신경망 - 2
Neural Networks

이건명
충북대학교 소프트웨어학과

학습 내용

- 분류 문제의 종류에 대해서 살펴본다.
- 분류 문제에 출력값 표현 형태에 대해서 알아본다.
- 소프트맥스 층의 역할에 대해서 알아본다.
- 분류 문제에 대한 오차 함수들에 대해서 알아본다.
- RBF 모델의 구조와 학습 방법에 대해서 알아본다.

1. 분류 문제의 종류

❖ **이진 분류**(binary classification)
- 2개의 부류 중에서 하나 선택

- 스팸
- 비-스팸

❖ **다부류 분류**(multiclass classification)
- 3개 이상의 부류 중에서 하나 선택

- 개
- 고양이
- 토끼
- 사슴
- 양

❖ **다중레이블 분류**(multilabel classification)
- 하나의 대상에 대해서 여러 개의 부류 지정 가능

- 개
- 고양이
- 토끼
- 사슴
- 양

www.eltiempo.com

2. 분류 문제의 출력값 표현

❖ **출력값의 표현**

- **문자열** 또는 **기호** — 'cold', 'cold', 'warm', 'cold', 'hot',
- **정수 인코딩**(integer encoding) — 0 0 2 0 1
- **one-hot 인코딩** (one-hot encoding) — [1. 0. 0.] [1. 0. 0.] [0. 0. 1.] [1. 0. 0.] [0. 1. 0.]

❖ **학습 데이터**

$$D = \{(\boldsymbol{x}_1, \boldsymbol{t}_1), (\boldsymbol{x}_2, \boldsymbol{t}_2), \cdots, (\boldsymbol{x}_N, \boldsymbol{t}_N)\}$$

- i번째 데이터의 입력 : $\boldsymbol{x}_i$
- i번째 데이터의 입력 : $\boldsymbol{t}_i = (t_{i1}.t_{i2}, \ldots, t_{iK})$ $t_{ij} \in \{0, 1\}$, $\sum_{j=1}^{K} t_{ij} = 1$
  one-hot 벡터 표현

[실습] 출력값 인코딩

```
from numpy import array
from numpy import argmax
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder

data = ['cold', 'cold', 'warm', 'cold', 'hot', 'hot', 'warm', 'cold', 'warm', 'hot']
values = array(data)

label_encoder = LabelEncoder() # 정수 인코딩
integer_encoded = label_encoder.fit_transform(values)
print('₩n정수 인코딩 ₩n',integer_encoded)

onehot_encoder = OneHotEncoder(sparse=False)  # one-hot 인코딩
integer_encoded = integer_encoded.reshape(len(integer_encoded), 1)
onehot_encoded = onehot_encoder.fit_transform(integer_encoded)
print('₩none-hot 인코딩 ₩n', onehot_encoded)

inverted = label_encoder.inverse_transform([argmax(onehot_encoded[0, :])])
print('₩n', onehot_encoded[0], ' => ', inverted)
```

```
정수 인코딩
 [0 0 2 0 1 1 2 0 2 1]

one-hot 인코딩
 [[1. 0. 0.]
 [1. 0. 0.]
 [0. 0. 1.]
 [1. 0. 0.]
 [0. 1. 0.]
 [0. 1. 0.]
 [0. 0. 1.]
 [1. 0. 0.]
 [0. 0. 1.]
 [0. 1. 0.]]

 [1. 0. 0.]  =>  ['cold']
```

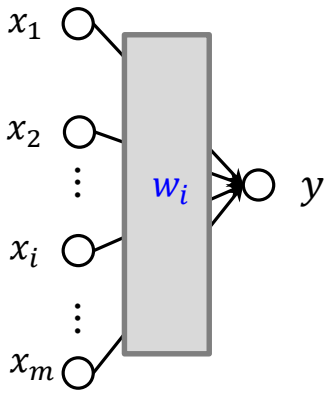
3. 오차 함수(손실 함수)

❖ **오차 함수**(error function, loss function)

- 기대하는 출력과 모델의 출력의 차이를 축적하여 표현하는 함수

❖ **회귀 문제의 오차 함수**

- **학습 데이터**
  - $D = \{(\boldsymbol{x}_1, t_1), (\boldsymbol{x}_2, t_2), \ \ldots, \ (\boldsymbol{x}_N, t_N)\}$



