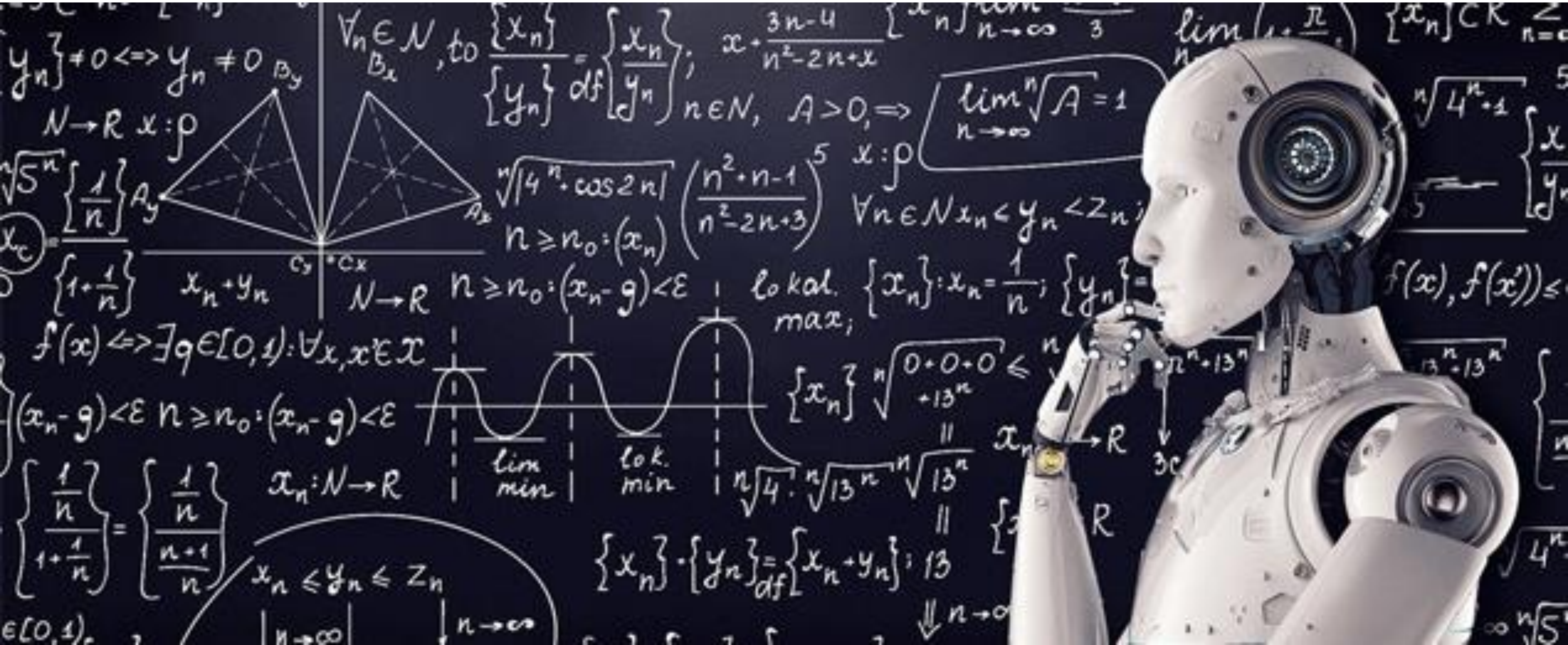
大数据分析技术

Chap. 12 Regression model

王怡洋 副教授

大连海事大学 信息科学技术学院

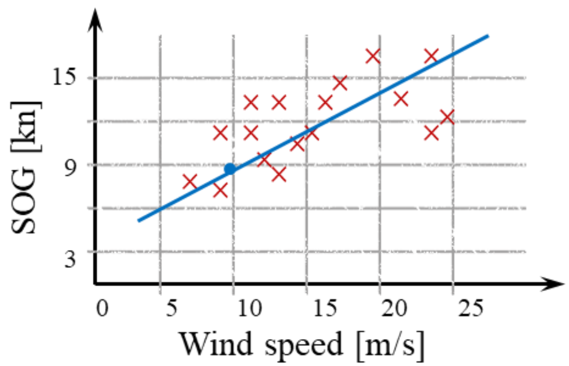


内容提纲

注：该ppt请不要传播，☺。

12.1 Univariate Linear regression revisit

- Wind speed → SOG

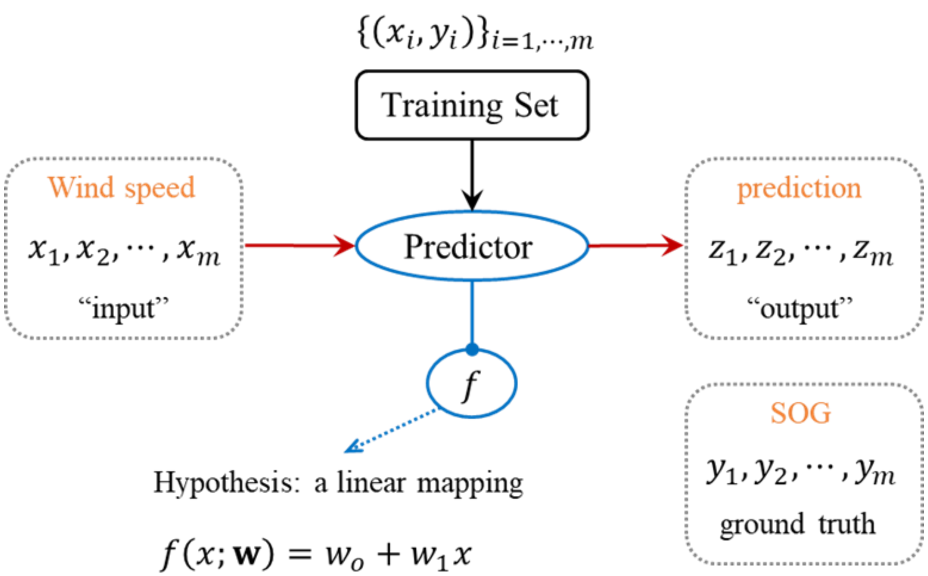


$\boldsymbol{m}$: the number of training examples

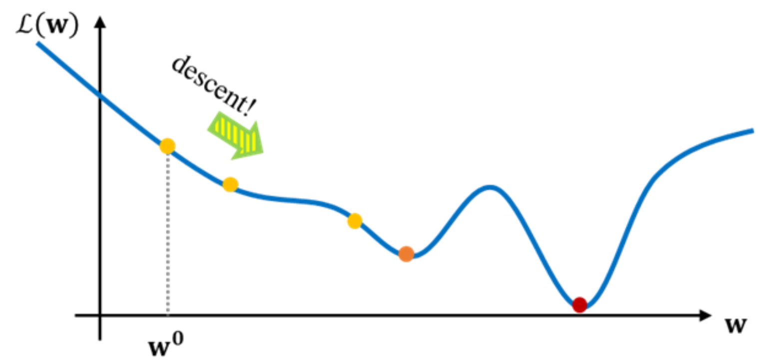
$\mathbf{x}$: "input" variable/feature (wind speed) $\mathbf{x} = (x_1, x_2, \cdots, x_m)^\top$

$\mathbf{y}$: ground truth/label (SOG) $\mathbf{y} = (y_1, y_2, \cdots, y_m)^\top$

$\mathbf{z}$: "output" variable/target (prediction) $\mathbf{z} = (z_1, z_2, \cdots, z_m)^\top$



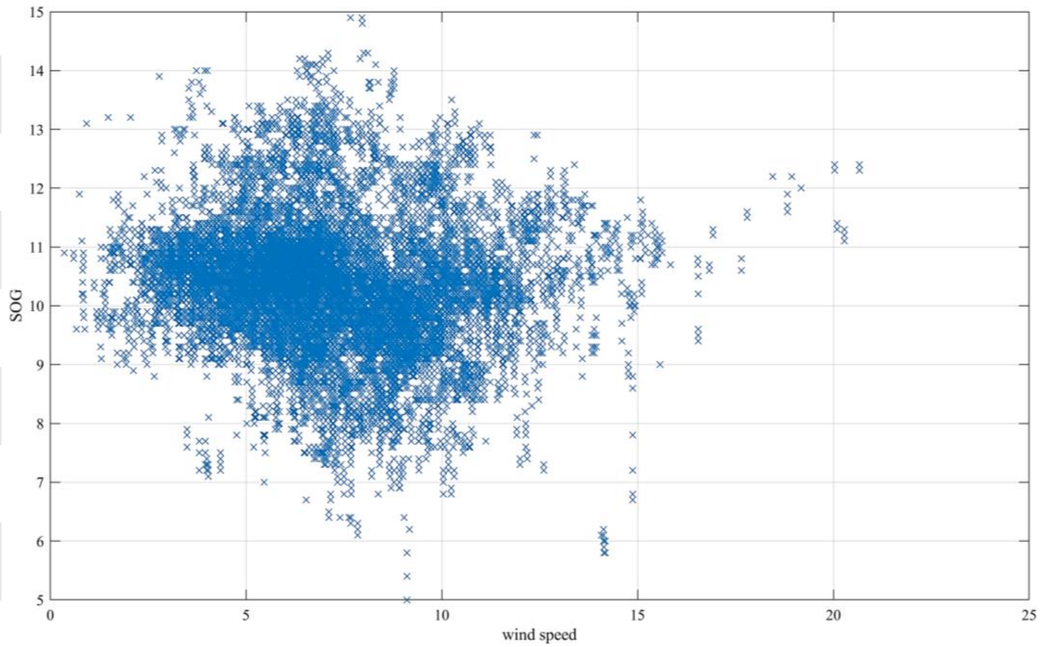
$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \mathcal{L}(\mathbf{w}) = \arg\min_{\mathbf{w}} \sum_{i=1}^{m} (f(x_i; \mathbf{w}) - y_i)^2$$



Gradient descent Alg.: $\mathbf{w}^{t+1} = \mathbf{w}^t - \alpha \nabla \mathcal{L}(\mathbf{w}^t)$

12.2 Multivariate linear regression

NAME	NOTES
SOG (kn)	Speed over ground，对地速度
DRAUGHT (m)	吃水
COG (°)	Course over ground，对地航向
HDG (°)	Heading，船艏向
CURRENT (°)	流向
TRUE WIND (°)	风向
WAVE (°)	风浪方向
SWELL (°)	涌浪方向
CURRENT (kn)	流速
TRUE WIND (m/s)	风速
WAVE (m)	浪高
SWELL (m)	涌浪高度
GUST (m/s)	阵风风速
SEAS (m)	耦合浪高



12.2.1 Notations

m: the number of training examples

Pre.

