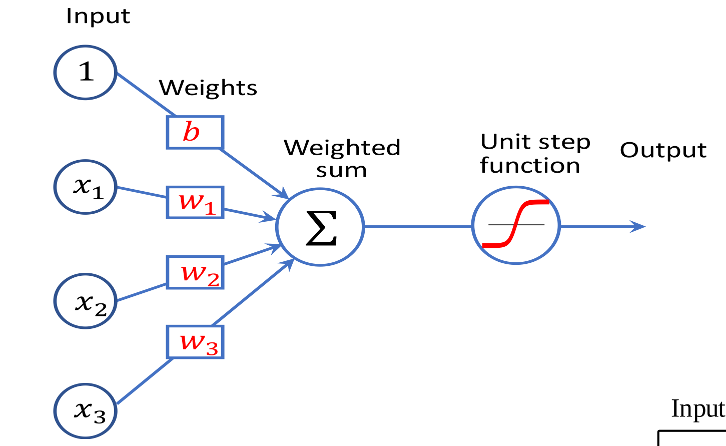
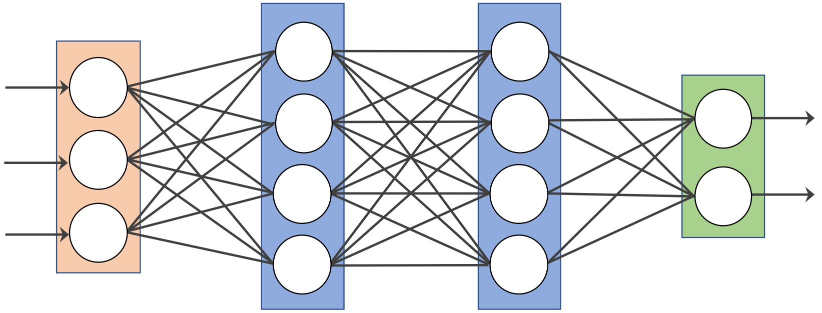
In this Course

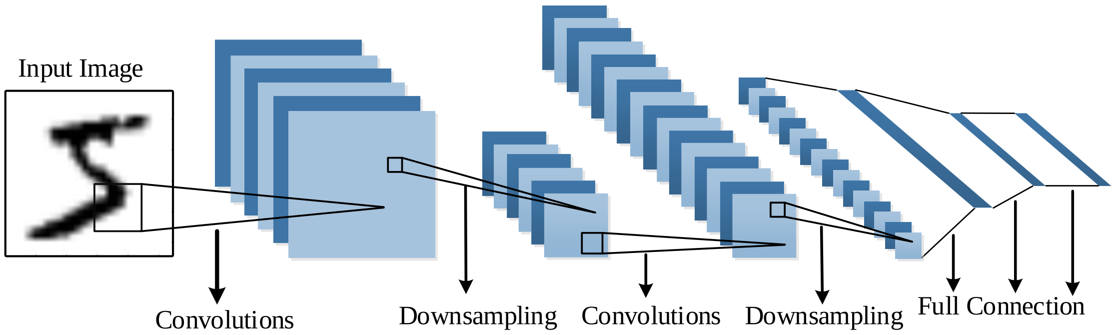
1. DL basics, linear regression, logistic regression etc.



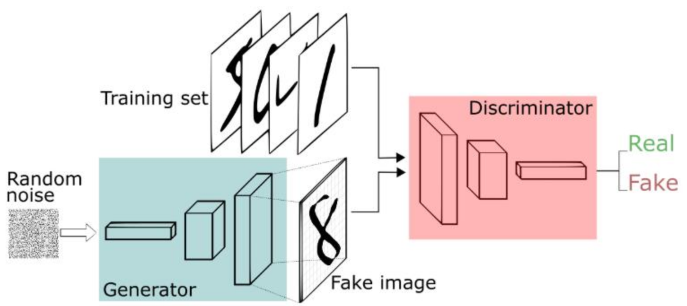
2. Multilayer neural networks, backpropagation



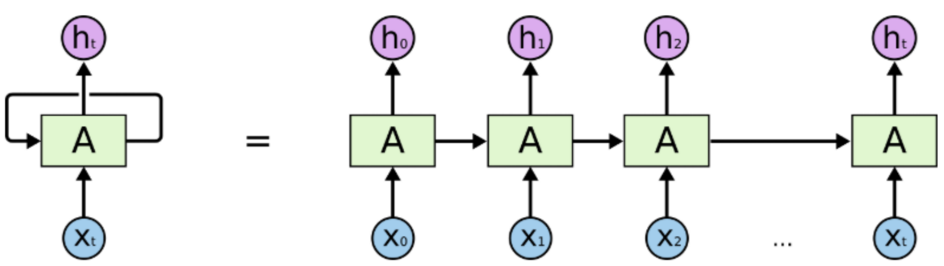
3. Convolutional Neural Networks and Applications



4. Generative Adversarial Networks



5. Recurrent networks and applications



Previous Lecture

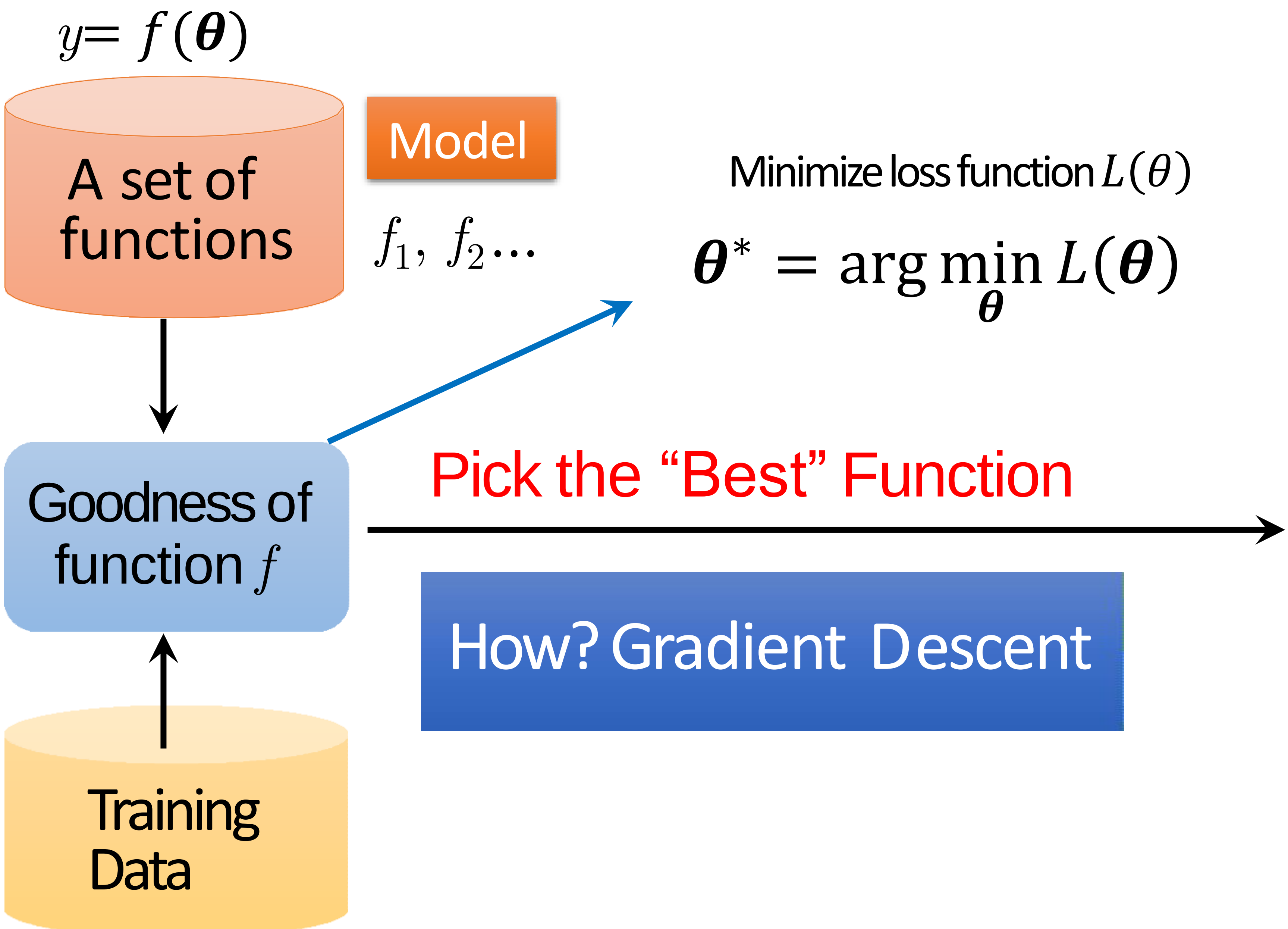
- Course Information
- Introduction to Deep Learning
- Deep Learning Basics
  - Linear Regression
  - Loss Function
  - Gradient Descent
  - Regularization

Lecture 2
Gradient Descent
Stochastic Gradient Descent
Logistic Regression

Gradient Descent

Many slides adapted from Andrew Ng and Hungyi Lee

Review: Step 3 in Machine Learning



Review: Gradient Descent

- In step 3, we have to solve the following optimization problem:

$$\theta^* = \arg\min_{\theta} L(\theta)$$

L: loss function θ: parameters

Suppose that θ has two variables $\{\theta_1, \theta_2\}$

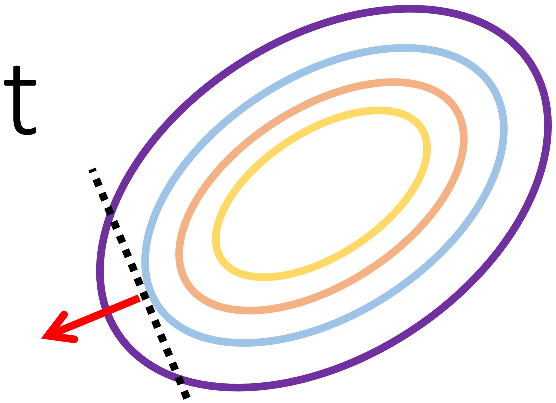
Randomly start at $\theta^{(0)} = \begin{bmatrix} \theta_1^{(0)} \\ \theta_2^{(0)} \end{bmatrix}$

$$\nabla L(\theta) = \begin{bmatrix} \partial L(\theta_1)/\partial \theta_1 \\ \partial L(\theta_2)/\partial \theta_2 \end{bmatrix}$$

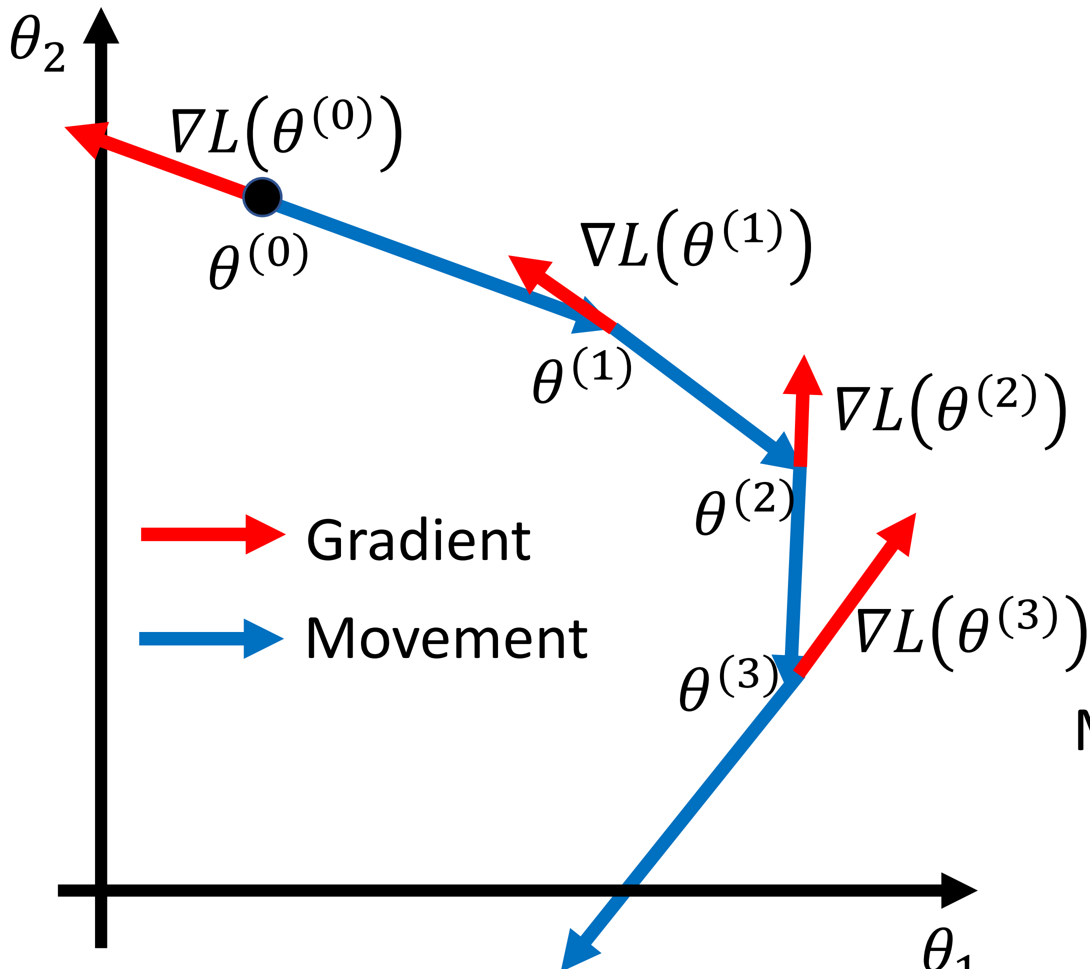
$$\begin{bmatrix} \theta_1^{(1)} \\ \theta_2^{(1)} \end{bmatrix} = \begin{bmatrix} \theta_1^{(0)} \\ \theta_2^{(0)} \end{bmatrix} - \eta \begin{bmatrix} \partial L\left(\theta_1^{(0)}\right)/\partial \theta_1 \\ \partial L\left(\theta_2^{(0)}\right)/\partial \theta_2 \end{bmatrix} \Rightarrow \theta^{(1)} = \theta^{(0)} - \eta \nabla L\left(\theta^{(0)}\right)$$

$$\begin{bmatrix} \theta_1^{(2)} \\ \theta_2^{(2)} \end{bmatrix} = \begin{bmatrix} \theta_1^{(1)} \\ \theta_2^{(1)} \end{bmatrix} - \eta \begin{bmatrix} \partial L\left(\theta_1^{(1)}\right)/\partial \theta_1 \\ \partial L\left(\theta_2^{(1)}\right)/\partial \theta_2 \end{bmatrix} \Rightarrow \theta^{(2)} = \theta^{(1)} - \eta \nabla L\left(\theta^{(1)}\right)$$

