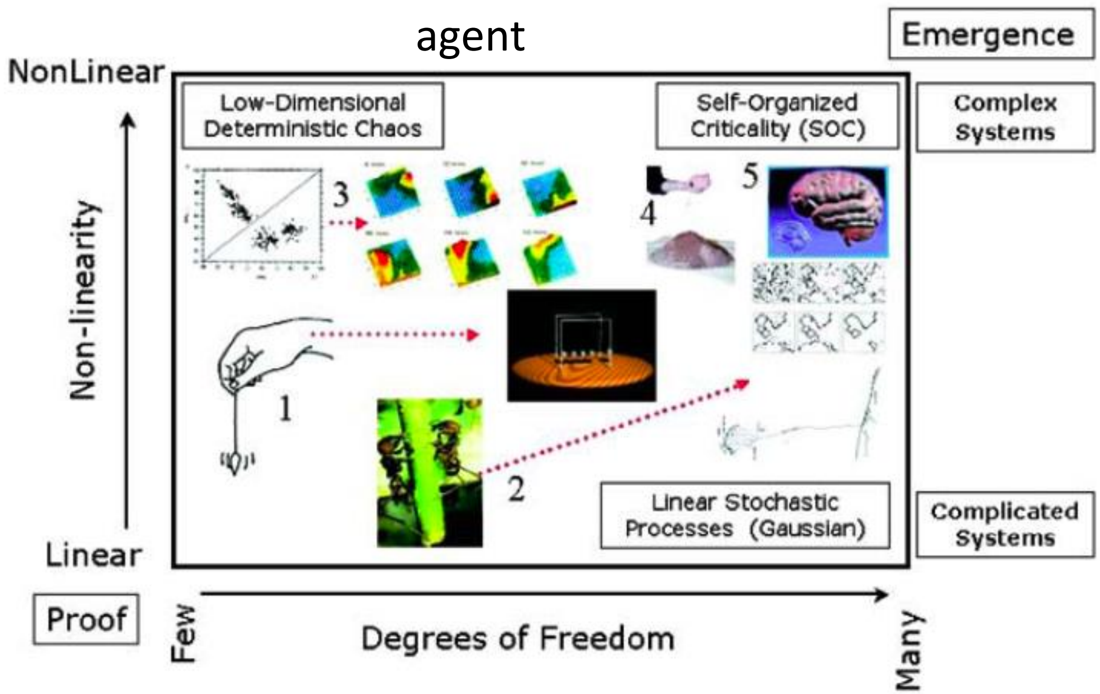
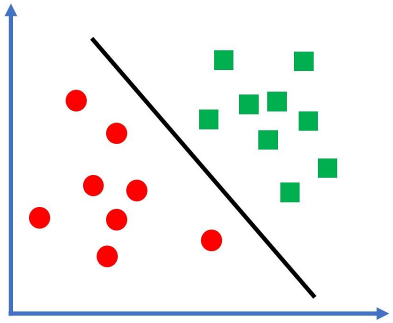
线性模型

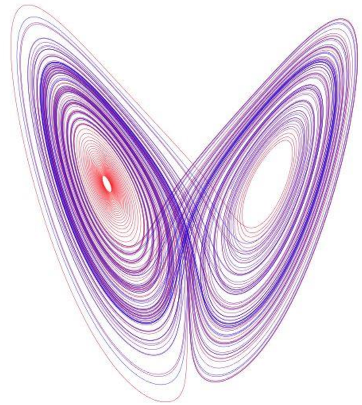
CSL

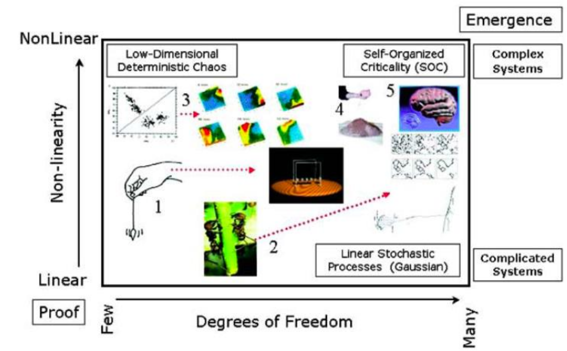
复杂系统

I think the next century will be the century of complexity. — Stephen Hawking （2000）









线性系统 → 非线性系统 → 复杂系统

1+1=2

1+1＞2

Linear model

机器学习：数据驱动的方法论 （工具）

程序员：能通过经验（数据）自动改进算法的计算机程序

应用数学：逼近论，函数拟合 （神经网络, ...）

统计学：不需要解释的简易统计学应用 （线性回归， ...）

CSL

数据挖掘：数据->信息->知识->智慧 （目标）

Linear model

经验主义：强调知识源于经验、强调归纳推理

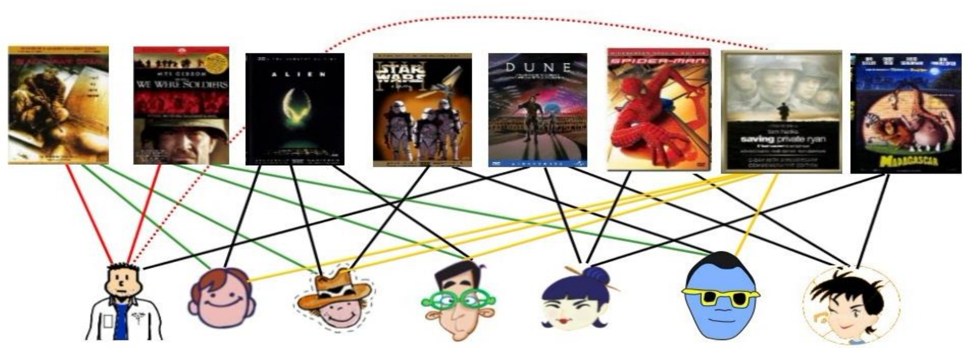
两类任务

- 机器学习人的认知能力 （理性）： speech, NLP，CV

CSL

- 机器学习/理解人的行为 （有限理性、非理性）

model



LEARNING = REPRESENTATION + EVALUATION +OPTIMIZATION

机器学习 = 模型 + 评估 + 优化

Representation	Evaluation	Optimization
Instances	Accuracy/Error rate	Combinatorial optimization
K-nearest neighbor	Precision and recall	Greedy search
Support vector machines	Squared error	Beam search
Hyperplanes	Likelihood	Branch-and-bound
Naive Bayes	Posterior probability	Continuous optimization
Logistic regression	Information gain	Unconstrained
Decision trees	K-L divergence	Gradient descent
Sets of rules	Cost/Utility	Conjugate gradient
Propositional rules	Margin	Quasi-Newton methods
Logic programs		Constrained
Neural networks		Linear programming
Graphical models		Quadratic programming
Bayesian networks		
Conditional random fields		

Domingos P M. *A few useful things to know about machine learning*[J]. Communications of The ACM, 2012, 55(10): 78-87.

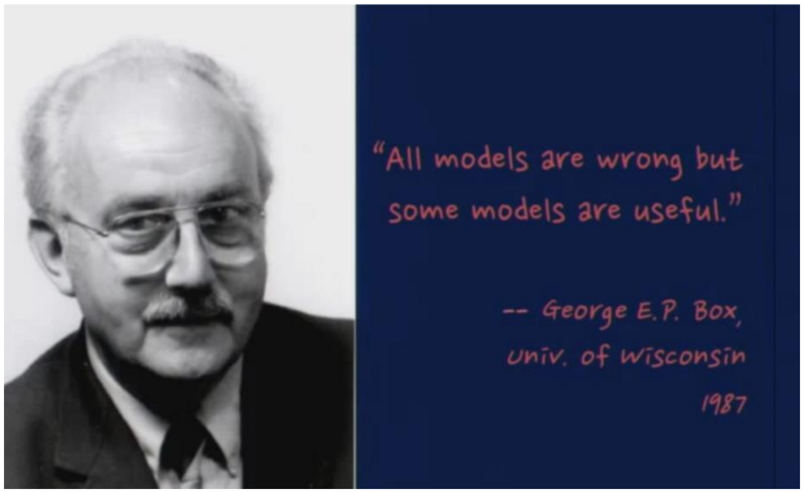
机器学习 = **模型** + 评估 + 优化

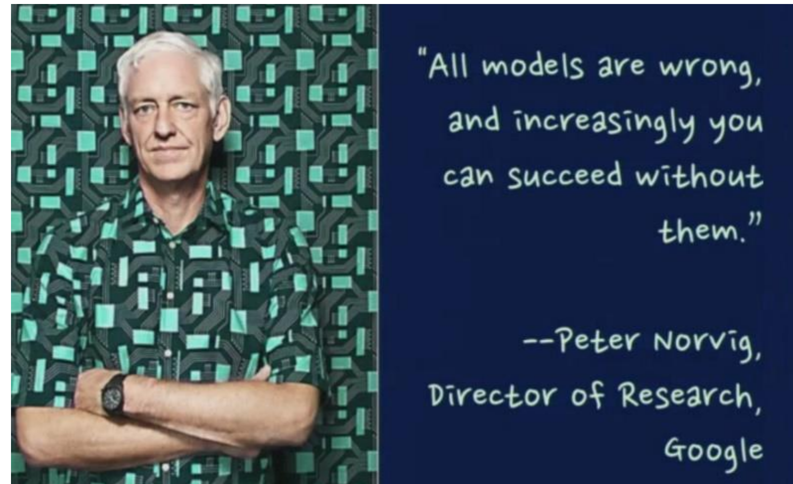
假设空间

$$\mathcal{F} = \{f \mid Y = f(X)\} \qquad \mathcal{F} = \{f \mid Y = f_\theta(X), \theta \in \mathbf{R}^n\}$$

$$\mathcal{F} = \{P \mid P(Y \mid X)\} \qquad \mathcal{F} = \{P \mid P_\theta(Y \mid X), \theta \in \mathbf{R}^n\}$$

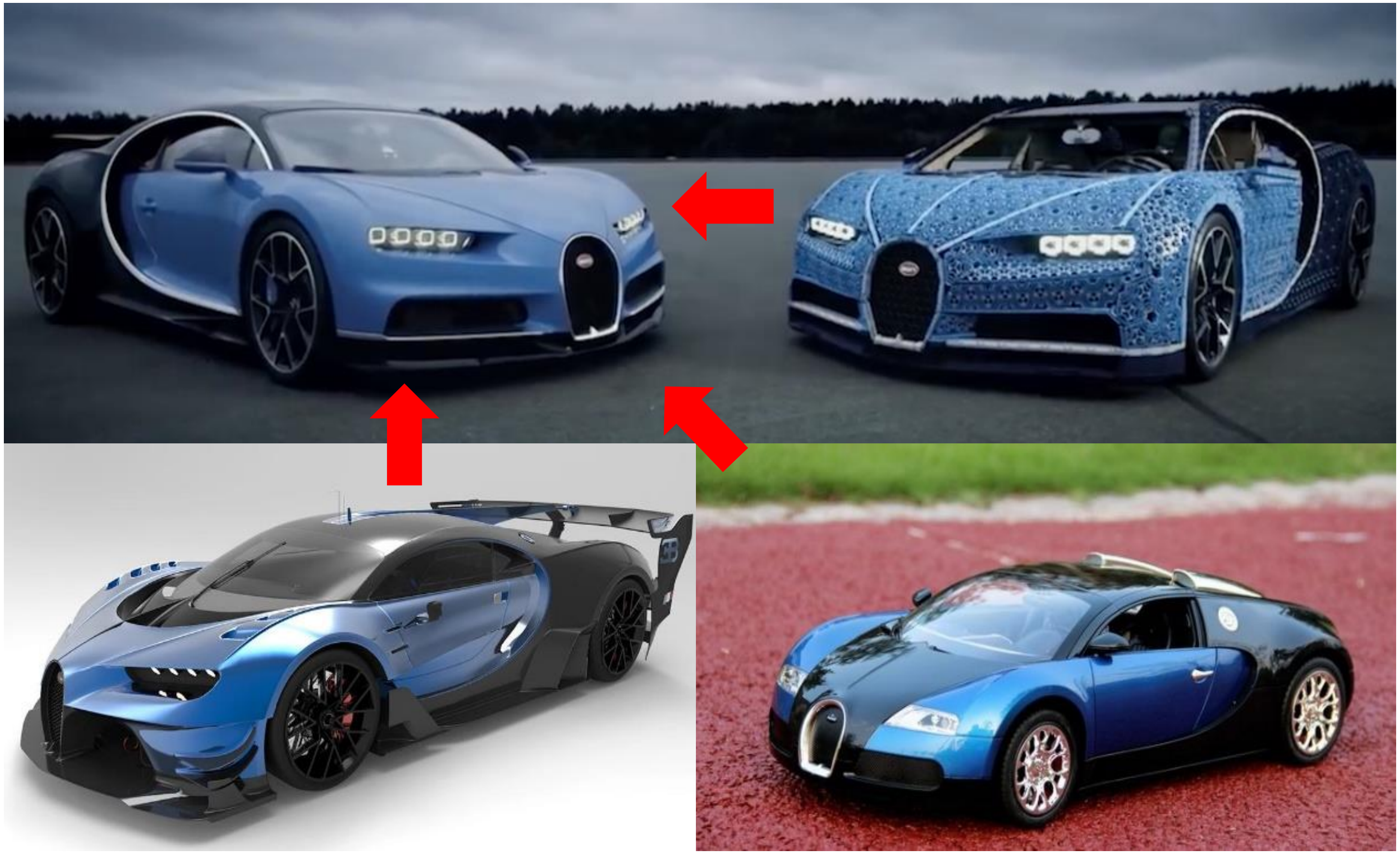
All models are wrong ?





