

K近邻 k-NN

一、通用知识

(一)、标准化

1. Z-score标准化: $x_{i, std} = \frac{x_i - \mu_i}{\sigma_i}$

2. 用Mahalanobis距离。当 Σ 如下时，相当于欧几里得标准化距离

$$\Sigma = \begin{pmatrix} \sigma_1^2 & 0 & 0 & 0 \\ 0 & \sigma_2^2 & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_n^2 \end{pmatrix}$$

(二)、测距方法

$L1 - norm$	$\sum_i u_i - v_i $	沿坐标轴折线距离相加，距离相同点在一个正方形上
$L2 - norm$	$\sqrt{\sum_i (u_i - v_i)^2}$	几何距离，距离相同点在一个圆上
$L_\infty - norm$	$\max x_i $	
Mahalanobis距离	$\sqrt{(u - v)^T \Sigma^{-1} (u - v)}$	因为 Σ 存在，距离相同点可以在椭圆上， Σ 为（半）正定对称矩阵

(三)、如何选取超参数？

$$\left\{ \begin{array}{l} \text{学习集 } D_l \left\{ \begin{array}{l} \text{训练集 } D_T 80\% \\ \text{验证集 } D_v 10\% \end{array} \right. \\ \text{测试集 } D_t 10\% \end{array} \right.$$

- D_v 可以选择k，但 D_t 应视作未来数据，测试集的结果不应反馈到模型中。只有在模型训练好后，才用 D_t 测试模型好坏
- 若验证集结果不错，但测试集结果很差，可能需要换一个分布，模型（不采用K-NN算法）

如何采样

- 1 random sampling 随机采样，在非常不幸的情况下，训练集正好没有囊括所有的标签，训练中缺失的标签永远不会被作为预测结果
2. (推荐) stratified sampling, 按原比例保留训练数据确保所有标签都在 D_T 中出现

(四)、二元分类质量评估** Performance evaluation

预测 \ 实际		$y = 1$	$y = 0$
$y = 1$	$y = 1$	TP	FP
	$y = 0$	FN	TN

- 准确度 $acc = \frac{TP + TN}{ALL}$
- 精确度 $prec = \frac{TP}{TP + FP}$ acc与prec一般负相关
- 敏感度 $rec = \frac{TP}{TP + FN}$

二、KNN用途

1. 分类

- 无加权，即比较周围k个最近数据的类型，取最常见的类
- 加权，距离越远权重越低， $Z = \sum \frac{1}{d(x, x_i)}$ 为归一化常数，若只是预测类型则不用管Z

2. 回归 (当 y_i 为数字而非标签)

$$\hat{y} = \frac{1}{Z} \sum_{i \in N_i} \frac{1}{d(x, x_i)} y_i$$

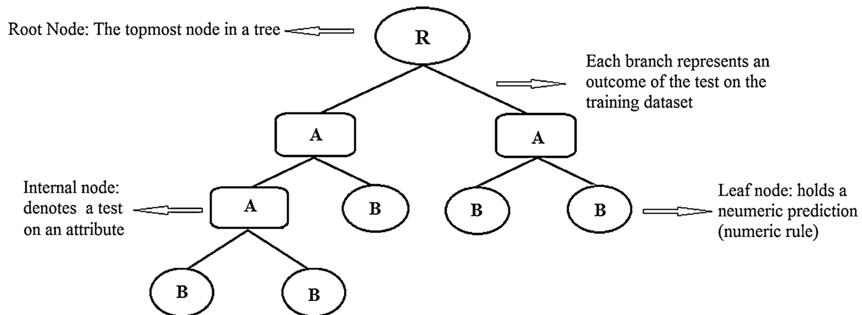
k-NN弊端

- 维数多时，出现Empty-space Problem

每增加一个维度，可能的数据空间呈指数级上涨，而训练数据增长没那么快→新数据离训练数据太远

- 内存memory消耗，检查时间inference时间复杂性均为Order(N)，即随着数据量线性增加
- 需要注意不平衡的类别imbalanced classes，当一个种类占据数据集多数时，预测结果往往会一直是这个类

决策树 Decision tree



- inner nodes: 判断指标
- leaf nodes: 判断结果

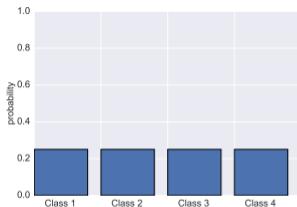
一、不纯净度 (impurity measure)

指标 $i(t)$ 描述在 t 节点某类别 C 的(不)纯度， $\pi_{C_i} = p(y = c|t)$ 即 t 结点上分类到 C 的概率。

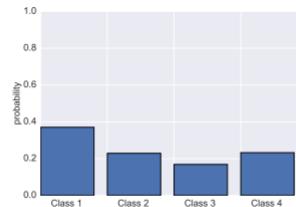
1. 误拣率 Misclassification rate	$i_E = 1 - \max_c \pi_{C_i}$, 即 $1 - \text{主要类别}$	对不同分类可能有相同增益
2. 香农熵 Shannon Entropy	$i_H = -\sum_i^n [\pi_{C_i} \cdot \log_2 \pi_{C_i}]$	离散 X 值
3. Gini指数	$i_G = \sum_{c_i \in t} \pi_{c_i} (1 - \pi_{c_i}) = \sum \pi_{c_i} - \sum \pi_{c_i}^2$ $= 1 - \sum (\pi_{c_i})^2$	

所有的 $i(t)$ 均是越低越好。

信息熵：越高则越“混乱”，各类雨露均沾 (e.g. 墨水在水中散开)；越低则越“秩序” (墨水还未散开)



$$H = 1.3863$$



$$H = 1.3445$$



$$H = 0$$

二、优化决策树

$$\text{一次二叉分类后, 纯净度增益为: } \Delta i(s, t) = \underbrace{i(t)}_{\text{母结点纯净度}} - [\underbrace{p_L}_{\text{分类到左侧概率}} \cdot \underbrace{i(t_L)}_{\text{左侧纯净度}} + p_R \cdot \underbrace{i(t_R)}_{\text{右侧纯净度}}]$$

根节点 (不) 纯净度设为1。 Δi 越大则分类效果越好。

何时停止分叉?

1. 某一枝纯净 $i(t) = 0 \rightarrow$ 但可能过拟合
2. 达到最大深度
3. 每一枝节点数量小于阈值 t_n
4. 分枝增益小于阈值 $\Delta i(s, t) < t_\Delta$
5. 验证集精确度 acc 足够

概率论

一、频率学派——最大似然估计MLE

$p(D|\theta)$: 观察到序列D, 哪个 θ 能让这个D出现的概率最大?

写出事件D的方程 $f(\theta)$, 再求 $\arg \max_{\theta} f(\theta) = \arg \max_{\theta} \log f(\theta)$

缺陷: 样本少时容易过拟合

	观察序列	MLE
二项分布	$p(D \theta) = \theta^{ T }(1-\theta)^{ H }$	$\theta_{MLE} = \frac{ T }{ T + H }$
高斯分布 (方差为1)	$p(x \mu) = \mathcal{N}(x \mu, 1) = \frac{1}{\sqrt{2\pi}} \cdot \exp(-\frac{1}{2}(x-\mu)^2)$ $p(D \mu) = \prod_{i=1}^n \mathcal{N}(x_i \mu, 1)$	$\mu_{MLE} = \frac{\sum_{i=1}^n x_i}{n}$

二、贝叶斯派——观察+先验=后验MAP

$p(\theta|D) = \frac{p(D|\theta) \cdot p(\theta)}{p(D)} \propto p(D|\theta) \cdot p(\theta)$: 引入前置分布 **prior** $p(\theta)$, 在样本数太少时也有比较好的估计

同样计算 $\arg \max_{\theta} \log[f(\theta) \cdot p(\theta)]$, 找到最可能的序列D

缺陷: 由于乘法运算, 先验 $p(\theta) = 0$ 处后验永远为0

(一)、如何选取 $p(\theta)$?

选择和观察序列D公式形式上相同的, 可以让后验与先验形式一致, 简化计算

观察序列	共轭先验方程	后验形式	MAP
$p(D \theta) = \theta^{ T }(1-\theta)^{ H }$	$Beta(\theta a, b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \theta^{a-1} (1-\theta)^{b-1}$	$p(\theta D) \propto \theta^{ T +a-1} (1-\theta)^{ H +b-1}$	$\theta_{MAP} = \frac{ T +a-1}{ H + T +a+b-2}$
$p(D \mu) = \prod_{i=1}^n \mathcal{N}(x_i \mu, 1)$	$\mathcal{N}(\mu 0, \underbrace{\alpha^{-1}}_{\alpha \text{为精度}=\sigma^2}) = \sqrt{\frac{\alpha}{2\pi}} \cdot \exp(-\frac{\alpha}{2}\mu^2)$	$p(\mu D, \alpha) \propto \exp(-\frac{N+\alpha}{2}\mu^2 + \sum_{i=1}^N x_i \cdot \mu)$	$\mu_{MAP} = \frac{1}{N+\alpha} \cdot \sum_{i=1}^N x_i$

$$\begin{aligned}
p(\mu|D, \alpha) &= \frac{p(D|\mu) \cdot p(\mu|\alpha)}{p(D|\alpha)} \propto p(D|\mu) \cdot p(\mu|\alpha) \\
&\propto \left(\prod_{i=1}^N p(x_i|\mu) \right) \cdot p(\mu|\alpha) \\
&\propto \left(\prod_{i=1}^N \mathcal{N}(x_i|\mu, 1) \right) \cdot \mathcal{N}(\mu|0, \alpha^{-1}) \\
\text{高斯分布后验证证明: } &\propto \left(\prod_{i=1}^N \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}(x_i - \mu)^2\right) \right) \cdot \sqrt{\frac{\alpha}{2\pi}} \exp\left(-\frac{\alpha}{2}\mu^2\right) \\
&\propto \exp\left(\sum_{i=1}^N \left[-\frac{1}{2}(x_i - \mu)^2\right] - \frac{\alpha}{2}\mu^2\right) \\
&\propto \exp\left(-\frac{1}{2} \sum x_i^2 + \mu \sum x_i - \frac{1}{2} \sum \mu^2 - \frac{\alpha}{2}\mu^2\right) \\
&\propto \exp\left(\mu \sum x_i - \frac{N}{2}\mu^2 - \frac{\alpha}{2}\mu^2\right) \\
&\propto \exp\left(-\frac{N+\alpha}{2}\mu^2 + \sum x_i \cdot \mu\right)
\end{aligned}$$

(二)、全贝叶斯分布函数

求MAP时，只关注 $\operatorname{argmax}_\theta [\alpha \cdot p(D|\theta) \cdot p(\theta)] \propto \operatorname{argmax}_\theta [p(D|\theta) \cdot p(\theta)]$ ，常数 α 可以忽略

要求得真正的分布函数，必须找到归一化常数 α ，使得 $\int_0^1 p(\theta|D)d\theta = 1$ 。实际上先验方程选的好就一定能合并观察公式

