# *Deep Learning*

*Deep Learning*

# Softmax(Multinominal) 분류기

*Yoonjoong Kim*

*Department of Computer Engineering, Hanbat National University*

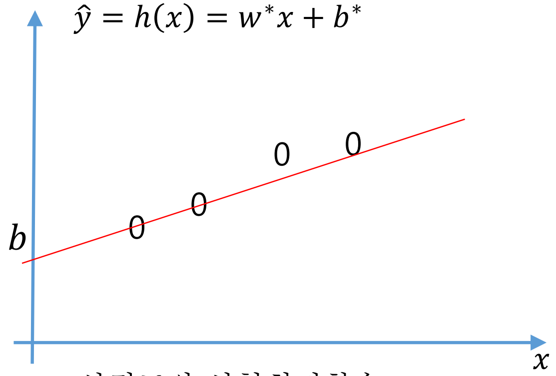
*yjkim@hanbat.ac.kr*

# Contents

# 1. 이진분류(logistic classifier)

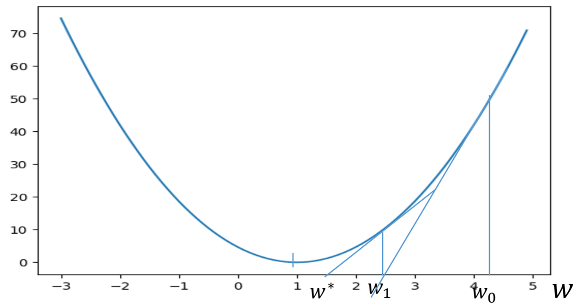
- Linear Regression의 개념
  - 모델
    - $\hat{y} = h(\mathrm{x}) = wx + b$ $[-\infty \sim +\infty]$
  - 손실함수(mean square error)
    - $loss(\mathrm{X}, \mathrm{Y}) = \frac{1}{N}\sum(\hat{y}_i - y_i)^2$
  - 학습
    - $w^*, b^* = argmin_{w,b}(loss(w,b)|X,Y))$
    - Gradient Descent Algorithm
  - 예측
    - $\hat{y} = h(x) = \sigma(\hat{y}_l), \hat{y}_l = w^*x + b^*$
  - 평가지수
    - $R^2 = 1 - \frac{\sum(y_\mathrm{i} - \hat{y}_i)^2}{\sum(y_\mathrm{i} - \bar{y})^2}$
    - 

| x (hours) | y (grade) |
|---|---|
| 2 | 70 |
| 3 | 80 |
| 5 | 85 |
| 6 | 90 |



산점도와 선형회귀함수

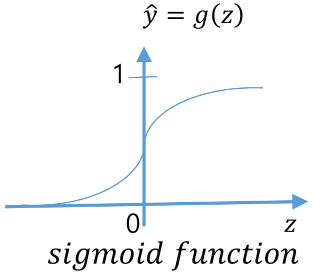
$loss(w)$ in Linear Regression



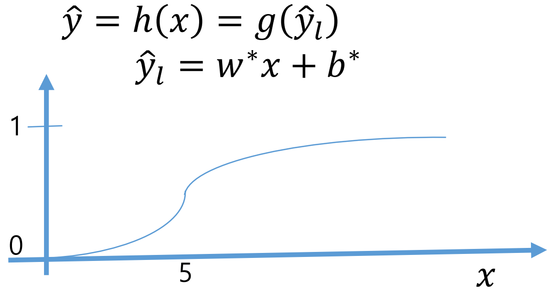
GDA 학습

# 1. 이진분류(logistic classifier)

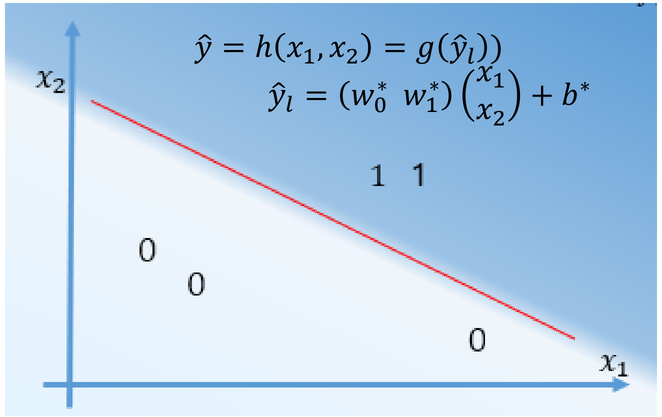
- Logistic Regression의 개념
  - **모델**
    - Logistic Regression 함수
      - $\hat{y} = h(x) = g(\hat{y}_l), g(z) = \frac{1}{1+e^{-z}}$ [0 ~1]
      - $\hat{y}_l = h_l(x) = wx + b$ $[-\infty \sim +\infty]$
  - 손실함수
      - $loss(X, Y) = \frac{1}{N}\sum cross\_entropy(\hat{y}_i, y_i))$
  - 학습
      - $w^*, b^* = argmin_{w,b}(loss(w, b)|X, Y))$
      - Gradient Descent Algorithm
  - 예측
      - $\hat{y} = h(x) = g(\hat{y}_l), \hat{y}_l = w^*x + b^*$
  - 평가지수
      - accuracy $= \frac{1}{N}\sum(\hat{y}_i = y_i)$




| x (hours) | y (grade) |
|---|---|
| 2 | 0 |
| 3 | 0 |
| 9 | 1 |
| 10 | 1 |
| | |




| X | | Y |
|---|---|---|
| x1 (hours) | x2 (attendance) | y (grade) |
| 2 | 4 | 0 |
| 3 | 3 | 0 |
| 9 | 5 | 1 |
| 10 | 5 | 1 |
| 11 | 1 | 0 |

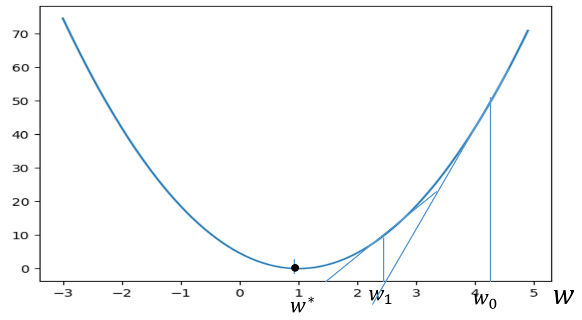



산점도와 로지스틱회귀함수

# 1. 이진분류(cont.)

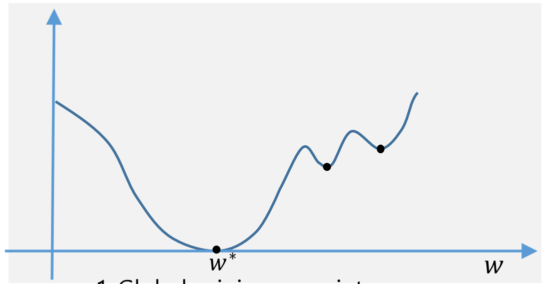
- Linear Regression
  - 모델
    - $\hat{y}_l = h_l(\mathrm{x}) = wx + b$
  - 손실함수 (mean square error)
    - $loss(X, Y) = \frac{1}{N}\sum(y_i - \hat{y}_i)^2$
  - **학습**
    - Gradient Descent Algorithm(GDA)
- Logistic Regression
  - 모델
    - $\hat{y} = h(x) = g(\hat{y}_l), g(z) = \frac{1}{1+e^{-z}}$ $[0 \sim 1]$
    - $\hat{y}_l = h_l(\mathrm{x}) = wx + b$ $[-\infty \sim +\infty]$
  - 손실함수 (binary cross entropy)
    - $loss(X, Y) = -\frac{1}{m}\sum_{i=1}^{m}(y log(\hat{y}_i) + (1 - y)\log(1 - \hat{y}_i))$
  - **학습**
    - Gradient Descent Algorithm