机器学习 = 模型 + 评估 + 优化

度量模型与真实世界的相似性



机器学习 = 模型 + **评估** + 优化

度量模型与真实世界的相似性

Y_i 损失函数 $L(y_i, f(x_i))$ $f(x_i)$

Linear model

经验风险（Empirical risk） (in-sample error)

$$R_{emp}(f) = 1/N \sum_{i=1}^{N} L(y_i, f(x_i))$$

泛化误差（General error） (out-of-sample error)

$$R_{gen}(f) = E_{p(x,y)} L(y_i, f(x_i))$$

机器学习 = **模型** + **评估** + 优化

度量模型与真实世界的相似性

$$Y_i \longrightarrow L(y_i, f(x_i)) \longleftarrow f(x_i)$$

0/1 损失:

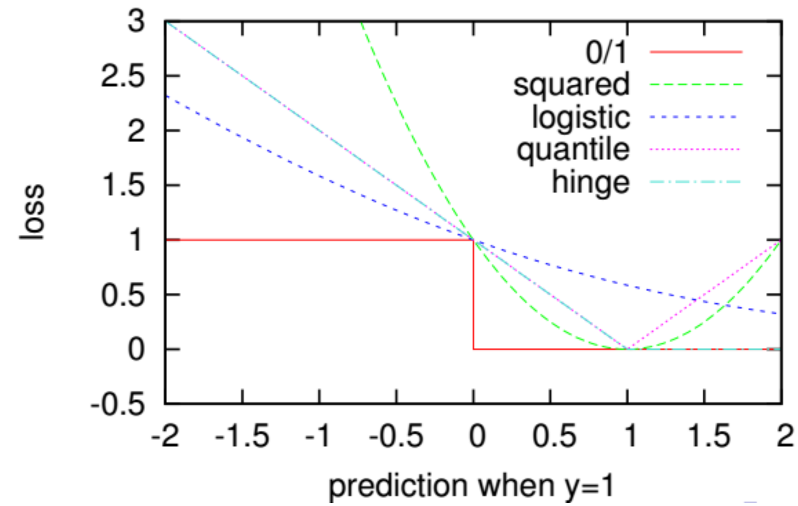
$$L\big(Y, f(x)\big) = \begin{cases} 1, Y \neq f(X) \\ 0, Y = f(X) \end{cases}$$

平方损失（L2, MSE）：

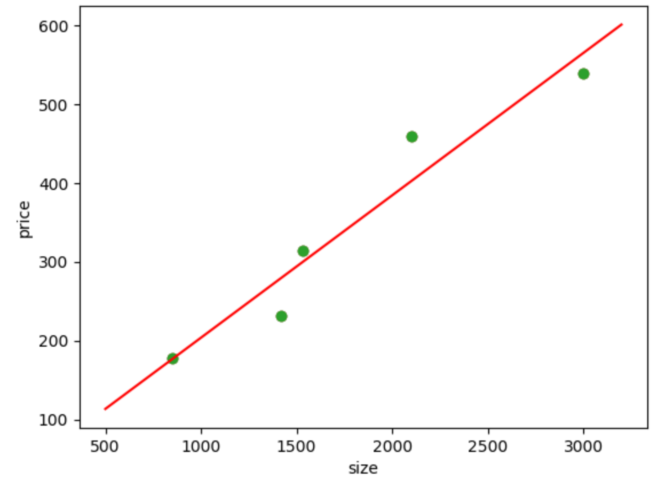
$$L\big(Y, f(x)\big) = \big(Y - f(X)\big)^2$$

对数损失（交叉熵损失）：

$$L\big(Y, f(x)\big) = -\log P(Y|X)$$



机器学习 = 模型 + 评估 + 优化



经验风险（Empirical risk）（in-sample error)

min $$R_{emp}(f) = 1/N \sum_{i=1}^{N} L(y_i, f(x_i))$$

X = (2104, 1416, 1534, 852, 3000), Y = (460, 232, 315,178, 540)

$$f(x) = ax + b$$

$$J(a, b) = \frac{1}{N} \sum_i (ax_i + b - y_i)^2$$

Linear

$$X = \begin{pmatrix} 2104,1 \\ 1416,1 \\ 1534,1 \\ 852,1 \\ 3000,1 \end{pmatrix}, \quad w = \begin{pmatrix} a \\ b \end{pmatrix}, \quad Y = \begin{pmatrix} 460 \\ 232 \\ 315 \\ 178 \\ 540 \end{pmatrix}$$

normal equation

$$\min J(\mathbf{w}) = \frac{1}{2} \sum_i \left(\mathbf{Y}_i - \mathbf{w}^T X_i\right)^2$$

$$X = \begin{pmatrix} X_1^T \\ X_2^T \\ \dots \\ X_N \end{pmatrix} \qquad J = \frac{1}{2} (Y - Xw)^T (Y - Xw)$$

$$X^T(Y - Xw) = 0$$

$$X^T X w = X^T Y$$

$$\widehat{w}_{OLS} = \left(X^T X\right)^{-1} X^T Y$$

array([[0.18078802],
[22.98037744]])

机器学习 = **模型** + **评估** + **优化**

经验风险（Empirical risk） (in-sample error)

min $$R_{emp}(f) = 1/N\sum_{i=1}^{N} L(y_i, f(x_i)) \quad f?$$

CSL

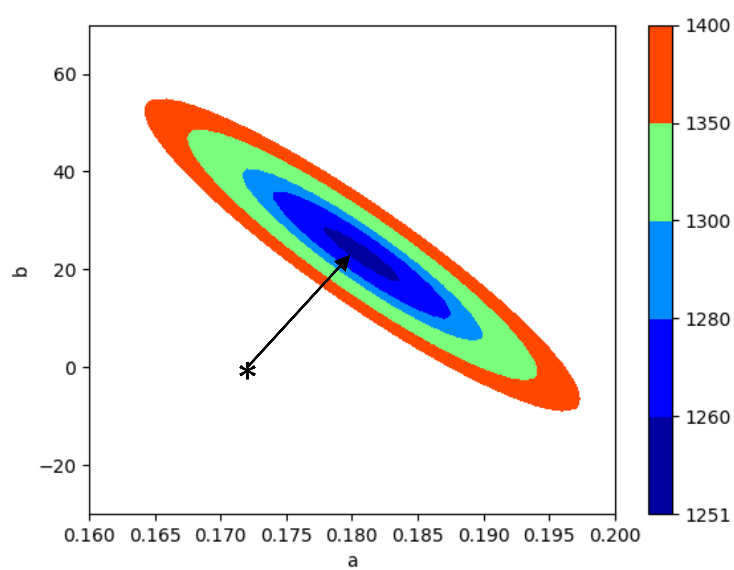
X = (2104, 1416, 1534, 852, 3000), Y = (460, 232, 315,178, 540)

模型： $f(x) = ax + b$

评估： $J(a,b) = \dfrac{1}{N}\sum_i (ax_i + b - y_i)^2$

Linear

优化： $\min J(a,b)$



机器学习 = 模型 + 评估 + 优化

Taylor展开

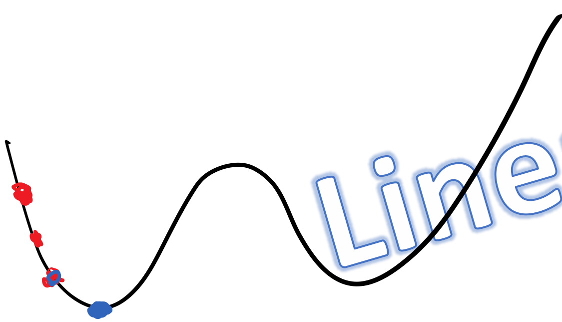
$$f(x+\Delta x)=f(x)+f'(x)\Delta x+1/2\,f''(x)\Delta x^2+o(\Delta x^2)$$

梯度下降法 (Gradient decent)

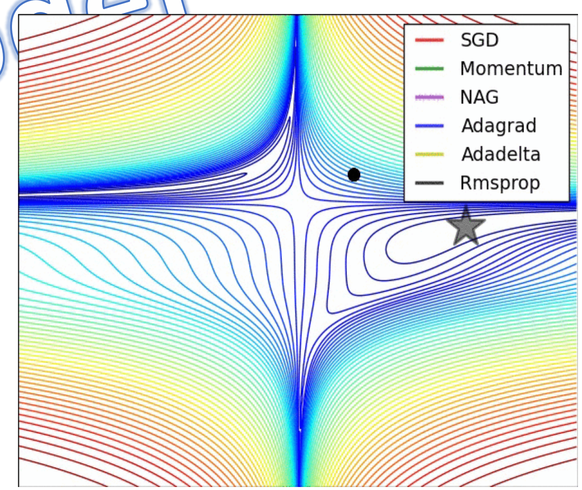
$$f(x+\Delta x)=f(x)+\nabla f(x)\Delta x+\frac{1}{2\alpha}\Delta x^2+o(\Delta x^2)$$

$$min\ f(x_0+\Delta x)=f(x_0)+\nabla f(x_0)\Delta x+\frac{1}{2\alpha}\Delta x^2+o(\Delta x^2)$$

$$\Delta x=-\alpha\nabla f(x_0) \qquad x_{k+1}=x_k-\alpha\nabla f(x_k)$$



$$f(x_0)\to f(x_1)\to f(x_2)\to\cdots$$



梯度下降法: 收敛率 $O(\frac{1}{K})$　　**近似 （已知->未知）**

机器学习 = 模型 + 评估 + 优化

Taylor展开

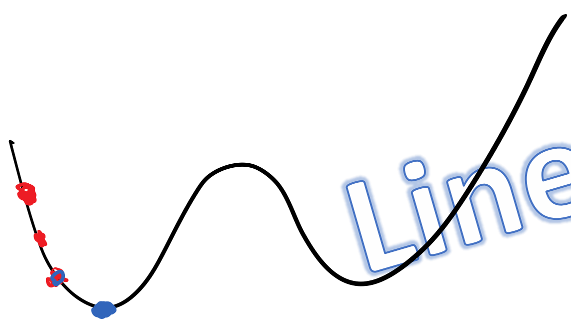
$$f(x+\Delta x) = f(x) + f'(x)\Delta x + 1/2\, f''(x)\Delta x^2 + o(\Delta x^2)$$

Newton法

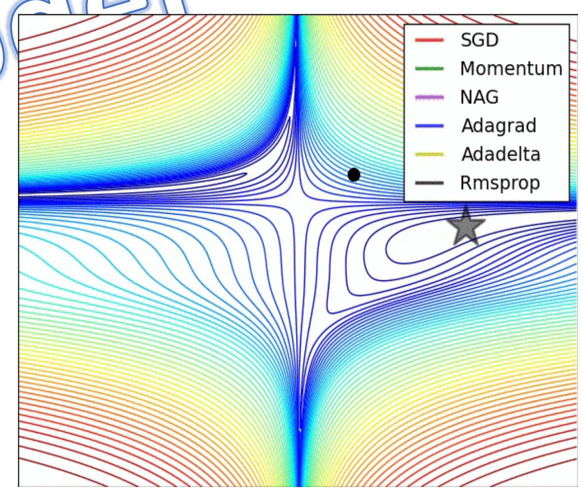
$$f(x+\Delta \mathrm{x}) = f(x) + \nabla f(x)\Delta x + \frac{1}{2}\nabla^2 f(x)\Delta x^2 + o(\Delta x^2)$$

$$min\, f(x_0+\Delta \mathrm{x}) = f(x_0) + \nabla f(x_0)\Delta x + \frac{1}{2}\nabla^2 f(x_0)\Delta x^2 + o(\Delta x^2)$$

$$\Delta x = -\nabla f(x_0)/\nabla^2 f(x_0) \qquad x_{k+1} = x_k - \nabla f(x_0)/\nabla^2 f(x_0)$$



$$f(\mathbf{x}_0) \rightarrow \mathbf{f}(\mathbf{x}_1) \rightarrow f(x_2) \rightarrow \cdots$$