观察序列	分布函数
$p(D \theta) = \theta^{ T } (1-\theta)^{ H }$	$p(\theta D) = \text{Beta}(\theta a+ T , b+ H)$
$p(D \mu) = \prod_{i=1}^n \mathcal{N}(x_i \mu, 1)$	$p(\mu D) = \mathcal{N}(\mu \frac{1}{N+\alpha} \sum_{i=1}^N x_i, \frac{1}{N+\alpha})$

$$\begin{aligned}
\mathcal{N}(\mu|m, \beta^{-1}) &= \sqrt{\frac{\beta}{2\pi}} \cdot \exp\left(-\frac{\beta}{2} \cdot (\mu - m)^2\right) \\
\text{高斯分布分布函数证明: 设分布函数为} &= \sqrt{\frac{\beta}{2\pi}} \cdot \exp\left(-\frac{\beta}{2}\mu^2 + \beta\mu m - \frac{\beta}{2}m^2\right) \\
&\propto \exp\left(-\frac{\beta}{2}\mu^2 + \beta m \cdot \mu\right)
\end{aligned}$$

$$\text{与后验形式 } p(\mu|D, \alpha) \propto \exp\left(-\frac{N+\alpha}{2}\mu^2 + \sum_{i=1}^N x_i \cdot \mu\right) \text{ 比较: } \left\{ \begin{array}{l} -\frac{\beta}{2} = -\frac{N+\alpha}{2} \\ \sum_{i=1}^N x_i = \beta m \end{array} \right. \implies \left\{ \begin{array}{l} \beta = N+\alpha \\ m = \frac{\sum_{i=1}^N x_i}{N+\alpha} \end{array} \right.$$

三、MLE、MAP的联系

贝叶斯概率中所谓的先验（或前置分布），就是一种预设，可以是主观的（我认为新疆来的苹果更可能是红的），也可以是客观的（统计规律，一枚正常的硬币落地正反面概率应各为 $\frac{1}{2}$ ）——但不论主观客观，归根结底都是笼统的统计结果（生活中我们看到新疆来的苹果的确红色的更多，日复一日形成了“主观认识”）。

在样本数量 N 偏少时，先验极大地影响最终（后验）概率。可以认为，当观察样本量稀少时，加上先验可以提供比较客观（基于统计学）的结果。（比如有出租车撞人，目击者声称肇事车是蓝色的，然而城市中出租车颜色绿色远多于蓝色，最后计算结果仍旧是肇事车为绿色的概率更大）。

先验的方差 σ 越大，则分布函数越“平”，提供的信息量越少，对观察结果的影响也越小甚至趋向MLE。

样本数量 N 越大，观察序列占得比重越高，先验逐渐失去权重（一枚灌铅的硬币不论怎么抛就是永远正面朝上，原先认为正反面各 $\frac{1}{2}$ 的预设就破产了）。MAP与MLE的差异也就越小直至0。

四、预测

对于MLE, MAP, θ 为一个固定的值，新数据概率为 $p(x_{new}|\theta)$ ；

对于全贝叶斯分布， θ 在每一处均不同。

$$p(f|D, a, b) = \int_0^1 p(f|\theta) \cdot p(\theta|D, a, b) d\theta$$

观察序列

全贝叶斯预测

$p(D \theta) = \theta^{ T } (1-\theta)^{ H }$	$p(f D, a, b) = Ber\left(f \frac{ T +a}{ T +a+ H +b}\right)$	相当于求此Beta函数的期望
---	--	----------------

$$p(D|\mu) = \prod_{i=1}^n \mathbb{N}(x|\mu, 1) \quad \mathcal{N}(x|m, 1 + \beta^{-1})$$

$$\begin{aligned} & p(x_{new}|D, \alpha) \\ &= \int p(x|\mu) \cdot p(\mu|D, \alpha) d\mu \\ &= \int \mathcal{N}(x|\mu, 1) \cdot \mathcal{N}(\mu|m, \beta^{-1}) d\mu \\ & \frac{1}{\sqrt{2\pi}} \exp(-\frac{1}{2}(x-\mu)^2) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{1}{2}((x-\mu)-0)^2) = \int \mathcal{N}(x-\mu|0, 1) \cdot \mathcal{N}(\mu|m, \beta^{-1}) d\mu \\ & \text{令 } y = x - \mu \rightarrow \mathcal{N}(y|0, 1) \\ & x = y + \mu \rightarrow \mathcal{N}(x|0+m, 1+\beta^{-1}) = \mathcal{N}(x|m, 1+\beta^{-1}) \end{aligned}$$

线性回归

一、定义

当一个输入有D个分量，称为D个维度Dimension，找到一个线性系数向量 \mathbf{w} ，使得原输出
 $y \approx f_w(x_i)$

$$\begin{aligned} \tilde{\mathbf{x}}_i &= (1, x_{i1}, \dots, x_{iD})^T \\ \tilde{\mathbf{w}} &= (w_0, w_1, \dots, w_D)^T \\ \rightarrow f_w(\mathbf{x}_i) &= \tilde{\mathbf{w}}^T \tilde{\mathbf{x}}_i \end{aligned}$$

i 代表第 i 个 x

二、如何选取最佳 \mathbf{w} ？

损失函数： $E_{LS}(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^N (f_w(x_i) - y_i)^2$ ，最小平方和least square

$\frac{1}{2}$ 只是个系数，方便之后运算

最佳系数向量：

$$\begin{aligned} w^* &= \arg \min_w E_{LS}(w) \\ \text{仅一维} &= \arg \min_w \frac{1}{2} \sum_{i=1}^N (f_w(x_i) - y_i)^2 \\ \text{可多维, } \mathbf{x} \in \mathbb{R}^{N \times D} &= \arg \min_w \frac{1}{2} (\mathbf{X}\mathbf{w} - \mathbf{y})^T (\mathbf{X}\mathbf{w} - \mathbf{y}) \end{aligned}$$

如何求解？ \rightarrow 令梯度为0，类似求导：

$$\begin{aligned}\nabla_W E_{LS}(\mathbf{w}) &= \mathbf{X}^T \mathbf{X} \mathbf{w} - \mathbf{X}^T \mathbf{y} \doteq 0 \\ \mathbf{w}^* &= (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \\ &\triangleq \mathbf{x}^\dagger, \text{ 称为伪逆矩阵}\end{aligned}$$

三、非线性回归

(一)、一维数据

一维数据情况下可以用多项式polynomial来逼近非线性的数据比如sinx

$$f_w(x) = w_0 + \sum_{j=1}^M w_j x^j, \quad \text{对 } x \text{ 做 } M \text{ 次变换并乘以系数 } w \text{ 再相加}$$

此时对于x已经不是线性，但对于w仍为线性，依旧可以叫“线性”回归

(二)、多维数据

$$\text{多维情况下，更一般地， } f_w(\mathbf{x}) = w_0 + \sum_{j=1}^M w_j \phi_j \stackrel{(\phi_0=1)}{=} \mathbf{w}^T \phi(\mathbf{x}), \text{ 其中，}$$

ϕ_j 为 $\mathbb{R}^D \rightarrow \mathbb{R}$ 的函数, $\phi(x) \in \mathbb{R}^{M+1}$

损失函数为：

$$\begin{aligned}E_{LS} &= \frac{1}{2} \sum_{i=1}^N (\mathbf{w}^T \phi(\mathbf{x}_i) - y_i)^2 \\ &= \frac{1}{2} (\Phi \mathbf{w} - \mathbf{y})^T (\Phi \mathbf{w} - \mathbf{y}) \\ \mathbf{w}^* &= (\Phi^T \Phi)^{-1} \Phi^T \mathbf{y} \quad \text{与不添加 } \Phi(X) \text{ 类似}\end{aligned}$$

其中， Φ 称为设计矩阵Design Matrix,

$$\Phi = \begin{pmatrix} \phi_0(x_1) & \phi_1(x_1) & \cdots & \phi_M(x_1) \\ \phi_0(x_2) & \phi_1(x_2) & & \vdots \\ \vdots & \vdots & \ddots & \\ \phi_0(x_N) & \phi_1(x_N) & \cdots & \phi_M(x_N) \end{pmatrix} \in \mathbb{R}^{N \times (M+1)}$$

在一维线性回归中, $\Phi = [1, \mathbf{x}]$, $\phi_0 = 1, \phi_j = x$

(三)、惩罚规则Regularization

然而, 当w系数过大时常导致过拟合(过度贴合数据中的噪音) → 引入惩罚规则

$$\begin{aligned}E_{ridge}(w) &= \frac{1}{2} \sum_{i=1}^N (\mathbf{w}^T \phi(\mathbf{x}_i) - y_i)^2 + \frac{\lambda}{2} \|w\|_2^2 \\ &= \frac{1}{2} (\Phi \mathbf{w} - \mathbf{y})^T (\Phi \mathbf{w} - \mathbf{y}) + \frac{\lambda}{2} \mathbf{w}^T \mathbf{w} \\ \nabla_w E &= \Phi^T \Phi \mathbf{w} - \Phi^T \mathbf{y} + \lambda \mathbf{w} \doteq 0 \\ \therefore \mathbf{w}^* &= (\Phi^T \Phi + \lambda \mathbf{I})^{-1} \Phi^T \mathbf{y}\end{aligned}$$

- $\|w\|_2^2$: w各项平方和= $\mathbf{w}^T \mathbf{w}$

- λ : 惩罚力度

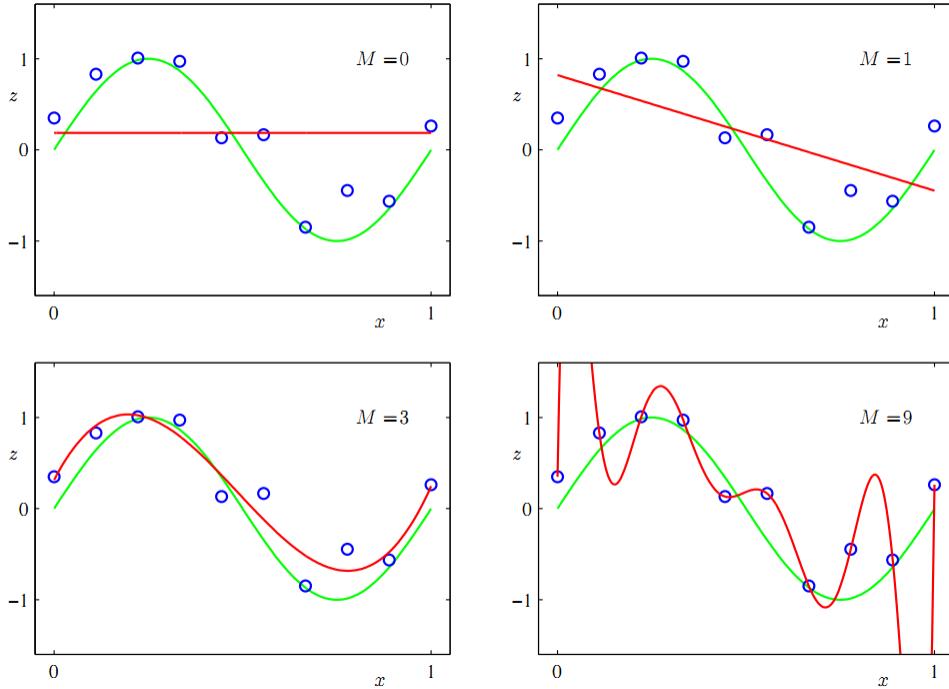
通常选一个高维度的 j (确保至少不会欠拟合), 并用 λ 控制, 防止过拟合

(四)、线性回归模型评价

Bias: 模型与数据图形相似度。若模型维度太低，欠拟合则Bias高。

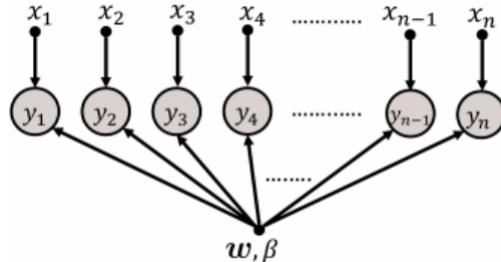
Variance: 预测值变化剧烈程度，过拟合时Variance高

当且仅当模型临界拟合数据集时，Bias与Variance均低。



四、正态分布序列与线性回归的关联

(一)、MLE



$$y_i \sim \mathcal{N}(f_w(\mathbf{x}_i), \beta^{-1})$$

- 序列的概率即为 $p(\mathbf{y}|\mathbf{X}, \mathbf{w}, \beta) = \prod_i p(y_i|f_w(x_i), \beta)$

