Classification:

Logistic Regression

Hung-yi Lee

李宏毅

We just found out that $P(C_i \mid x) = \sigma(wx+b)$, so it means that we can use regression to implement classification.

Add sigmoid

Step 1: Function Set

Function set: Including all different w and b

$$\begin{cases} P_{w,b}(C_1|x) \geq 0.5 & \text{class 1} \\ P_{w,b}(C_1|x) < 0.5 & \text{class 2} \end{cases}$$

Step 1: Function Set

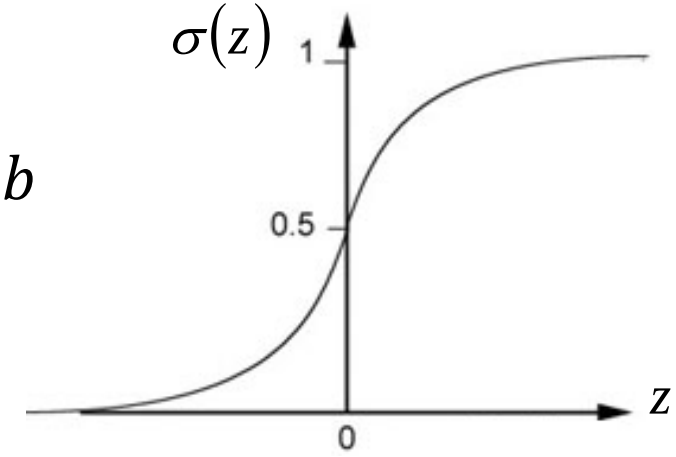
Function set: Including all different w and b

σ(z) ≥ 0.5	z	≥	0	class 1
σ(z) < 0.5	z	<	0	class 2

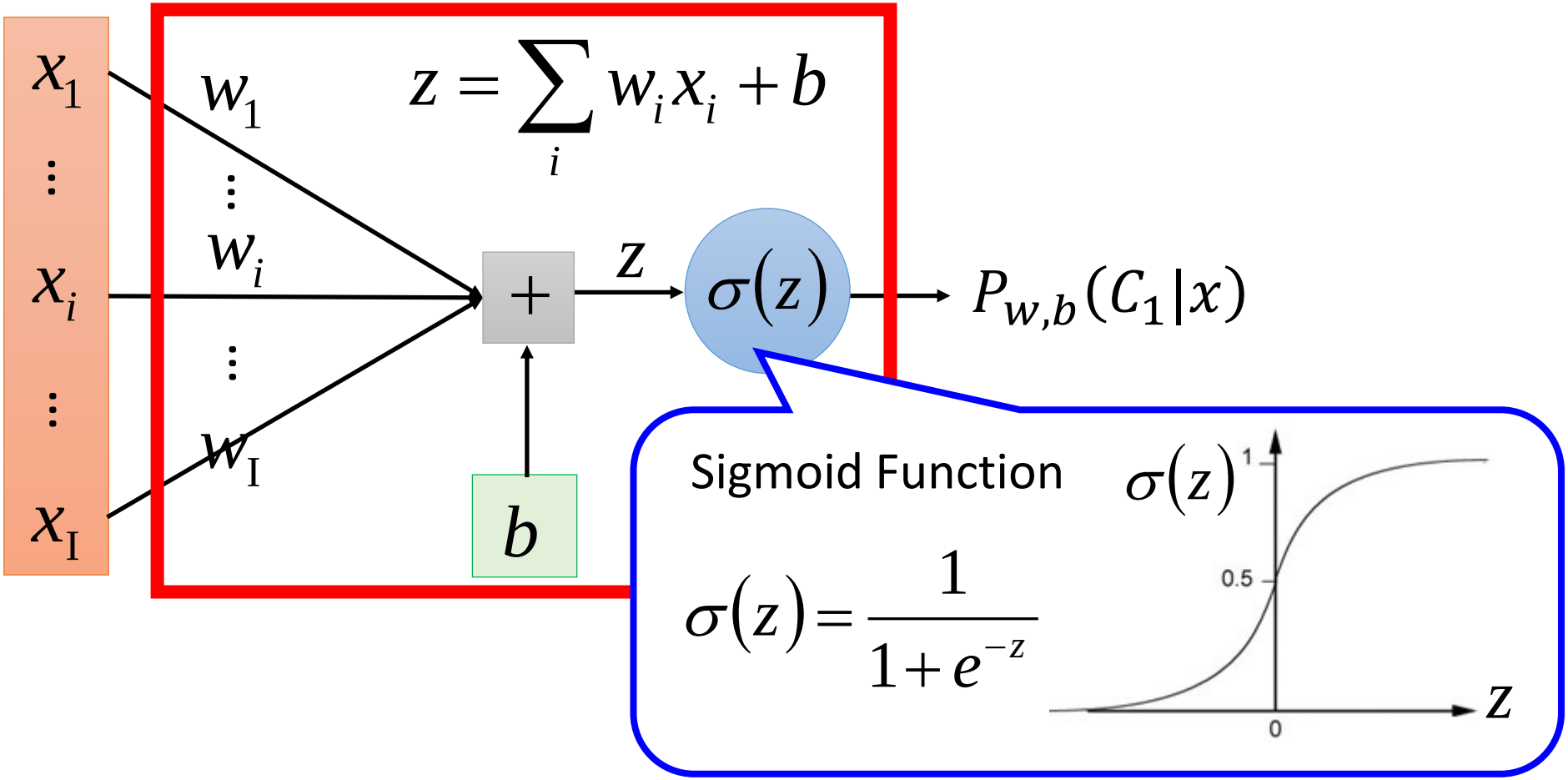
$$P_{w,b}(C_1|x) = \sigma(z)$$

$$z = w \cdot x + b = \sum_i w_i x_i + b$$

$$\sigma(z) = \frac{1}{1 + exp(-z)}$$



Step 1: Function Set



Logistic Regression

Step 1: $f_{w,b}(x) = \sigma\left(\sum_i w_i x_i + b\right)$

Output: between 0 and 1

Linear Regression

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

Output: any value

Logistic Regression = Linear Regression + sigmoid

Step 2:

Step 3:

Approximate ≡ cross entropy

Approximate v.s. Real: P.38

Step 2: Goodness of a Function

Training Data

Binary classification

x^1	x^2	x^3	… …	x^N
C_1	C_1	C_2		C_1

Notice that we are considering C_1 instead of C_2

Assume the data is generated based on $f_{w,b}(x) = P_{w,b}(C_1|x)$

Given a set of w and b, what is its probability of generating the data?

likelihood

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

The most likely w^* and b^* is the one with the largest $L(w,b)$.

$$w^*, b^* = arg \max_{w,b} L(w,b)$$

x^1	x^2	x^3	……
C_1	C_1	C_2	

x^1	x^2	x^3	……
$\hat{y}^1 = 1$	$\hat{y}^2 = 1$	$\hat{y}^3 = 0$	

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

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

$$w^*, b^* = arg \max_{w,b} L(w,b) \quad = \quad w^*, b^* = arg \min_{w,b} -lnL(w,b)$$

We want to change from "max" to "min" in order to do gradient "descent".

$$-lnL(w,b)$$

$$= -lnf_{w,b}(x^1) \rightarrow -\left[\hat{y}^1 lnf(x^1) + (1-\hat{y}^1) ln\left(1 - f(x^1)\right)\right]$$

$$-lnf_{w,b}(x^2) \rightarrow -\left[\hat{y}^2 lnf(x^2) + (1-\hat{y}^2) ln\left(1 - f(x^2)\right)\right]$$

$$-ln\left(1 - f_{w,b}(x^3)\right) \rightarrow -\left[\hat{y}^3 lnf(x^3) + (1-\hat{y}^3) ln\left(1 - f(x^3)\right)\right]$$

$\vdots$

We need to change to the unified format so that we can do the summation.

x^1	x^2	x^3	……
C_1	C_1	C_2	

x^1	x^2	x^3	……
$\hat{y}^1 = 1$	$\hat{y}^2 = 1$	$\hat{y}^3 = 0$	

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

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

$$w^*, b^* = arg \max_{w,b} L(w,b) = w^*, b^* = arg \min_{w,b} -lnL(w,b)$$

$$-lnL(w,b)$$

$$= -lnf_{w,b}(x^1) \rightarrow -[1 \; lnf(x^1) + 0 \; ln(1 - f(x^1))]$$

$$-lnf_{w,b}(x^2) \rightarrow -[1 \; lnf(x^2) + 0 \; ln(1 - f(x^2))]$$

$$-ln\left(1 - f_{w,b}(x^3)\right) \rightarrow -[0 \; lnf(x^3) + 1 \; ln(1 - f(x^3))]$$

$\hat{y}^i$ $1-\hat{y}^i$

Step 2: Goodness of a Function

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

$$-lnL(w,b) = lnf_{w,b}(x^1) + lnf_{w,b}(x^2) + ln\left(1 - f_{w,b}(x^3)\right) \cdots$$

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

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

Cross entropy between two Bernoulli distribution

Distribution p:

$p(x=1) = \hat{y}^n$

$p(x=0) = 1 - \hat{y}^n$

cross entropy

Distribution q:

$q(x=1) = f(x^n)$

$q(x=0) = 1 - f(x^n)$

If two distributions were identical, the cross entropy of these two distributions would be zero.

$$H(p,q) = -\sum_x p(x) ln\left(q(x)\right)$$

Step 2: Goodness of a Function

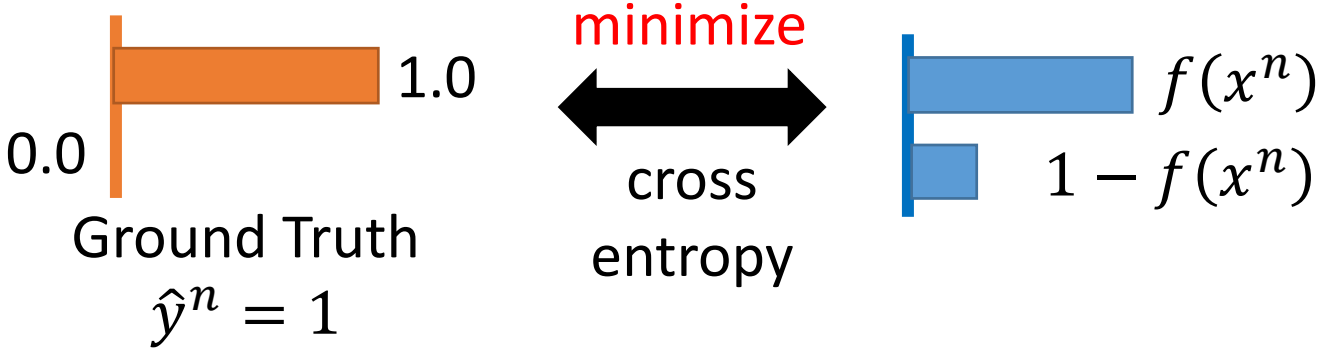
$$L(w,b) = f_{w,b}(x^1) f_{w,b}(x^2) \left(1 - f_{w,b}(x^3)\right) \cdots f_{w,b}(x^N)$$

$$-lnL(w,b) = lnf_{w,b}(x^1) + lnf_{w,b}(x^2) + ln\left(1 - f_{w,b}(x^3)\right) \cdots$$

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

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

Cross entropy between two Bernoulli distribution



Logistic Regression

Step 1: $f_{w,b}(x) = \sigma\left(\sum_i w_i x_i + b\right)$

Output: between 0 and 1

Step 2: Training data: $(x^n, \hat{y}^n)$

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

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

Cross entropy:

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

∵ σ(z) + square error is not suitable for gradient descent.

