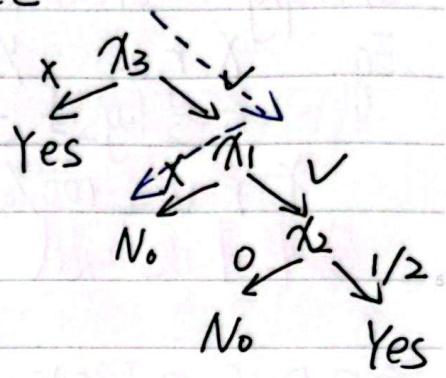
No.

Date

CS182: Introduction to Machine Learning

决策树 Decision Tree

x_1	x_2	x_3	y
✓	0	✓	No
✗	1	✓	No
✗	0	✗	Yes
✓	1	✓	Yes
✓	2	✗	Yes



想从 Feature x_1, x_2, x_3 判断 y，可用决策树建立流程

上表格用来做 training set，建右上图的树。则遇到：

$(x_1, x_2, x_3) = (✗, 2, ✓)$ 样例，决策流程如蓝笔所示

遇到 y 状态叶子结点，流程停止

那么，在树中，依据有先 x_3 后 x_1 后 x_2，能换顺序么？

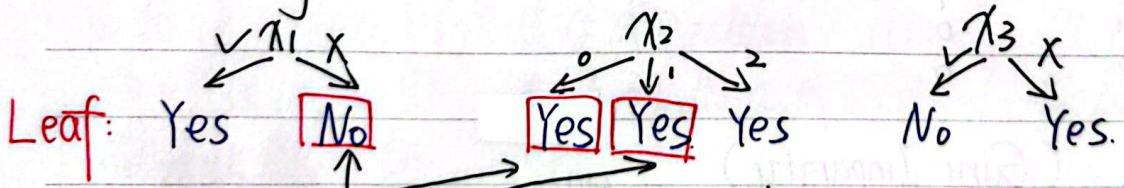
⇓

- Def: A splitting criterion is a function that measures how good or useful splitting on a particular feature is for a specified dataset.

Feature 顺序如何选？由 splitting criterion 量化

有以下常用标准：

① Training Error Rate:



判断 Leaf Node 为谁时：填啥 error 最低，就填啥；一样则均可

Error Rate: 4/5 3/5 4/5

则选 x_1 / x_3 均可

KOKUYO

② Mutual Information Gain

Entropy: $H(X) = -\sum_{v \in V(X)} P(X=v) \log_2 [P(X=v)]$

Eg: X r.v.: ½为0, ½为1. 则称X的熵为:

$-\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2} = 1$

Y r.v.: 100%为0, $H(Y) = -1\log_2 1 = 0$

感性来说: 越不确定, 则熵越大

而又Def: $I(Y;X) = H(Y) - H(Y|X)$

$= H(Y) - \sum_{v \in V(X)} P(X=v)\, H(Y|X=v)$, i.e., $I(y; x_d)$

$= H(Y) - \sum_{v \in V(x_d)} f_v (H(Y_{x_d=v}))$

f_v → fraction of data points where $x_d = v$

$Y_{x_d=v}$ → set of all labels where $x_d = v$

Eg:

x_d	y
1	0
0	0

$I(y, x_d) = H(Y) - \sum_v H(Y_{x_d=v}) \cdot f_v$

$= 1 - \frac{1}{2} H(Y_{x_d=1}) - \frac{1}{2} H(Y_{x_d=0})$

$= 1$

x_d	y
1	1
0	0
0	0

$I(y, x_d) = H(Y) - \cdots$

$= 1 - \frac{1}{2} H(Y_{x_d=1}) - \frac{1}{2} H(Y_{x_d=0})$

$= 1 - \frac{1}{2}\cdot 1 - \frac{1}{2}\cdot 1 = 0$

则在:

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

这样的情况: 选 x_1 作为第一个决策点

③ 基尼指数 (Gini Impurity)

$\text{Gini}(D) = \sum_{k=1}^{|y|} \sum_{k' \neq k} p_k p_{k'} = 1 - \sum_{k=1}^{|y|} p_k^2$

$\text{Gini\_index} = \sum_{v=1}^{V} \frac{|D^v|}{|D|} \text{Gini}(D^v) \Rightarrow a^* = \underset{a \in A}{\text{argmin}}\ \text{Gini\_index}(D, a)$

No.

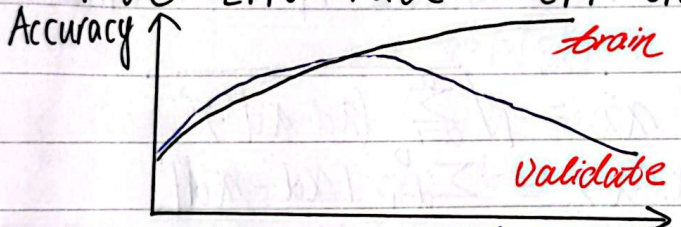
Date

在选出第一个feature之后，取 $x_i = v_i$ 的样本子集，抛开 x_i 因素，考虑选下一个 x_i

Decision Tree Pros: 可解释；高效；可用于分类/回归
但 Cons: splits consider immediate impact ; Overfit *

若 x_i 特征是 real-valued feature，则可以考虑划分分类区间，如：$(-\infty, a]$ 一类 $[a,b)$，$[b,+\infty)\cdots(b>a)$

*: Training Error = err(h, Dtrain)
Test Error = err(h, Dtest)
True Error rate = err(h)，h为所有样品



why overfitting ? 有时划分过程重复，branch过多
为了对付 Decision Tree 中的 overfitting，可采用剪枝处理：
分为 prepruning & post-pruning
预剪枝：对每个结点在划分前估计，若当前结点的划分不能带来泛化性能提升，则停止划分并当前结点标记为叶结点
后剪枝：先从训练集生成Decision Tree，然后自底向上对非叶结点考察，若该结点生成的子树换成叶子结点能带来性能升级，则换。
在 CS182 PPT中展示的仅为 post-pruning，并且顺序是自上而下。以西瓜书为主。
两策略(pre & post)均在讨论：一个非叶子结点是否有必要划分出新特征；pre是生成时便讨论 splitting feature (selected through criterion) 有必要么；post 是生成后自下而上讨论这一点。

KOKUYO

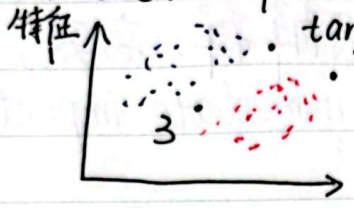
No.

Date . .

KNN : K-Nearest Neighbors

— Duck test: If it looks like a duck, swims like a duck, and quacks like a duck, then it probably is a duck

考虑一个Dataset:



two classes

那么新样本 target1 是 blue(B) or red(R)? 直观上是R，因为它距离红点"更近点"；target 2 & 3 呢？也许就要多看自己身边的样本点"们" label 是啥。

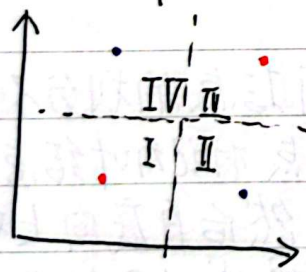
上一段有几个透露的关键：

① "近" => Distance Metric！如何衡量样本间在特征空间上的距离？Emperically：

Euclidean: $d(x, x') = \sqrt{\sum_{d=1}^{D} (x_d - x_d')^2}$

Manhattan: $d(x, x') = \sum_{d=1}^{D} |x_d - x_d'|$

② "们" => 看身边哪些点？Intuitively, k-nearest samples! KNN 算法 idea 便呼之欲出。



在左例中，黑线划分 Feature Space 为4份，构成 Voroni Diagram。划分依据：e.g., 在I中的点，身边最近的一个点是红点；III中是红。(KNN, k=1)

可视构成 Voroni Diagram 的过程为 training，i.e., training 是无error的，因为 $d(x, x) = 0$；在 Test 时，对于一个新 sample，找 K-Nearest Points，取 labels 然后 majority vote 决定它的 label

No.

Date

既然有 majority vote，在 Binary Classification 中，我便希望 ties 不会发生，因此可设 k 为奇数。当然也可：① 再看额外一个点 ② 移去 k 个中最远的 ③ 距离 ⇒ 权重 ④ another distance metric

在 K-NN 中，Distance 为欧式距离 $d: \mathbb{R}^M \times \mathbb{R}^M \to \mathbb{R}$

$$d(u,v) = \sqrt{\sum_{m=1}^{M} (u_m - v_m)^2}$$

当然其他 Distance 也可以。那么这个 algorithm complexity 为？

设 $x \in \mathbb{R}^M$，N samples for training，则：

train time: $O(1)$

Predict time (one sample): $O(MN)$ — on average ↓

可用 k-d tree 优化，则上述两者变为：$O(MN\log N)$；$O(2^M \log N)$

在 M 大时，GG！实践中常用 stochastic approximation

那么 KNN Performance 如何呢？有 Theoretical Guarantee：

(by Cover & Hart, 1967)：$h(x)$ 为 KNN ($k=1$) 的二分类器：

$$\text{error}_{\text{true}}(h) < 2 \times \text{Bayes Error Rate}.$$

BER 可近似理解为：the best you could probably do

✡ Inductive Bias：(归纳偏差) 是算法在进行归纳学习时所持有的先验假设。如决策树的 Inductive Bias 就是尝试找到 (最小的) 决策树 s.t. 训练误差↓且互信息↑

Occam's Razor：try to find the 'simplest' classifier that explains the training dataset

那么 KNN 的 Inductive Bias 便有：{ 1. similar point → similar label; 2. 所有维度 are created equally

第2点 ⇒ 有一个大问题：特征不同的 scale 对 Distance 有影响！

因此假设应舍第2点；当然也可人为定义权重以纠正

KOKUYO

No.

