

# Classification

4. Classification.ipynb

# How machine learns from data?

- Define a function to be learned:  $y = f(x)$
- Define a loss function  $\mathcal{L}$  to describe the error between  $\hat{y} = f(x)$  and  $y$
- Find the optimal parameters of  $f$  that minimize  $\mathcal{L}$

# How machine learns from data?

$$L = \frac{1}{N} \sum_{i=1}^N (y^i - \hat{y}^i)^2$$

- Regression
- Classification

$$y = f(x)$$

$$L = \frac{1}{N} \sum_{i=1}^N CE(p, \hat{p})$$

$$CE(p, \hat{p}) = - \sum_{k=1}^C p_k \ln(\hat{p}_k)$$

# Cross entropy

Measures the differences between the true probability  $p_i$  and the predicted probability  $\hat{p}_i$

$$CE(p, \hat{p}) = - \sum_{k=1}^C p_k \ln(\hat{p}_k)$$

Use Excel to verify  
Excel formula: =LN(x)

動物	實際機率分佈	預測機率分佈	Entropy
Cat	0%	2%	$0\% * -\log(2\%) = 0$
Dog	0%	30%	$0\% * -\log(30\%) = 0$
Fox	0%	45%	$0\% * -\log(45\%) = 0$
Cow	0%	0%	$0\% * -\log(0\%) = 0$
Red Panda	100%	25%	$100\% * -\log(25\%) = 1.386$
Bear	0%	5%	$0\% * -\log(5\%) = 0$
Dolphin	0%	0%	$0\% * -\log(0\%) = 0$
總計: cross-entropy = 1.386			

# Entropy

More information → more uncertain → larger entropy

$$Entropy = - \sum_i p_i \log_2(p_i)$$



$$\begin{aligned} &75\% \times 0.41 \\ &+ 25\% \times 2 \\ &= 0.81 \text{ bits} \end{aligned}$$



Calculate the entropy:


Sun shine: 50%


Cloudy: 25%

Rain: 25%

# Softmax

```
tensor([[ -0.0180,  0.0855],  
        [-0.0244,  0.0741],  
        [-0.0187,  0.0850],  
        [-0.0258,  0.0687],  
        [-0.0267,  0.0617]],
```

0: 

1: 

`torch.softmax(tensor, 1)`



```
: # apply softmax  
tensorY = torch.softmax(tensorY, 1)  
print(tensorY.shape, "\n", tensorY)
```

$\frac{e^{y_1}}{e^{y_1} + e^{y_2}}$  ← `torch.Size([5, 2])`  
`tensor([[0.4742, 0.5258],`  
          `[0.4754, 0.5246],`  
          `[0.4741, 0.5259],`  
          `[0.4764, 0.5236],`  
          `[0.4779, 0.5221]], device='cu`

→  $\frac{e^{y_2}}{e^{y_1} + e^{y_2}}$

# Torch.max

```
tensor([[0.4742, 0.5258],  
        [0.4754, 0.5246],  
        [0.4741, 0.5259],  
        [0.4764, 0.5236],  
        [0.4779, 0.5221]],
```

0:   
1: 

`torch.max(tensor, 1)`

```
In [17]: MaxOfEachRow = torch.max(tensorY, 1)  
print(MaxOfEachRow)  
  
torch.return_types.max(  
  values=tensor([0.5258, 0.5246, 0.5259, 0.5236, 0.5221], device='c  
    grad_fn=<MaxBackward0>),  
  indices=tensor([1, 1, 1, 1, 1], device='cuda:0'))
```

`torch.max(tensor, 1)[1]`

# Classification

## 4. Classification.ipynb

1.  $x_{1,i}$  and  $x_{2,i}$  are sampled from the same distribution
2. One attribute is from different distribution
3.  $x_{1,i}$  and  $x_{2,i}$  are sampled from the close distribution



