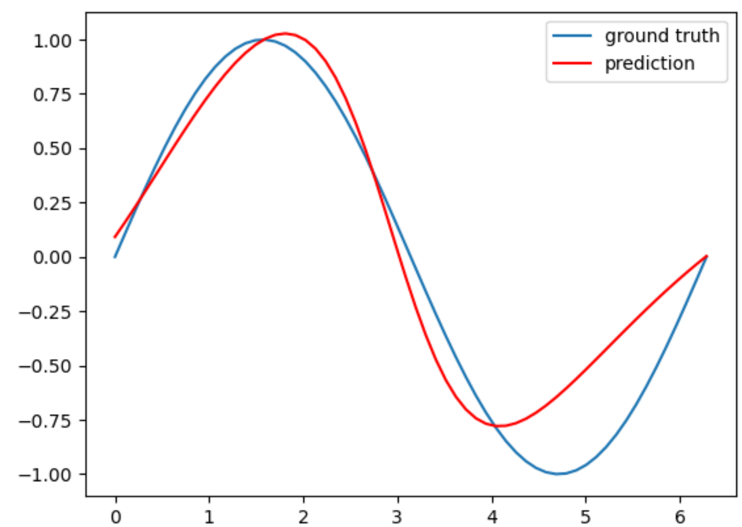


$$E = \frac{1}{2}(y_i - \hat{y})^2$$

$$w_1 \leftarrow w_1 - \alpha\Delta w_1$$

$$w_2 \leftarrow w_2 - \alpha\Delta w_2$$

```
    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

```python
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

❖ **신경세포의 계산 모델에서 가중치의 역할은?**

① 출력 생성
② 입력 신호의 중요도 조절
③ 활성화 함수 선택
④ 오류 계산

❖ **신경세포의 계산 모델에서 활성화 함수의 주요 목적은?**

① 가중치 업데이트
② 선형 출력의 비선형으로 변환
③ 오류 최소화
④ 학습률 조절

❖ **신경세포의 계산 모델의 출력 값이 특정 임계값을 초과하면 활성화 되는 활성화 함수는?**

① Sigmoid 함수
② Tanh 함수
③ ReLU 함수
④ Step 함수

Quiz

❖ **신경세포에 대한 설명으로 옳지 않은 것은?**

① 수상돌기는 다른 세포에서 전기화학적 신호를 받아들이는 역할을 한다.
② 축색돌기는 전기화학적 신호를 외부로 내보는 역할을 한다.
③ 수상돌기와 축색돌기를 연결하는 부위에서 전달되는 신호가 증폭되거나 감쇄되기도 한다.
④ 인공 신경세포는 생물체의 신경세포의 동작과 동일한 동작을 하도록 만든 모델이다.

❖ **신경망에서 활성화 함수의 역할에 대한 잘못된 설명은?**

① 비선형성을 도입하여 복잡한 함수를 모델링한다.
② 뉴런의 출력을 결정하는 데 사용된다.
③ 모든 신경망에는 동일한 활성화 함수가 사용된다.
④ 시그모이드나 ReLU와 같은 함수가 사용될 수 있다.

❖ **신경망에서 오차 역전파 알고리즘의 목적은?**

① 신경망의 가중치를 초기화한다.
② 출력층에서 발생하는 오차를 입력층으로 전파하여 가중치를 조정한다.
③ 신경망의 각 뉴런의 출력값을 계산한다.
④ 신경망의 학습률을 결정한다.

Quiz

❖ **신경망에 대한 설명으로 옳지 않는 것은?**

① 퍼셉트론은 로젠블랏이 제안한 단일 신경세포에 해당하는 연산을 수행할 수 있다.
② 퍼셉트론은 비선형 결정경계를 표현할 수 있다.
③ 신경망에서 학습은 모델의 파라미터들을 결정하는 일을 의미한다.
④ 다층 퍼셉트론을 사용하면 XOR 문제를 해결할 수 있는 모델을 만들 수 있다.

❖ **다층 퍼셉트론에 대한 설명으로 옳지 않은 것은?**

① 뉴런의 활성화 함수가 계단 모양의 함수이면, 경사하강법을 적용할 수 없다.
② 시그모이드 함수의 미분값은 시그모이드 함수의 값으로 계산할 수 있다.
③ 오차역전파 알고리즘은 경사하강법에 기반한 학습 방법이다.
④ 다층 퍼셉트론을 사용할 때 학습율은 -1에서 1 사이의 값을 사용한다.

❖ **퍼셉트론 모델을 개발한 사람은?**

① Geoffrey Hinton
② Yann LeCun
③ Frank Rosenblatt
④ Andrew Ng

Quiz

❖ 신경망의 오차 함수에 대한 설명으로 옳지 않은 것은?

① 이진 분류에서 오차함수로 음의 로그 가능도를 사용할 수 있다.
② 음의 로그 가능도를 사용하는 오차 함수를 최소화하는 것은 학습 데이터의 가능도를 최대로 하는 파라미터를 찾는 최대 가능도 추정을 하는 것과 같은 효과를 갖는다.
③ 음의 로그 가능도를 나타내는 식을 교차 엔트로피라고도 한다.
④ 교차 엔트로피를 오차 함수로 사용할 때는 경사 상승법을 사용한다.

❖ 다음 설명 중에서 옳지 않은 것은?

① 다부류 분류 문제에서는 부류가 3개 이상 있을 수 있다.
② 다층 퍼셉트론에서 마지막에 소프트맥스 층을 추가하면 출력의 값을 확률로 해설할 수 있다.
③ 분류 문제에서는 mean squared error를 오차 함수로 사용할 수 없다.
④ 다부류 분류 문제에서는 출력값을 one-hot 인코딩으로 나타낼 수 있다.