Review: Gradient Descent



Gradient: the normal direction of the contour of loss function



Start at position $\theta^{(0)}$

Compute gradient at $\theta^{(0)}$

Move to $\theta^{(1)} = \theta^{(0)} - \eta\nabla L(\theta^{(0)})$

Compute gradient at $\theta^{(1)}$

Move to $\theta^{(2)} = \theta^{(1)} - \eta\nabla L(\theta^{(1)})$

⋮

Gradient Descent

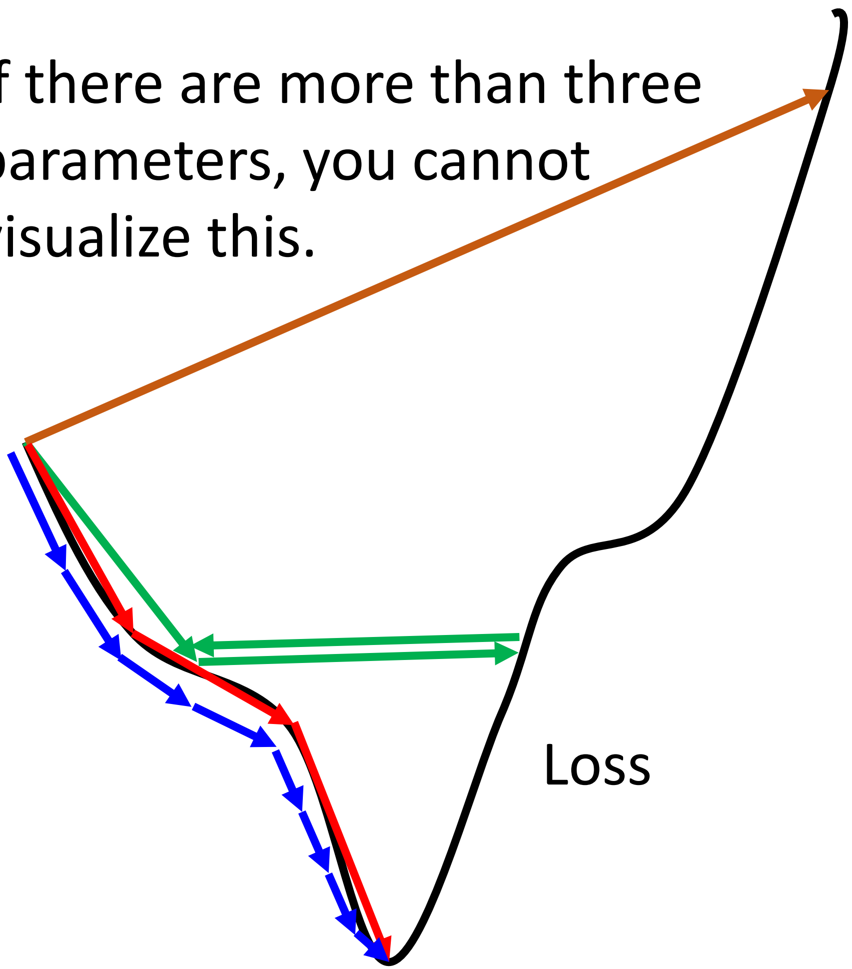
Tip 1: Tuning your learning rates

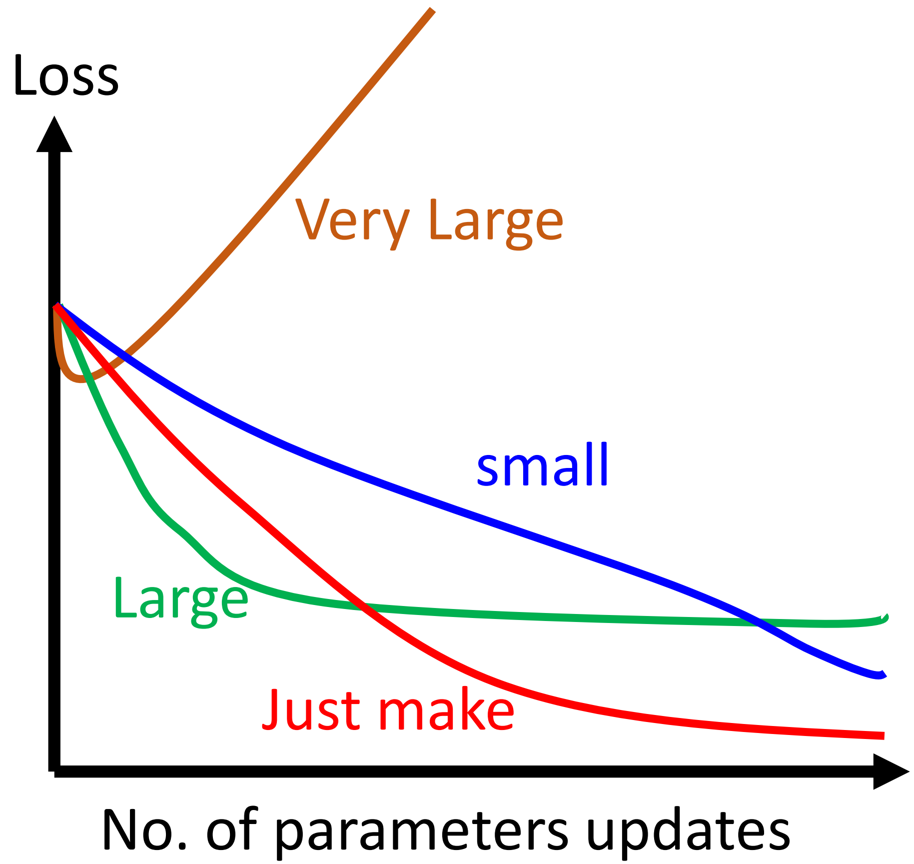
Learning Rate

$$\theta^{(i)} = \theta^{(i-1)} - \eta \nabla L(\theta^{(i-1)})$$

Set the learning rate η carefully

If there are more than three parameters, you cannot visualize this.





But you can always visualize this.

Adaptive Learning Rates

- Popular and Simple Idea: Reduce the learning rate by some factor every few epochs.
  - At the beginning, we are far from the destination, so we use larger learning rate
  - After several epochs, we are close to the destination, so we reduce the learning rate to make sure it converges to the minimum.
  - *e.g.* $1/t$ decay: $\eta^{(t)} = \eta/\sqrt{t+1}$
- Learning rate **cannot be "one size fits all"**
  - Giving different parameters different learning rates

Adagrad

- Divide the learning rate of each parameter by the ***root mean square of all its previous derivatives***

Vanilla Gradient descent

$$w^{(t+1)} \leftarrow w^{(t)} - \eta^{(t)} g^{(t)}$$

w is one parameters

$$\eta^{(t)} = \frac{\eta}{\sqrt{t+1}}$$

$$g^{(t)} = \frac{\partial L(w)}{\partial w} |_{w=w^{(t)}}$$

Adagrad

$$w^{(t+1)} \leftarrow w^{(t)} - \frac{\eta^{(t)}}{\sigma^{(t)}} g^{(t)}$$

$\sigma^{(t)}$: ***root mean square*** of all the previous derivatives of parameter w up to iteration t.

Parameter dependent

Adagrad

$$\eta^{(t)} = \frac{\eta}{\sqrt{t+1}}$$

$$g^{(t)} = \frac{\partial L(w)}{\partial w}|_{w=w^{(t)}}$$

$\sigma^{(t)}$: ***root mean square*** of all the previous derivatives of parameter w up to iteration *t*.

$$w^{(1)} \leftarrow w^{(0)} - \frac{\eta^{(0)}}{\sigma^{(0)}} g^{(0)} \qquad \sigma^{(0)} = \sqrt{(g^{(0)})^2}$$

$$w^{(2)} \leftarrow w^{(1)} - \frac{\eta^{(1)}}{\sigma^{(1)}} g^{(1)} \qquad \sigma^{(1)} = \sqrt{\frac{1}{2}[(g^{(0)})^2 + (g^{(1)})^2]}$$

$$w^{(3)} \leftarrow w^{(2)} - \frac{\eta^{(2)}}{\sigma^{(2)}} g^{(2)} \qquad \sigma^{(2)} = \sqrt{\frac{1}{3}[(g^{(0)})^2 + (g^{(1)})^2 + (g^{(2)})^2]}$$

$$\vdots$$

$$w^{(t+1)} \leftarrow w^{(t)} - \frac{\eta^{(t)}}{\sigma^{(t)}} g^{(t)} \qquad \sigma^{(t)} = \sqrt{\frac{1}{t+1}\sum_{i=0}^{t}(g^{(i)})^2}$$

Adagrad

$$\eta^{(t)} = \frac{\eta}{\sqrt{t+1}}$$

$$g^{(t)} = \frac{\partial L(w)}{\partial w}|_{w=w^{(t)}}$$

- Divide the learning rate of each parameter by the ***root mean square of all its previous derivatives***

$$w^{(t+1)} \leftarrow w^{(t)} - \frac{\eta^{(t)}}{\sigma^{(t)}} g^{(t)}$$

$$\eta^{(t)} = \frac{\eta}{\cancel{\sqrt{t+1}}}$$

$$\sigma^{(t)} = \sqrt{\frac{1}{\cancel{t+1}} \sum_{i=0}^{t} (g^{(i)})^2}$$

$$w^{(t+1)} \leftarrow w^{(t)} - \frac{\eta}{\sqrt{\sum_{i=0}^{t} (g^{(i)})^2}} g^{(t)}$$

- Other adaptive learning rate methods: Adadelta, Adam,…

Contradiction?

$$\eta^{(t)} = \frac{\eta}{\sqrt{t+1}}$$

$$g^{(t)} = \frac{\partial L(w)}{\partial w}|_{w=w^{(t)}}$$

Vanilla Gradient descent

$$w^{(t+1)} \leftarrow w^{(t)} - \eta^{(t)} \boxed{g^{(t)}}$$

Larger gradient, larger step

Adagrad

$$w^{(t+1)} \leftarrow w^{(t)} - \frac{\eta}{\sqrt{\underbrace{\sum_{i=0}^{t} (g^{(i)})^2}}} \boxed{g^{(t)}}$$

Larger gradient, larger step

Larger gradient, smaller step

Gradient Descent

Tip 2: Stochastic Gradient Descent

Make the training faster

Stochastic Gradient Descent

$$L = \boxed{\sum_n} \left(y^{(n)} - \left(b + \sum_i w_i x_i^{(n)} \right) \right)^2$$

Loss is the summation over all training examples

- ***Gradient Descent***

$$\theta^{(i+1)} \leftarrow \theta^{(i)} - \eta \nabla L(\theta^{(i)})$$

- ***Stochastic Gradient Descent (SGD)***

Faster!

Randomly pick one example $x^{(n)}$ to update parameters

$$\boxed{L^{(n)}} = \left(y^{(n)} - \left(b + \sum_i w_i x_i^{(n)} \right) \right)^2$$

$$\theta^{(i+1)} \leftarrow \theta^{(i)} - \eta \nabla L(\theta^{(i)})$$

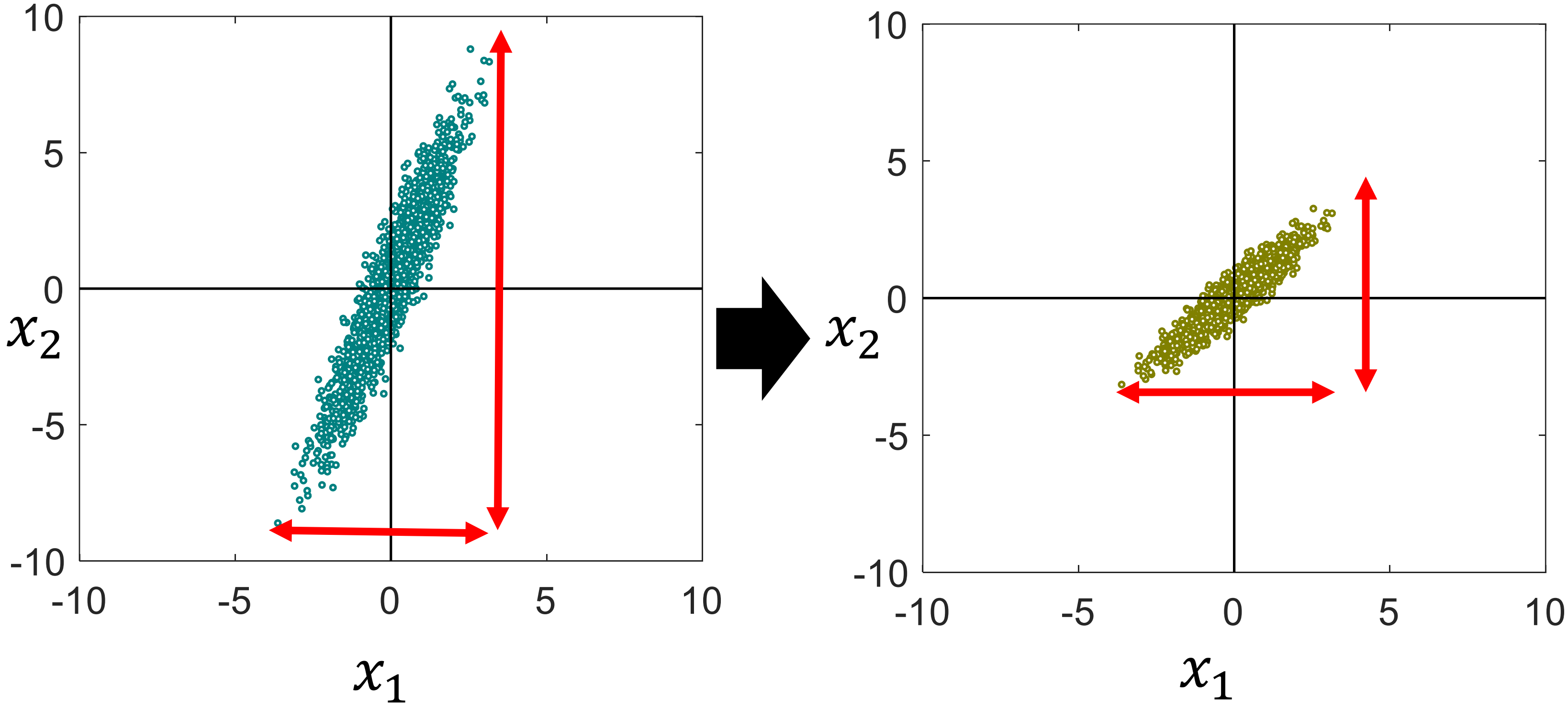
Loss for only one example, *i.e.* n^th^ sample