Newton法：收敛率 $O(\frac{1}{K^2})$

$$\min J(\mathbf{w}) = \frac{1}{2N}\sum_i \left(\mathbf{Y}_i - \mathbf{w}^T \boldsymbol{X}_i\right)^2$$

$$J(a,b) = \frac{1}{2N}\sum_i (ax_i + b - y_i)^2$$

while k > 最大步数 or $\nabla J(a_k, b_k) \approx 0$

$$a_{k+1} = a_k - \alpha \frac{\partial \mathrm{J}(a_k, b_k)}{\partial a}$$

$$b_{k+1} = b_k - \alpha \frac{\partial \mathrm{J}(a_k, b_k)}{\partial b_k}$$

$$J'(\mathbf{w}) = \frac{1}{N}\sum \boldsymbol{X}_i^T\left(\mathbf{w}^T \boldsymbol{X}_i - \mathbf{Y}_i\right)$$

$$\frac{1}{N}\sum_i x_i(ax_i + b - y_i)$$

$$\frac{1}{N}\sum_i 1\,(ax_i + b - y_i)$$

批量梯度下降法

(Batch Gradient Descent)

Repeat {

$$w_{k+1} = w_k - \frac{1}{N}\sum \boldsymbol{X}_i^T\left(\mathbf{w}^T\boldsymbol{X}_i - Y\right)$$

}

随机梯度下降法

(Stochastic Gradient Descent)

1. Randomly shuffle dataset;
2. **repeat**{
   for i=1, ... , N
   {
   $$w_{k+1} = w_k - \boldsymbol{X}_i^T\left(\mathbf{w}^T\boldsymbol{X}_i - Y\right)$$
   }
   }

Mini-batch Gradient Descent: batch size

批量梯度下降法
(Batch Gradient Descent)

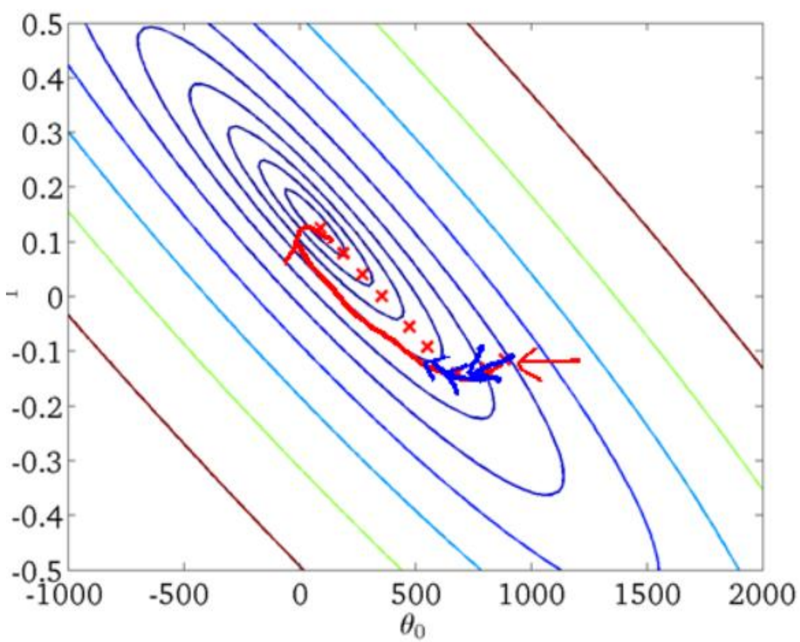
Repeat {

$$w_{k+1} = w_k - \frac{1}{N}\sum \boldsymbol{X}_i^T(\mathbf{w}^T\boldsymbol{X}_i - \boldsymbol{Y})$$

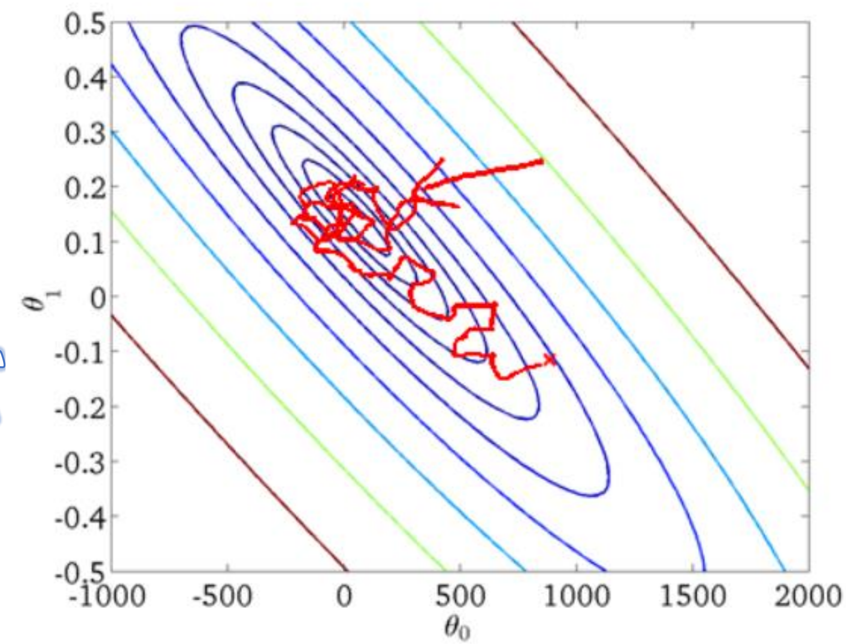
}

随机梯度下降法
(Stochastic Gradient Descent)

1. Randomly shuffle dataset;
2. **repeat**{
   for i=1, … , N
   {
   $$w_{k+1} = w_k - \boldsymbol{X}_i^T(\mathbf{w}^T\boldsymbol{X}_i - \boldsymbol{Y})$$
   }
   }



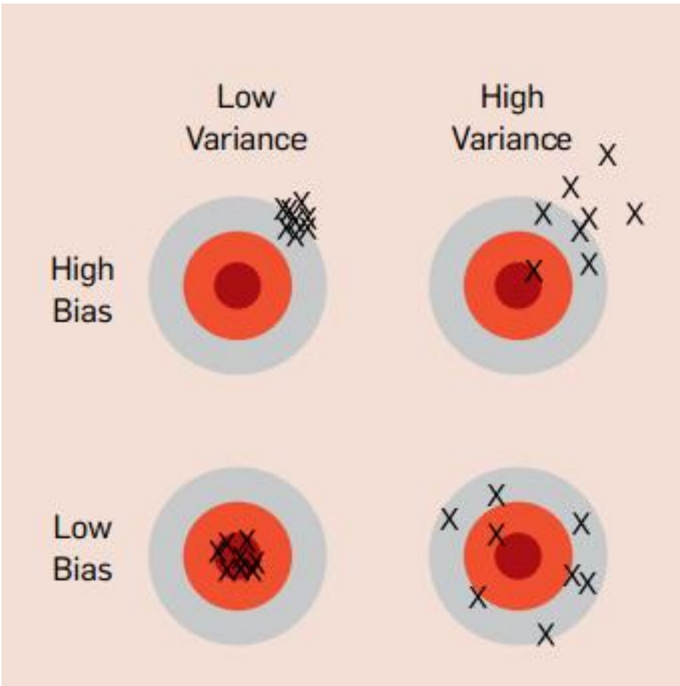
收敛率 $O(\frac{1}{K})$



收敛率 $O(\frac{1}{\sqrt{K}})$

偏差与方差

$$R_{emp}(f) = 1/N\sum_{i=1}^{N} L(y_i, f(x_i)) \qquad R_{gen}(f) = E_{p(x,y)}L(y_i, f(x_i))$$



$$\begin{aligned}
\mathrm{E}\big[(y-\hat{f})^2\big] &= \mathrm{E}[y^2 + \hat{f}^2 - 2y\hat{f}] \\
&= \mathrm{E}[y^2] + \mathrm{E}[\hat{f}^2] - \mathrm{E}[2y\hat{f}] \\
&= \mathrm{Var}[y] + \mathrm{E}[y]^2 + \mathrm{Var}[\hat{f}] + \mathrm{E}[\hat{f}]^2 - 2f\mathrm{E}[\hat{f}] \\
&= \mathrm{Var}[y] + \mathrm{Var}[\hat{f}] + (f^2 - 2f\mathrm{E}[\hat{f}] + \mathrm{E}[\hat{f}]^2) \\
&= \mathrm{Var}[y] + \mathrm{Var}[\hat{f}] + (f - \mathrm{E}[\hat{f}])^2 \\
&= \mathrm{Var}[y] + \mathrm{Var}[\hat{f}] + \mathrm{E}[f - \hat{f}]^2 \\
&= \sigma^2 + \mathrm{Var}[\hat{f}] + \mathrm{Bias}[\hat{f}]^2
\end{aligned}$$

$$Err(x) = \mathrm{Bias}^2 + \mathrm{Variance} + \mathrm{Irreducible\ Error}$$

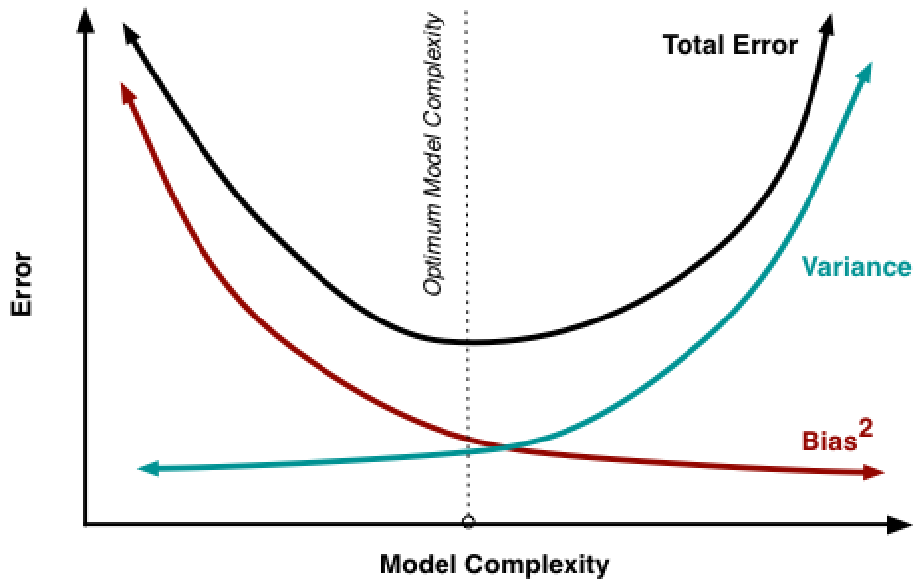
Bias: $\left(E[\hat{f}(x)] - f(x)\right)^2$ 度量了学习算法的平均估计结果能逼近学习目标的程度。

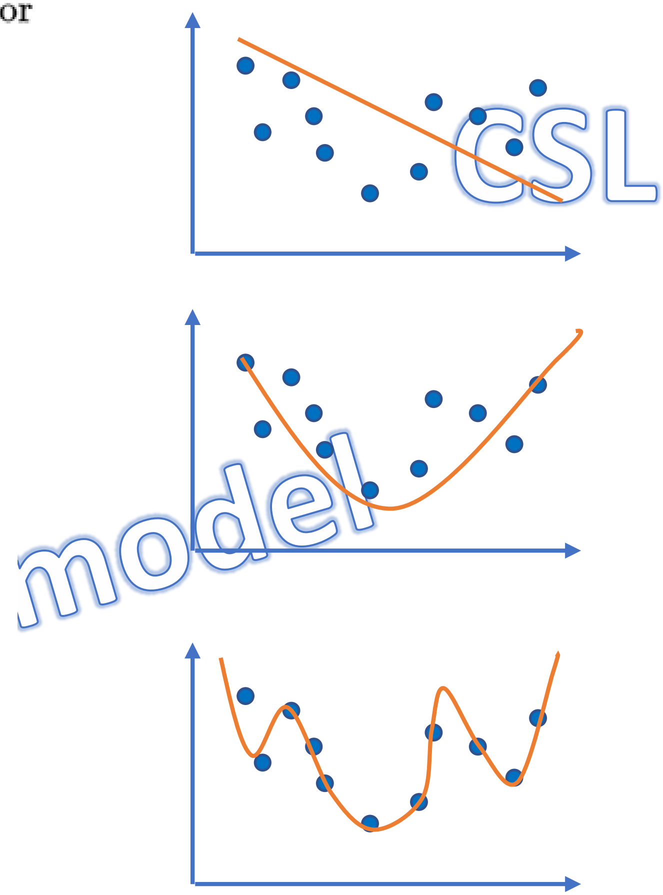
Variance: $E\left[\left(\hat{f}(x) - E[\hat{f}(x)]\right)^2\right]$ 度量了面对不同训练集时，学习算法的估计结果发生变动的程度。

偏差与方差

$$R_{emp}(f) = 1/N \sum_{i=1}^{N} L(y_i, f(x_i)) \qquad R_{gen}(f) = E_{p(x,y)} L(y_i, f(x_i))$$