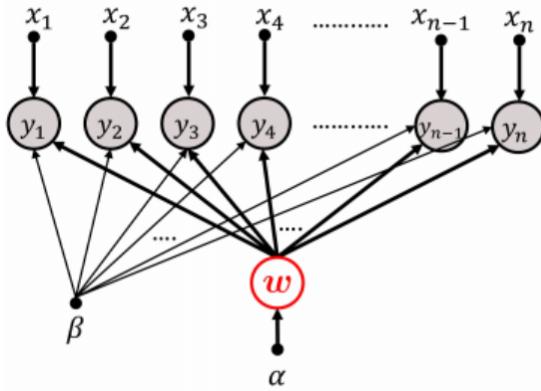
$$\begin{aligned} \mathbf{w}_{MLE}, \beta_{MLE} &= \arg \max_{\mathbf{w}, \beta} p(\mathbf{y}|\mathbf{X}, \mathbf{w}, \beta) \\ &= \arg \min_{\mathbf{w}, \beta} -\ln p(\mathbf{y}|\mathbf{X}, \mathbf{w}, \beta) \\ &= \arg \min_{\mathbf{w}, \beta} -\ln \left[\prod_i \mathcal{N}(y_i|f_w(\mathbf{x}_i), \beta^{-1}) \right] \\ &= \arg \min_{\mathbf{w}, \beta} -\sum_i \ln \left[\sqrt{\frac{\beta}{2\pi}} \exp(-\frac{\beta}{2}(\mathbf{w}^T \phi(\mathbf{x}_i) - y_i)^2) \right] \\ &= \arg \min_{\mathbf{w}, \beta} \frac{\beta}{2} \sum_i (\mathbf{w}^T \phi(\mathbf{x}_i) - y_i)^2 - \frac{N}{2} \ln \beta + \frac{N}{2} \ln 2\pi \end{aligned}$$

$$\mathbf{w}_{MLE} = \arg \min_{\mathbf{w}} \frac{1}{2} \sum_i (\mathbf{w}^T \phi(\mathbf{x}_i) - y_i)^2$$

$$\beta_{MLE} = \arg \min_{\beta} \frac{\beta}{2} \sum_i (\mathbf{w}^T \phi(\mathbf{x}_i) - y_i)^2 - \frac{N}{2} \ln \beta$$

解 \mathbf{w}_{MLE} 恰好就是线性回归的损失函数

(二)、MAP



$$p(\mathbf{w}|\mathbf{X}, \mathbf{y}, \beta, \alpha) \propto p(\mathbf{y}|\mathbf{X}, \mathbf{w}, \beta) \cdot p(\mathbf{w}|\alpha)$$

给 \mathbf{w} 加一个先验 $p(\mathbf{w}|\alpha) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I}) = (\frac{\alpha}{2\pi})^{\frac{M}{2}} \exp(-\frac{\alpha}{2}\mathbf{w}^T\mathbf{w})$, 均值为0 (中心化), 各维度方差相同

β 不加先验, 仍是一个定值

$$\begin{aligned} \mathbf{w}_{MAP} &= \arg \max_{\mathbf{w}} p(\mathbf{w}|\mathbf{X}, \mathbf{y}, \alpha, \beta) = \arg \max_{\mathbf{w}} \frac{p(\mathbf{y}|\mathbf{X}, \mathbf{w}, \beta) \cdot p(\mathbf{w}|\alpha)}{p(\mathbf{X}, \mathbf{y})} \\ &= \arg \max_{\mathbf{w}} \ln p(\mathbf{y}|\mathbf{X}, \mathbf{w}, \beta) + \underbrace{\ln p(\mathbf{w}|\alpha) - \ln p(\mathbf{X}, \mathbf{y})}_{\text{常数}} \\ &= \arg \min_{\mathbf{w}} - [\ln p(\mathbf{y}|\mathbf{X}, \mathbf{w}, \beta) + \ln p(\mathbf{w}|\alpha)] \\ &= \arg \min_{\mathbf{w}} \frac{\beta}{2} \sum_i (\mathbf{w}^T \phi(\mathbf{x}_i) - y_i)^2 - \frac{N}{2} \ln \beta + \frac{N}{2} \ln 2\pi - \ln(\frac{\alpha}{2\pi})^{\frac{M}{2}} + \frac{\alpha}{2} \mathbf{w}^T \mathbf{w} \\ &= \arg \min_{\mathbf{w}} \frac{\beta}{2} \sum_i (\mathbf{w}^T \phi(\mathbf{x}_i) - y_i)^2 + \frac{\alpha}{2} \mathbf{w}^T \mathbf{w} \\ &= \arg \min_{\mathbf{w}} \frac{1}{2} \sum_i (\mathbf{w}^T \phi(\mathbf{x}_i) - y_i)^2 + \frac{\lambda}{2} \|\mathbf{w}\|_2^2 \end{aligned}$$

因此高斯分布加上一个对 \mathbf{w} 的先验 (\mathbf{w} 均值为0, 各向同性), w_{MAP} 既是使序列出现概率最大, 也使带Ridge的线性回归损失函数最小。 $\lambda = \frac{\alpha}{\beta} = \frac{\text{先验的方差倒数}}{\text{数据分布的方差倒数}}$, 即线性回归中加一个惩罚项 λ 就是在序列上加一个先验 $p(w)$

(三)、Sequential 贝叶斯线性回归

如果数据 D_2, D_1 不能同时获得, 我们可以计算

$$p(\mathbf{w}|D_2, D_1) \propto p(D_2|\mathbf{w})p(D_1|\mathbf{w})p(\mathbf{w}|\alpha) \propto p(D_2|\mathbf{w}) \cdot p(\mathbf{w}|D_1)$$

即有先验 $p(\mathbf{w}|\alpha)$ 后, 先计算 \mathbf{w} 对 D_1 的后验 $p(\mathbf{w}|D_1)$ 。获得 D_2 数据后, 将对 D_1 的后验作为此时的先验, 计算 $p(D_2|\mathbf{w}) \cdot p(\mathbf{w}|D_1)$, 等于 \mathbf{w} 对 D_2, D_1 的后验。

线性分类

一、hard decision based model

损失函数为 zero-one loss

$$l_{01}(y, \hat{y}) = \sum_{i=1}^N \mathbb{I}(\hat{y} \neq y_i)$$

硬边界二元分类——perception函数：

$$y = \begin{cases} 1, & \text{when } \omega^T \mathbf{x} + \omega_0 > 0 \\ 0, & \text{when } \omega^T \mathbf{x} + \omega_0 < 0 \end{cases}$$

二、generative model

$$\begin{aligned} \mathbf{y}_{new} &\sim p(\mathbf{y}|\hat{\theta}) \\ \mathbf{x}_{new} &\sim p(\mathbf{x}|y=y_{new}, \hat{\psi}) \end{aligned}$$

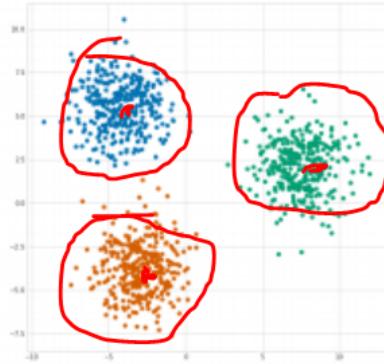
- $\vec{\theta} \in \mathbb{R}^c$ 代表每个类别的概率
- $\theta_c \in [0, 1], \sum_{c=1}^C \theta_c = 1$

此时， $\theta_c^{MLE} = \frac{N_c}{N} = \frac{\text{种类}C\text{的数量}}{\text{总数}}$

(一)、LDA (Linear discriminant analysis)

每个类别的期望输入值：自身平均值，和一个公用的协方差矩阵的正态分布

$$p(\mathbf{x}|y=c) = \mathcal{N}(\mathbf{x}|\mu_c, \Sigma)$$



1. 二元分类：

$$\begin{aligned} p(y=1|\mathbf{x}) &\equiv \frac{p(\mathbf{x}|y=1)p(y=1)}{p(\mathbf{x}|y=1)p(y=1) + p(\mathbf{x}|y=0)p(y=0)} \\ &= \frac{1}{1 + \exp(-\alpha)} \\ &= \text{sigmoid } \sigma(\alpha) \end{aligned}$$

即 $y|\mathbf{x} \sim \text{Bernoulli}(\sigma(\mathbf{w}^T \mathbf{x} + w_0))$

其中， $\alpha = \ln \frac{p(\mathbf{x}|y=1)p(y=1)}{p(\mathbf{x}|y=0)p(y=0)}$ → (一通数学运算) $= \mathbf{w}^T \mathbf{x} + w_0$ 是一个线性分割，
 $\mathbf{w} = \Sigma^{-1}(\mu_1 - \mu_0)$

当 $\alpha > 0$ 时，标签判定为 1

2. 多元分类

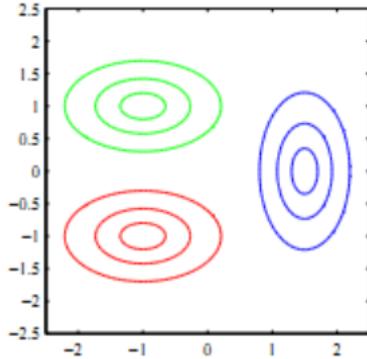
$$\begin{aligned} p(y=c|\mathbf{x}) &= \frac{p(\mathbf{x}|y=c)p(y=c)}{\sum_{ci=1}^C p(\mathbf{x}|y=ci)p(y=ci)} \\ &= \frac{\exp(\mathbf{w}^T \mathbf{x} + w_{c0})}{\sum_{ci=1}^C \exp(\mathbf{w}_{ci}^T \mathbf{x} + w_{ci0})} \\ &= \text{softmax } \sigma(\alpha) \end{aligned}$$

softmax输出一个向量，各元素为各类别的概率，比如 $p_i = \begin{bmatrix} 0.8 \\ 0.1 \\ 0.1 \end{bmatrix}$

LDA生成线性的分类边界

(二)、naive 贝叶斯

不同于LDA，让每个标签的数据有单独的协方差矩阵 Σ



为了简化运算，假设 $\mathbf{x} = (x_0, x_1, \dots, x_n)$ 互相独立，当 \mathbf{x} 的概率正态分布时， Σ 为对角阵（特征间不能有关联）

$$p(x_1, x_2, \dots, x_d | y = c) = \prod_{i=1}^N p(x_i | y = c)$$
$$p(\mathbf{x} | y = c) = \mathcal{N}(\mathbf{x} | \mu_c, \Sigma_c)$$