$\mathbf{x}$: "input" variable/feature (wind speed) $\mathbf{x} = (x_1, x_2, \cdots, x_m)^\intercal$

$\mathbf{y}$: ground truth/label (SOG) $\mathbf{y} = (y_1, y_2, \cdots, y_m)^\intercal$

$\mathbf{z}$: "output" variable/target (prediction) $\mathbf{z} = (z_1, z_2, \cdots, z_m)^\intercal$

New

m: the number of training examples

n: the number of features

$\mathbf{x}$: "input" variables/features (attribute features)

$$\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m)^\intercal \in \mathbb{R}^{\mathrm{m}\times\mathrm{n}}$$ (all training data)

$$\mathbf{x}_{i,\cdot} = \left(x_{i,1}, x_{i,2}, \cdots, x_{i,n}\right)^\intercal \in \mathbb{R}^{\mathrm{n}}$$ (the i[th] sample with n features)

$\mathbf{y}$: ground truth/label (SOG)

$$\mathbf{y} = (y_1, y_2, \cdots, y_m)^\intercal \in \mathbb{R}^{\mathrm{m}}$$

$\mathbf{z}$: "output" variable/target (prediction)

$$\mathbf{z} = (z_1, z_2, \cdots, z_m)^\intercal \in \mathbb{R}^{\mathrm{m}}$$

$$\begin{array}{cc} & \begin{array}{c}\text{feature}\\ \mathbf{x}_{\cdot,j}\end{array} \\ \begin{array}{c}\mathbf{x}_{i,\cdot}\\ \text{data}\\ \text{sample}\end{array} & \begin{bmatrix} x_{1,1} & \cdots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{m,1} & \cdots & x_{m,n} \end{bmatrix} \end{array}$$

12.2.2 Function hypothesis

Pre. $x_1, x_2, \cdots, x_m$

New $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m$

Wind speed

"input"

Training Set

Predictor

f

prediction

$z_1, z_2, \cdots, z_m$

"output"

SOG

$y_1, y_2, \cdots, y_m$

ground truth

Hypothesis: a linear mapping

Pre. $f(x) = w_0 + w_1 x \xrightarrow{\mathbf{w} = (w_0, w_1)^\top} f(x; \mathbf{w}) = \mathbf{w}^\top \begin{bmatrix} 1 \\ x \end{bmatrix}$

New $f(\mathbf{x}) = w_0 + w_1 x_1 + \cdots + w_n x_n \xrightarrow{\mathbf{w} = (w_0, w_1, \cdots, w_n)^\top} f(\mathbf{x}; \mathbf{w}) = \mathbf{w}^\top \begin{bmatrix} 1 \\ \mathbf{x} \end{bmatrix}$

12.2.3 Loss function

Pre.

$$\min_{w_o, w_1} (f(x_i; \mathbf{w}) - y_i)^2$$

$$\min_{\mathbf{w}} \sum_{i=1}^{m} (f(x_i; \mathbf{w}) - y_i)^2$$

$$\|$$

$$\min_{w_o, w_1} \frac{1}{2m} \sum_{i=1}^{m} (w_o + w_1 x_i - y_i)^2$$

Let's define: $\mathcal{L}(w_o, w_1) = \frac{1}{2m} \sum_{i=1}^{m} (w_o + w_1 x_i - y_i)^2$

$$\min_{w_o, w_1} \mathcal{L}(w_o, w_1)$$

loss/cost function (squared error function)

- Different hypothesis on f, different $\mathcal{L}$
- Different w_o and w_1, different value of $\mathcal{L}$.
- The best w_o and w_1 are corresponding to the lowest value of $\mathcal{L}$, which are defined as $\mathbf{w}^*$, i.e.

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \mathcal{L}(\mathbf{w})$$

- $\mathcal{L}(w_0^*, w_1^*)$ is the best predictor under the linear hypothesis

New

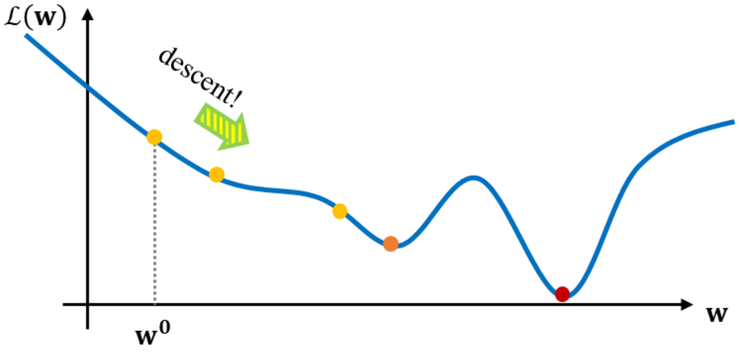
$$\min_{w_0, w_1, \cdots, w_n} \frac{1}{2m} \sum_{i=1}^{m} \left(w_0 + w_1 x_{i,1} + \cdots + w_n x_{i,n} - y_i\right)^2$$

Let's define:

$$\mathcal{L}(w_0, w_1, \cdots, w_n) = \frac{1}{2m} \sum_{i=1}^{m} \left(w_0 + w_1 x_{i,1} + \cdots + w_n x_{i,n} - y_i\right)^2$$

$$\mathbf{w}^* = \arg\min_{\mathbf{w} \in \mathbb{R}^{n+1}} \frac{1}{2m} \sum_{i=1}^{m} (f(\mathbf{x}_i; \mathbf{w}) - y_i)^2$$

12.2.4 Gradient descent



Gradient descent Alg.: $\mathbf{w}^{t+1} = \mathbf{w}^t - \alpha \nabla \mathcal{L}(\mathbf{w}^t)$

Learning rate

Pre.

$$\begin{bmatrix} w_0^{t+1} \\ w_1^{t+1} \end{bmatrix} = \begin{bmatrix} w_0^t \\ w_1^t \end{bmatrix} - \alpha \begin{bmatrix} \frac{1}{m}\sum_{i=1}^{m}(w_0^t + w_1^t x_i - y_i) \\ \frac{1}{m}\sum_{i=1}^{m}(w_0^t + w_1^t x_i - y_i)x_i \end{bmatrix} = \begin{bmatrix} w_0^t \\ w_1^t \end{bmatrix} - \frac{\alpha}{m}\sum_{i=1}^{m}\left((\mathbf{w}^t)^\intercal \begin{bmatrix} 1 \\ x_i \end{bmatrix} - y_i\right)\begin{bmatrix} 1 \\ x_i \end{bmatrix}$$

New

$$\begin{bmatrix} w_0^{t+1} \\ w_1^{t+1} \\ \vdots \\ w_n^{t+1} \end{bmatrix} = \begin{bmatrix} w_0^t \\ w_1^t \\ \vdots \\ w_n^t \end{bmatrix} - \alpha \begin{bmatrix} \frac{1}{m}\sum_{i=1}^{m}(w_0^t + w_1^t x_{i,1} + \cdots + w_n^t x_{i,n} - y_i) \\ \frac{1}{m}\sum_{i=1}^{m}(w_0^t + w_1^t x_{i,1} + \cdots + w_n^t x_{i,n} - y_i)\, x_{i,1} \\ \vdots \\ \frac{1}{m}\sum_{i=1}^{m}(w_0^t + w_1^t x_{i,1} + \cdots + w_n^t x_{i,n} - y_i) x_{i,n} \end{bmatrix} = \begin{bmatrix} w_0^t \\ w_1^t \\ \vdots \\ w_n^t \end{bmatrix} - \frac{\alpha}{m}\sum_{i=1}^{m}\left((\mathbf{w}^t)^\intercal \begin{bmatrix} 1 \\ \mathbf{x}_i \end{bmatrix} - y_i\right)\begin{bmatrix} 1 \\ \mathbf{x}_i \end{bmatrix}$$

12.2.5 A unified framework

$\boldsymbol{m}$: the number of training examples

$\boldsymbol{n}$: the number of features

$\mathbf{x}$: "input" variables/features (attribute features)

$$\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m)^\top \in \mathbb{R}^{m\times n}$$

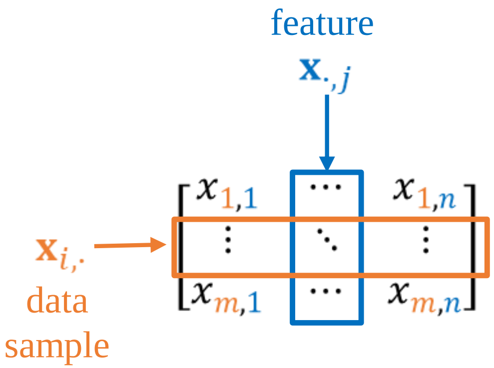
$$\mathbf{x}_{i,\cdot} = \left(x_{i,1}, x_{i,2}, \cdots, x_{i,n}\right)^\top \in \mathbb{R}^{n}$$