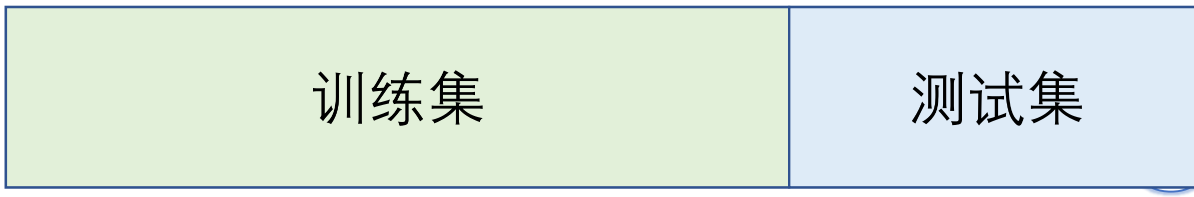
$$Err(x) = \mathrm{Bias}^2 + \mathrm{Variance} + \mathrm{Irreducible\ Error}$$

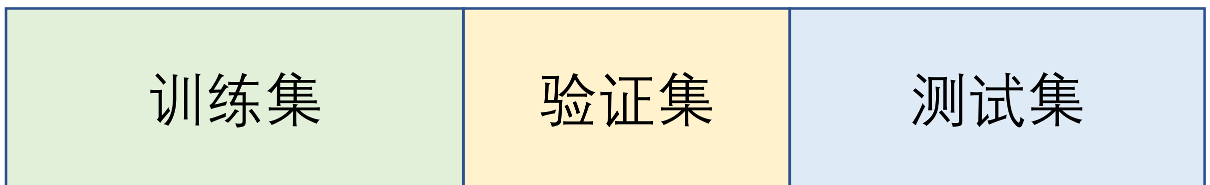




交叉验证（cross validation)

全部数据

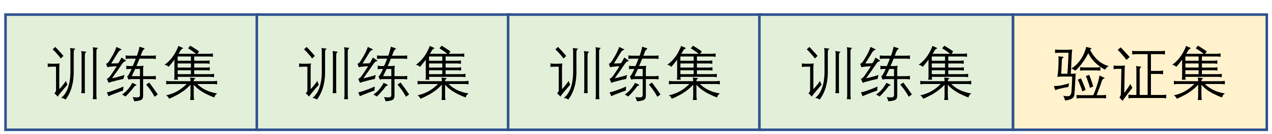


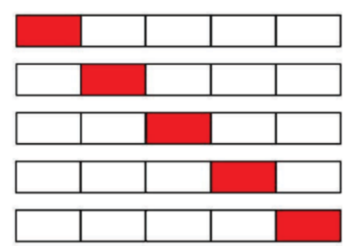


训练集 （training set）： 用于训练模型

验证集 （validation set）： 用于模型选择

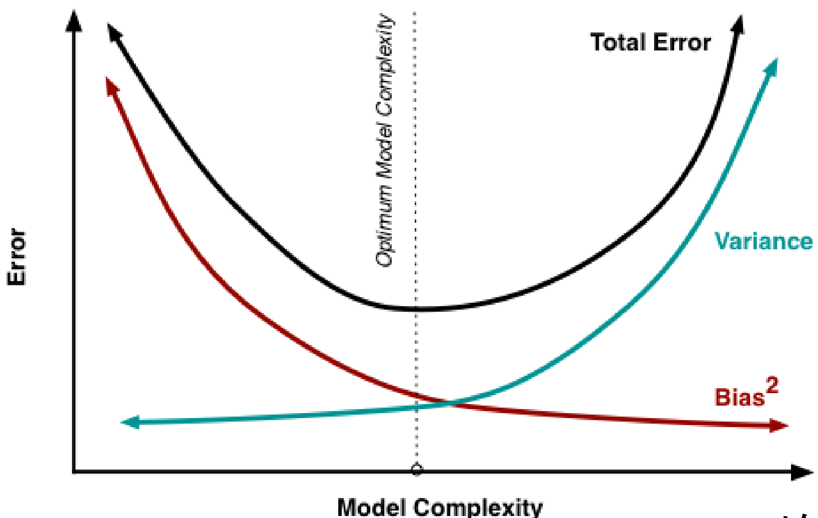
测试集 （test set）： 用于最终对学习方法的评估





正则化（regularization）

偏差与方差 $Err(x) = \text{Bias}^2 + \text{Variance} + \text{Irreducible Error}$

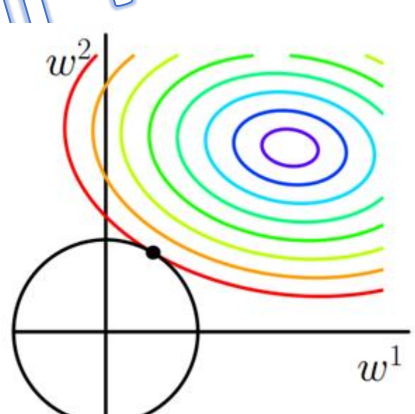


L_p 正则化

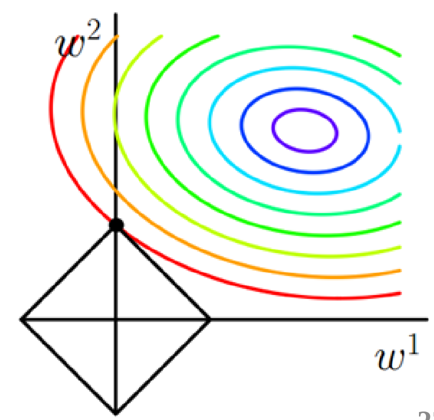
$$\hat{w}^{L_p} = \arg\min_{w\in R^d} \sum_{i=1}^{n}(w^T X_i - Y_i)^2 \text{ subject to } \sum_{j=1}^{d}|w_j|^p \le A.$$

$$\hat{w}^{L_p} = \arg\min_{w\in R^d} \sum_{i=1}^{n}(w^T X_i - Y_i)^2 + \lambda\sum_{j=1}^{d}|w_j|^p$$

岭回归(Ridge Regression)



LASSO



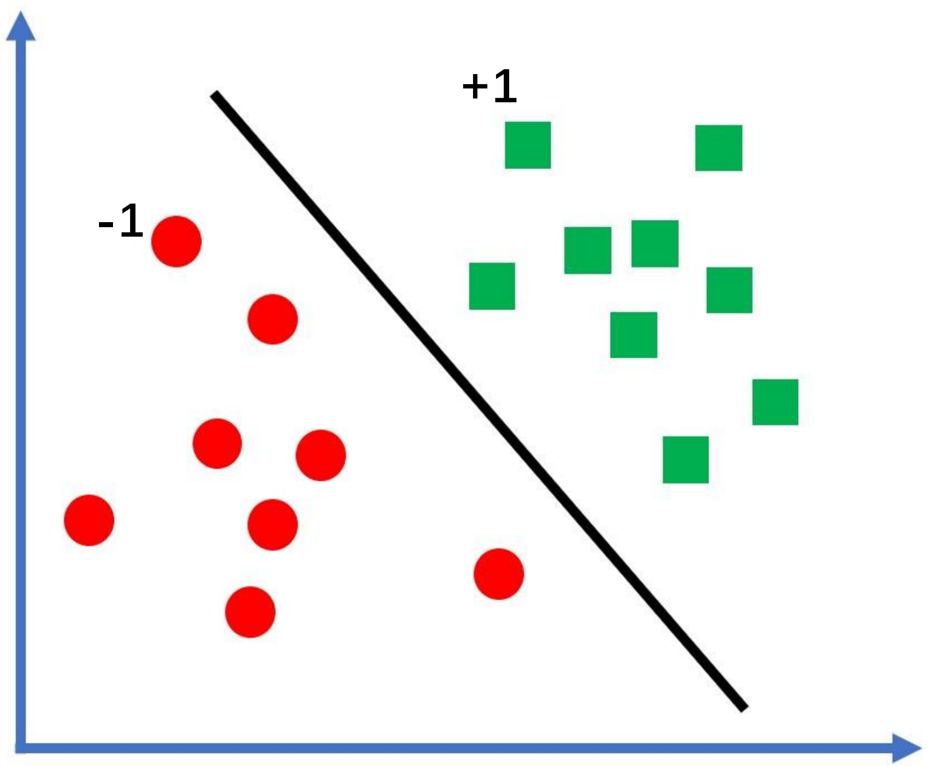
L_2正则化：$\Omega(w) = \lambda||w||_2^2 = \lambda\sum_1^d w_j^2$

L_1正则化：$\Omega(w) = \lambda||w||_1 = \lambda\sum_1^d |w_j|$

L_0正则化：$\Omega(w) = \lambda||w||_0$

线性分类器

$$f(x_i) = h(w^T x_i + b)$$



感知机（Perceptron）

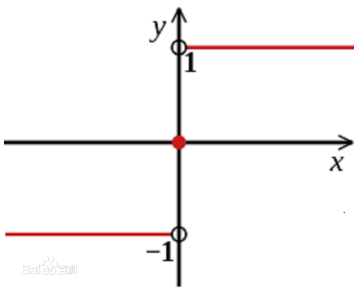
$$f(x_i) = sign(w^T x_i + b)$$

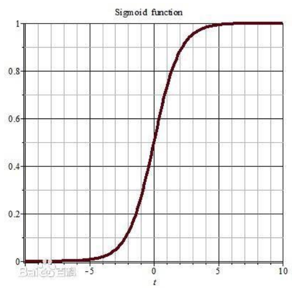
逻辑回归（Logistic regression）

$$f(x_i) = sigmoid(w^T x_i + b)$$

SVM（Support vector machine）

$$f(x_i) = sign(w^T x_i + b)$$

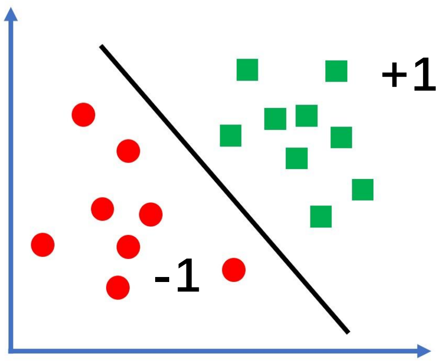


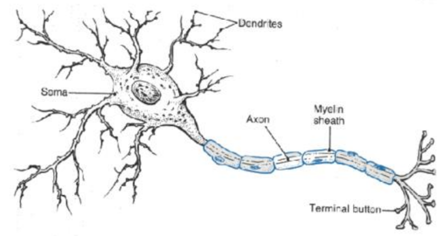


线性分类器

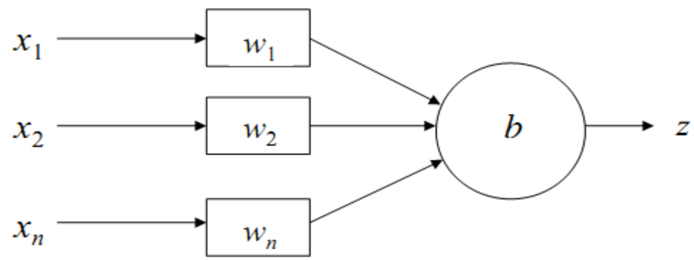
感知机

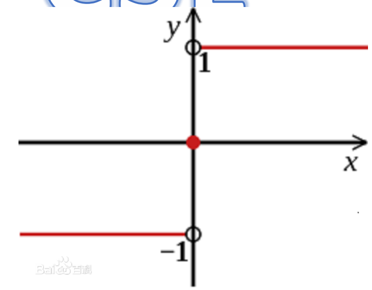
1957年由Rosenblatt提出





McCulloch-Pitts模型 $\boldsymbol{z = sign(w^T x_i + b)}$





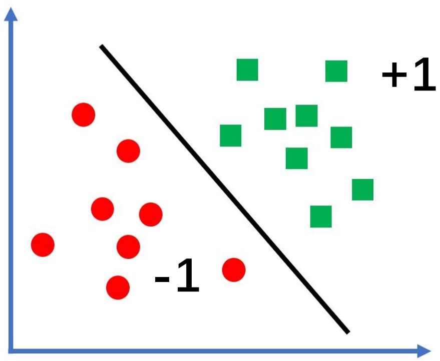
模型： $\boldsymbol{\hat{y} = w^T x_i + b}$

评估： $\boldsymbol{L(y_i, \hat{y}_i) = -y_i(w^T x_i + b)}$

优化： $\min \boldsymbol{J(w)} = \sum \boldsymbol{L(y_i, \hat{y}_i)} = -\sum_{\boldsymbol{i}} \boldsymbol{y_i(w^T x_i + b)}$

线性分类器

感知机



模型： $\hat{y} = w^T x_i + b$

评估： $L(y, \hat{y}) = -y_i(w^T x_i + b)$

优化： $\min J(w) = -\sum_i y_i(w^T x_i + b)$

PLA (Perception Learning Algorithm)

（1）选取初值w_0, b_0
（2）训练集中选取数据（xi, yi）
（3）if $y_i\left(w^T x_i + b\right) \le 0$:
$w = w + \alpha y_i x_i$
$b = b + \alpha y_i$
(4) 转至(2)，直到训练集中没有误分类点