# Bayesian's rule to understand $y_1 = p(C_1|x)$

$$y_1 = P(C_1|x) = \frac{P(x|C_1)P(C_1)}{P(x|C_1)P(C_1) + P(x|C_2)P(C_2)}$$

Generative Model  $P(x) = P(x|C_1)P(C_1) + P(x|C_2)P(C_2)$

# Probabilistic generative model

$$f_{\mu^1, \Sigma^1}(x) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma^1|^{1/2}} \exp \left\{ -\frac{1}{2} (x - \mu^1)^T (\Sigma^1)^{-1} (x - \mu^1) \right\}$$

$$\mu^1 = \begin{bmatrix} 75.0 \\ 71.3 \end{bmatrix} \quad \Sigma^1 = \begin{bmatrix} 874 & 327 \\ 327 & 929 \end{bmatrix}$$

$$P(C_1) = 79 / (79 + 61) = 0.56$$

$$P(C_1|x) = \frac{P(x|C_1)P(C_1)}{P(x|C_1)P(C_1) + P(x|C_2)P(C_2)}$$

$$f_{\mu^2, \Sigma^2}(x) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\Sigma^2|^{1/2}} \exp \left\{ -\frac{1}{2} (x - \mu^2)^T (\Sigma^2)^{-1} (x - \mu^2) \right\}$$

$$\mu^2 = \begin{bmatrix} 55.6 \\ 59.8 \end{bmatrix} \quad \Sigma^2 = \begin{bmatrix} 847 & 422 \\ 422 & 685 \end{bmatrix}$$

$$P(C_2) = 61 / (79 + 61) = 0.44$$

$$\text{If } P(C_1|x) > 0.5$$

Assuming  $x^n$  are sampled from a Gaussian distribution, then we can use maximum likelihood to find the best Gaussian distribution behind them.

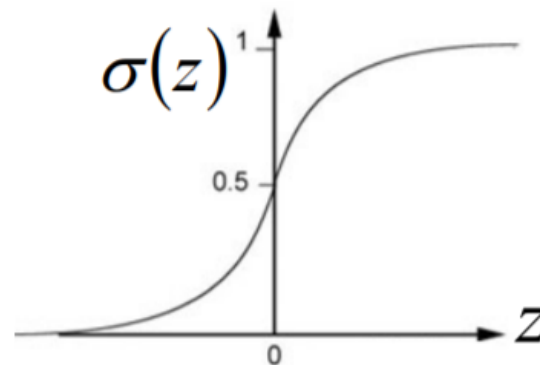
Represent  $y_1 = p(C_1|x)$  as a sigmoid function of  $z$

$$y_1 = P(C_1|x) = \frac{P(x|C_1)P(C_1)}{P(x|C_1)P(C_1) + P(x|C_2)P(C_2)}$$

$$= \frac{1}{1 + \frac{P(x|C_2)P(C_2)}{P(x|C_1)P(C_1)}} = \frac{1}{1 + \exp(-z)} = \sigma(z)$$

Sigmoid function

$$z = \ln \frac{P(x|C_1)P(C_1)}{P(x|C_2)P(C_2)}$$



# Represent $y_1 = p(C_1|x)$ as a sigmoid function of $z$

$$P(C_1|x) = \sigma(z)$$

Assuming the covariance matrices of the two classes are the same

$$z = \ln \frac{|\Sigma^2|^{1/2}}{|\Sigma^1|^{1/2}} - \frac{1}{2} x^T (\Sigma^1)^{-1} x + (\mu^1)^T (\Sigma^1)^{-1} x - \frac{1}{2} (\mu^1)^T (\Sigma^1)^{-1} \mu^1 + \frac{1}{2} x^T (\Sigma^2)^{-1} x - (\mu^2)^T (\Sigma^2)^{-1} x + \frac{1}{2} (\mu^2)^T (\Sigma^2)^{-1} \mu^2 + \ln \frac{N_1}{N_2}$$

$$\Sigma_1 = \Sigma_2 = \Sigma$$

$$z = \underbrace{(\mu^1 - \mu^2)^T \Sigma^{-1} x}_{\mathbf{w}^T} - \underbrace{\frac{1}{2} (\mu^1)^T \Sigma^{-1} \mu^1 + \frac{1}{2} (\mu^2)^T \Sigma^{-1} \mu^2 + \ln \frac{N_1}{N_2}}_b$$

$$y_1 = P(C_1|x) = \sigma(\mathbf{w} \cdot x + b)$$

How about directly find  $\mathbf{w}$  and  $b$ ?