Gradient Descent

Tip 3: Feature Scaling

Feature Scaling/Feature Normalization

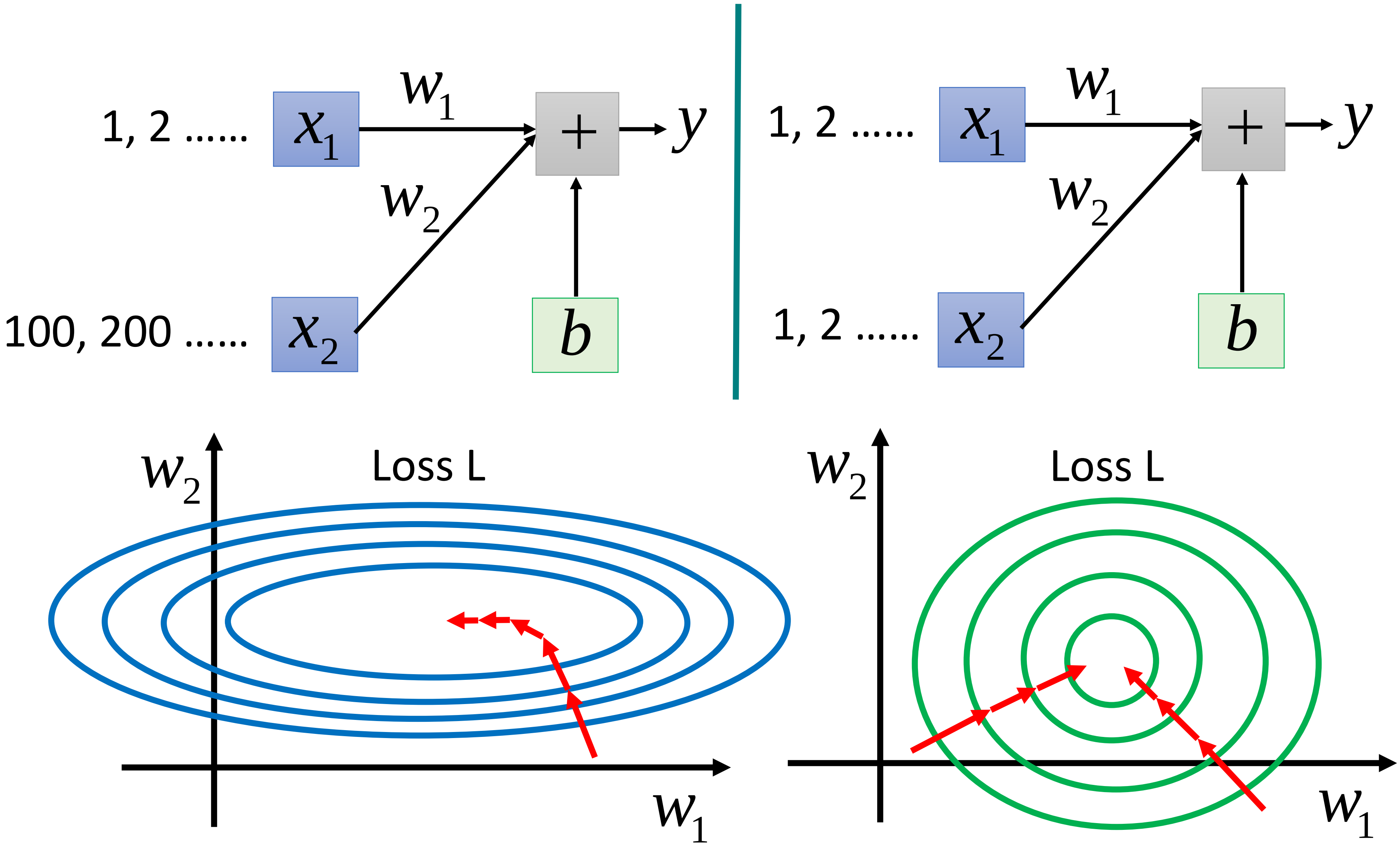
$$y = b + w_1x_1 + w_2x_2$$



Make different features have the same scaling

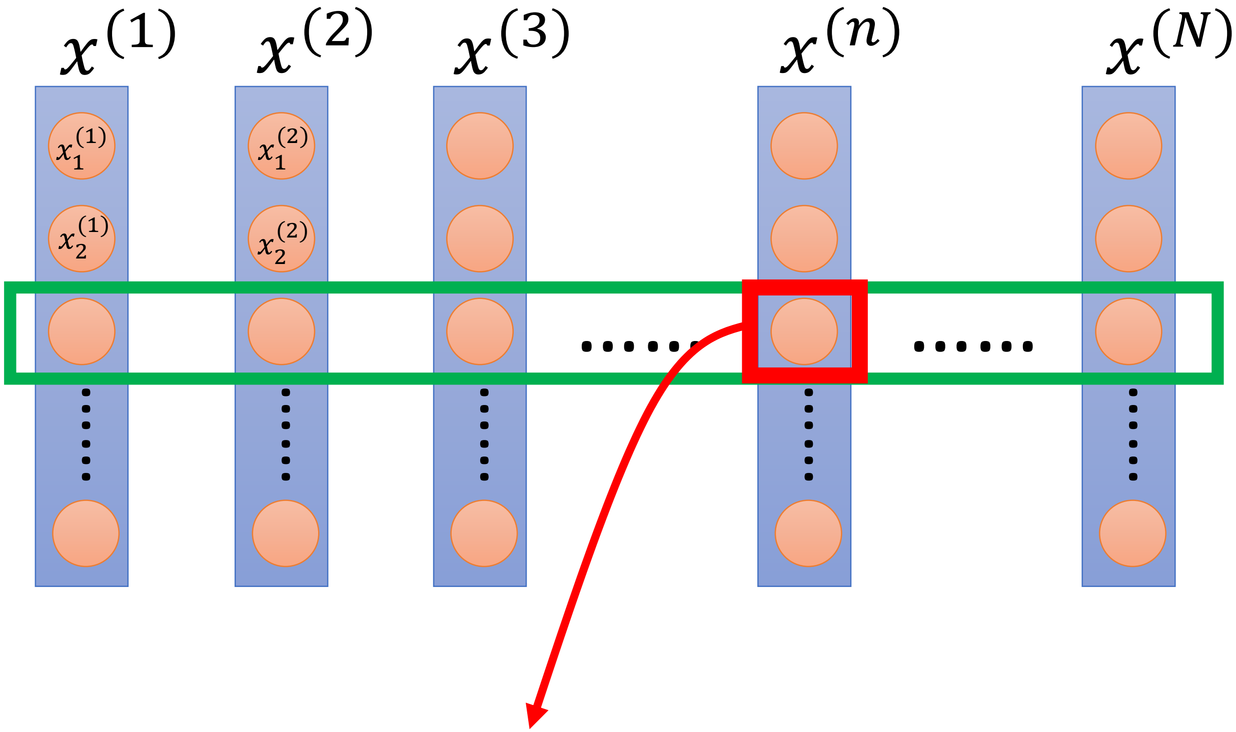
Feature Scaling

$$y = b + w_1 x_1 + w_2 x_2$$



Feature Scaling

$$m_i = \frac{1}{N}\sum_{i=1}^{N} x_i^{(n)}$$



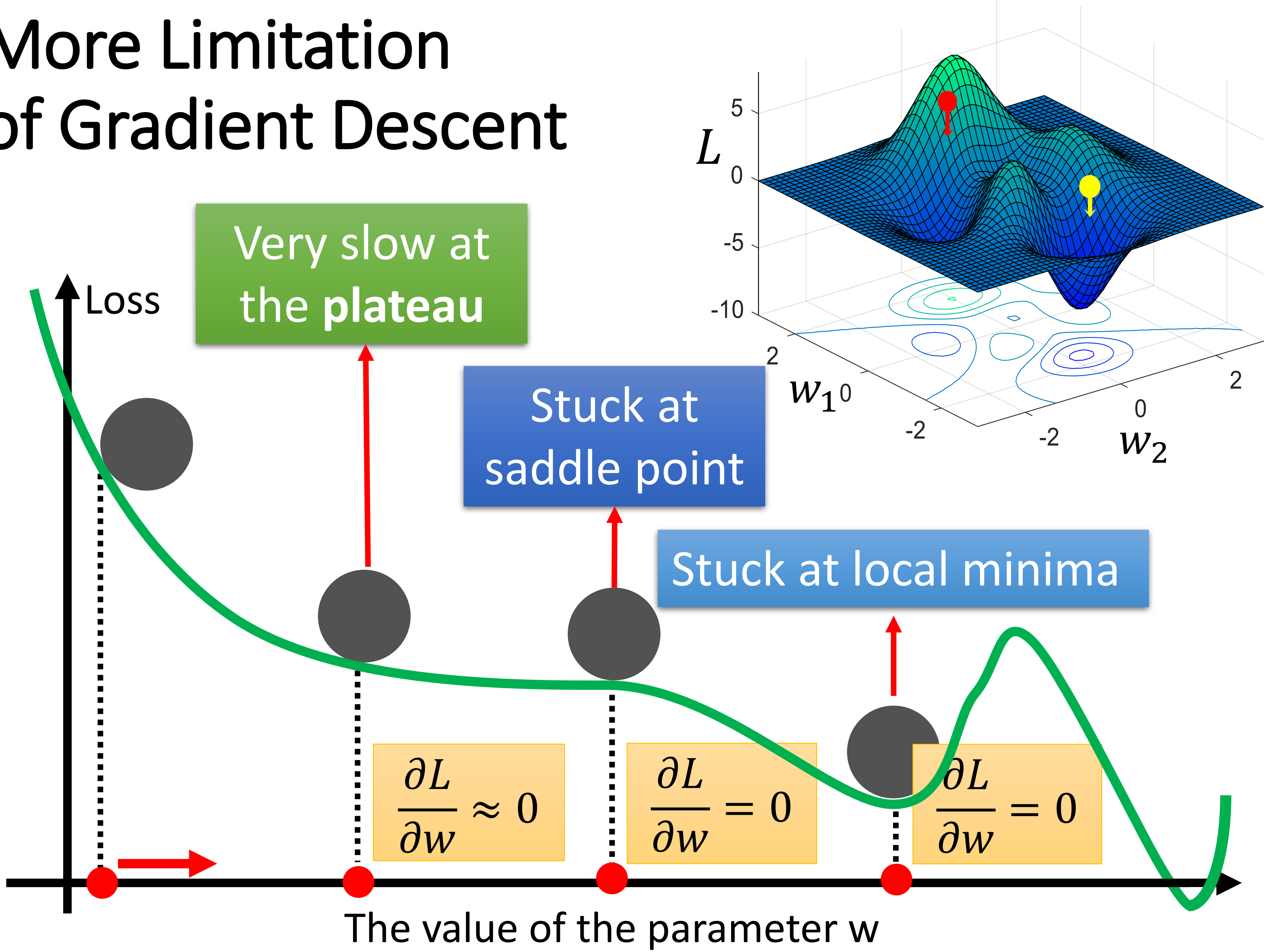
For each dimension i:

- mean: m_i
- standard deviation: σ_i

$$x_i^{(n)} \leftarrow \frac{x_i^{(n)} - m_i}{\sigma_i}$$

Make each feature component zero mean and unit standard deviation.