Date . .

(Voroni Diagram中)

之前提到k的影响：那么一个重要现象是：k↑，boundary越平滑；ppt中的例子十分直观！

Extension: Model Selection & Experiment Design

一个model可以设很多不同 setting：如KNN，k=?；Decision Tree中，criterion？因此与其说是一个"model"，不如是"一类"model.

setting 有参数，也有超参。model training 的是为了什么？是当前 model structure、hyperparameter下的最优 parameters.

因此model selection，就是在众多 model中挑最好，而众多model不同的是超参！如何挑超参呢？需要实验！

Training exp: input: 超参 & T Data; output: Best parameters

Hyper exp: input: T & V Data; Output: Best hyperparameters

关于Training Exp，有Cross-Validate特殊技巧：E.g., Data Fold [1,2,3]，则可[1,2]训[3]验，[2,3]训[1]验 … 它们Loss的和可作为为实验设计训练Loss的重要参考！

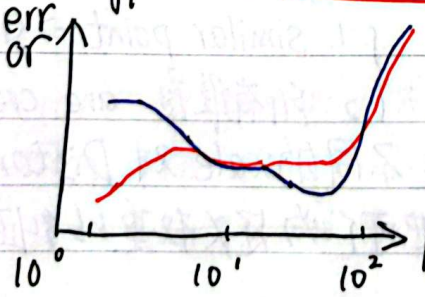
挑超参作实验规范流程：

① Data ⇒ [train] [val] [test]. 挑出来

② [train]上训，更换hyper；[val]上测的最好的hyper

③ 以这个hyper，在 [train]+[val] 上训，用[test]看表现

Eg: K-NN:



超参选择实验：Grid！

α, β, … 所有可能取值作组合一一实验。也可Random选α, β, 重复多次。

△：有限时间内，Random比Grid更有可能找到好的hyper，尤其高维空间中

No.

Date

那么：如果 data 并不 linearly separable 呢？那么 perceptron 不会收敛。但有定理发现：在 examples 上训一遍，依然能得出近似边界

Theorem：设 $\langle (x_1, y_1), \cdots, (x_m, y_m) \rangle$，$\|x_i\| \le R$，令 u 为任一单位向量，$\gamma > 0$. 定义每一个 example deviation 为：

$$d_i = \max\{0, \gamma - y_i(u \cdot x_i)\}$$

再定义：$D = \sqrt{\sum_{i=1}^{m} d_i^2}$，则 Perceptron 在这个集上 mistakes 数量：

$$\le \left(\frac{R+D}{\gamma}\right)^2$$

在 w^* 上投影　　点到 w^* 平面最短距离

* Proof: Claim1: $w_{t+1} \cdot w^* \ge w_t \cdot w^* + \gamma$

w^* 为 max-margin 权重向量，且 $\|w^*\| = 1$

Claim2: $\|w_{t+1}\|^2 \le \|w_t\|^2 + R^2$，因为 $w_{t+1} = w_t \pm x$，$\|x\| \le R$

则 $w_{M+1} \cdot w^* \ge \gamma M$，$\|w_{M+1}\| \le R\sqrt{M}$ 且 $w_{M+1} \cdot w^* \le \|w_{M+1}\|$

$\therefore \gamma M \le R\sqrt{M}$，$M \le \left(\frac{R}{\gamma}\right)^2$

KOKUYO

No.

Date

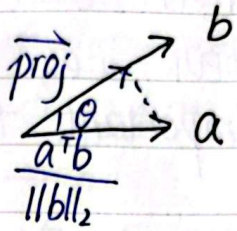
Perceptron

Key idea: Try to learn a hyperplane

设 $\vec{a}$ 为列向量 (by default)，$a^Tb = \sum_{d=1}^{D} a_d b_d$

L_2-norm: $\|\vec{a}\|_2 = \sqrt{a^Ta}$

则 $\vec{a}$ 在 $\vec{b}$ 上的投影为？



△: $\vec{a}\cdot\vec{b} = \|b\|\|a\|\cos\theta$

而 $a^Tb = \|b\| \cdot \|proj\|$，$\|proj\| = \frac{a^Tb}{\|b\|_2}$

故: $\overrightarrow{proj} = \frac{(a^Tb)}{\|b\|_2^2} b$，因为方向为 $\vec{b}$ 方向

在 2D 中，$w_1x_1 + w_2x_2 + b = 0$ 为 Line；3D 中为 plane

4+D 中表示为 hyperplane，$w^Tx + b = 0$. 值得注意的是：

w 向量指向总为 $w^Tx+b > 0$ 的部分！

因此，hyperplane creates 2 halfspaces.

Perceptron 算法：如何更新 w 与 b？

初始化：$w = \vec{0}$，$b = 0$

input: $x^{(t)}$ $y^{(t)} = \{\pm 1\}$，$\hat{y} = \text{sign}(w^Tx+b) = \begin{cases} +1 & \text{if } w^Tx+b \geq 0 \\ -1 & \text{otherwise} \end{cases}$

若 misclassify 一个 +1 例子 ($y^{(t)} = +1$, $\hat{y} = -1$)：

$w \leftarrow w + x^{(t)}$　　$b \leftarrow b+1$

若 misclassify 一个 -1 例子 ($y^{(t)} = -1$, $\hat{y} = +1$)：

$w \leftarrow w - x^{(t)}$　　$b \leftarrow b-1$

为何这样更新？　若：$b + w^Tx < 0$，$y^{(t)} = +1$，欲 $w^Tx \uparrow$.

$w \leftarrow w + x^{(t)}$，$w^Tx = (w + x^{(t)})^Tx = w^Tx + x^Tx > 0$，↑

「反之亦然」　同时 b 也 ↑，$b + w^Tx \uparrow$

而更新逻辑可简化为：$\begin{cases} w \leftarrow w + y^{(t)}x^{(t)} \\ b \leftarrow b + y^{(t)} \end{cases}$

可见，w 可视为 $x^{(1)}, \cdots, x^{(n)}$ 的线性组合

KOKUYO

No.

Date

同时还有一个trick: $\vec{x} = \begin{bmatrix} 1 \\ \vec{x} \end{bmatrix}$ $\vec{\theta} = \begin{bmatrix} b \\ w \end{bmatrix}$. 则:

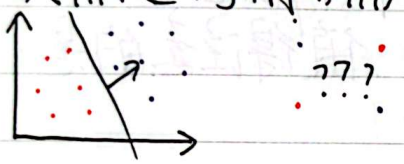
$\hat{y} = \text{sign}(w^T x + b) = \text{sign}(\theta^T x)$

$\theta \leftarrow \theta + y^{(t)} x'^{(t)}$, where $x'(t) = \begin{bmatrix} 1 \\ x^{(t)} \end{bmatrix}$

那么 Perceptron 的 Inductive Bias 是什么? Decision Boundary should be linear, i.e., hyperplane, 且 Recent mistakes are more important than older ones.

而当 θ. b 训到不断产生正确预测时, 我们就认为感知机收敛 (converge) 了。同时, Perceptron 也可能受 overfitting 影响

之前介绍到, 学的其实是一个超平面; 在 binary 分类中, 如果两类点之间的分隔不是"面"而是"space"呢?

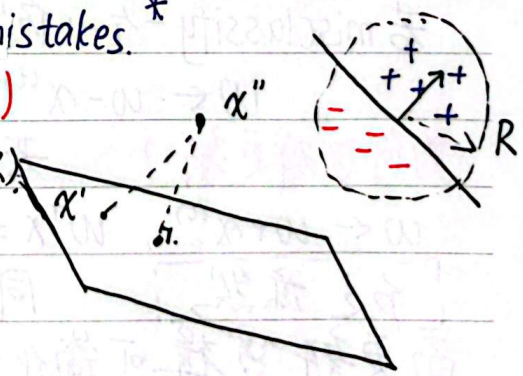


Def: 对二分类问题, 样本集 S is linearly seperable if there exists a linear boundary that seperate the points

"space"大小又引出下面定义: Def: The margin γ for dataset D is the greatest possible distance between a linear separator and the closest point in D to the linear separator

Theorem: 若 Data margin 为 γ. 所有点都在半径为 R 的球内, 则 online perceptron 算法 makes $\leq (R/\gamma)^2$ mistakes. *

那么 margin 如何计算? 相当于是 $(x'' - x')$ 向量在 $w/\|w\|_2$ 上的投影长度 (i.e., 内积)



则 $\left| \frac{w^T(x'' - x')}{\|w\|_2} \right| = \frac{|w^T x'' - w^T x'|}{\|w\|_2}$

$= \frac{|w^T x'' + b|}{\|w\|_2}$

No.

Date

SVM

感知机中，inductive bias 中认为二分任务中的边界是线性的

但若 if not linearly seperable 呢？

这里介绍 kernel method（当然方法不限于此）

Def: A kernel K is a legal def of dot-product, i.e., 有一个隐式映射 Φ, s.t., $K(x,y) = \Phi(x)\cdot\Phi(y)$.

Eg. $K(x,y) = (x\cdot y+1)^d$, Φ: n维 $\to$ n^d维

这个定义看起来没啥用；但 perceptron 中因 $x\cdot z$ 是跳不出 linear 的，而 $x\cdot z$ 换为 $K(x,z)$ 就能拥抱高维

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

· $\exists \phi: X \to \mathbb{R}^N$ s.t. $K(x,z) = \phi(x)\cdot\phi(z)$, range of ϕ: ϕ-space

但 ϕ 是隐式的！这仅是为了方便 view kernel as inner product.