In generative model, we estimate  $N_1, N_2, \mu^1, \mu^2, \Sigma$

Then we have  $\mathbf{w}$  and  $b$

# Logistic regression

If we use gradient decent to find optimal  $w$  and  $b$  for the posterior probability  $y_1 = p(C_1|x) = \sigma(w \cdot x + b)$ , then the problem becomes logistic regression.

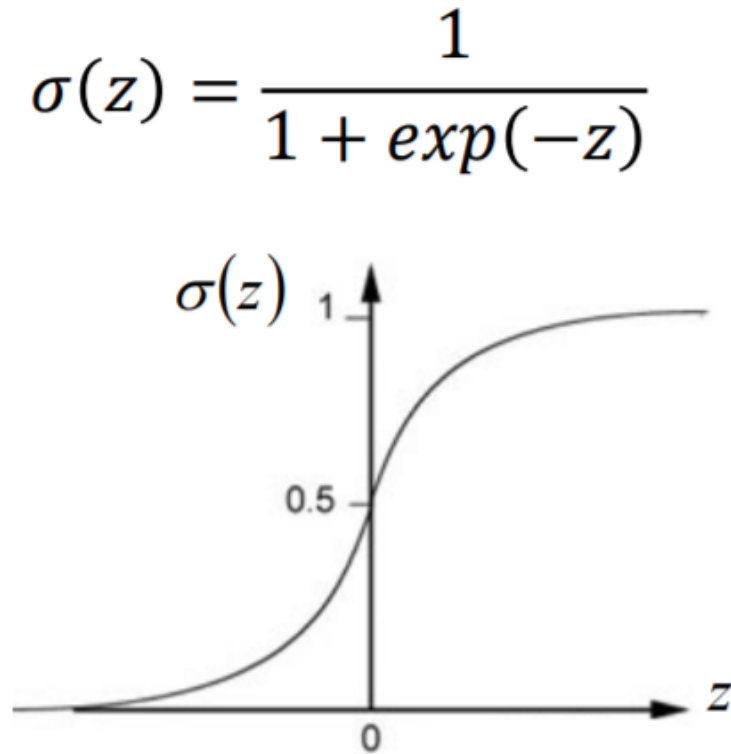
We want to find  $P_{w,b}(C_1|x)$

If  $P_{w,b}(C_1|x) \geq 0.5$ , output  $C_1$

Otherwise, output  $C_2$

$$P_{w,b}(C_1|x) = \sigma(z)$$

$$z = w \cdot x + b$$



# Logistic regression vs regression

## **Logistic Regression**

$$f_{w,b}(x) = \sigma \left( \sum_i w_i x_i + b \right)$$

Output: between 0 and 1

## **Linear Regression**

$$f_{w,b}(x) = \sum_i w_i x_i + b$$

Output: any value

# Maximum likelihood

Assuming the training data is generated from  $y_1 = P_{w,b}(C_1 | x) = \sigma(w \cdot x +$

Training Data	$x^1$	$x^2$	$x^3$	$\dots \dots$	$x^N$
	$C_1$	$C_1$	$C_2$		$C_1$

$$\max L(w, b) = f_{w,b}(x^1) f_{w,b}(x^2) (1 - f_{w,b}(x^3)) \cdots f_{w,b}(x^N)$$

$$\min -\ln L(w, b) = -\ln f_{w,b}(x^1) - \ln f_{w,b}(x^2) - \ln(1 - f_{w,b}(x^3)) \cdots$$

$\hat{y}^n$ : 1 for class 1, 0 for class 2

$$= \sum_n - \left[ \hat{y}^n \ln f_{w,b}(x^n) + (1 - \hat{y}^n) \ln (1 - f_{w,b}(x^n)) \right]$$

Cross entropy between two Bernoulli distribution

# Loss function

Training data:  $(x^n, \hat{y}^n)$

$\hat{y}^n$ : 1 for class 1, 0 for class 2

$$L(f) = \sum_n C(f(x^n), \hat{y}^n)$$

Training data:  $(x^n, \hat{y}^n)$

$\hat{y}^n$ : a real number

$$L(f) = \frac{1}{2} \sum_n (f(x^n) - \hat{y}^n)^2$$

Cross entropy:

$$C(f(x^n), \hat{y}^n) = -[\hat{y}^n \ln f(x^n) + (1 - \hat{y}^n) \ln(1 - f(x^n))]$$



# Multi-class classification