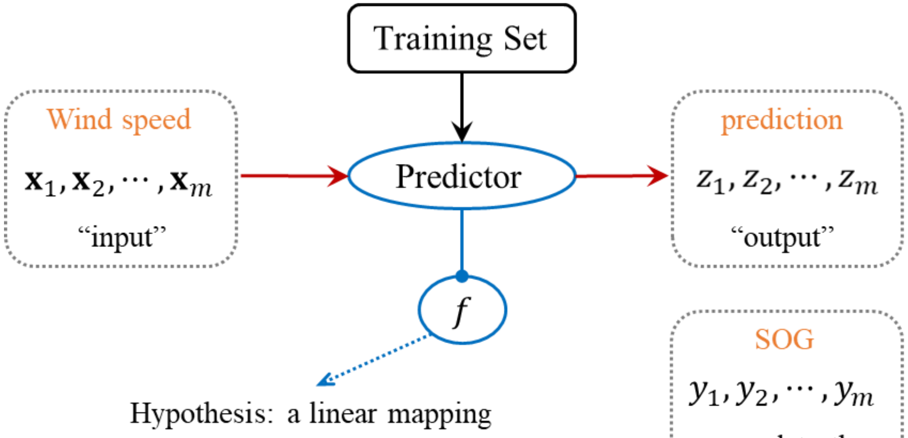
$\mathbf{y}$: ground truth/label (SOG)

$$\mathbf{y} = (y_1, y_2, \cdots, y_m)^\top \in \mathbb{R}^{m}$$

$\mathbf{z}$: "output" variable/target (prediction)

$$\mathbf{z} = (z_1, z_2, \cdots, z_m)^\top \in \mathbb{R}^{m}$$





$$f(\mathbf{x};\mathbf{w}) = \mathbf{w}^\top \begin{bmatrix} 1 \\ \mathbf{x} \end{bmatrix}$$

$$\mathbf{w}^* = \arg\min_{\mathbf{w}\in\mathbb{R}^{n+1}} \frac{1}{2m}\sum_{i=1}^{m}\left(f(\mathbf{x}_i;\mathbf{w}) - y_i\right)^2$$

$$\mathbf{w}^{t+1} = \mathbf{w}^{t} - \alpha\nabla\mathcal{L}(\mathbf{w}^{t})$$

$$\begin{bmatrix} w_0^{t+1} \\ w_1^{t+1} \\ \vdots \\ w_n^{t+1} \end{bmatrix} = \begin{bmatrix} w_0^{t} \\ w_1^{t} \\ \vdots \\ w_n^{t} \end{bmatrix} - \frac{\alpha}{m}\sum_{i=1}^{m}\left((\mathbf{w}^t)^\top \begin{bmatrix} 1 \\ \mathbf{x}_i \end{bmatrix} - y_i\right)\begin{bmatrix} 1 \\ \mathbf{x}_i \end{bmatrix}$$

12.2.6 More discussions – part I

$$\mathbf{w}^* = \arg\min_{\mathbf{w}\in\mathbb{R}^{n+1}} \frac{1}{2m}\sum_{i=1}^{m}\left(f(\mathbf{x}_i;\mathbf{w}) - y_i\right)^2 \qquad f(\mathbf{x};\mathbf{w}) = \mathbf{w}^\top\begin{bmatrix}1\\ \mathbf{x}\end{bmatrix}$$

- See the computation in the brackets:

$$\mathbf{w}^\top\begin{bmatrix}1\\ \mathbf{x}_i\end{bmatrix} - y_i = [1 \quad \mathbf{x}_i^\top]\mathbf{w} - y_i$$

- Thus we have:

$$\sum_{i=1}^{m}\left(\mathbf{w}^\top\begin{bmatrix}1\\ \mathbf{x}_i\end{bmatrix} - y_i\right)^2 = \left\|\begin{bmatrix}1 & \mathbf{x}_1^\top\\ 1 & \mathbf{x}_2^\top\\ & \vdots \\ 1 & \mathbf{x}_m^\top\end{bmatrix}\mathbf{w} - \begin{bmatrix}y_1\\ y_2\\ \vdots\\ y_m\end{bmatrix}\right\|^2$$

- Let's define:

$$\mathbb{X} = \begin{bmatrix}1 & \mathbf{x}_1^\top\\ 1 & \mathbf{x}_2^\top\\ & \vdots \\ 1 & \mathbf{x}_m^\top\end{bmatrix} \in \mathbb{R}^{m\times(n+1)}$$

- Then with $\mathbf{y} = (y_1, y_2, \cdots, y_m)^\top \in \mathbb{R}^{\mathrm{m}}$, we have:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}\in\mathbb{R}^{n+1}} \frac{1}{2m}\|\mathbb{X}\mathbf{w} - \mathbf{y}\|^2$$
$$= \arg\min_{\mathbf{w}\in\mathbb{R}^{n+1}} \frac{1}{2m}(\mathbb{X}\mathbf{w} - \mathbf{y})^\top(\mathbb{X}\mathbf{w} - \mathbf{y})$$

- Thus the optimal $\mathbf{w}^*$ satisfies:

$$\mathbb{X}^\top(\mathbb{X}\mathbf{w}^* - \mathbf{y}) = 0$$

- **Ref: details of derivation**

$$\partial\left((\mathbf{y} - \mathbb{X}\mathbf{w})^\top(\mathbf{y} - \mathbb{X}\mathbf{w})\right)$$
$$= \partial(\cancel{\mathbf{y}^\top\mathbf{y}} - \mathbf{y}^\top\mathbb{X}\mathbf{w} - \mathbf{w}^\top\mathbb{X}^\top\mathbf{y} + \mathbf{w}^\top\mathbb{X}^\top\mathbb{X}\mathbf{w})$$

$$(\text{Since } \mathbf{y}^\top\mathbb{X}\mathbf{w} = (\mathbb{X}\mathbf{w})^\top\mathbf{y} = \mathbf{w}^\top\mathbb{X}^\top\mathbf{y}, \text{ then})$$
$$= \partial(-2\mathbf{w}^\top\mathbb{X}^\top\mathbf{y} + \mathbf{w}^\top\mathbb{X}^\top\mathbb{X}\mathbf{w})$$

$$(\text{Since } \frac{\partial\mathbf{x}^\top\mathbf{a}}{\partial\mathbf{x}} = \frac{\partial\mathbf{a}^\top\mathbf{x}}{\partial\mathbf{x}} = \mathbf{a}, \text{ then})$$
$$= -2\mathbb{X}^\top\mathbf{y} + \mathbb{X}^\top\mathbb{X}\mathbf{w} + \mathbf{w}^\top\mathbb{X}^\top\mathbb{X}$$
$$= -2\mathbb{X}^\top\mathbf{y} + 2\mathbb{X}^\top\mathbb{X}\mathbf{w}$$

- Thus we need to solve the following linear equation:

$$\mathbb{X}^\top\mathbb{X}\mathbf{w}^* = \mathbb{X}^\top\mathbf{y}$$

- If $\mathbb{X}^\top\mathbb{X}$ is full rank, then:

$$\mathbf{w}^* = (\mathbb{X}^\top\mathbb{X})^{-1}\mathbb{X}^\top\mathbf{y}$$

12.2.7 More discussions – part II

$$\mathbf{w}^{t+1} = \mathbf{w}^{t} - \alpha \nabla \mathcal{L}(\mathbf{w}^{t})$$

$$\begin{bmatrix} w_0^{t+1} \\ w_1^{t+1} \\ \vdots \\ w_n^{t+1} \end{bmatrix} = \begin{bmatrix} w_0^{t} \\ w_1^{t} \\ \vdots \\ w_n^{t} \end{bmatrix} - \frac{\alpha}{m}\sum_{i=1}^{m}\left((\mathbf{w}^t)^\intercal \begin{bmatrix} 1 \\ \mathbf{x}_i \end{bmatrix} - y_i\right)\begin{bmatrix} 1 \\ \mathbf{x}_i \end{bmatrix} = \begin{bmatrix} w_0^{t} \\ w_1^{t} \\ \vdots \\ w_n^{t} \end{bmatrix} - \alpha \begin{bmatrix} \frac{1}{m}\sum_{i=1}^{m}(w_0^t + w_1^t x_{i,1} + \cdots + w_n^t x_{i,n} - y_i) \\ \frac{1}{m}\sum_{i=1}^{m}(w_0^t + w_1^t x_{i,1} + \cdots + w_n^t x_{i,n} - y_i)\, x_{i,1} \\ \vdots \\ \frac{1}{m}\sum_{i=1}^{m}(w_0^t + w_1^t x_{i,1} + \cdots + w_n^t x_{i,n} - y_i) x_{i,n} \end{bmatrix}$$

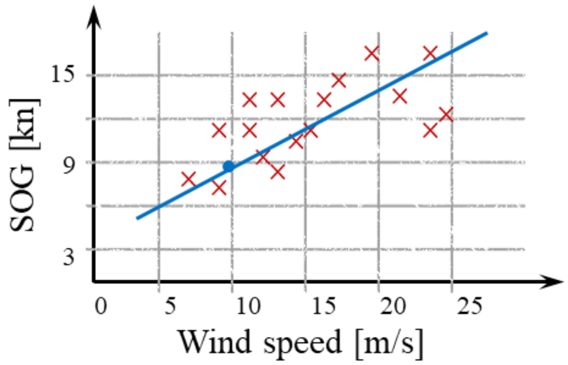
$$\begin{aligned} w_0^{t+1} &= w_0^t - \alpha\frac{1}{m}\sum_{i=1}^{m}(w_0^t + w_1^t x_{i,1} + \cdots + w_n^t x_{i,n} - y_i) \\ w_1^{t+1} &= w_1^t - \alpha\frac{1}{m}\sum_{i=1}^{m}(w_0^t + w_1^t x_{i,1} + \cdots + w_n^t x_{i,n} - y_i) x_{i,1} \\ &\vdots \\ w_n^{t+1} &= w_n^t - \alpha\frac{1}{m}\sum_{i=1}^{m}(w_0^t + w_1^t x_{i,1} + \cdots + w_n^t x_{i,n} - y_i) x_{i,n} \end{aligned}$$

- **Gradient descent algorithm**

- **Coordinate descent algorithm** (faster ☺)

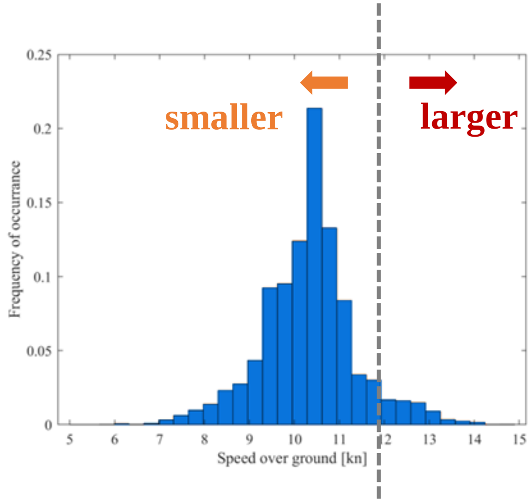
$$\begin{aligned} w_0^{t+1} &= w_0^t - \alpha\frac{1}{m}\sum_{i=1}^{m}(w_0^t + w_1^t x_{i,1} + \cdots + w_n^t x_{i,n} - y_i) \\ w_1^{t+1} &= w_1^t - \alpha\frac{1}{m}\sum_{i=1}^{m}(w_0^{t+1} + w_1^t x_{i,1} + \cdots + w_n^t x_{i,n} - y_i) x_{i,1} \\ &\vdots \\ w_n^{t+1} &= w_n^t - \alpha\frac{1}{m}\sum_{i=1}^{m}(w_0^{t+1} + w_1^{t+1} x_{i,1} + \cdots + w_n^t x_{i,n} - y_i) x_{i,n} \end{aligned}$$

12.3 Logistic regression



- Continuous variable
- $x \uparrow, y \uparrow$

- Suppose we don't care about the exact prediction of SOG, we just want to know whether SOG is larger than the service speed, or not.