$loss(w)$ in Linear Regression



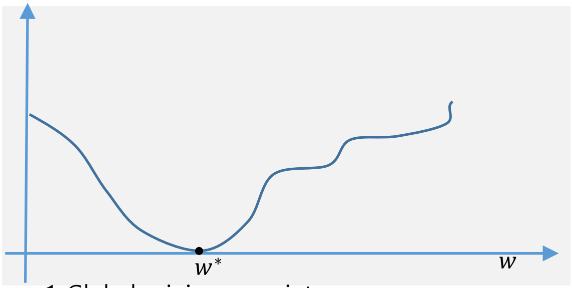

1 Global minimum point
No Local minimum points
GDA : good

$loss_{mse}(w)$ in Logistic Regression




1 Global minimum point
some Local minimum points
GDA : difficult

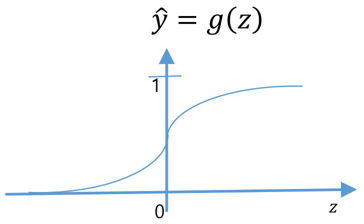
$loss_{ce}(w)$ in Logistic Regression



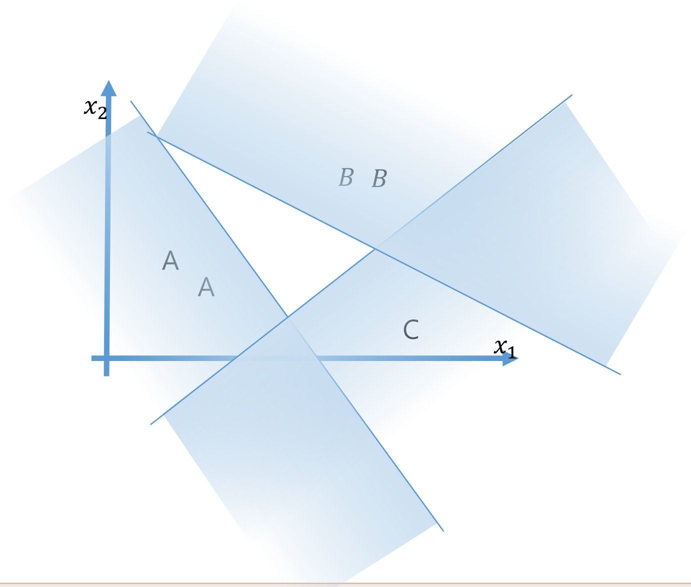

1 Global minimum point
No Local minimum points
GDA : good

# 2. 다중분류(1) – Logistic classifier

- How to classify 3 classes with logistic classifier?
  - Logistic regression
    - Model
      - $\hat{y} = h(x) = g(\hat{y}_l), g(z) = \frac{1}{1+e^{-z}}$ [0 ~1]
      - $\hat{y}_l = h_l(x) = wx + b,$ $[-\infty \sim +\infty]$
    - $loss(X, Y) = \frac{1}{N}\sum cross\_entropy(H_{Lo}(X_i), Y_i))$
  - How to classify X ?

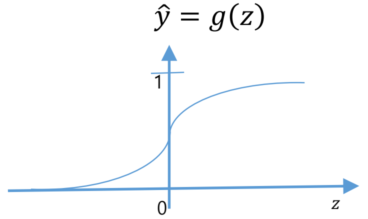



| X | | Y |
|---|---|---|
| x1 (hours) | x2 (attendance) | y (grade) |
| 10 | 5 | A |
| 9 | 5 | A |
| 3 | 2 | B |
| 2 | 4 | B |
| 11 | 1 | C |



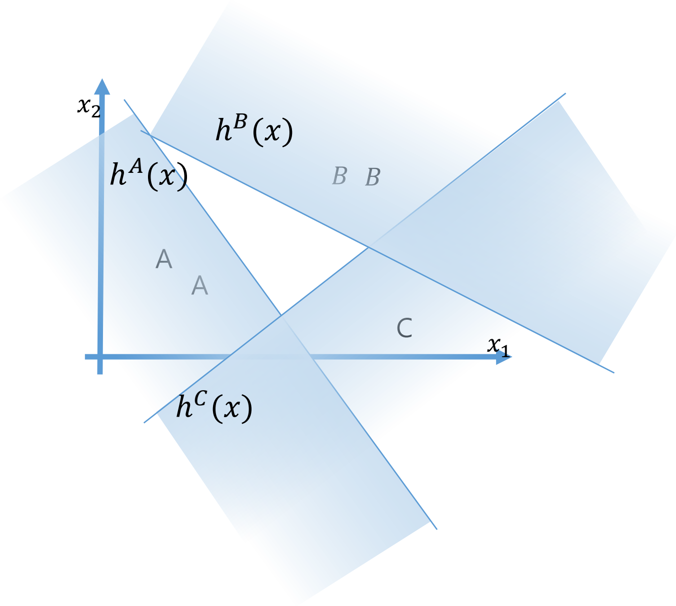
# 2. 다중분류(1)(cont.)

- Logistic regression
  - $\hat{y} = h(x) = g(\hat{y}_l), g(z) = \frac{1}{1+e^{-z}}$ [0 ~1]
    - $\hat{y}_l = wx + b,$ $[-\infty \sim +\infty]$
  - $loss(X, Y) = \frac{1}{N}\sum cross\_entropy(\hat{y}_i, y_i))$
- How to classify X ?



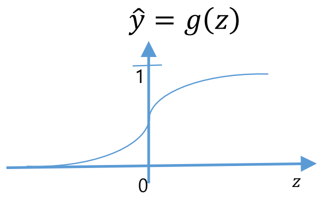

- Dataset 분리
  - $(X_A, Y_A), (X_B, Y_B), (X_C, Y_C)$
- 3의 로지스틱 회귀모델 생성 및 학습
  - $h^A(x)$ on $(X_A, 1)$ ,$(X_B, 0)$,$(X_C, 0)$
  - $h^B(x)$ on $(X_A, 0)$ ,$(X_B, 1)$,$(X_C, 0)$
  - $h^C(x)$ on $(X_A, 0)$ ,$(X_B, 0)$,$(X_C, 1)$
- 예측방법?
  - $\hat{y} = h(x)$ ?
  - if $h^A(x)$=1,$h^B(x)$=0,$h^C(x)$=0 : $x \in A$
  - if $h^A(x)$=0,$h^B(x)$=1,$h^C(x)$=0 : $x \in B$

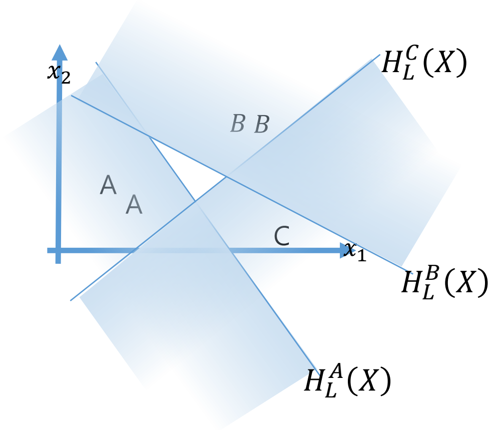
| X | | Y |
|---|---|---|
| x1 (hours) | x2 (attendance) | y (grade) |
| 10 | 5 | A |
| 9 | 5 | A |
| 3 | 2 | B |
| 2 | 4 | B |
| 11 | 1 | C |