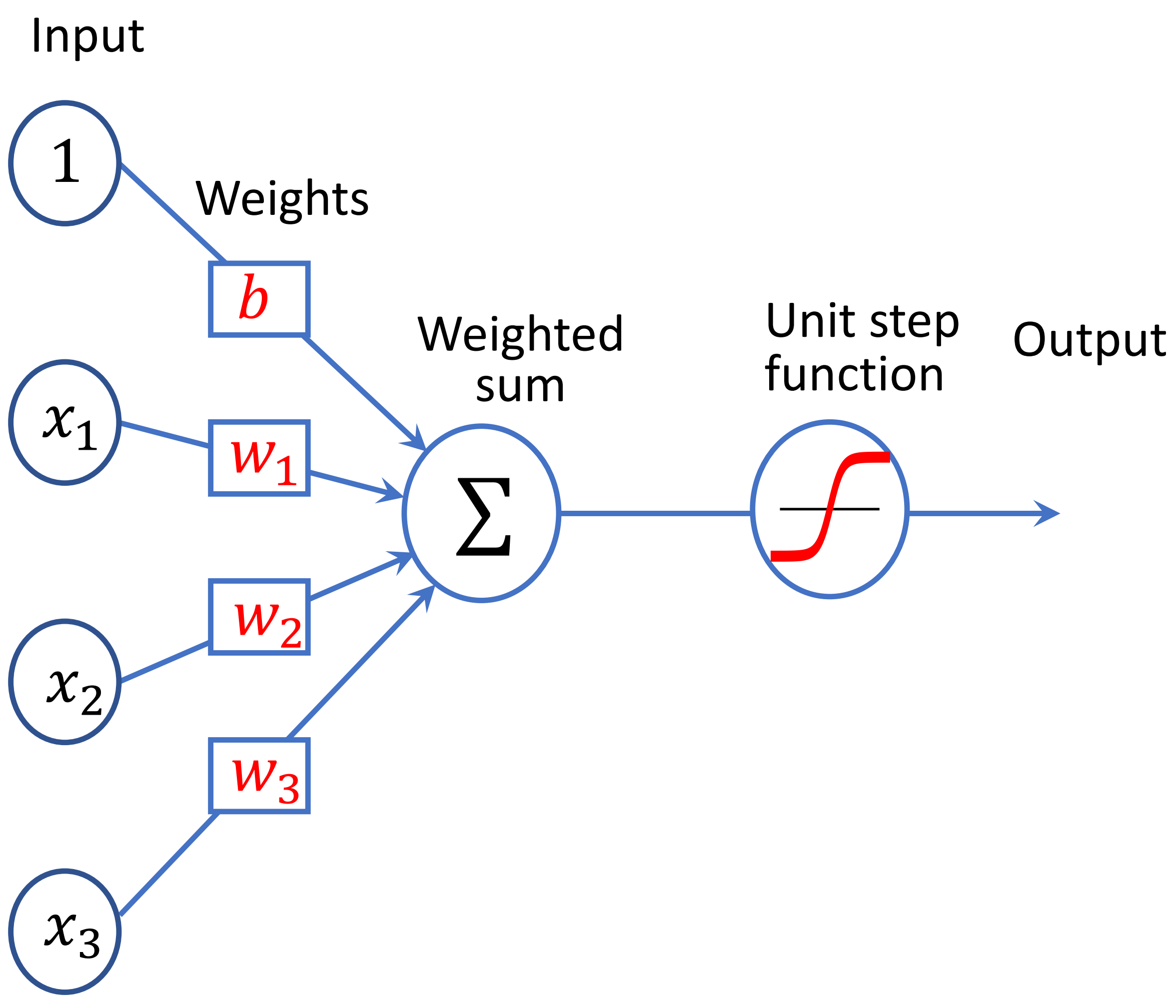
More Limitation of Gradient Descent



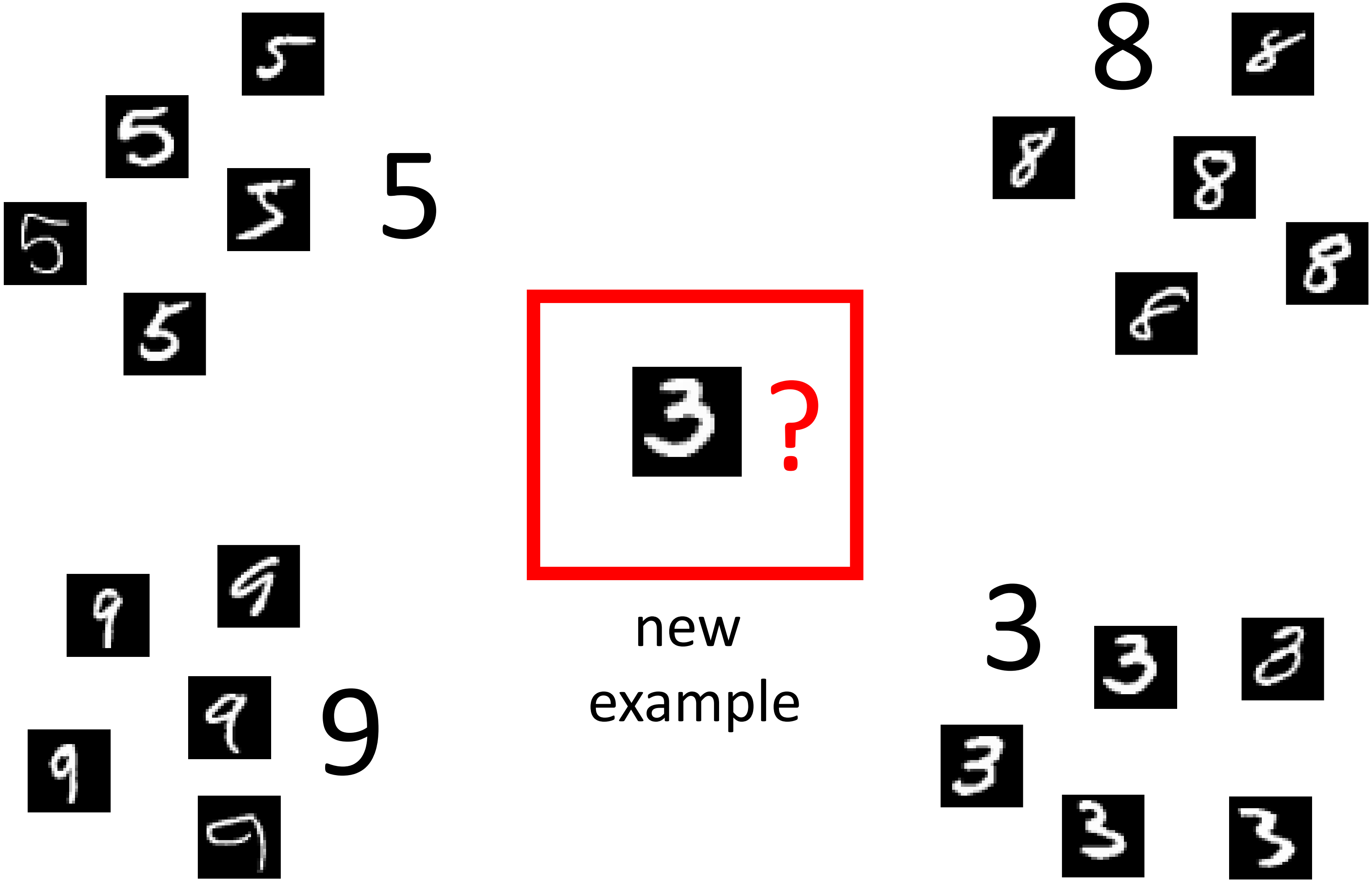
An overview of gradient descent optimization algorithms

- https://arxiv.org/pdf/1609.04747.pdf

Classification: Logistic Regression



Supervised Classification



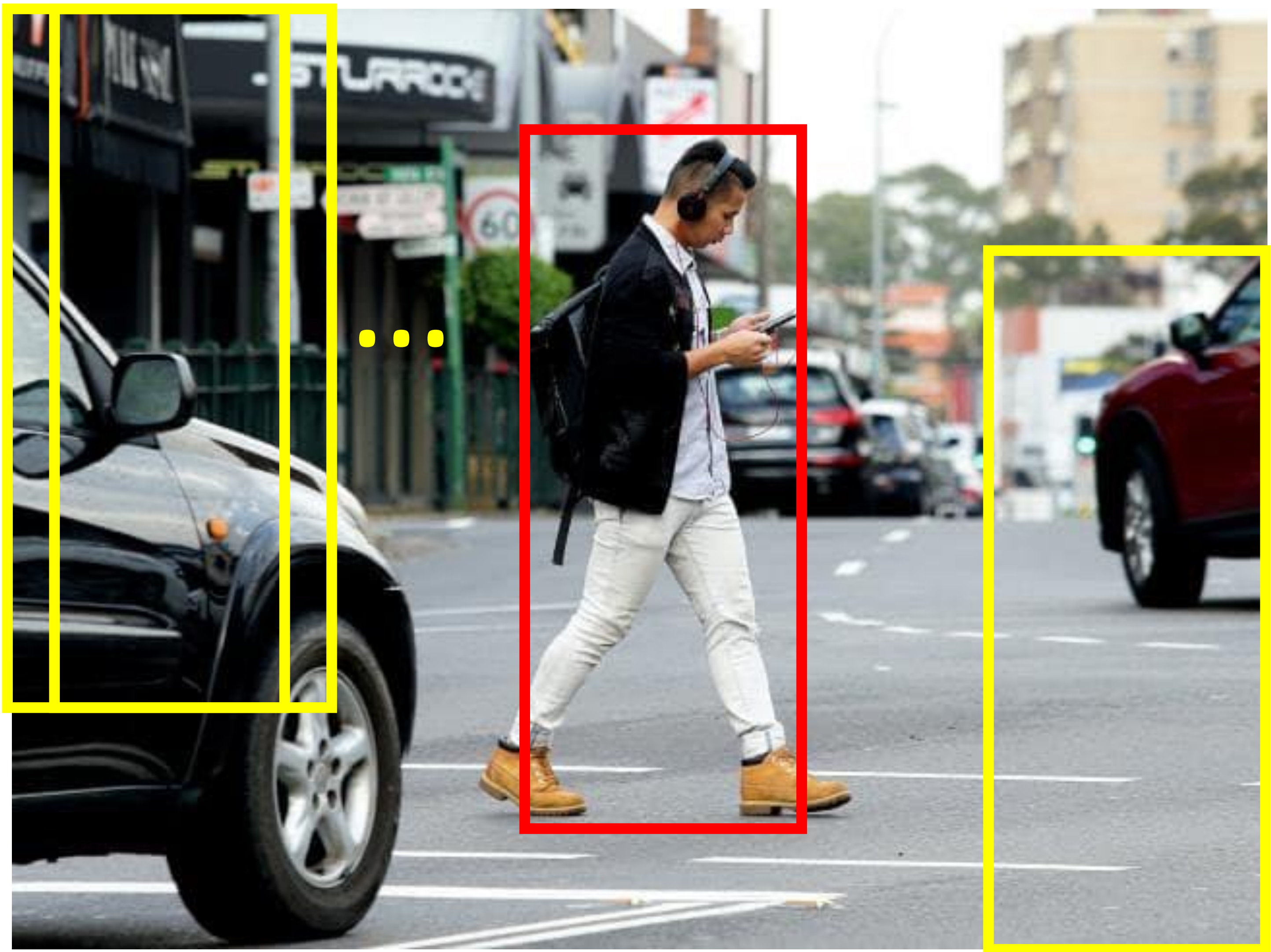
Classification: object classification

ImageNet Challenge: classification of 1000 object category

Classification: face recognition

Pedestrian Detection

Pedestrian Detection



Classification

- Email: Spam / Not Spam?
- Pedestrian Detection: Pedestrian / Not Pedestrian?
- Tumor: Malignant / Benign ?

$y \in \{0,1\}$

0: "Negative Class" (*e.g.* Not Pedestrian)
1: "Positive Class" (*e.g.* Pedestrian)

Values 0 and 1 are somewhat arbitrary.

$y \in \{-1,1\}$

0: "Negative Class" (*e.g.* Not Pedestrian)
1: "Positive Class" (*e.g.* Pedestrian)

Linear Classifier

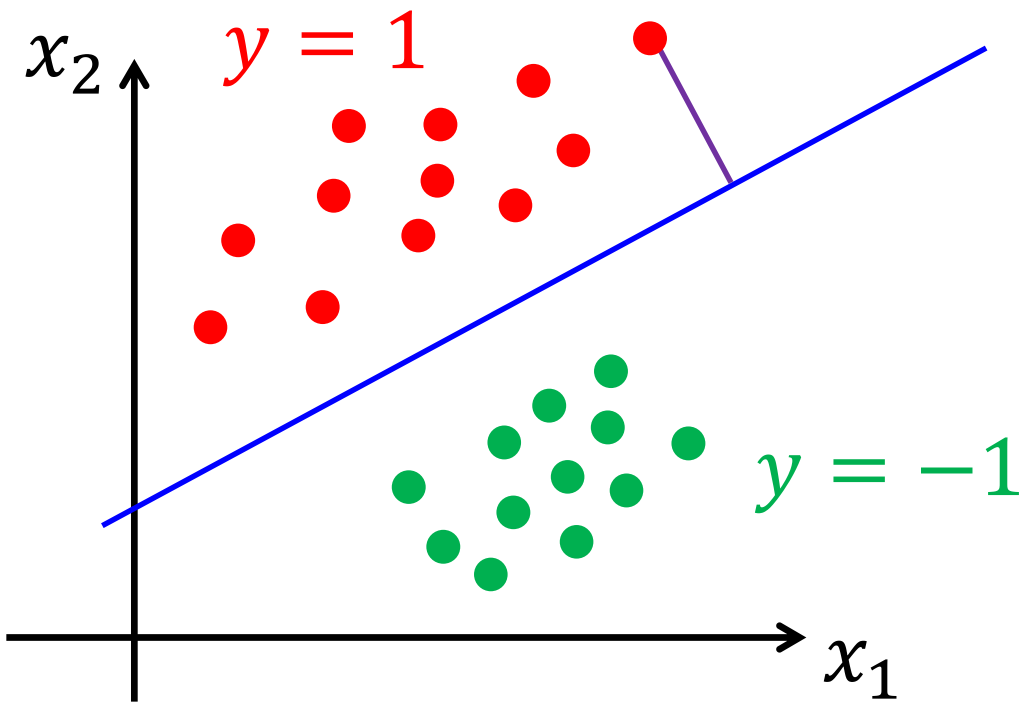
- Training data for Classification

$$\left\{\left(\boldsymbol{x}^{(k)}, y^{(k)}\right)\right\}_{k=1}^{N} \qquad y^{(k)} \in \{-1,1\}$$

Prediction function:

$$f_{w,b}(x) = sgn(\boldsymbol{w}^T\boldsymbol{x} + b)$$

$$\frac{|\boldsymbol{w}^T\boldsymbol{x} + b|}{\sqrt{\sum_i w_i^2}}$$



Logistic Regression Model

Want $0 \leq f_{w,b}(x) \leq 1$

$\sigma(z) \geq 0.5$, class 1

$\sigma(z) < 0.5$, class 2

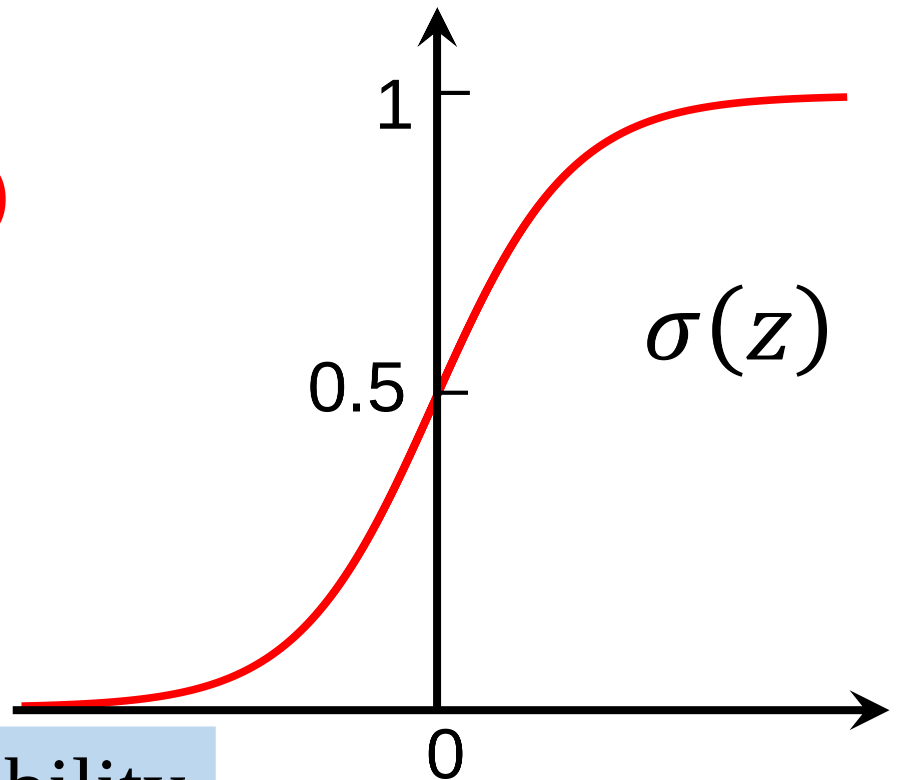
$$f_{w,b}(x) = \boldsymbol{w}^T\boldsymbol{x} + b?$$