- Discrete variable
- Contain two class, one is "larger" than the service speed, the other one is "smaller" than the service speed
- Can we use linear regression to solve this task? ☹

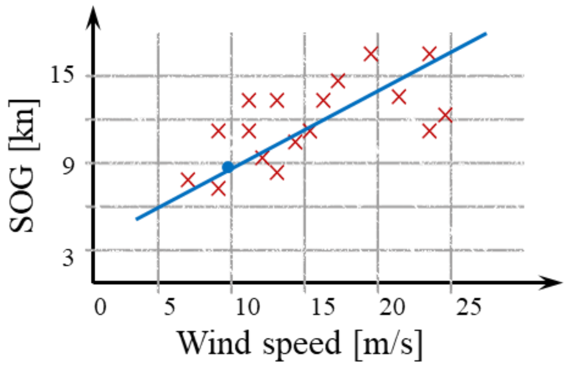
y: ground truth/label (not SOG, but {0, 1})

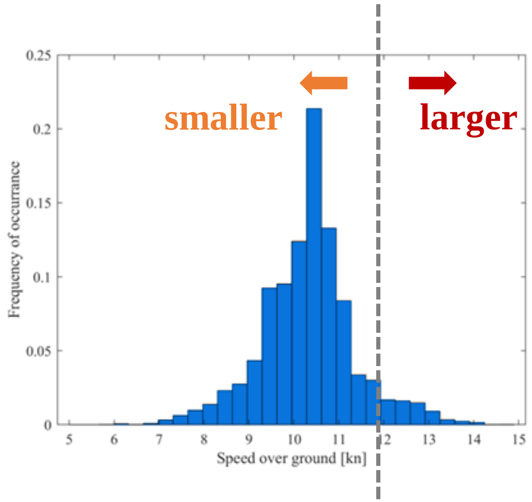
$$\mathbf{y} = (y_1, y_2, \cdots, y_m)^\intercal \in \{0, 1\}^m$$

z: "output" variable/target (prediction: closer to 1/closer to 0)

Reference: Christopher M. Bishop. *Pattern recognition and machine learning*, Springer, 2006. (P186)

12.3.1 Ideal alternatives





- Linear regression:

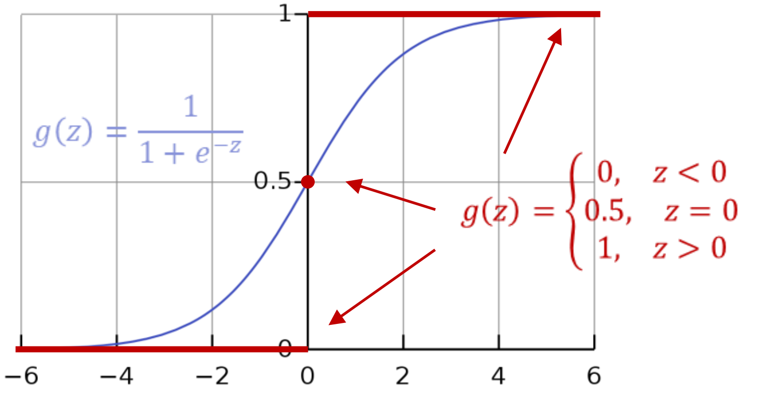
$$z := f(\mathbf{x}; \mathbf{w}) = \mathbf{w}^\intercal \begin{bmatrix} 1 \\ \mathbf{x} \end{bmatrix} \qquad \text{Let } \hat{\mathbf{x}} := \begin{bmatrix} 1 \\ \mathbf{x} \end{bmatrix}$$

- Find an *unit-step function*, such that:

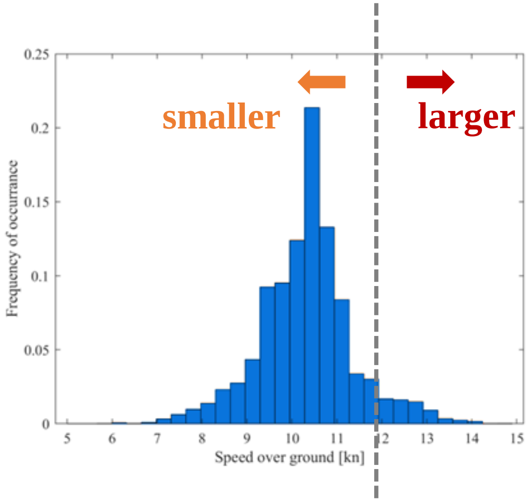
$$g(z) = \begin{cases} 0, & z < 0 \\ 0.5, & z = 0 \\ 1, & z > 0 \end{cases}$$

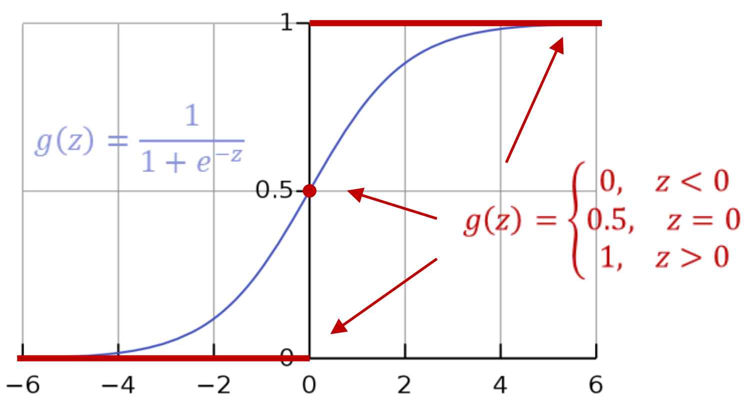
- Discontinuity ☹, we need to find a surrogate function
- Logistic/sigmoid function

$$u := g(z) = \frac{1}{1 + e^{-z}} \qquad \left(= \frac{1}{1 + e^{-\mathbf{w}^\intercal \hat{\mathbf{x}}}}\right)$$



12.3.2 Log-odds





- Logistic/sigmoid function

$$u := g(z) = \frac{1}{1+e^{-z}} \qquad \left(= \frac{1}{1+e^{-\mathbf{w}^\intercal \hat{\mathbf{x}}}}\right)$$

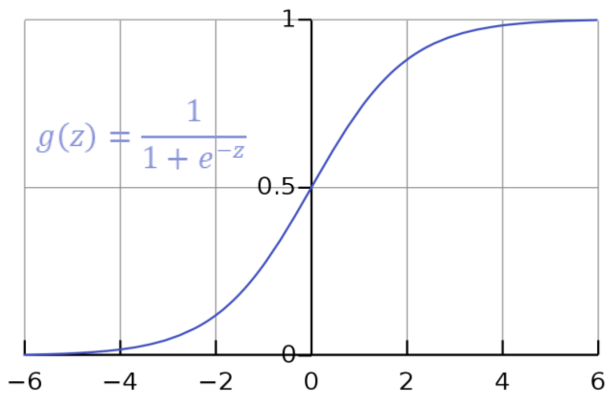
- Make a transformation

$$ln\frac{u}{1-u} = \mathbf{w}^\intercal \hat{\mathbf{x}}$$

- If we regard u as the probability of class “larger”, then $1-u$ can be seen as the probability of class “smaller”

- Thus, $\frac{u}{1-u}$ can be seen as the odds *(see wikipedia)*, thus the log-odds (the logarithm of the odds) for the label “larger" is a linear combination of input features

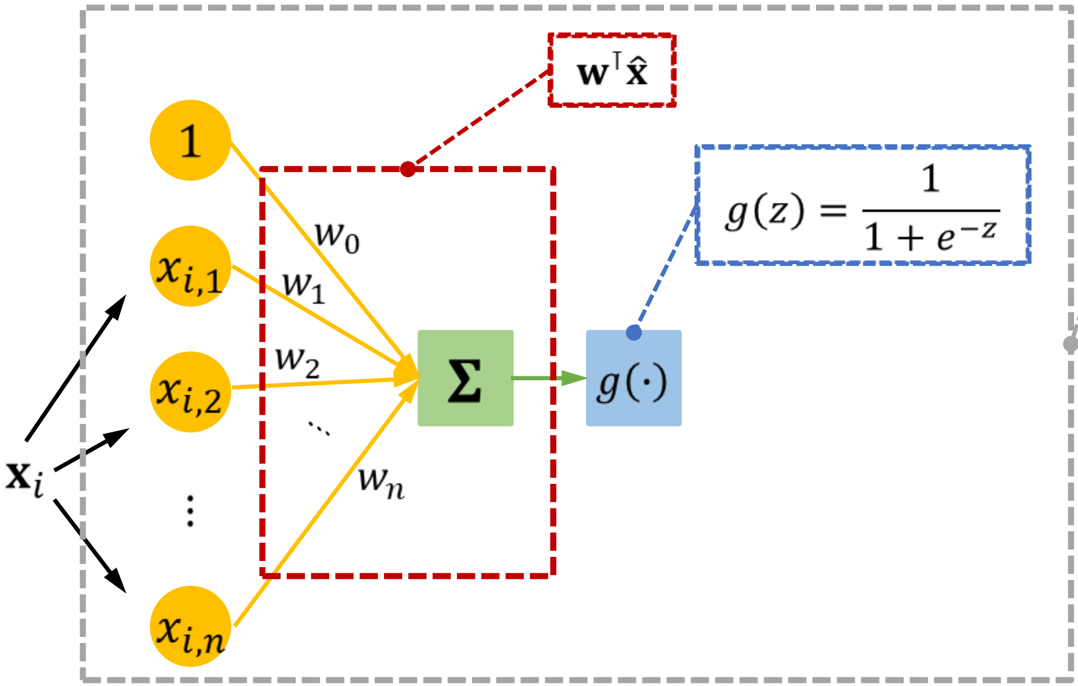
12.3.3 Graphics mode



- Logistic/sigmoid function

$$u := g(z) = \frac{1}{1+e^{-z}} \qquad \left(= \frac{1}{1+e^{-\mathbf{w}^{\top}\hat{\mathbf{x}}}}\right)$$

