CS182 Introduction to Machine Learning

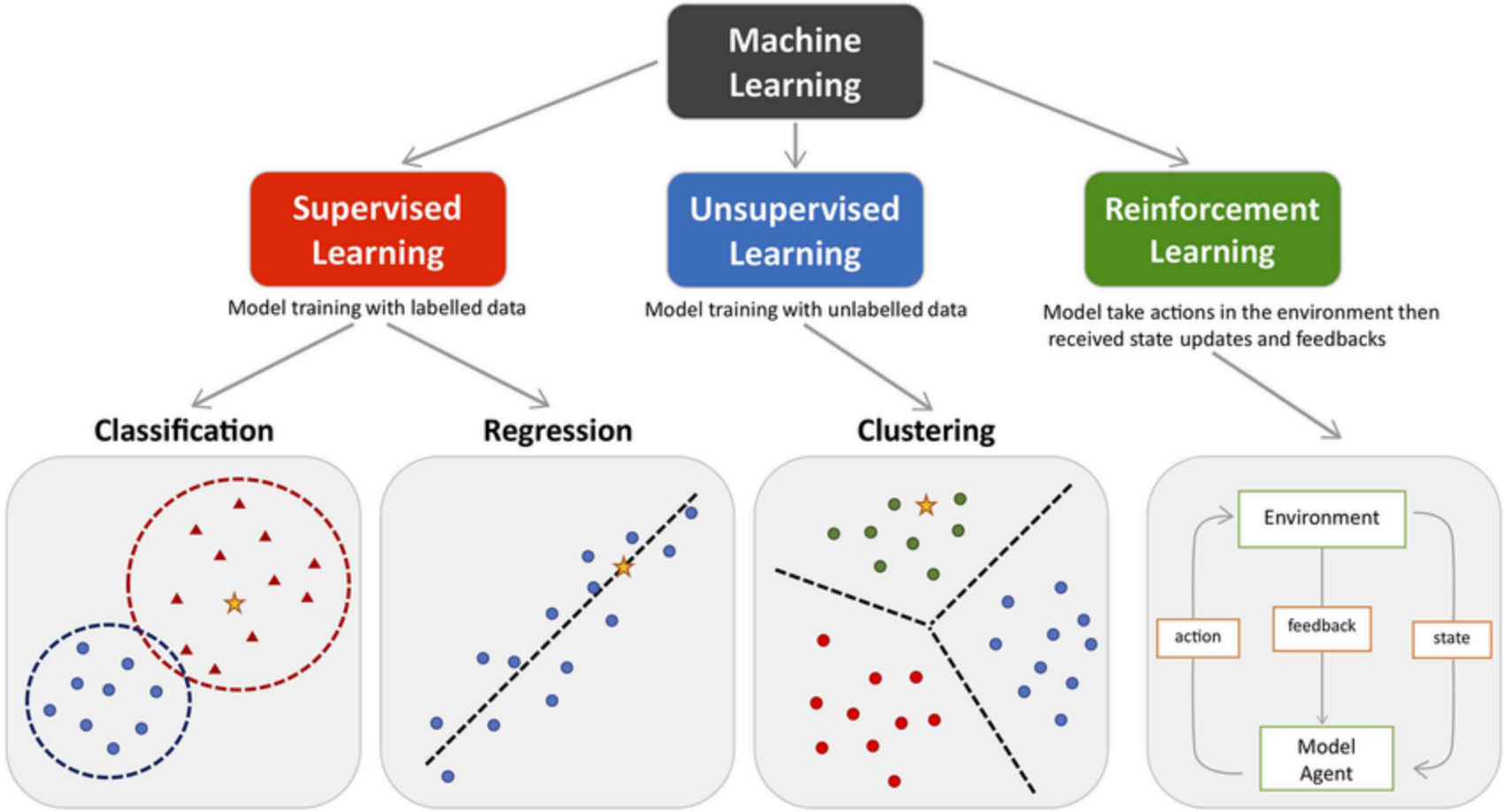
Recitation 12

2025.6.4

Outline

- Decision Trees
- KNN (k-Nearest Neighbors)
- Perceptron
- Linear Regression
- Kernel Methods
- SVM (Support Vector Machines)
- Logistic Regression
- MLE (Maximum Likelihood Estimation)
- Neural Networks & Backpropagation
- Matrix Factorization
- K-Means Clustering
- GMM (Gaussian Mixture Models)
- EM (Expectation-Maximization)
- PCA (Principal Component Analysis)

Types of Machine Learning



Supervised Learning

Classification & Regression

- Decision Tree
- KNN
- Linear Classification
- Perceptron
- SVM
- Logistic Regression

Regression

- Linear Regression

Mixed

- Neural Networks
- Ensemble Learning

Unsupervised Learning

- K-Means
- Gaussian Mixture Models
- PCA

Mathematics methods

- Matrix derivative
- Lagrangian function
- KKT
- MAP & MLE
- Optimization methods
- Matrix Factorization
- ELBO
- EM algorithm

Machine Learning Concepts

- Kernel Method
- Bias & Variance
- Overfitting & Underfitting
- Regularization
- ...

Supervised Learning

Classification & Regression

- Decision Tree
- KNN
- Linear Classification
- Perceptron
- SVM
- Logistic Regression

Regression

- Linear Regression

Mixed

- Neural Networks
- Ensemble Learning

Decision Tree

Entropy 熵

$\log x$若无特殊说明, 默认为$\log_2 x$, $0 \log 0 = 0$.
离散型随机变量$\mathcal{X}$看作是有限的, i.e. $|\mathcal{X}| < +\infty$.
事件x发生的概率为$p(x)$, 则x的信息量为$\log \dfrac{1}{p(x)}$.
离散型随机变量X的熵 (entropy) $H(X)$ 或写作 $H(p)$: 所有事件发生的期望信息量

$$
\begin{aligned}
H(X) &= -\sum_{x \in \mathcal{X}} p(x) \log p(x) \\
&= \sum_{x \in \mathcal{X}} p(x) \log \frac{1}{p(x)} \\
&= \mathbb{E}\left[\log \frac{1}{p(x)}\right]
\end{aligned}
$$

Decision Tree

条件熵 (conditional entropy) $H(Y|X)$:

$$
\begin{aligned}
H(Y|X) &= \sum_{x\in\mathcal{X}} p(x) H(Y|X=x) \\
&= -\sum_{x\in\mathcal{X}} p(x) \sum_{y\in\mathcal{Y}} p(y|x) \log p(y|x) \\
&= \sum_{x\in\mathcal{X}} \sum_{y\in\mathcal{Y}} p(x,y) \log \frac{1}{p(y|x)} \\
&= \mathbb{E}\left[\log \frac{1}{p(y|x)}\right]
\end{aligned}
$$

Decision Tree

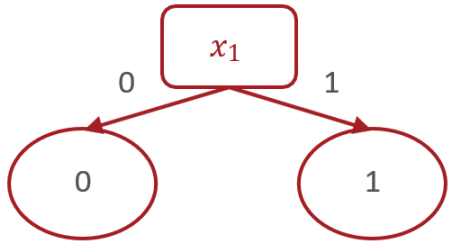
Mutual Information 互信息

$$
\begin{aligned}
I(X;Y) &= \sum_{x,y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)} = D\left(p(x,y) \| p(x)p(y)\right) \\
&= H(X) - H(X|Y) \\
&= H(Y) - H(Y|X)
\end{aligned}
$$

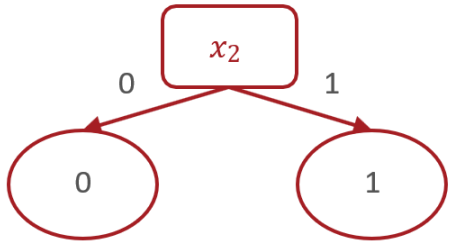
Decision Tree

- 离散属性的决策树

x_1	x_2	y
1	0	0
1	0	0
1	0	1
1	0	1
1	1	1
1	1	1
1	1	1
1	1	1



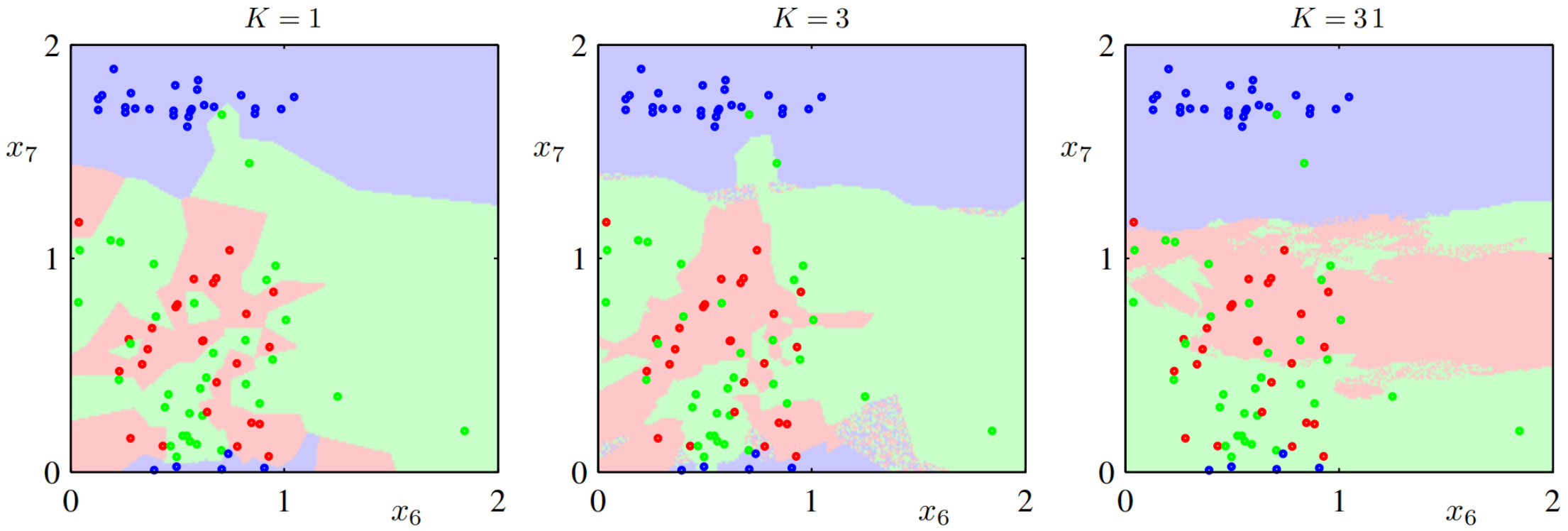
Mutual Information: 0



Mutual Information: $-\frac{2}{8}\log_2\frac{2}{8}-\frac{6}{8}\log_2\frac{6}{8}-\frac{1}{2}(1)-\frac{1}{2}(0)\approx 0.31$

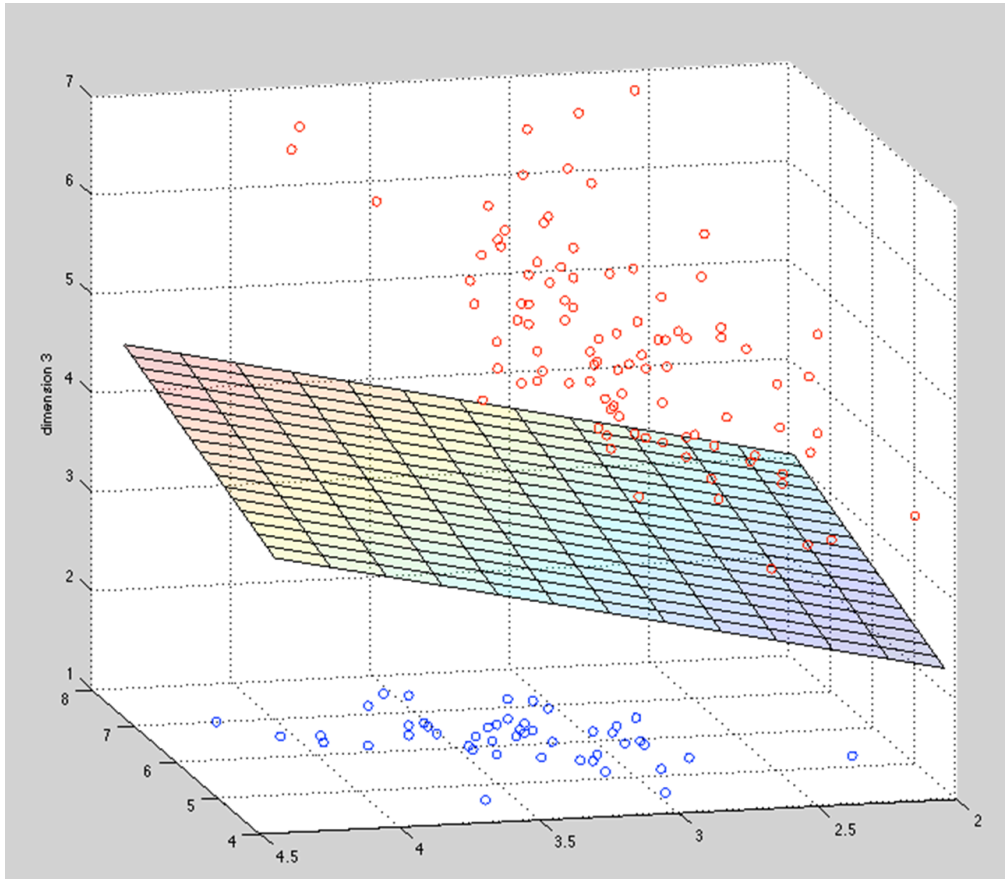
> 按互信息高的方式划分

KNN (K-Nearest Neighbors)