独立 Σ 好处：可以更容易处理不同的分布，以及不同的数据类型

Naive贝叶斯生成方形的分类边界

三、discrimitive model 或 Logistic regression 逻辑回归

generative model 生成模型 不能自主选择 \mathbf{w} 与 w_0 ，而是由数据决定（好处是可以生成新的数据，在数据缺失时比较有用）

(一)、区分模型推导

直接给 $y|\mathbf{x} \sim \text{Bern}(\sigma(\mathbf{w}^T \mathbf{x} + w_0))$ 建模，自由选择 \mathbf{w} 与 w_0 （概率模型，可以有错分）

$$p(\mathbf{y}|\mathbf{w}, \mathbf{X}) = \prod_{i=1}^N p(y_i | \mathbf{x}_i, \mathbf{w})$$
$$(\text{二元分类}) = \prod_i \underbrace{p(\hat{y} = 1 | \mathbf{x}_i, \mathbf{w})^{y_i}}_{\hat{y}_i = 1 \text{的概率, } y_i = 0 \text{ 时为1消失}} \cdot \underbrace{(1 - p(\hat{y} = 1 | \mathbf{x}_i, \mathbf{w}))^{1-y_i}}_{\hat{y}_i = 0 \text{ 的概率, } y_i = 1 \text{ 时消失}}$$

损失函数：负对数概率，即cross entropy

$$\begin{aligned}
E(\mathbf{w}) &= -\ln p(\mathbf{y}|\mathbf{w}, \mathbf{X}) \\
(\text{二元}) &= -\sum_{i=1}^N [y_i \ln \sigma(\mathbf{w}^T \mathbf{x}_i) + (1 - y_i) \ln(1 - \sigma(\mathbf{w}^T \mathbf{x}_i))] \\
(\text{多元}) &= -\sum_i^N \sum_c^C y_{ic} \ln \frac{\exp(\mathbf{w}_c^T \mathbf{x})}{\sum_{ci} \exp(\mathbf{w}_{ci}^T \mathbf{x})} \\
&= -\sum_i \ln \underbrace{\hat{p}_{ik(i)}}_{\text{正确类概率}} \\
&= -\sum_i \ln \text{softmax}(\mathbf{W} \cdot \mathbf{x}_i)
\end{aligned}$$

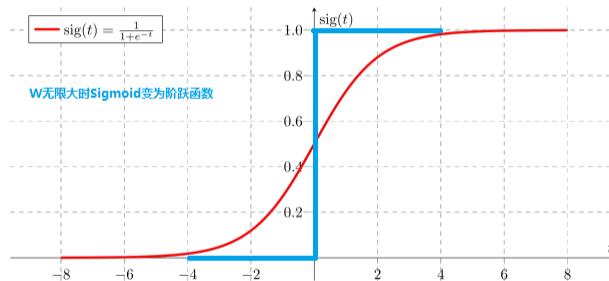
即只将正确的分类的概率相加

$$\mathbf{w}^* = \arg \min_w E(\mathbf{w})$$

(二)、一个小问题

由于逻辑回归中 \mathbf{w} (假设包含 w_0) 是自己训练出来的，因此可能过拟合。

回顾损失函数 $E = -\sum_{i=1}^N [y_i \ln \sigma(\mathbf{w}^T \mathbf{x}_i) + (1 - y_i) \ln(1 - \sigma(\mathbf{w}^T \mathbf{x}_i))]$ ，为了使 E 尽可能小， $y_i \ln \sigma(\mathbf{w}^T \mathbf{x}_i)$ 要尽可能大 $\rightarrow \mathbf{w}^T \mathbf{x}_i$ 要最大，才能取到 sigmoid 函数最大值 1 $\rightarrow \mathbf{w}$ 要趋向 $+\infty$ 。
 $\mathbf{w} = +\infty$ 意味着 sigmoid 函数退化成阶跃函数。可能导致过拟合？一般加一个 L2 惩罚项防止 \mathbf{w} 取值过大。



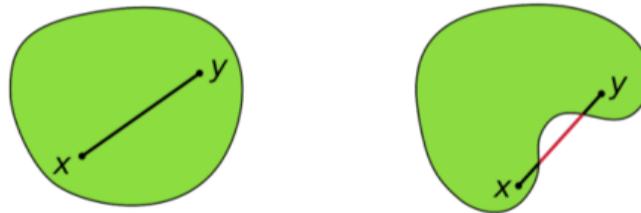
优化

一、数学基础——凸集

(一)、定义

凸集 convex set: 凸集中任意两点连线上的点仍在凸集中

对于 $x, y \in \mathbf{X}$, 有 $\lambda \in [0, 1]$, 使得 $\lambda x + (1 - \lambda)y \in \mathbf{X}$



凸集的顶点 vertex: $\lambda x + (1 - \lambda)y \notin \mathbf{X}, \lambda > 1$, 若凸集中任意点与 x 的连线延伸线不再属于 \mathbf{X} , 则 x 是凸集的顶点。

凸函数 convex function:

抽象函数:	$\lambda f(x_1) + (1 - \lambda)f(x_2) \geq f(\lambda x_1 + (1 - \lambda)x_2) \triangleq f(x_\lambda)$	或写作 $f(\mathbf{y}) - f(\mathbf{x}) \geq (\mathbf{y} - \mathbf{x})^T \nabla f(\mathbf{x})$
解析式函数:	$\frac{d^2 f(x)}{dx^2} \geq 0$	或 $\nabla^2 f(\mathbf{x})$ 为半正定

- 凸函数没有多个极小值，唯一的极小值就是最小值，在导数为0处（方便求最小值问题）
- 凸函数的最大值一定在凸集某个顶点上，即 $\mathbf{x}^* \in Ve\{\mathbf{X}\}$
- 凸集若缺失一块，可以先补成凸集，凸函数最大值在填补上的凸集某个顶点上，即 $\mathbf{x}^* \in Ve\{Conv\{\mathbf{X}\}\}$

(二)、凸函数传递性质

- $h(\mathbf{x}) = f_1(\mathbf{x}) + f_2(\mathbf{x})$, 若 f_1, f_2 为凸，则 h 也为凸
- $h(\mathbf{x}) = \max\{f_1, f_2\}$
- $h = c \cdot f_1, c \geq 0$
- $h = c \cdot g(\mathbf{x}), c \leq 0$, $g(\mathbf{x})$ 为凹函数
- $h = f_1(\mathbf{Ax} + \mathbf{b})$, 线性变化后仍为凸
- $h = m(f_1)$, 当且仅当函数 m 是单调增凸函数

二、迭代寻找导数为0点

θ 为函数自变量， $f(\theta)$ 为在 θ 处函数值。要找到令 $f(\theta)$ 最小值只要求导即可，但实际情况下 $f'(\theta) = 0$ 不那么好找。

(一)、牛顿法(近似法)

将 $f(\theta)$ 泰勒展开：

$$\begin{aligned} f(\vec{\theta}_t + \vec{\delta}) &= f(\vec{\theta}_t) \\ &\quad + \vec{\delta}^T \cdot \nabla f(\vec{\theta}_t) \\ &\quad + \frac{1}{2} \cdot \vec{\delta}^T \nabla^2 f(\vec{\theta}_t) \vec{\delta} \\ &\quad + O(\delta^3) \end{aligned}$$

令 $\nabla_{\delta} f(\theta) = \nabla f(\theta_1) + \delta^T \nabla^2 f(\theta_1) \doteq 0$

所以 $\delta = -\nabla f(\theta_t) [\nabla^2 f(\theta_t)]^{-1}$ 为一次迭代的变化量

$\theta_{t+1} \leftarrow \theta_t + \delta = \theta_t - [\nabla^2 f(\theta_t)]^{-1} \cdot \nabla f(\theta_t)$, 直到收敛

求逆矩阵计算量巨大，牛顿法一般只用于低维度数据

(二)、梯度下降 Gradient descent

通用方法: Exact Line Search

变种一: Backtracking

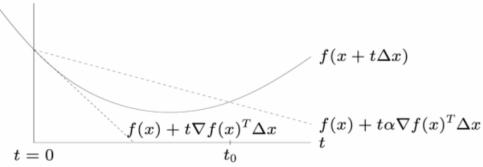
变种二: 直接选取 t

1. $\Delta \vec{\theta} = -\nabla f(\vec{\theta})$	依然 $\vec{\theta} = \vec{\theta} + t \cdot \Delta \vec{\theta}$, 但 t 的选取不同,	$\theta_{t+1} \leftarrow \theta_t - \tau \cdot \nabla f(\theta_t)$ τ 称为学习率
--	---	--

2. $t = \underset{t>0}{\operatorname{argmin}} f(\vec{\theta} + t \cdot \vec{\theta})$ 找梯度方向最小值, 比较慢	有 $\beta \in (0, 1)$, $\alpha \in (0, \frac{1}{2})$, 从 $t_0 = 1$ 开始, 令 $t_{i+1} = \beta \cdot t_i$, 不断缩小 β , 直到 $f(\mathbf{x} + t \cdot \Delta \mathbf{x}) < f(\mathbf{x}) + t \cdot \alpha \cdot \nabla f(\mathbf{x})^T \Delta \mathbf{x}$	τ 太大, 一下子越过目标 τ 太小, 太慢; 还可能在 鞍点上卡住 (非凸函数)
---	--	---

$$3. \vec{\theta} = \vec{\theta} + t \cdot \Delta \vec{\theta}$$

原理: 若 $f(x + t\Delta x)$ 比 $f(x) + t\alpha \nabla f(x)^T \Delta x$ 高, 那么在其下方一定有更好的另一个解



(三)、随机梯度下降 Stochastic GD

针对大数据集, 计算最小化 $\sum_i^n Loss_i(\theta)$ 特别慢

理想情况下, 数据集的一部分 s 个样本能反映整体 n 个样本的特征, 所以有:

$$\begin{aligned} \frac{1}{n} \sum_i^n L_i(\theta_i) &\approx \frac{1}{|s|} \sum_{i \in S} L_i(\theta_i) \\ \text{即 } \sum_i^n L_i(\theta_i) &\approx \frac{n}{|s|} \sum_{j \in S} L_j(\theta_j) \end{aligned}$$

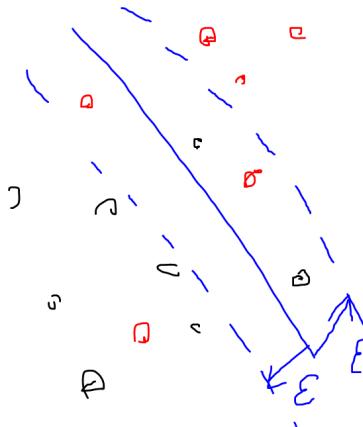
$$\theta_{t+1} \leftarrow \theta_t - \tau \cdot \frac{n}{|s|} \cdot \sum_{j \in S} \nabla L_j(\theta_j)$$

三、随机梯度下降应用——perception 函数

$$y_i = \begin{cases} 1, & \text{if } \mathbf{w}^T \mathbf{x} + b > 0 \\ -1, & \text{else} \end{cases}$$

目标 $\min_{\mathbf{w}, b} \sum_i Loss(\underbrace{y_i}_{u \triangleq y_i}, \underbrace{\mathbf{w}^T \mathbf{x}_i + b}_{v \triangleq \mathbf{w}^T \mathbf{x}_i + b})$, 其中 $u \cdot v = y_i(\mathbf{w}^T \mathbf{x}_i + b)$,