$$f_{w,b}(x) = \sigma(\boldsymbol{w}^T\boldsymbol{x} + b)$$

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

Sigmoid function

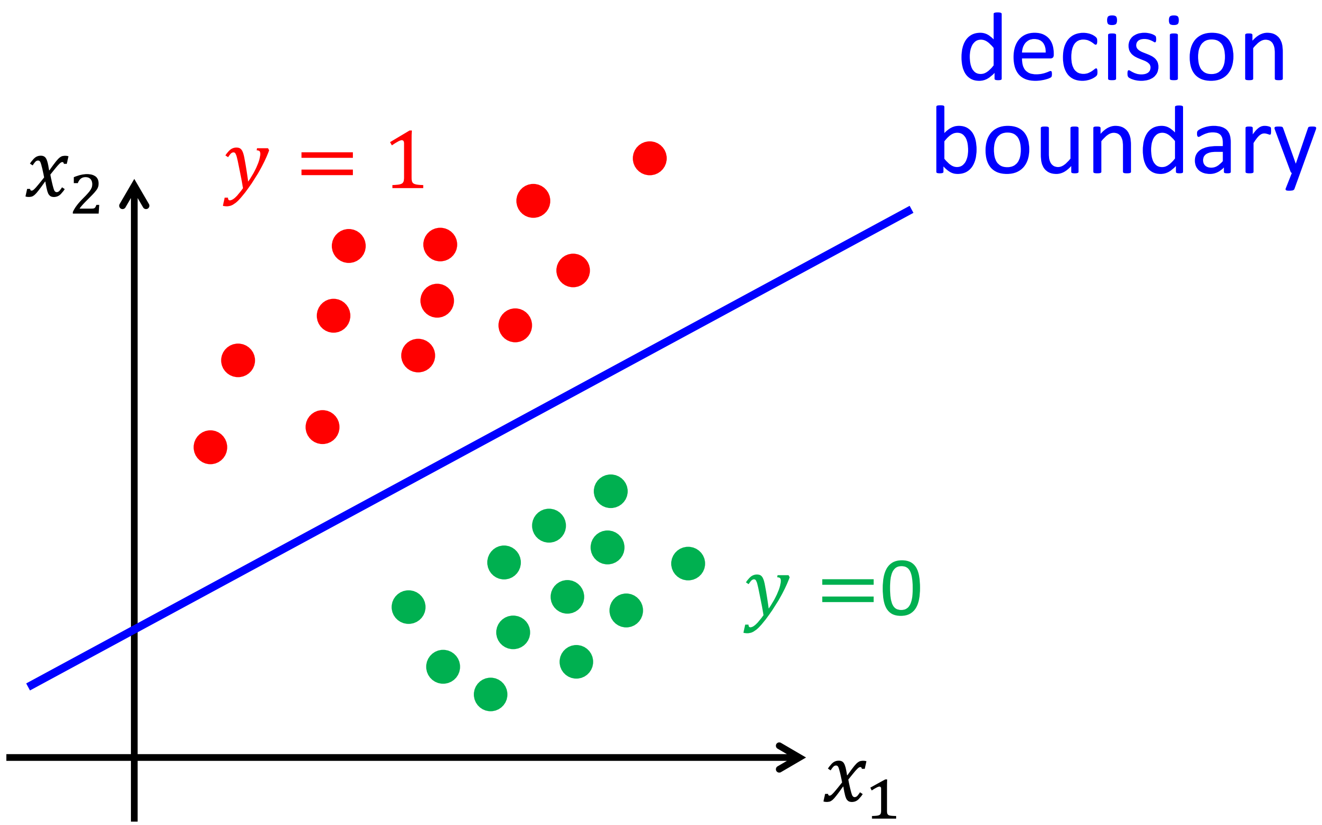
Logistic function



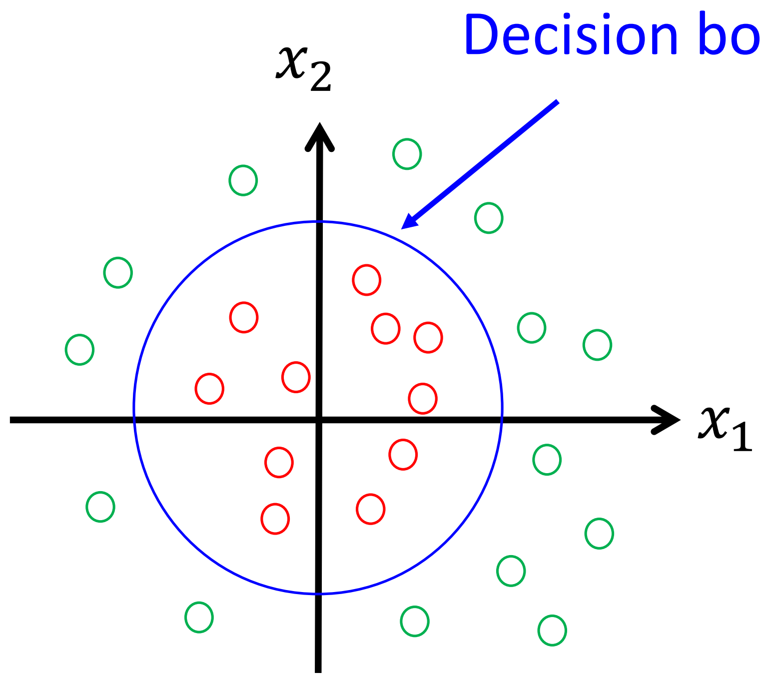
$\sigma(z)$ means posterior Probability

Decision Boundary

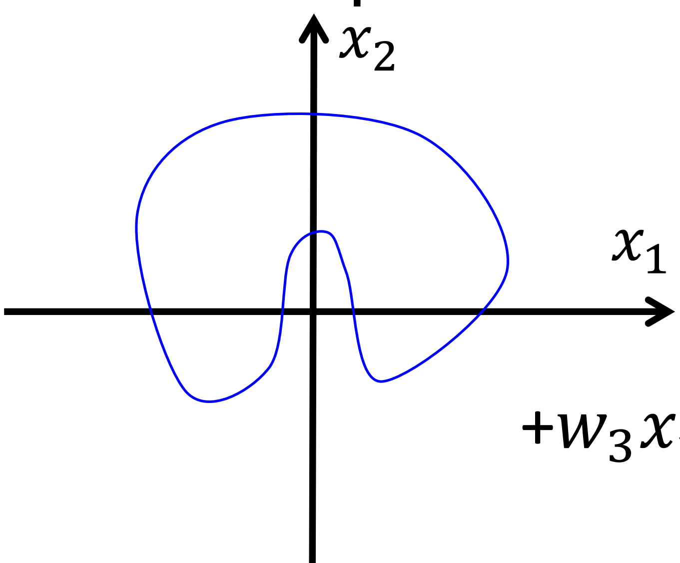
$$f_{\boldsymbol{w},b}(x) = \sigma(b + w_1 x_1 + w_2 x_2)$$



Nonlinear decision boundaries



$$f_{w,b}(x) = \sigma(w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_2^2)$$



$$f_w(x) = \sigma(w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_1^2 x_2 + w_5 x_1^2 x_2^2 + w_6 x_1^3 x_2 + \cdots)$$

Step 1: Function Set

- Function set: Including all different w and b

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

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

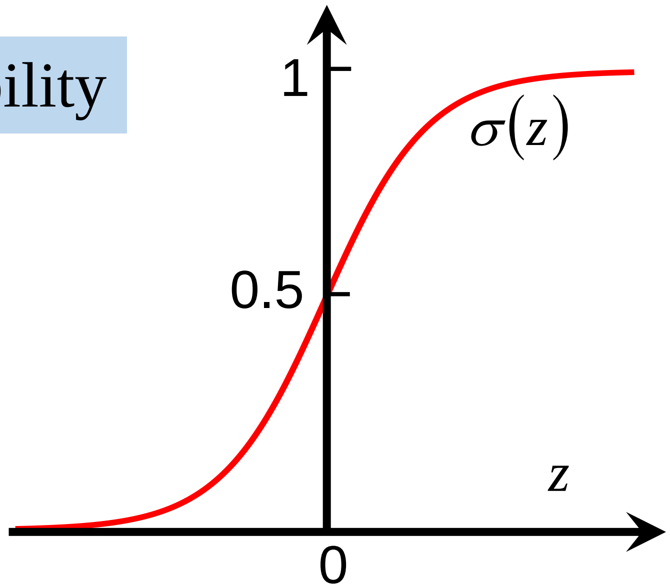
Posterior Probability

- Hypothesis

sigmoid function (logistic function)

$$z = \boldsymbol{w}^T \boldsymbol{x} + b \ = \sum_i w_i x_i + b$$

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



Step 1: Function Set

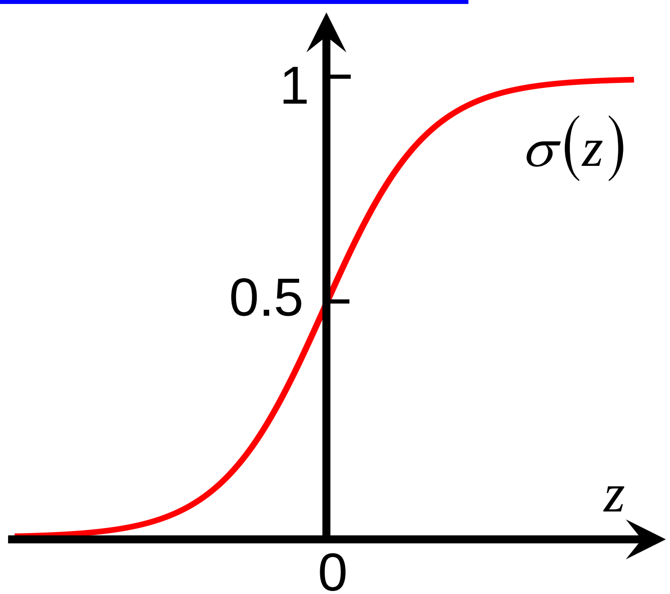
Function set: **Including all different w and b**

$$\begin{cases} z \geq 0 & \text{class 1} \\ z < 0 & \text{class 2} \end{cases}$$