Linear Classification

$$\hat{y} = \text{sign}(\mathbf{w}^\top \mathbf{x} + b) = \begin{cases} 1 & \text{if } \mathbf{w}^\top \mathbf{x} + b \geq 0 \\ -1 & \text{otherwise} \end{cases}$$



Perceptron

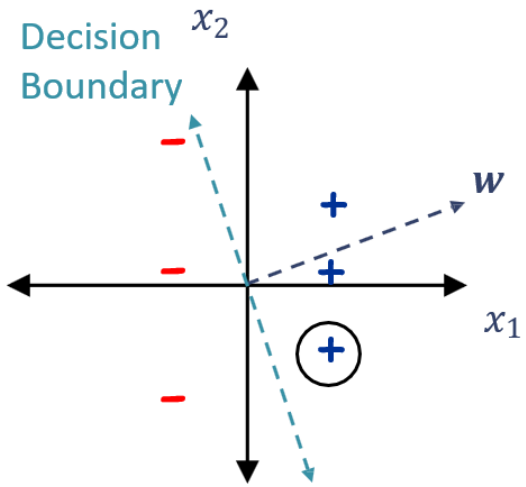
update rules

Perceptron Algorithm: (without the intercept term)

- Set t=1, start with all-zeroes weight vector w_1.
- Given example x, predict positive iff $w_t \cdot x \geq 0$.
- On a mistake, update as follows:
  - Mistake on positive, update $w_{t+1} \leftarrow w_t + x$
  - Mistake on negative, update $w_{t+1} \leftarrow w_t - x$

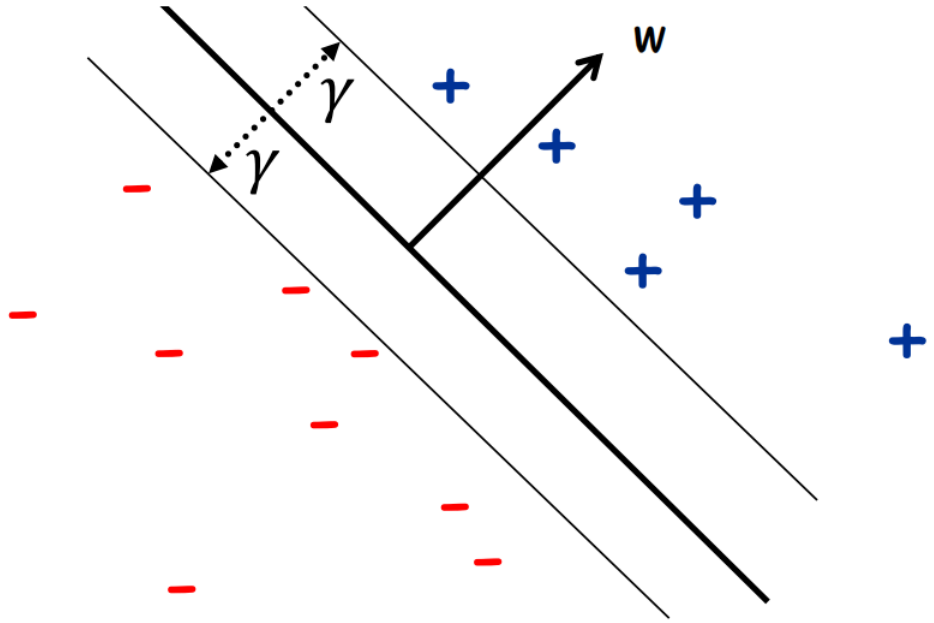
x_1	x_2	$\hat{y}$	y	Mistake?
-1	2	$+$	$-$	Yes
1	0	$+$	$+$	No
1	1	$-$	$+$	Yes
-1	0	$-$	$-$	No
-1	-2	$+$	$-$	Yes
1	-1	$+$	$+$	No

$$w = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$$



Support Vector Machine(SVM) 支持向量机

> Max Margin Classifier
>
> Margin γ: **Support Vector** 到 Hyperplane $\mathcal{H}$ 的距离
>
> $\mathcal{H} = \{\mathbf{x} | \mathbf{w}^\top \mathbf{x} + b = 0\}$



SVM 线性可分

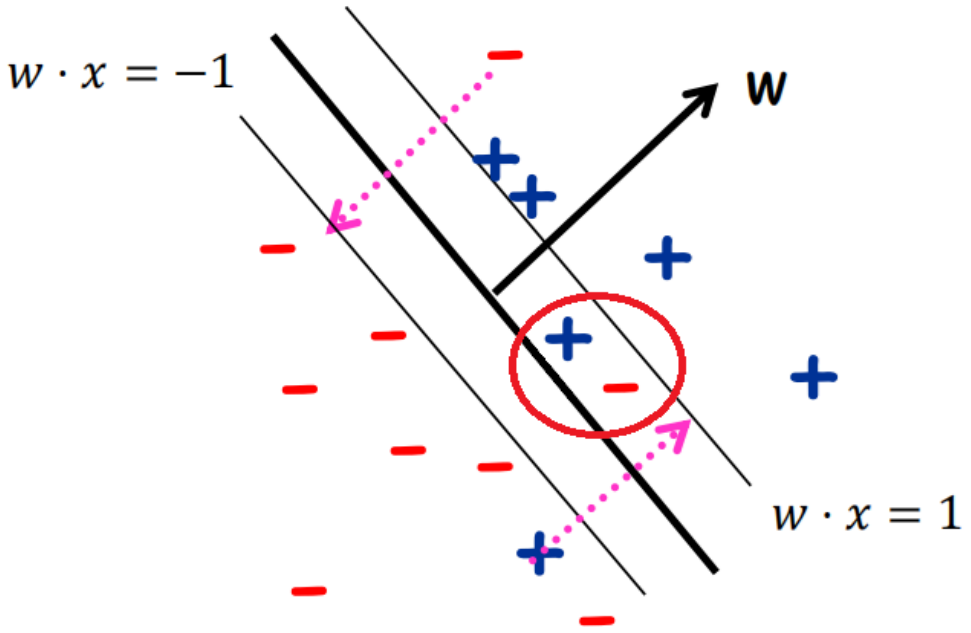
$$
\begin{aligned}
\max_{w,\gamma} \quad & \gamma \\
\text{subject to} \quad & \|w\| = 1 \\
& y_i(x_i \cdot w + b) \geq \gamma,\ \forall i \in \{1, 2, \cdots, n\} \\
& \Downarrow \\
\min_{w'} \quad & \|w'\|^2 \\
\text{subject to} \quad & y_i(x_i \cdot w' + b') \geq 1,\ \forall i \in \{1, 2, \cdots, n\}
\end{aligned}
$$

SVM 线性不可分

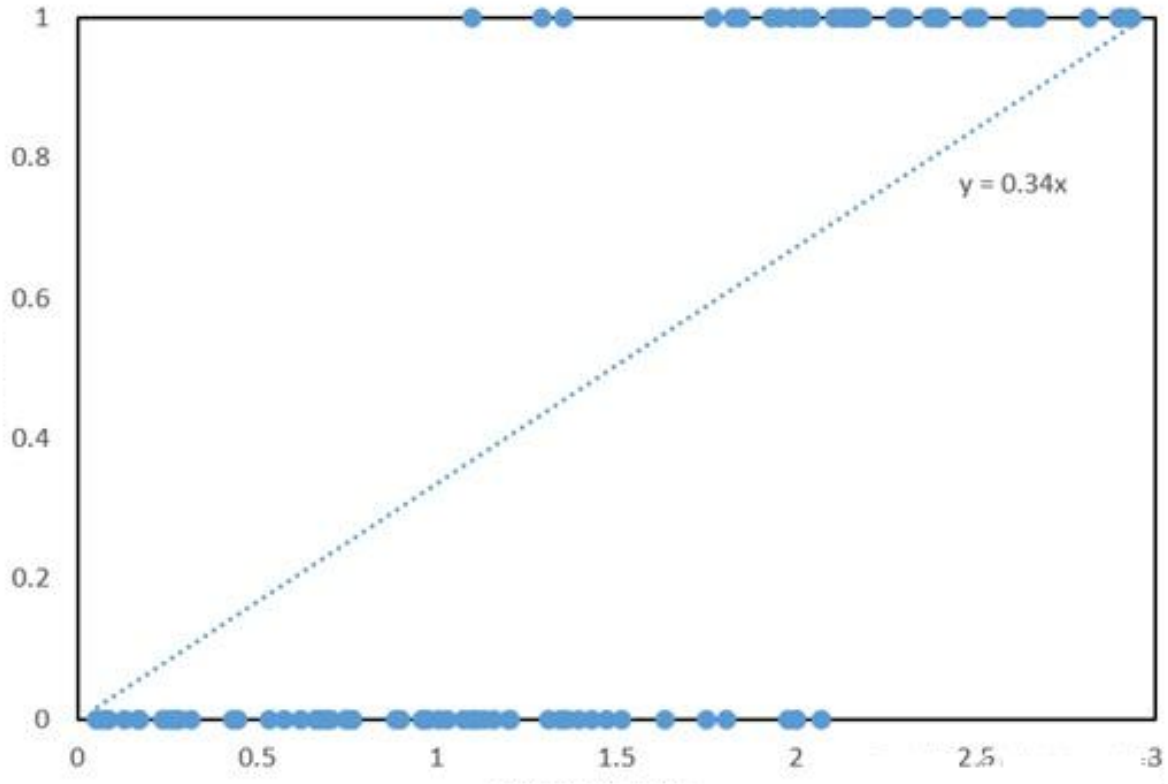
$$\min_{w,\xi} \quad \|w\|^2 + \lambda \sum_{i=1}^{n} \xi_i$$

$$\text{subject to} \quad y_i(x_i \cdot w + b) \geq 1 - \xi_i, \ \forall i \in \{1, 2, \cdots, n\}$$

$$\xi_i \geq 0, \ \forall i \in \{1, 2, \cdots, n\}$$

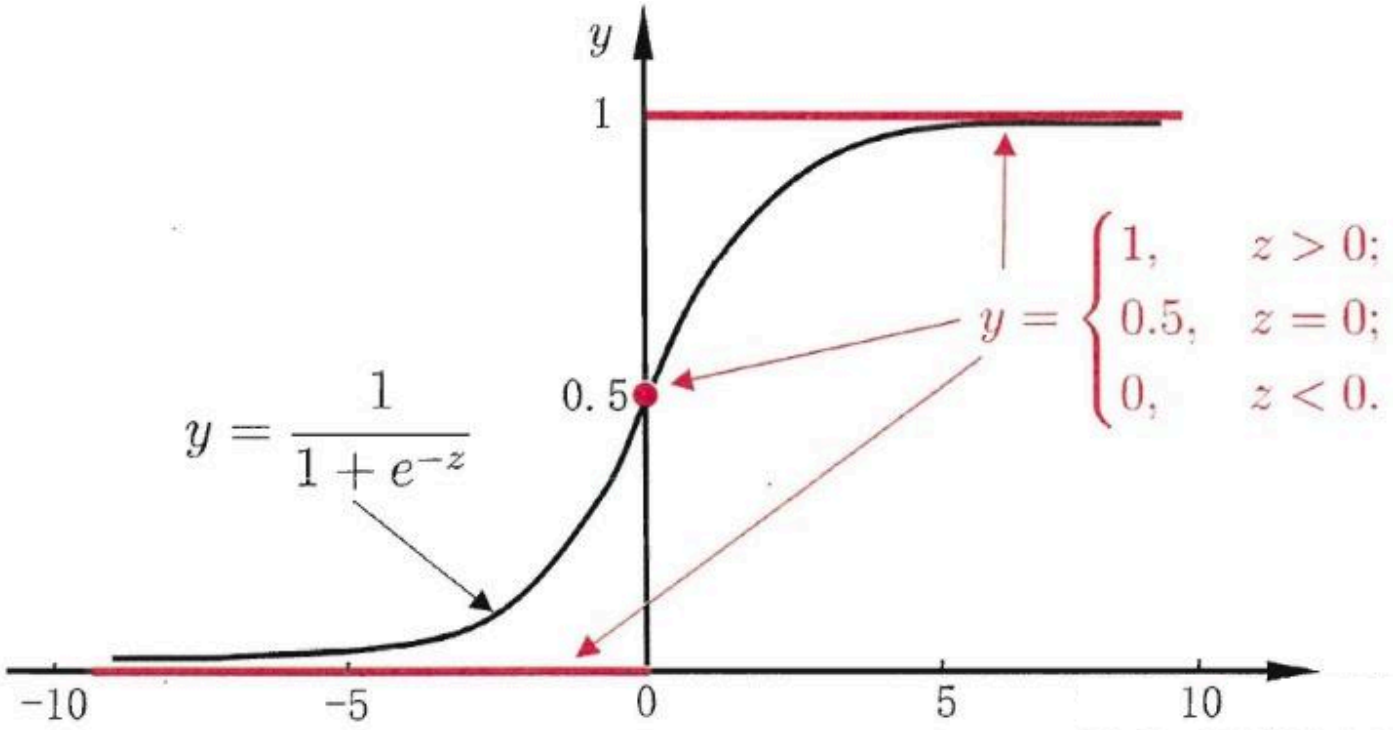


Logistic Regression



> https://zhuanlan.zhihu.com/p/139122386

Sigmoid function



$$p(y_i = 1|x_i;\theta) = \sigma(\theta^\top x_i) = \frac{1}{1+e^{-\theta^\top x_i}}$$

$$p(y_i = 0|x_i;\theta) = 1 - \sigma(\theta^\top x_i) = \frac{e^{-\theta^\top x_i}}{1+e^{-\theta^\top x_i}}$$