# 2. 다중분류(1)(cont.)

- How to predict(calculate Hypothesis function) ?
  - $\hat{y} \;\; = h(X)$ ?
    - $\hat{y}_l = WX + B$
    - $\hat{y} = \mathrm{sig}(\hat{y}_l)$




$$\hat{y}_l = \begin{bmatrix} h^A(X) \\ h^B(X) \\ h^C(X) \end{bmatrix} = \begin{bmatrix} w_{A1} \; w_{A2} \\ w_{B1} \; w_{B2} \\ w_{C1} \; w_{C2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + b = \begin{bmatrix} w_{A1}x_1 + \; w_{A2}x_2 + b \\ w_{B1}x_1 \; + w_{B2}\,x_2 + b \\ w_{C1}x_1 + w_{C2}x_2 + b \end{bmatrix} = \begin{bmatrix} 2.0 \\ 0.3 \\ 0.1 \end{bmatrix}$$

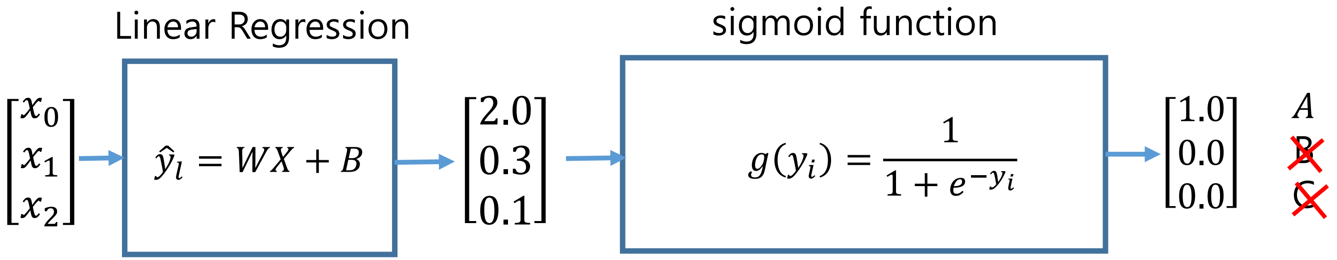
$$\hat{y} \;\; = \begin{bmatrix} \hat{y}^A \\ \hat{y}^B \\ \hat{y}^C \end{bmatrix} = g\left(\begin{bmatrix} h^A(X) \\ h^B(X) \\ h^C(X) \end{bmatrix}\right) \equiv \begin{bmatrix} 1.0 \\ 0.0 \\ 0.0 \end{bmatrix} \Rightarrow \begin{matrix} A \\ \text{\sout{B}} \\ \text{\sout{C}} \end{matrix}$$

$$g(z) = \frac{1}{1 + e^{-z}}$$

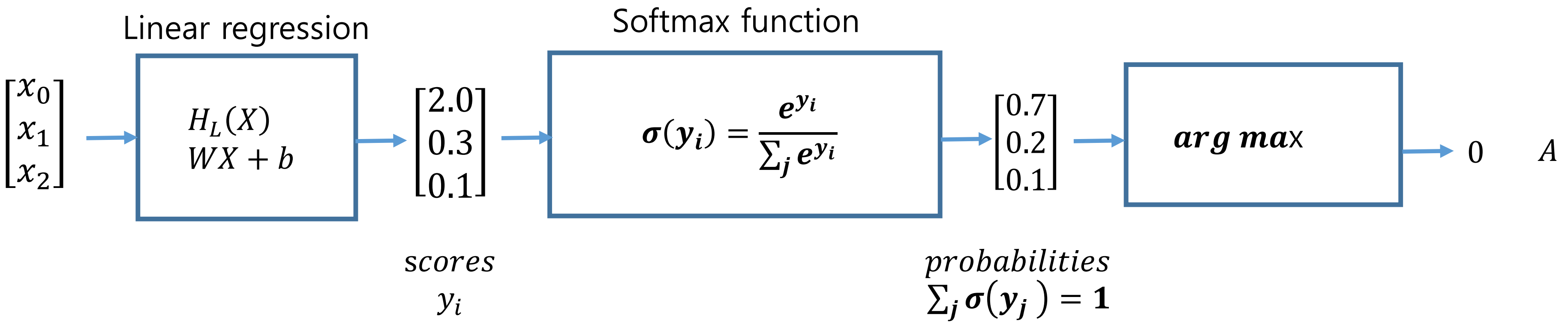
$cf.$ $LinearSVC()$ in sklearn

# 3. 다중분류(2) – Softmax classifier

- softmax 분류기와 3 로지스틱 분류기의 비교
  - 3중분류(로지스틱 회귀)




  - Softmax 3중분류



# 3. 다중분류(2) (cont.)

- 선형회귀, 로지스틱 분류와 softmax 분류모델의 수학적 해석

- **로지스틱 회귀모델, 이진분류**
  - 데이터셋 $X = \{x_i\}, Y = \{y_i\}$
  - 모델 $\hat{y} = h(x) = \sigma(\hat{y}_l), \sigma(z) = \frac{1}{1+e^{-z}}$ ,sigmoid

    $\hat{y}_l = wx + b$
  - 손실함수(binary_crossentropy)

    $loss(w,b) = -\frac{1}{m}\sum_{i=1}^{m}(ylog(\bar{y}_i) + (1-y)\log(1-\bar{y}_i))$
  - 학습 $w^*, b^* = argmin_{w,b}(loss(w,b))$
  - 예측 $\hat{y} = h(x) = \sigma(\hat{y}_l), \hat{y}_l = w^*x + b^*$
  - 평가지수 $accuracy = \frac{1}{N}\sum(y_i = \hat{y}_i)$

- **선형회귀 모델**
  - 데이터셋 $X = \{x_i\}, Y = \{y_i\}$
  - 모델 $\hat{y} = h(x) = wx + b$
  - 손실함수(mean square error)

    $loss(w,b) = \frac{1}{N}\sum_{i=1}^{N}(y_n - \hat{y}_i)^2$
  - 학습 $w^*, b^* = argmin_{w,b}(loss(w,b))$
  - 예측 $\hat{y} = h(x) = \sigma(\hat{y}_l), \hat{y}_l = w^*x + b^*$
  - 평가지수 $R^2 = 1 - \frac{\sum(y-\hat{y})^2}{\sum(y-\bar{y})^2}$

- **Softmax 분류모델**
  - 데이터셋 $X = \{x_i\}, Y = \{y_i\}$
  - 모델 $\hat{y} = h(x) = \sigma(\hat{y}_l), \sigma(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$, softmax

    $\hat{y}_l = wx + b$
  - 손실함수(categorical_crossentropy)

    $loss(w,b) = -\frac{1}{N}\sum_j y_j \log(\hat{y}_i)$
  - 학습 $w^*, b^* = argmin_{w,b}(loss(w,b))$
  - 예측 $\hat{y} = h(x) = \sigma(\hat{y}_l), \hat{y}_l = w^*x + b^*$
  - 평가지수 $accuracy = \frac{1}{N}\sum(y_i = \hat{y}_i)$