例：$K(x,z) = (x\cdot z)^d$ 对应：$\Phi(x) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)$ $\leftarrow (x_1,x_2)$

可见 $\Phi: \mathbb{R}^2 \to \mathbb{R}^3$, $\Phi(x)\cdot\Phi(z) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)\cdot(z_1^2, z_2^2, \sqrt{2}z_1z_2)$

$= (x_1z_1 + x_2z_2)^2 = (x\cdot z)^2 = K(x,z)$

※：同一个 Kernel，Φ 也许并不唯一！Eg: $(x_1,x_2) \to \Phi(x) = (x_1^2, x_2^2, x_1x_2, x_2x_1)$

应避免显式地扩到高维，i.e., Φ！因为 feature space 会长的飞快!!

因此可以尝试用 $K(x,z)$ 代替 $x\cdot z$。例如 Perceptron 中：

$w_t = a_{i_1}x_{i_1} + \cdots + a_{i_k}x_{i_k}$（可视为未正确分类的 sample 的线性组合）.

对于第 t 的错误：mistake on positive: $a_{i_t} \leftarrow 1$, store x_{i_t}

mistake on negative: $a_{i_t} \leftarrow -1$, store x_{i_t}

$w_t\cdot x = a_{i_1}K(x_{i_1},x) \cdots + a_{i_k}K(x_{i_k},x)$

KOKUYO

No.

Date . .

这个方法在 margin 恰是 Φ 维空间中 linear boundary 时表现很好！

可知：若 margin 在 Φ-space 中，则 Perceptron makes $(\frac{R}{\gamma})^2$ mistakes

Kernel Example:

Linear: $K(x,z) = x \cdot z$

Polynomial: $K(x,z) = (x \cdot z)^d$ or $K(x,z) = (1 + x \cdot z)^d$

Gaussian: $K(x,z) = \exp\left[-\frac{\|x-z\|^2}{2\sigma^2}\right]$

Laplace: $K(x,z) = \exp\left[-\frac{\|x-z\|}{2\sigma^2}\right]$

Properties of Kernel: K is kernel iff: ① K is symmetric

② For any training points $x_1, \cdots, x_m$, and $\forall a_1, \cdots a_m \in \mathbb{R}$:

$$\sum_{i,j} a_i a_j K(x_i, x_j) \geq 0$$

i.e., $K = (K(x_i, x_j))_{i,j=1,\cdots,n}$ 是半正定的：$a^T K a \geq 0$

若 $K_1(\cdot,\cdot)$ $K_2(\cdot,\cdot)$ 均为 kernels. 则 $\forall c_1, c_2 \geq 0$, $K(x,z) = c_1 K_1(x,z)$ $K_2(x,z)$ $+ c_2 K_2(x,z)$ 也是 kernel！且 $K_1(\cdot,\cdot)$ $K_2(\cdot,\cdot)$ 也是！ $K_1(x,z)$ $\phi_2(z)$

Proof: $\phi(x) = (\sqrt{c_1}\phi_1(x), \sqrt{c_2}\phi_2(x))$ $\cdot$ $\Phi(x)\Phi(z) = c_1\phi_1(x)\phi_1(z) + c_2\phi_2(x)$

$\phi(x) = (\phi_{1,i}(x)\ \phi_{2,j}(x))$ $i \in [1,\cdots n]$ $j \in [1,\cdots m]$.

$$\phi(x)\phi(z) = \sum_{i,j} \phi_{1,i}(x)\phi_{2,j}(x)\phi_{1,i}(z)\phi_{2,j}(z)$$

$K_1(x,z)$ $K_2(x,z)$

$$= \sum_i \phi_{1,i}(x)\phi_{1,i}(z)\left(\sum_j \phi_{2,j}(x)\phi_{2,j}(z)\right)$$

缔造上述所有铺垫，SVM, Support Vector Machine 问世！

Def: Margin of example x w.r.t. linear seperation w is the distance from x to plane $w \cdot x = 0$

Def: Margin γ_w of a set S is min margin over $x \in S$, w.r.t. a linear separator w.

No.

Date

Def: Margin γ of set S is the maximum γ_w over all w linear sep.

因此 SVM目标为: Input: $S=\{(x_1,y_1)\cdots(x_m,y_m)\}$

Find: some w and largest maximum γ. where:

① $\|w\|^2=1$ ② For all i, $y_i w\cdot x_i\geq\gamma$

Output: Maximum marginal separator

等效为: $y_i\cdot\frac{w}{\gamma}\cdot x_i\geq 1$, 令 $w'=w/\gamma$. 可见是 $y_i w'\cdot x_i\geq 1$

满足前提下: minimize $\|w'\|^2$, i.e.,

$\arg\min_w \|w\|^2$, s.t., $\forall i$, $y_i w\cdot x_i\geq 1$

Dual Problem*: (Equivalent to:)

$$\max_\alpha \sum\alpha_i - \frac{1}{2}\sum_{i=1}^m\sum_{j=1}^m \alpha_i\alpha_j y_i y_j \phi(x_i)^T\phi(x_j)$$

$$=\max_\alpha \sum\alpha_i - \frac{1}{2}\sum_{i=1}^m\sum_{j=1}^m \alpha_i\alpha_j y_i y_j K(x_i,x_j)$$

$$\text{s.t.}\ \sum_{i=1}^m \alpha_i y_i=0,\ \alpha_i\geq 0$$

最终: classifier 为: $w=\sum_i \alpha_i y_i x_i$

*: 这一坨式子咋得到的?

在最大化间隔同时, 允许部分样本不满足约束(尽可能少)

$$\min_{w,b}\frac{1}{2}\|w\|^2+C\sum_{i=1}^m \ell_{0/1}(y_i(w^Tx_i+b)-1),\ C: \text{constant}$$

$$\ell_{0/1}(z)=\begin{cases}1 & \text{if } z<0\\ 0 & \text{otherwise}\end{cases}$$, 取 hinge loss: $\ell_{hinge}(z)=\max(0,1-z)$

$$\min_{w,b}\frac{1}{2}\|w\|^2+C\sum_{i=1}^m \max(0,1-y_i(w^Tx_i+b))$$, 引入slack变量 $\xi_i\geq 0$:

$$=\min_{w,b,\xi_i}\frac{1}{2}\|w\|^2+C\sum_{i=1}^m\xi_i \quad ①$$

三个约束条件

$$\text{s.t.}\ y_i(w^Tx_i+b)\geq 1-\xi_i,\ \xi_i\geq 0$$

② ③

KOKUYO

No.

Date . .

则构造 Lagrange: $L(w,b,\alpha,\xi,\mu) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i + \sum_{i=1}^{m}\alpha_i(1-\xi_i - y_i(w^Tx_i+b)) - \sum_{i=1}^{m}\mu_i\xi_i$

$\frac{\partial L}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^{m}\alpha_i y_i x_i$ ①

$0 = \sum_{i=1}^{m}\alpha_i y_i$ ② $C = \alpha_i + \mu_i$ ③ 代回去: *

Dual Problem: $\max_{\alpha} \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j x_i^T x_j$

s.t. $\sum_{i=1}^{m}\alpha_i y_i = 0$, $0 \le \alpha_i \le C$, $i = 1, 2, \cdots, m$ $= K(x_i, x_j)$

这是 $K(x,z) = x\cdot z$ 时; 若为 $\Phi(x)$, 则 $\Phi(x_i)^T\Phi(x_j) = \Phi(x_i)\cdot\Phi(x_j)$

△: 附: 使用拉格朗日乘子法得到的便是"对偶问题"

*: $\inf_{w,b} L(w,b,\alpha) = \frac{1}{2}w^Tw + \sum_{i=1}^{m}\alpha_i - \sum_{i=1}^{m}\alpha_i y_i w^T x_i - \sum_{i=1}^{m}\alpha_i y_i b$

$= \frac{1}{2}w^T\sum_{i=1}^{m}\alpha_i y_i x_i - w^T\sum_{i=1}^{m}\alpha_i y_i x_i + \sum_{i=1}^{m}\alpha_i - b\sum_{i=1}^{m}\alpha_i y_i$

$= -\frac{1}{2}w^T\sum_{i=1}^{m}\alpha_i y_i x_i + \sum_{i=1}^{m}\alpha_i$

$= -\frac{1}{2}\sum_{i=1}^{m}\alpha_i y_i x_i^T \sum_{i=1}^{m}\alpha_i y_i x_i + \sum_{i=1}^{m}\alpha_i$

$= \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j x_i^T x_j$

No.