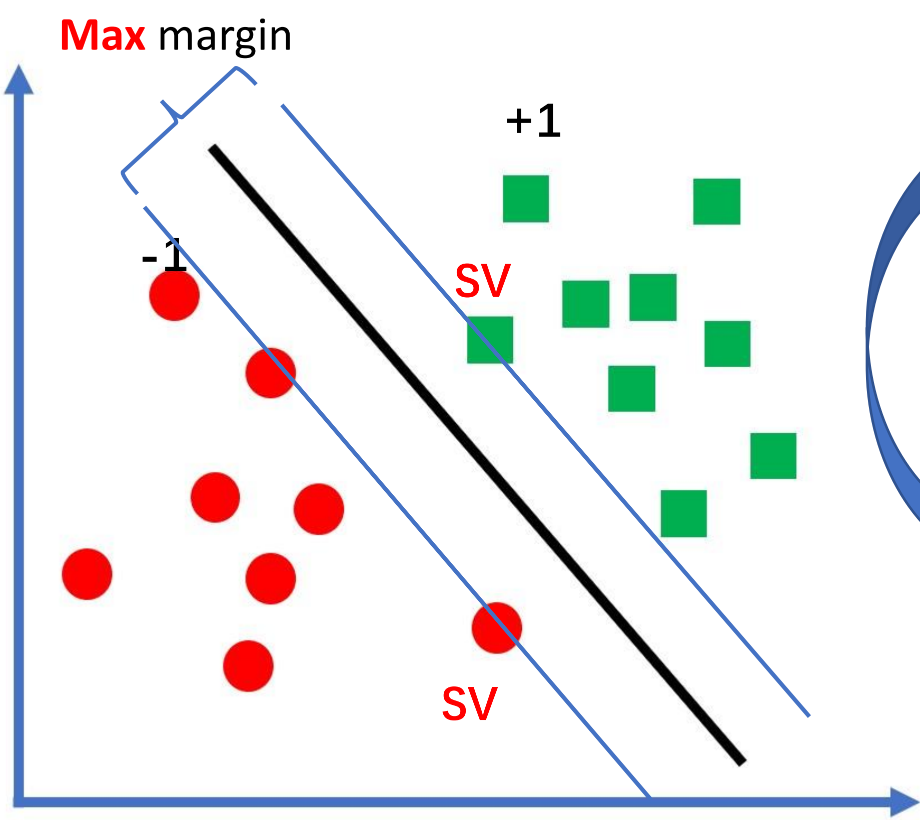
SGD

$$w_{k+1} = w_k - \alpha \frac{\partial J(w_k, b_k)}{\partial a}$$

$$b_{k+1} = b_k - \alpha \frac{\partial J(w_k, b_k)}{\partial b_k}$$

线性分类器

$$f(x_i) = h(w^T x_i + b)$$



感知机（Perceptron）

$$f(x_i) = sign(w^T x_i + b)$$

逻辑回归（Logistic regression）

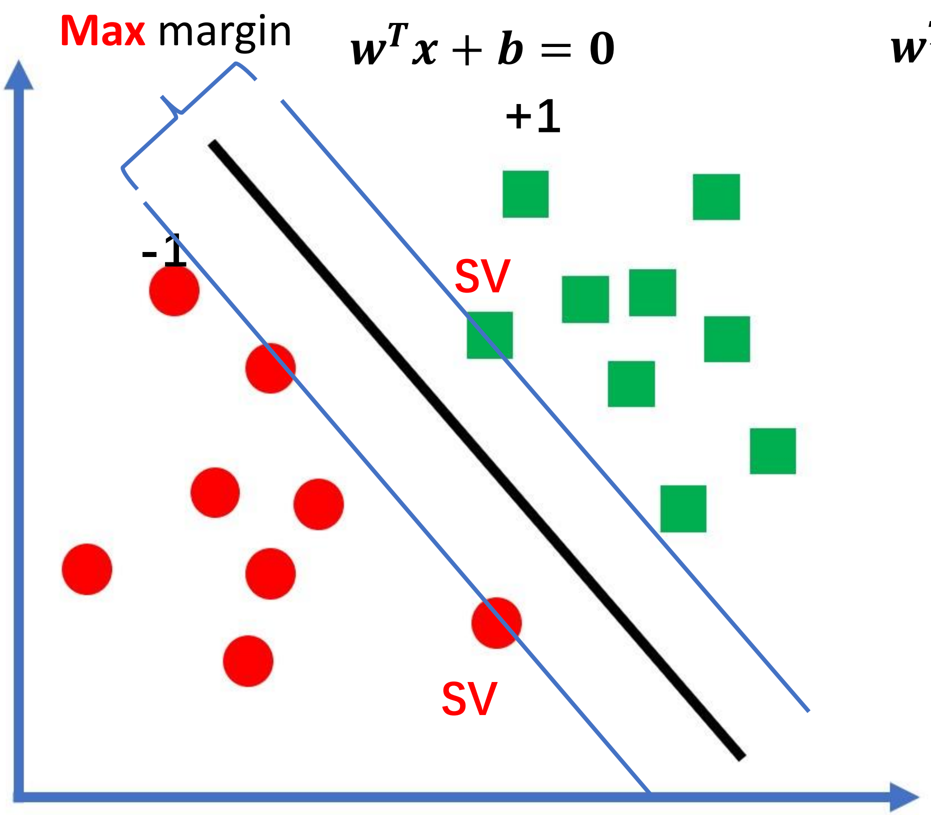
$$f(x_i) = sigmoid(w^T x_i + b)$$

SVM（Support vector machine）

$$f(x_i) = sign(w^T x_i + b)$$

线性分类器 SVM

$$\boldsymbol{f(x_i) = h(w^T x_i + b)}$$



$$\boldsymbol{w^T x + b = 0} \qquad \boldsymbol{d} = \frac{\boldsymbol{w^T x + b}}{||w||_2}$$

$$\max_{w,b} \min_i d_i = \frac{\boldsymbol{y_i(w^T x_i + b)}}{||w||_2}$$

$$\gamma = \min_i d_i = \frac{\boldsymbol{y_i(w^T x_i + b)}}{||w||_2}$$

$$\max_{w,b} \gamma \qquad \text{s.t. } \frac{y_i(w^T x_i + b)}{||w||_2} \geq \gamma$$

$$\boldsymbol{y_i}\left(\frac{\boldsymbol{w^T}}{||\boldsymbol{w}||_2 \boldsymbol{\gamma}}\boldsymbol{x} + \frac{\boldsymbol{b}}{||w||_2 \boldsymbol{\gamma}}\right) \geq 1$$

$$\max_{w,b} \frac{1}{||w'||}$$

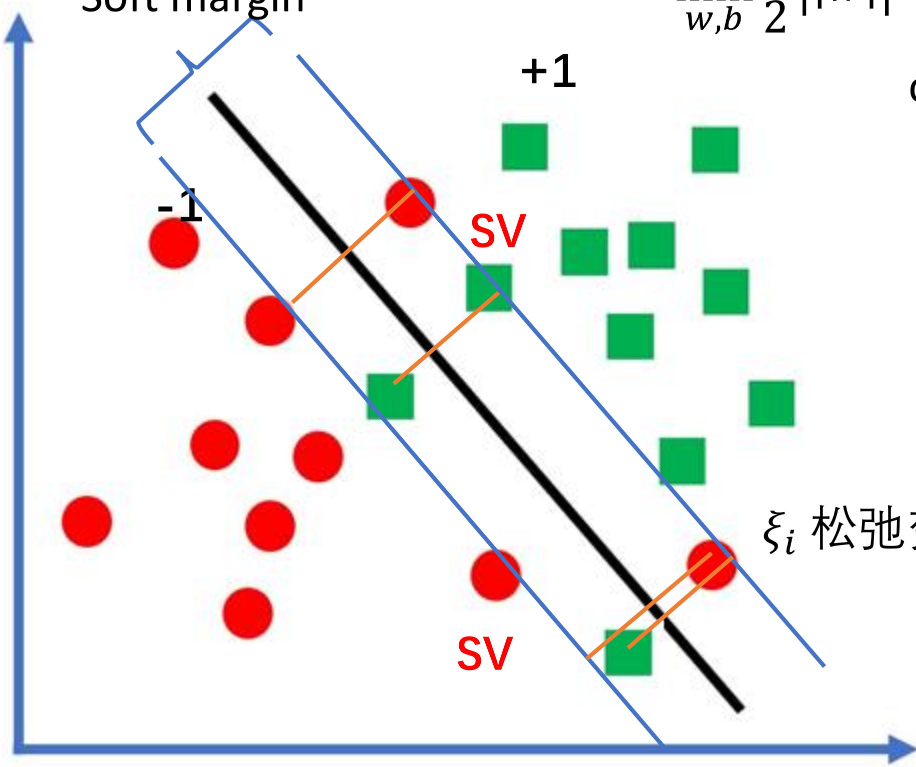
$$\boldsymbol{s.t.}\ \ \boldsymbol{y_i(w'x + b')} \geq 1$$

$$\min_{w,b} \frac{1}{2} ||w||^2$$

$$\boldsymbol{s.t.}\ \ \boldsymbol{y_i(wx_i + b)} \geq 1$$

线性分类器 SVM

Soft margin



$$\min_{w,b} \frac{1}{2}||w||^2 \qquad \boldsymbol{s.t.}\ \ \boldsymbol{y_i(wx_i + b)} \geq 1$$

convex quadratic programming

$$\min_{w,b} \frac{1}{2}||w||^2 + C\sum_i \xi_i$$

$$\boldsymbol{s.t.}\ \ \boldsymbol{y_i(wx_i + b)} \geq 1 - \xi_i$$

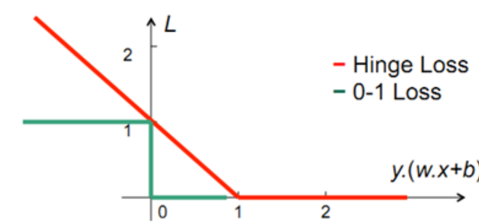
$$\xi_i \geq 0$$

ξ_i 松弛变量

$$\min_{w,b} \frac{1}{2C}||w||^2 + \sum_i \max(1 - y_i(wx_i + b), 0)$$

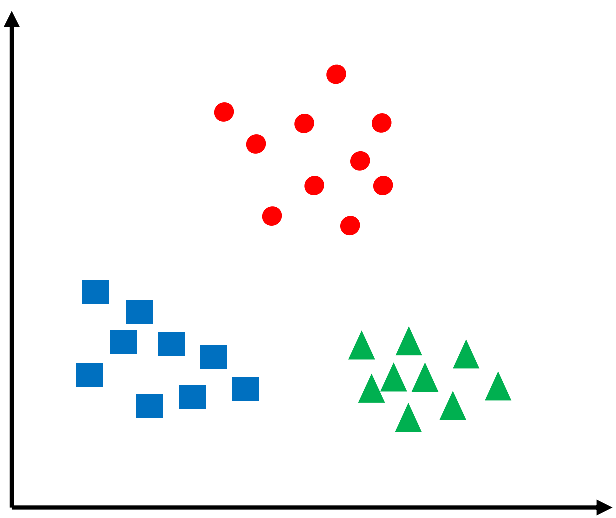
$$\min_{w,b} \sum_i \max(1 - y_i(wx_i + b), 0) + \lambda||w||^2$$

Hinge Loss: $\mathrm{L}(\mathrm{y_i}, \mathrm{wx_i} + \mathrm{b}) = \max(1 - y_i(wx_i + b), 0)$