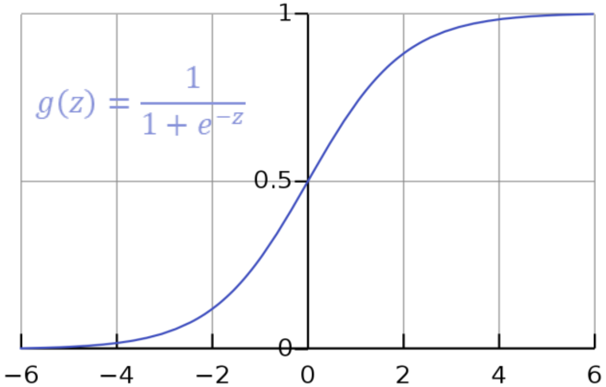
$$f(\mathbf{x}; w) = g\left(\mathbf{w}^{\top}\begin{bmatrix}1\\ \mathbf{x}\end{bmatrix}\right) = \frac{1}{1+e^{-\mathbf{w}^{\top}\begin{bmatrix}1\\ \mathbf{x}\end{bmatrix}}}$$



For reference:

- Loss function: cross-entropy
- Optimization algorithm

12.3.4 Decision boundary

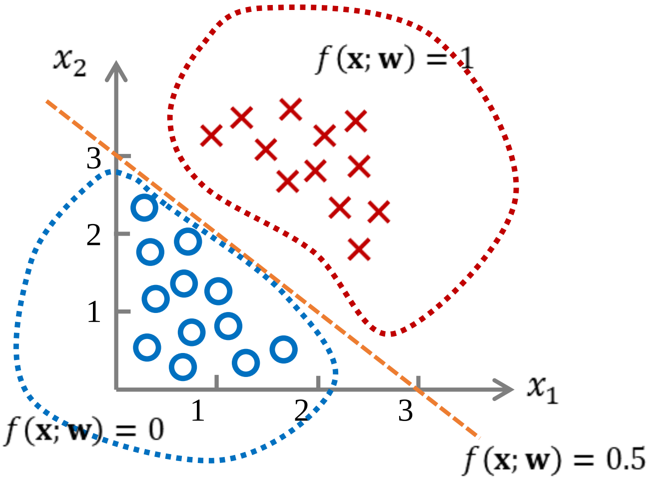


- Logistic/sigmoid function

$$\boxed{f(\mathbf{x}; w) = g\left(\mathbf{w}^\intercal \begin{bmatrix} 1 \\ \mathbf{x} \end{bmatrix}\right) = \frac{1}{1 + e^{-\mathbf{w}^\intercal \begin{bmatrix} 1 \\ \mathbf{x} \end{bmatrix}}}}$$

$$f(\mathbf{x}; \mathbf{w}) = 1 \text{ if } f(\mathbf{x}; \mathbf{w}) \geq 0.5\text{, or equally, } \mathbf{w}^\intercal \begin{bmatrix} 1 \\ \mathbf{x} \end{bmatrix} \geq 0$$

$$f(\mathbf{x}; \mathbf{w}) = 0 \text{ if } f(\mathbf{x}; \mathbf{w}) < 0.5\text{, or equally, } \mathbf{w}^\intercal \begin{bmatrix} 1 \\ \mathbf{x} \end{bmatrix} < 0$$



- Suppose that we have

$$f(\mathbf{x}; \mathbf{w}) = g(w_0 + w_1 x_1 + w_2 x_2)$$

and the parameters have already been optimized.

$$w_0 = -3;\ w_1 = 1;\ w_2 = 1$$

- From the above, we know:

$$f(\mathbf{x}; \mathbf{w}) = 1 \text{ if } -3 + x_1 + x_2 \geq 0$$
$$f(\mathbf{x}; \mathbf{w}) = 0 \text{ if } -3 + x_1 + x_2 < 0$$

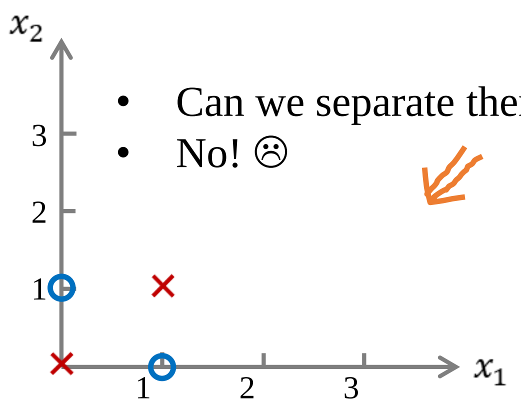
- So, its decision boundary is:

$$x_1 + x_2 = 3$$

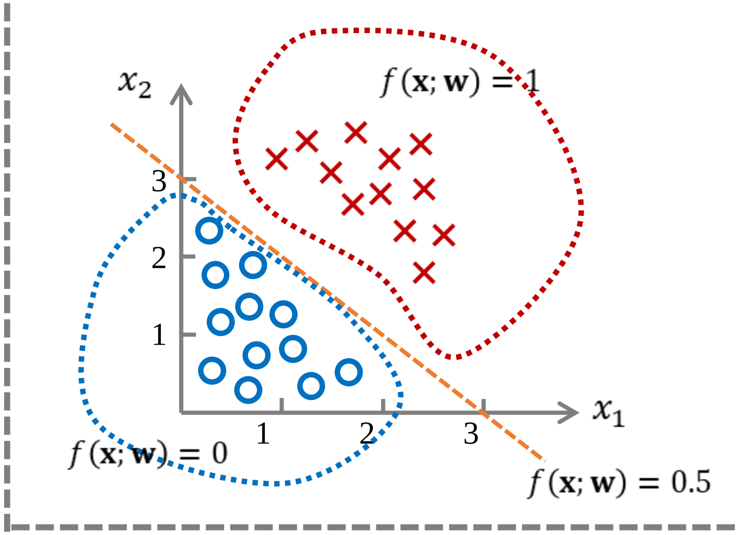
on which,

$$f(\mathbf{x}; \mathbf{w}) = 0.5$$

12.3.5 Limitation of Logistic regression



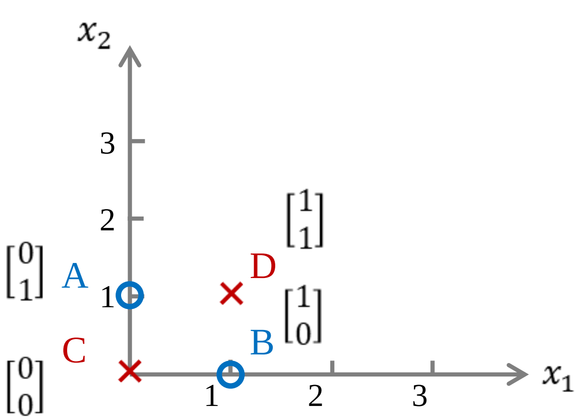
- Can we separate them by a decision boundary?
- No! ☹

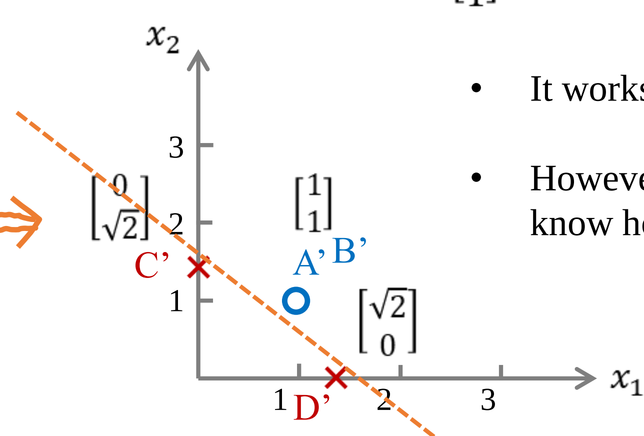


- If you insist on using Logistic regression, you may need feature transformation

our goal: $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \rightarrow \begin{bmatrix} x_1' \\ x_2' \end{bmatrix}$ we design: x_1' is the distance to $\begin{bmatrix} 0 \\ 0 \end{bmatrix}$