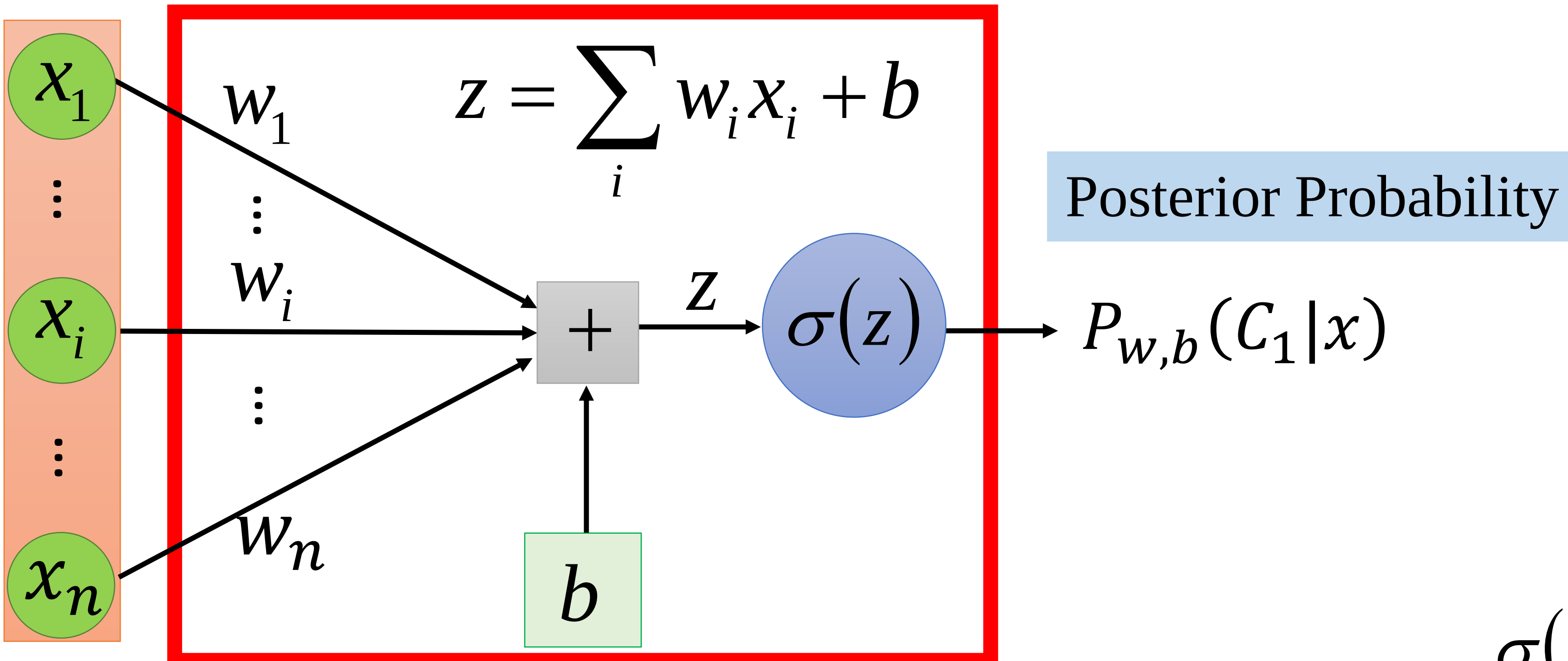
$$P_{w,b}(C_1|x) = \sigma(z)$$

$$z = \boldsymbol{w}^T \boldsymbol{x} + b \quad = \sum_i w_i x_i + b$$

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



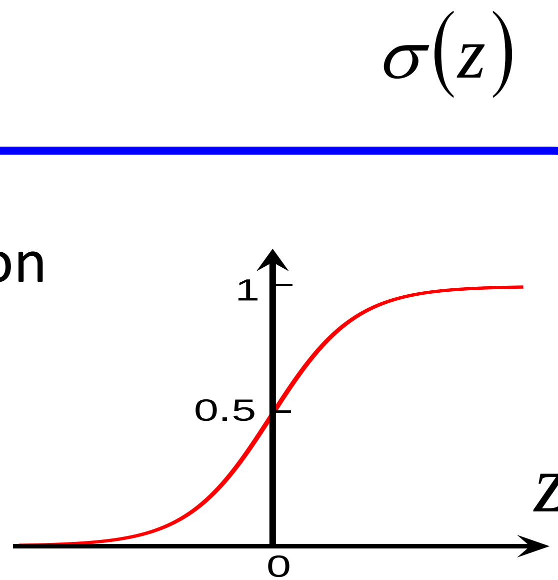
Step 1: Function Set



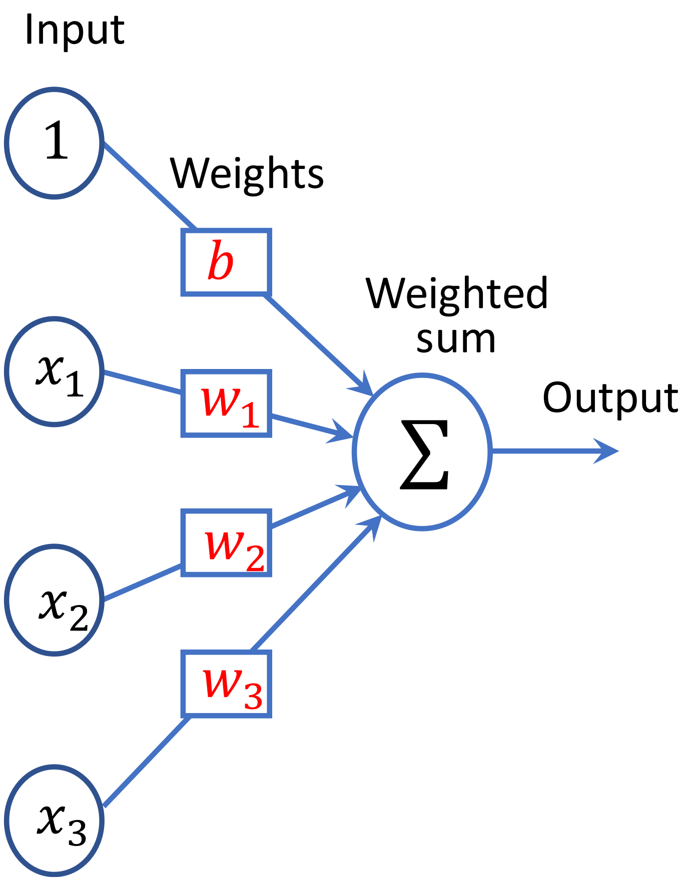
Logistic Regression

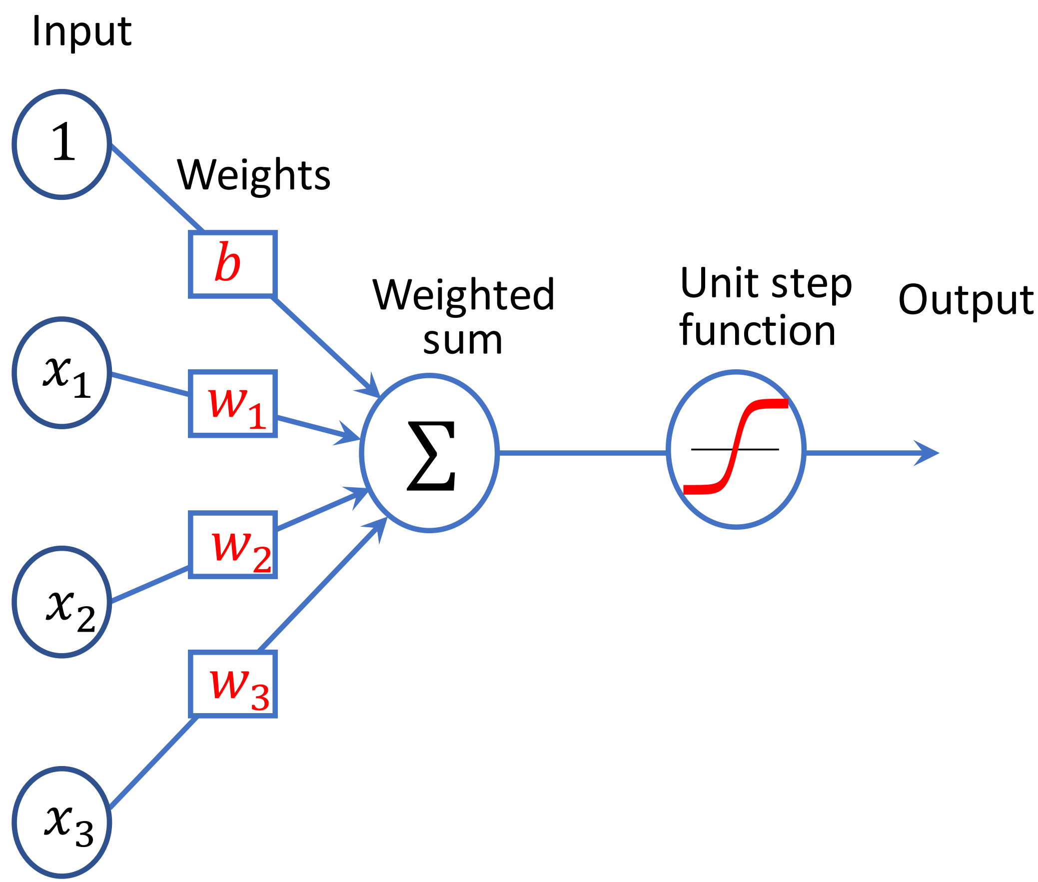
Sigmoid Function

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



Linear Regression (Prediction) vs. Logistic Regression (Classification)





Step 2: Goodness of a Function

Training Data

$x^{(1)}$	$x^{(2)}$	$x^{(3)}$	……	$x^{(N)}$
C_1	C_1	C_2		C_1

$$P_{w,b}(C_1|x) = \frac{1}{1+exp(-z)}$$

$$z = \boldsymbol{w}^T\boldsymbol{x} + b$$

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?

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

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

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

$$\begin{array}{ccc} x^{(1)} & x^{(2)} & x^{(3)} \\ C_1 & C_1 & C_2 \end{array} \quad \cdots\cdots \quad \Rightarrow \quad \begin{array}{ccc} x^{(1)} & x^{(2)} & x^{(3)} \\ y^{(1)}=1 & y^{(2)}=1 & y^{(3)}=0 \end{array} \quad \cdots\cdots$$

$y^{(k)}$: 1 for class 1, 0 for class 2

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

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

$$-lnL(w,b)$$

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

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

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

$$\vdots$$

$$\boxed{\begin{matrix} x^{(1)} & x^{(2)} & x^{(3)} & \\ C_1 & C_1 & C_2 & \cdots\cdots \end{matrix}} \Rightarrow \boxed{\begin{matrix} x^{(1)} & x^{(2)} & x^{(3)} & \\ y^{(1)} = 1 & y^{(2)} = 1 & y^{(3)} = 0 & \cdots\cdots \end{matrix}}$$

$y^{(k)}$: 1 for class 1, 0 for class 2

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

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

$$-lnL(w,b)$$

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

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

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

$$\vdots$$

$$\|$$

$$\sum_{k=1}^{N} -\left[y^{(k)} lnf_{w,b}\left(x^{(k)}\right) + \left(1 - y^{(k)}\right) ln\left(1 - f_{w,b}\left(x^{(k)}\right)\right)\right]$$

Step 2: Goodness of a Function

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

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

$y^{(k)}$: 1 for class 1, 0 for class 2

$$= \sum_{k=1}^{N} \underbrace{- \left[y^{(k)} lnf_{w,b}\left(x^{(k)}\right) + \left(1 - y^{(k)}\right) ln\left(1 - f_{w,b}\left(x^{(k)}\right)\right) \right]}_{\text{Cross entropy between two Bernoulli distribution}}$$

Distribution p:

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

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

$\longleftrightarrow$ cross entropy

Distribution q:

$q(x=1) = f\left(x^{(n)}\right)$

$q(x=0) = 1 - f\left(x^{(n)}\right)$

$$H(p,q) = -\sum_{x} p(x) ln\left(q(x)\right)$$

Step 2: Goodness of a Function

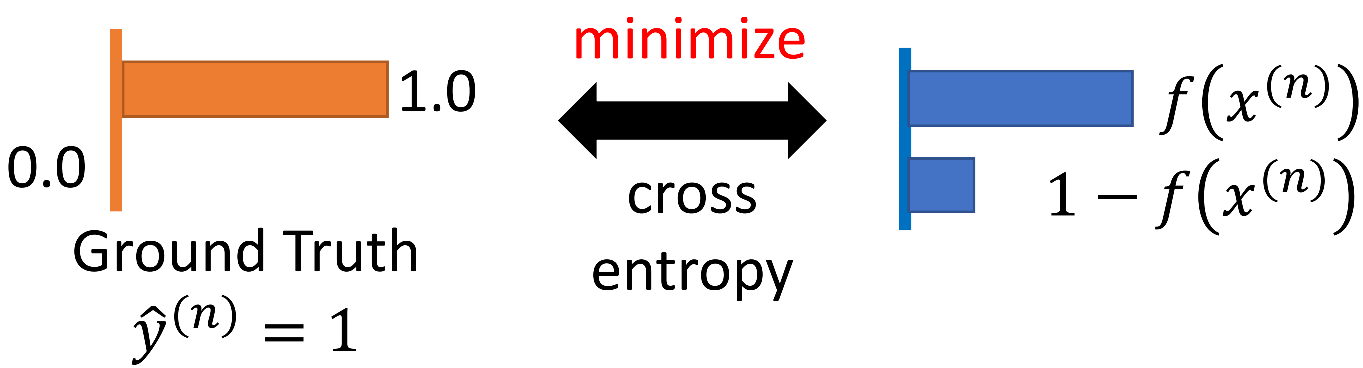
$$L(w,b) = f_{w,b}\left(x^{(1)}\right) f_{w,b}\left(x^{(2)}\right)\left(1 - f_{w,b}\left(x^{(3)}\right)\right) \cdots f_{w,b}\left(x^{(N)}\right)$$

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

$y^{(k)}$: 1 for class 1, 0 for class 2

$$= \sum_{k=1}^{N} - \left[y^{(k)} lnf_{w,b}\left(x^{(k)}\right) + \left(1 - y^{(k)}\right) ln\left(1 - f_{w,b}\left(x^{(k)}\right)\right) \right]$$

Minimize cross entropy between two Bernoulli distribution



Step 3: Find the best function

$$\frac{d \ln x}{dx} = \frac{1}{x}$$

chain rule

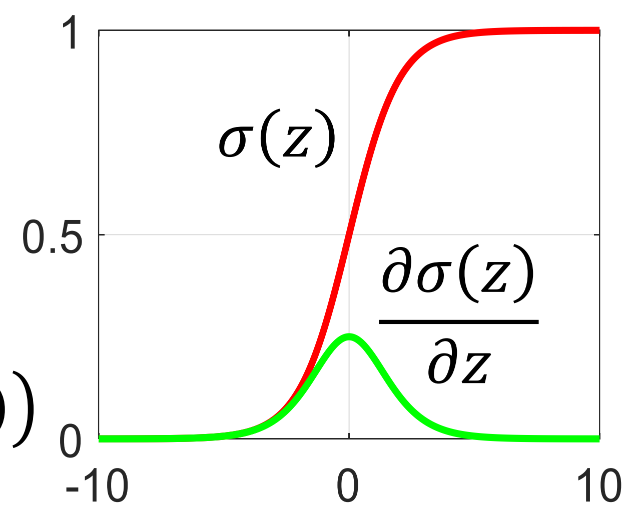
$$\left(1 - f_{w,b}\left(x^{(k)}\right)\right) x_i^{(k)}$$

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

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

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

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



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

$$z = \boldsymbol{w}^T \boldsymbol{x} + b = \sum_i w_i x_i + b$$

Step 3: Find the best function

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

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

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

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

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

$$z = \boldsymbol{w}^T\boldsymbol{x} + b = \sum_i w_i x_i + b$$

Step 3: Find the best function

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

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

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

$$= \sum_k -\left[y^{(k)} - \cancel{y^{(k)} f_{w,b}\left(x^{(k)}\right)} - f_{w,b}\left(x^{(k)}\right) + \cancel{y^{(k)} f_{w,b}\left(x^{(k)}\right)}\right] x_i^{(k)}$$

$$= \sum_k -\left(y^{(k)} - f_{w,b}\left(x^{(k)}\right)\right) x_i^{(k)}$$

Larger difference, larger update

$$w_i \leftarrow w_i - \eta \sum_k -\left(\underline{y^{(k)} - f_{w,b}\left(x^{(k)}\right)}\right) x_i^{(k)}$$

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

Multiclass Classification (3 classes as example)

C$_1$: $\boldsymbol{w}^{(1)}, b_1$; $z_1 = \boldsymbol{w}^{(1)} \cdot \boldsymbol{x} + b_1$