损失函数 $Loss = max(0, \epsilon - u \cdot v)$, ϵ 是一个预设的距离, 为正数



	$Loss$	
在错误一侧	$uv < 0 < \epsilon$	$\epsilon - uv > 0$
在正确一侧但是离分界线太近	$0 < uv < \epsilon$	$\epsilon - uv > 0$
在正确一侧并且离分界线足够远	$uv > \epsilon$	0

$$\nabla_w \max(0, y_i(\mathbf{w}^T \mathbf{x}_i + b)) = \begin{cases} -y_i \cdot \mathbf{x}_i, & \text{if } uv < \epsilon \\ 0, & \text{if } uv \geq \epsilon \end{cases}$$

$$\nabla_b = \begin{cases} -y_i, & uv < \epsilon \\ 0, & uv \geq \epsilon \end{cases}$$

设n为训练数据样本量，取|s|=1（每次更新只选一个点作为数据集），

只要有分类不正确，就更新w与b，直到分类完全正确，不允许有错分

$$\mathbf{w} \leftarrow \mathbf{w} + \tau \cdot n \cdot y_i \cdot \mathbf{x}_i$$

$$\mathbf{b} \leftarrow \mathbf{b} + \tau \cdot n \cdot y_i$$

特殊情况：取 $\tau = \frac{1}{n}$

$$\begin{cases} w = w + x, class1 = 1 \\ w = w - x, class0 = -1 \end{cases}$$

$$\begin{cases} b = b + 1, class1 = 1 \\ b = b - 1, class0 = -1 \end{cases}$$

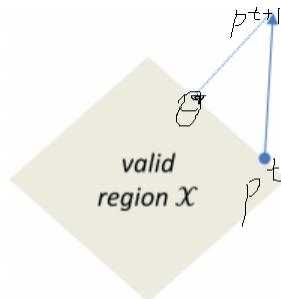
约束优化

一、投影+梯度下降

梯度下降不能用于求约束优化，因为可能会跳出允许的参数域范围→只能将取得的新值投影到参数域

找到 \mathcal{X} 内最近的一点 $\vec{\theta}$

$$\pi_{\mathcal{X}}(\mathbf{p}) = \underset{\vec{\theta} \in \mathcal{X}}{\operatorname{argmin}} \|\vec{\theta} - \mathbf{p}\|$$



最优解一定垂直于允许的参数域表面

二、拉格朗日

定义约束优化问题：

$$f_0, f_1 : \mathbb{R}^d \rightarrow \mathbb{R}$$

$$\underset{\vec{\theta}}{\operatorname{minimize}} f_0(\vec{\theta})$$

$$\text{subject to } f_i(\vec{\theta}) \leq 0$$

结论：最优解处的梯度负数共线所有约束的梯度的线性组合

$$-\nabla f_0(\theta^*) = \sum_i^M \alpha_i \nabla f_i(\theta^*)$$

并且 $\alpha_i \geq 0$

反证法：若 $-\nabla f_0(\theta^*)$ 与 $\nabla f_1(\theta^*)$ 不共线，一定可以写出

$$\begin{aligned} \nabla f_0(\theta^*) &= \alpha \cdot \nabla f_1(\theta^*) + \beta \cdot \mathbf{v}, \mathbf{v} \text{ 是一个和 } \nabla f_1 \text{ 正交的向量} \\ \therefore \nabla f_0(\theta^*) + \epsilon \mathbf{v}^T &\rightarrow (\text{泰勒展开}) \nabla f_0(\theta^*) + \epsilon \cdot \mathbf{v}^T \nabla f_0(\theta^*) \\ &= \nabla f_0(\theta^*) - (\epsilon \mathbf{v}^T \alpha \nabla f_1 + \epsilon \mathbf{v}^T \beta \mathbf{v}) \\ &= \nabla f_0(\theta^*) - \epsilon \beta \|\mathbf{v}\|_2^2 \\ &= \nabla f_0(\theta^*) - \epsilon \beta \|\mathbf{v}\|_2^2 \\ &< \nabla f_0(\theta^*) \text{ 矛盾} \end{aligned}$$

拉格朗日即为

$$L(\theta, \alpha) = f_0(\vec{\theta}) + \sum_i^M \alpha_i f_i(\vec{\theta})$$

$$\text{且 } \alpha_i \geq 0, f_i(\vec{\theta}) \leq 0$$

令拉格朗日梯度等于0：

$$\nabla_{\theta^*} L(\theta, \alpha) = \nabla f_0(\vec{\theta}) + \sum_i^M \alpha_i \nabla f_i(\vec{\theta}) \doteq 0, \text{ 即可算出无约束最小值 } \min_{\theta \in \mathbb{R}} L(\theta, \alpha)$$

但无约束最小值一定比约束最小值更小，是一个下界

$$\begin{aligned} \forall \alpha, \min_{\theta \in \mathbb{R}} L(\vec{\theta}, \alpha) &\leq L(\vec{\theta}^*, \alpha) = f_0(\theta^*) + \sum_i^M \alpha_i f_i(\theta^*) \\ (\because a_i \geq 0, f_i(\vec{\theta}) \leq 0, \therefore \sum_i^M \alpha_i f_i(\theta^*) \leq 0) \\ &\leq f_0(\theta^*) = \text{受约束的 } \min_{f_i(\theta) \leq 0} f_0(\theta) \end{aligned}$$

三、 dual problem

设拉格朗日无约束最小值 $g(\alpha) = \min_{\theta \in \mathbb{R}} L(\vec{\theta}, \alpha) \triangleq d^*$, 是一个concave函数

真正的受约束最小值记为 p^*

在Slater's constraint qualification 充分非必要条件下， $\exists \alpha$, 使得 $d^* = p^*$

1. f_i 均为convex，并且.....

2. 有一处 $\vec{\theta}$,

(a) 约束均为线性，即 $f_i(\theta) = \mathbf{w}_i^T \theta + b_i$

或 (b) 严格小于 $f_i(\theta) < 0$

此时，求解 $\max(g(\alpha))$ 即 $= p^*$

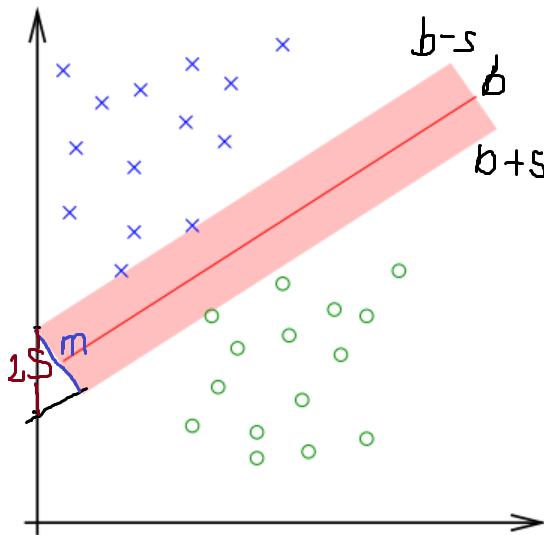
四、 KKT推论

$$\begin{aligned}
\exists \alpha, \min_{\theta \in \mathbb{R}} L(\vec{\theta}, \alpha) = L(\vec{\theta}^*, \alpha) &= f_0(\theta^*) + \sum_i^M \alpha_i f_i(\theta^*) = f_0(\theta^*) = \min_{f_i(\theta) \leq 0} f_0(\theta) \\
\therefore f_0(\theta^*) + \sum_i^M \alpha_i f_i(\theta^*) &= f_0(\theta^*) \\
\rightarrow \sum_i^M \alpha_i^* f_i(\theta^*) &= 0 \\
\because \alpha_i^* f_i(\theta^*) \leq 0 & \\
\rightarrow \alpha_i^* f_i(\theta^*) = 0 & \text{(complementary slackness)}
\end{aligned}$$

SVM

一、硬边界SVM

找到一条线分割2元数据，同时保证空白区间最大



$$\begin{cases} \mathbf{w}^T \mathbf{x} + b \geq s, & \text{当分类为1类} \\ \mathbf{w}^T \mathbf{x} + b \leq -s, & \text{当分类为0类} \end{cases}$$

\therefore 空白区间 $margin = \frac{2s}{\|\mathbf{w}\|}$, 通常直接令 $s = 1$, 这样只需优化 \mathbf{w}

$$\text{即 } margin = \frac{2}{\|\mathbf{w}\|} = \frac{2}{\mathbf{w}^T \mathbf{w}}$$

如果设1类标签值为1, 0类标签值为-1, 上式可以化简为

$$y_i(\mathbf{w}^T \mathbf{x} + b) - 1 \geq 0$$

因此优化问题为:

$$\begin{aligned}
f_0 &= \frac{1}{2} \mathbf{w}^T \mathbf{w} \\
\text{约束 } f_i &= 1 - y_i(\mathbf{w}^T \mathbf{x} + b) \leq 0
\end{aligned}$$

拉格朗日优化问题即转为:

$$L(\mathbf{w}, b) = \frac{1}{2} \mathbf{w}^T \mathbf{w} + \sum_i a_i [1 - y_i(\mathbf{w}^T \mathbf{x} + b)]$$

求导得:

$$\nabla_{\mathbf{w}} L(\mathbf{w}, b, \vec{\alpha}) = \mathbf{w} - \sum_i \alpha_i y_i \mathbf{x}_i \doteq 0$$

$$\nabla_b L = \sum_i \alpha_i y_i \doteq 0$$

$\therefore \mathbf{w}$ 最优值 $\mathbf{w}_{\vec{\alpha}}^* = \sum_i \alpha_i y_i \mathbf{x}_i$, 是标签值与数据值的线性组合, 将 \mathbf{w} 代回, 得 L 无约束最小值

$$\begin{aligned} L &= \frac{1}{2} \mathbf{w}^T \mathbf{w} + \sum_i a_i - \sum_i a_i y_i (\mathbf{w}^T \mathbf{x} + b) \\ &\stackrel{\mathbf{w}_{\vec{\alpha}}^* = \sum_i \alpha_i y_i \mathbf{x}_i}{=} \frac{1}{2} \sum_i \sum_j y_i y_j \alpha_i \alpha_j \mathbf{x}_i^T \mathbf{x}_j + \sum_i a_i - \sum_i a_i y_i (\sum_j a_j y_j \mathbf{x}_j) \mathbf{x}_i - \sum_i a_i y_i b \\ &\stackrel{\sum_i \alpha_i y_i = 0}{=} \frac{1}{2} \sum_i \sum_j y_i y_j \alpha_i \alpha_j \mathbf{x}_i^T \mathbf{x}_j - \sum_i a_i y_i b \end{aligned}$$

dual problem: 最大化拉格朗日无约束最小值

$$\begin{aligned} \text{最大化 } g(\vec{\alpha}) &= \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j y_i y_j \alpha_i \alpha_j \mathbf{x}_i^T \mathbf{x}_j \\ \text{约束 } &\begin{cases} 1. \sum_i \alpha_i y_i = 0 \\ 2. \alpha_i \geq 0 \end{cases} \end{aligned}$$

矩阵形式即为:

$$\begin{aligned} \text{最大化 } g(\vec{\alpha}) &= \frac{1}{2} \vec{\alpha}^T \mathbf{Q} \vec{\alpha} + \vec{\alpha}^T \mathbf{1}_N \\ \text{其中, } \mathbf{Q} &= -(\mathbf{y} \odot \mathbf{X}) \cdot (\mathbf{y} \odot \mathbf{X})^T = -\mathbf{y} \mathbf{y}^T \odot \mathbf{X} \mathbf{X}^T \end{aligned}$$