Logistic Regression

$$p(y_i = 1|x_i;\theta) = \sigma(\theta^\top x_i) = \frac{1}{1+e^{-\theta^\top x_i}}$$

$$p(y_i = 0|x_i;\theta) = 1 - \sigma(\theta^\top x_i) = \frac{e^{-\theta^\top x_i}}{1+e^{-\theta^\top x_i}}$$

$$\Rightarrow p(y_i|x_i;\theta) = \left[\sigma(\theta^\top x_i)\right]^{y_i} \cdot \left[1-\sigma(\theta^\top x_i)\right]^{1-y_i}$$

$$\hat{\theta} = \arg\max_\theta \prod_{i=1}^{n} p(y_i|x_i;\theta)$$

$$= \arg\max_\theta \prod_{i=1}^{n} \left[\sigma(\theta^\top x_i)\right]^{y_i} \cdot \left[1-\sigma(\theta^\top x_i)\right]^{1-y_i}$$

$$= \arg\max_\theta \sum_{i=1}^{n} y_i \log\left[\sigma(\theta^\top x_i)\right] + (1-y_i)\log\left[1-\sigma(\theta^\top x_i)\right]$$

Linear Regression

Sample points $\{(x_i, y_i)\}_{i=1}^n$.
Linear: 关于参数是线性的.

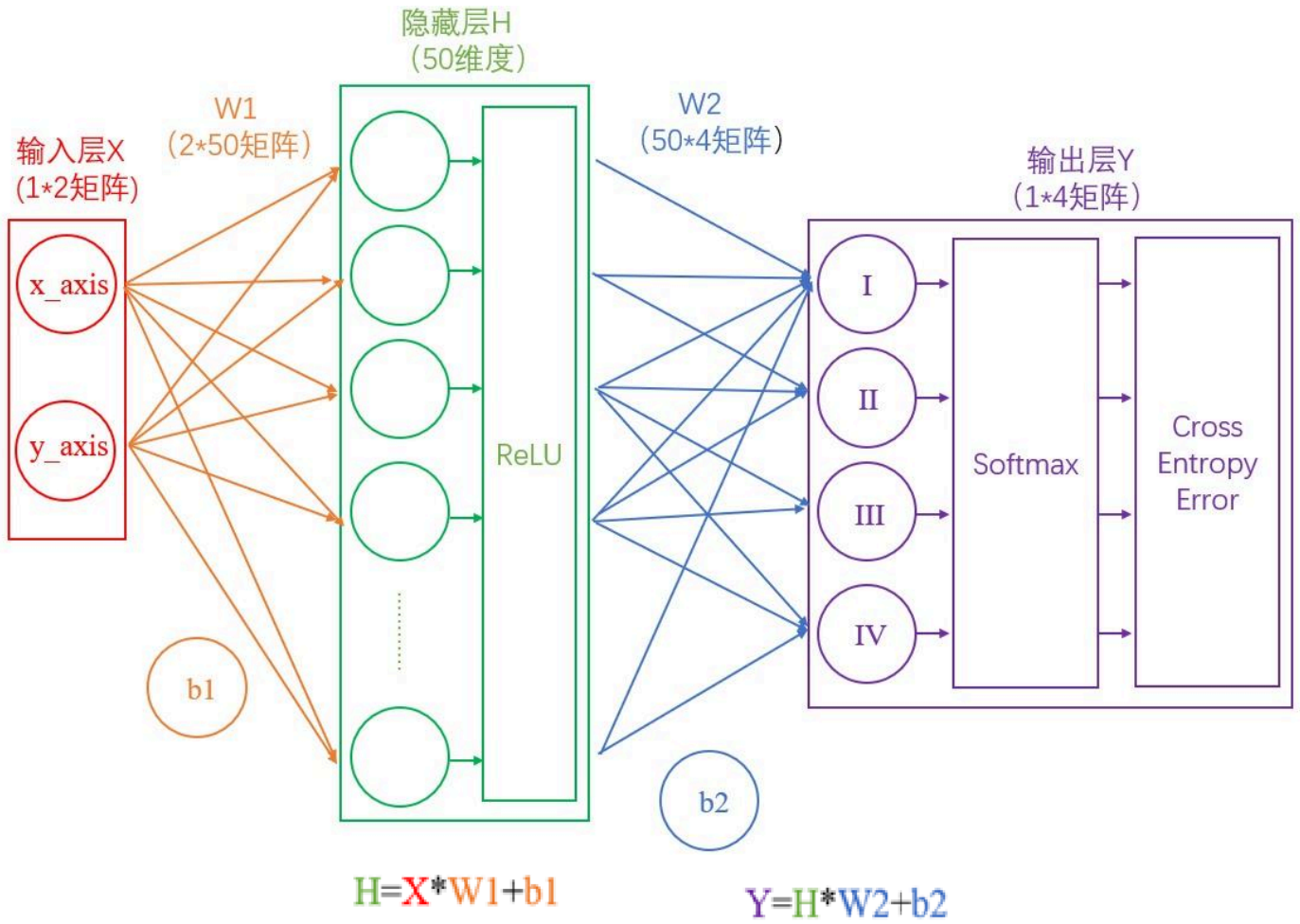
$$\hat{y} = w^\top x$$

$$\min_w \sum_{i=1}^n \left(y_i - \hat{y}_i\right)^2$$

$$\min_w \sum_{i=1}^n \|\mathbf{y} - Xw\|^2$$

> Least Square, Lasso Regression, Ridge Regression, ...

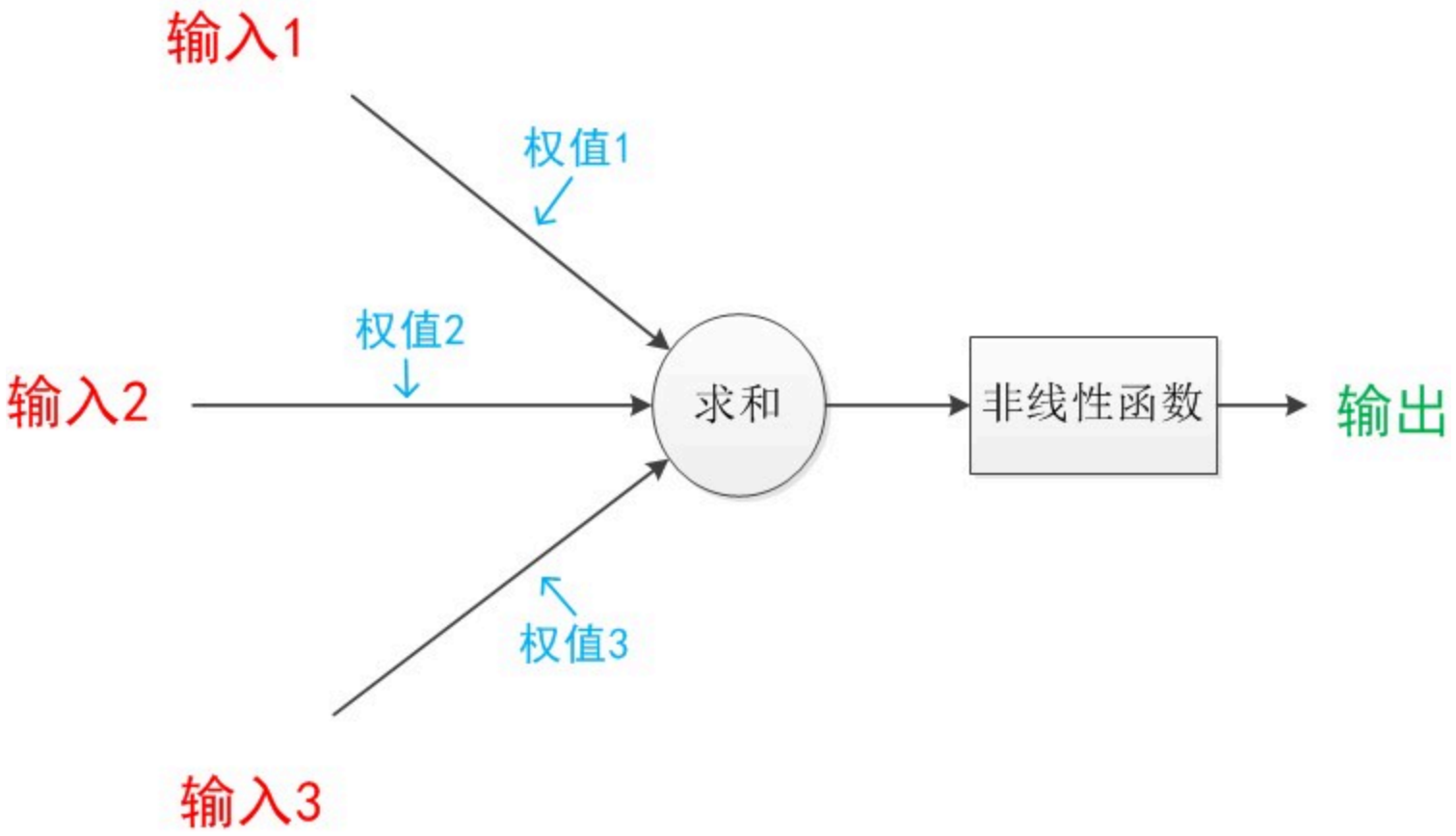
Neural Networks



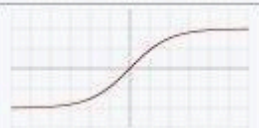
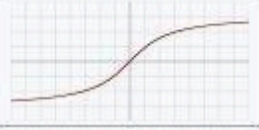
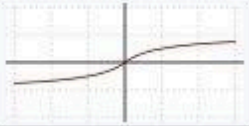
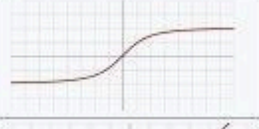
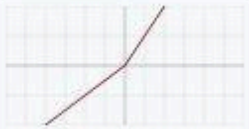
have a play! https://playground.tensorflow.org/

Neural, Activation Functions(激活函数)