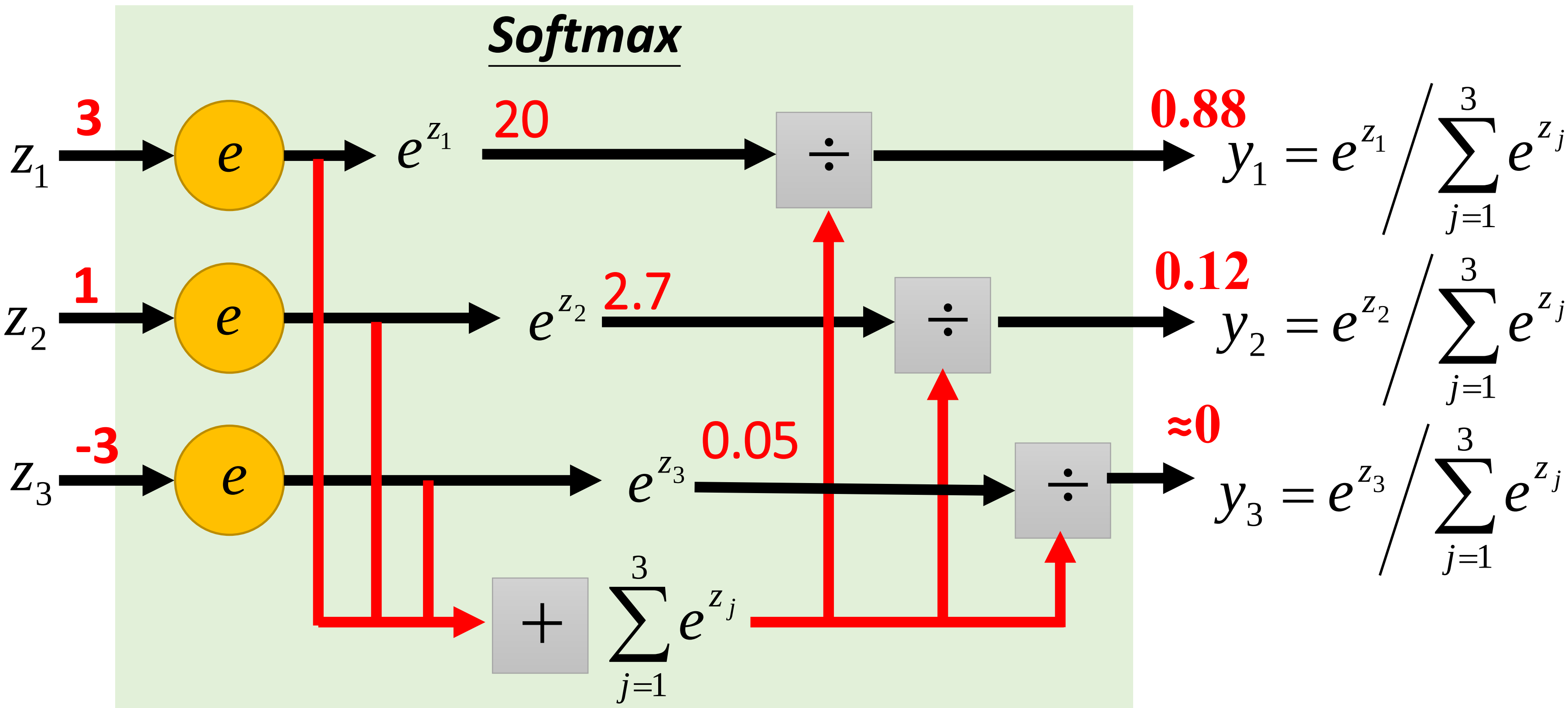
C$_2$: $\boldsymbol{w}^{(2)}, b_2$; $z_2 = \boldsymbol{w}^{(2)} \cdot \boldsymbol{x} + b_2$

C$_3$: $\boldsymbol{w}^{(3)}, b_3$; $z_3 = \boldsymbol{w}^{(3)} \cdot \boldsymbol{x} + b_3$

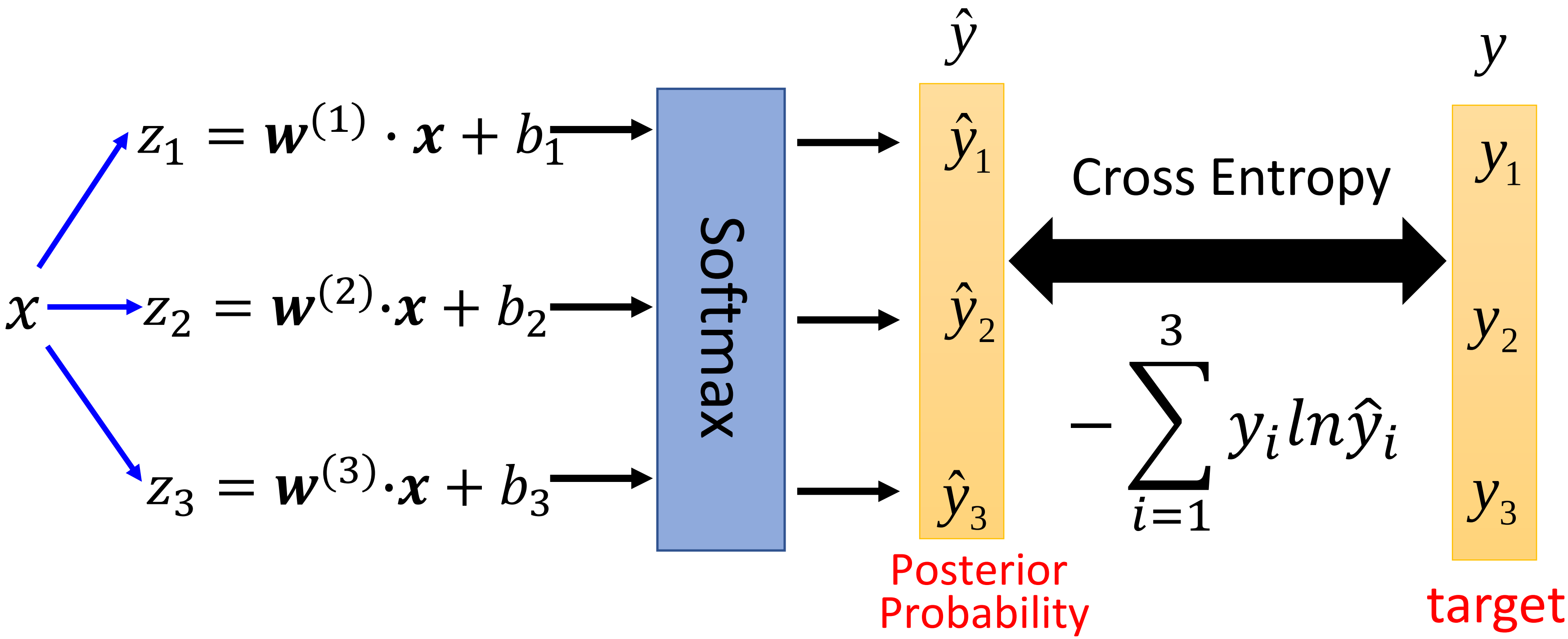
Probability:

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

$$y_i = P(C_i \mid x)$$



Multiclass Classification (3 classes as example)



If $\boldsymbol{x} \in$ class 1

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

$$-ln\hat{y}_1$$

If $\boldsymbol{x} \in$ class 2

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

$$-ln\hat{y}_2$$

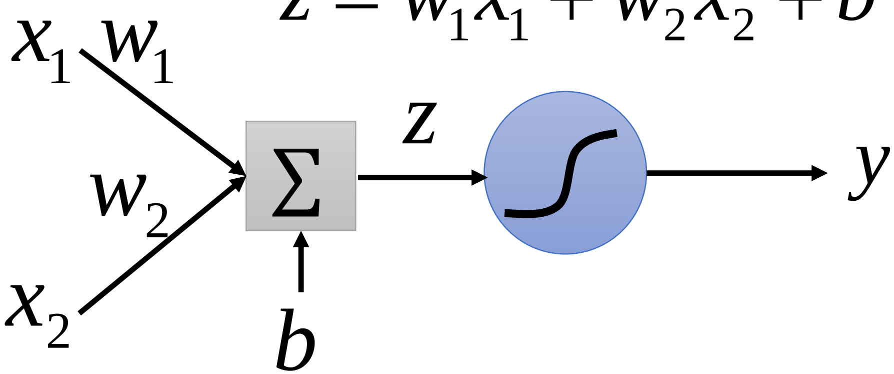
If $\boldsymbol{x} \in$ class 3

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

$$-ln\hat{y}_3$$

Limitation of Logistic Regression

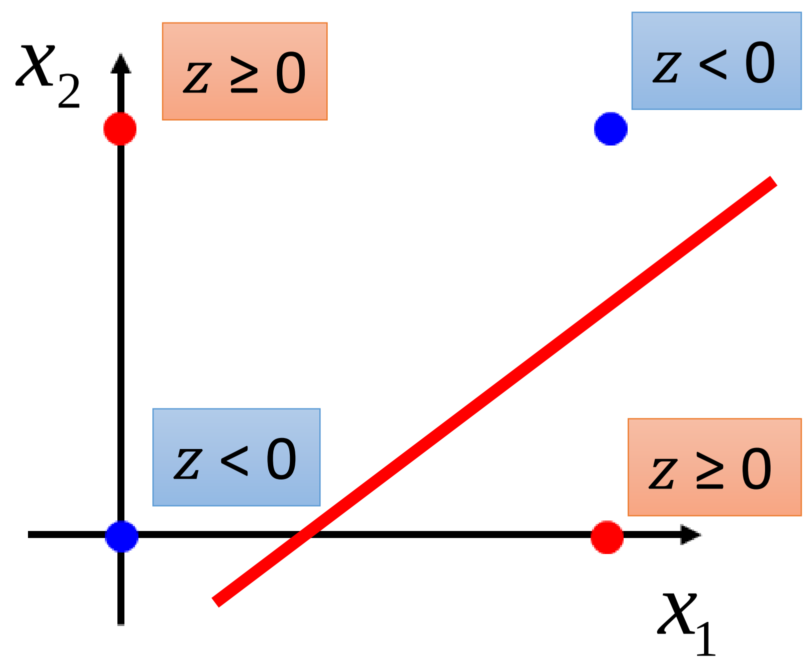
$$z = w_1 x_1 + w_2 x_2 + b$$



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

Can we?