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

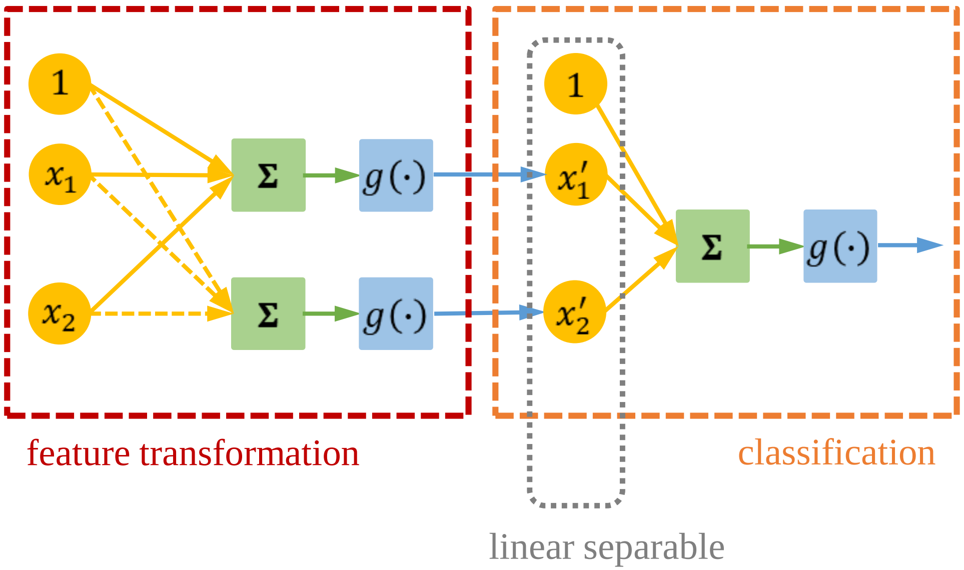


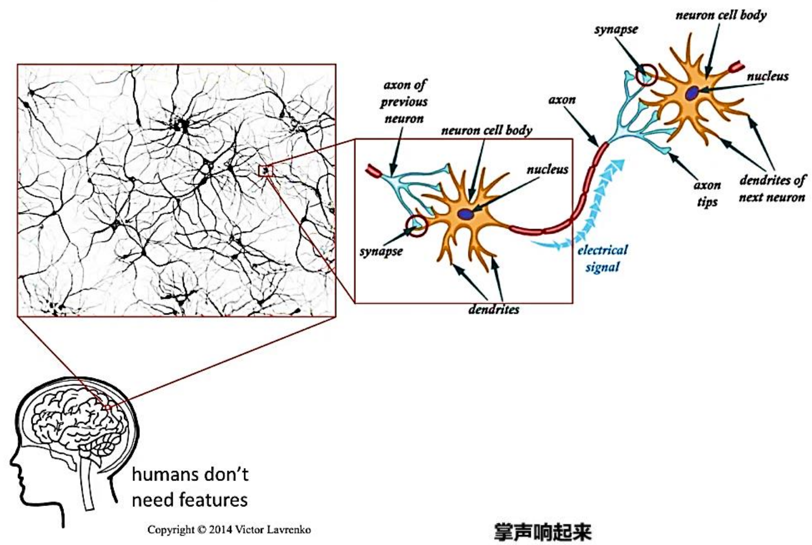


- It works! ☺
- However, the trouble is: most time, we don't know how to do feature transformation

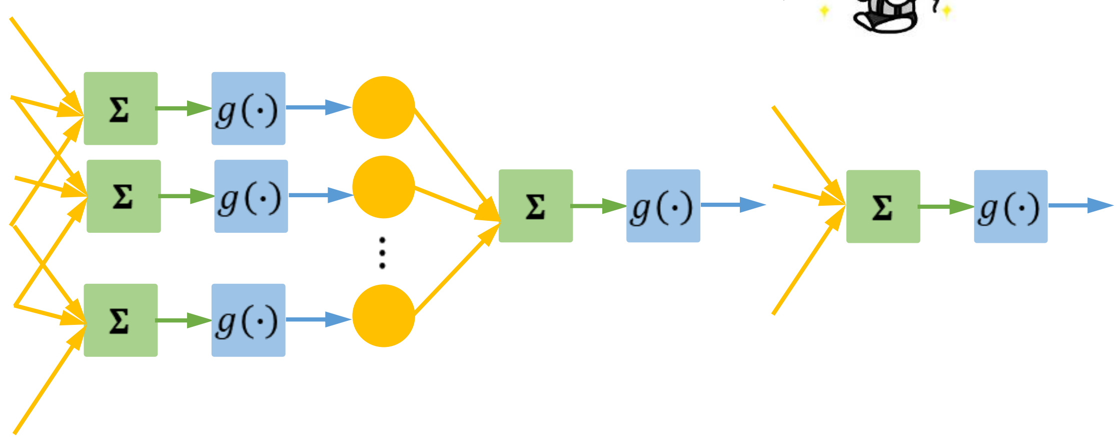
12.3.6 Cascading Logistic regression

- simulate feature transformation by Logistic regression



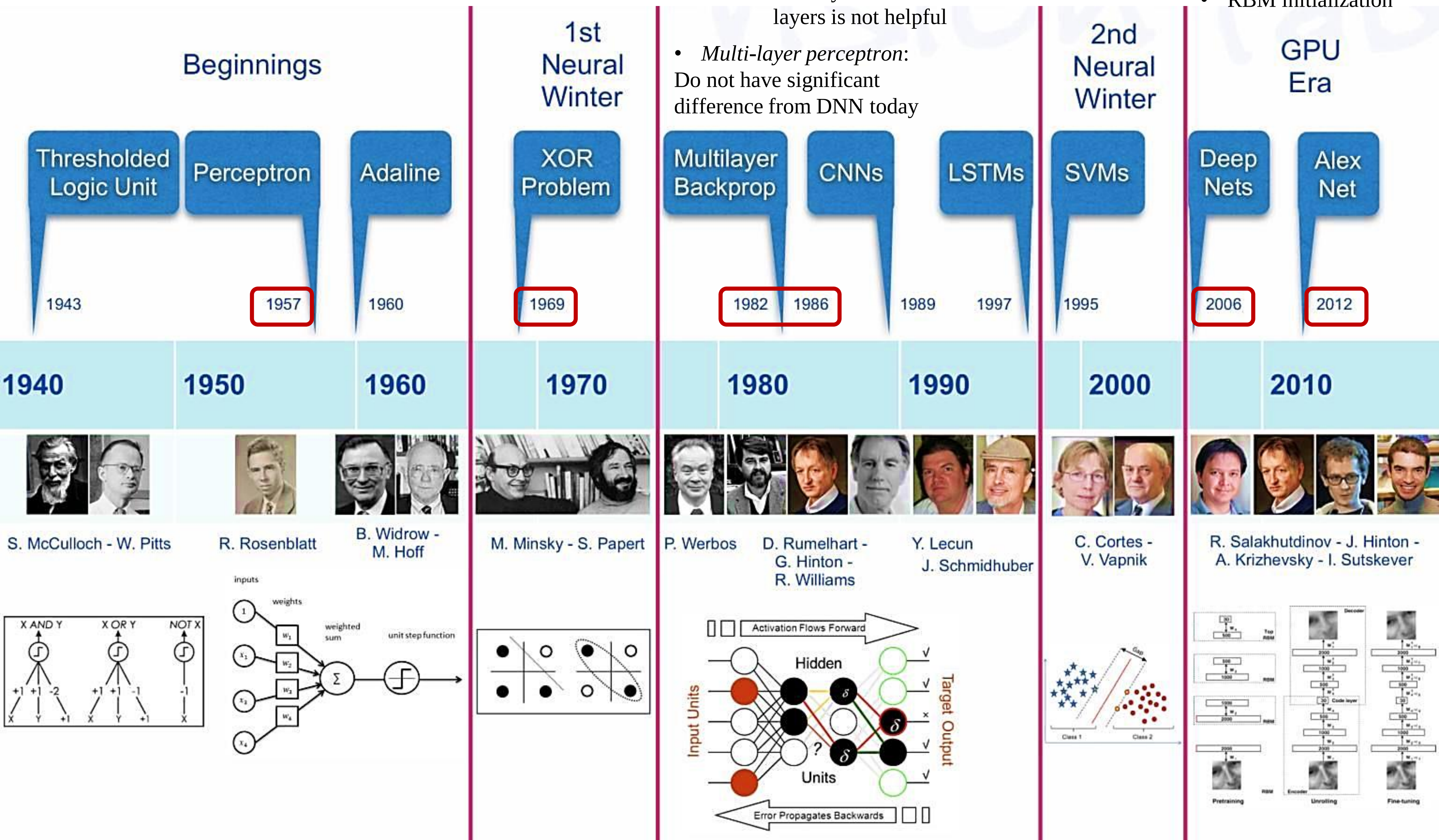


We can built a cascade of Logistic regression in any structure! ☺

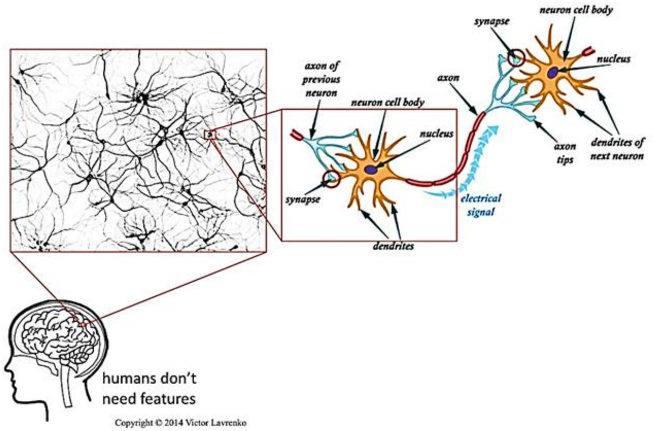


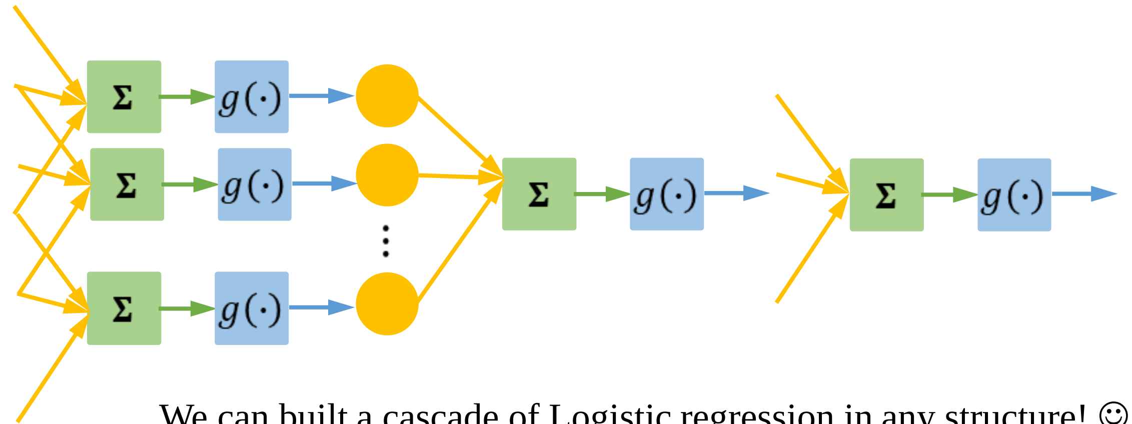
12.4 Deep learning

- 1 hidden layer is “good enough”
- Usually more than 3 hidden layers is not helpful
- RBM initialization
- *Multi-layer perceptron*: Do not have significant difference from DNN today