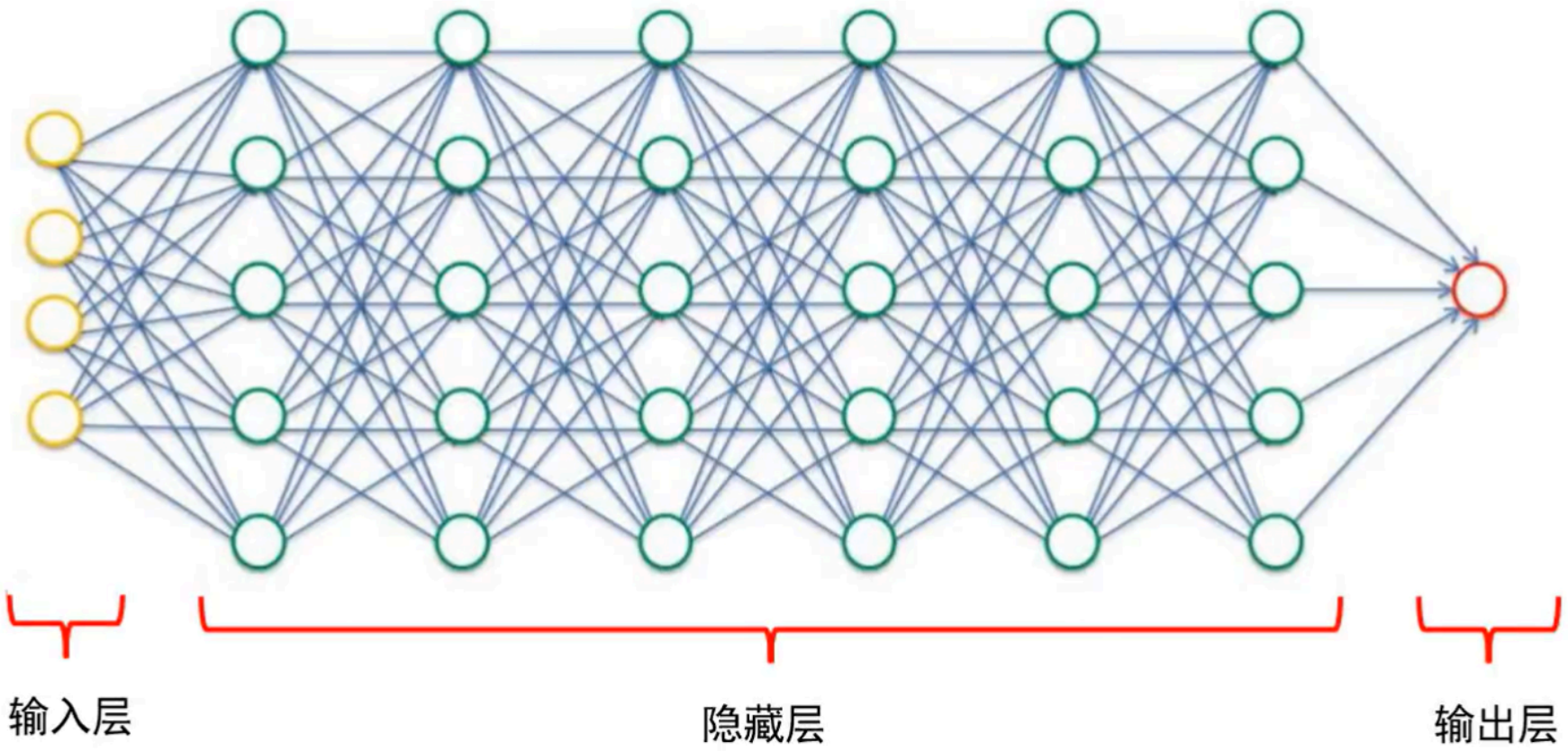
> 没有激活函数, 不管多少层MLP都可以用一层替代



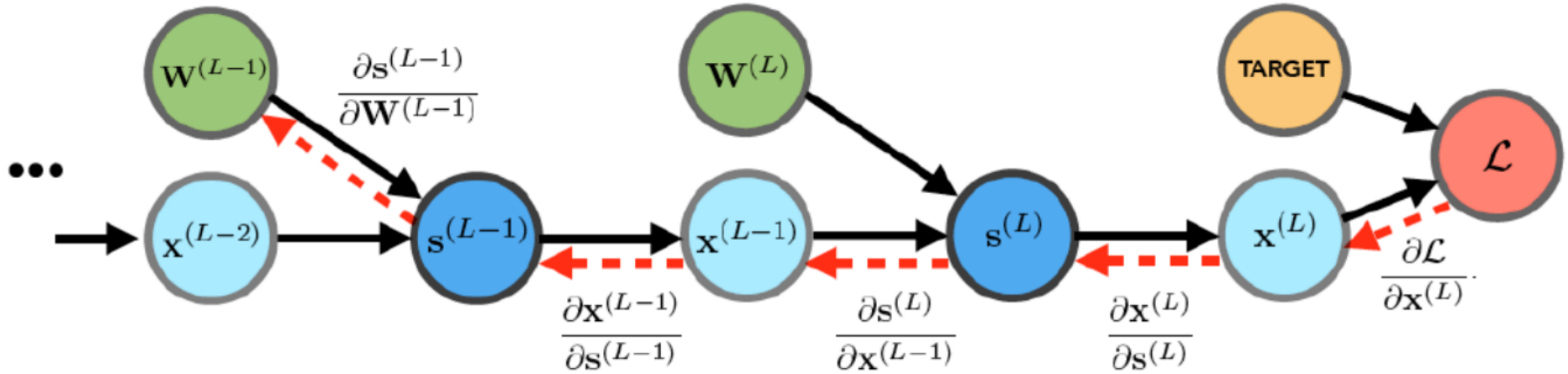
Activation Functions

Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Logistic (a.k.a. Sigmoid or Soft step)		$f(x) = \sigma(x) = \frac{1}{1+e^{-x}}$ [1]
TanH		$f(x) = \tanh(x) = \frac{(e^{x} - e^{-x})}{(e^{x} + e^{-x})}$
ArcTan		$f(x) = \tan^{-1}(x)$
Softsign[9][10]		$f(x) = \frac{x}{1+\lvert x\rvert}$
Inverse square root unit (ISRU)[11]		$f(x) = \frac{x}{\sqrt{1+\alpha x^2}}$
Rectified linear unit (ReLU)[12]		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$
Leaky rectified linear unit (Leaky ReLU)[13]		$f(x) = \begin{cases} 0.01x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$

Multi-layer Perceptron(MLP) 多层感知机 / 全连接层



BackPropagation(BP) 反向传播



$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(L)}} = \frac{\partial \mathcal{L}}{\partial \mathbf{x}^{(L)}} \frac{\partial \mathbf{x}^{(L)}}{\partial \mathbf{s}^{(L)}} \frac{\partial \mathbf{s}^{(L)}}{\partial \mathbf{x}^{(L-1)}} \frac{\partial \mathbf{x}^{(L-1)}}{\partial \mathbf{s}^{(L-1)}} \frac{\partial \mathbf{s}^{(L-1)}}{\partial \mathbf{W}^{(L-1)}}$$

BackPropagation

- Chain Rule 矩阵求导链式法则
  注意将矩阵的维度对上

$$\frac{\partial z}{\partial \mathbf{x}} = \left(\frac{\partial \mathbf{y}}{\partial \mathbf{x}}\right)^{\top} \frac{\partial z}{\partial \mathbf{y}}$$

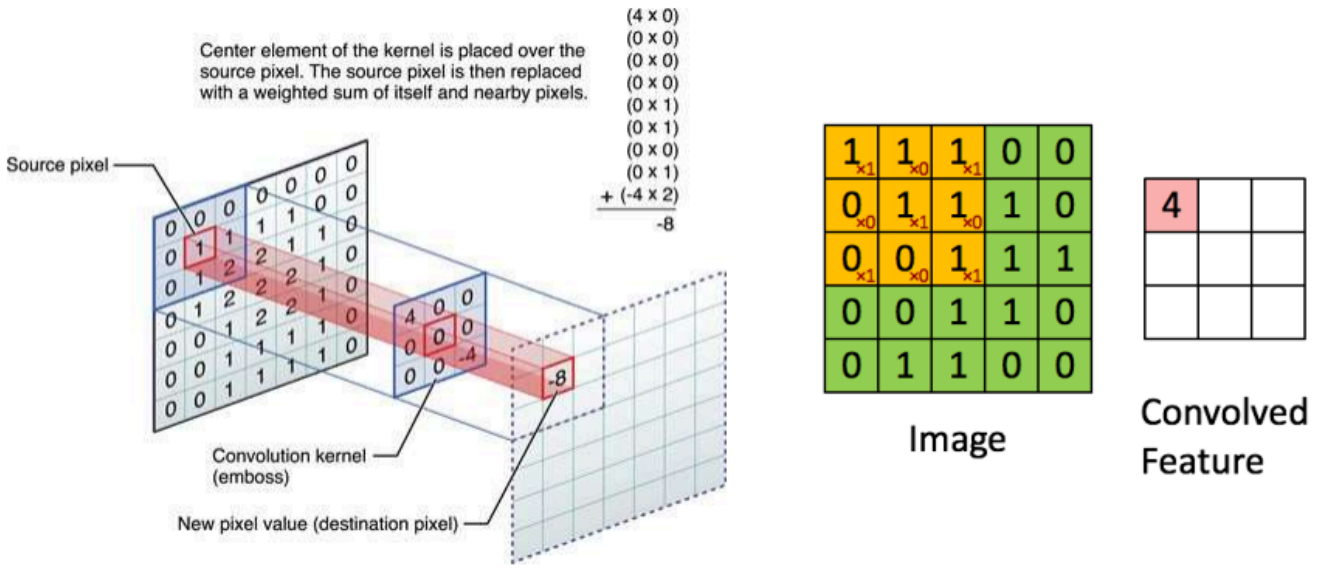
$$\frac{\partial z}{\partial \mathbf{y}_1} = \left(\frac{\partial \mathbf{y}_n}{\partial \mathbf{y}_{n-1}} \frac{\partial \mathbf{y}_{n-1}}{\partial \mathbf{y}_{n-2}} \cdots \frac{\partial \mathbf{y}_2}{\partial \mathbf{y}_1}\right)^{\top} \frac{\partial z}{\partial \mathbf{y}_n}$$

> https://www.cnblogs.com/yifanrensheng/p/12639539.html

2D Convolution

If A and B are two 2-D arrays, then:

$$(A * B)_{ij} = \sum_s \sum_t A_{st} B_{i-s,j-t}.$$



卷积核反转一下就是做一个相关, 因为卷积核的参数是可训练的, 所以在CNN里具体是做的卷积还是做的相关其实并不重要. 实际上torch就是用correlation算的

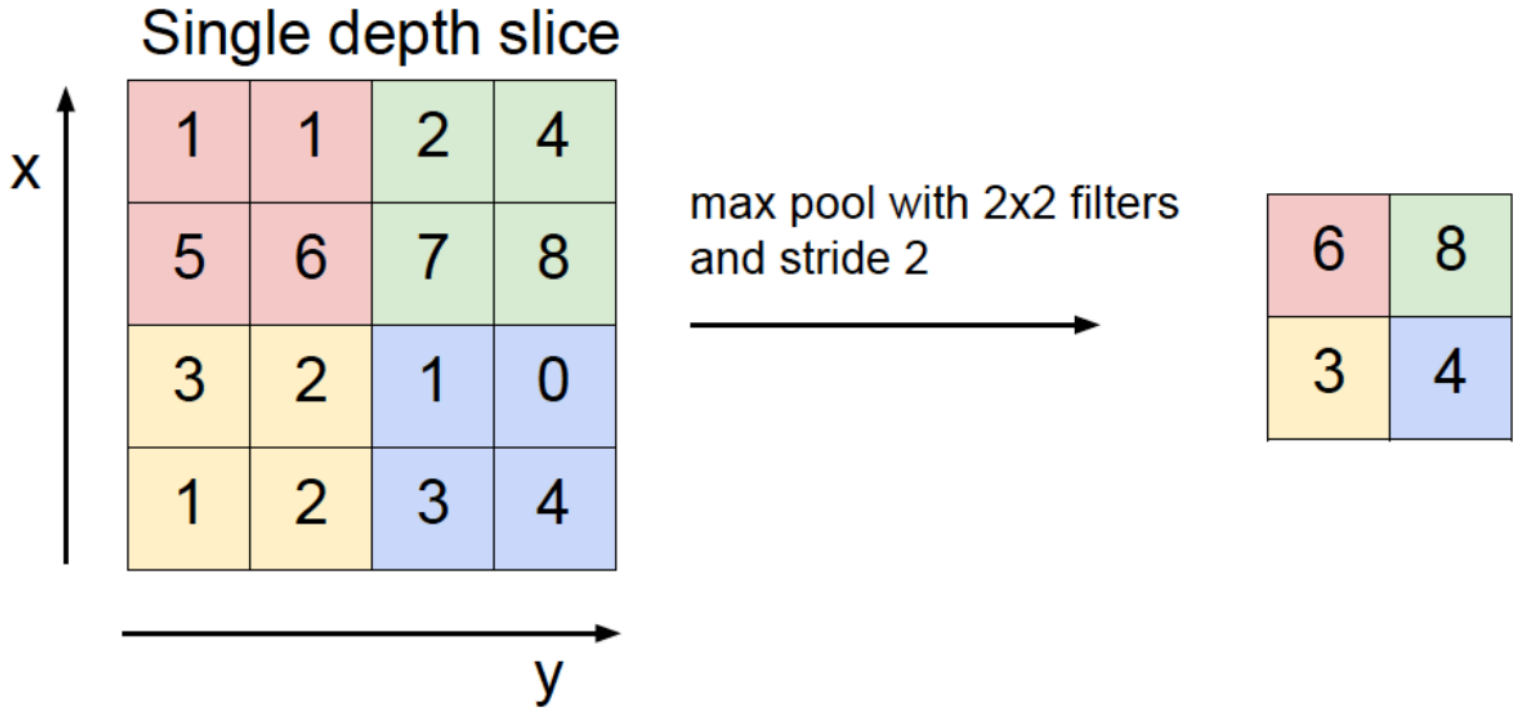
https://pytorch.org/docs/stable/generated/torch.nn.Conv2d.html#torch.nn.Conv2d

Dimension of convolution operation 卷积操作的维度

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters K,
  - their spatial extent F,
  - the stride S,
  - the amount of zero padding P.
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$ (i.e. width and height are computed equally by symmetry)
  - $D_2 = K$
- With parameter sharing, it introduces $F \cdot F \cdot D_1$ weights per filter, for a total of $(F \cdot F \cdot D_1) \cdot K$ weights and K biases.
- In the output volume, the d-th depth slice (of size $W_2 \times H_2$) is the result of performing a valid convolution of the d-th filter over the input volume with a stride of S, and then offset by d-th bias.

一个$W_1 * H_1 * D_1$的图像, 经过K个$F * F * D_1$的卷积核, padding的大小为P, 步长为S
最终得到一个$W_2 * H_2 * D_2$的特征图

Upsampling 上采样 & Downsampling 下采样



- upsample: 时域 / 频域 补零
- downsample: max pooling / average pooling / ...