# 3. 다중분류(2) (cont.)

- sklearn 패키지로 3모델 구현

```
#Linear Regression Model
X        =np.array([[1.1, 2.3], [2.0, 3.6]]) ;  Y    =np.array([[10.25], [11.2]])
X_val    =np.array([[1.3, 1.8]]);               y_val=np.array([[10.2]])

model=Sequential()
model.add(  Dense(1                             #모델정의,output dim
                activation='linear'             #선형회귀모델
                input_dim=2)
model.compile(                                  #학습
            loss='mse'
            optimizer='adam')
y_hat=model.predict(np.array([[1.1, 1.7]]))#[[11.33]]#예측
```

```
#Logistic regression(Binary classification) Model
X        =np.array([[1, 2], [2, 3]]) ;     Y    =np.array([[0],[1]])
X_val    =np.array([[1, 1]]);              y_val=np.array([[0]])

model=Sequential()
model.add(  Dense(1,                            #모델정의,output dim
                activation='sigmoid',           #로지스틱회귀모델
                input_dim=2)
model.compile(                                  #학습
            loss='binary_crossentropy',
            optimizer='adam',
            metrics=['accuracy'])
y_hat    = model.predict(X)                     #예측, [[0.3] [0.4]]
Acc      = model.evaluate(X_val, y_val, verbose=0)[1]   # 평가정확도 0.7
```

```
#Multinomial(softmax) classification
X        = np.array([[1, 2, 1, 4],  [2, 1, 3, 5],  [3, 1, 3, 1]])
Y_ohe    = np.array([[0, 0, 1],  [0, 1, 0],  [1, 0, 0]])

model=Sequential()
model.add(Dense(3,                              #모델정의, output_dim
            activation='softmax',               #소프트맥스 분류모델
            input_dim=4)
model.compile(                                  #학습
        loss='categorial_crossentropy',
        optimizer='adam',
        metrics=['accuracy'])

Y_hat    = model.predict(np.array([[1,2,1,3]])) #예측
                                                #[[0.0188113  0.15894766 0.822241  ]]
Acc      = model.evaluate(X_val, y_val, verbose=0)[1]   #평가 정확도 0.7
```

# Examples

- Example 1, Simple softmax classifier
- Example 2, Fancy animal classification

# Example 1. Simple softmax classifier

- Simple softmax classifier
  - 제목
    - Softmax 분류기 작성
  - 목적
    - Softmax 분류기의 개발 방법을 습득한다.
  - 내용
    - Simple dataset으로 모델의 생성, 학습 및 성능분석하시오
  - 절차
    - 데이터셋 생성
    - 모델 정의
    - 모델 학습 방법 설정
    - 모델 학습하기
    - 모델 평가하기
    - 모델 사용하기

# Example 1. Simple softmax classifier

## 1.1 데이터셋 생성

- Y(label)데이터를 one_hot_encoding한다.
  - nb_class =3
  - 0 => 1 0 0
    1 => 0 1 0
    2 => 0 0 1

| x1 | x2 | x3 | x4 | Y |
|---|---|---|---|---|
| 1 | 2 | 1 | 1 | A |
| 2 | 1 | 3 | 2 | A |
| 3 | 1 | 3 | 4 | A |
| 4 | 1 | 5 | 5 | B |
| 1 | 7 | 5 | 5 | B |
| 1 | 2 | 5 | 6 | B |
| 1 | 6 | 6 | 6 | C |
| 1 | 7 | 7 | 7 | C |

| x1 | x2 | x3 | x4 | Y | Y1 | Y_ohe |
|---|---|---|---|---|---|---|
| 1 | 2 | 1 | 1 | A | 0 | 1 0 0 |
| 2 | 1 | 3 | 2 | A | 0 | 1 0 0 |
| 3 | 1 | 3 | 4 | A | 0 | 1 0 0 |
| 4 | 1 | 5 | 5 | B | 1 | 0 1 0 |
| 1 | 7 | 5 | 5 | B | 1 | 0 1 0 |
| 1 | 2 | 5 | 6 | B | 1 | 0 1 0 |
| 1 | 6 | 6 | 6 | C | 2 | 0 0 1 |
| 1 | 7 | 7 | 7 | C | 2 | 0 0 1 |

- 데이터 X,Y의 행렬표현과 코딩

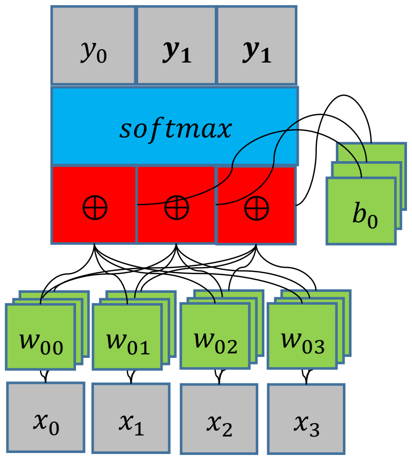
$$X = \begin{bmatrix} 1, & 2, & 1, & 1 \\ 2, & 1, & 3, & 2 \\ 3, & 1, & 3, & 4 \\ 4, & 1, & 5, & 5 \\ 1, & 7, & 5, & 5 \\ 1, & 2, & 5, & 6 \\ 1, & 6, & 6, & 6 \\ 1, & 7, & 7, & 7 \end{bmatrix} \quad Y= \begin{bmatrix} 0, & 0, & 1 \\ 0, & 0, & 1 \\ 0, & 0, & 1 \\ 0, & 1, & 0 \\ 0, & 1, & 0 \\ 0, & 1, & 0 \\ 1, & 0, & 0 \\ 1, & 0, & 0 \end{bmatrix} \quad class= \begin{bmatrix} A \\ A \\ A \\ B \\ B \\ B \\ C \\ C \end{bmatrix}$$

$$h(X)=WX+B= \begin{bmatrix} w_{11}, w_{12}, w_{13} \\ w_{21}, w_{22}, w_{23} \\ w_{31}, w_{32}, w_{33} \\ w_{41}, w_{42}, w_{43} \end{bmatrix} X + \begin{bmatrix} b_1, \\ b_2 \\ b_3 \end{bmatrix}$$

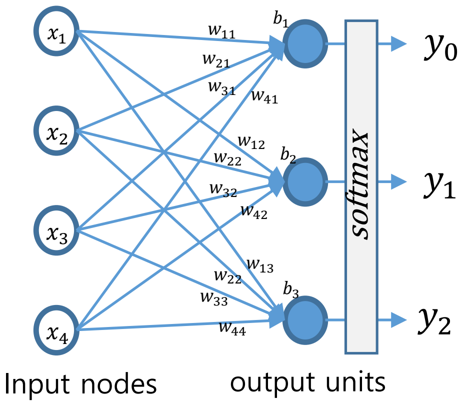
```
# 1. 데이터셋 생성하기
X = np.array([[1, 2, 1, 1],  [2, 1, 3, 2],  [3, 1, 3, 4],  [4, 1, 5, 5],  [1, 7, 5, 5], [1, 2, 5, 6],   [1, 6, 6, 6],  [1, 7, 7, 7]])
Y = np.array([[0, 0, 1],   [0, 0, 1],  [0, 0, 1],   [0, 1, 0],   [0, 1, 0],   [0, 1, 0],   [1, 0, 0],  [1, 0, 0]])
nb_classes=3
X,X_val,Y,Y_val=train_test_split(X,Y, random_state=0) #(6,4) (2,4)  (6,3) (2,3)
```

# Example 1. Simple softmax classifier(cont.)

## 1.2 모델의 생성




개념도(1)




개념도(2)

**Softmax classifier**

모델 (Softmax function)

$$\hat{y} = h(x) = g(\hat{y}_l),\ g(y_i) = \frac{e^{y_i}}{\sum_k e^{y_k}}$$

$$\hat{y}_l = wx + b$$

Loss function : catagorial_crossentropy

$$\text{loss}(X, Y) = -\frac{1}{N}\sum_j y_j \log(\hat{y}_i)$$

Gradient descent algorithm

$$W_{t+1} = W_t - \alpha \frac{\partial}{\partial W} loss(W|X, Y)|_{W=W_t}$$

```
X = np.array(
    [[1, 2, 1, 1],
     [2, 1, 3, 2],
     …
    ])
```

```
Y = np.array(
    [[0, 0, 1],
     [0, 0, 1],
     …
    ])
```

$$X = \begin{bmatrix} 1, & 2, & 1, & 1 \\ 2, & 1, & 3, & 2 \\ 3, & 1, & 3, & 4 \\ 4, & 1, & 5, & 5 \\ 1, & 7, & 5, & 5 \\ 1, & 2, & 5, & 6 \\ 1, & 6, & 6, & 6 \\ 1, & 7, & 7, & 7 \end{bmatrix} \quad Y = \begin{bmatrix} 0, & 0, & 1 \\ 0, & 0, & 1 \\ 0, & 0, & 1 \\ 0, & 1, & 0 \\ 0, & 1, & 0 \\ 0, & 1, & 0 \\ 1, & 0, & 0 \\ 1, & 0, & 0 \end{bmatrix}$$