P.16

Linear Regression

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

Output: any value

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

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

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

Why don't we simply use square error as in linear regression?

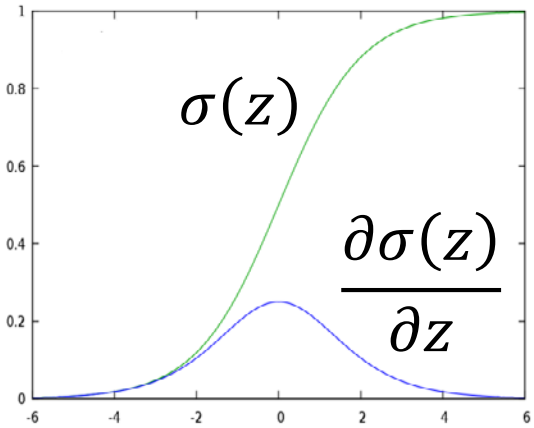
Step 3: Find the best function

result: $\left(1 - f_{w,b}(x^n)\right) x_i^n$

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

$$\frac{\partial lnf_{w,b}(x)}{\partial w_i} = \frac{\partial lnf_{w,b}(x)}{\partial z} \frac{\partial z}{\partial w_i} \rightarrow \frac{\partial z}{\partial w_i} = x_i$$

$$\frac{\partial ln\sigma(z)}{\partial z} = \frac{1}{\sigma(z)} \frac{\partial \sigma(z)}{\partial z} = \frac{1}{\sigma(z)} \sigma(z)\left(1 - \sigma(z)\right)$$



$$f_{w,b}(x) = \sigma(z)$$
$$= 1/1 + exp(-z)$$

$$z = w \cdot x + b = \sum_i w_i x_i + b$$

Step 3: Find the best function

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

$$\left(1-f_{w,b}(x^n)\right)x_i^n$$

result: $-f_{w,b}(x^n)x_i^n$

$$\frac{\partial ln\left(1-f_{w,b}(x)\right)}{\partial w_i} = \frac{\partial ln\left(1-f_{w,b}(x)\right)}{\partial z}\frac{\partial z}{\partial w_i}$$

$$\frac{\partial z}{\partial w_i} = x_i$$

$$\frac{\partial ln\left(1-\sigma(z)\right)}{\partial z} = -\frac{1}{1-\sigma(z)}\frac{\partial \sigma(z)}{\partial z} = -\frac{1}{1-\sigma(z)}\sigma(z)\left(1-\sigma(z)\right)$$

$$f_{w,b}(x) = \sigma(z)$$
$$= 1/1+exp(-z)$$

$$z = w \cdot x + b = \sum_i w_i x_i + b$$

Step 3: Find the best function

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

$\left(1-f_{w,b}(x^n)\right)x_i^n$ $\quad -f_{w,b}(x^n)x_i^n$

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

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

$$= \sum_n -\left(\hat{y}^n - f_{w,b}(x^n)\right) x_i^n$$

Larger difference, larger update

Conclusion:

$$w_i \leftarrow w_i - \eta \sum_n -\left(\hat{y}^n - f_{w,b}(x^n)\right) x_i^n$$

Logistic Regression

Step 1: $f_{w,b}(x) = \sigma\left(\sum_i w_i x_i + b\right)$

Output: between 0 and 1

Step 2:

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

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

$$L(f) = \sum_n \underbrace{l(f(x^n), \hat{y}^n)}_{\text{cross entropy}}$$

Step 3:

Logistic regression: $w_i \leftarrow w_i - \eta \sum_n -\left(\hat{y}^n - f_{w,b}(x^n)\right) x_i^n$

Linear Regression

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

Output: any value

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

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

$$L(f) = \frac{1}{2}\sum_n \underbrace{(f(x^n) - \hat{y}^n)^2}_{\text{square error}}$$

Linear regression: $w_i \leftarrow w_i - \eta \sum_n -\left(\hat{y}^n - f_{w,b}(x^n)\right) x_i^n$

Logistic Regression + Square Error

Why can't we use square error with σ(z) when we are doing gradient descent?

Step 1: $f_{w,b}(x) = \sigma\left(\sum_i w_i x_i + b\right)$

Step 2: Training data: $(x^n, \hat{y}^n)$, $\hat{y}^n$: 1 for class 1, 0 for class 2

$$L(f) = \frac{1}{2}\sum_n \left(f_{w,b}(x^n) - \hat{y}^n\right)^2$$ ← square error

Step 3: $\frac{\partial}{\partial w_i}$

$$\frac{\partial \left(f_{w,b}(x) - \hat{y}\right)^2}{\partial w_i} = 2\left(f_{w,b}(x) - \hat{y}\right)\frac{\partial f_{w,b}(x)}{\partial z}\frac{\partial z}{\partial w_i}$$

$$= 2\left(f_{w,b}(x) - \hat{y}\right) f_{w,b}(x)\left(1 - f_{w,b}(x)\right) x_i$$

$\hat{y}^n = 1$ If $f_{w,b}(x^n) = 1$ (close to target) ➡ $\partial L/\partial w_i = 0$

If $f_{w,b}(x^n) = 0$ (far from target) ➡ $\partial L/\partial w_i = 0$?

Normally, when we are far from the target, the gradient should be very large.

Logistic Regression + Square Error

Step 1: $f_{w,b}(x) = \sigma\left(\sum_i w_i x_i + b\right)$

Step 2: Training data: $(x^n, \hat{y}^n)$, $\hat{y}^n$: 1 for class 1, 0 for class 2

$$L(f) = \frac{1}{2}\sum_n \left(f_{w,b}(x^n) - \hat{y}^n\right)^2$$

Step 3:

$$\frac{\partial \left(f_{w,b}(x) - \hat{y}\right)^2}{\partial w_i} = 2\left(f_{w,b}(x) - \hat{y}\right)\frac{\partial f_{w,b}(x)}{\partial z}\frac{\partial z}{\partial w_i}$$

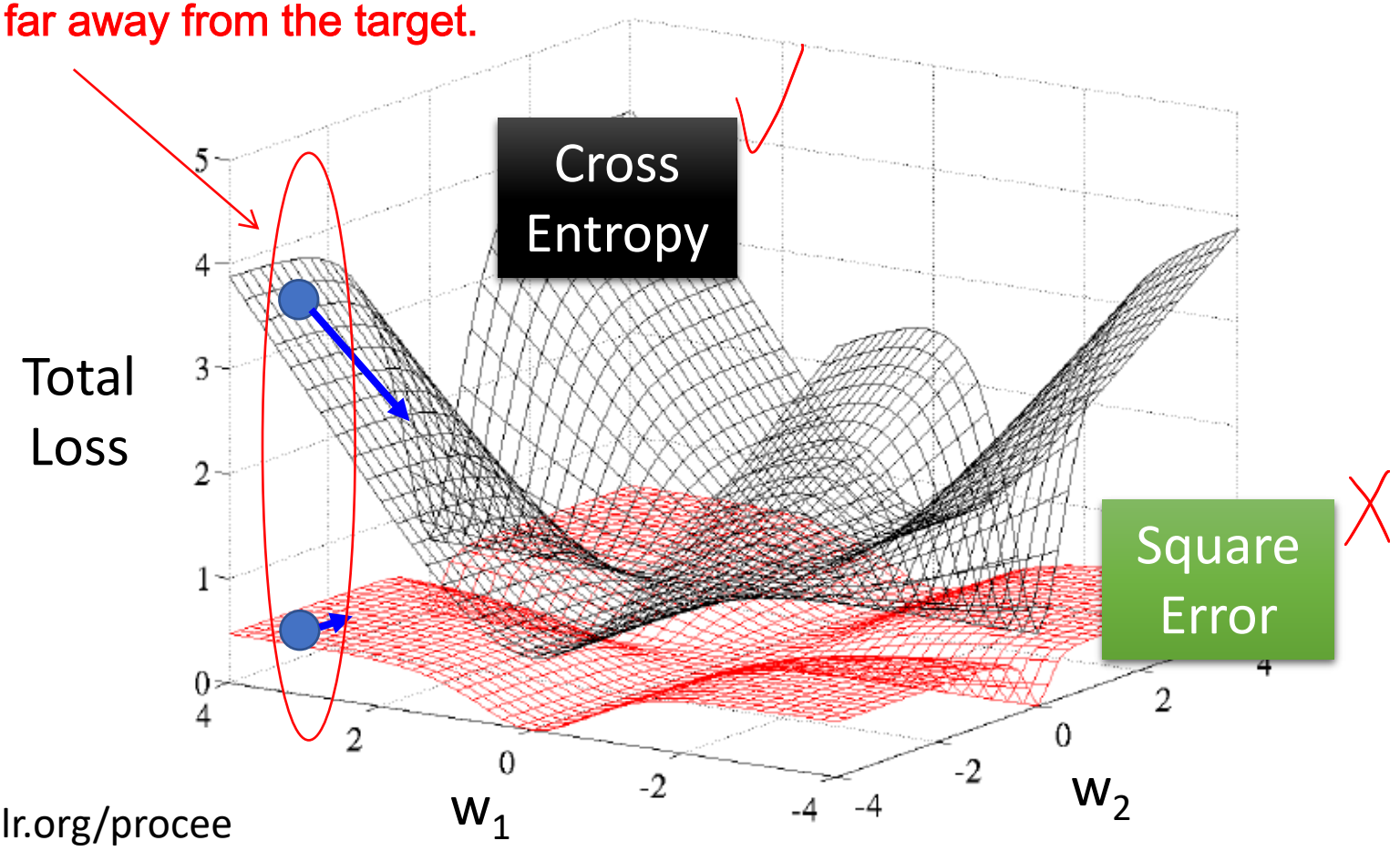
$$= 2\left(f_{w,b}(x) - \hat{y}\right) f_{w,b}(x)\left(1 - f_{w,b}(x)\right) x_i$$

$\hat{y}^n = 0$ If $f_{w,b}(x^n) = 1$ (far from target) ➡ $\partial L/\partial w_i = 0$?