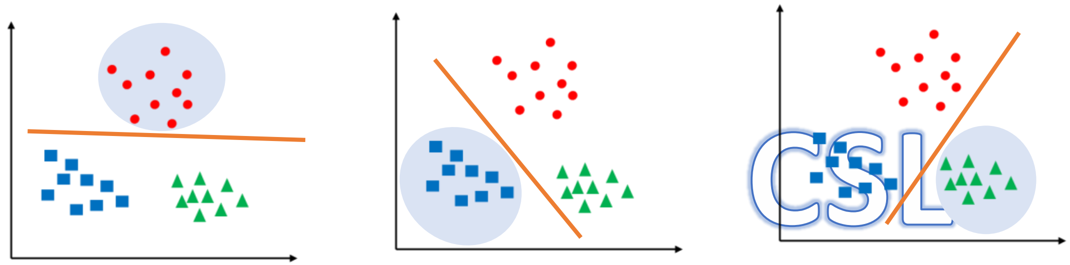


线性分类器

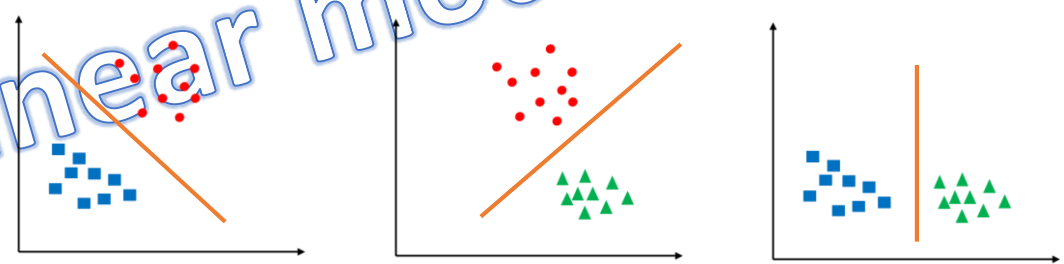
2分类 -> 多分类



One-versus-all （one-versus-rest）

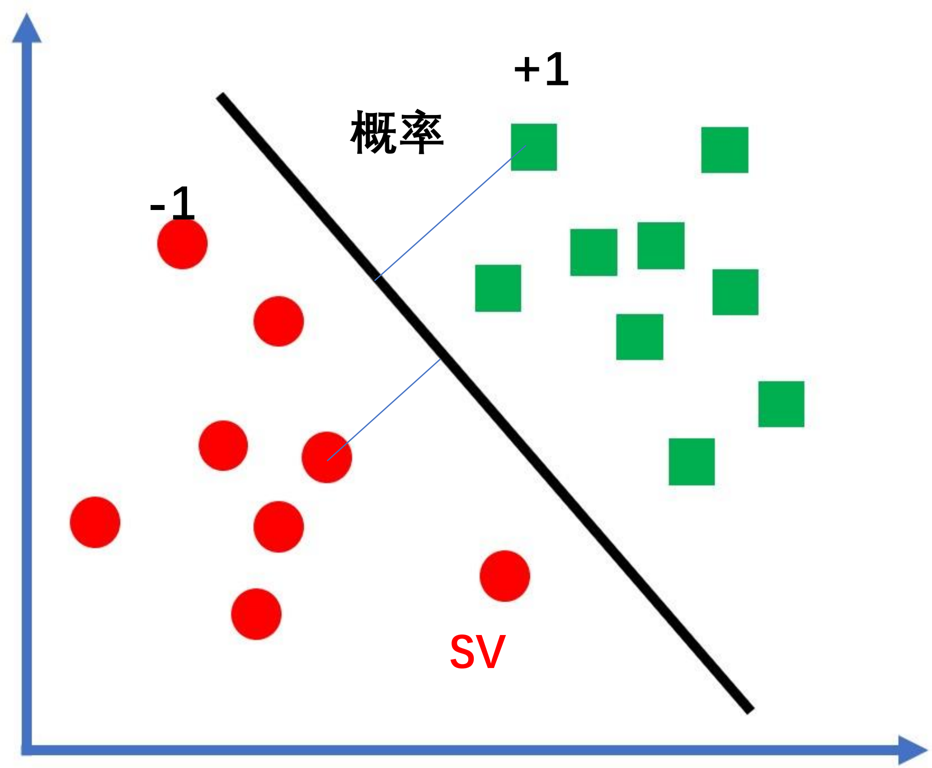


One-versus-one



线性分类器

$$f(x_i) = h(w^T x_i + b)$$



感知机（Perceptron）

$$f(x_i) = sign(w^T x_i + b)$$

逻辑回归（Logistic regression）

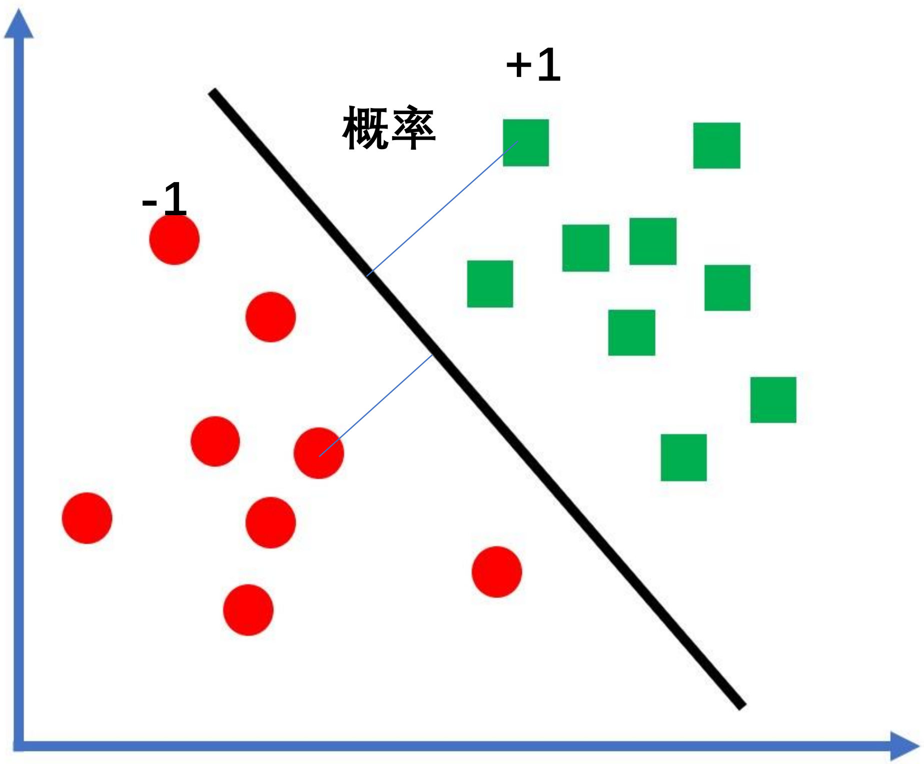
$$f(x_i) = sigmoid(w^T x_i + b)$$

SVM（Support vector machine）

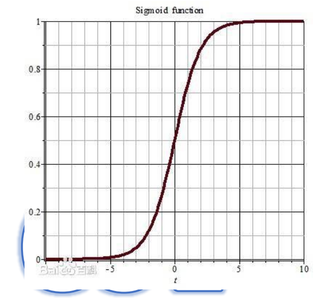
$$f(x_i) = sign(w^T x_i + b)$$

线性分类器　LR

$$f(x_i) = h(w^T x_i + b)$$



$$h(x) = \frac{1}{1 + e^{-x}}$$



(1) 认知神经科学
(2) 物理学（Boltzmann分布）
(3) 经济学（离散选择模型）

$$P(y_i = 1 | x_i) = \frac{1}{1 + e^{-(w^T x_i + b)}}$$

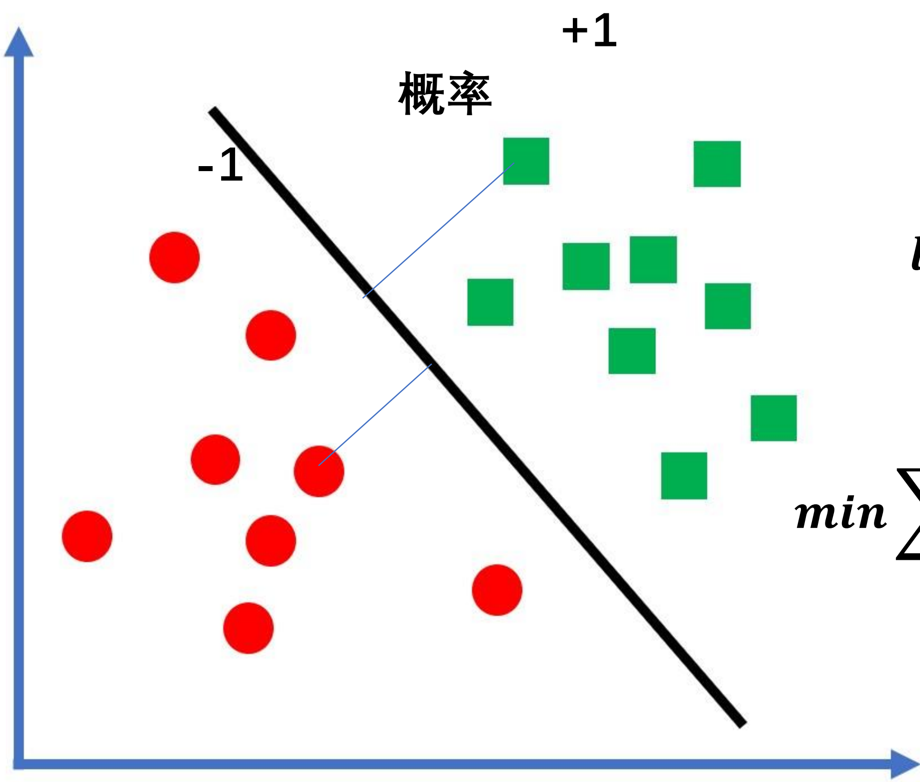
$$P(y_i = -1 | x_i) = \frac{1}{1 + e^{(w^T x_i + b)}}$$

$$\prod_i P(y_i | x_i) = \prod_i \frac{1}{1 + e^{-y_i (w^T x_i + b)}}$$

线性分类器 LR

$$P(y_i = 1|x_i) = \frac{1}{1 + e^{-(w^T x_i + b)}}$$

$$P(y_i = -1|x_i) = \frac{1}{1 + e^{(w^T x_i + b)}}$$



$$\prod_i P(y_i|x_i) = \prod_i \frac{1}{1 + e^{-y_i(w^T x_i + b)}}$$

$$log \prod_i P(y_i|x_i) = log \prod_i \frac{1}{1 + e^{-y_i(w^T x_i + b)}}$$

$$min \sum_i -log\, P(y_i|x_i) = \sum_i log(1 + e^{-y_i(w^T x_i + b)})$$

$$\min_{w,b} log(1 + \mathrm{e}^{-y_i(w^T x_i + b)}) + \lambda||w||^2$$

Logistic Loss: $L(y_i, wx_i + b) = log(1 + e^{-y_i(w^T x_i + b)})$

线性分类器 LR

多项逻辑回归（multi-nominal logistic regression）

Softmax: $\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$

$$P(y_i = 1|x_i) = \frac{1}{1 + e^{-(w^T x_i + b)}}$$

$$P(y_i = -1|x_i) = \frac{e^{-(w^T x_i + b)}}{1 + e^{-(w^T x_i + b)}}$$

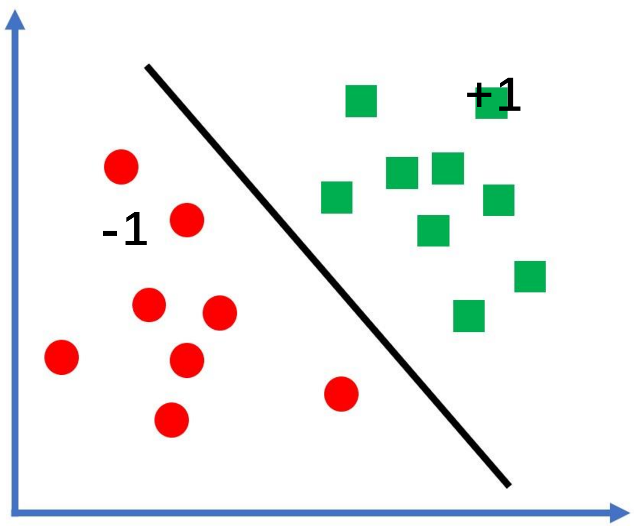
$$P(y_i = k|x_i) = \frac{e^{w_k^T x_i + b_k}}{\sum_k e^{w_k^T x_i + b_k}} \propto e^{w_k^T x_i + b_k}$$

$$log \prod_i \prod_k P(y_i = k|x_i)^{I(y_i = k)} = \sum_i \sum_k I(y_i = k) log \frac{e^{w_k^T x_i + b_k}}{\sum e^{w_k^T x_i + b_k}}$$

Cross entropy Loss: $L = \sum_k I(y_i = k) log \frac{e^{w_k^T x_i + b_k}}{\sum e^{w_k^T x_i + b_k}}$

线性分类器

$$f(x_i) = h(w^T x_i + b)$$



感知机（Perceptron）

$$f(x_i) = sign(w^T x_i + b)$$

逻辑回归（Logistic regression）

$$f(x_i) = sigmoid(w^T x_i + b)$$

SVM（Support vector machine）

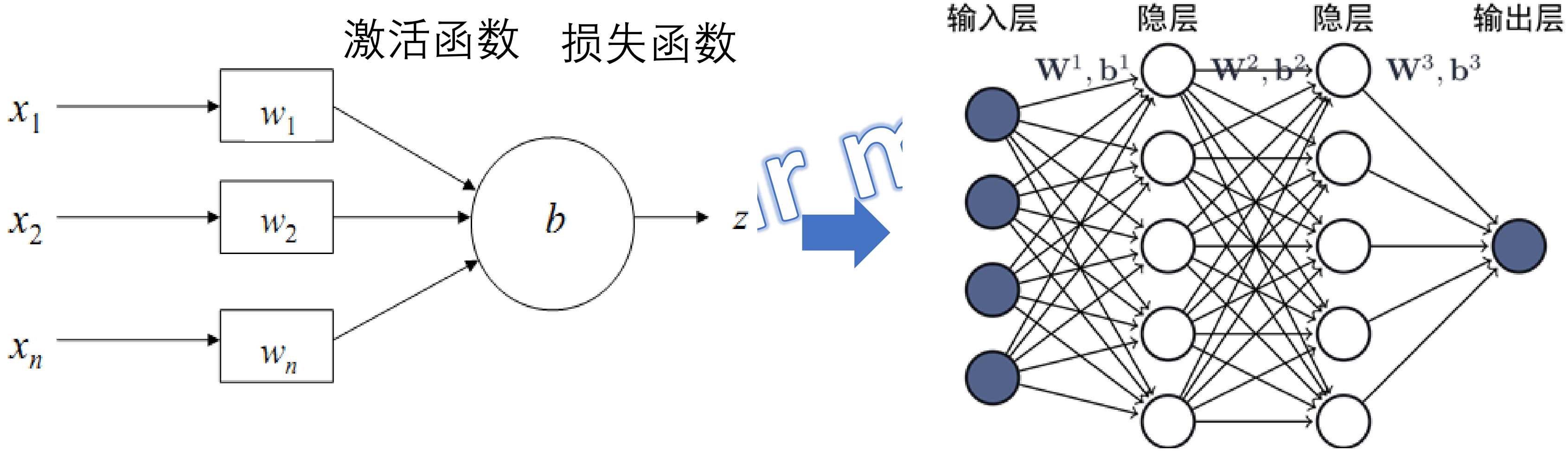
$$f(x_i) = sign(w^T x_i + b)$$

	模型	评价	优化
感知机	$\hat{y}_i = w^T x_i + b$	$L = -\hat{y}_i y_i$	SGD
逻辑回归	$\hat{y}_i = w^T x_i + b$	$L = log(1 + e^{-\hat{y}_i y_i})$	SGD
SVM	$\hat{y}_i = w^T x_i + b$	$L = max(1 - y_i \hat{y}_i, 0)$	SGD