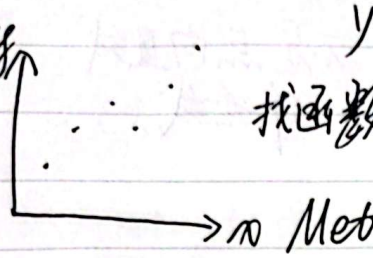
Date

Regression 回归

Goal: Given $\{(\vec{x}, y)\}$ dataset, learn a function:

$y' = h(\vec{x})$, 应与 y 接近

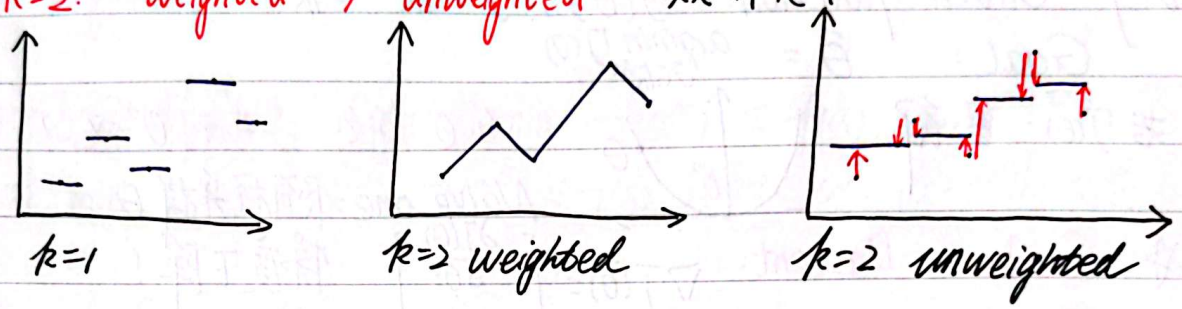
Eg: 找函数拟合左曲线



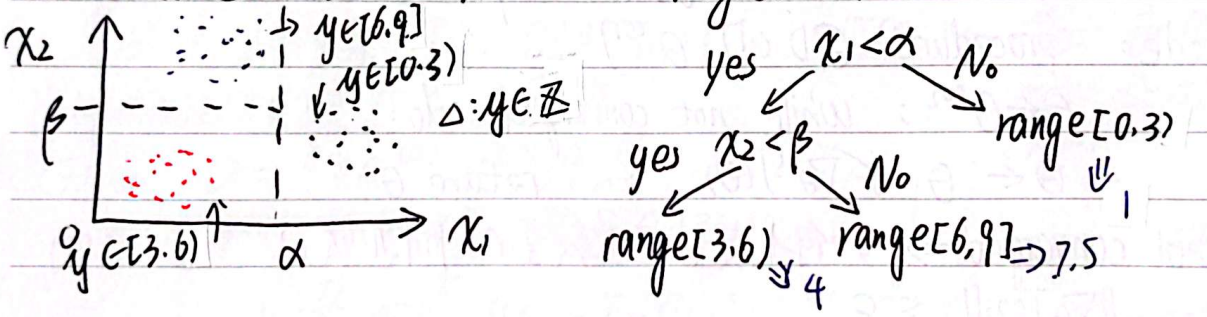
Method ①: k-NN回归

k=1: pick the nearest $\vec{x}$, return its y

or: k=2: weighted / unweighted 效果见下



Method ②: Decision Tree Regression



划分 $\vec{x}_i$; 然后 $\{y_i\}$ 聚为一类, 看mean (Decision Tree分至最后时)

Method ③: Linear Functions, Residuals & MSE

⚠ Linear functions ≠ Linear decision Boundaries

Def: Regression is predicting real-valued outputs

$D = \{(\vec{x}^{(i)}, y^{(i)})\}_{i=1}^{n}$, $\vec{x}^{(i)} \in \mathbb{R}^M$, $y^{(i)} \in \mathbb{R}$

Eg: $y = w^T x + b$

Eg: $y = \text{sign}(w^T x + b)$

KOKUYO

No.

Date

Key idea: Find the linear function h (with w, b) that minimizes the squares of residual for training set.

目标

Residual: $e_i = |y^{(i)} - h(w^T x^{(i)} + b)|$ 不是点向直线作垂线

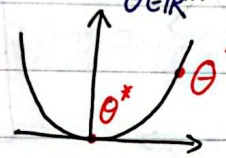
MSE: $\text{Loss} = \frac{1}{n}\sum_{i=1}^{n} e_i^2$

有MSE有什么用? The big picture: Optimization for ML

Def: Given function: $J(\theta)$, $J: \mathbb{R}^M \to \mathbb{R}$

Goal: $\hat{\theta} = \underset{\theta \in \mathbb{R}^m}{\operatorname{argmin}} J(\theta)$

形象理解:



在θ'如何一步步到θ^*呢?

Naive one: 随机采样θ

☆: Gradient Descent: $\nabla J(\vec{\theta}) = \begin{bmatrix} \frac{\partial J(\theta)}{\partial \theta_1} \\ \frac{\partial J(\theta)}{\partial \theta_m} \end{bmatrix}$ 梯度下降!

$\vec{\theta} \leftarrow \vec{\theta} - \delta t \nabla J(\vec{\theta})$. where δt is step size, $\in \mathbb{R}$

Pseudocode: procedure $GD(J, \theta^{(0)})$:

$\theta \leftarrow \theta^{(0)}$; while not converged do:

$\theta \leftarrow \theta - \gamma \nabla_\theta J(\theta)$; return θ

那么 not converged $\Rightarrow$ 如何判定θ已收敛至θ^*附近呢?

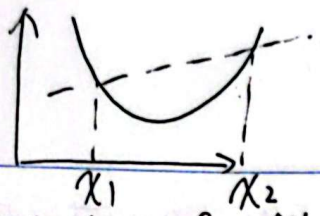
△: $\|\nabla_\theta J(\theta)\|_2 \leq \varepsilon$

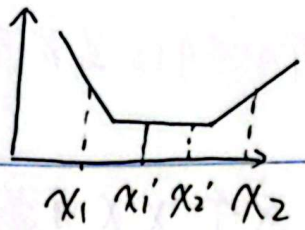
而 MSE Loss: $J^{(i)}(\theta) = (y^{(i)} - \theta^T x^{(i)})^2$, $J(\theta) = \frac{1}{N}\sum_{i=1}^{N} J^{(i)}(\theta)$

则 $\frac{d}{d\theta_k} J^{(i)}(\theta) = 2(\theta^T x^{(i)} - y^{(i)}) \frac{d}{d\theta_k}(\theta^T x^{(i)} - y^{(i)})$

$= 2(\theta^T x^{(i)} - y^{(i)}) \frac{d}{d\theta_k}\left(\sum_{j=1}^{K} \theta_j x_j^{(i)} - y^{(i)}\right) = 2(\theta^T x^{(i)} - y^{(i)}) x_k^{(i)}$

$\therefore \frac{d}{d\theta_k} J(\theta) = \sum_{i=1}^{N} (\theta^T x^{(i)} - y^{(i)}) x_k^{(i)}$, $\nabla_\theta J(\theta) = \begin{bmatrix} \frac{d}{d\theta_1} J(\theta) \\ \vdots \\ \frac{d}{d\theta_m} J(\theta) \end{bmatrix} = \sum_{i=1}^{N} (\theta^T x^{(i)} - y^{(i)}) x^{(i)}$





No.

Date

Optimization for ML:

Convexity: A function $f: \mathbb{R}^D \to \mathbb{R}$, convex if:
$\forall x^{(1)} \in \mathbb{R}^D$ $x^{(2)} \in \mathbb{R}^D$, $c \in [0,1]$, 有:

$$f(cx^{(1)} + (1-c)x^{(2)}) \leq cf(x^{(1)}) + (1-c)f(x^{(2)})$$

上述为 not strictly convex*; 下述为 strictly:* → 如fig1

$c \in (0,1)$, 有: $f(cx^{(1)} + (1-c)x^{(2)}) < cf(x^{(1)}) + (1-c)f(x^{(2)})$

*: 如fig2, 取x_1' x_2', 上面运行实现不了!

对于凸函数来说: global min: $f(x^*) \leq f(x)$, $\forall x \in \mathbb{R}^D$

local min: $\exists \varepsilon$ s.t. $f(x^*) \leq f(x)$. 且 s.t. $\forall x$. $\|x - x^*\|_2 < \varepsilon$

Closed form Optimization:

key: 找到 θ^*, s.t., $\nabla f(\theta^*) = 0$. 其中 f 凸

若无法确保 f 凸. 则找 $\{\theta_i\}$. s.t. $\nabla f(\theta_i) = 0$ for $i \in$ range

并确认它们是极大 or 小值

Given $D = \{(x^{(n)}, y^{(n)})\}_{n=1}^N$, 令: $X = \begin{bmatrix} 1 & x_1^{(1)} & \cdots & x_D^{(1)} \\ 1 & x_1^{(2)} & & \\ \vdots & & \ddots & \\ 1 & x_1^{(N)} & & x_D^{(N)} \end{bmatrix} \in \mathbb{R}^{N \times (D+1)}$

而 $y = [y^{(1)}, \cdots, y^{(N)}] \in \mathbb{R}^N$ 是目标:

$$f(\theta) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{2} (y^{(i)} - \theta^T x^{(i)})^2 = \frac{1}{2N} \sum_{i=1}^{N} (x^{(i)T}\theta - y^{(i)})^2$$

$$= \frac{1}{2N} (X\theta - y)^T (X\theta - y) = \frac{1}{2N} (\theta^T X^T X \theta - 2\theta^T X^T y + y^T y)$$

$$\nabla_\theta f(\theta)^* = \frac{1}{2N} (2X^T X\theta - 2X^T y) = 0 \quad \therefore X^T X\hat{\theta} = X^T y$$

$$\hat{\theta} = (X^T X)^{-1} X^T y$$ *: 矩阵求导!!

*: $\frac{1}{2}$ 无伤大雅; $f(\theta)$ 含义为: 最小 Mean Square Error!!

△: 原: $w^T x + b$, 令 $x = \begin{bmatrix} 1 \\ x \end{bmatrix}$, $\theta = \begin{bmatrix} b \\ w \end{bmatrix}$, 则 => $\theta^T x$

KOKUYO

No. *：直觉上：N=2，那么超平面显然有很多种划分

Date . .

$\hat{\theta} = (X^TX)^{-1}X^Ty$，这个 X^TX 可逆么？一般认为 $N >> D+1$ 时，几乎总是 full-rank 的*。但是关键在于计算 X^TX 逆时间复杂度多少？算 X^TX：$O(ND^2)$；算它的逆：$O(D^3)$.