If $f_{w,b}(x^n) = 0$ (close to target) ➡ $\partial L/\partial w_i = 0$

Cross Entropy v.s. Square Error

When we are far away from the target.



http://jmlr.org/proceedings/papers/v9/glorot10a/glorot10a.pdf

logistic regression

probabilistic model

Discriminative v.s. Generative

without any assumptions

with some assumptions

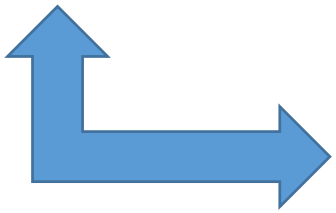
$$P(C_1|x) = \sigma(w \cdot x + b)$$

Gradient descent

MLE (Gaussian distribution)

directly find ***w*** and b

Find $\mu^1, \mu^2, \Sigma^{-1}$



$$w^T = (\mu^1 - \mu^2)^T \Sigma^{-1}$$

$$b = -\frac{1}{2}(\mu^1)^T (\Sigma^1)^{-1} \mu^1 + \frac{1}{2}(\mu^2)^T (\Sigma^2)^{-1} \mu^2 + ln\frac{N_1}{N_2}$$

Will we obtain the same set of w and b? No

The same model (function set), but different function may be selected by the same training data.

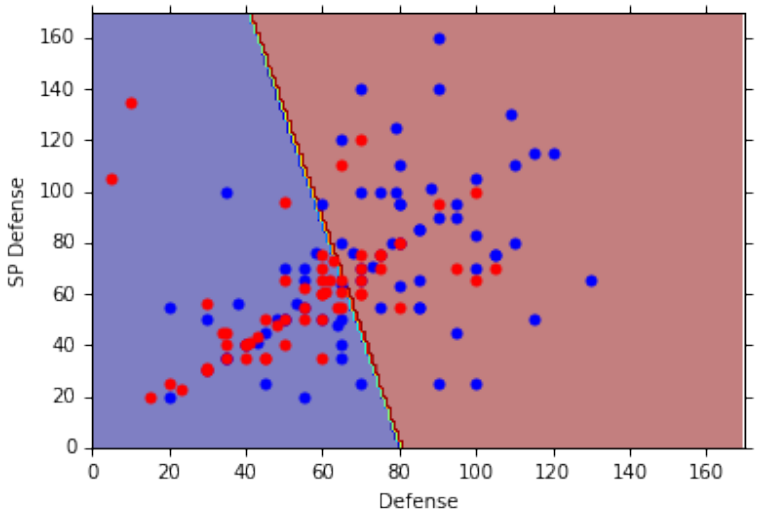
different parameters (w, b)

Example 1:

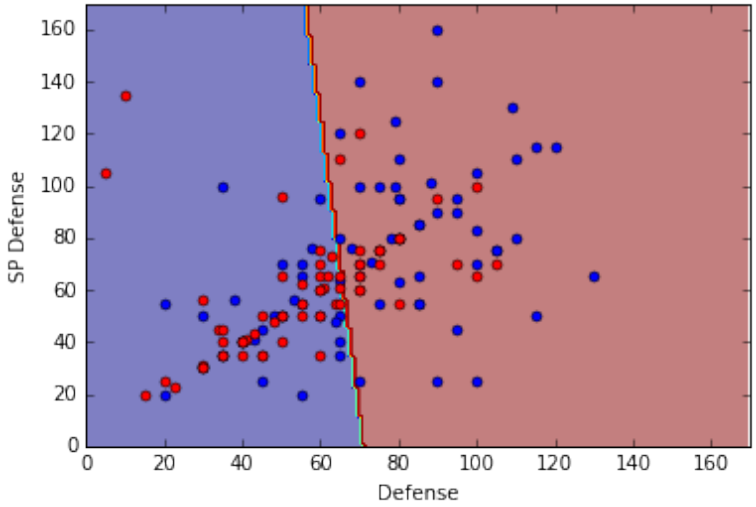
Generative v.s. Discriminative

Normally, discriminative should have better performance.

Generative



Discriminative



All: hp, att, sp att, de, sp de, speed

73% accuracy

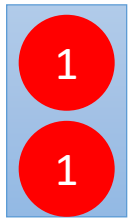
79% accuracy

Example 2: (This case exemplifies the disadvantage of the generative model.)

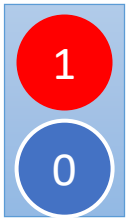
Generative v.s. Discriminative

- Example

Training Data

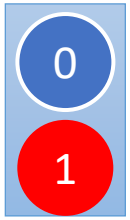


Class 1



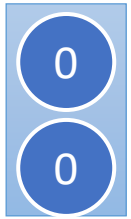
X 4

Class 2



X 4

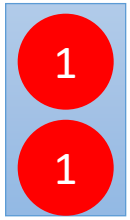
Class 2



X 4

Class 2

Testing Data



Class 1?
Class 2?

The assumption is that every attribute is independent.

How about Naïve Bayes?

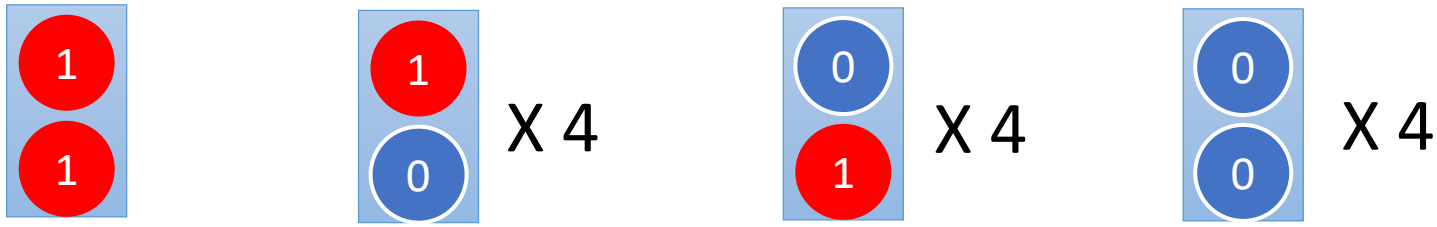
$$P(x|C_i) = P(x_1|C_i)P(x_2|C_i)$$

It's very clear that we should classify this testing data into Class 1.

Generative v.s. Discriminative

- Example

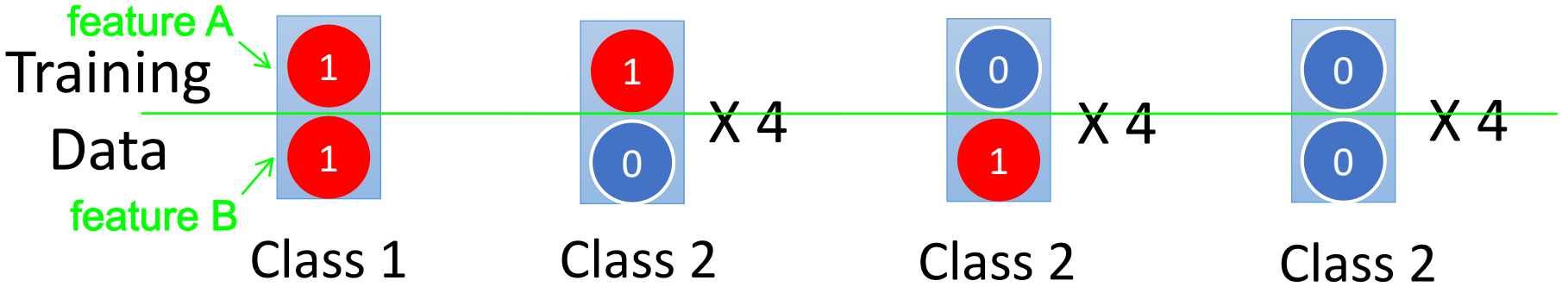
Training Data



Class 1 Class 2 Class 2 Class 2

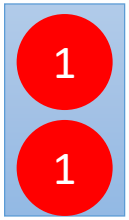
$$P(C_1) = \frac{1}{13} \qquad P(x_1 = 1|C_1) = 1 \qquad P(x_2 = 1|C_1) = 1$$

$$P(C_2) = \frac{12}{13} \qquad P(x_1 = 1|C_2) = \frac{1}{3} \qquad P(x_2 = 1|C_2) = \frac{1}{3}$$



In Class 2, both feature A and feature B have appeared 1, and because feature A and feature B are independent, the Naïve Bayes model thinks that it's possible to appear 1 on both feature A and feature B. Also, the amount of Class 2's data is much bigger than Class 1, so the model classifies the testing data into Class 2.

Testing Data



$$P(C_1|x) = \frac{P(x|C_1)P(C_1)}{P(x|C_1)P(C_1) + P(x|C_2)P(C_2)} < 0.5$$

$P(x|C_1)$: 1×1 — $P(C_1)$: $\frac{1}{13}$

$P(x|C_2)$: $\frac{1}{3} \times \frac{1}{3}$ — $P(C_2)$: $\frac{12}{13}$

$$P(C_1) = \frac{1}{13} \qquad P(x_1 = 1|C_1) = 1 \qquad P(x_2 = 1|C_1) = 1$$

$$P(C_2) = \frac{12}{13} \qquad P(x_1 = 1|C_2) = \frac{1}{3} \qquad P(x_2 = 1|C_2) = \frac{1}{3}$$

mind-picturing

Generative v.s. Discriminative

- Usually people believe discriminative model is better
- Benefit of generative model
  - With the assumption of probability distribution
    - less training data is needed
    - more robust to the noise
  - Priors and class-dependent probabilities can be estimated from different sources.

class independent

ex: In speech recognition, we can collect a lot of text files and compute its probability. It is independent with the speech files.