线性分类器

线性分类器 = 浅层神经网络？

	模型	评价	优化
感知机	$\hat{y}_i = w^T x_i + b$	$L = -\hat{y}_i y_i$	SGD
逻辑回归	$\hat{y}_i = w^T x_i + b$	$L = log(1 + e^{-\hat{y}_i y_i})$	SGD
SVM	$\hat{y}_i = w^T x_i + b$	$L = max(1 - y_i \hat{y}_i, 0)$	SGD



由浅入深

总结

机器学习 = 模型 + 评估 + 优化

梯度下降法：

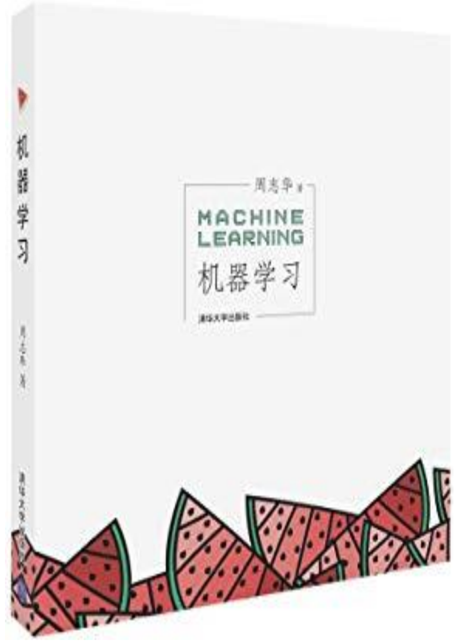
```
while True:
  weights_grad = evaluate_gradient(loss_fun, data, weights)
  weights += - step_size * weights_grad
```

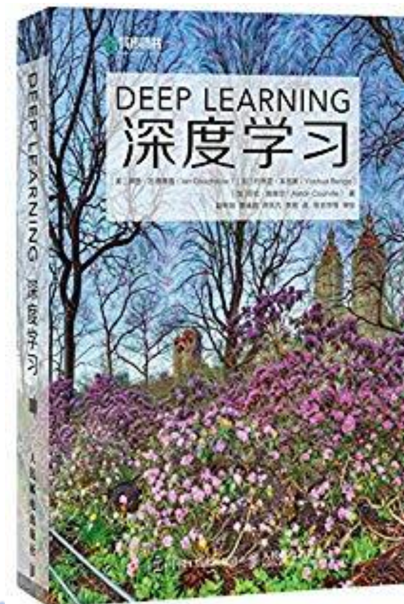

正则化： $\hat{w}^{L_p} = \arg\min_{w \in R^d} \sum_{i=1}^{n} (w^T X_i - Y_i)^2 + \lambda \sum_{j=1}^{d} |w_j|^p$ 　　交叉验证

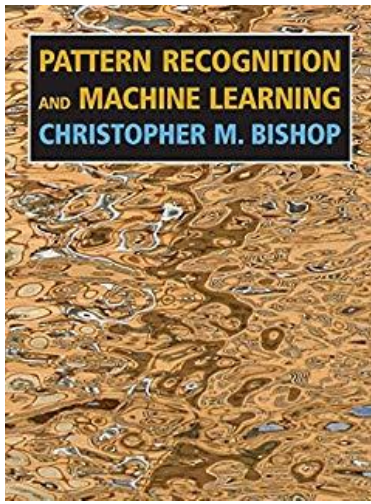
线性分类器：

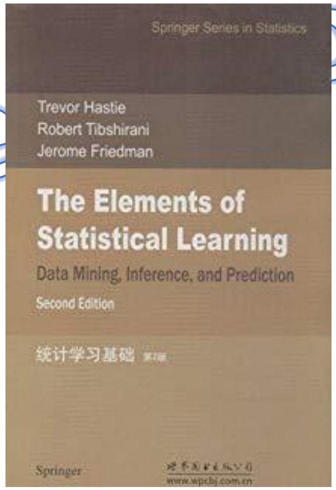
	模型	评价	优化
感知机	$\hat{y}_i = w^T x_i + b$	$L = -\hat{y}_i y_i$	SGD
逻辑回归	$\hat{y}_i = w^T x_i + b$	$L = log(1 + e^{-\hat{y}_i y_i})$	SGD
SVM	$\hat{y}_i = w^T x_i + b$	$L = max(1 - y_i \hat{y}_i, 0)$	SGD

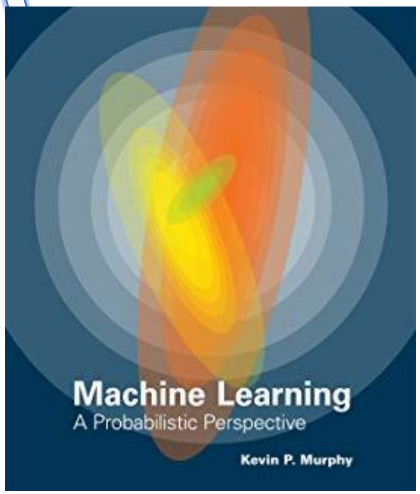

统计学习方法
李航 著
清华大学出版社


周志华 著
MACHINE LEARNING
机器学习
清华大学出版社


DEEP LEARNING
深度学习


PATTERN RECOGNITION
AND MACHINE LEARNING
CHRISTOPHER M. BISHOP

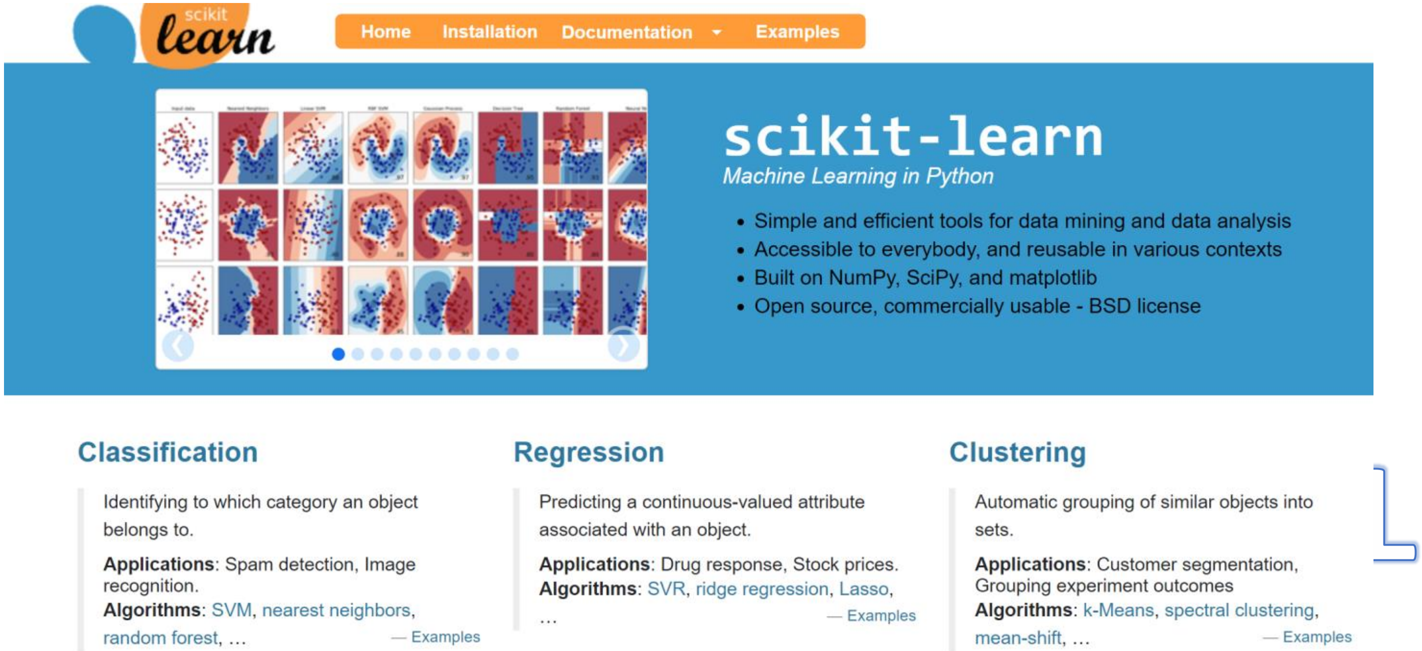

Springer Series in Statistics
Trevor Hastie
Robert Tibshirani
Jerome Friedman
The Elements of Statistical Learning
Data Mining, Inference, and Prediction
Second Edition
统计学习基础
Springer
www.wpcbj.com.cn


Machine Learning
A Probabilistic Perspective
Kevin P. Murphy

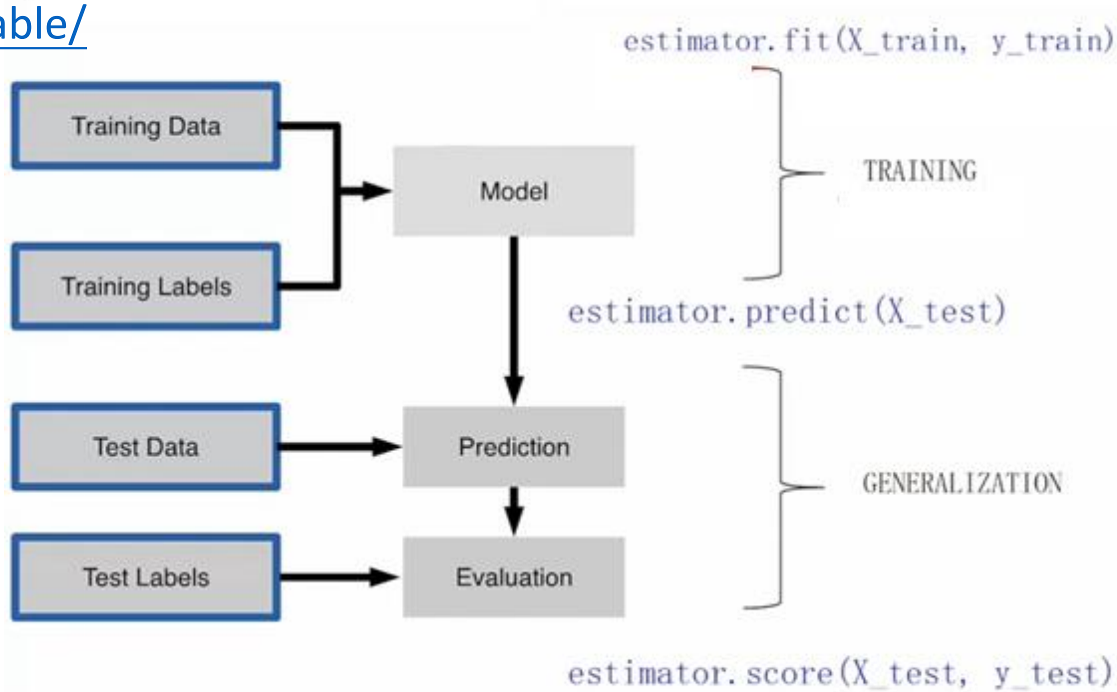
谢 谢！

CSL

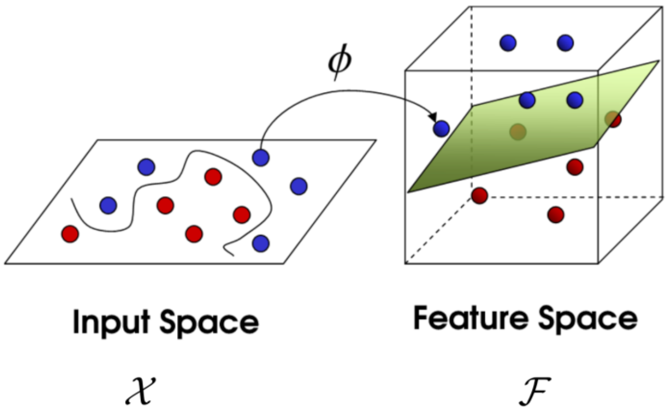
Linear model



https://scikit-learn.org/stable/



核方法 (Kernel trick)



$$\min J(w) = 1/N \sum_{i=1}^{N} L\left(y_i, w^T \phi(x_i)\right) + \lambda \left|\left|w\right|\right|^2$$