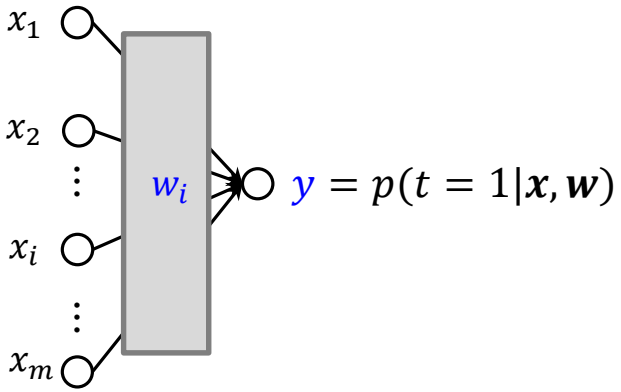
- **오차 함수 = mean squared error (MSE)**

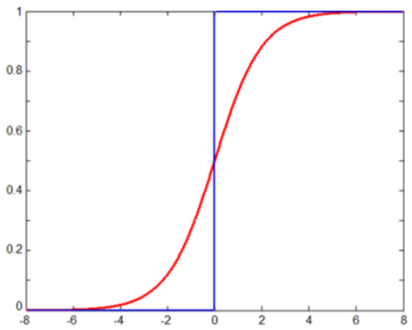
$$E = \frac{1}{2}\sum_{i=1}^{N}(y(\boldsymbol{x}_i, w) - t_i)^2$$

오차 함수

❖ 이진 분류 문제의 오차 함수

- 목표 부류 C_1의 목표값 $t = 1$, 목표 부류 C_2의 목표값 $t = 0$
- 활성화 함수로 **시그모이드(로직스틱) 함수** 사용
  - **구간 (0,1)** 사이의 값 출력 : **확률**로 해석 가능





$$y(\boldsymbol{x}, \boldsymbol{w}) = f(x, w) = \frac{1}{1+\exp(-\boldsymbol{w}^T \boldsymbol{x})}$$

- $y(\boldsymbol{x}, \boldsymbol{w})$는 조건부 확률 $p(C_1|\boldsymbol{x})$, $1 - y(\boldsymbol{x}, \boldsymbol{w})$는 조건부 확률 $p(C_2|\boldsymbol{x})$
- 입력 $\boldsymbol{x}$와 가중치 $\boldsymbol{w}$에 대한 목표값 t 에 대한 조건부 확률

$$p(t|\boldsymbol{x}, \boldsymbol{w}) = y(\boldsymbol{x}, \boldsymbol{w})^t \{1 - y(\boldsymbol{x}, \boldsymbol{w})\}^{1-t}$$

$t = 1$일 때, $p(t = 1|\boldsymbol{x}, \boldsymbol{w}) = y(\boldsymbol{x}, \boldsymbol{w})$
$t = 0$일 때, $p(t = 0|\boldsymbol{x}, \boldsymbol{w}) = 1 - y(\boldsymbol{x}, \boldsymbol{w})$

오차 함수

❖ 이진 분류 문제의 오차 함수 – cont.

- **학습 데이터** D의 독립 가정

$$D = \{(\boldsymbol{x}_1, t_1), (\boldsymbol{x}_2, t_2), \ \ldots, \ (\boldsymbol{x}_N, t_N)\}$$

- 가중치 $\boldsymbol{w}$에 대한 학습 데이터 D의 **가능도**(likelihood)

$$p(D; \boldsymbol{w}) = \prod_{i=1}^{N} y(\boldsymbol{x}_i, \boldsymbol{w})^{t_i}\{1 - y(\boldsymbol{x}_i, \boldsymbol{w})^{1-t_i}$$

- **오차 함수 = 음의 로그 가능도**(negative log likelihood)

$$E(\boldsymbol{w}) = -\log \prod_{i=1}^{N} y(\boldsymbol{x}_i, \boldsymbol{w})^{t_i}\{1 - y(\boldsymbol{x}_i, \boldsymbol{w})\}^{1-t_i}$$

$$= -\sum_{i=1}^{N}[t_i \log y(\boldsymbol{x}_i, \boldsymbol{w}) + (1 - t_i)\log\{1 - y(\boldsymbol{x}_i, \boldsymbol{w})\}]$$