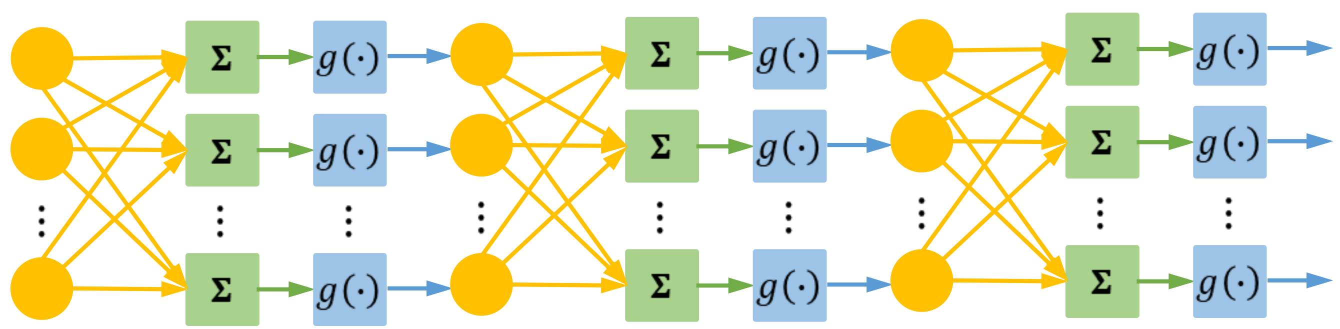
12.4.1 Define a set of function





We can built a cascade of Logistic regression in any structure! ☺

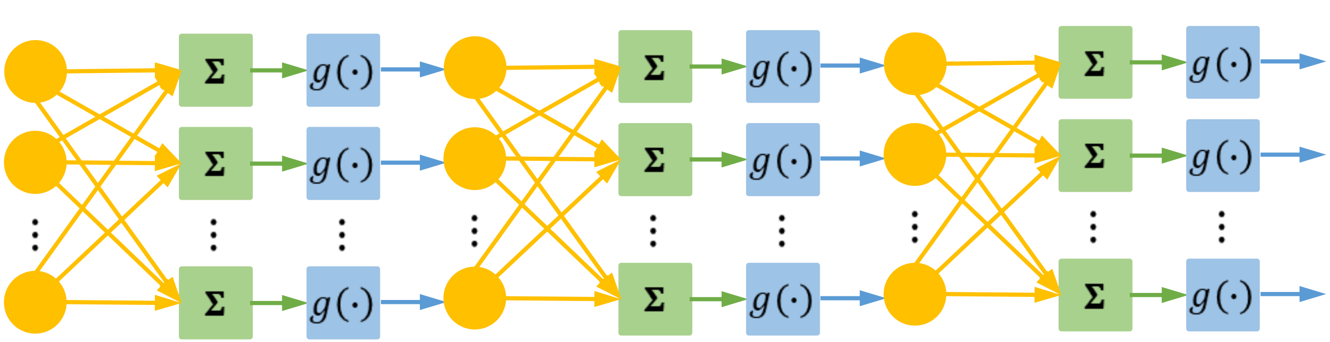
- Different connection leads to different structures; all the weights in the "neurons" make up the network parameter
- *Fully-connected feedforward network*



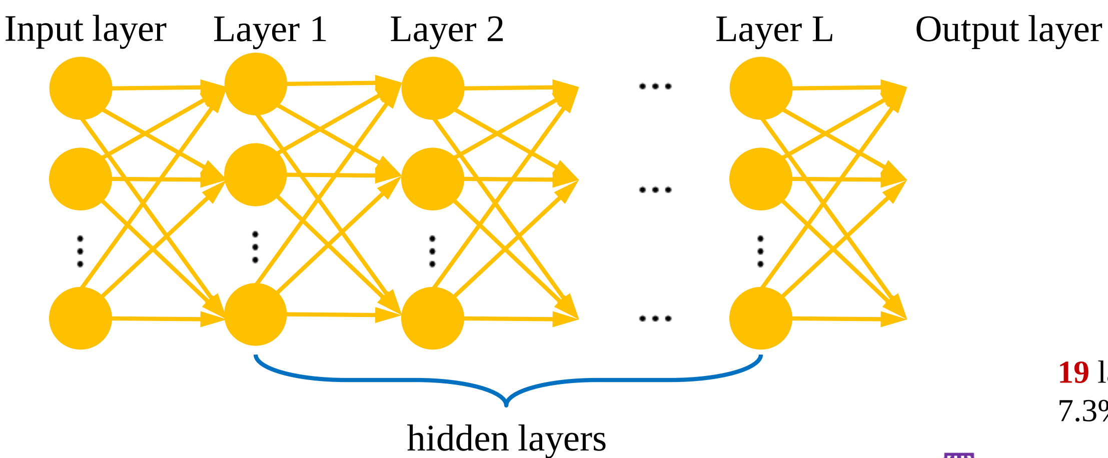
- Suppose you already know all the parameters (weights) in this neural network, then it is a function that maps the input vector to a specific output vector

- If we don't know the parameters, but the network structure is given, which means, we have already define a *function set*

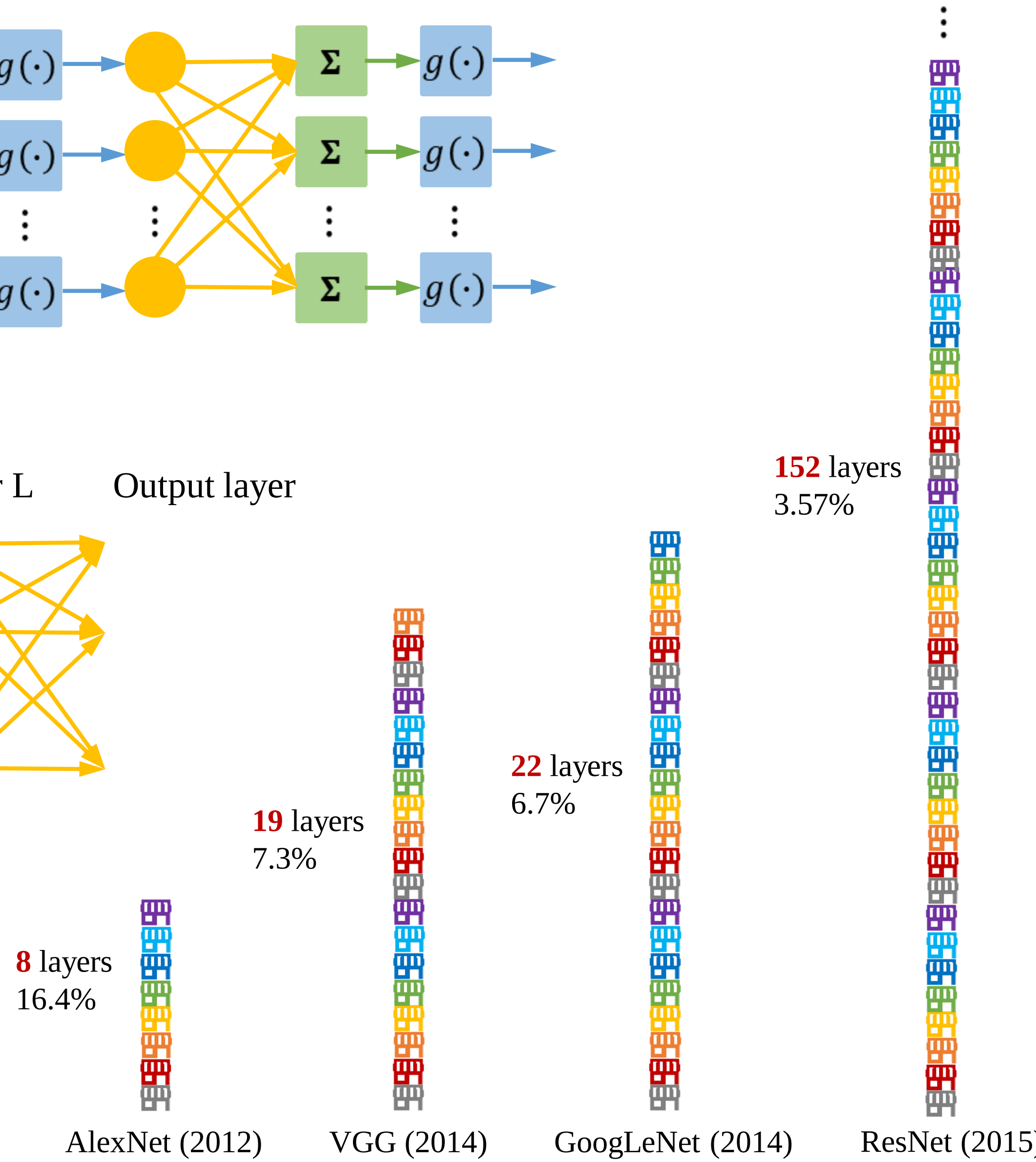
12.4.2 Deep means many hidden layers



- Sketch chart:

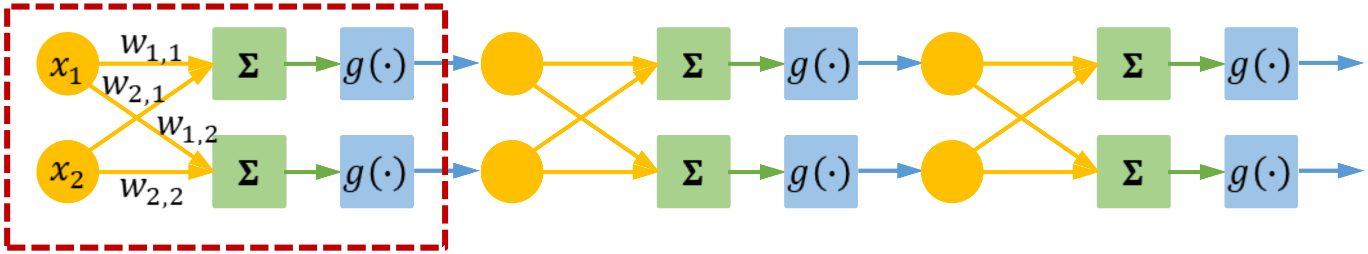


- Deep: many hidden layers. How many?