那么，有没有不那么精确但复杂度能够接受的近似求法呢？

GD & SGD:

Gradient Descent:

$\theta \leftarrow \theta^{(0)}$

while not converged do

$\theta \leftarrow \theta - \gamma \nabla J(\theta)$

return θ

N 个项，$\theta \in R^M$

Δ: $J(\theta) = \frac{1}{N} \cdot$ * $\sum_{i=1}^{N} J^{(i)}(\theta)$

而 Stochastic Gradient Descent:

$\theta \leftarrow \theta^{(0)}$

while not converged do

$i \sim$ uniform $(\{1, 2, \cdots, N\})$

$\theta \leftarrow \theta - \gamma \nabla_\theta J^{(i)}(\theta)$

return θ

实质：原来用全部样本，现在一步用一个

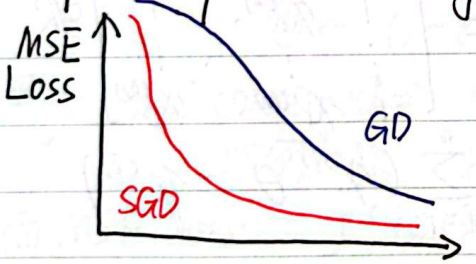
why SGD 能 work？因为 $E[\nabla_\theta J^{(i)}(\theta)] = \sum_{i=1}^{N} p(x^{(i)}, y^{(i)}) \nabla_\theta J^{(i)}(\theta)$

$= \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta J^{(i)}(\theta) = \nabla_\theta J(\theta)$

黑：GD

蓝：SGD

Def: An epoch is a single passing through training data



GD: One update per epoch

SGD: N update per epoch

	step to convergence	Computation per step
GD	$O(\log 1/\varepsilon)$	$O(NM)$
SGD	$O(1/\varepsilon)$	$O(M)$

Theoretical Comparison:

理论上 SGD 收敛更慢，但实践中更快

那么在上一面的 Linear Regression 情景中，$J(\theta) = \frac{1}{N} \sum_{i=1}^{N} (y^{(i)} - \theta^T x^{(i)})^2$

则 SGD: $\theta \leftarrow \theta^{(0)}$

while not converged do:

for $i \in$ shuffle $(\{1, \cdots, N\})$ do ⇐ 为了保证 N 次可过一个 epoch 故是 range(1,N) shuffle 而非模拟 Uniform.

$\theta \leftarrow \theta - \gamma (\theta^T x^{(i)} - y^{(i)}) x^{(i)}$

return θ

GD: $\theta \leftarrow \theta - \gamma \frac{1}{N} \sum_{i=1}^{N} (\theta^T x^{(i)} - y^{(i)}) x^{(i)}$

No.

Date

MLE MAP

MLE 回顾 Likelihood：有 N 个 i.i.d 样本：$D=\{x^{(1)},\dots,x^{(N)}\}$ of r.v. X

若 X 离散，PMF 为 $P(X|\theta)$，则 Likelihood of D is：

$$L(\theta)=\prod_{n=1}^{N} P(x^{(n)}|\theta)$$

若 X 连续，PDF 为 $f(X|\theta)$，则：

$$L(\theta)=\prod_{n=1}^{N} f(x^{(n)}|\theta)$$

考虑 toss coin：$P(Head)=\theta$　$P(Tail)=1-\theta$

$$\hat{\theta}=\arg\max_{\theta} P(D|\theta)=\arg\max_{\theta}\prod_{i=1}^{n} P(x_i|\theta)$$

$$=\arg\max_{\theta}\ \theta^{\alpha_H}(1-\theta)^{\alpha_T}\quad \prod(\theta)$$

$\partial \prod(\theta)/\partial\theta=0 \Rightarrow \hat{\theta}_{MLE}=\frac{\alpha_H}{\alpha_H+\alpha_T}$

考虑 $X\sim Gaussian(\mu,\sigma)$，欲 $\arg\max_{\mu,\sigma^2}$：$\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^n e^{-\sum_{i=1}^{n}(x_i-\mu)^2/2\sigma^2}$　$\prod(\theta)$

$\frac{\partial J}{\partial\mu}=\sum_{i=1}^{n}(x_i-\mu)/\sigma^2=0$；$\frac{\partial J}{\partial\sigma^2}=-n+\alpha*\frac{1}{\sigma^2}=0$

$\therefore \hat{\mu}_{MLE}=\frac{1}{n}\sum_{i=1}^{n}x_i$　$\hat{\sigma}^2_{MLE}=\frac{1}{n}\sum_{i=1}^{n}(x_i-\hat{\mu})^2$

△ 附：MLE 推出的 σ^2 有 bias！　$\hat{\sigma}^2_{unbiased}=\frac{1}{n-1}\sum_{i=1}^{n}(x_i-\hat{\mu})^2$

why？因为 $\hat{\sigma}^2$ MLE 中用的是 sample mean 而非 true mean

*：若 $X\sim Beta(\alpha,\beta)$，则 $P(X)$ max：$x^*=\frac{\alpha-1}{\alpha+\beta-2}$　$E(X)=\frac{\alpha}{\alpha+\beta}$

MAP：The Bayesian Way：

$$P(\theta|D)=\frac{P(D|\theta)P(\theta)}{P(D)\text{常数}}\propto P(D|\theta)P(\theta)$$

Likelihood　prior　or：后验

欲 $\arg\max_{\theta} P(D|\theta)P(\theta)\Leftrightarrow \arg\max_{\theta}[\log P(D|\theta)+\log P(\theta)]$

考虑 flip coin：$P(D|\theta)=\binom{n}{\alpha_H}\theta^{\alpha_H}(1-\theta)^{\alpha_T}$，if $P(\theta)\sim Beta(\beta_H,\beta_T)$

则 $P(\theta|D)\sim Beta(\beta_H+\alpha_H,\beta_T+\alpha_T)$，$\hat{\theta}^*_{MAP}=\frac{\alpha_H+\beta_H-1}{\alpha_H+\beta_H+\alpha_T+\beta_T-2}$

（$\hat{\theta}_{MAP}=\arg\max_{\theta} P(\theta|D)$）

总结：Frequentist：Sample 少时表现不好；Bayesians：不同 prior 不同 answer

KOKUYO

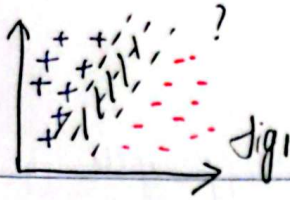


fig1

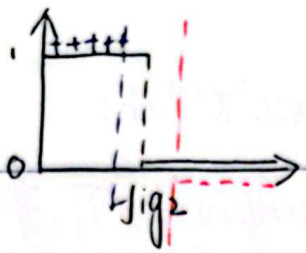


fig2

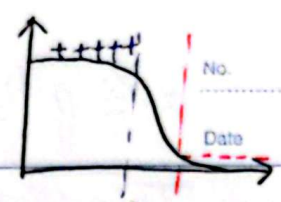


fig3

Logistic Regression

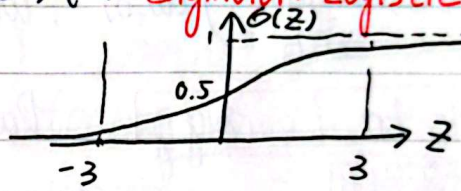
Recall: Perceptron 是简单二分模型，但它无法处理不可分的数据；同时，Perceptron 无法建模 uncertainty：如fig1，超平面划分有很多种，但若有点在"心"区域呢？Perceptron 被迫给出 ±1，但直观上，给出概率可能更合理！

如fig2&fig3：同样如fig中的 sample point，在红蓝中间处，perceptron 只能硬性找一个地方然后作分界；但感性上，希望如fig3这样建模

用怎样的函数能很好建模fig3曲线？Sigmoid / Logistic 函数

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

fig4:



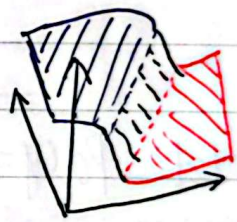
$$\Downarrow$$

$$g(x) = \sigma(w^T x+b) = \frac{1}{1+\exp\{-(w^T x+b)\}}$$

如何预测：predict +1 if probability >0.5

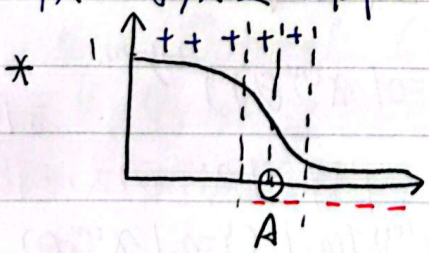
$$\Rightarrow \sigma(w^T x+b) > 0.5$$

$$\Rightarrow \frac{1}{1+\exp\{-(w^T x+b)\}} > 0.5 \Rightarrow \exp\{-(w^T x+b)\} < 1$$



$$\Rightarrow w^T x+b > 0$$ ，可见依然是要训w.b参数，预测时，$w^T x+b>0 \Rightarrow +1$

但！Sigmoid 建模能引入 Uncertainty，并且在数据不可分时，建模仍有质量保障。



虽然A点处有两种 sample point
但因为给出的是 probability distribution
因此有 quality guarantees

How do we learn a classifier?

No.

Data . .