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

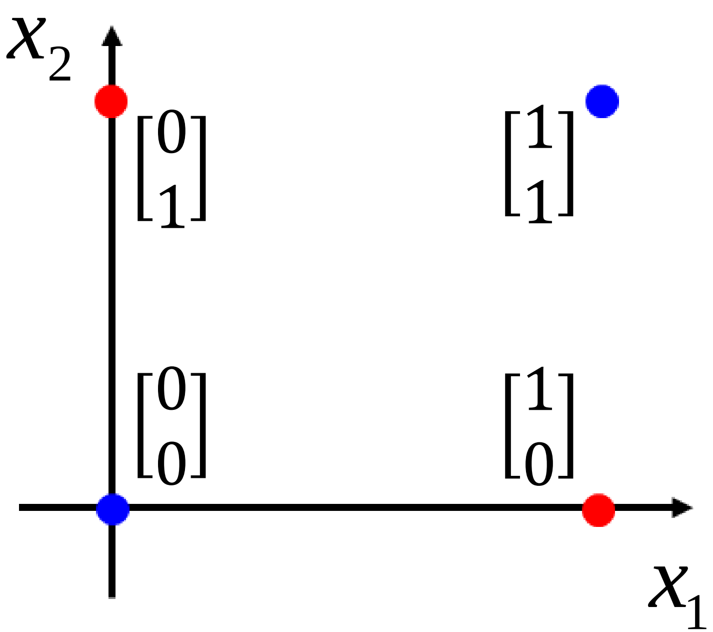


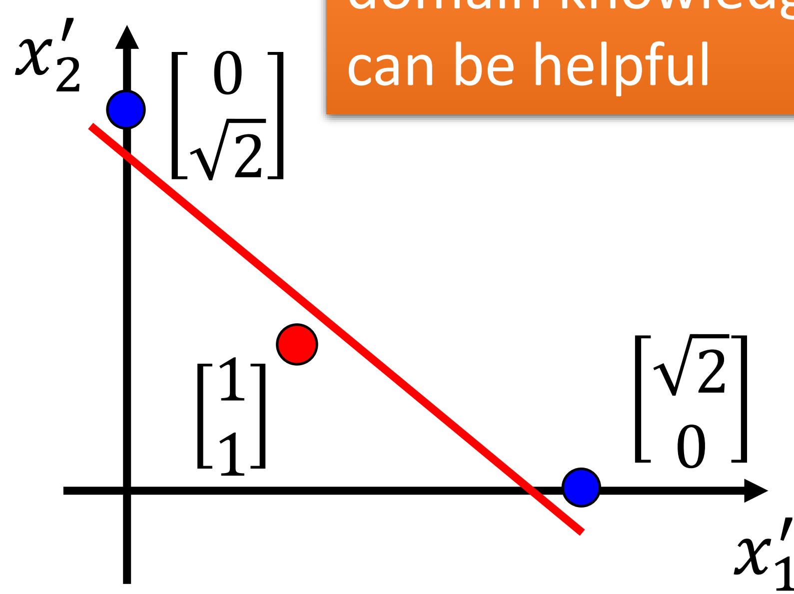
Limitation of Logistic Regression

- ***Feature Representation***

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

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

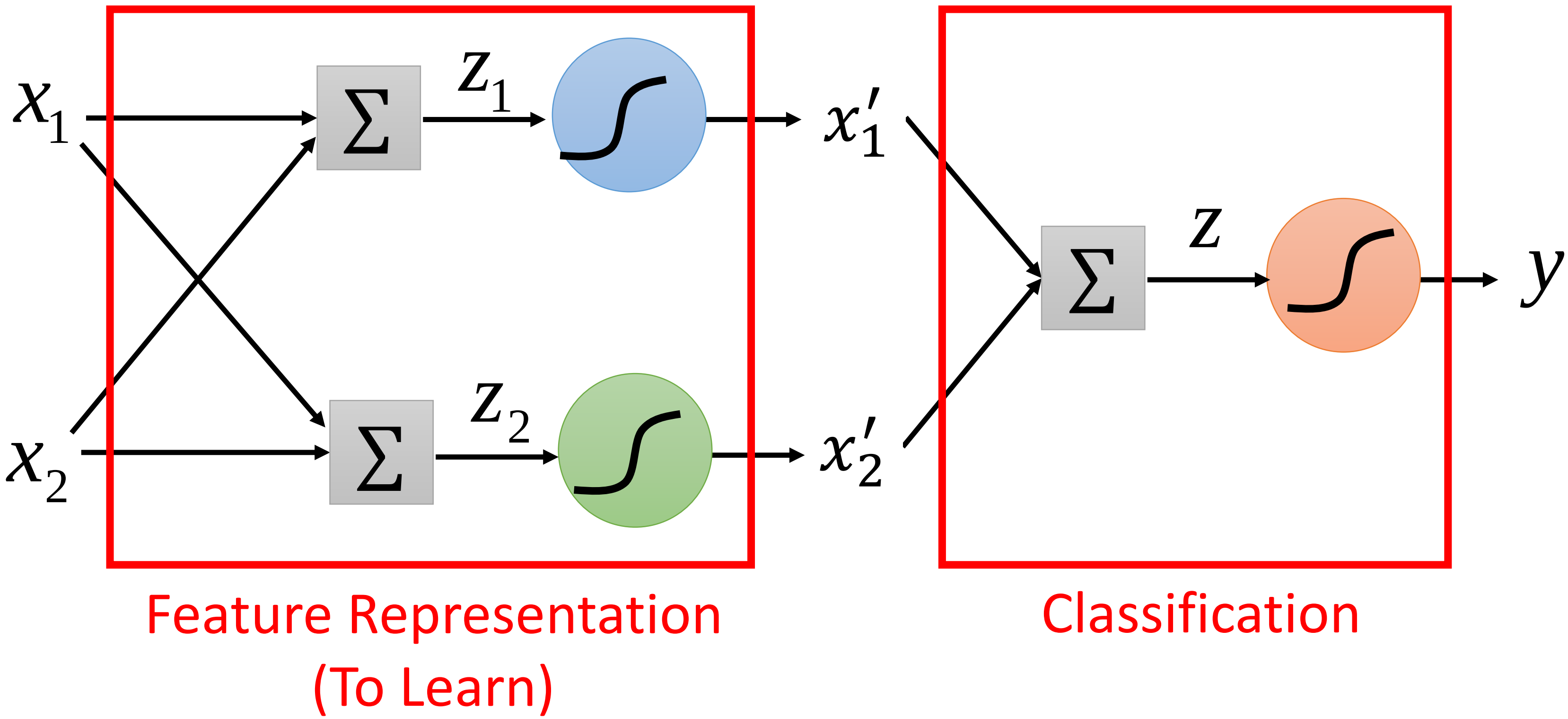




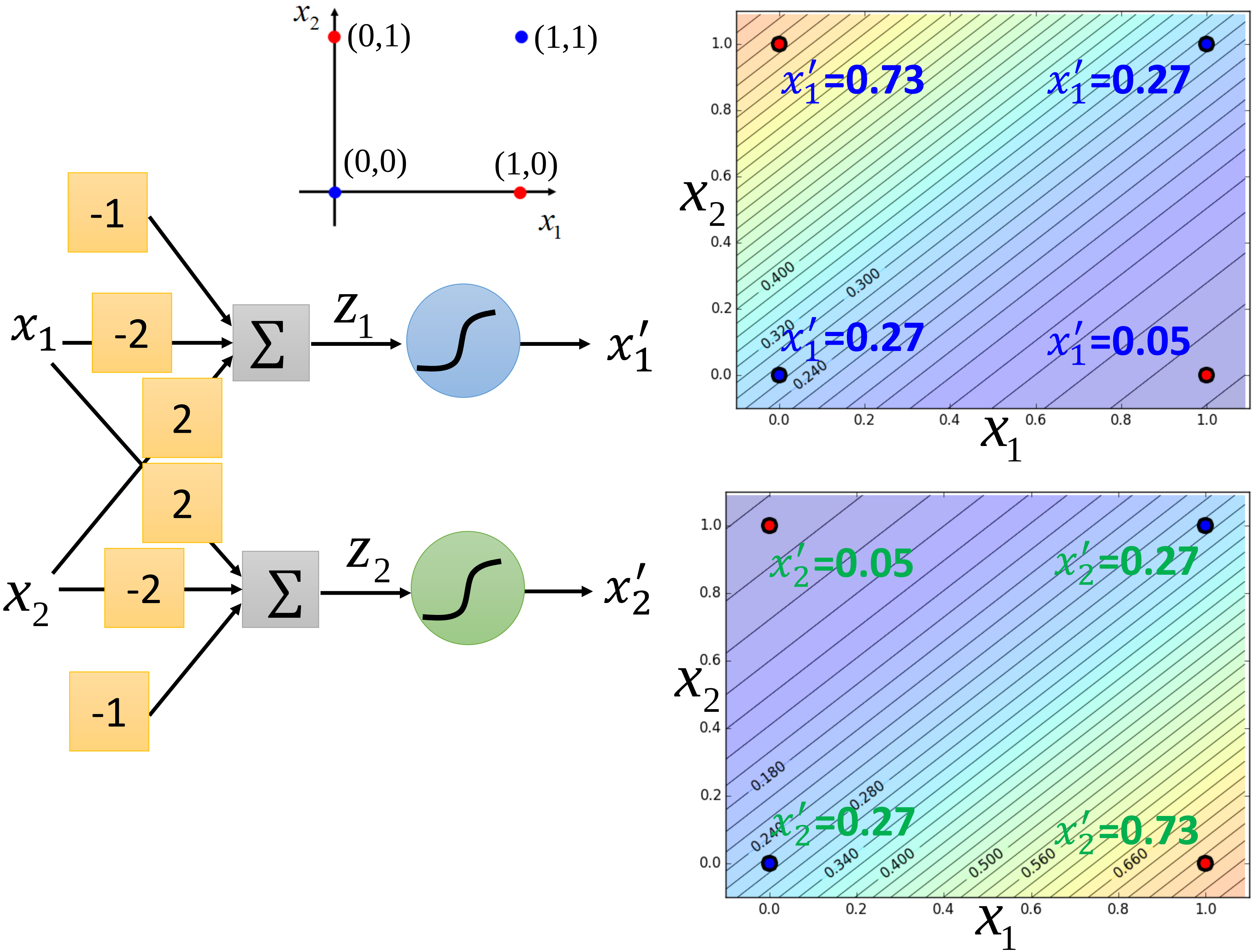
Not always easy ….. domain knowledge can be helpful

Limitation of Logistic Regression

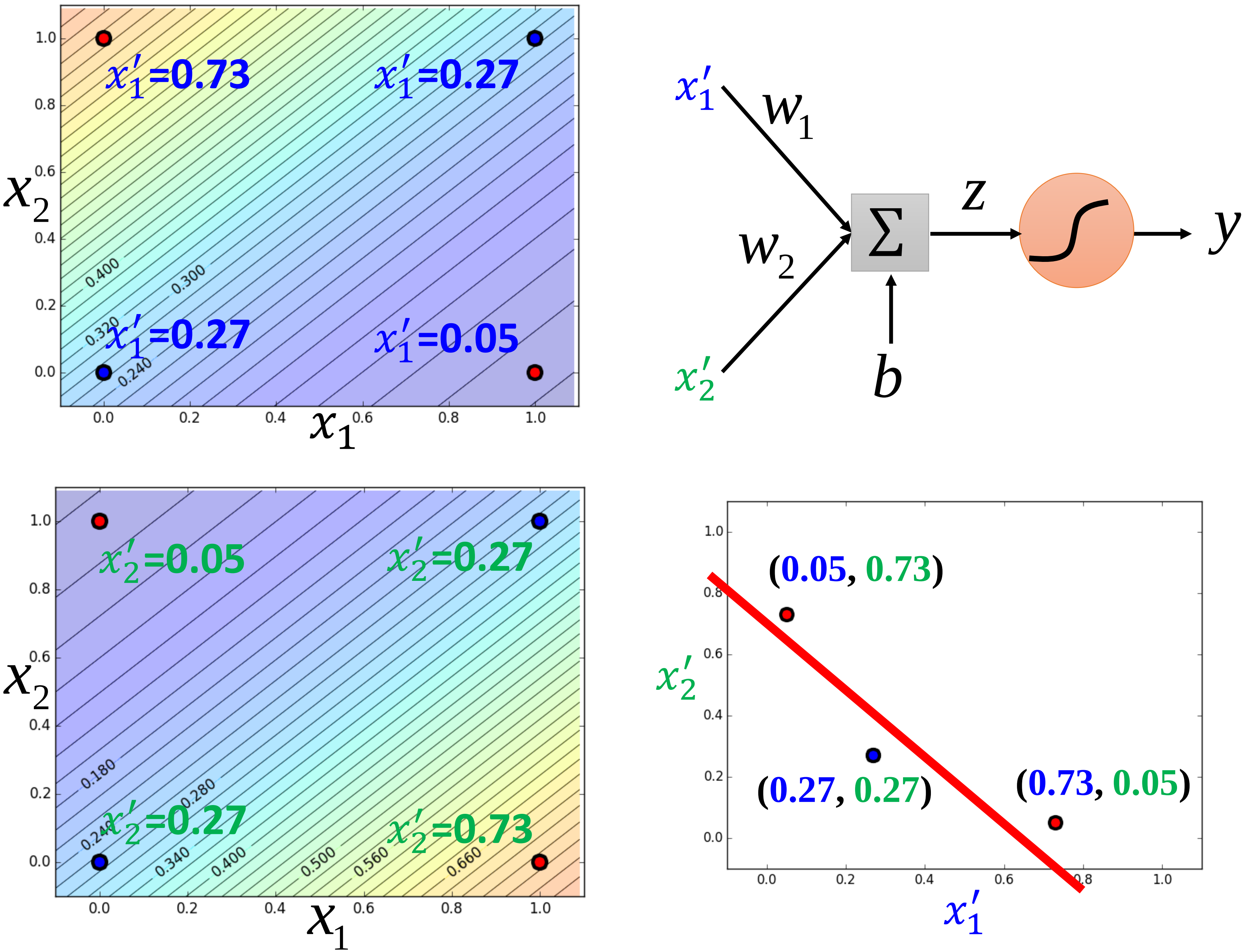
- Cascading logistic regression models



(ignore bias in this figure)



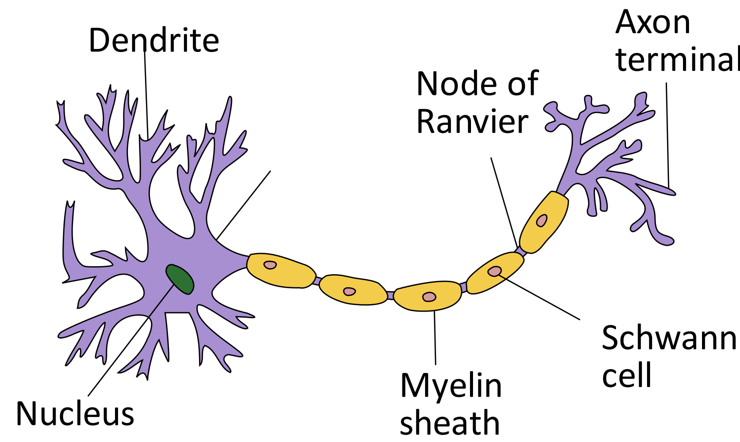
(0,1)
(1,1)
(0,0)
(1,0)
x_1
x_2
-1
-2
2
2
-2
-1
z_1
z_2
x'_1
x'_2
x'_1=0.73
x'_1=0.27
x'_1=0.27
x'_1=0.05
x'_2=0.05
x'_2=0.27
x'_2=0.27
x'_2=0.73

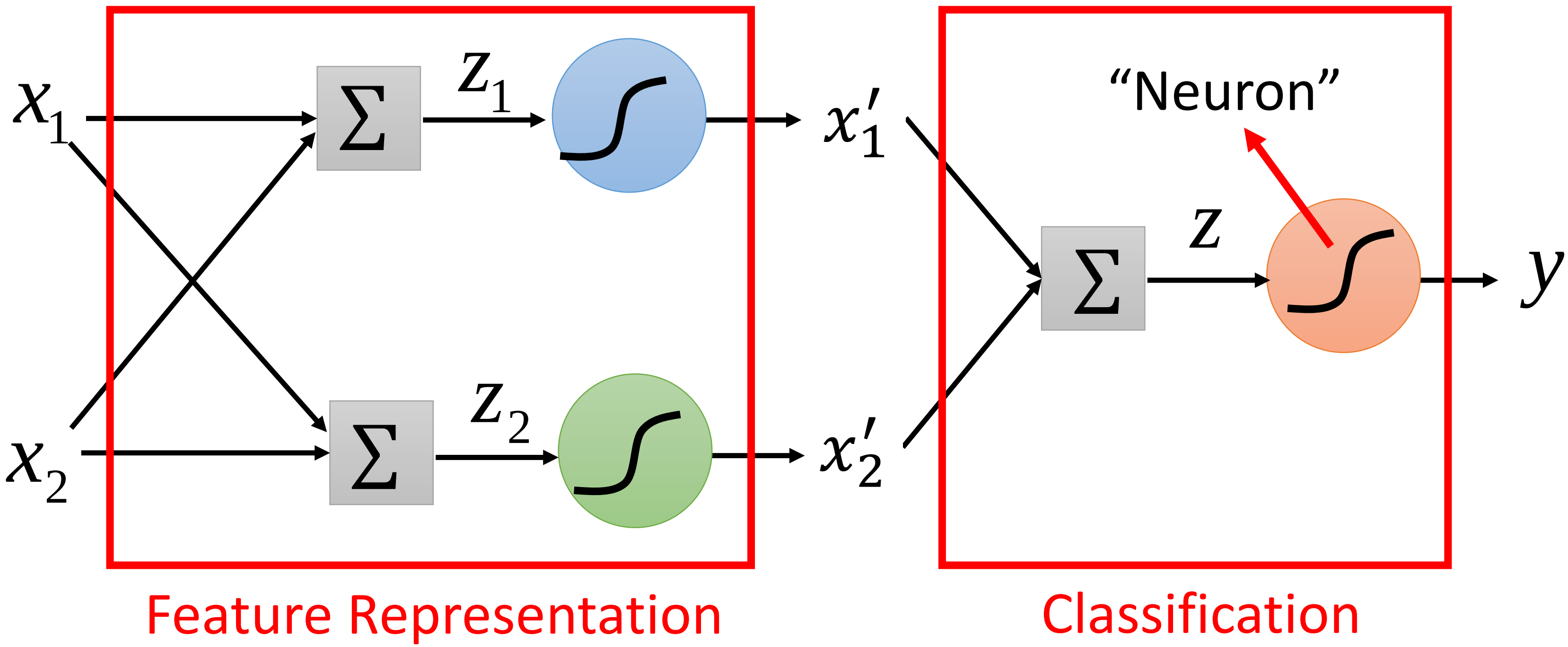


x_1'=0.73
x_1'=0.27
x_1'=0.27
x_1'=0.05
x_2
x_1
x_1'
x_2'
w_1
w_2
Σ
z
y
b
x_2'=0.05
x_2'=0.27
x_2'=0.27
x_2'=0.73
x_2
x_1
(0.05, 0.73)
(0.27, 0.27)
(0.73, 0.05)
x_2'
x_1'

Deep Learning!

All the parameters of the logistic regressions are jointly learned.





Neural Network

Summary and Next Lecture

(Check WebOodi for time)

- Neural Networks
- Multilayer Neural Networks
- Backpropagation