ILSVRC (*ImageNet Large Scale Visual Recognition Challenge*)

12.4.3 Highly nonlinear function



$$[w_{1,1} \quad w_{2,1}]\begin{bmatrix}x_1\\x_2\end{bmatrix}$$
$$[w_{2,1} \quad w_{2,2}]\begin{bmatrix}x_1\\x_2\end{bmatrix}$$

$$\Rightarrow \underbrace{\begin{bmatrix}w_{1,1} & w_{2,1}\\ w_{1,2} & w_{2,2}\end{bmatrix}}_{\mathbf{W}^\top}\begin{bmatrix}x_1\\x_2\end{bmatrix} \Rightarrow g(\mathbf{W}^\top\mathbf{x})$$

Layer 1

$g(\mathbf{W}_1^\top\mathbf{x})$

Layer 2

$g\left(\mathbf{W}_2^\top g(\mathbf{W}_1^\top\mathbf{x})\right)$

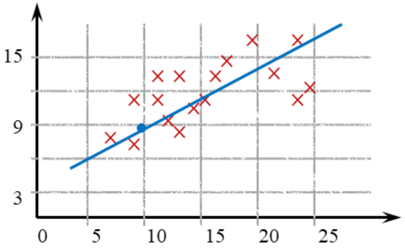
Layer 3

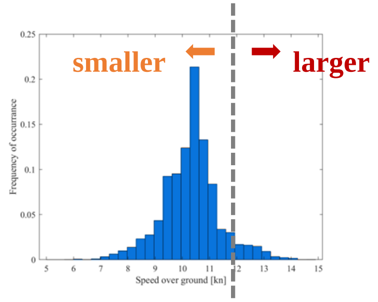
$g\left(\mathbf{W}_3^\top g\left(\mathbf{W}_2^\top g(\mathbf{W}_1^\top\mathbf{x})\right)\right)$

- Thus a neural network actually does a series of matrix operations

$$f(\mathbf{x};\boldsymbol{\theta}) = g\left(\mathbf{W}_L^\top \cdots g\left(\mathbf{W}_2^\top g(\mathbf{W}_1^\top\mathbf{x})\right)\right)$$

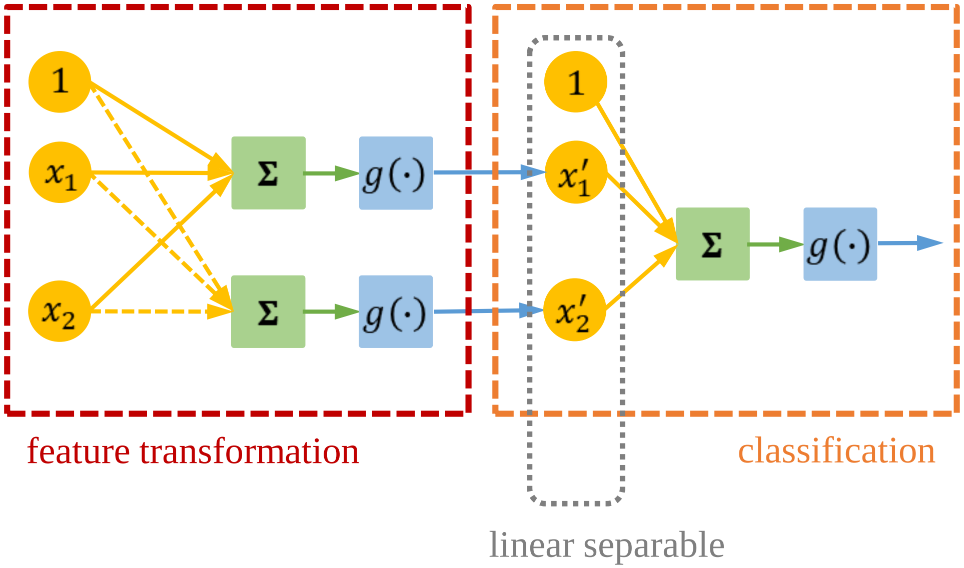
- A highly nonlinear function (*is able to fit any function in theory*)
- Only the input and output dimension are fixed

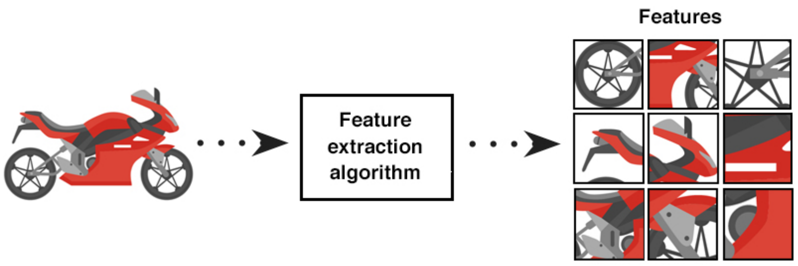




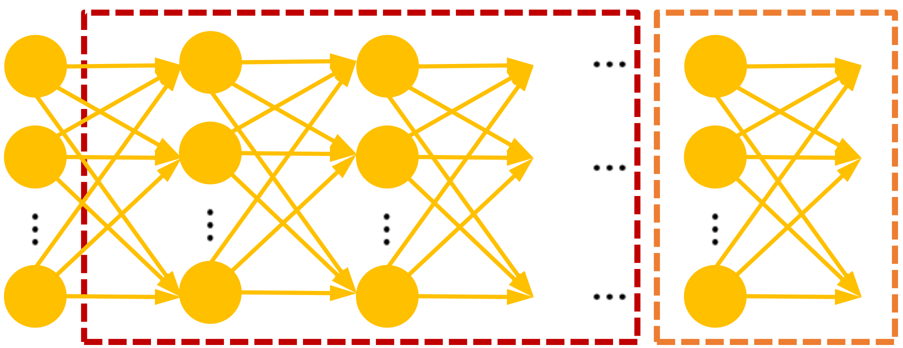
12.4.4 Feature extractor

- Logistic regression:



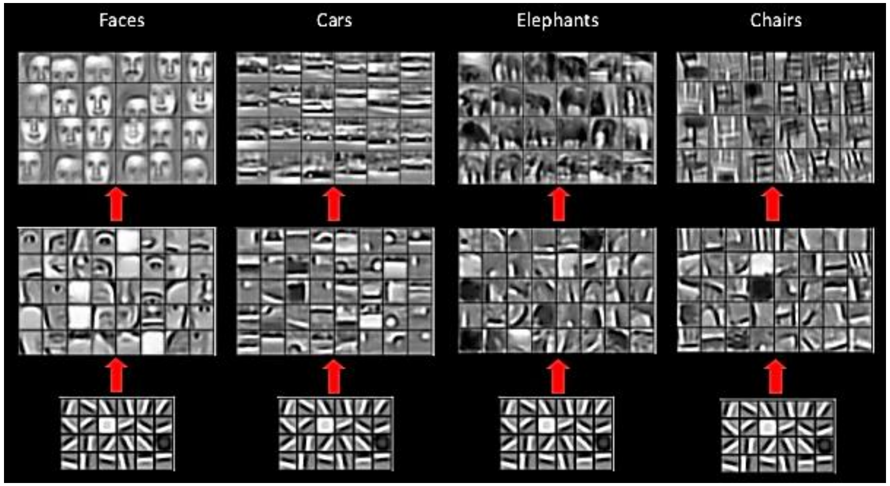


- Neural network:



Feature extractor
replace traditional feature engineering

classification/
regression



12.4.5 Discussions

- **Step 1. Define a function set**
  ① any neuron number
  ② any layer number
  ③ many nonlinear function to choose
  ④ any connection mode
  ⑤ linear + nonlinear? nonlinear + linear?
  ⑥ other style, e.g. residual unit, convolution, batch normalization, …

- **Step 2. Define the loss function**
  ① regression: square error function, …
  ② classification: cross entropy, …
  ③ …

- **Step 3. Optimize parameters**
  ① A family of gradient descent
  ② Others
  ③ backpropagation

- **Questions:**
  ① How many neurons?
  ② How many layers?
  ③ How to choose nonlinear function?
  ④ How to connect?
  ⑤ …
  ⑥ Can the structure be automatically determined? (evolutionary algorithm)
  ⑦ Does it converge? (local optimal)
  ⑧ …
  ⑨ Why deep?

AlexNet (2012)	VGG (2014)	GoogLeNet (2014)	ResNet (2015)
8 layers	**19** layers	**22** layers	**152** layers
16.4%	7.3%	6.7%	3.57%

- bigger function space

12.4.6 Why deep

AlexNet (2012)	VGG (2014)	GoogLeNet (2014)	ResNet (2015)
8 layers	**19** layers	**22** layers	**152** layers
16.4%	7.3%	6.7%	3.57%

Universality theorem

Any continuous function $f: \mathbb{R}^n \rightarrow \mathbb{R}^m$ can be realized by a network with one hidden layer, by giving enough hidden neurons.

From http://neuralnetworksanddeeplearning.com/chap4.html

- more parameters, bigger function space

- Why "deep" learning not "fat" learning?
- **Q.** Can you compare a "fat and short" network with a "thin and tall" network: be careful to keep the same number of parameters!

Size
- Smallest
- Small
- Medium
- Large
- Giant

Characteristics
- Hypoallergenic
- Fluffy
- Best family
- Smartest
- Best guard
- Kid friendly
- Best watch
- Easy to train
- Low shedding

Color
- White
- Black
- Blue
- Brown
- Red
- Grey
- Golden

Group
- Toy
- Sporting
- Hound
- Terrier
- Working
- Herding

- Modularization: cut a big complex problem into small simple sub-problems