Dimension of pooling operation 池化操作的维度

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires three hyperparameters:
  - their spatial extent F,
  - the stride S,
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F)/S + 1$
  - $H_2 = (H_1 - F)/S + 1$
  - $D_2 = D_1$
- Introduces zero parameters since it computes a fixed function of the input
- Note that it is not common to use zero-padding for Pooling layers

> pooling操作只改变图像的大小, 不改变特征的维度
> 一个$W_1 * H_1 * D_1$的图像, 经过**感受野**$F * F$池化, 步长为S
> 最终得到一个$W_2 * H_2 * D_1$的特征图

Ensemble Learning (集成学习)

- Boosting
- Bagging(Boostrap AGGregretion)

Boosting: 串行：基学习器按顺序训练，后一个学习器专注于修正前一个模型的错误, 目标是降低偏差(e.g. AdaBoost, XGBoost)
Bagging: 并行：每棵基学习器在相互独立的数据子集上同时训练，训练过程互不影响, 目标是降低方差(e.g. Random Forest)

> Key idea: 把许多单个模型组合成一个整体更强的模型

Boosting

- given training set $(x_1, y_1), \ldots, (x_m, y_m)$
- $y_i \in \{-1, +1\}$ correct label of instance $x_i \in X$
- for $t = 1, \ldots, T$:
  - construct distribution D_t on $\{1, \ldots, m\}$
  - find weak classifier ("rule of thumb")

$$h_t : X \to \{-1, +1\}$$

with error ϵ_t on D_t:

$$\epsilon_t = \Pr_{i \sim D_t}[h_t(x_i) \neq y_i]$$

- output final/combined classifier H_{final}

AdaBoost(Adaptive Boosting)

- constructing D_t:
  - $D_1(i) = 1/m$
  - given D_t and h_t:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } y_i = h_t(x_i) \\ e^{\alpha_t} & \text{if } y_i \neq h_t(x_i) \end{cases}$$

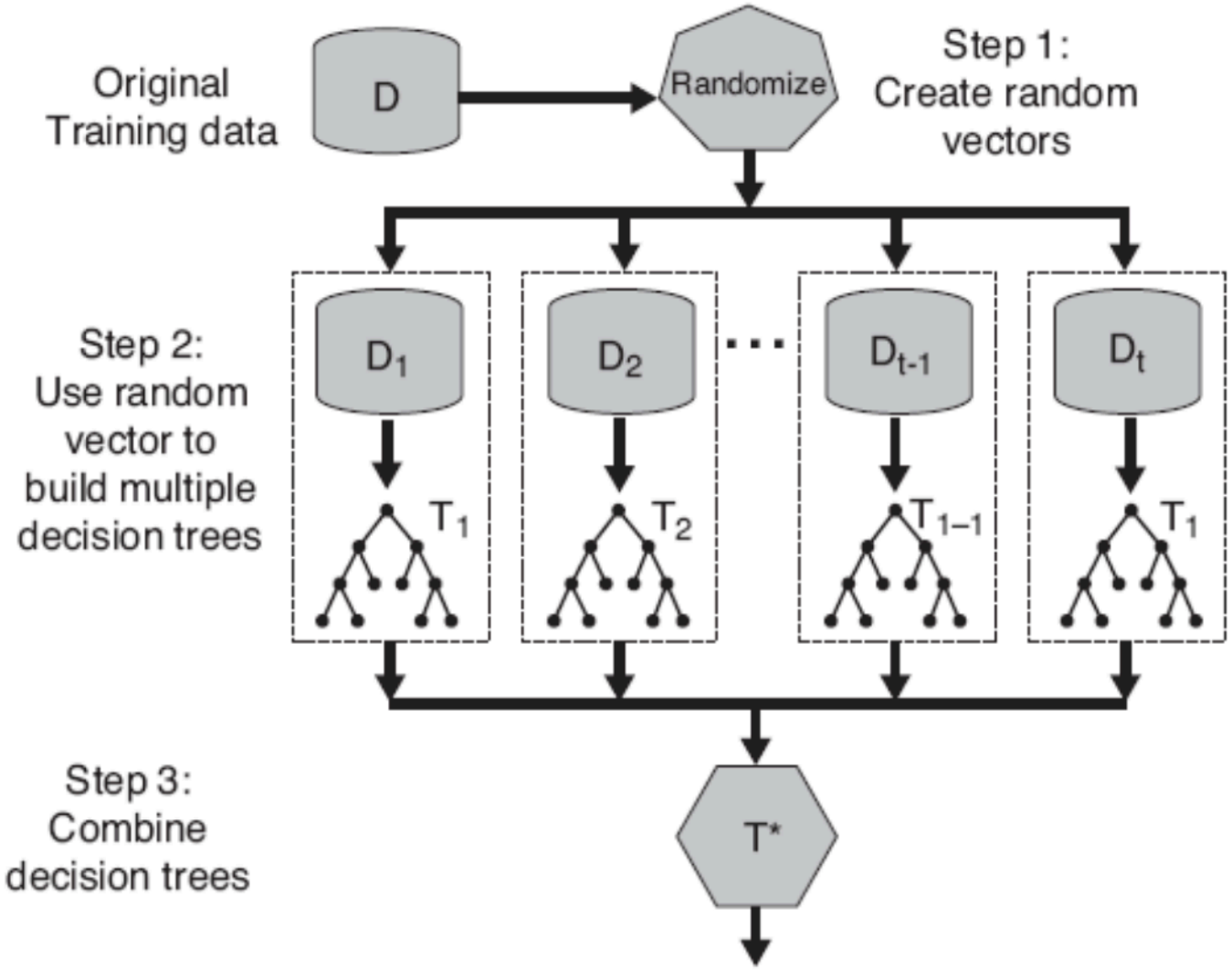
$$= \frac{D_t(i)}{Z_t} \exp(-\alpha_t \, y_i \, h_t(x_i))$$

where Z_t = normalization factor

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right) > 0$$

- final classifier:
  - $H_{\text{final}}(x) = \text{sign} \left(\sum_t \alpha_t h_t(x) \right)$

Random Forest



Unsupervised Learning

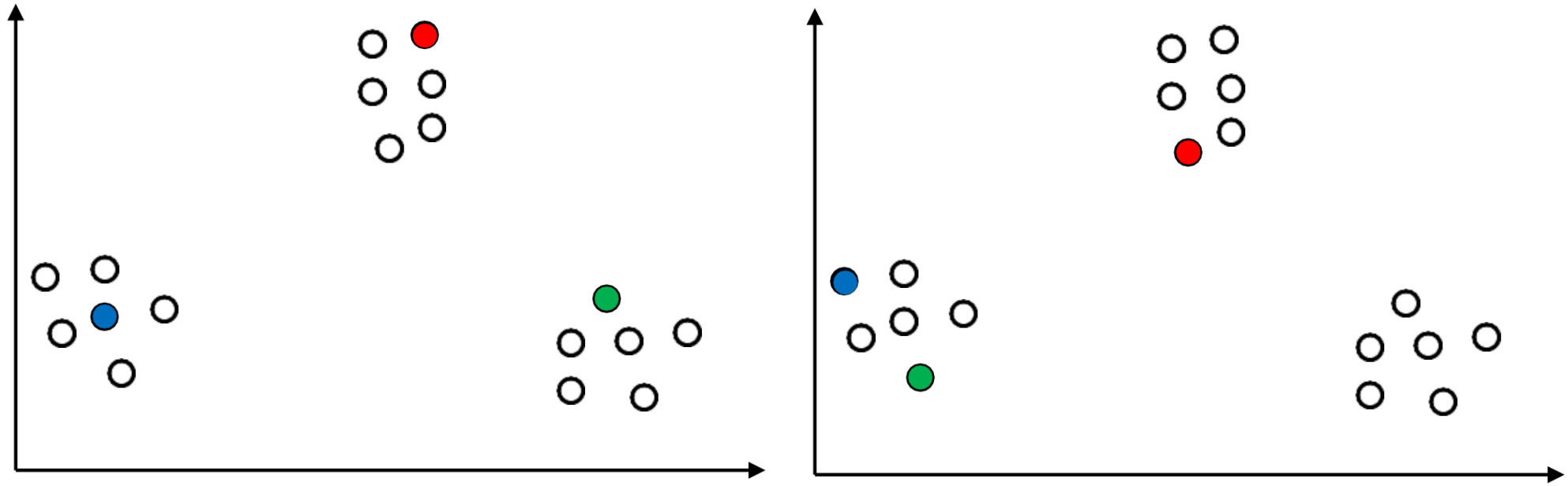
- K-Means
- Gaussian Mixture Models
- PCA

Kmeans

E-step: $z_i = \arg\min_k \|x_i - \mu_k\|_2^2$

M-step: $\mu_k = \frac{1}{n_k} \sum_{i:z_i=k} x_i$

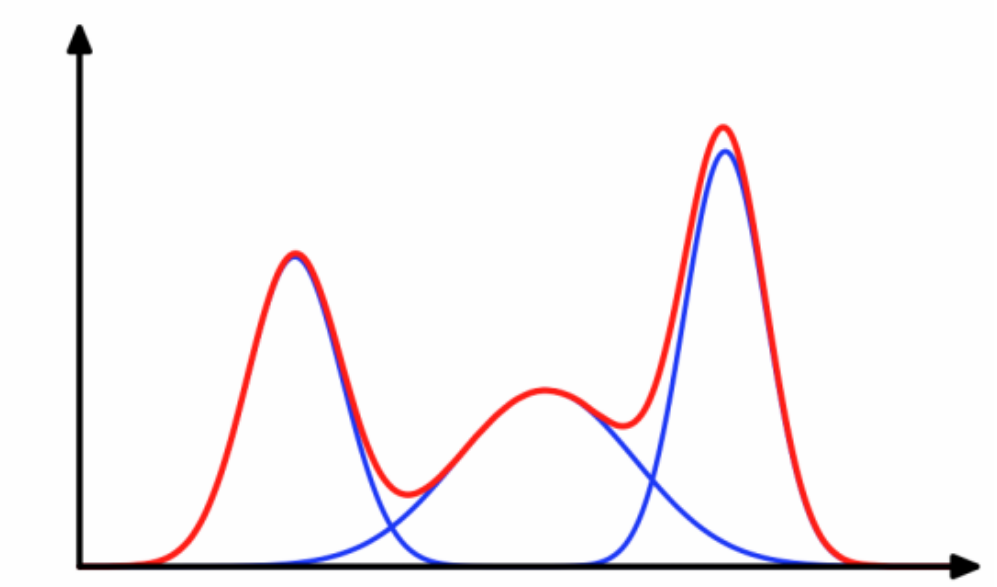
Initialization

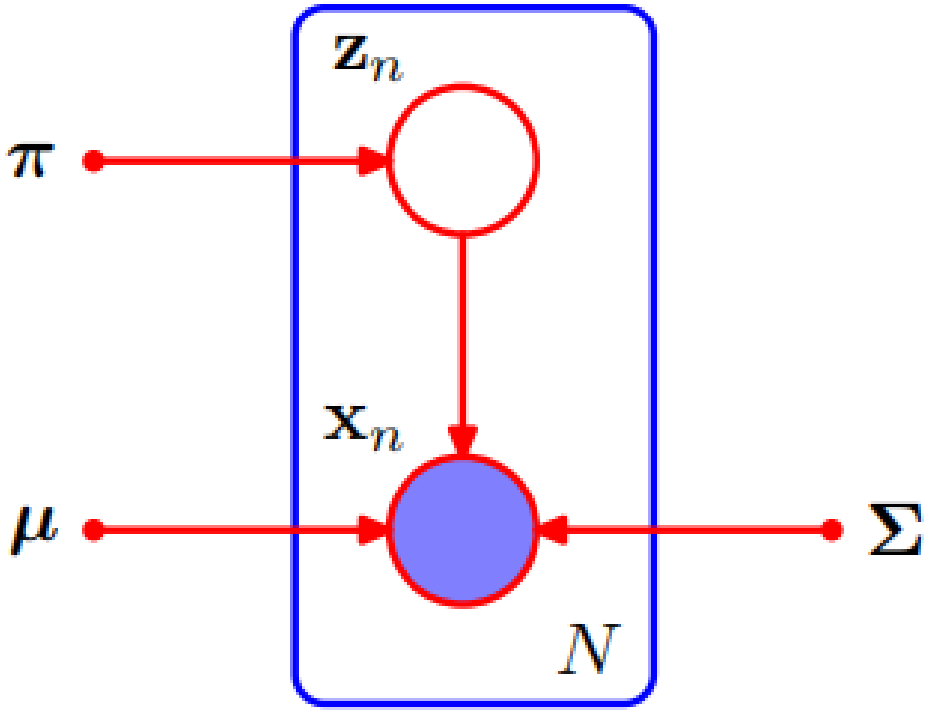


Kmeans++

1. 从所有点中均匀随机选择一个, 作为第一个簇的中心c_1. 所有簇的中心的集合为 $C = \{c_1\}$
2. 对于每个非中心点x_i, 计算x_i到C中每个簇中心的距离
   $$D^2(x) = \min_{c \in C} \|x - c\|^2, \quad x \notin C$$
3. 选择下一个中心:
   $$\Pr(x_i\text{被选作下一个中心}) = \frac{D^2(x_i)}{\sum_{x \notin C} D^2(x)}$$
4. 重复步骤2和步骤3, 直到$|C| = k$

Gaussian Mixture Model(GMM) 高斯混合模型





Given a Gaussian mixture model, the goal is to maximize the likelihood function with respect to the parameters (comprising the means and covariances of the components and the mixing coefficients).