*: 令 $g^{(i)} = \sigma(w^T x^{(i)} + b)$

$$p(\text{data}) = \prod_i p(\text{data point } i) = \prod_i \begin{cases} g^{(i)}, & \text{if } y^{(i)} = +1 \\ 1 - g^{(i)}, & \text{else} \end{cases}$$

$$= \prod_i \left\{ (g^{(i)})^{1\{y^{(i)}=+1\}} (1-g^{(i)})^{1\{y^{(i)}\neq+1\}} \right.$$

$$\Rightarrow \text{Loss} = \frac{1}{n} \sum_{i=1}^{n} -\left(1\{y^{(i)}=+1\} \log g^{(i)} + 1\{y^{(i)}\neq+1\} \log(1-g^{(i)})\right)$$

取负对数 (negative log likelihood loss) (g for guess, a for actual)

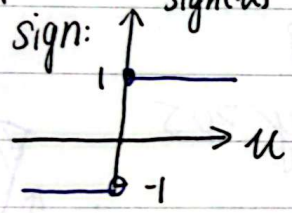
$$-L_{nll}(g,a) = 1\{a=+1\} \log(g) + 1\{a\neq+1\} \log(1-g)$$

目标便是: $J(w,b) = \frac{1}{n} \sum_{i=1}^{n} L_{nll}(\sigma(w^T x^{(i)} + b), y^{(i)})$

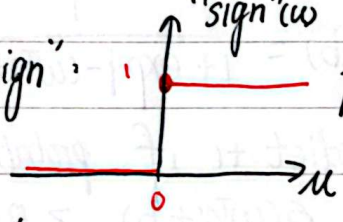
找 $\underset{w,b}{\operatorname{argmin}} J(w,b)$, where: $-L_{nll}(g,a) = 1\{a=+1\} \log g + 1\{a\neq+1\} \log(1-g)$

Back to Learning: Data: $D = \{\vec{x}^{(i)}, y^{(i)}\}_{i=1}^{N}$, $\vec{x} \in \mathbb{R}^M$, $y \in \{0,1\}$.

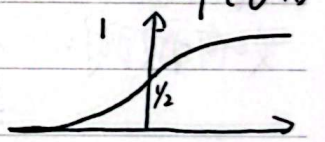
原先预测用: sign: $\text{sign}(u)$



"sign": "sign"(u)



现在: sigmoid: $P_\theta(y=1|\vec{x}) = \frac{1}{1+\exp(-\theta^T \vec{x})}$



Model: $P_\theta(y=1|x) = \frac{1}{1+\exp(-\theta^T x)} = g(x) \Rightarrow P(y|x,\theta) = \begin{cases} g(x) & \text{if } y=1 \\ 1-g(x) & \text{if } y=0 \end{cases}$

Learning: objective:

$$J(\theta) = \frac{1}{n} \sum_{i=1}^{n} -\log P(y^{(i)}|x^{(i)},\theta)$$

欲 $\theta^* = \underset{\theta}{\operatorname{argmin}}\ \ell(\theta)$, $\ell(\theta) = -\frac{1}{N} \log \prod_{n=1}^{N} P(y^{(n)}|\vec{x}^{(n)},\theta)$

$$= -\frac{1}{N} \log \prod_{n=1}^{N} P(Y=1|x^{(n)},\theta)^{y^{(n)}} \left(P(y=0|x^{(n)};\theta)\right)^{1-y^{(n)}} \quad *$$

*: 若 $y^{(n)}=1$, 欲 $P(Y=1|x^{(n)},\theta)$ 大; 反之, 欲 $P(y=0|x^{(n)},\theta)$ 大; 对象不同.

$$= -\frac{1}{N} \sum_{n=1}^{N} y^{(n)} \log P(Y=1|x^{(n)};\theta) + (1-y^{(n)}) \log P(Y=0|x^{(n)};\theta)$$

$$= -\frac{1}{N} \sum_{n=1}^{N} y^{(n)} \theta^T x^{(n)} - \log(1+\exp(\theta^T x^{(n)}))$$

No.

Date

Gradient: Target: $J(\theta) = -\frac{1}{N}\sum_{n=1}^{N} y^{(n)}\theta^T x^{(n)} - \log(1+\exp(\theta^T x^{(n)}))$

$$\nabla_\theta J(\theta) = -\frac{1}{N}\sum_{n=1}^{N}\left[y^{(n)}\nabla_\theta(\theta^T x^{(n)}) - \nabla_\theta \log(1+\exp(\theta^T x^{(n)}))\right]$$

$$= -\frac{1}{N}\sum_{n=1}^{N}\left[y^{(n)}x^{(n)} - \frac{\exp(\theta^T x^{(n)})}{1+\exp(\theta^T x^{(n)})}x^{(n)}\right]$$

$$= \frac{1}{N}\sum_{n=1}^{N} x^{(n)}\left(P(Y=1|x^{(n)},\theta) - y^{(n)}\right)$$

$\frac{\exp(\theta^T x^{(n)})}{1+\exp(\theta^T x^{(n)})} \to \frac{1}{1+\exp(-\theta^T x^{(n)})}$

Then for learning with Gradients given. we can GD/SGD

☆: No close form solution! (无closed form解 for MLE参数)

附: 以 LMS (Least Mean Square), $h_\theta(x) = \theta^T x$,

则 $J_\theta = \frac{1}{n}\sum_{i=1}^{n}(\theta^T x^{(i)} - y^{(i)})^2$, stochastically, (针对一个sample)

$\nabla_\theta J(\theta)^{(i).*} = 2(\theta^T x^{(i)} - y^{(i)})x^{(i)}$, 则 $\theta_k \leftarrow \theta_k + \nabla_\theta J(\theta)$

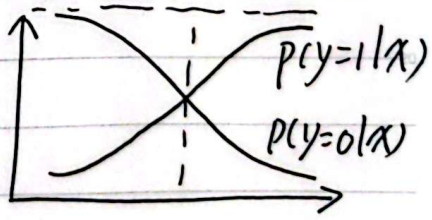
$= \theta_k + \lambda(h_\theta(x^{(i)}) - y^{(i)})$; $\nabla_\theta J(\theta)^{(i).*} = x^{(i)}(P(Y=1|x^{(n)},\theta) - y^{(i)})$

发现 Linear Regression 的 SGD也是: $\theta_k \leftarrow \theta_k + \lambda(h_\theta(x^{(i)}) - y^{(i)})$

*△: 再次recall: SGD: for $i \in$ shuffle($\{1,2,\cdots,N\}$) do:

$\theta \leftarrow \theta \ominus \gamma \nabla_\theta J^{(i)}(\theta)$ ☆注意这里是'-', 且 $\gamma > 0$!

可见, 之前以 $\theta_k \leftarrow \theta_k + \lambda(h_\theta(x^{(i)}) - y^{(i)})$ 中的 λ 应是负数



Bayes Optimal Classifier:

若 $p^*(y|x)$ 这个先验信息被告知呢?

如: 测核酸时. P(阳) << P(阴)

则: $l(y,\hat{y}) = \begin{cases} 1000000, & y \neq \hat{y}, y = \text{阳} \\ 1000, & y \neq \hat{y}, y = \text{阴} \\ 0, & y = \hat{y} \end{cases}$

KOKUYO

No.

Date

则可考虑：Bayes Decision Rule：$\hat{y} = h(x) = \begin{cases} 1, & \text{if } P^*(y=1|x) \geq \alpha \\ 0, & \text{otherwise} \end{cases}$

若 $\alpha = 0.5$，则 $l(y, \hat{y}) = \mathbb{1}(y \neq \hat{y})$

这样在 $\alpha \neq 0.5$ 下，可以设计 asymmetric loss

带这种思想，能用于修改 Log-Loss：

原：$J(\theta) = -\frac{1}{n} \sum_{i=1}^{n} y^{(i)} \log P(Y=1|x^{(i)}, \theta) + (1-y^{(i)}) \log P(Y=0|x^{(i)}, \theta)$

如例中，若 $y=1$，但 $P(Y=1|x^{(i)}, \theta)$ 很小，则 Loss 更大

则赋以权重：$J(\theta) = -\frac{1}{n} \sum_{i=1}^{n} w_1 y^{(i)} \log P(Y=1|x^{(i)}, \theta) + w_2 (1-y^{(i)}) \log P(Y=0|x^{(i)}, \theta)$

$w_1/w_2 > 1$，则 $y=1$ 预测错的后果更严重！i.e., $\nabla_\theta J(\theta) \uparrow \Delta\theta \uparrow$

No.

Date

Recommendation Systems + Boosting

关于前面算法介绍，详见另一文件

(unconstrained!) + 啥意思？后面将会知道

Matrix Factorization (用于 Latent factor methods) (MF)

有许多方式可将一个matrix表示为多个matrix的乘法，比如SVD

而MF就是一个套入了梯度下降式学习的方法：

define model → define 目标 func → optimize with SGD

High level idea: rating matrix (R) ⇒ $U \times V$

where: $U \rightarrow$ users, $V \rightarrow$ items

If $R \in \mathbb{R}^{m \times n}$, and R has rank $k \ll \min(m,n)$, then: Insight.

$\exists\ U \in \mathbb{R}^{m \times k}$ and $V \in \mathbb{R}^{n \times k}$, s.t., $R = UV^T$

若 R rank 为 l, $l > k$ 呢？则是 $\exists\ U_{(m \times k)}\ V_{(n \times k)}$, $R \approx UV^T$

这就是低秩分解的优点！(low rank matrix's rank k: $k \ll \min(m,n)$)

很幸运的是，User-Item 矩阵R被普遍认为低秩！那么如何找 U, V？

Unconstrained MF: Let $E = R - UV^T$

欲E接近零矩阵！故 定义 Objective Function:

$$J(U,V) = \frac{1}{2}\|E\|_2^2 = \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n} E_{i,j}^2 = \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n}(R_{i,j} - U_{j,\cdot}\, V_{\cdot,i}^T)^2$$

问题在于，这便要 $R_{i,j}$ 均已知；但现实是：User 只给极少 item 打分！

即：有 set: $Z = \{(i,j) : R_{i,j} \text{ is known}\}$, $J = \frac{1}{2}\sum_{(i,j)\in Z}(R_{i,j} - U_{j,\cdot}\, V_{\cdot,i}^T)^2$

则：$\nabla_{U_{j,\cdot}} J_{i,j}(U,V) = -E_{i,j} V_{\cdot,i}^T + \lambda U_{j,\cdot}$ $\quad + \frac{\lambda}{2}(\|U\|_2^2 + \|V\|_2^2)$

$\nabla_{V_{\cdot,i}^T} J_{i,j}(U,V) = -E_{i,j} U_j + \lambda V_{\cdot,i}$ ← 正则化

$U_{j,\cdot} \leftarrow U_{j,\cdot} - \eta \nabla_{U_{j,\cdot}} J_{i,j}(U,V)$; $V_{\cdot,i}^T \leftarrow V_{\cdot,i}^T - \eta \nabla_{V_{\cdot,i}^T} J_{i,j}(U,V)$

上述便是用SGD优化！每一次，挑一个 $(i,j) \in Z$，以更新参数

KOKUYO

No.