- 모델의 구현

nb_classes=3

X.shape[1]=4

```
# 2. 모델 구성하기
model=Sequential(name='Softmax_Sample_Classification')
model.add(
    Dense(units=3,                 #클래스의 수
          input_dim=4,             #입력 특징의 수,입력크기
          activation='softmax’)) #활성화함수
```

# Example 1. Simple softmax classifier(cont.)

## 1.3 모델의 학습방법설정

**Softmax classifier**

모델 (Softmax function)

$$\hat{y} = h(x) = g(\hat{y}_l),\ g(y_i) = \frac{e^{y_i}}{\sum_k e^{y_k}}$$

$$\hat{y}_l = wx + b$$

모델 학습

Loss function : catagorial_crossentropy

$$\text{loss}(X, Y) = -\frac{1}{N}\sum_j y_j \log(\hat{y}_i)$$

학습방법(optimizer)

Gradient descent algorithm

$$W_{t+1} = W_t - \alpha \frac{\partial}{\partial W} loss(W|X,Y)|_{W=W_t}$$

Stochastic Gradient Descent (sgd)

Gradient Descent Optimization Algorithms [link]

평가지수(Metrics)

Accuracy

```
# 2. 모델 구성하기
model=Sequential(name='Softmax_Sample_Classification')
model.add(
      Dense(units=3,                  #클래스의 수
            input_dim=4,              #입력 특징의 수,입력크기
            activation='softmax'))    #활성화함수

# 3. 모델 학습방법 설정하기
model.compile(
        loss='categorical_crossentropy', #범주형 엔트로피
        optimizer= ' sgd',               #학습방법(최적화기)
        metrics=['accuracy'])            #평가지수
model.summary()                          #모델구조 요약 출력
```

# Example 1. Simple softmax classifier(cont.)

## 1.4 모델의 학습

- 학습데이터 설정 : X,Y
- 데이터의 반복학습 횟수 : epochs=1000
- 학습정보 출력 량 verbose = 0,1,2
- 매회 데이터 학습을 마치고 모델을 평가하여 학습되 정도를 확인하기 위하여 평가방법을 지정한다.
  validation=[‘accuracy’]

## 1.5 모델의 평가

- model.evaulate(X,Y)
  모델학습과정에서 지정한 평가방법으로 평가 결과를 반환한다.

## 1.6 데이터에 대한 모델예측

- model.predict(X[0:1])
  데이터를 입력하여 모델이 예측한 값을 반환한다.

```
# 4. 모델 학습시키기
hist=model.fit(X,Y,                              # 학습데이터 설정
          epochs=200,                            # 학습반복횟수
          verbose=1,                             # 출력량 모드
          validation_data=(X_val, Y_val))        #검증용 데이터셋

# 5. 학습과정 살펴보기
print('\nfitted-history :')
print('\tacc_max:{:.2f},\tval_acc_max:{:.2f},\tloss_min:{:.6f},\tval_loss_min:{:.6f}'.format(
      max(hist.history['accuracy’]), max(hist.history['val_accuracy']),
      min(hist.history['loss']),max(hist.history['val_loss'])))

# 6. 모델 평가하기
print('\nmodel.evaluate(X_val, Y_val): ')
loss_and_acc = model.evaluate(X_val, Y_val,verbose=0)   #평가데이터 정확도
print('\tloss and_acc :', loss_and_acc)

# 7. 모델 사용하기
y_hat = model.predict(X[0:1])                           #예측
print('\ny_hat=model.predict(X[0:1])')
print('\tX[0:1] : ',X[0:1])
print('\ty_hat:   ',y_hat)
print('\targamx(yhat) : ',np.argmax(y_hat,axis=1))

yhat = model.predict(X[1:4])
print('\ny_hat=nmodel.predict(X[1:4])')
print('\tX[1:4] : ',X[1:4])
print('\ty_hat       : ',y_hat)
print('\targamx(yhat) : ',np.argmax(y_hat,axis=1))
```

# Example 1. Simple softmax classifier(cont.)

- 전체 코드

```
from keras.models import Sequential
from keras.layers import Dense
from sklearn.model_selection import train_test_split
import numpy as np

# 1. 데이터셋 생성하기
X = np.array([[1, 2, 1, 1],  [2, 1, 3, 2],  [3, 1, 3, 4],  [4, 1, 5, 5],  [1, 7, 5, 5], [1, 2, 5, 6],
              [1, 6, 6, 6],   [1, 7, 7, 7]])
Y = np.array([[0, 0, 1],   [0, 0, 1],  [0, 0, 1],   [0, 1, 0],   [0, 1, 0],   [0, 1, 0],   [1, 0, 0],  [1, 0, 0]])
nb_classes=3
X,X_val,Y,Y_val=train_test_split(X,Y,random_state=0)
#(6,4) (2,4)  (6,3) (2,3)

# 2. 모델 구성하기
model=Sequential(name='Softmax_Sample_Classification')
model.add(Dense(units=nb_classes,              #클래스 수
                input_dim=X.shape[1],          #입력 데이터 사이즈
                activation='softmax’))         #활성화 함수

# 3. 모델 학습방법 설정하기
model.compile(
          loss= ' categorical_crossentropy’,  # 손실함수 계산방법 설정
          optimizer= ' sgd’,                  #학습방법
          metrics=[ ' accuracy’])             #평가지수
model.summary()                               #모델의 구조 출력
```

```
# 4. 모델 학습시키기
hist=model.fit(X,Y,                                 #학습데이터
               epochs=200,                          #데이터의 반복학습 횟수
               verbose=1,                           #학습과정 출력모드
               validation_data=(X_val, Y_val))      #평가데티터셋의정확도

# 5. 학습과정 살펴보기
print('\nfitted-history :')
print('\tacc_max:{:.2f},\tval_acc_max:{:.2f},\tloss_min:{:.6f},\tval_loss_min
:{:.6f}'.format(
max(hist.history['accuracy']),max(hist.history['val_accuracy']),
min(hist.history['loss']),max(hist.history['val_loss'])))

# 6. 모델 평가하기
print('\nmodel.evaluate(X_val, Y_val): ')
loss_and_acc = model.evaluate(X_val, Y_val,verbose=0)
print('\tloss and_acc :', loss_and_acc)

# 7. 모델 사용하기
y_hat = model.predict(X[0:1])
print('\ny_hat=model.predict(X[0:1])')
print('\tX[0:1] : ',X[0:1])
print('\ty_hat:   ',y_hat)
print('\targamx(yhat) : ',np.argmax(y_hat,axis=1))

y_hat = model.predict(X[1:4])
print('\ny_hat=nmodel.predict(X[1:4])')
print('\tX[1:4] : ',X[1:4])
print('\ty_hat        : ',y_hat)
print('\targamx(yhat) : ',np.argmax(y_hat,axis=1))
```