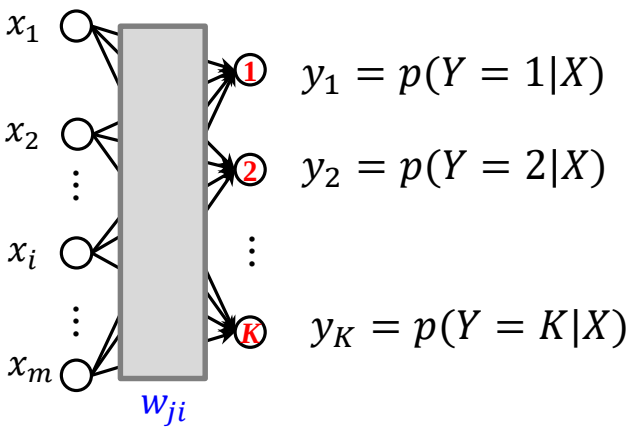
오차 함수

❖ 다부류 분류 문제의 오차 함수

- 학습 데이터 D

$$D = \{(\boldsymbol{x}_1, \boldsymbol{t}_1), (\boldsymbol{x}_2, \boldsymbol{t}_2), \ \ldots, \ (\boldsymbol{x}_N, \boldsymbol{t}_N)\}$$

$$\boldsymbol{t}_i = (t_{i1}, t_{i2}, \ldots, t_{iK}), \ t_{ik} \in \{0,1\}, \ \sum_{k=1}^{K} t_{ik} = 1$$ 출력 : one hot 인코딩

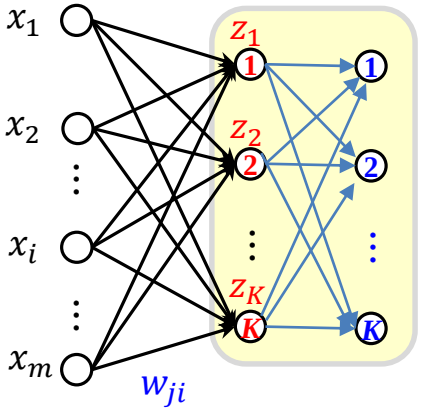


출력의 합 = 1
출력 ≥ 0

오차 함수

❖ 다부류 분류 문제의 오차 함수 – cont.

- **소프트맥스 층** (softmax layer)
  - 최종 출력을 **분류 확률**(classification probability)로 변환하는 층
    - 출력의 합 = 1



$$y_1 = p(Y = 1|X)$$

$$y_2 = p(Y = 2|X)$$

$$y_K = p(Y = K|X)$$

$$y_k = \frac{e^{z_k}}{\sum_{i=1}^{K} e^{z_i}}$$

10	11.8
6	0.21
12	87.9

```
def softmax(x):
    return np.exp(x) / np.sum(np.exp(x), axis=0)
```

오차 함수

❖ 다부류 분류 문제의 오차 함수 – cont.

- **학습 데이터**

$$D = \{(\boldsymbol{x}_1, \boldsymbol{t}_1), (\boldsymbol{x}_2, \boldsymbol{t}_2), \cdots, (\boldsymbol{x}_N, \boldsymbol{t}_N)\}$$

  - i번째 데이터의 입력 : $\boldsymbol{x}_i$
  - i번째 데이터의 입력 : $\boldsymbol{t}_i = (t_{i1}.t_{i2}, \ldots, t_{iK})$ $t_{ij} \in \{0, 1\}$, $\sum_{j=1}^{K} t_{ij} = 1$
    one-hot 벡터 표현

- **학습 데이터 $(\boldsymbol{x}_i, \boldsymbol{t}_i)$의 조건부 확률**

$$p(\boldsymbol{t}_i | \boldsymbol{x}_i, \boldsymbol{w}) = \prod_{k=1}^{K} y_k(\boldsymbol{x}_i, \boldsymbol{w})^{t_{ik}}$$

$\boldsymbol{w}$: 신경망의 동작을 결정하는 전체 가중치 벡터

- **전체 데이터 D에 대한 가능도**(likelihood)

$$p(D; \boldsymbol{w}) = \prod_{i=1}^{N} \prod_{k=1}^{K} y_k(\boldsymbol{x_i}, \boldsymbol{w})^{t_{ik}}$$

오차 함수

❖ 다부류 분류 문제의 오차 함수 – cont.