Date

Trick: 若知 U, V^T 中一个，则 solve 是十分简单的！回想 $J(\theta)=\frac{1}{2}\sum_{i=1}^{N}(y^{(i)}-\theta^T x^{(i)})^2$

则可考虑：初始化 U, V^T 后，while not converged:

- Fix V^T and solve U
- Fix U and solve V^T

SVD for collaborative filtering

SVD: $A = U\Lambda V^T$, where Λ: diagonal, U, V: orthogonal

考虑：对于 User-item R matrix，若已知：

$$R = Q\Sigma P^T$$

then truncate each of Q, Σ and P s.t.: Q, P has k columns, Σ has $k \times k$ entries: $R \approx Q_k \Sigma_k P_k^T$

Then let $U \triangleq Q_k \Sigma_k$, $V \triangleq P_k$

$$\Rightarrow U, V = \underset{U,V}{\operatorname{argmin}} \frac{1}{2}\|R - UV^T\|_2^2$$

Theorem: R 是 fully observed 时，SVD优化出的 U, V 与 MF 的，是一样的

Non-negative MF:

通常公司收集的 R 矩阵中是无负数的！(如：打星：1~5颗，无负！)

故问题变成了：$U, V = \underset{U,V}{\operatorname{argmin}} \frac{1}{2}\|R - UV^T\|_2^2$

s.t. $U_{ij} \geq 0$, $V_{ij} \geq 0$

与先前介绍的 Unconstrained MF 不同，此处有限制！

Ensemble method: 集成学习

Weighted Majority Algorithm: (example of ensemble method)

Given: pool of binary classifier — make new prediction

Goal: design a new learner that uses predictions of pool to

则加权 majority vote 流程如下：

No.

Date

初始化 classifier weights: $\alpha_t = 1$, $\forall t \in [1, \cdots, T]$

对于每个样本 $(\vec{x}, y)$, do:

$\hat{h}(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

if $\hat{h}(x) \neq y$:

for each classifier $h_t \in [1, \cdots, T]$, do:

if $h_t(x) \neq y$, $\alpha_t \leftarrow \beta \alpha_t$ ← hyperparameter

根据个体学习器生成方式，目前 ensemble learning 可分两类：

① 个体学习器间存在依赖关系，必须串行生成 ⇒ boosting 家族

② ····· 不存在强依赖关系，也可并行化式 ⇒ Random forest

Adaboost: ⚠只作二分类

流程: Dataset $D = \{(x_1, y_1), \cdots, (x_m, y_m)\}$, 基学习算法 $\mathfrak{L}$ ← 基模型个数

$D_1(x) = y_m$ $x = [\vec{x_1}, \vec{x_2}, \cdots, \vec{x_m}]$

for $t = 1, \cdots, T$. do: $D_t(x)$: 样本权值分布

(寻找当前误差最小基分类器为 h_t) $h_t = \mathfrak{L}(D_t, D)$ 用 D_t 分布从 D 训练一个弱分类器 h_t

$\epsilon_t = P_{x \sim D_t}(h_t(x) \neq f(x))$ → h_t 误差估计

(if $\epsilon_t > 0.5$, break) (若 >0.5, 则比随机还差!)

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

规范化，确保 $D_{t+1}(x)$ 也是分布

$$D_{t+1}(x) = \frac{D_t(x)}{Z_t} \times \begin{cases} \exp(-\alpha_t), & \text{if } h_t(x) = f(x) \\ \exp(\alpha_t), & \text{if } \neq \end{cases}$$

$$= \frac{D_t(x)}{Z_t} \exp(-\alpha_t h_t(x) f(x))$$

end for

输出: $F(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

*: 用 D_t 分布，可重采样出一个新数据集，在这个数据集上，优化 h_t 参数以 minimize loss. (题目中，此步可能不进行)

No.

Date

理解："加性模型"，$H(x)=\sum_{t=1}^{T}\alpha_t h_t(x)$

以 minimize 指数损失函数：$\ell_{exp}(H|D)=\mathbb{E}_{x\sim D}[e^{-f(x)H(x)}]$

推导：$\frac{\partial \ell_{exp}(H|D)}{\partial H(x)}=-e^{-H(x)}P(f(x)=1|x)+e^{H(x)}P(f(x)=-1|x)$.

$=0$

Δ：$\frac{\partial e^{-H(x)}}{\partial H(x)}=-e^{-H(x)}$　　$\frac{\partial e^{H(x)}}{\partial H(x)}=e^{H(x)}$.

$\ell_{exp}(H|D)=\sum_{i=1}^{|D|}D(x_i)\left(e^{-H(x_i)}\mathbb{I}(f(x_i)=1)+D(x_i)(e^{H(x_i)}\mathbb{I}(f(x_i)=-1)\right)$

$=\sum_{i=1}^{|D|}\left(e^{-H(x_i)}P(f(x_i)=1|x_i)+e^{H(x_i)}P(f(x_i)=-1|x_i)\right)$

$\Rightarrow H(x)=\frac{1}{2}\ln\frac{P(f(x)=1|x)}{P(f(x)=-1|x)}$，则 $sign(H(x))=\begin{cases}1, & P(f(x)=1|x)>P(f(x)=-1|x)\\ -1, & otherwise.\end{cases}$

$=\underset{y\in\{-1,1\}}{\operatorname{argmax}}P(f(x)=y|x)$.

补：$Z_t=\sum_{i=1}^{n}D_t(i)\exp(-\alpha_t y_i h_t(x_i))$

$=e^{\alpha_t}\sum_{i=1}^{n}D_t(i)\mathbb{I}(y_i\neq h_t(x_i))+e^{-\alpha_t}\sum_{i=1}^{n}D_t(i)\mathbb{I}(y_i=h_t(x_i))$

$=e^{\alpha_t}\epsilon_t+e^{-\alpha_t}(1-\epsilon_t)$

训后：最终：

$\epsilon=\frac{1}{n}\sum_{i=1}^{n}\begin{cases}1, & \text{if } y_i\left(\sum_{t=1}^{T}\alpha_t h_t(x_i)\right)\le 0\\ 0, & \text{else}\end{cases}\le\frac{1}{n}\sum_{i=1}^{n}\exp\left(-y_i\left(\sum_{t=1}^{T}\alpha_t h_t\right)\right)$

而 $D_{T+1}(i)=\frac{D_T(i)}{Z_T}\exp(-\alpha_T y_i h_T(x_i))$.　左右连乘：

$\epsilon\le\frac{1}{n}\sum_{i=1}^{n}\exp\left(-y_i\left(\sum_{t=1}^{T}\alpha_t h_t(x_i)\right)\right)=\prod_{t=1}^{T}Z_t\cdot\sum_{i=1}^{n}D_{T+1}(i)=1.$

$\therefore\epsilon\le\prod_{t=1}^{T}Z_t$，Upper bound ↓，又 $Z_t=e^{\alpha_t}\epsilon_t+e^{-\alpha_t}(1-\epsilon_t)$.

$\therefore\frac{\partial Z_t}{\partial\alpha_t}=0$　$(\frac{\partial Z_t}{\partial\alpha_t}>0$, convex$)\Rightarrow$ $\boxed{\alpha_t=\frac{1}{2}\log\frac{1-\epsilon_t}{\epsilon_t}}$.

再代回 Z_t：$Z_t=2\sqrt{\epsilon_t(1-\epsilon_t)}$

No.

Date

Bagging & Random forest

Bagging: Boostrap AGGregretion: ①: Sample bragging

Key: Repeatedly sample (with replacement!) a collection of training examples and train a model on that sample

流程: for $t = 1, \cdots, T$ do

for $s = 1, \cdots, S$ do

$i_s \sim \text{Uniform}(1, \cdots, N)$

$S_t = \{(x^{(i_s)}, y^{(i_s)})\}_{s=1}^{S}$ ← boostrap sample.

$h_t = \text{train}(S_t)$. ← classifier

return $\hat{h}(x) = \text{aggregate}^*(h_1, \cdots, h_T)$

注意，是对每一个 h_t 都重采样一份 S_t

* for classification: majority vote

for regression: average

② Feature Bagging key: select subset of feature as well

for $t = 1, \cdots, T$ do

for $s = 1, \cdots, S$ do

$m_s \sim \text{Uniform}(1, \cdots, M)$

for $i = 1, \cdots, N$ do

$\tilde{x}^{(i)} = [x_{m_1}^{(i)}, x_{m_2}^{(i)}, \cdots, x_{m_S}^{(i)}]^T$. ← sample feature

$D_t = \{(\tilde{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$ ← subspace

$h_t = \text{train}(D_t)$

return $\hat{h}(x) = \text{aggregate}(h_1, \cdots, h_T)$

Random Forest: Key: Combine prediction of many diverse decision trees to reduce variability

If B r.v. all have variance σ^2, Then $\frac{1}{B}\sum_{b=1}^{B} x^{(b)}$'s variance is: σ^2/B !