# Example 1. Simple softma

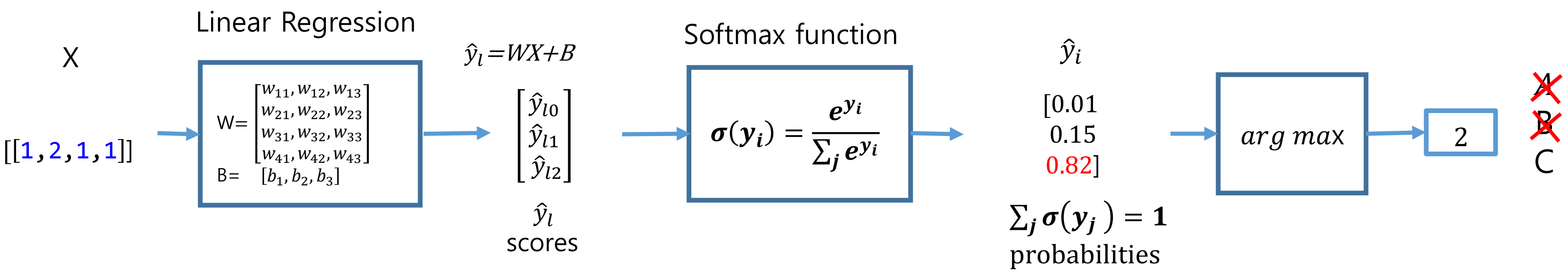
```
from keras.models import Sequential
from keras.layers import Dense
from sklearn.model_selection import train_test_split
import numpy as np

# 1. 데이터셋 생성하기
X = np.array([[1, 2, 1, 1],  [2, 1, 3,
Y = np.array([[0, 0, 1],   [0, 0, 1],
nb_classes=3
X,X_val,Y,Y_val=train_test_split(
#(6,4) (2,4)  (6,3) (2,3)

# 2. 모델 구성하기
model=Sequential(name='Softmax
model.add(Dense(units=nb_classes
                input_dim=X.sha
                activation='softm

# 3. 모델 학습방법 설정하기
model.compile(
          loss= ' categorical_cro
          optimizer= ' sgd',
          metrics=[ ' accuracy']]
model.summary()

# 4. 모델 학습시키기
hist=model.fit(X,Y,
          epochs=200,
          verbose=1,
          validation_data=(X_va
```

```
# 5. 학습과정 살펴보기
print('\nfitted-history :')
print('\tacc_max:{:.2f},\tval_acc_max
:{:.6f}'.format(
max(hist.history['accuracy']),max(his
min(hist.history['loss']),max(hist.histo

# 6. 모델 평가하기
print('\nmodel.evaluate(X_val, Y_val
loss_and_acc = model.evaluate(X_va
print('\tloss and_acc :', loss_and_acc)

# 7. 모델 사용하기
y_hat = model.predict(X[0:1])
print('\ny_hat=model.predict(X[0:1])'
print('\tX[0:1] : ',X[0:1])
print('\ty_hat:   ',y_hat)
print('\targamx(yhat) : ',np.argmax(y_

yhat = model.predict(X[1:4])
print('\ny_hat=nmodel.predict(X[1:4]
print('\tX[1:4] : ',X[1:4])
print('\ty_hat        : ',y_hat)
print('\targamx(yhat) : ',np.argmax(y_
```

```
Model: "Softmax_Model_Example"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 3)                 15
=================================================================
Total params: 15
Trainable params: 15
Non-trainable params: 0
_________________________________________________________________

Train on 6 samples, validate on 2 samples
Epoch 1/500
6/6 [==============================] - 0s 41ms/step - loss: 2.4977 - accuracy: 0.3333 - val_loss: 3.4202 - val_accuracy: 0.5000
Epoch 2/500
6/6 [==============================] - 0s 1ms/step - loss: 2.3945 - accuracy: 0.3333 - val_loss: 3.3331 - val_accuracy: 0.5000
Epoch 3/500
6/6 [==============================] - 0s 831us/step - loss: 2.3006 - accuracy: 0.3333 - val_loss: 3.2460 - val_accuracy: 0.5000

Epoch 498/500
6/6 [==============================] - 0s 997us/step - loss: 0.5423 - accuracy: 0.6667 - val_loss: 1.4775 - val_accuracy: 0.0000e+00
Epoch 499/500
6/6 [==============================] - 0s 14ms/step - loss: 0.5419 - accuracy: 0.6667 - val_loss: 1.4776 - val_accuracy: 0.0000e+00
Epoch 500/500
6/6 [==============================] - 0s 831us/step - loss: 0.5415 - accuracy: 0.6667 - val_loss: 1.4776 - val_accuracy: 0.0000e+00
X.shape:(6, 4) X_val.Shape:(2, 4)

fitted-history :
        acc_max:0.67,   val_acc_max:0.50,       loss_min:0.541481,      val_loss_min:3.420193

model.evaluate(X_val, Y_val):
loss and_acc : [1.4776464700698853, 0.0]

y_hat=model.predict(X[0:1])
        X[0:1]  :  [[2 1 3 2]]
        y_hat   :  [[0.1172336  0.28952873 0.59323764]]
 argamx(yhat) :  [2]

y_hat=nmodel.predict(X[1:4])
        X[1:4]  :  [[1 7 7 7]   [4 1 5 5]   [1 2 1 1]]
y_hat           :  [[0.446096   0.51772845 0.0361755 ]
                    [0.02800328 0.8031788  0.16881788]
                    [0.17695168 0.31350178 0.5095466 ]]
 argamx(yhat) :  [1 1 2]
```

# Example 1. Simple softmax classifier(cont.)

- Softmax 모델  예측 개념 – 한 샘플의 예측
  - $\hat{y} = \mathrm{h}(x) = h([1, 2, 1, 1])$ ?




$\hat{y} = h(X[0:1])$
$= h([[1, 2, 1, 1]])$
$= argmax(model.predict([[1, 2, 1, 1]]))$
$= argmax(softmax(W[[1, 2, 1, 1]]+B)$
$= argmax([0.0188113 \;\; 0.15894766 \;\; 0.822241])$
$=2$

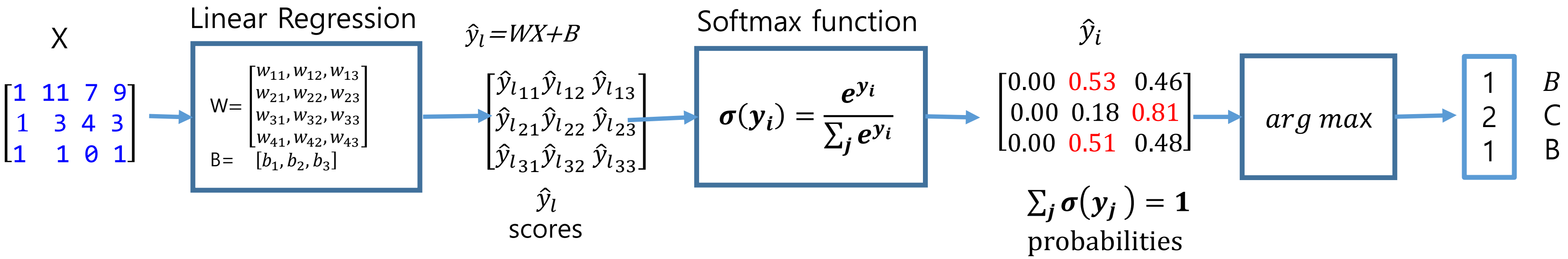
```
# 7. 모델 사용하기
yhat = model.predict(X[0:1])
print('X[0:1] : ',X[0:1])                              #[[1, 2, 1, 1]]
print('yhat  = h(X[:1])  = {}'.format(yhat))  #[0.01 0.15 0.82]
print('yhat1 = argamx(yhat) = {}'.format(np.argmax(yhat))) #2
```

```
X[0:1] :                    =  [[1 2 1 1]]
yhat    = H(X[0:1])         = [[0.0188113  0.15894766 0.822241  ]]
yhat1   = argamx(yhat,1) = 2
```

# Example 1. Simple softmax classifier(cont.)

- 모델의 예측처리 개념 –복수샘플(3)의 예측

$$\hat{y} = h(X) = h\begin{pmatrix} [[2\ 1\ 3\ 2] \\ [3\ 1\ 3\ 4] \\ [4\ 1\ 5\ 5]] \end{pmatrix}\ ?$$