表示定理（Representer theory）：

$L(w)$对于w是凸函数，则满足$\min J(w)$的w可以表示为$w = \sum \alpha_j \phi(x_j)$

核函数： $k\left(x_i, x_j\right) = \langle \phi(x_i), \phi(x_j) \rangle$

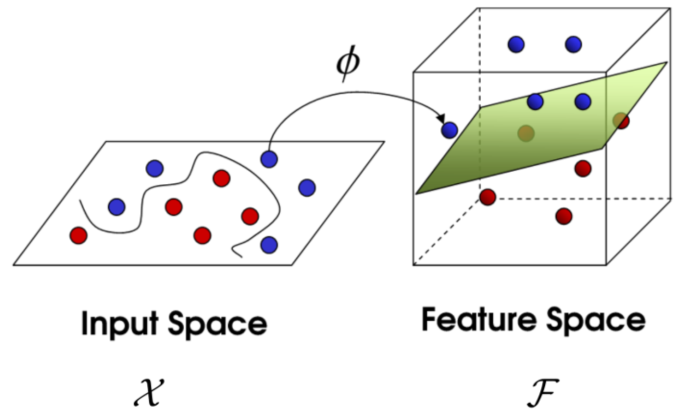
$$\min_\alpha J(w) = 1/N \textstyle\sum_{i=1}^{N} L\left(y_i, \sum \alpha_j < \phi\left(x_j\right), \phi\left(x_i\right) >\right) + \lambda < \sum \alpha_i \phi(x_i), \sum \alpha_i \phi(x_i) >$$

Linear model

$$\min_\alpha J(w) = 1/N \textstyle\sum_{i=1}^{N} L\ \left(y_i, \sum \alpha_j k(x_j, x_i)\ \right)\ \ + \lambda < \sum \alpha_i \phi(x_i), \sum \alpha_i \phi(x_i) >$$

$$\min_\alpha J(w) = 1/N \sum_{i=1}^{N} L\left(y_i, \sum \alpha_j k\left(x_j, x_i\right)\right) + \lambda \alpha^T K a$$

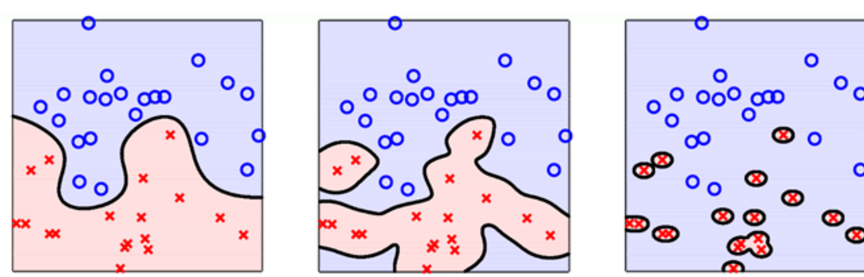
核方法 (Kernel trick)



$$\boldsymbol{min_\alpha\, J(w) = 1/N \sum_{i=1}^{N} L\left(y_i, \sum \alpha_j k(x_j, x_i)\right)} + \lambda \boldsymbol{\alpha^T K \alpha}$$

$$k(x_i, x_j) = \exp\left(-\gamma \left\|x_i - x_j\right\|^2\right) = \exp\left(-\gamma(\left\|x_i\right\|^2 + \left\|x_j\right\|^2 - 2x_i^T x_j)\right)$$

Kernel	Function
Linear	$k(x,x') = \langle x, x' \rangle$
Hom. Polynomial	$k(x,x') = \langle x, x' \rangle^d$
Inho. Polynomial	$k(x,x') = (\langle x, x' \rangle + c)^d$
Gaussian RBF	$k(x,x') = \exp\left(-\frac{\|x-x'\|^2}{2\sigma^2}\right)$
Laplacian	$k(x,x') = \exp\left(-\frac{\|x-x'\|}{\sigma}\right)$
Sigmoid	$k(x,x') = \tanh(\alpha\langle x, x' \rangle + c)$



$\exp(-1\|\mathbf{x}-\mathbf{x}'\|^2)$ $\exp(-10\|\mathbf{x}-\mathbf{x}'\|^2)$ $\exp(-100\|\mathbf{x}-\mathbf{x}'\|^2)$

Lagrange对偶

$$\min_{w,b} \frac{1}{2}||w||^2 \qquad \boldsymbol{s.t.}\ \ \boldsymbol{y_i}(\boldsymbol{w x_i} + \boldsymbol{b}) \geq 1$$

$$L(w,b,\alpha) = \frac{1}{2}||w||^2 + \alpha(1 - y_i(w x_i + b))$$

$$\min_{w,b} \max_{\alpha} L(w,b,\alpha) \ = \max_{\alpha} \min_{w,b} L(w,b,\alpha)$$

$$\nabla_w L(w,b,\alpha) = \mathrm{w} - \sum \alpha_i y_i x_i = 0$$

$$\nabla_b L(w,b,\alpha) = \sum \alpha_i y_i = 0$$

$$\min_{w,b} L(w,b,\alpha) = -\frac{1}{2} \sum\sum \alpha_i \alpha_j y_i y_j < x_i, x_j > + \sum \alpha_i$$

Some models are easy to interpret

Linear/Logistic regression

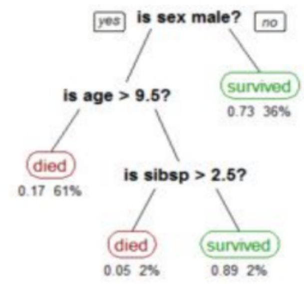
- Weight on each feature
- Know the exact contribution of each feature, negative or positive

$$Y = 3 * X1 - 2 * X2$$

Increasing X1 by 1 unit increases Y by 3 units

Single Decision Tree

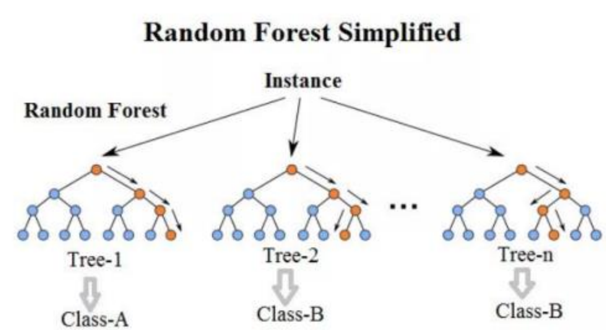
- Easy to understand how a decision was made by reading from top to bottom



Some models are harder to interpret

Ensemble models (random forest, boosting, etc…)

- Hard to understand the role of each feature
- Usually comes with **feature importance**
- Doesn't tell us if feature affects decision positively or negatively

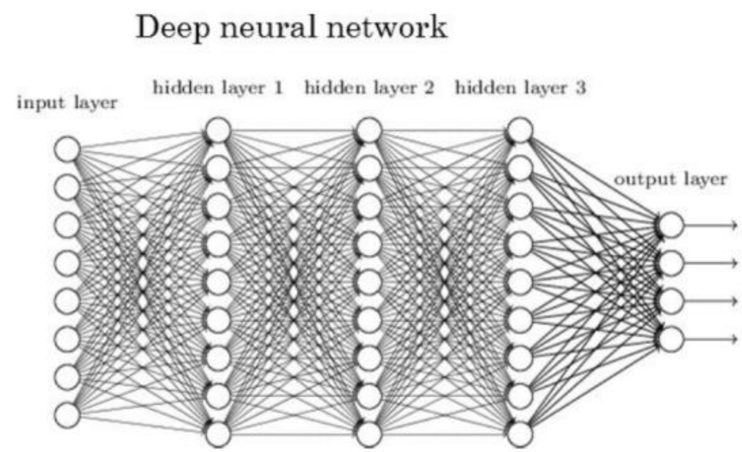


可解释性机器学习

Some are really hard to interpret

Deep Neural Networks

- No straightforward way to relate output to input layer
- “Black-box”



个体的行为选择

- **决策者**（Decision Makers），即做出选择行为的主体；

CSL

- **备选方案**（Alternatives），通常会有多个方案供决策者选择
- **选择方案的属性**（Attributes of Alternatives）。每一种考虑因素称之为一个属性（Attributes），以q_j表示
- **决策准则**（Decision Rules） Model （选择函数）

  基本假设：人类行为是理性的（偏好是完备的并具有传递习性）

Linear model

Utility-based agent

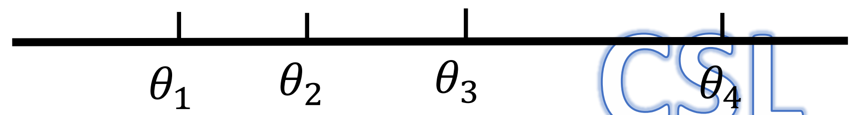
- 定义（效用函数）： p_i是定义在特征向量空间上的线性泛函$v_{p_i} = p_i(q)$。由Riesz表示定理,偏好可以表示为p_i,则有$v_{p_i}(q) =< p_i, q >$.

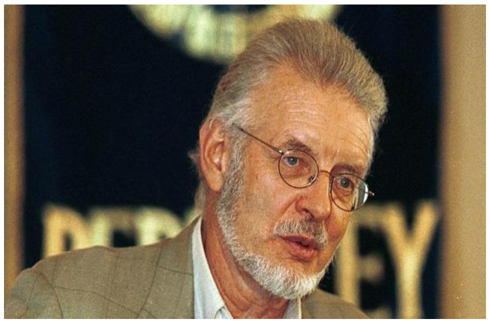
Linear model

- 定义（行为选择）： $\pi_k: S \times A \to R$,其中U为个体i的效用空间 $\pi(a|s) = P(A_t = a|S_t = s)$

离散选择模型

$$u_{k,j} = v_{k,j} + \varepsilon_k = \langle p_k, q_j \rangle + \varepsilon_k$$





Daniel L. McFadden(2000诺奖)

当 ε_i 满足Gumbel 分布（type I extreme value）时，

Utility Maximization

$$\begin{aligned} P_{ni} &= \text{Prob}(V_{ni} + \varepsilon_{ni} > V_{nj} + \varepsilon_{nj} \ \forall j \neq i) \\ &= \text{Prob}(\varepsilon_{nj} < \varepsilon_{ni} + V_{ni} - V_{nj} \ \forall j \neq i). \end{aligned}$$

$$P_{ni} = \int \left(\prod_{j \neq i} e^{-e^{-(\varepsilon_{ni} + V_{ni} - V_{nj})}} \right) e^{-\varepsilon_{ni}} e^{-e^{-\varepsilon_{ni}}} \, d\varepsilon_{ni}. \quad \Rightarrow \quad P_{ni} = \frac{e^{V_{ni}}}{\sum_j e^{V_{nj}}},$$

当只有2种行为时，上述模型退化为binary logit模型.

$$p(Y=1|x)=\exp\big(f_w(x)\big)/\big(1+\exp\big(f_w(x)\big)\big)$$

- Logistic model

$$f_w(\boldsymbol{x}) = w_{k,0} + \sum_{i=1}^{n} w_{k,i}x_i = w_{k,0} + \langle p_{k}, q_{j}\rangle$$

Utility-based agent: $u_{k,j} = w_{k,0} + \langle p_{k}, q_{j}\rangle + \varepsilon_k$

CSL

- Factorization Machine model

$$f_w(\boldsymbol{x}) = w_{k,0} + \sum_{i=1}^{n} w_{k,i}x_i + \sum_{i=1}^{n}\sum_{j=i+1}^{n} \langle v_i, v_j\rangle x_i x_j$$

Utility-based agent: $u_{k,j} = w_{k,0} + \langle p_{k}, q_{j}\rangle + \sum_{i<j}^{n} \langle v_i, v_j\rangle x_i x_j + \varepsilon_k$

Linear model

- 一般化

Utility-based agent: $u_{k,j}(t) = w_{k,0}(t) + h\big(p_k(t), q_j(t)\big) + \varepsilon_k(t)$

$$h\left(p_k, q_j\right) = F_{GBDT}\left(p_k,\ q_j\right) \qquad h\left(p_k, q_j\right) = F_{DNN}\left(p_k,\ q_j\right)$$

- 安装python （Anaconda）
  - 熟悉python的基本数据结构， tuple, list和dict.
  - 熟悉基本numpy操作
- 阅读sklearn中有关LR和SVM的文档， 并尝试使用
- 找到非零x满足1.0+x==1.0?
- 尝试实现GD（SGD） 求解LR
- 尝试实现算法求解LASSO的线性回归问题
- 比较GD和Newton法的收敛速度