KOKUYO

So we can combine (sample) bagging and a specific variant of feature bagging to train decision trees

基决策树

流程：↓ 重复：抽一个样本 (with replacement)

→对树的每个结点，先从该结点属性集合中随机选一个子集，再在子集中选最优属性。设|集合|=d，|子集|=k，若 k=d：则 vanilla decision tree splitting node；若 k=1，则随机选一个作结点

建议：$k = \log_2 d$

上述算法中的采样有一个优点：每个基学习器只用了dataset中63.2%的样本，剩下36.8%可用作 validation set 来对泛化能力作'包外估计' (out of bag estimate)；这便可用于超参优化！

Random forest 简单易实现，且performance 很不错，而且收敛性与 Bagging 相似

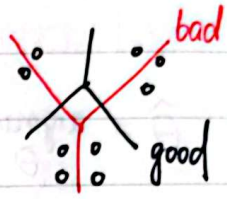
*: $\lim_{m\to\infty} (1-\frac{1}{m})^m \approx 0.368$

8 Expectation Maximization (EM)

Unsupervised Learning: Kmeans & Gaussian Mixture Models (GMM)

考虑下述任务：

Clustering (informal goals): Goal: data 为unlabeled, 将它们分成数组，每组中点相似。



这个任务极为常用！

问题在于：如何建模出 partitions 的好坏？

Input: unlabeled data: $D = \{x^{(i)}\}_{i=1}^{N}$, $x^{(i)} \in R^M$

⇒ cluster centers: $C = [c_1, \cdots, c_K]$ $c_j \in R^M$.

cluster assignments: $z = [z^{(1)}, z^{(2)}, \cdots, z^{(N)}]$, $z^{(i)} \in \{1, \cdots, K\}$

Objective: $\hat{C} = \underset{C}{\operatorname{argmin}} \sum_{i=1}^{N} \min_j \|x^{(i)} - c_j\|_2^2$

$= \underset{C}{\operatorname{argmin}} \sum_{i=1}^{N} \min_{z^{(i)}} \|x^{(i)} - c_{z^{(i)}}\|_2^2$

$\Leftrightarrow \hat{C}, \hat{z} = \underset{C, z}{\operatorname{argmin}} \sum_{i=1}^{N} \|x^{(i)} - c_{z^{(i)}}\|_2^2$ $\quad J(C, z)$.

Algorithm: 1) Initialize $C = \{c_1, \cdots, c_K\}$ ← * 随机K个样本作初始均值向量

2) Repeat until Convergence:

a) for i in $\{1, \cdots, N\}$.

$z^{(i)} \leftarrow \operatorname{argmin}_j (\|x^{(i)} - c_j\|_2)^2$ ← 每个 x_i 选最近 c_j

b) for j in $\{1, \cdots, k\}$.

$c_j \leftarrow \underset{\uparrow}{\operatorname{argmin}} \sum_{i: z^{(i)}=j} (\|x^{(i)} - c_j\|_2)^2$

此优化 i.e.: $\mu_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$, $C_i \leftarrow \mu_i$

* 不止随机，也可 Furthest Point Heuristic：先随机选一个，之后每一个选取尽可能离先前选的点们远！（但 outlier 处理是问题）

也可 K-means++：先随机选 c_1，则 $j = 2, \cdots, K$, pick c_j from distribution:

$P(c_j = x^{(i)}) \propto \min_{j' < j} \|x^{(i)} - c_{j'}\|^2$, where $x^{(i)}$ are not picked before

No.

Date EM

与Kmeans不同，高斯混合聚类采用概率模型表达聚类。先介绍EM：

考虑 $p(X,Z|\theta)=\prod_{n=1}^{N} p(x_n,z_n)$，其中：

observed：$X=\{x_n\}_{n=1}^{N}$；latent：$Z=\{z_n\}_{n=1}^{N}$

Goal：Estimate θ via MLE (or MAP).

$$\hat{\theta}=\arg\max_\theta \log p(X|\theta)=\arg\max_\theta \log \sum_Z p(X,Z|\theta) \quad (\text{discrete})$$

$$=\arg\max_\theta \log \int_Z p(X,Z|\theta)dZ. \quad (\text{continuous})$$

Z：

$$\log p(X|\theta)=\log \sum_Z p(X,Z|\theta)=\log \sum_Z q_i(Z)\frac{p(X,Z|\theta)}{q_i(Z)}$$

$$\geq \sum_Z q_i(Z)\log \frac{p(X,Z|\theta)}{q_i(Z)} \quad (\text{since } \log(\cdot) \text{ is concave; Jensen})$$

$$=\sum_Z q_i(Z)\log p(X,Z|\theta)-\sum_Z q_i(Z)\log q_i(Z)$$

constant! since irrelevant with θ

(if $q_i(Z)=p(Z|X,\theta)$，则为等号)

故令 $q_i(Z)=p(Z|X,\theta)$（独立）：$\log p(X|\theta)=[\mathbb{E}[\log p(X,Z|\theta)]]+\text{const}$

* lower-bound factor! EM就是最大化它！

EM流程：E：以当前θ^t推后验 $P(Z|X,\theta^t)$，计算：（后验 ⇑ 先验 ⇑ θ）

$$Q(\theta|\theta^t)=\mathbb{E}_{Z|X,\theta^t}[\log p(Z,X|\theta)]=\sum_Z p(Z|X,\theta^t)\log p(X,Z|\theta)$$

M：$\theta^{t+1}=\arg\max_\theta\{Q(\theta|\theta^t)+[\log p(\theta)]\}$ 若加D：MAP；不加D：MLE

latent variable；cluster中，$|Z|=k$，k类

$\sum_{k=1}^{K}\pi_k=1$.

GMM：假设生成 N个点：$x_n, n=1,2,\cdots,N$

首先从K个高斯中选一个。此选择基于混合成分先验概率π的：

$z_n \sim \text{Multinomial}(z_n|\pi)$ △（多项式分布）

然后生成点服从 $\mathcal{N}(x_n|\mu_k,\Sigma_k)$，其中$\mu_k$是第$k$个高斯的值，$\Sigma_k$是第$k$个高斯的协方差 (suppose $z_n=k$)

△：说白了就是在 $\{\pi_i\}_{i=1}^{K}$ 的概率下，抽一个 z_i

No.

Date

上述描述了在有 π_i, μ_i, Σ_i 下，如何抽 n 个点出来。

那么学 GMM 呢？考虑：

observed: $X=\{x_1, \cdots, x_N\}$. latent: $Z=\{z_1, \cdots, z_N\}$.

$z_i \in \{1, \cdots, K\}$. ⇒N点分K类

则 complete-data log-likelihood:

$$\log p(X, Z|\theta) = \sum_{i=1}^{N} \log \pi_{z_i} + \log N(x_i|\mu_{z_i}, \Sigma_{z_i})$$

则 E-step: posterior: $\gamma_{ik} = p(z_i=k|x_i, \theta^{old})$

则 $\gamma_{ik} = \dfrac{p(x_i|z_i=k, \theta^{old})\, p(z_i=k|\theta^{old})}{p(x_i|\theta^{old})}$ Bayes

$$= \frac{\pi_k^{old} N(x_i|\mu_k^{old}, \Sigma_k^{old})}{\sum_{j=1}^{K} \pi_j^{old} N(x_i|\mu_j^{old}, \Sigma_j^{old})}$$ Bring in Gaussian

M-step: $Q(\theta, \theta^{old}) = \mathbb{E}_{Z|X,\theta^{old}}[\log p(X, Z|\theta)]$

$$= \sum_{i=1}^{N}\sum_{k=1}^{K} \underbrace{\gamma_{ik}}_{\text{先验}} \underbrace{[\log \pi_k + \log N(x_i|\mu_k, \Sigma_k)]}_{\text{后验}}, \text{ constraint: } \sum_k \pi_k = 1$$

$$\mathcal{L} = Q(\theta, \theta^{old}) + \lambda(1 - \sum_{k=1}^{K} \pi_k)$$

$$\frac{\partial \mathcal{L}}{\partial \pi_k} = \sum_{i=1}^{N} \frac{\gamma_{ik}}{\pi_k} - \lambda = 0 \Rightarrow \pi_k \propto \sum_{i=1}^{N} \gamma_{ik}$$

又 $\sum_k \pi_k = 1$, 故 $\sum_{k=1}^{K}\sum_{i=1}^{N} \gamma_{ik} = 1$

推出：

$$\pi_k^{new} = \frac{1}{N}\sum_{i=1}^{N} \gamma_{ik}$$

$$\mu_k^{new} = \frac{\sum_{i=1}^{N} \gamma_{ik} x_i}{\sum_{i=1}^{N} \gamma_{ik}}$$

$$\Sigma_k^{new} = \frac{\sum_{i=1}^{N} \gamma_{ik} (x_i - \mu_k^{new})(x_i - \mu_k^{new})^T}{\sum_{i=1}^{N} \gamma_{ik}}$$

同理：$\frac{\partial Q}{\partial \mu_k} = \sum_{i=1}^{N} \gamma_{ik} \Sigma_k^{-1}(x_i - \mu_k) = 0$

$$\frac{\partial Q}{\partial \Sigma_k^{-1}} = \frac{1}{2}\sum_{i=1}^{N} \gamma_{ik}[\Sigma_k - (x_i - \mu_k)(x_i - \mu_k)^T] = 0$$

KOKUYO

No.

Date

附：PCA：主成分分析 $\rightarrow \in \mathbb{R}^d$

Input：$D=\{x_1, x_2, \cdots x_m\}$，低维空间维数 d'

Algorithm：中心化：$x_i \leftarrow x_i - \frac{1}{m}\sum_{i=1}^{m} x_i$

计算协方差矩阵：XX^T

特征值分解

取最大 d' 个特征值对应的 eigenvector $w_1, w_2, \cdots, w_{d'}$

$\Rightarrow$ Output $W^* = (w_1, w_2, \cdots, w_{d'}) \in \mathbb{R}^{d' \times d}$.

$x_i \cdot W^{*T}$ 后便进入低维空间

PCA 是最常用的降维方法！