$\hat{y} = h(X[:3])$

$$=argmax(model.predict\begin{pmatrix} [[2\ 1\ 3\ 2] \\ [3\ 1\ 3\ 4] \\ [4\ 1\ 5\ 5]] \end{pmatrix})$$

$$= argmax(softmax(W \begin{matrix} [[2\ 1\ 3\ 2] \\ [3\ 1\ 3\ 4] \\ [4\ 1\ 5\ 5]] \end{matrix} +B))$$

$= argmax$([[1.0841307e−03 5.3002125e−01 4.6889463e−01]
[3.4792713e−05 1.8826017e−01 8.1082493e−01]
[1.2−05 5.1698810e−01 4.8297709e−01]])
= [1,2,1]

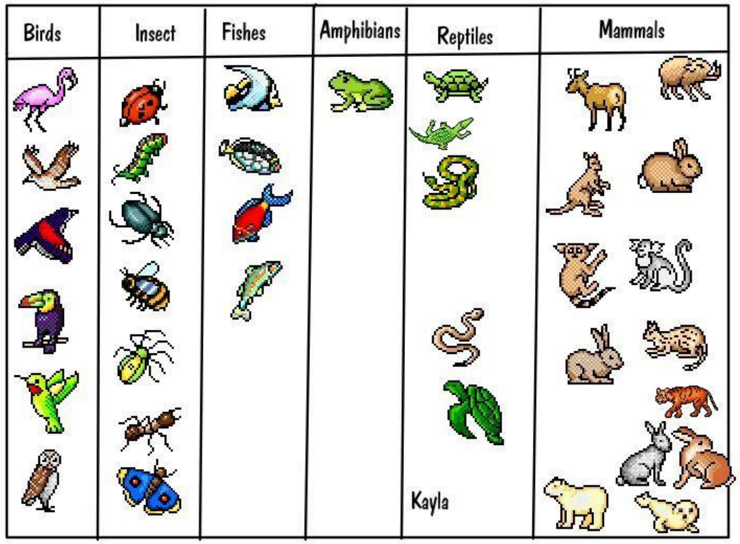
```
# 7. 모델 사용하기
yhat = model.predict(X[:3])
print('X[0:1] : ',X[0:1])
print('yhat  = h(X[:1])  = {}'.format(yhat))
print('yhat1 = argamx(yhat) = {}'.format(np.argmax(yhat)))
```

```
X[1:4]  :      [[2 1 3 2]
                [3 1 3 4]
                [4 1 5 5]]
yhat  = H(X[1:4])   =   [[1.0841307e-03 5.3002125e-01 4.6889463e-01]
                         [3.4792713e-05 .1496942e-04 1.8826017e-01 8.1082493e-01]
                         [1.2-05 5.1698810e-01 4.8297709e-01]]
yhat1 = argamx(yhat ,axis=1)= [1 2 1]
```

# Example 2. Fancy animal classification

- 제목
  - 동물 종 분류기
- 내용
  - 동물의 17가지 특성의 데이터 셋에 대하여 7 종 동물을 판별하는 softmax classifier를 작성한다.
- 절차
  1. 데이터셋 생성
     - 데이터(data-04-zoo.csv)를 다운로드하고
     - DeepLearningZeroToAll
       - data-04-zoo.csv
  2. Softmax 모델의 정의
  3. 학습방법 설정
  4. 학습하기
  5. 모델 평가
     1. 분류실험을 하여 개발한 모델의 분류성능을 분석하시오.

# Example 2. Fancy animal classification




## 2.1 데이터 셋 생성

- Data-04-zoo.txt의 분석

```
# https://archive.ics.uci.edu/ml/machine-learning-databases/zoo/zoo.data,,,,,,,,,,,,,,,,
#    1. animal name:      (deleted),,,,,,,,,,,,,,,,
#    2. hair       Boolean",,,,,,,,,,,,,,,,
#    3. feathers       Boolean",,,,,,,,,,,,,,,,
#    4. eggs       Boolean",,,,,,,,,,,,,,,,
#    5. milk       Boolean",,,,,,,,,,,,,,,,
#    6. airborne       Boolean",,,,,,,,,,,,,,,,
#    7. aquatic       Boolean",,,,,,,,,,,,,,,,
#    8. predator       Boolean",,,,,,,,,,,,,,,,
#    9. toothed       Boolean",,,,,,,,,,,,,,,,
#   10. backbone       Boolean",,,,,,,,,,,,,,,,
#   11. breathes       Boolean",,,,,,,,,,,,,,,,
#   12. venomous       Boolean",,,,,,,,,,,,,,,,
#   13. fins       Boolean",,,,,,,,,,,,,,,,
#   14. legs       Numeric (set of values: {0",2,4,5,6,8}),,,,,,,,,,,
#   15. tail       Boolean",,,,,,,,,,,,,,,,
#   16. domestic       Boolean",,,,,,,,,,,,,,,,
#   17. catsize       Boolean",,,,,,,,,,,,,,,,
#   18. type       Numeric (integer values in range [0",6]),,,,,,,,,,,,,,
1,0,0,1,0,0,1,1,1,1,0,0,4,0,0,1,0
1,0,0,1,0,0,0,1,1,1,0,0,4,1,0,1,0
0,0,1,0,0,1,1,1,1,0,0,1,0,1,0,0,3
1,0,0,1,0,0,1,1,1,1,0,0,4,0,0,1,0
1,0,0,1,0,0,1,1,1,1,0,0,4,1,0,1,0
...
```

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 4 | 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 3 |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 4 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 4 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 4 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 4 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 3 |
| 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 3 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 4 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 4 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 2 | 1 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 3 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 6 |
| 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 6 |
| 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 6 | 0 | 0 | 0 | 6 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 2 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 4 | 1 | 0 | 1 | 0 |

*X*      *Y*

```
XY=np.loadtxt('data/data-04-zoo-1.csv',delimiter=',',dtype='float')  #(101,17)
X=XY[:,:-1]                  #(101,16)
Y=XY[:,[-1]].astype(int) #(101,1)
```

# Example 2. Fancy animal classification

- 데이터 셋 생성

X

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 4 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 4 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 4 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 4 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 4 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 4 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 4 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 2 | 1 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 6 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 2 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 4 | 1 | 0 | 1 |

| Y |
|---|
| 0 |
| 3 |
| 0 |
| 0 |
| 0 |
| 0 |
| 3 |
| 3 |
| 0 |
| 0 |
| 1 |
| 3 |
| 6 |
| 6 |
| 6 |
| 1 |
| 0 |

Y_ohe

```
1.  0.  0.  0.  0.  0.  0
0.  0.  0.  1   0.  0.  0.
1.  0.  0.  0.  0.  0.  0.
1.  0.  0.  0.  0.  0.  0.
1.  0.  0.  0.  0.  0.  0.
1.  0.  0.  0.  0.  0.  0.
0.  0.  0.  1.  0.  0.  0.
0.  0.  0.  1.  0.  0.  0.
1.  0.  0.  0.  0.  0.  0.
1.  0.  0.  0.  0.  0.  0.
0.  1.  0.  0.  0.  0.  0.
0.  0.  0.  1.  0.  0.  0.
0.  0.  0.  0.  0.  0.  1.
0.  0.  0.  0.  0.  0.  1.
0.  0.  0.  0.  0.  0.  1.
......
```

```
# 1. 데이터셋 생성하기
XY=np.loadtxt('data/data-04-zoo-1.csv',delimiter=',',dtype='float')  #(101,17)   다운로드
X=XY[:,:-1]                                    #(101,16)  16열까지 슬라이싱하여 X 생성
Y=XY[:,[-1]].astype(int)                       #(101,1)    17열로label Y를 생성

nb_classes = np.unique(Y).size                 #7          클래스의 수를 계산
Y_ohe=np_utils.to_categorical(Y,nb_classes)    #(101,7)  label y를 one_hot_encoding한다.
#Y_ohe=np.eye(nb_classes)[Y.flatten()]         #(101,7)

X,X_val,Y,Y_val=train_test_split(X,Y_ohe,random_state=0) #(75,16) (26,16) (75,7) (26,7)
```

```
Y=[[0],[3],[0],...]

Y_ohe=one_hot_encoding(Y)
   =[[1,0,0,0,0,0,0],
     [0,0,0,1,0,0,0]
     [1,0,0,0,0,0,0]
      ....]
```