GMM

1. Initialize the means $\boldsymbol{\mu}_k$, covariances $\boldsymbol{\Sigma}_k$ and mixing coefficients π_k.

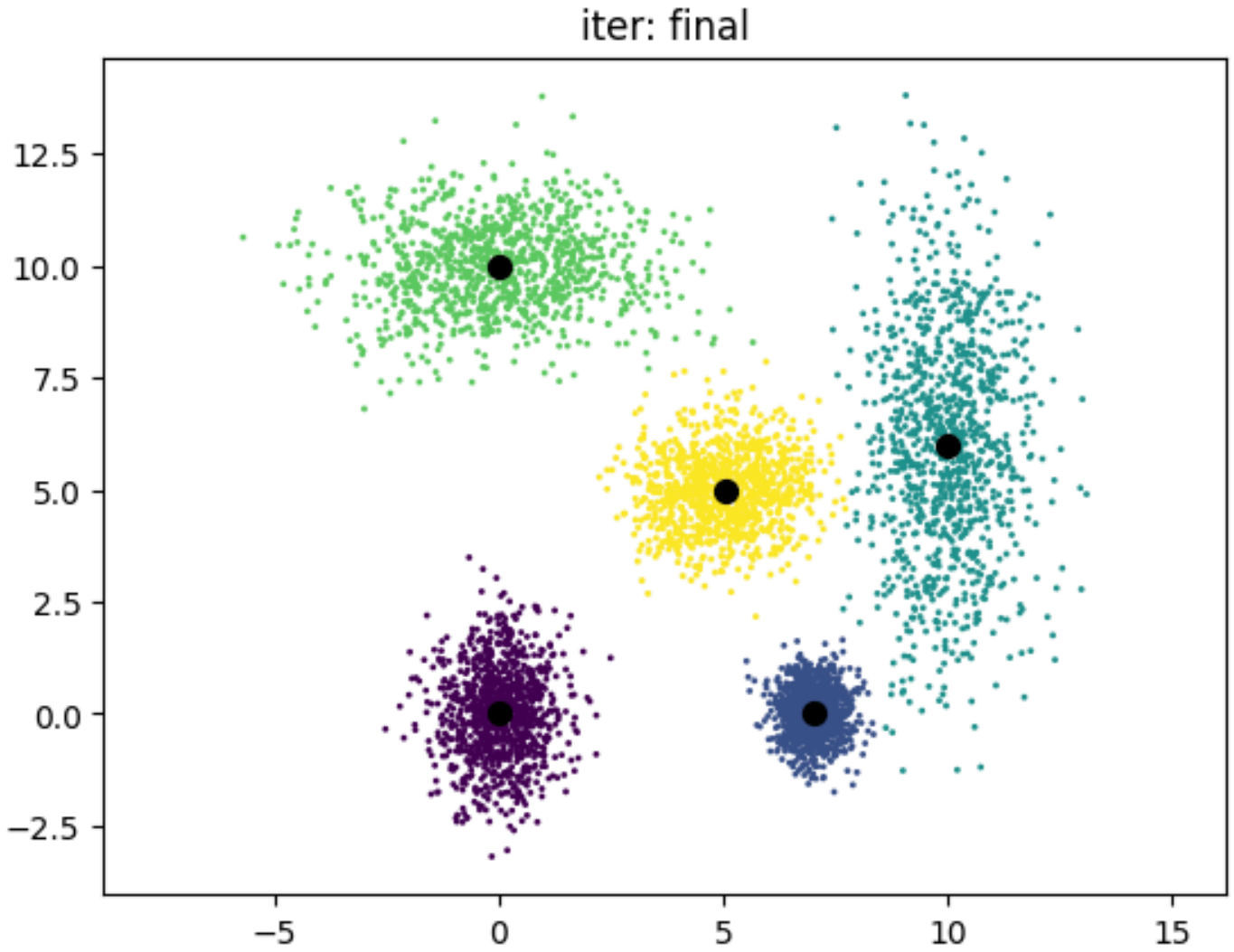
2. E step. Evaluate the responsibilities using the current parameter values

$$\gamma\left(z_{nk}\right)=\frac{\pi_k \mathcal{N}\left(\mathbf{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)}{\sum_{j=1}^K \pi_j \mathcal{N}\left(\mathbf{x}_n \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\right)}.$$

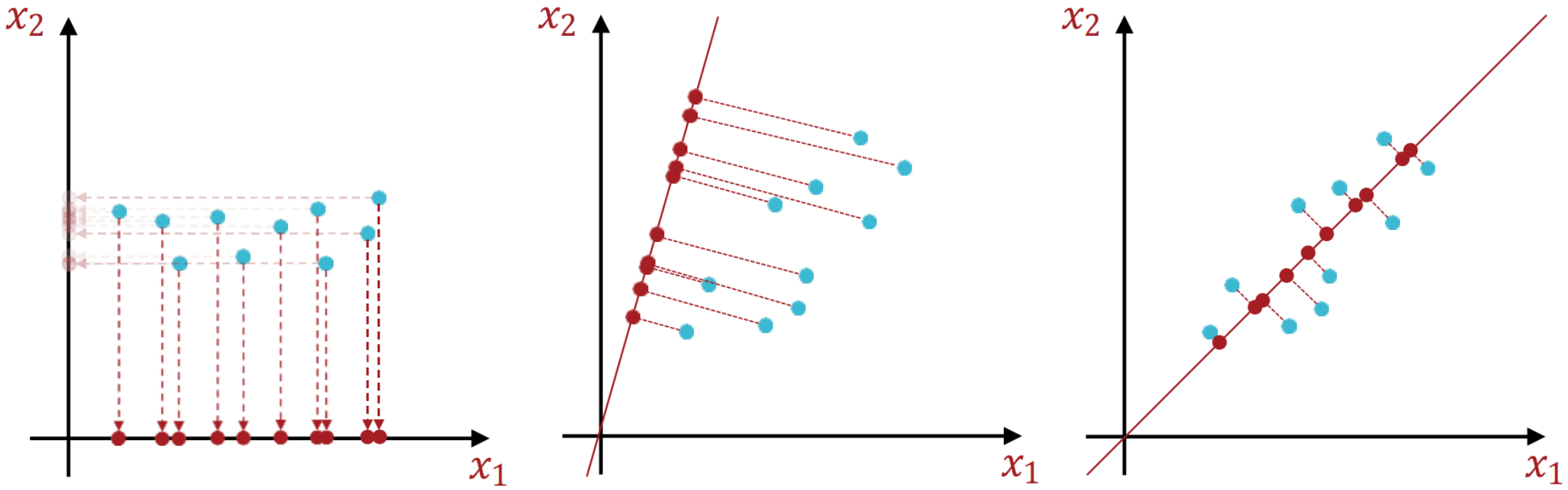
3. M step. Re-estimate the parameters using the current responsibilities

$$\begin{aligned}
\boldsymbol{\mu}_k^{\text{new}} &= \frac{1}{N_k} \sum_{n=1}^N \gamma\left(z_{nk}\right) \mathbf{x}_n \\
\boldsymbol{\Sigma}_k^{\text{new}} &= \frac{1}{N_k} \sum_{n=1}^N \gamma\left(z_{nk}\right)\left(\mathbf{x}_n-\boldsymbol{\mu}_k^{\text{new}}\right)\left(\mathbf{x}_n-\boldsymbol{\mu}_k^{\text{new}}\right)^{\top} \\
\pi_k^{\text{new}} &= N_k / N
\end{aligned}$$

GMM



PCA



- 最大化投影后的方差 / 最小化重建误差

PCA

1. Centerization
   $X = X - \mu$
2. Eigenvalue Decomposition / SVD
   $X = U\Sigma V^\top$
   (U是$XX^\top$的特征向量, V是$X^\top X$的特征向量)
3. 取出$V_1, V_2, \ldots, V_k$(假设数据矩阵式**行向量**拼起来)
4. Projection
   $X_i' = X_i[V_1, V_2, \ldots, V_k]$

Mathematics methods

- Matrix derivative
- Lagrangian function
- KKT
- MAP & MLE
- Optimization methods
- Matrix Factorization
- ELBO
- EM algorithm

Matrix Derivatives 矩阵求导

Types	Scalar	Vector	Matrix
Scalar	$\frac{\mathrm{d}y}{\mathrm{d}x}$	$\frac{\mathrm{d}\mathbf{y}}{\mathrm{d}x}$	$\frac{\mathrm{d}\mathbf{Y}}{\mathrm{d}x}$
Vector	$\frac{\partial y}{\partial \mathbf{x}}$	$\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$	
Matrix	$\frac{\partial y}{\partial \mathbf{X}}$		

layout

- 分子布局
  numerator layout:
  求导结果的维度以分子为主
- 分母布局
  denominator layout:
  求导结果的维度以分母为主
- 机器学习通常使用混合布局:
  向量或者矩阵对标量求导，
  则使用分子布局为准，如果是标量对向量或者矩阵求导，则以分母布局为准

具体总结如下：

自变量\因变量	标量y	列向量$\mathbf{y}$	矩阵$\mathbf{Y}$
标量x	/	$\frac{\partial \mathbf{y}}{\partial x}$ 分子布局：m维列向量（默认布局） 分母布局：m维行向量	$\frac{\partial \mathbf{Y}}{\partial x}$ 分子布局：$p \times q$矩阵（默认布局） 分母布局：$q \times p$矩阵
列向量$\mathbf{x}$	$\frac{\partial y}{\partial \mathbf{x}}$ 分子布局：n维行向量 分母布局：n维列向量（默认布局）	$\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$ 分子布局：$m \times n$雅克比矩阵（默认布局） 分母布局：$n \times m$梯度矩阵	/
矩阵$\mathbf{X}$	$\frac{\partial y}{\partial \mathbf{X}}$ 分子布局：$n \times m$矩阵 分母布局：$m \times n$矩阵（默认布局）	/	/

Matrix Derivatives

常见求导:

- $\frac{\partial \mathbf{a}^\top \mathbf{x}}{\partial \mathbf{x}} = \frac{\partial \mathbf{x}^\top \mathbf{a}}{\partial \mathbf{x}} = \mathbf{a}$
- $\frac{\partial \mathbf{x}^\top \mathbf{A}\mathbf{x}}{\partial \mathbf{x}} = (\mathbf{A} + \mathbf{A}^\top)\mathbf{x}$
- more details:

> Matrix cookbook

- Chain Rule 矩阵求导链式法则
  注意将矩阵的维度对上

> https://www.cnblogs.com/yifanrensheng/p/12639539.html

Lagrangian function

- 对于一个优化问题:

$$\begin{aligned} \min_{\mathbf{x}} \quad & f_0(\mathbf{x}) \\ s.t. \quad & f_i(\mathbf{x}) \leq 0, i = 1, 2, \cdots, m \\ & h_i(\mathbf{x}) = 0, i = 1, 2, \cdots, n \end{aligned}$$

- 其拉格朗日函数为:

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = f_0(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i f_i(\mathbf{x}) + \sum_{i=1}^{n} \nu_i h_i(\mathbf{x})$$

其中$\boldsymbol{\lambda}$和$\boldsymbol{\nu}$是拉格朗日乘子, $\lambda_i \geq 0$, ν_i无约束

KKT Condition

- primal feasibility:

$$\begin{cases} f_i(\mathbf{x}) \leq 0, i = 1, 2, \cdots, m \\ h_i(\mathbf{x}) = 0, i = 1, 2, \cdots, n \end{cases}$$

- dual feasibility:

$$\boldsymbol{\lambda} \succeq 0$$

- complementary slackness:

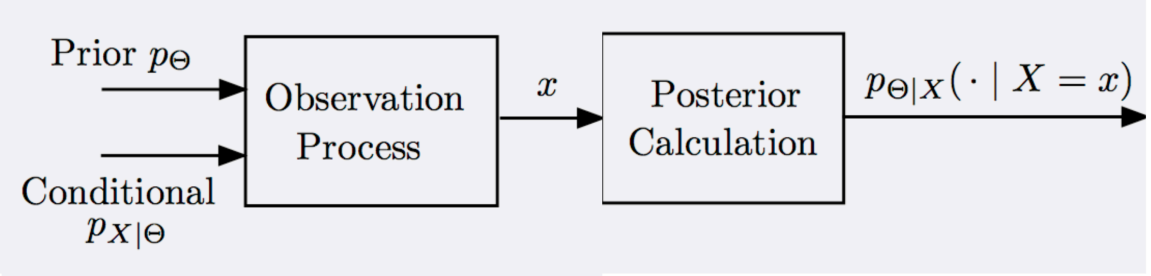
$$\lambda_i f_i(\mathbf{x}) = 0, i = 1, 2, \cdots, m$$