❖ 퍼셉트론은 주로 어떤 용도로 사용되는가?

① 비선형 회귀 문제 해결
② 분류 문제 해결
③ 클러스터링 문제 해결
④ 차원 축소

Quiz

❖ **ReLU 활성화 함수에서 음수 값에 대한 출력은?**

① 0
② 1
③ 입력 값 그대로
④ -1

❖ **Sigmoid 활성화 함수의 출력 범위는?**

① 0에서 1 사이
② -1에서 1 사이
③ 0에서 무한대
④ 모든 실수

❖ **어떤 활성화 함수가 출력의 중심을 0 주변으로 이동시키기 위해 사용되는가?**

① Sigmoid
② ReLU
③ Tanh
④ Step 함수

Quiz

❖ **단일 퍼셉트론이 형성하는 결정 경계의 형태는?**

① 비선형
② 다항식
③ 로그
④ 선형

❖ **다층 퍼셉트론에서 가중치를 업데이트하는 데 사용되는 알고리즘은 무엇입니까?**

① K-means 알고리즘
② 오차 역전파 알고리즘
③ 교차 엔트로피
④ 정보이득비

❖ **다층 퍼셉트론의 학습은 어떤 학습 알고리즘을 기반으로 하는가?**

① 강화 학습
② 비지도 학습
③ 지도 학습
④ 반지도 학습

Quiz

❖ **퍼셉트론의 가장 큰 제한점 중 하나는?**

① 복잡한 네트워크 구조를 필요로 한다.
② XOR 문제와 같은 비선형 문제를 해결할 수 없다.
③ 과적합의 위험이 없다.
④ 학습률을 조절할 수 없다.

❖ **여러 개의 퍼셉트론을 쌓아올린 구조를 무엇이라고 하는가?**

① 컨볼루션 신경망
② 순환 신경망
③ 딥러닝 신경망
④ 다층 퍼셉트론

❖ **퍼셉트론의 학습률은 무엇을 결정하는가?**

① 가중치 업데이트의 크기
② 네트워크의 깊이
③ 학습의 반복 횟수
④ 출력 뉴런의 수

Quiz

❖ $\boldsymbol{f(g(x))}$**의 도함수는?**

① $f'(g(x)) \times g'(x)$

② $f'(x) + g'(x)$

③ $f(g'(x))$

④ $f(x) \times g(x)$

❖ $\boldsymbol{f(x, y) = x^2 y}$**에 대한** x**에 대한 편미분은?**

① $2y$

② $2xy$

③ x^2

④ y^2

❖ **함수** $\boldsymbol{f(x, y) = x^3 + y^2}$**의** $\boldsymbol{y}$**에 대한 편미분은?**

① $3x^2$

② $2y$

③ $6x$

④ x^3

Quiz

❖ **신경망에서 손실 함수의 값이 크다면, 이는 무엇을 의미하는가?**

① 신경망이 잘 학습되었다.
② 신경망의 출력이 정확하다.
③ 신경망의 예측이 목표 값과 멀리 떨어져 있다.
④ 신경망의 가중치가 작다.

❖ **오차 역전파 알고리즘은 신경망의 어느 부분을 수정하기 위해 사용되는가?**

① 활성화 함수
② 학습률
③ 가중치와 편향
④ 입력 데이터

❖ **그레디언트의 방향은?**

① 함수 값이 증가하는 방향
② 함수 값이 감소하는 방향
③ 항상 정적인 방향
④ 원점의 방향

Quiz

❖ **신경망에서 가중치(Weight)의 역할에 대한 잘못된 설명은?**

① 입력 신호의 중요도를 조절한다.
② 학습 과정에서 조정된다.
③ 초기에는 무작위 값으로 설정된다.
④ 신경망의 구조를 결정한다.

❖ **신경망에서 학습률(Learning Rate)의 역할에 대한 잘못된 설명은?**

① 가중치의 업데이트 속도를 결정한다.
② 너무 높으면 학습이 불안정해질 수 있다.
③ 너무 낮으면 학습이 느려질 수 있다.
④ 모든 신경망에서 동일한 값으로 설정되어야 한다.

❖ **신경망에서 과적합(Overfitting)을 방지하는 방법 중 가장 잘못된 것은?**

① 드롭아웃(Dropout) 사용
② 데이터 증강(Data Augmentation)
③ 학습률 증가
④ 조기 종료(Early Stopping)

Quiz

❖ **신경망의 뉴런이 수행하는 연산에 대한 잘못된 설명은?**

① 여러 입력 신호가 가중치와 결합된다.
② 활성화 함수를 통해 최종 출력이 결정된다.
③ 모든 입력 신호는 동일한 가중치를 가진다.
④ 가중치와 입력 신호의 곱의 합이 계산된다.

❖ **다층 퍼셉트론(MLP)의 특징 중 잘못된 것은?**

① 여러 개의 은닉층을 포함할 수 있다.
② 복잡한 비선형 문제를 해결할 수 있다.
③ 모든 은닉층은 동일한 수의 뉴런을 가져야 한다.
④ 역전파 알고리즘을 통해 학습이 이루어진다.

❖ **신경망에 대한 설명으로 옳지 않은 것은?**

① 퍼셉트론은 단일 신경세포를 모사한 계산 모델이다.
② 퍼셉트론은 선형 분리불가 문제는 해결할 수 없다.
③ 퍼셉트론은 경사하강법으로 학습할 수 있다.
④ 퍼셉트론을 여러 층으로 구성한 것이 다층 퍼셉트론이다.