由上一章KKT slackness推论:

$$\sum_i a_i [1 - y_i (\mathbf{w}^T \mathbf{x} + b)] = 0$$

$\alpha_i \neq 0$ 处, $y_i (\mathbf{w}^T \mathbf{x} + b) = 1 \implies \mathbf{w}^T \mathbf{x} + b = \frac{1}{y_i} = y_i, (\because y_i \in \{1, -1\})$

只有那些恰好在边缘上的点可以决定分割线的走向, 因此 $\alpha_i \neq 0$ 处的点被称为 support vector

$$\therefore \frac{b = \text{average}(y_i - \mathbf{w}^T \mathbf{x})}{\mathbf{w} = \sum_{i \in S} \alpha_i y_i \mathbf{x}} \quad \left| \begin{array}{l} \text{取平均值更稳定} \\ \mathbf{w} \text{ 实际上只取决于支持向量} \end{array} \right.$$

二、模糊边缘SVM

加一个slack variable $\xi_i \geq 0$

$$\begin{cases} \mathbf{w}^T \mathbf{x} + b \geq +1 - \xi_i, & \text{当 } y_i = +1 \\ \mathbf{w}^T \mathbf{x} + b \leq -1 + \xi_i, & \text{当 } y_i = -1 \end{cases}$$

同样可化简为 $y_i (\mathbf{w}^T \mathbf{x} + b) \geq 1 - \xi_i$

约束优化问题即为:

$$\begin{aligned} f_0 &= \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \cdot \sum_i \xi_i \\ \text{约束 } &\begin{cases} 1. y_i (\mathbf{w}^T \mathbf{x} + b) - 1 + \xi_i \geq 0 \\ 2. \xi_i \geq 0 \end{cases} \end{aligned}$$

常数 C 的作用: $C \rightarrow \infty$ 容忍误差为 0, 当于硬边界; $C \rightarrow 0$ 相当于误差范围无限大, 没有意义

拉格朗日为：

$$\begin{aligned} L(\mathbf{w}, b, \xi) &= \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \cdot \sum_i \xi_i \\ &\quad + \sum_i \alpha_i [1 - \xi_i - y_i (\mathbf{w}^T \mathbf{x} + b)] \\ &\quad + \sum_i \mu_i (-\xi_i) \end{aligned}$$

求导得：

$$\begin{aligned} \nabla_{\mathbf{w}} L(\mathbf{w}, b, \vec{\alpha}) &= \mathbf{w} - \sum_i \alpha_i y_i \mathbf{x}_i \doteq 0 \\ \nabla_b L &= \sum_i \alpha_i y_i \doteq 0 \\ \frac{\partial L}{\partial \xi_i} &= C - \alpha_i - \mu_i \doteq 0 \implies \alpha_i = C - \mu_i \in [0, C] \end{aligned}$$

拉格朗日即转为：

$$\begin{aligned} \text{最大化 } g(\vec{\alpha}) &= \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j y_i y_j \alpha_i \alpha_j \mathbf{x}_i^T \mathbf{x}_j \\ \text{约束} &\left\{ \begin{array}{l} 1. \sum_i \alpha_i y_i = 0 \\ 2. 0 \leq \alpha_i \leq C \end{array} \right. \end{aligned}$$

但实际上，有更简单的方法

\because 最优情况下，若一个点落在空白区间内部，都应该调整 ξ 使其正好落在新的边界上

$$\begin{aligned} \xi &= \begin{cases} 1 - y_i (\mathbf{w}^T \mathbf{x} + b), & \text{如果 } y_i (\mathbf{w}^T \mathbf{x} + b) < 1 \\ 0, & \text{else} \end{cases} \\ \xi &= \max \{0, 1 - y_i (\mathbf{w}^T \mathbf{x} + b)\} \end{aligned}$$

此时原来的两个约束自动满足

$$\begin{aligned} 1. y_i (\mathbf{w}^T \mathbf{x} + b) - 1 + \xi_i &\geq 0 \\ 2. \xi_i &\geq 0 \end{aligned}$$

\therefore 原问题转为无约束问题

$$\min \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \cdot \underbrace{\sum_i \max \{0, 1 - y_i (\mathbf{w}^T \mathbf{x} + b)\}}_{\epsilon=1 \text{ 的 perception 函数}}$$

模糊SVM损失函数后半段 $\sum_i \max \{0, 1 - y_i (\mathbf{w}^T \mathbf{x} + b)\}$ 实际是预设距离 $\epsilon = 1$ 的 perception 函数，然而SVM在此基础上还要让空白区间最大($\frac{1}{2} \mathbf{w}^T \mathbf{w}$)，因此是进化版

三、软边界SVM与逻辑回归的关联

SVM是典型的二分类器，而逻辑回归可二元分类，也可多元分类。因此必须把问题限定在二分类。

在之前的章节中，逻辑回归（二元）两个标签为{1,0}，因此损失函数推导过程为

$$\begin{aligned}
p(y_i = 0) &= 1 - \sigma(\mathbf{w}^T \mathbf{x}_i + b) \\
p(y_i = 1) &= \sigma(\mathbf{w}^T \mathbf{x}_i + b) \\
\therefore p(\mathbf{y}) &= \prod_i \sigma(\mathbf{w}^T \mathbf{x}_i + b)^{y_i} \cdot (1 - \sigma(\mathbf{w}^T \mathbf{x}_i + b))^{(1-y_i)} \\
E &= -\ln p(\mathbf{y}) = -\sum_i y_i \cdot \ln \sigma(\mathbf{w}^T \mathbf{x}_i + b) + (1 - y_i) \ln[1 - \sigma(\mathbf{w}^T \mathbf{x}_i + b)]
\end{aligned}$$

如果标签像SVM一样设为{1, -1}，则损失函数可以写成比较简单的形式

$$\begin{aligned}
p(y = -1) &= 1 - \sigma(\mathbf{w}^T \mathbf{x}_i + b) = \sigma(-(\mathbf{w}^T \mathbf{x}_i + b)) \\
p(y_i = 1) &= \sigma(\mathbf{w}^T \mathbf{x}_i + b) \\
p(\mathbf{y}) &= \prod_i \sigma(y_i(\mathbf{w}^T \mathbf{x}_i + b)) = \frac{1}{1 + y_i(\mathbf{w}^T \mathbf{x}_i + b)} \\
E &= -\ln p(\mathbf{y}) = -\sum_i -\ln(1 + e^{-y_i(\mathbf{w}^T \mathbf{x}_i + b)}) = \sum_i \ln(1 + e^{-y_i(\mathbf{w}^T \mathbf{x}_i + b)})
\end{aligned}$$

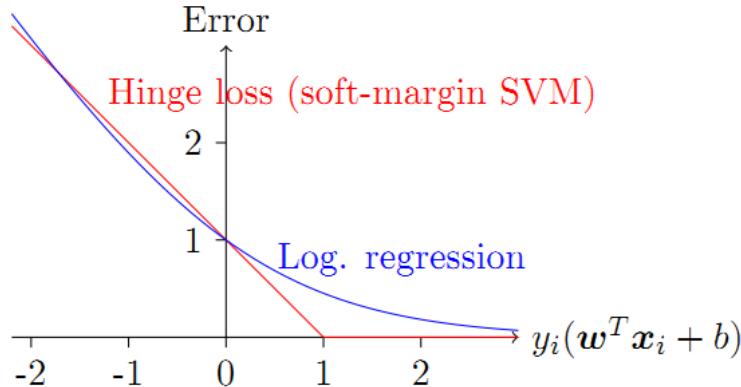
在此基础上加上一个L2惩罚，就得到标签为{1, -1}时逻辑回归损失函数：

$$E_{\text{逻辑回归}} = \sum_i \ln(1 + e^{-y_i(\mathbf{w}^T \mathbf{x}_i + b)}) + \lambda \mathbf{w}^T \mathbf{w}$$

和SVM的损失函数比较：

$$\begin{aligned}
E_{SVM} &= C \cdot \sum_i \max \{0, 1 - y_i(\mathbf{w}^T \mathbf{x}_i + b)\} + \frac{1}{2} \mathbf{w}^T \mathbf{w} \\
&= C \left[\sum_i \text{Hinge}(y_i(\mathbf{w}^T \mathbf{x}_i + b)) + \underbrace{\frac{1}{2C} \mathbf{w}^T \mathbf{w}}_{\lambda} \right]
\end{aligned}$$

至此可以清晰地看出两者的关系：模糊边界SVM损失函数是Hinge铰链函数，而逻辑回归损失函数是 $\ln(1 + e^{-x})$



四、kernel

$\text{kernel}(x, y) = \phi(x)^T \cdot \phi(y) \in \mathbb{R}^D \times \mathbb{R}^D \rightarrow \mathbb{R}$, 对应x, y映射到一个feature space后的内积。

离散kernel：一个包含所有输入数据的对称、半正定矩阵 \mathbf{K} (Gram矩阵)。对称、半正定的条件是为了保证优化问题是一个凸函数，否则可能找不到全局最优解。

$$\mathbf{K} = \begin{pmatrix} k(x_1, x_1) & k(x_1, x_2) & \cdots & k(x_1, x_n) \\ k(x_2, x_1) & \ddots & & \vdots \\ \vdots & & \ddots & \\ k(x_n, x_1) & & & k(x_n, x_n) \end{pmatrix}$$

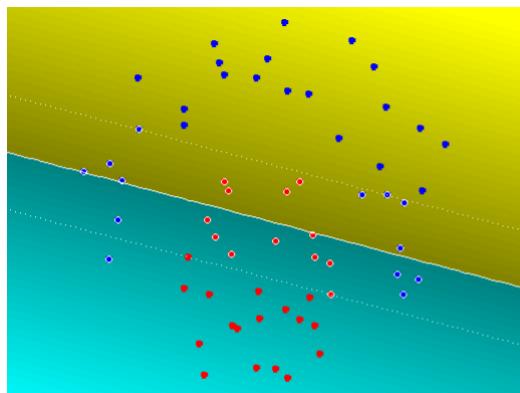
kernel性质:

1. $k(x_1, x_2) = k_1(x_1, x_2) + k_2(x_1, x_2)$, 其中 $k_1, k_2 \in \mathbb{R}^{N \times N} \rightarrow \mathbb{R}$
2. $k(x_1, x_2) = k_1(x_1, x_2) \times k_2(x_1, x_2)$
3. $k(x_1, x_2) = C \cdot k_1(x_1, x_2), C > 0$
4. $k(x_1, x_2) = \mathbf{x}_1 \mathbf{A} \mathbf{x}_2, \mathbf{A} \in \mathbb{R}^{N \times N}$, 为对称、半正定矩阵
5. $k(x_1, x_2) = k_3(\phi(x_1), \phi(x_2))$, 其中 $k_3 \in \mathbb{R}^M, \phi \in \mathbb{R}^N \rightarrow \mathbb{R}^M$