- **데이터의 가능도**를 **최대**로 하는 **파라미터 w**를 추정하는 것

$$\text{argmax}_{\boldsymbol{w}} \prod_{i=1}^{N}\prod_{k=1}^{K} y_k(\boldsymbol{x_i},\boldsymbol{w})^{t_{ik}}$$

- 오차함수 $E(\boldsymbol{w})$: **가능도**의 **음의 로그 가능도**(negative log likelihood)

$$E(\boldsymbol{w}) = -\log \prod_{i=1}^{N}\prod_{k=1}^{K} y_k(\boldsymbol{x_i},\boldsymbol{w})^{t_{ik}} = -\sum_{i=1}^{N}\sum_{k=1}^{K} t_{ik}\log y_k(\boldsymbol{x_i},\boldsymbol{w})$$

- **오차함수** = $(t_{i1}, t_{i2}, \cdots, t_{iK})$와 $(y_1, y_2, \cdots, y_K)$에 대한 **교차 엔트로피** (cross entropy)

$$E(\boldsymbol{w}) = -\sum_{k=1}^{K} t_{ik}\log y_k(\boldsymbol{x_i},\boldsymbol{w})$$

오차 함수

❖ **다중레이블 분류 문제의 오차 함수**

- **MSE**(mean squared error)

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

  - **최대값을 출력하는 복수 개의 노드**

- **교차 엔트로피**(cross enctropy)

$$E(\boldsymbol{w}) = -\sum_{k=1}^{K} t_{ik}\log y_k(\boldsymbol{x_i}, \boldsymbol{w})$$

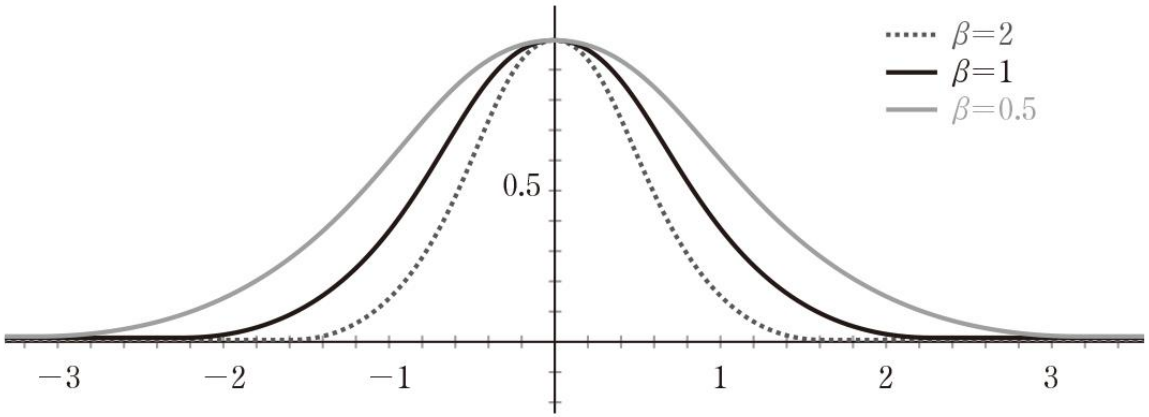
  - **최대값을 출력하는 복수 개의 노드**

4. RBF 망

❖ **RBF(radial basis function) 함수**

- **기존 벡터 μ 와 입력 벡터 x 의 유사도를 측정하는 함수**
- 예.

$$\phi(\boldsymbol{x}, \boldsymbol{\mu}) = \exp(-\beta\|\boldsymbol{x} - \boldsymbol{\mu}\|^2)$$