Multi-class Classification (3 classes as example)

Final output of

C_1: w^1, b_1 $\quad z_1 = w^1 \cdot x + b_1$

C_2: w^2, b_2 $\quad z_2 = w^2 \cdot x + b_2$

C_3: w^3, b_3 $\quad z_3 = w^3 \cdot x + b_3$

Probability:

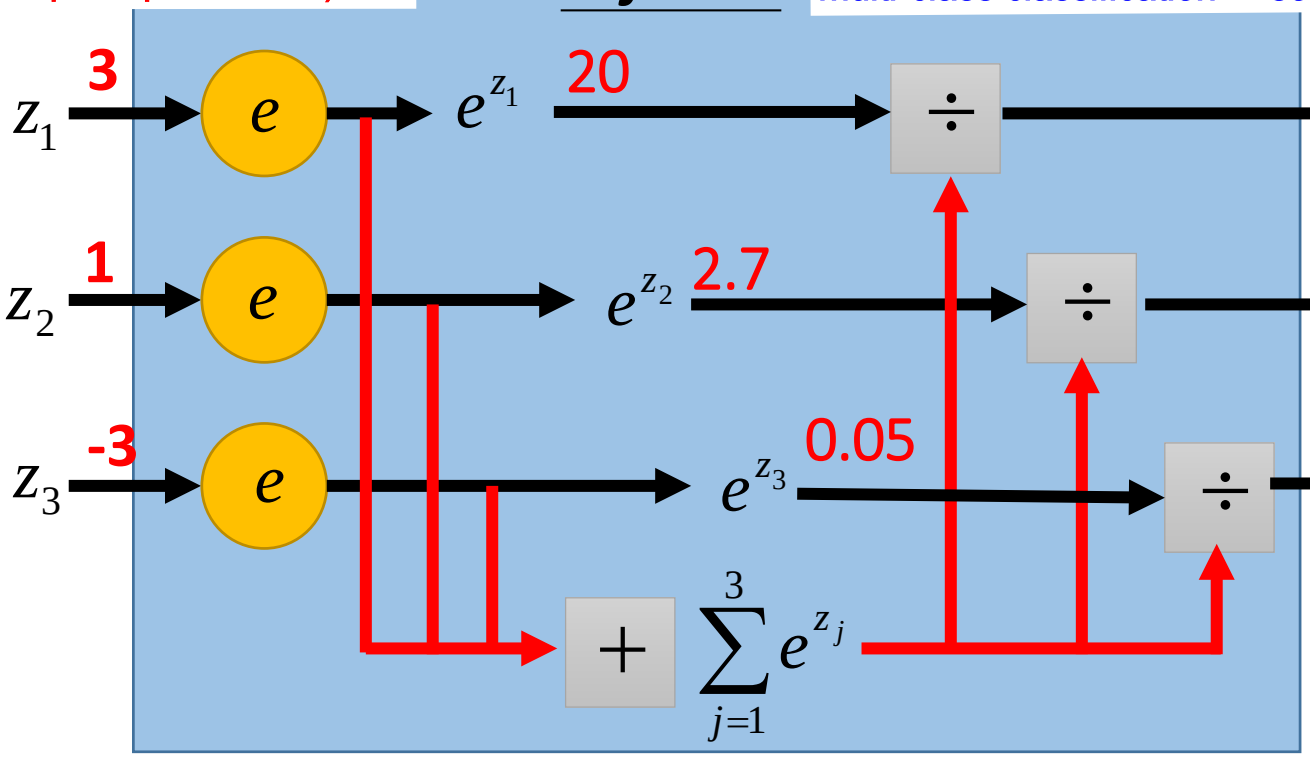
- $1 > y_i > 0$
- $\sum_i y_i = 1$

If the number of class is two in multi-class classification, we will fiind that the softmax is equal to sigmoid in logistic regression. (and w_1, w_2 can merge into w)

$$y_i = P(C_i \mid x) = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}}$$

Gaussian distribution with shared covariance matrix. (Skip the proof here.)

Normalize the final result:
binary classification ⇒ sigmoid
multi-class classification ⇒ softmax



$$y_1 = e^{z_1} \Big/ \sum_{j=1}^{3} e^{z_j}$$

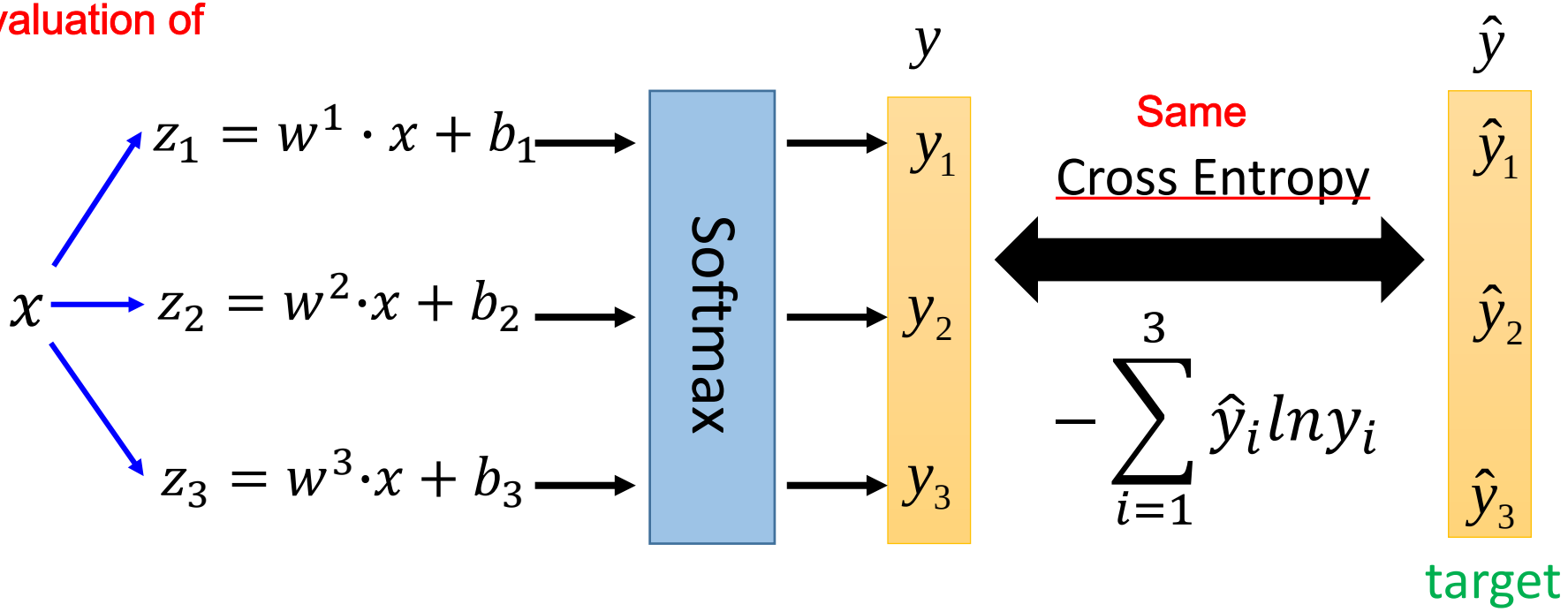
$$y_2 = e^{z_2} \Big/ \sum_{j=1}^{3} e^{z_j}$$

$$y_3 = e^{z_3} \Big/ \sum_{j=1}^{3} e^{z_j}$$

[Bishop, P209-210]

Multi-class Classification (3 classes as example)

Evaluation of



If x ∈ class 1

$$\hat{y} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$-ln y_1$$

If x ∈ class 2

$$\hat{y} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

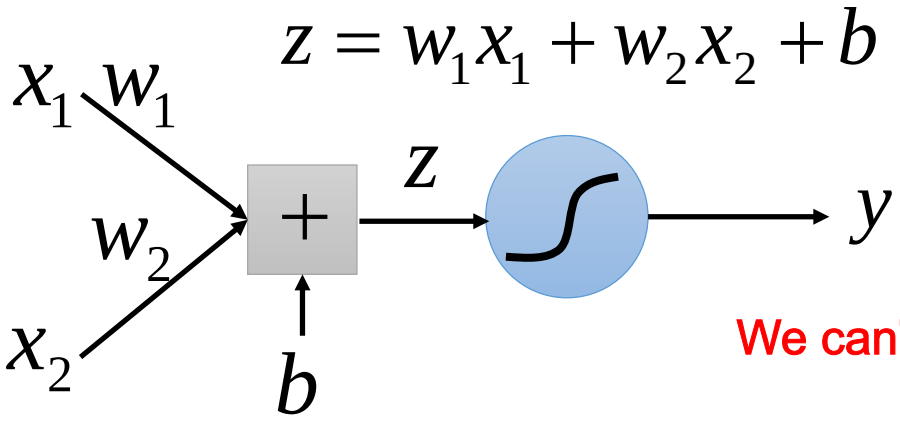
$$-ln y_2$$

If x ∈ class 3

$$\hat{y} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$-ln y_3$$

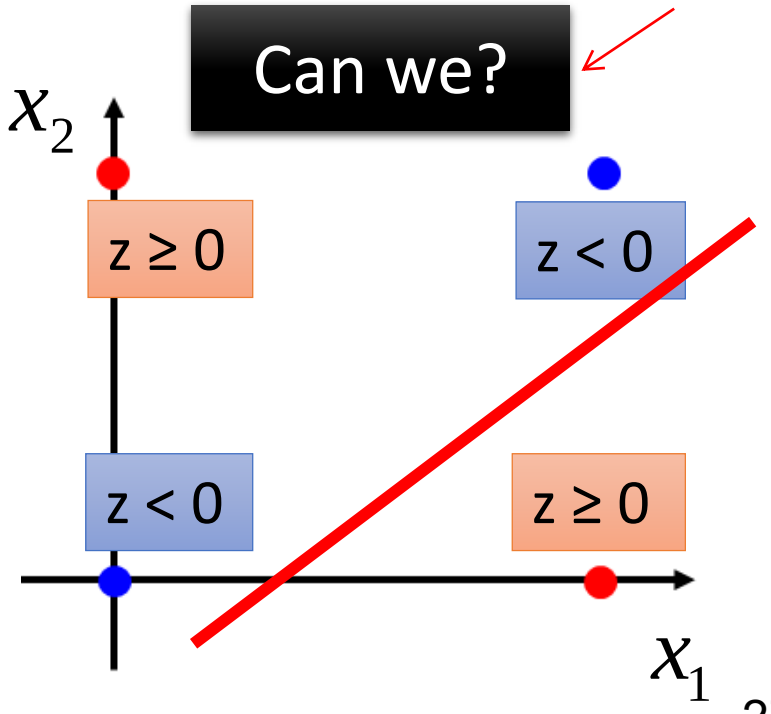
Limitation of Logistic Regression



$$\begin{cases} Class1 & y \geq 0.5 \quad (z \geq 0) \\ Class2 & y < 0.5 \quad (z < 0) \end{cases}$$

We can't find a line to separate the data perfectly.