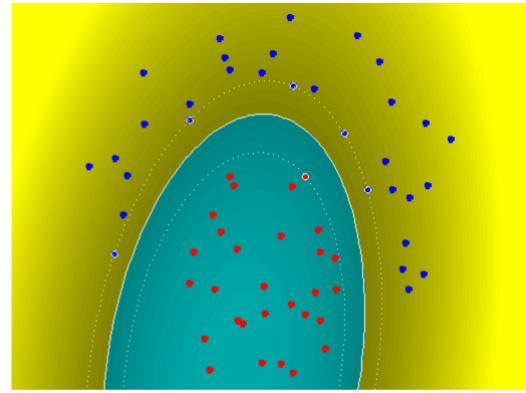
常用kernel:

1. 多项式: $k(\mathbf{a}, \mathbf{b}) = (\mathbf{a}^T \mathbf{b})^p$ 或 $(\mathbf{a}^T \mathbf{b} + 1)^p$
2. 高斯: $k(\mathbf{a}, \mathbf{b}) = \exp\left(-\frac{\|\mathbf{a}-\mathbf{b}\|^2}{2\sigma^2}\right)$
3. Sigmoid: $k(\mathbf{a}, \mathbf{b}) = \tanh \kappa \mathbf{a}^T \mathbf{b} - \delta$, 其中 $\kappa, \delta > 0$

当 $g(\alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j y_i y_j \alpha_i \alpha_j \mathbf{x}_i^T \mathbf{x}_j$ 时, 分类边界是一条直线。替换 $\mathbf{x}_i^T \mathbf{x}_j$ 为 $\phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) = k(x_i, x_j)$ 可以把数据映射到高维空间, 让 SVM 生成非线性分界边界。



Linear kernel (no kernel)
 $\mathbf{a}^T \mathbf{b}$



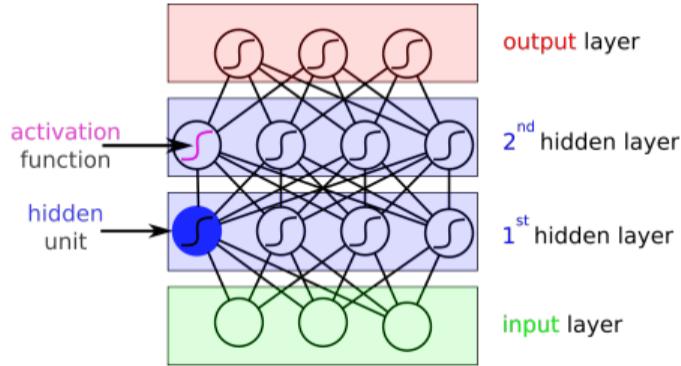
2nd order polynomial kernel
 $(\mathbf{a}^T \mathbf{b})^2$

深度学习

回想逻辑回归, 当遇到不可直接线性分割的数据集时, 通常用一个Basic fuction $\phi(\mathbf{X}) : \mathbb{R}^{N \times D} \rightarrow \mathbb{R}^{N \times M}$ 把数据集投影到一个可以线性分割的空间中

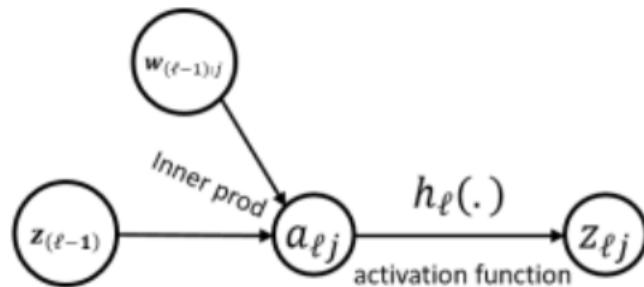
一、神经网络

一层隐藏层的神经网络: 如果只知道数据集 \mathbf{X} 和标签 \mathbf{y} , 不知道 $\phi()$, 就把 basis function 的各个参数也纳入优化范围。



每一层内部是线性的，但输出结果先经过一个非线性函数再给到下一层，否则几个线性函数叠在一起还是一个线性函数。

<i>neuron</i>	w_{lij}	第 l 层，第 <i>i</i> 输入，第 <i>j</i> 输出
	$z_{li}^{(n)}$	第 <i>n</i> 样本，第 <i>l</i> 层，第 <i>i</i> 分量
<i>logit</i>	$\mathbf{a}_{li}^{(n)}$	$a_{li}^{(n)} = \mathbf{W}_{l-1} \cdot \mathbf{z}_{l-1}^{(n)}$
激活函数	h_l	$z_{lj} = h_l(a_{lj})$



正向传播到损失函数的最后一个激活函数一般定义为 $h(a) = a$

二、反向传播

以分类问题举例：神经网络对一个样本会给出一个标签，汇集所有标签可以评价损失度（损失函数是交叉熵）。通过第一次评价不断优化矩阵 \mathbf{W} ，达到最好的效果。这是一个用梯度下降找最小值的问题。因此需要推导损失度对 \mathbf{W} 中各个参数的导数。

$$\begin{aligned}\frac{\partial E}{\partial w_{(l-1)ij}} &= \frac{\partial E}{\partial a_{lj}} \cdot \frac{\partial a_{lj}}{\partial w_{(l-1)ij}} \\ &\doteq \delta_{lj} \cdot z_{(l-1)i}\end{aligned}$$

反向传播得到损失函数对每个参数的梯度值后就可以用梯度下降法更新参数，直到训练完毕

学习步骤

- Define the neural network that has some learnable parameters (or weights)
- Iterate over a dataset of inputs
- Process input through the network
- Compute the loss (how far is the output from being correct)
- **Propagate gradients back into the network's parameters**
- Update the weights of the network, typically using a simple update rule: $\text{weight} = \text{weight} - \text{learning_rate} * \text{gradient}$

三、隐藏层

(一)、全连接

又称为Linear或Affine层，比如在分类问题中， $\mathbf{a} = \mathbf{w}^T \mathbf{z} + \mathbf{b}$ 。

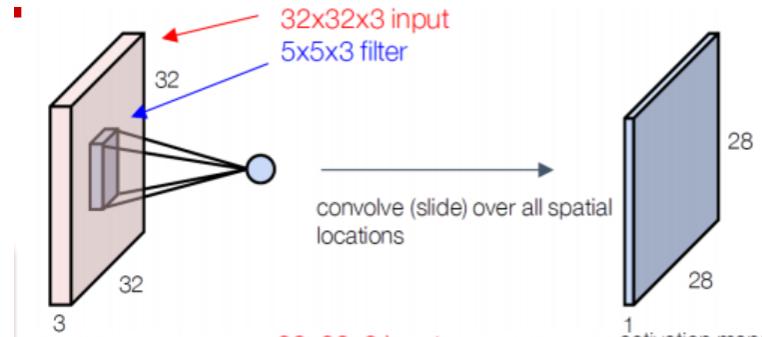
一般用在网络倒数几层，准备分类之前。只接受向量形式的数据，输入到全链接层之前要把高维度的数据“拍扁”到一维。

(二)、CNN

特征提取，用滑动窗口kernel在图像上“卷积”，实际只是计算cross relation

$$(x * k)(i, j) \approx \hat{x}(i, j) = \sum_l \sum_m x(i+l)y(j+m) \cdot k(l, m)$$

- 若输入图像有多个通道，与之对应的kernel也得有这么多通道，kernel是三维的（如下图中 $5 \times 5 \times 3$ ）。
- 一个kernel同时卷积输入图像的C个通道 (in channel)，才能得到1个输出通道。
- 输出通道数 (out channel) = kernel个数



因此，一个卷积层基础参数有3个：

kernel size 3×3

in channel 3

out channel 6

(三)、CNN: 补偿padding

卷积后图像尺寸会缩小，是否要补偿？

valid 不补偿，缩小就缩小 $D' = (D - K) + 1$

same 补到和输入尺寸一致 两端各加 $P = \lfloor \frac{K}{2} \rfloor$

full 比原来还大了 两边加 $P = K - 1$

D' ——新尺寸， D ——旧尺寸， K ——kernel大小

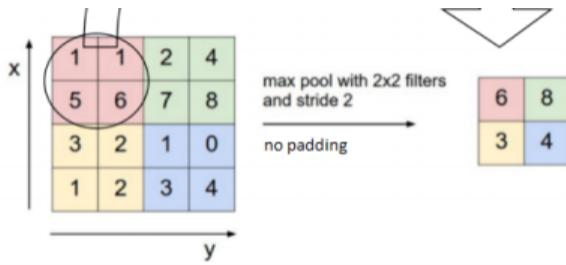
(四)、CNN: 步长stride

每次卷积相邻几个像素？当步长 $S > 1$ 时，会缩小图像尺寸 (Down-sampling)

$$D' = \left\lceil \frac{D + P - K}{S} \right\rceil + 1$$

P ——padding大小， S ——步长

(五)、CNN: 池pooling



kernel只能对窗口中的像素线性操作，pooling可以引入非线性操作比如max

四、其他技巧

(一)、Dropout

在每一层输出到下一层前，以概率 p 随机将某些结点置零，可以有效防止过拟合

(二)、minibatch normalization

给到激活函数之前，把这一层的结果规范到正态分布，避开激活函数两端梯度变化非常弱的区间

$$y = \frac{x - \bar{x}}{\sigma_x + \epsilon} \cdot \gamma + \beta$$

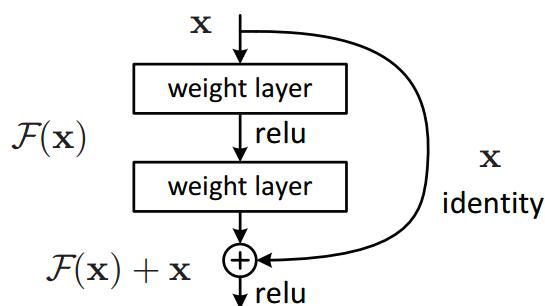
ϵ 是一个极小量，防止算式除零错误

输入N, L, C的数据时，对每一个C做一次。

- N - batch size / sample size
- L - sequence length
- C - the number of features / channels / filters

(三)、Residual Net

In traditional neural networks, each layer feeds into the next layer. In a network with residual blocks, each layer feeds into the next layer and directly into the layers about 2–3 hops away.



原理：层数多的网络反而不能很好处理简单的函数关系→需要层数少的网络→在深层网络中跳过一些层，得到浅层网络

如何跳过？skip the training of few layers using **skip-connections** (residual connections).

五、Xavier Glorot 初始化

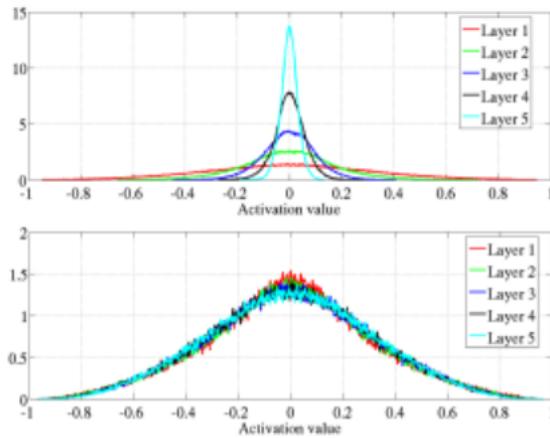
初始权重是优化步骤的起点，非常关键。然而有以下两个难题：

1. Weight Symmetric: 若两个结点权重完全相同，提取出的特征也必定相同，无法得到不同的学习结果→人为加上一些噪音
2. Weight Scale: fan-in过大时，输出与太多输入量有关，输入中稍有变动，输出就迥然不同，使得更新后的权重W剧烈变化，容易overshoot，错过最优的解

∴ 初始化权重矩阵时，令 $Mean(W) = 0, Var(W) = \frac{2}{fan-in+fan-out}$ ，这样可以保留输入信号特征（方差和均值）

$fan-in = kernel size * in channel$

$fan-out = ~ * out channel$



Activation histogram. Top:
Unnormalized. Bottom: Glorot init.