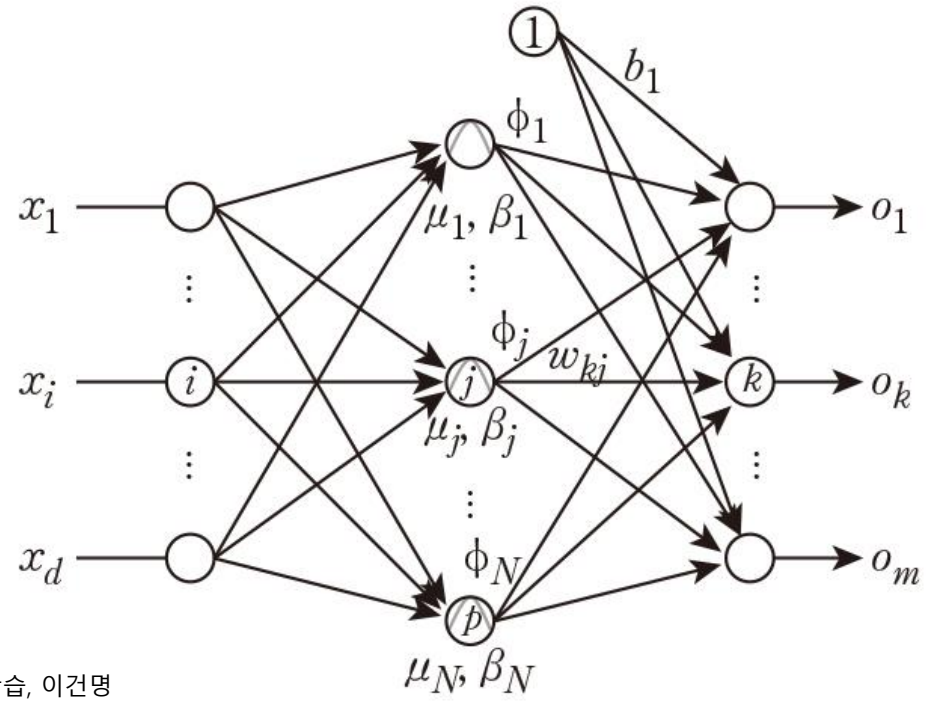


RBF 망

❖ **RBF 망 (RBF network)**

- 어떤 함수 $f_k(\boldsymbol{x})$를 다음과 같이 **RBF 함수들**의 **선형 결합 형태**로 근사시키는 모델

$$f_k(\boldsymbol{x}) \approx \sum_{i=1}^{N} w_{kj}\phi_i(\boldsymbol{x}, \boldsymbol{\mu}_i) + b_k$$



$$\phi_j = \exp(-\beta_j \|\boldsymbol{x} - \boldsymbol{\mu}_i\|^2)$$

$$o_k = \sum_{j=1}^{N} w_{kj}\phi_j + b_k$$

RBF 망

❖ RBF 망의 학습

- **오차 함수 E**

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

- **경사 하강법**(gradient-descent method) **사용**
  - 기준 벡터 $\boldsymbol{\mu}_j$와 파라미터 β_j, 가중치 w_{kj} 결정

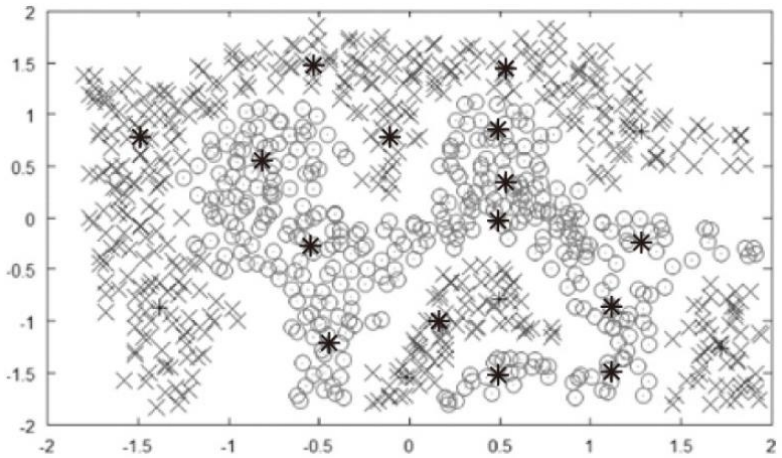
- 부류 별 군집화 결과를 사용한 기준 벡터 $\boldsymbol{\mu}_j$와 파라미터 β_j 초기화
  - 군집 중심 : 기준(평균) 벡터 $\boldsymbol{\mu}_j$
  - 분산의 역수 : β_j

$$\sigma = \frac{1}{m}\sum_{i=1}^{m}\|x_i - \mu\| \qquad \beta = \frac{1}{2\sigma^2}$$