Input Feature		Label
x_1	x_2	
0	0	Class 2
0	1	Class 1
1	0	Class 1
1	1	Class 2

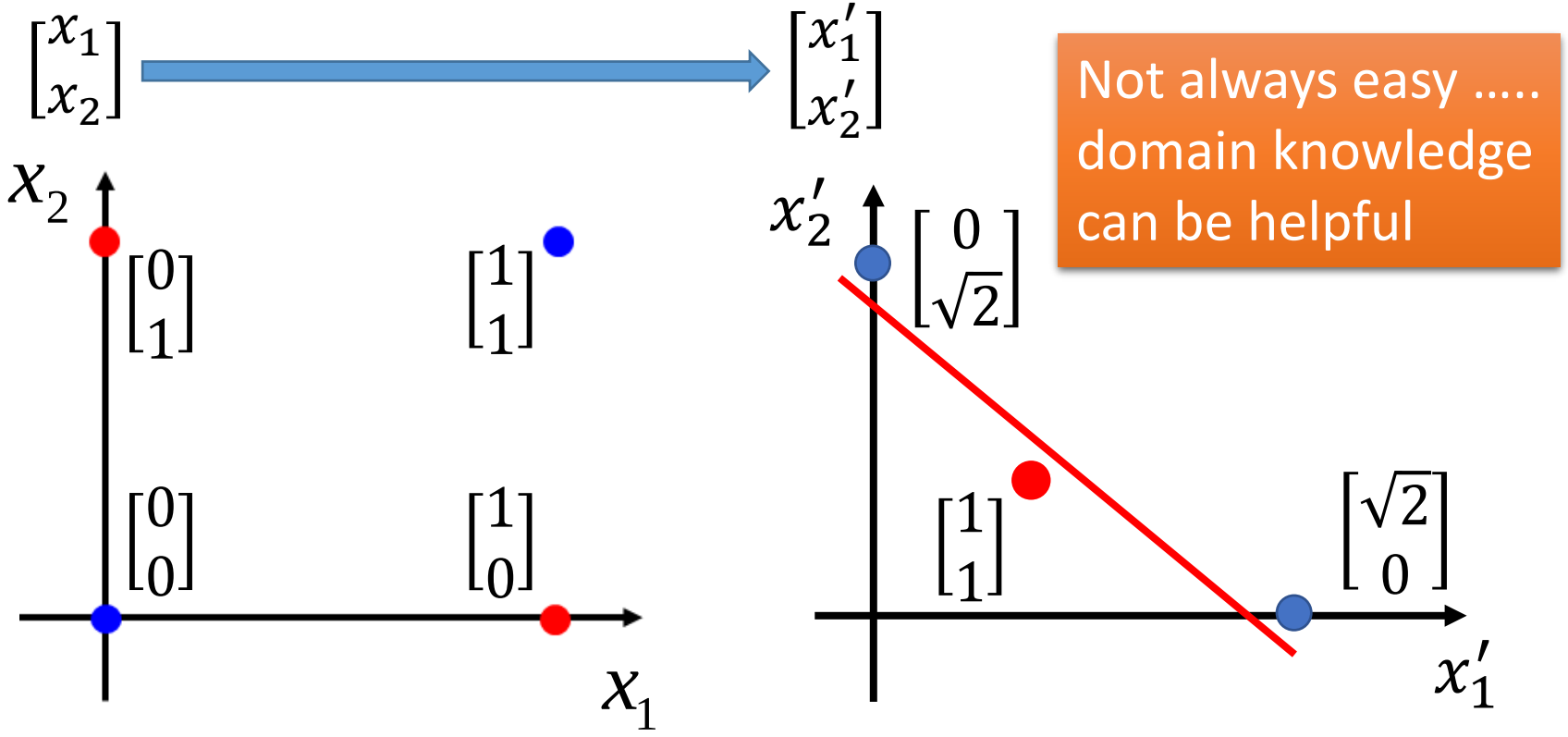


Limitation of Logistic Regression

- ***Feature transformation***

x_1': distance to $\begin{bmatrix}0\\0\end{bmatrix}$

x_2': distance to $\begin{bmatrix}1\\1\end{bmatrix}$

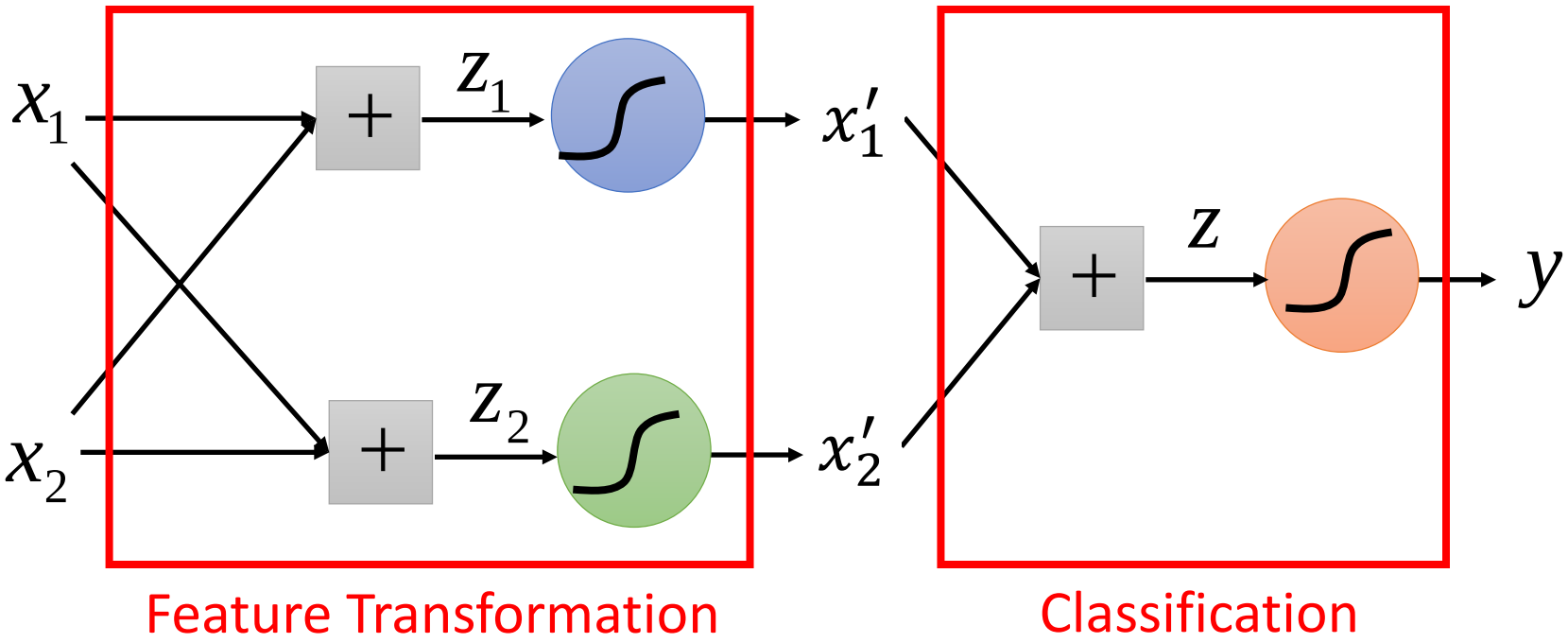


Not always easy ….. domain knowledge can be helpful