- gradient of Lagrangian:

$$\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = 0$$

MAP: maximum a posteriori 最大后验分布

> Bayesian 统计学派



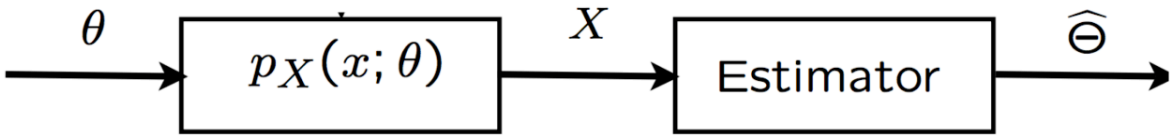
贝叶斯学派把一切变量看作随机变量, 利用过去的知识和抽样数据,将概率解释为信念度(degree of belief), 不需要大量的实验

Θ是个随机变量, 根据样本的具体情况来估计参数, 使样本发生发可能性最大(与时俱进,不断更新)

$$\hat{\theta} = \arg\max_{\theta} p(\theta|\mathcal{D}) \propto P(\mathcal{D}|\theta)P_\Theta(\theta)$$

MLE: maximum a likelihood 最大似然估计

> Frequentist 统计学派



频率学派把未知参数看作普通变量(固定值), 把样本看作随机变量, 仅仅利用抽样数据, 频率论方法通过大量独立实验将概率解释为统计均值(大数定律)

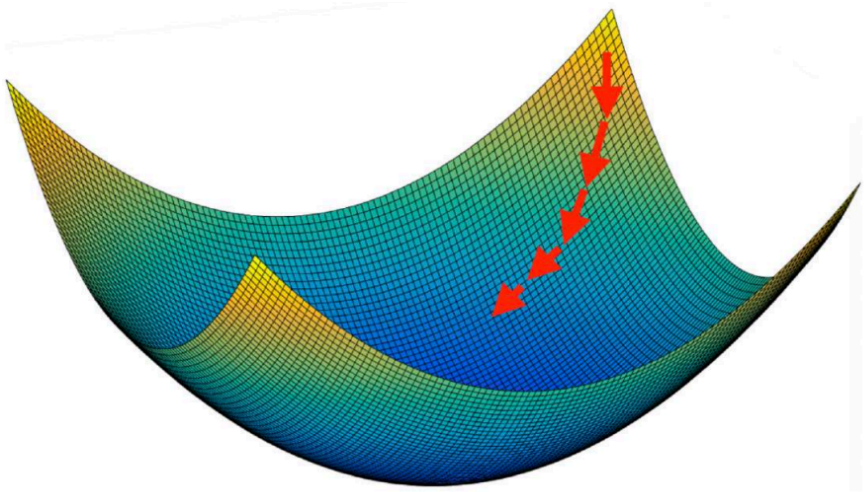
$$\hat{\theta} = \arg\max_{\theta} p(\mathcal{D};\theta) \text{ or } \hat{\theta} = \arg\max_{\theta} p(\mathcal{D}|\theta)$$

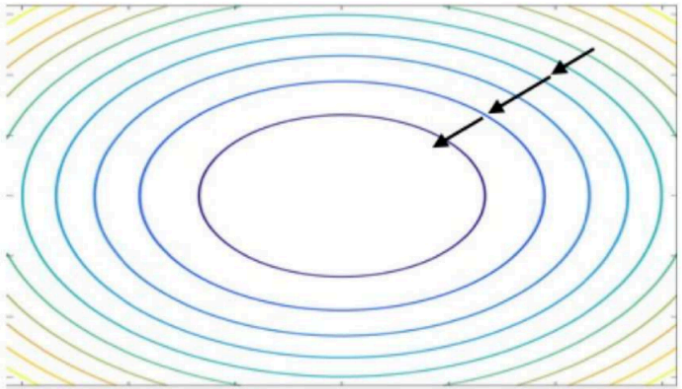
$$\overset{i.i.d.}{=} \arg\max_{\theta} \prod_{i=1}^{n} p(y_i|\theta)$$

$$= \arg\max_{\theta} \sum_{i=1}^{n} \log p(y_i|\theta)$$

Optimization methods

Gradient Descent 梯度下降





初始化一个位置, 然后迭代:

$$x^{k+1} \leftarrow x^k - \alpha_k \nabla f(x^k)$$

Until convergence: $\|\nabla f(x^k)\| < \epsilon$, 或 $\|x^{k+1} - x^k\| \leq \epsilon$, 或 $\|f(x^{k+1}) - f(x^k)\| \leq \epsilon$...

SVD

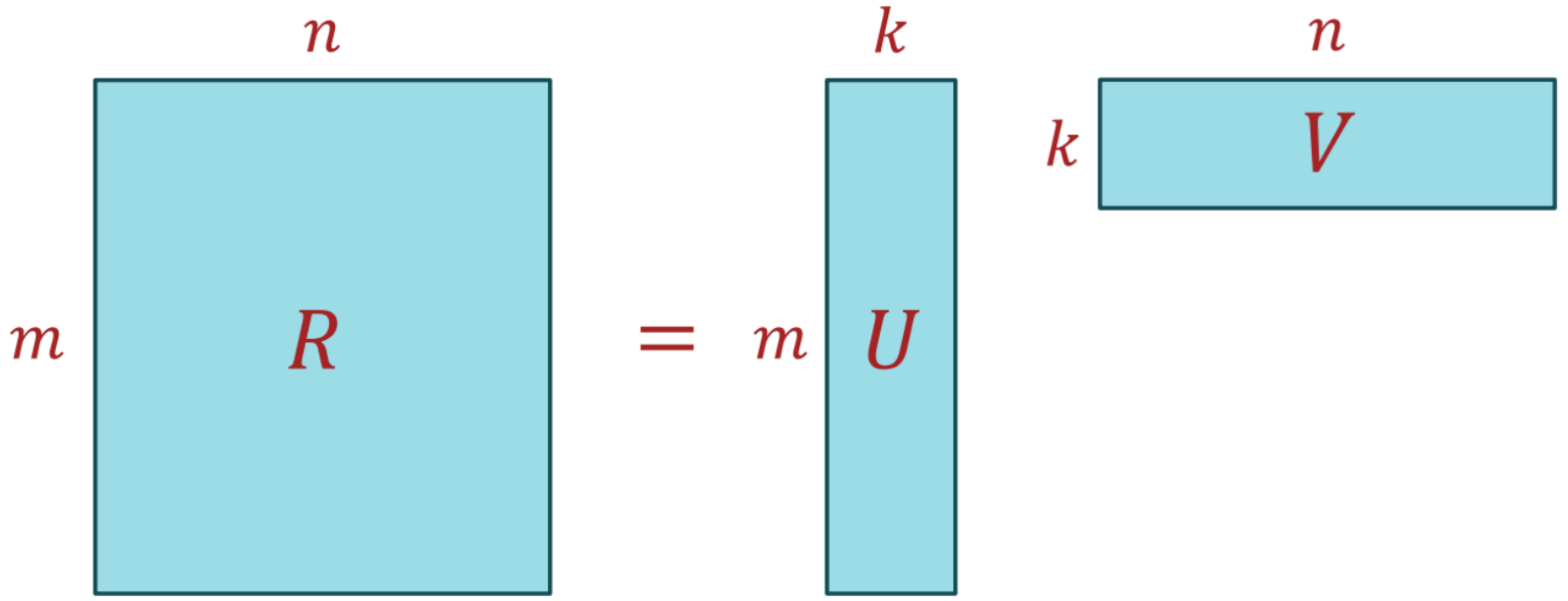
记录 A 的奇异值分解为 $A = U\Sigma V^\top$, 其中 U 和 V 是正交矩阵，Σ 是对角矩阵.

- V & Σ
  $A^\top A = V\Sigma^\top U^\top U\Sigma V^\top = V\Sigma^\top \Sigma V^\top$
  $AA^\top V = V\Sigma^2$
  对于V的每个列向量v_i, $A^\top A v_i = \sigma_i^2 v_i$
  所以Σ为$A^\top A$的特征值的平方根, V为正交规范化的特征向量拼成的矩阵
- U: 同理可得$AA^\top u_i = \sigma^2 u_i$
  设奇异值σ_i的左奇异向量为u_i, 右奇异向量为v_i, 则
  $Av_i = U\Sigma V^\top v_i = U\Sigma e_i = \sigma_i u_i$
  同理: $A^\top u_i = \sigma_i v_i$
  由于U是正交矩阵, 所以求解方程组 $\mathbf{x} \cdot \mathbf{u}_i = 0$的解即可得到$U$

SVD

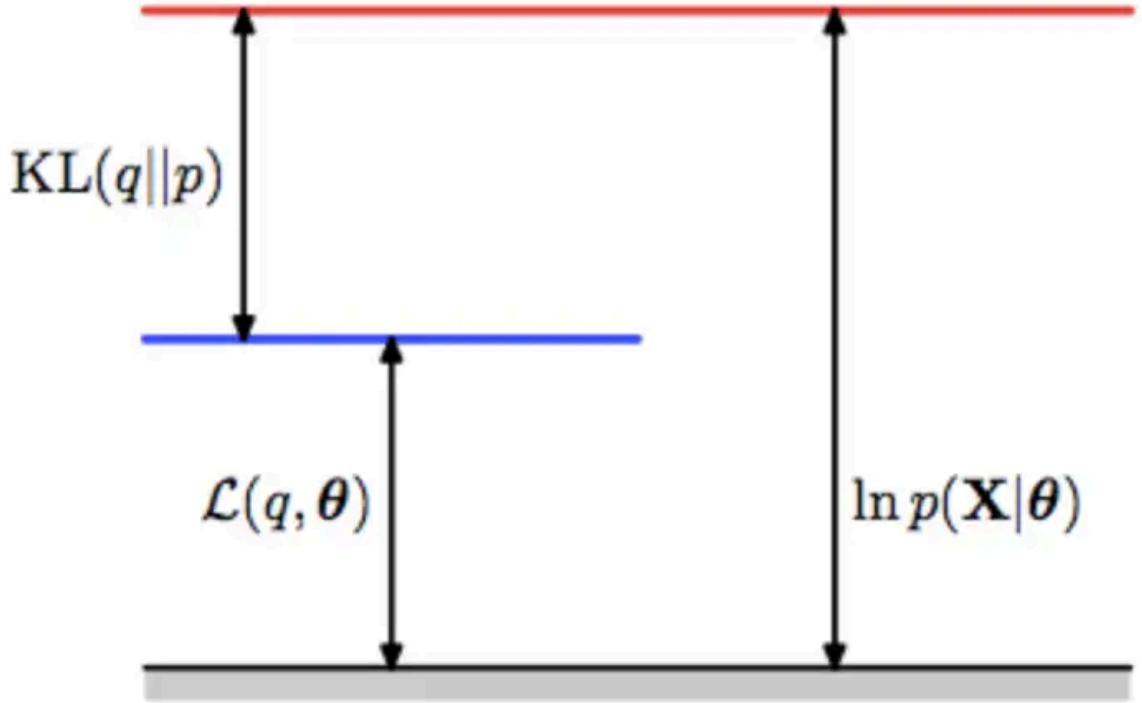
$$A = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{3}} & 0 & \frac{2}{\sqrt{6}} \\ \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{6}} \\ -\frac{1}{\sqrt{3}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{6}} \end{bmatrix} \begin{bmatrix} \sqrt{3} & 0 \\ 0 & \sqrt{2} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Matrix Factorization



$$J(U,V) = \left\| R - UV^\top \right\|_F^2 + \lambda(\|U\|_F^2 + \|V\|_F^2)$$

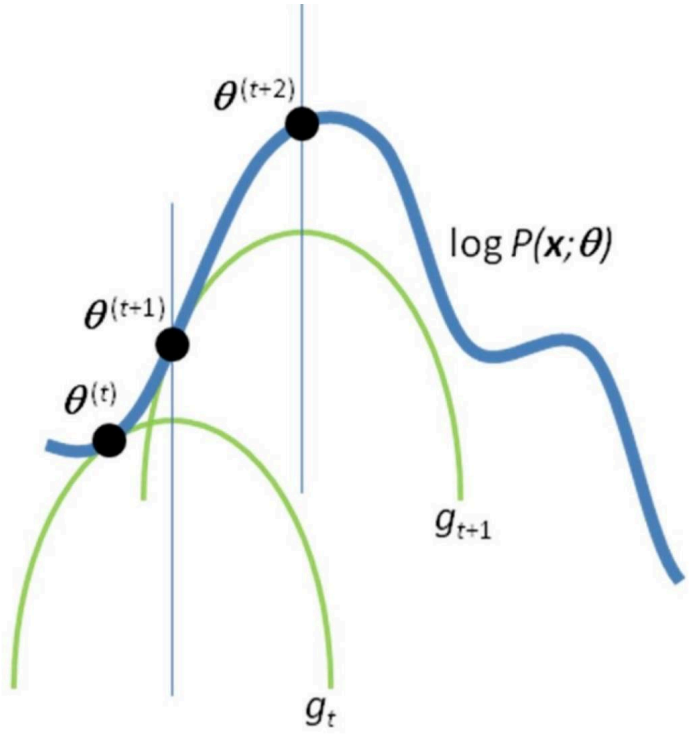
ELBO(Evidence Lower Bound)



$$\log p(X) = \underbrace{\int q(z) \log \frac{p(X, z)}{q(z)} dz}_{\text{Evidence Lower Bound(ELBO)}} + \mathrm{KL}(q(Z) \| p(Z \mid X))$$