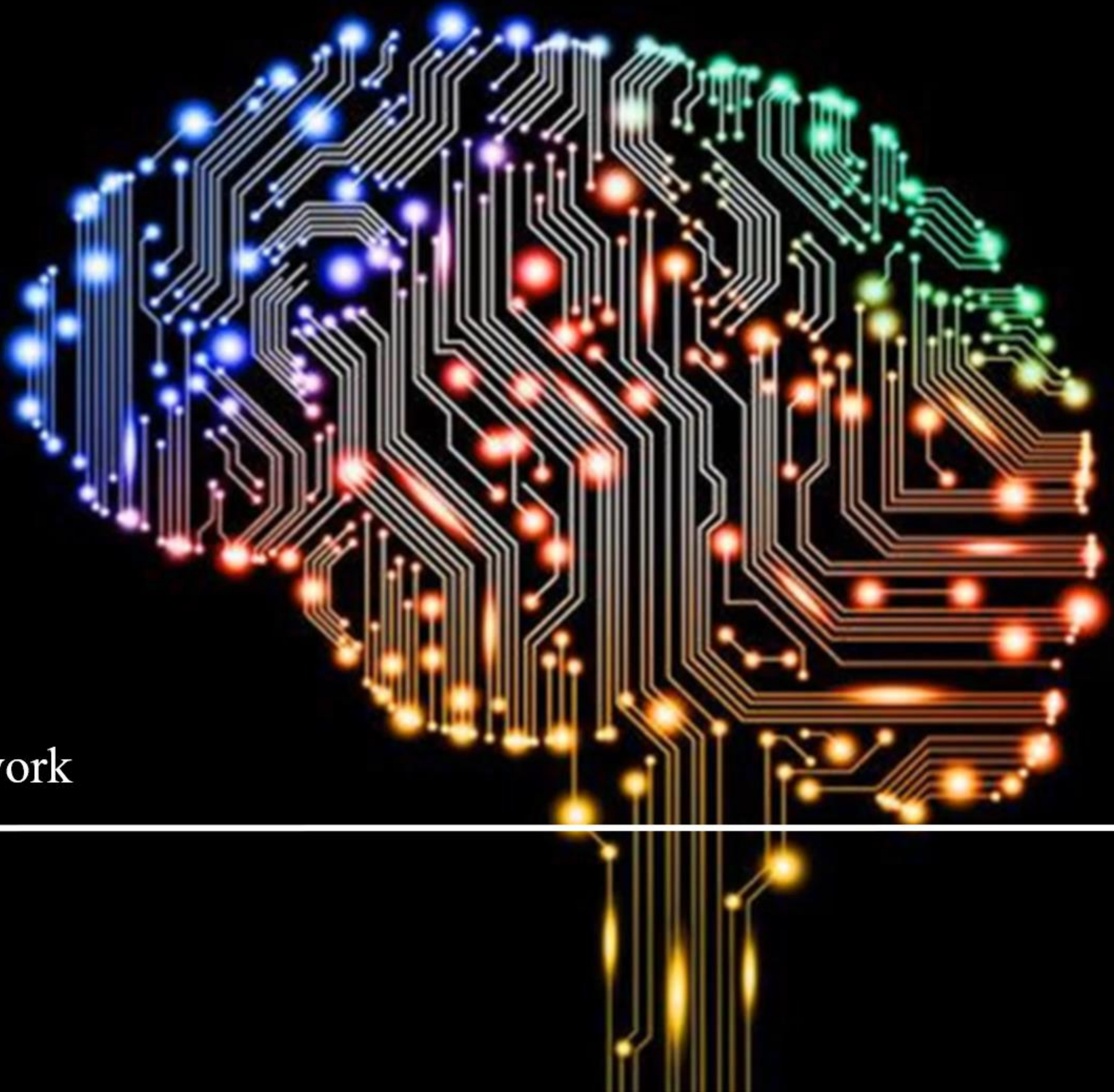
```
from keras.models import Sequential
from keras.layers import Dense
from sklearn.model_selection import train_test_split
from keras.utils import np_utils
import numpy as np
```

# Example 2. Fancy anir

다음의 과정을 작성하고 생행하여 오른쪽과 같은 출력을 얻고 출력내용을 분석하시오.

1. 데이터셋 생성
2. 모델 구성
3. 모델학습방법 설정
4. 모델학습
5. 학습과정 분석
6. 모델평가하기
7. 모델사용하기

```
Model: "Softmax_Fancy_Animal_Classification"
_________________________________________________________________
Layer (type)             Output Shape             Param #
=================================================================
dense_1 (Dense)             (None, 7)                119
=================================================================
Total params: 119
Trainable params: 119
Non-trainable params: 0
_________________________________________________________________

Train on 75 samples, validate on 26 samples
Epoch 1/200
75/75 [==============================] - 0s 4ms/step - loss: 1.8031 - accuracy: 0.4267 - val_loss: 1.7999 - val_accuracy: 0.4231
Epoch 2/200
75/75 [==============================] - 0s 2ms/step - loss: 1.7656 - accuracy: 0.4000 - val_loss: 1.7715 - val_accuracy: 0.4231
Epoch 3/200
75/75 [==============================] - 0s 292us/step - loss: 1.7372 - accuracy: 0.4000 - val_loss: 1.7407 - val_accuracy: 0.4231

Epoch 199/200
75/75 [==============================] - 0s 3ms/step - loss: 0.6282 - accuracy: 0.8667 - val_loss: 0.6113 - val_accuracy: 0.8846
Epoch 200/200
75/75 [==============================] - 0s 1ms/step - loss: 0.6268 - accuracy: 0.8667 - val_loss: 0.6098 - val_accuracy: 0.8846

fitted-history :
    acc_max:0.87,   val_acc_max:0.88,    loss_min:0.626761,    val_loss_min:1.799907

model.evaluate(X_val, Y_val):
    loss and_acc : [0.6097744703292847, 0.8846153616905212]

y_hat=model.predict(X[0:1])
    X[0:1]  :  [[1. 0. 0. 1. 0. 0. 0. 1. 1. 1. 0. 0. 4. 1. 1. 1.]]
    y_hat   :  [[0.9215614  0.02619649 0.0063446  0.00196954 0.00482013 0.02501752  0.01409036]]
    argamx(yhat) :  [0]

y_hat=nmodel.predict(X[1:4])
    X[1:4]  :  [[0. 1. 1. 0. 1. 0. 0. 0. 1. 1. 0. 0. 2. 1. 0. 0.]
                [0. 0. 1. 0. 0. 1. 0. 1. 1. 0. 0. 1. 0. 1. 0. 0.]
                [0. 0. 0. 0. 0. 1. 1. 1. 1. 0. 1. 0. 0. 1. 0. 0.]]
    y_hat   :  [[0.9215614  0.02619649 0.0063446  0.00196954 0.00482013 0.02501752   0.01409036]]
    argamx(yhat) :  [0]
```



# Deep Learning
# Deep Neural Network

Yoon Joong Kim,
Hanbat National University