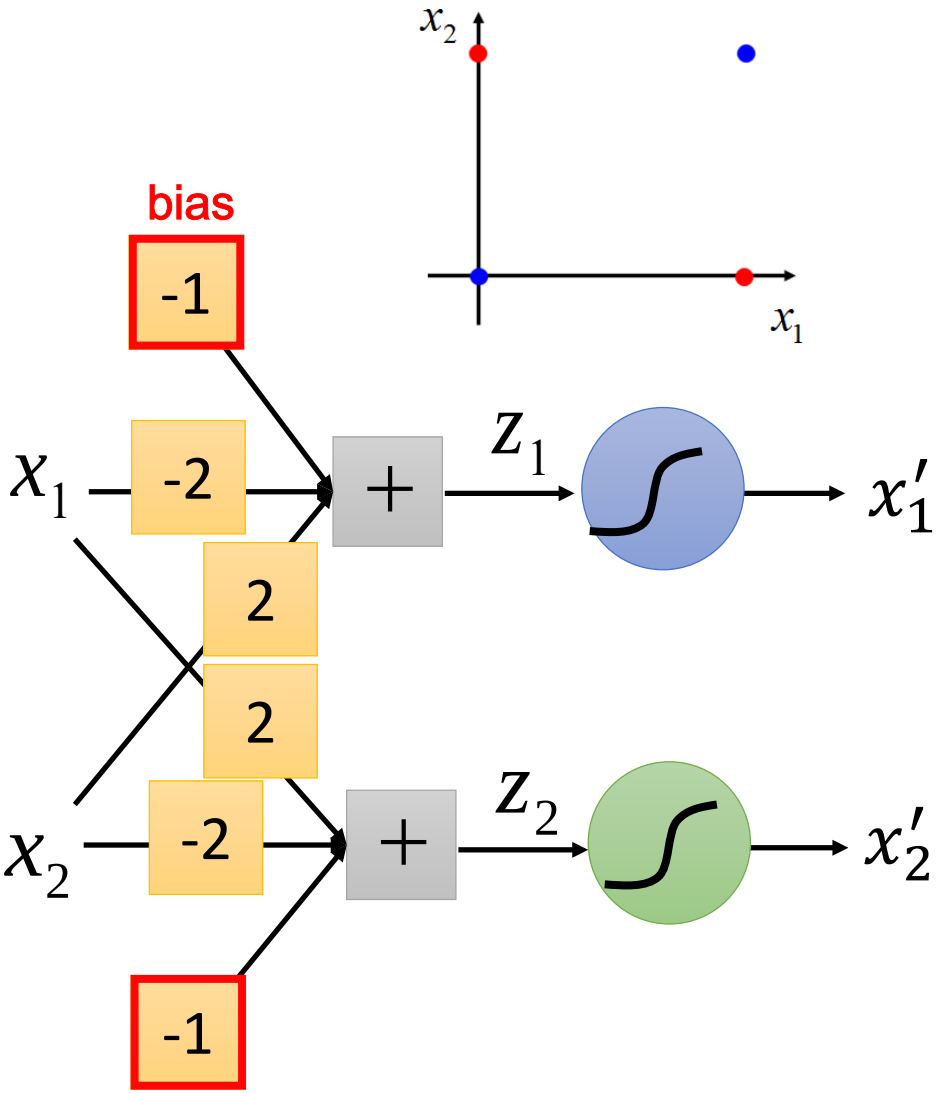
Limitation of Logistic Regression

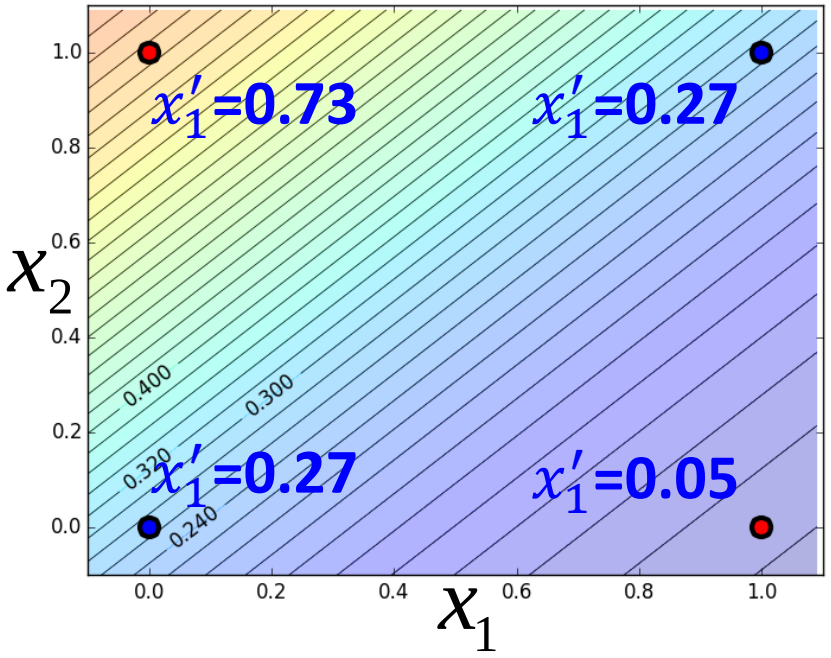
We use logistic regression to do feature transformation and classification respectively.

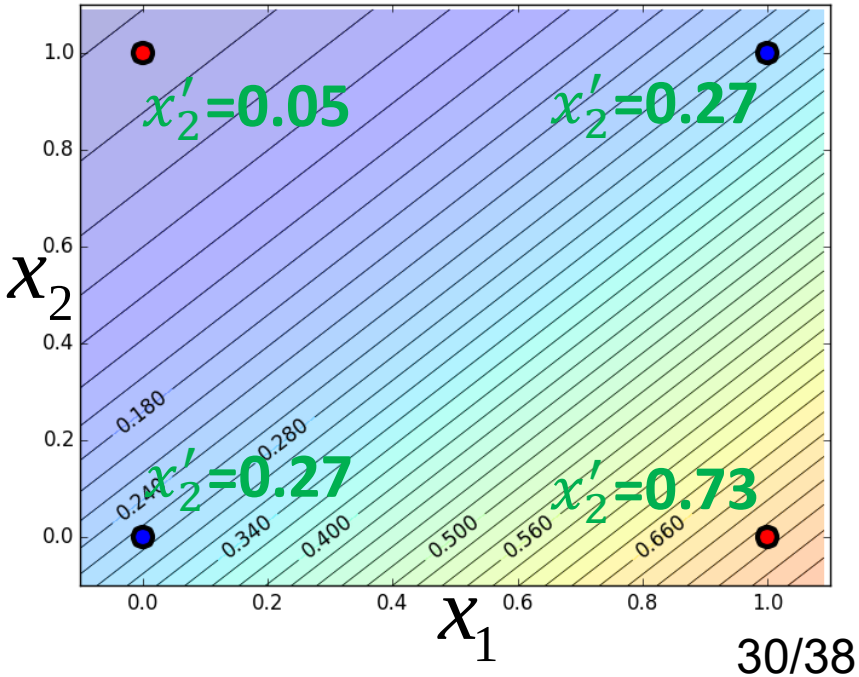
- Cascading logistic regression models

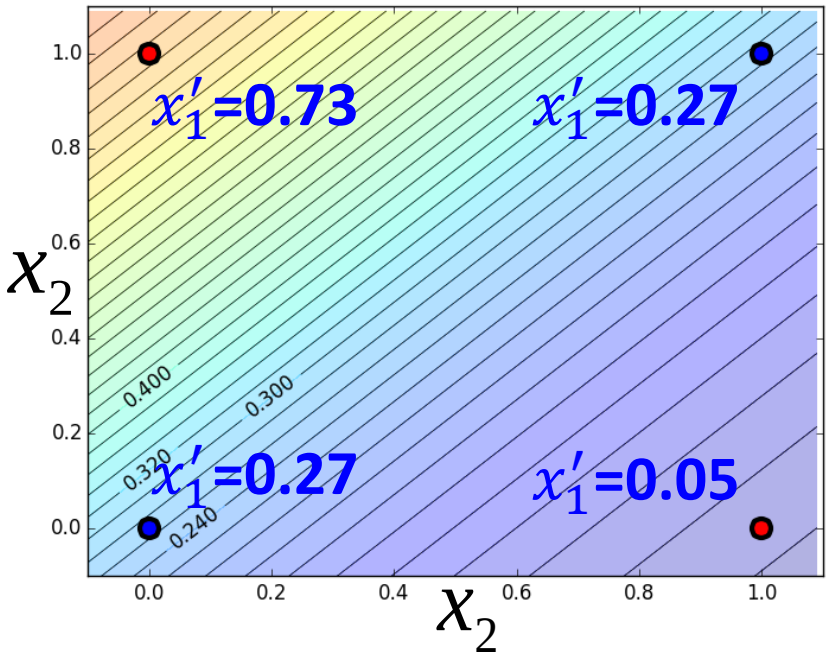


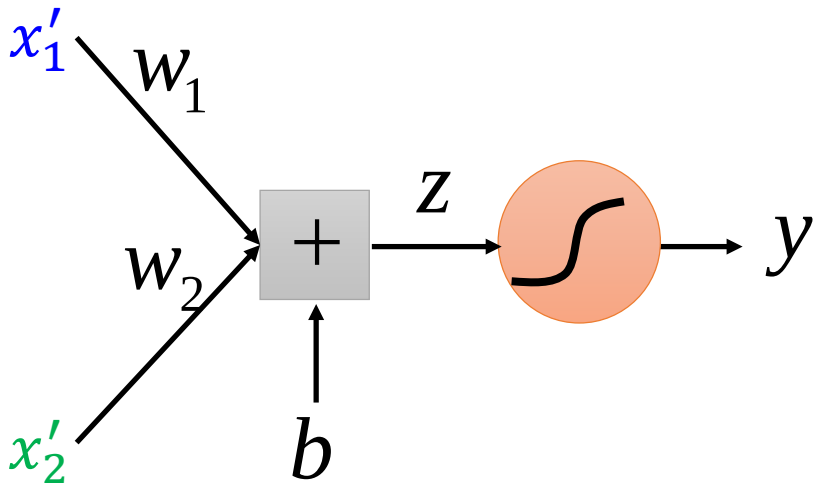
We ignore the bias here to simplify the illustration.