Expectation-Maximization(EM) Algorithm

- E step: 先根据当前模型把看不见的那块猜出来
- M step: 拿这份猜出来的数据重新调模型

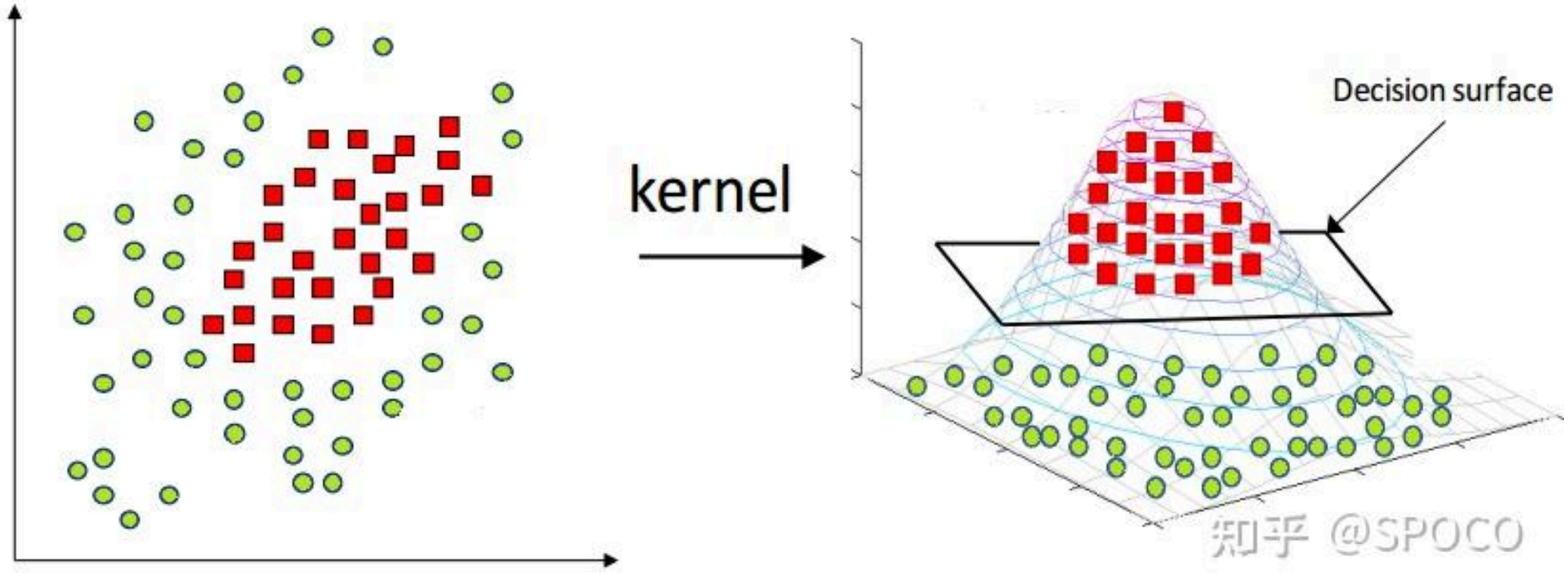


> https://www.zhihu.com/question/19824625/answer/275401651

Machine Learning Concepts

- Kernel Method
- Bias & Variance
- Overfitting & Underfitting
- Regularization
- ...

Kernel Methods 核方法

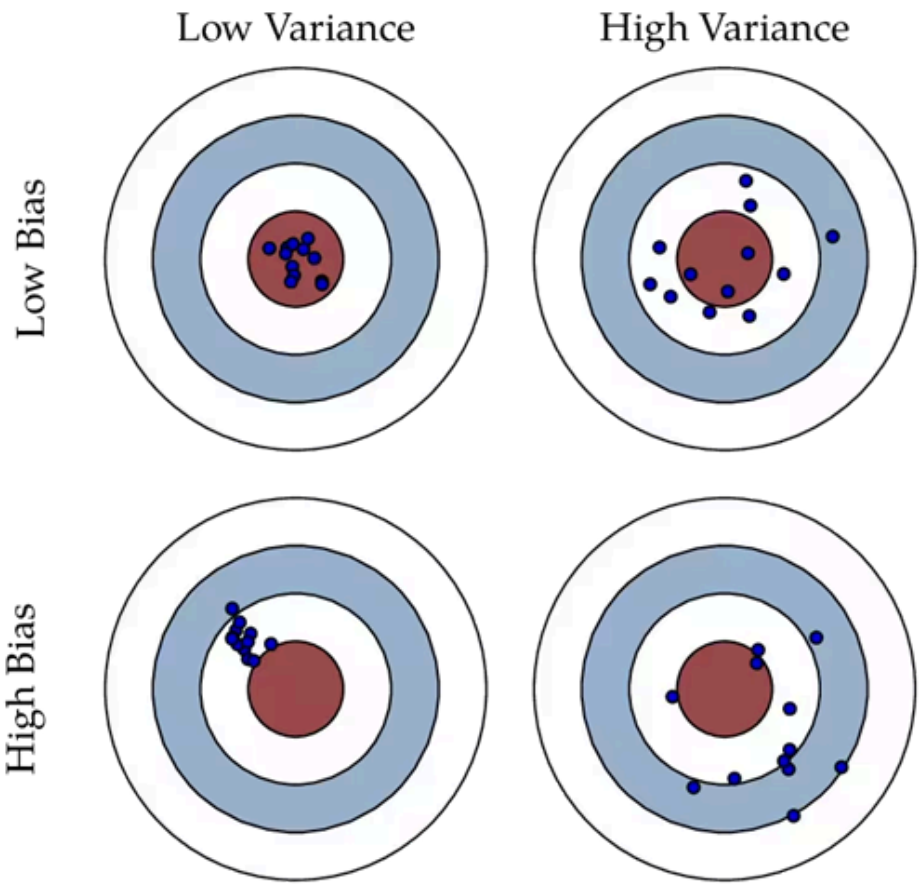


Definition: $K(\cdot,\cdot)$ is a kernel if it can be viewed as a legal definition of inner product:

$\exists \phi : K(x,z) = \phi(x) \cdot \phi(y)$

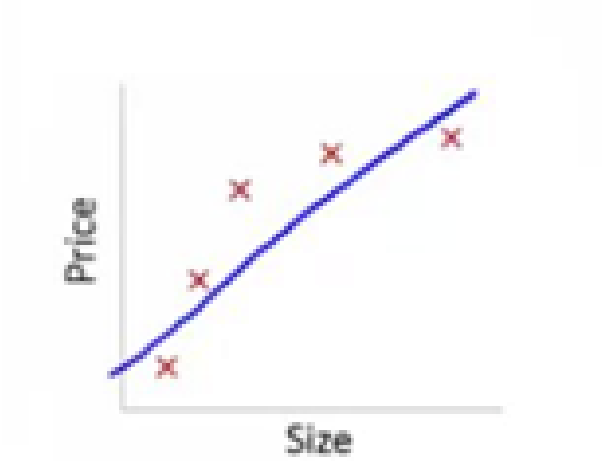
使用时将所有内积$x^\top z$替换为$K(x,z)$

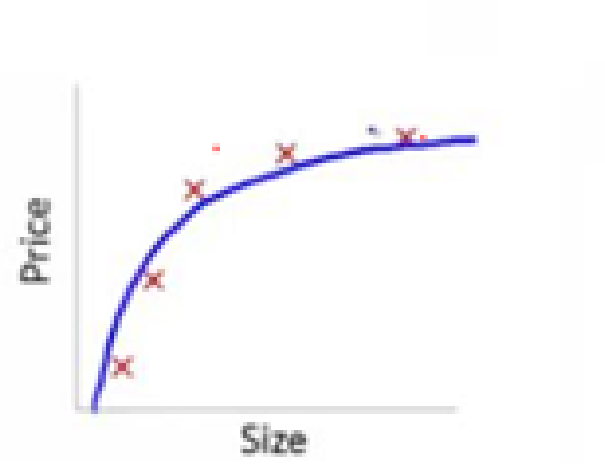
Bias & Variance

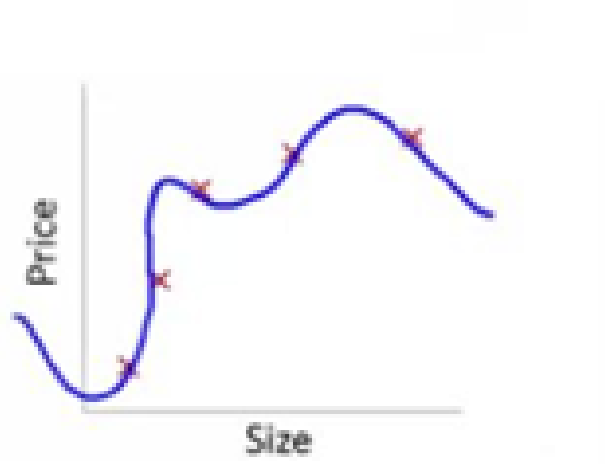


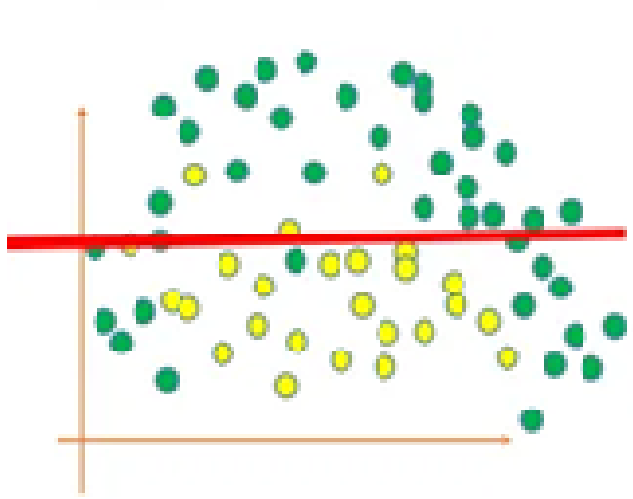
https://www.bilibili.com/opus/412801371382122346

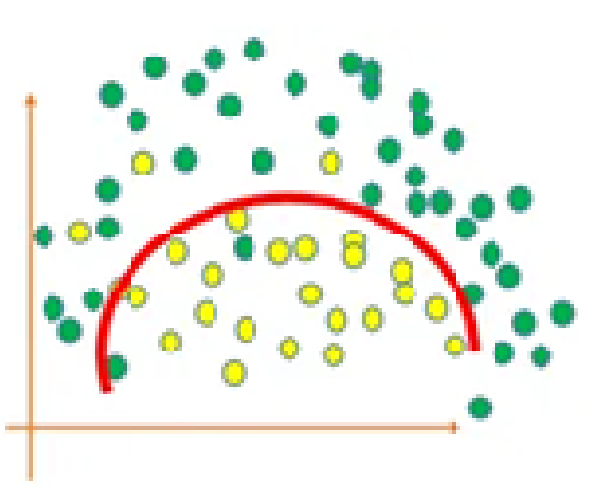
Underfitting & Overfitting

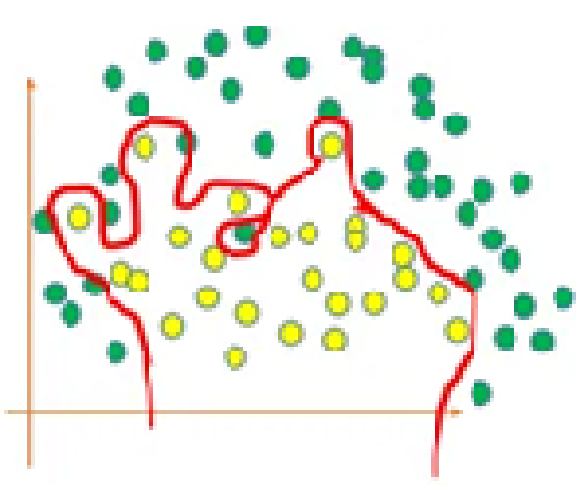












Regularization

降低模型复杂度, 增加 robustness

- L2 正则化(Ridge Regression)

$$\min \sum_{i=1}^{n} \left(y_i - \hat{y}_i\right)^2$$

$$s.t. \|\beta\|_2^2 \leq \lambda$$

$$\min \sum_{i=1}^{n} \left(y_i - \hat{y}_i\right)^2 + \lambda \|\beta\|_2^2$$

- L1 正则化(Lasso Regression)

Regularization for NNs

- L1 / L2 regularization
- Dropout