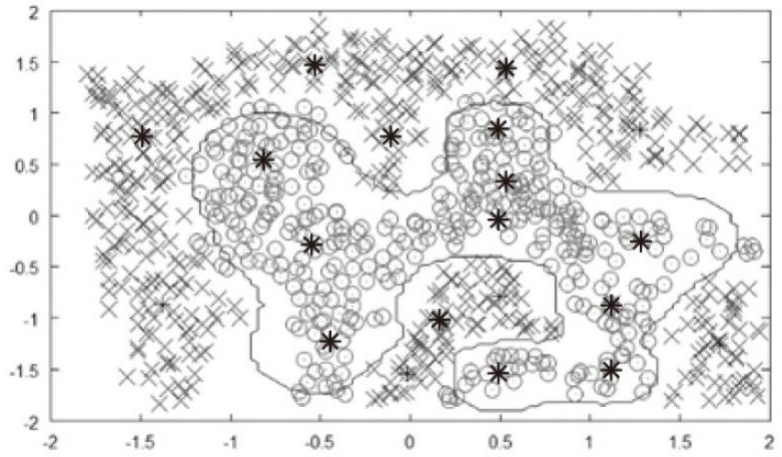
RBF 망

❖ **RBF 망을 이용한 분류의 예**

- **o 부류와 x 부류**



★ : 군집화 결과 군집 중심



곡선 : 분류 결정 경계

[실습] RBF망 학습

```
from scipy import *
from scipy.linalg import norm, pinv
from matplotlib import pyplot as plt
import numpy as np

class RBF:
    def __init__(self, indim, numCenters, outdim):
        self.indim = indim;   self.outdim = outdim;   self.numCenters = numCenters
        self.centers = [random.uniform(-1, 1, indim) for i in range(numCenters)]
        self.beta = 8
        self.W = random.random((self.numCenters, self.outdim))

    def basisFunc(self, c, d):
        assert len(d) == self.indim
        return np.exp(-self.beta * norm(c-d)**2)

    def activationFunc(self, X):
        G = np.zeros((X.shape[0], self.numCenters), float)
        for ci, c in enumerate(self.centers):
            for xi, x in enumerate(X):
                G[xi,ci] = self.basisFunc(c, x)
        return G

    def train(self, X, Y):
        rnd_idx = random.permutation(X.shape[0])[:self.numCenters]
        self.centers = [X[i,:] for i in rnd_idx]
        G = self.activationFunc(X)
        self.W = np.dot(pinv(G), Y)

    def predict(self, X):
        G = self.activationFunc(X)
        Y = np.dot(G, self.W)
        return Y
```

https://scikit-learn.org/

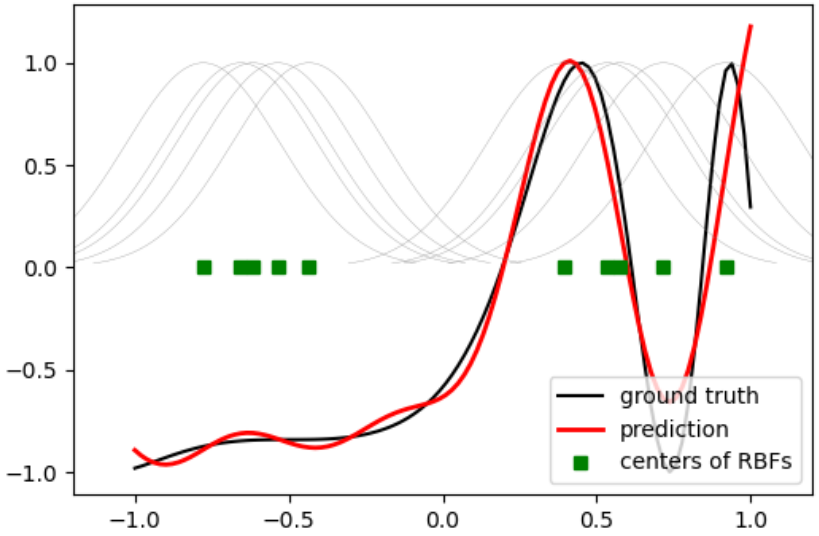
```
n = 100
x = mgrid[-1:1:complex(0,n)].reshape(n, 1)
y = np.sin(3*(x+0.5)**3 - 1)

rbf = RBF(1, 10, 1)
rbf.train(x, y)
z = rbf.predict(x)

plt.figure(figsize=(6, 4))
plt.plot(x, y, 'k-', label='ground truth')
plt.plot(x, z, 'r-', linewidth=2, label='prediction')
plt.plot(rbf.centers, np.zeros(rbf.numCenters), 'gs', label='centers of RBFs')

for c in rbf.centers:
    cx = np.arange(c-0.7, c+0.7, 0.01)
    cy = [rbf.basisFunc(np.array([cx_]), np.array([c])) for cx_ in cx]
    plt.plot(cx, cy, '-', color='gray', linewidth=0.2)

plt.xlim(-1.2, 1.2)
plt.legend( )
plt.show( )
```



https://scikit-learn.org/

Quiz

1. 다부류 분류 문제인 경우 출력을 one-hot 인코딩으로 표현해도 된다. (O,X)
2. 소프트맥스 층의 출력의 합은 1이다. (O,X)
3. 교차 엔트로피는 학습 데이터의 가능도를 최대로 하는 파라미터를 찾기 위한 분류 문제에 대한 오차함수이다. (O,X)
4. RBF 함수는 가우시안 함수와 유사한 형태를 갖는다. (O,X)
5. RBF 함수는 회귀 문제에는 적용될 수 있으나 분류 문제에는 적용할 수 없다. (O,X)