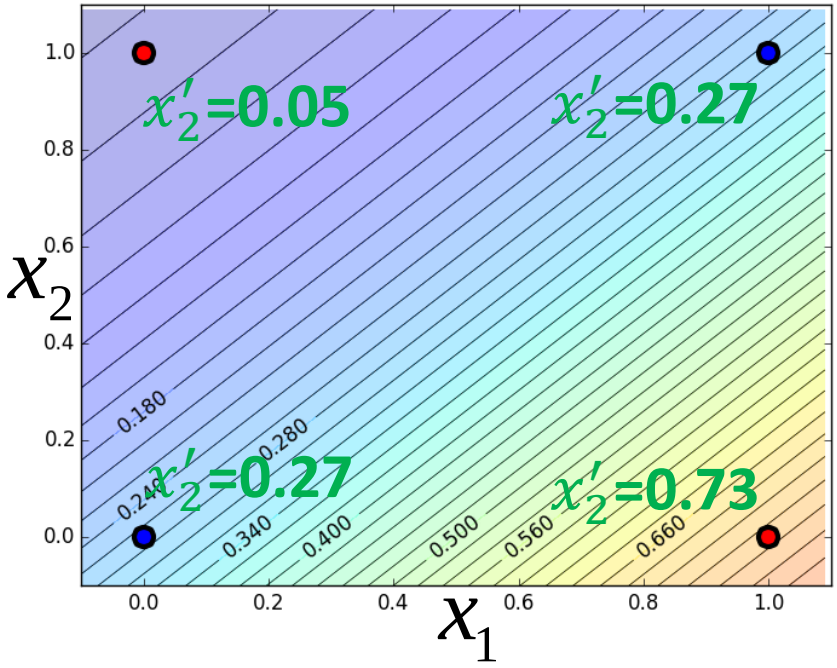

x_2
x_1
bias
-1
x_1
-2
+
z_1
x'_1
2
2
x_2
-2
+
z_2
x'_2
-1

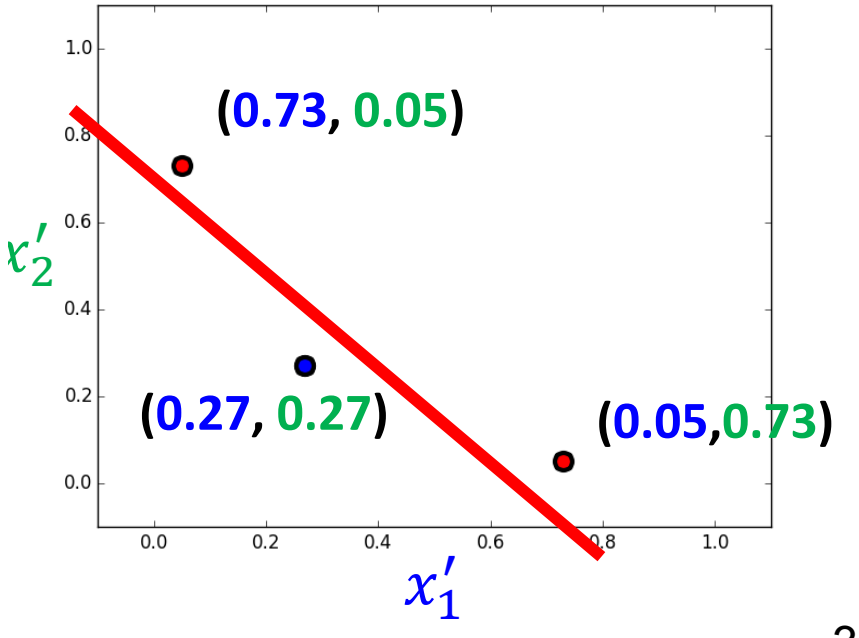

x'_1=0.73
x'_1=0.27
x'_1=0.27
x'_1=0.05
0.400
0.300
0.320
0.240
x_2
x_1
0.0
0.2
0.4
0.6
0.8
1.0


x'_2=0.05
x'_2=0.27
x'_2=0.27
x'_2=0.73
0.180
0.240
0.280
0.340
0.400
0.500
0.560
0.660
x_2
x_1
0.0
0.2
0.4
0.6
0.8
1.0

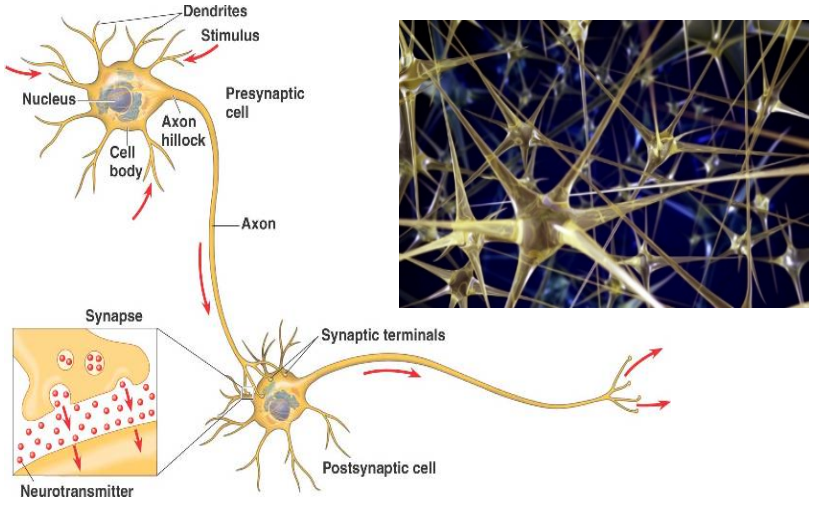

x_1'=0.73
x_1'=0.27
x_1'=0.27
x_1'=0.05
0.400
0.300
0.320
0.240
x_2
x_2


x_1'
w_1
w_2
x_2'
b
z
y


x_2'=0.05
x_2'=0.27
x_2'=0.27
x_2'=0.73
0.180
0.280
0.240
0.340
0.400
0.500
0.560
0.660
x_2
x_1

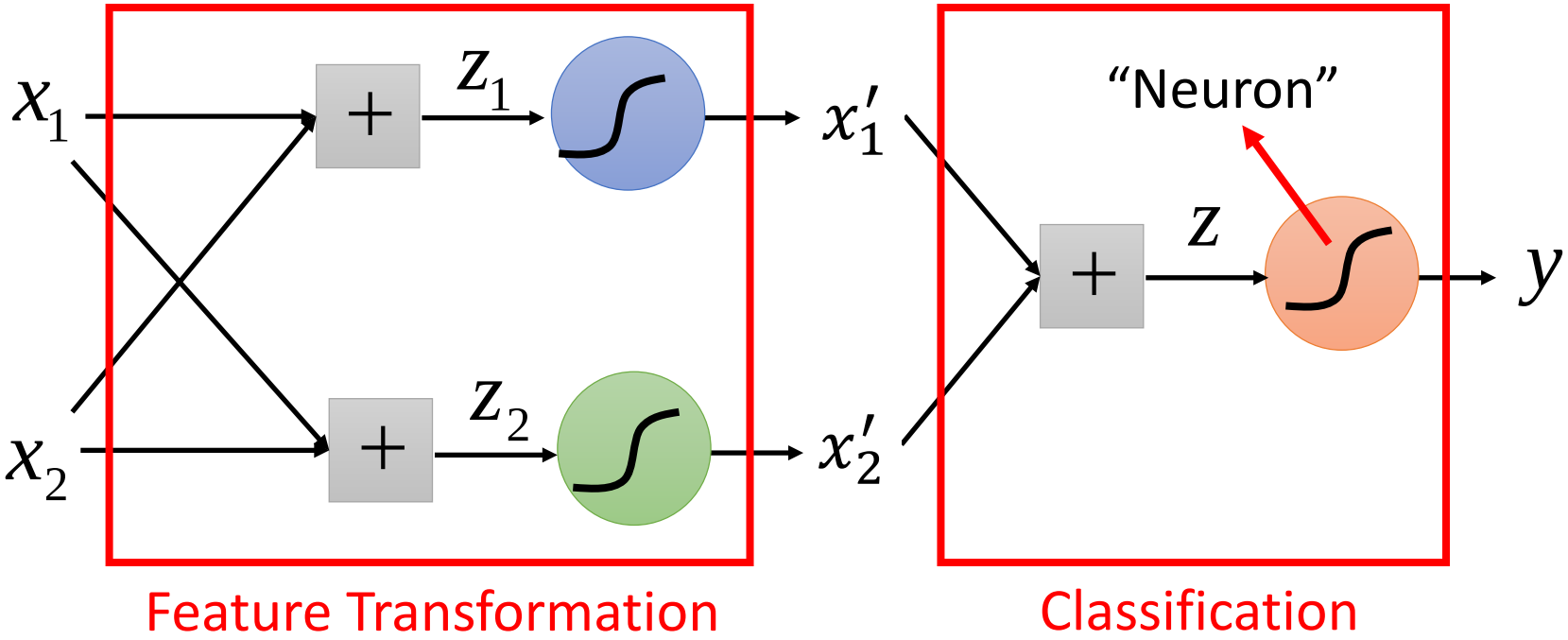

(0.73, 0.05)
(0.27, 0.27)
(0.05,0.73)
x_2'
x_1'

Deep Learning!



All the parameters of the logistic regressions are jointly learned.

using gradient descent



Neural Network

Reference

- Bishop: Chapter 4.3

Appendix

Three Steps

x^1	x^2	x^3	……	x^n
$\hat{y}^1$	$\hat{y}^2$	$\hat{y}^3$		$\hat{y}^n$

$\hat{y}^n = class\ 1, class\ 2$

of logistic regression

- Step 1. Function Set (Model)

x (feature) → If $P(C_1|x) > 0.5$, output: y = class 1; Otherwise, output: y = class 2 → y (class)

$$P(C_1|x) = \sigma(w \cdot x + b)$$

w and b are related to $N_1, N_2, \mu^1, \mu^2, \Sigma$

ideal

- Step 2. Goodness of a function Couldn't use gradient descent

$$L(f) = \sum_n \delta(f(x^n) \neq \hat{y}^n) \Rightarrow L(f) = \sum_n l(f(x^n) \neq \hat{y}^n)$$

- Step 3. Find the best function: gradient descent

Step 2: Loss function

$$f_{w,b}(x) = \begin{cases} z \geq 0 & +1 \\ z < 0 & -1 \end{cases}$$

Ideal loss:

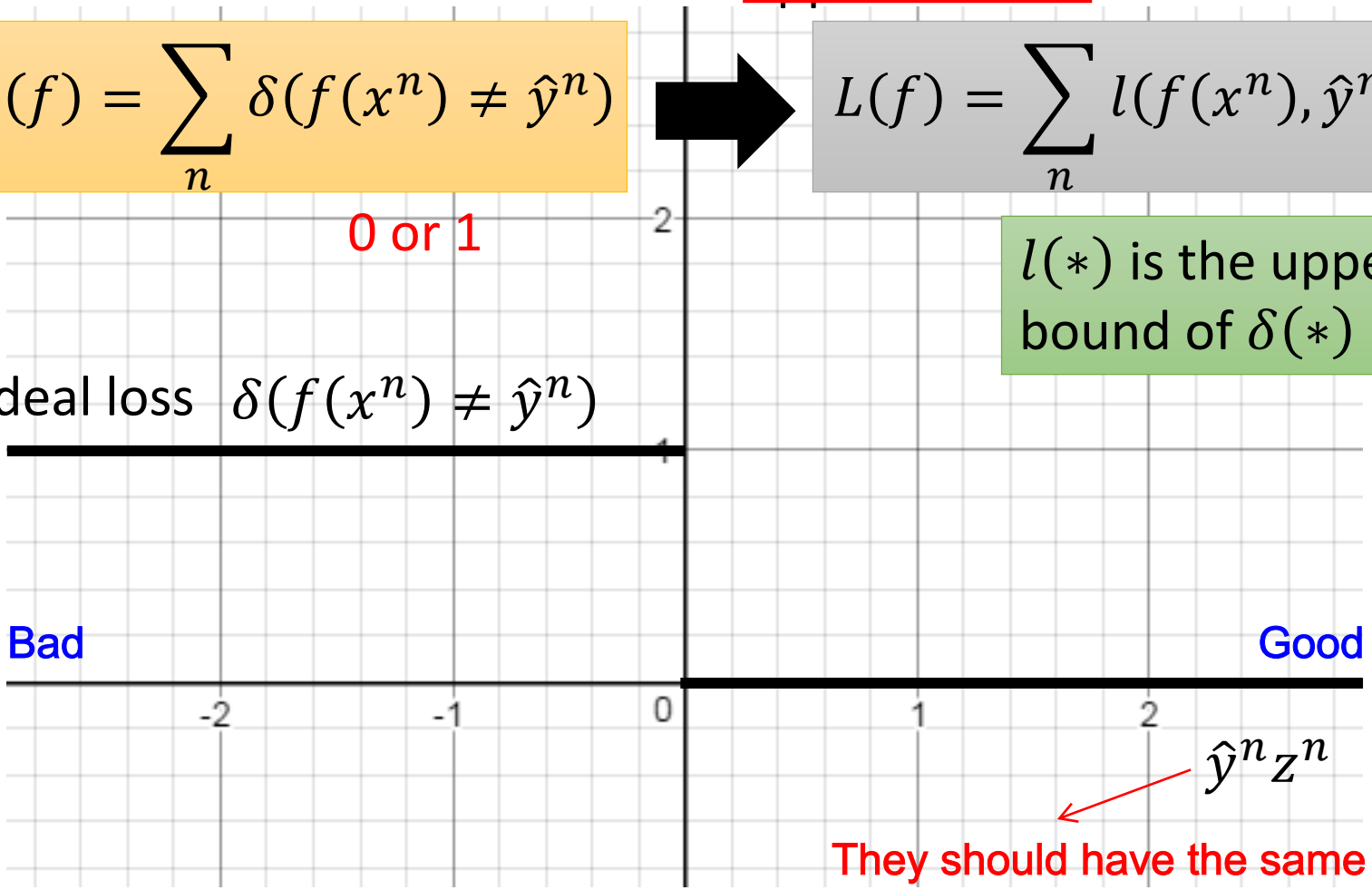
$$L(f) = \sum_n \delta(f(x^n) \neq \hat{y}^n)$$

0 or 1

Approximation:

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

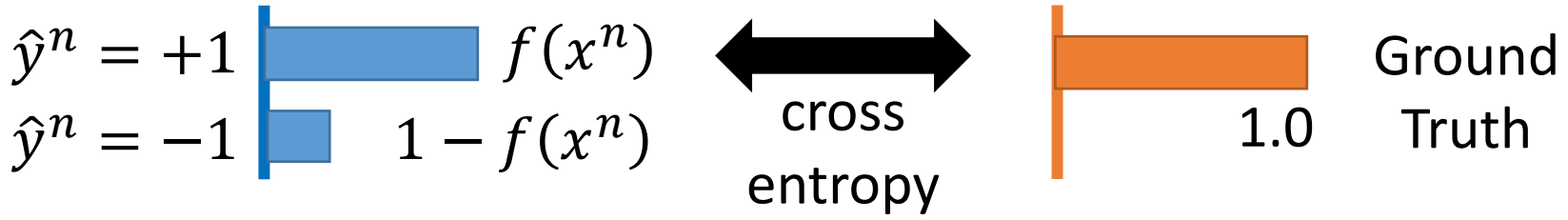
$l(*)$ is the upper bound of $\delta(*)$



They should have the same sign.
(Larger value, smaller loss)

Step 2: Loss function

$l(f(x^n), \hat{y}^n)$: cross entropy



If $\hat{y}^n = +1$:

$$l(f(x^n), \hat{y}^n) = -\ln f(x^n) = -\ln \sigma(z^n) = -\ln \frac{1}{1 + exp(-z^n)}$$

$$= \ln\left(1 + exp(-z^n)\right) = \underline{\ln\left(1 + exp(-\hat{y}^n z^n)\right)}$$

If $\hat{y}^n = -1$:

$$l(f(x^n), \hat{y}^n) = -\ln\left(1 - f(x^n)\right)$$

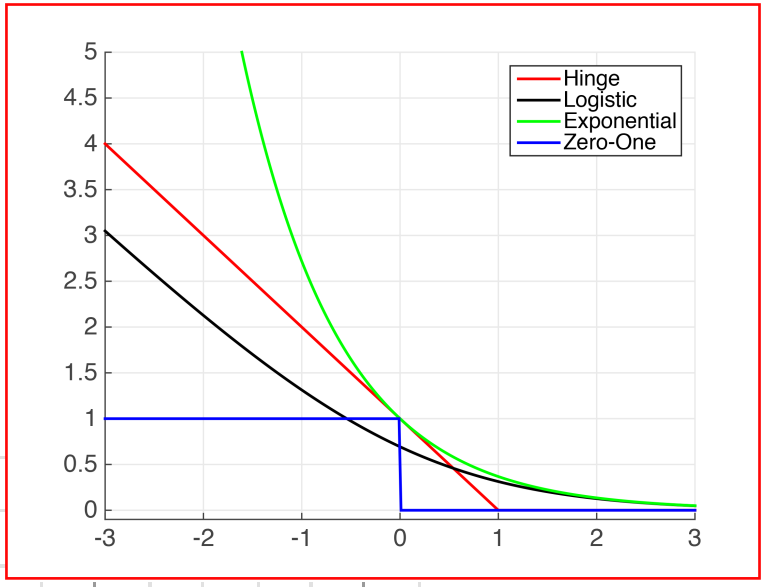
$$= -\ln\left(1 - \sigma(x^n)\right) = -\ln \frac{exp(-z^n)}{1 + exp(-z^n)} = -\ln \frac{1}{1 + exp(z^n)}$$

$$= \ln\left(1 + exp(z^n)\right) = \underline{\ln\left(1 + exp(-\hat{y}^n z^n)\right)}$$

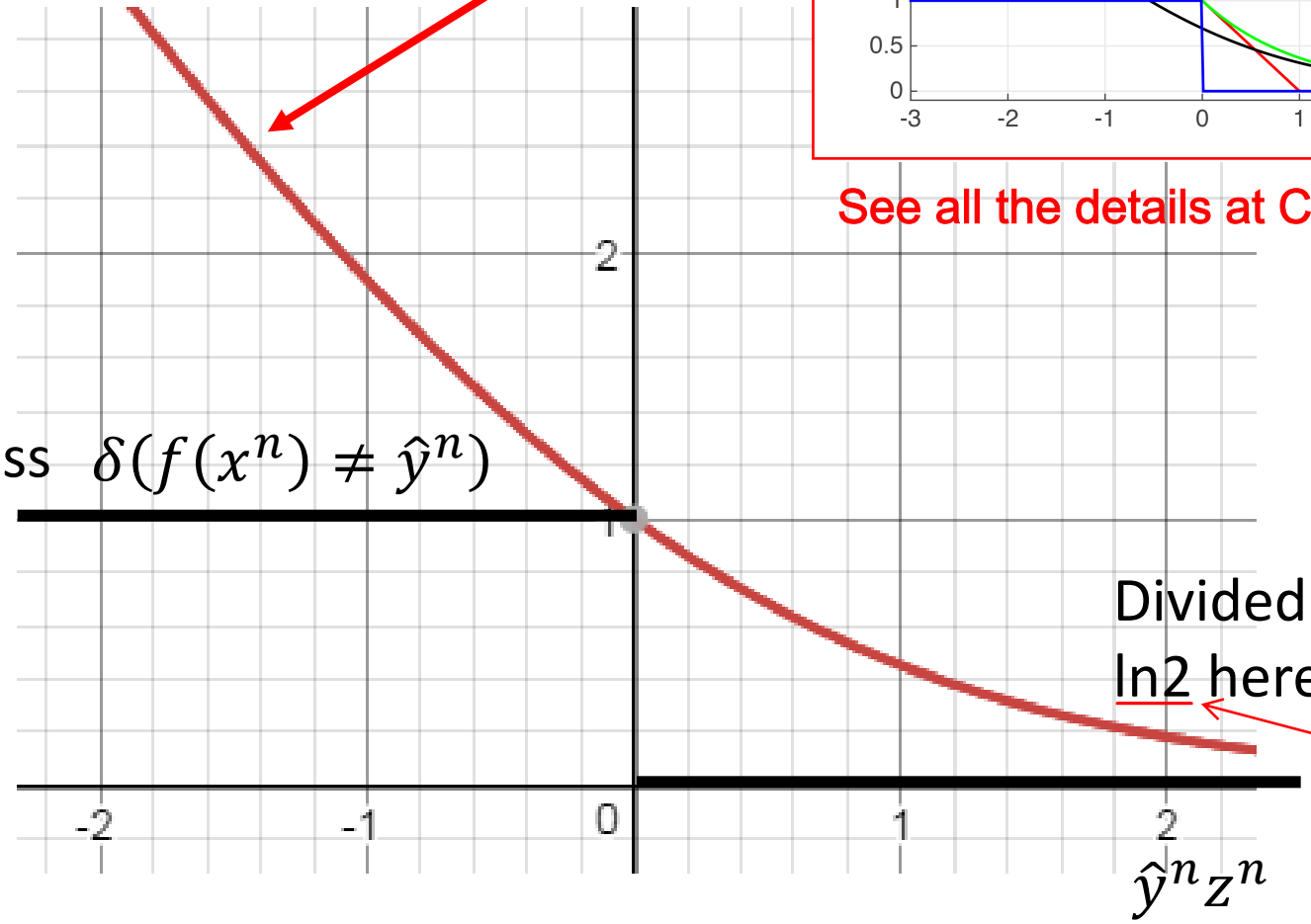
Step 2: Loss function

$l(f(x^n), \hat{y}^n)$: cross entropy

$l(f(x^n), \hat{y}^n) = \ln\left(1 + exp(-\hat{y}^n z^n)\right)$



See all the details at Ch.20 (SVM)



Ideal loss $\delta(f(x^n) \neq \hat{y}^n)$

Divided by ln2 here

0.693