六、防溢出

softmax常包含 e^{x_i} 项，在计算机中处理指数上升数据常要考虑数据溢出。为防止溢出可以计算

$$\log \sum_i^N e^{x_i} = a + \log \sum_i^N e^{x_i-a}$$

即将数据均值平移a，通常取 $a = \max x_i$

$$\log \sum_i^N e^{x_i} = a + (-a) + \log \sum_i^N e^{x_i} = a + \log e^{-a} + \log \sum_i^N e^{x_i} = a + \log \sum_i^N e^{x_i-a}$$

同理对softmax函数，可以计算

$$\frac{e^{x_i}}{\sum_i^N e^{x_i}} = \frac{e^{x_i-a}}{\sum_i^N e^{x_i-a}}$$

PyTorch

```
1 | import torch
```

Tensor 类型和numpy的ndarray一样，但是可以交给GPU运行

package: autograd

核心函数autograd: 计算 $(v^T J)^T$ ，此处的 v 为全1向量时就是在计算Jacobian 矩阵

如果输出out是一个标量， $out.backward()$ 相当于 $out.backward(torch.tensor(1.))$ ，

e.g:

$$x \rightarrow \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \quad y = x + 2 \rightarrow \begin{pmatrix} 3 & 3 \\ 3 & 3 \end{pmatrix} \quad z = 3y^2 \rightarrow \begin{pmatrix} 27 & 27 \\ 27 & 27 \end{pmatrix} \quad out = z. mean \quad 27$$

```
1 | out.backward() #这条命令一给，就会计算链条上所有变量的梯度
2 | x.grad#直接给出d(out)/dx的梯度值
```

在上例中

$$\begin{aligned} out &= \frac{1}{4} \sum_i 3(x_i + 2)^2 \\ \frac{\partial out}{\partial x} &= \frac{\partial out}{\partial z} \frac{\partial z}{\partial y} \frac{\partial y}{\partial x} = \frac{1}{4} \cdot 3 \times 2(x_i + 2) = \frac{3}{2}(x_i + 2) \\ \therefore x.grad &\text{为} \begin{pmatrix} \frac{9}{2} & \frac{9}{2} \\ \frac{9}{2} & \frac{9}{2} \end{pmatrix} \end{aligned}$$

package: nn

```
1 | import torch.nn as nn
2 | import torch.nn.functional as F
```

Dimension reduction 降维

非监督学习——聚类：找到隐藏结构 $p(z)$ 和 generative transformation $p(x|z)$ 来拟合 $p(x)$

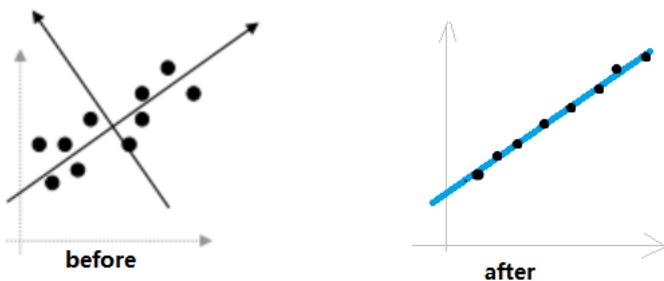
降维：指把多维的数据降到聚类中心，且避免信息损失

一、PCA (Principal Component Analyse)

原理：

1. 数据的多个特征往往互相关联 → 找到变换 F ，使变换后数据的几个特征间协方差为0（没有关联）
2. 方差低的维度包含的信息少 → 直接删除方差低的维度

若 $X' = X \cdot F$, 变换后协方差矩阵为 $F^T \Sigma_x F$



步骤：

1. 数据中心化：每一个维度减去平均值 $\tilde{x}_j = x_j - \bar{x}_j$
2. 计算协方差矩阵 $\Sigma_{\tilde{\mathbf{X}}} = \frac{1}{N} \cdot \mathbf{X}^T \mathbf{X} - \bar{\mathbf{x}} \bar{\mathbf{x}}^T$
3. 用特征向量分解 decomposition

$$\Sigma_{\tilde{x}} = \Gamma \Lambda \Gamma^T$$

$$\Lambda = \begin{pmatrix} \lambda_1 & & O \\ & \lambda_2 & \\ O & & \ddots \end{pmatrix}$$

Γ是正交阵

∴ 变化后矩阵 $\tilde{Y} = \tilde{X} \cdot \Gamma$, \tilde{Y} 的协方差矩阵 $\Sigma_{\gamma} = \Gamma^T \Sigma_{\tilde{x}} \Gamma \triangleq \Lambda$ 。在 \tilde{Y} 中, 每个维度均线性无关, 各维度方差为 λ_i

4. 删去最末几个 λ_i , Γ 中也相应删去对应的列向量

$$\Gamma' = \begin{bmatrix} | & | & | & | \\ \gamma_1 & \gamma_2 & \cdots & \cancel{\gamma_n} \\ | & | & | & | \end{bmatrix}$$

求解:

具体实现中, 不需要求对应 Λ 的整个 Γ , 只需要找到最大的 k 个 λ_i

1. Power Iteration: 找到最大 λ

ν 任意一个归一化向量

$\nu \leftarrow \frac{X \cdot \nu}{\|X \cdot \nu\|}$, 直到收敛

ν 最后会收敛到最大特征值 λ_1 对应的特征向量。可以通过 $X \cdot \nu = \lambda \cdot \nu$ 求出 λ

2. 找到第二大 λ

$$X = \Gamma \Lambda \Gamma^T = \sum_i \lambda_i \cdot \gamma_i \cdot \gamma_i^T$$

$$X' = X - \lambda_i \gamma_i \gamma_i^T$$

继续对 X' 使用 Power Iteration 即可

3. 何时停止? 如何知道到第几个 λ_i 结束?

$$\therefore Tr(\Sigma_x) = \sum_i^d (\Sigma_{xii})$$

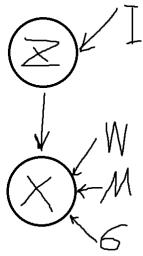
∴ 所有 λ_i 的和是已知的。求出前 i 个 λ , 比较 $\sum_i^k \lambda_i \geq 0.9 Tr(\Sigma_x)$ 即可

PCA 默认原始数据已经非常贴近一条直线, 对离群数据敏感

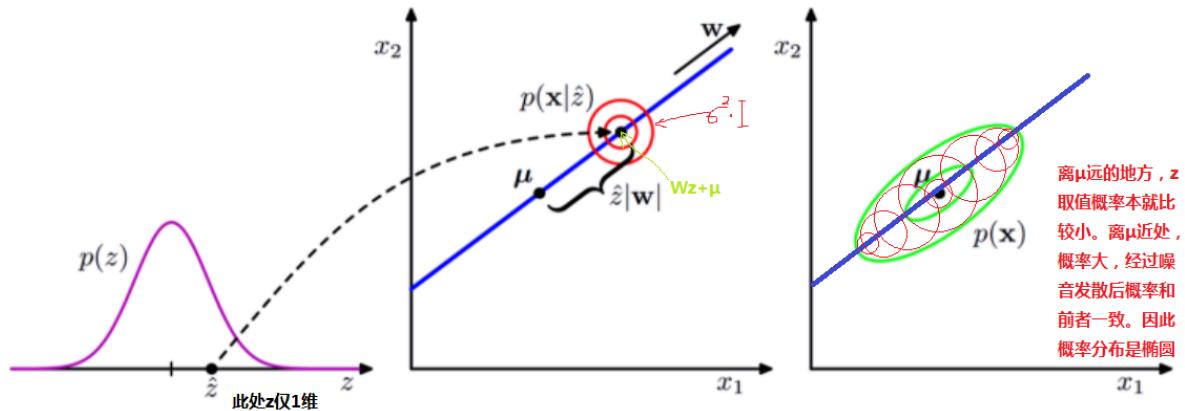
二、Probabilistic PCA

1. 建模: 线性回归模型:

$$\left| \begin{array}{l} X \sim W z_i + \mu + \epsilon_i \\ z_i \sim \mathcal{N}(0, \mathbf{I}) \\ x_i | z_i \sim \mathcal{N}(\mathbf{W} z_i + \vec{\mu}, \sigma^2 \mathbf{I}), \mathbf{W} \in \mathbb{R}^{D \times K} \end{array} \right| \text{默认噪音为 } diag(\sigma^2, \sigma^2, \dots, \sigma^2)$$



当 Z 一维正态分布， X 只有两个分量时，可以图形化表示：



2. 学习：积分除去 z , $p(x) = \int p(x|z) \cdot p(z) dz$

$$p(x) = \int \mathcal{N}(0, \mathbf{I}) \cdot \mathcal{N}(\mathbf{Wz}_i + \vec{\mu}, \sigma^2 \mathbf{I}) dz = \mathcal{N}(\mu, \mathbf{WW}^T + \sigma^2 \cdot \mathbf{I})$$

$$\begin{aligned} \mathbb{E}[\mathbf{x}] &= \mathbb{E}[\mathbf{Wz} + \mu + \epsilon] = \mathbf{W}\mathbb{E}[\mathbf{z}] + \mathbb{E}[\mu] + \mathbb{E}[\epsilon] = \mu \\ Cov[\mathbf{x}] &= \mathbb{E}[(\mathbf{x} - \mathbb{E}[\mathbf{x}])(\mathbf{x} - \mathbb{E}[\mathbf{x}])^T] = \mathbb{E}[(\mathbf{Wz} + \epsilon)(\mathbf{Wz} + \epsilon)^T] \\ &= \mathbb{E}[\mathbf{Wzz}^T \mathbf{W}^T + \epsilon \epsilon^T] = \mathbf{W}\mathbb{E}[\mathbf{zz}^T]\mathbf{W}^T + \mathbb{E}[\epsilon \epsilon^T] = \mathbf{WIW}^T + \sigma^2 \mathbf{I} = \mathbf{WW}^T + \sigma^2 \mathbf{I} \end{aligned}$$

3. 推理

观察到序列 \mathbf{x} , 训练出 $\mu, \mathbf{W}, \sigma^2$, 找到对应的 \mathbf{z} : $p(z|x, \mathbf{W}, \mu, \sigma^2)$

由于分布 \mathbf{z} 的维度比观察样本 \mathbf{x} 少, 因此实现了降维

pPCA无需计算 Σ_x 矩阵, 不需要全部数据 (只需要likelihood function), 因此可以处理缺失值

4. pPCA与PCA的关联

$$\begin{aligned} \mathbf{W}_{ML} &= \mathbf{U}_k (\Lambda_k - \sigma^2 \mathbf{I})^{\frac{1}{2}} \mathbf{V} \\ \text{当} \sigma^2 = 0 \text{ 时, } \mathbf{W}_{ML} &= \mathbf{U}_k \Lambda_k^{\frac{1}{2}} \\ \mathbf{WW}^T &= \mathbf{U}_k \Lambda_k^{\frac{1}{2}} \Lambda_k^{\frac{1}{2}} \mathbf{U}_k^T = \mathbf{U}_k \Lambda_k \mathbf{U}_k^T \end{aligned}$$

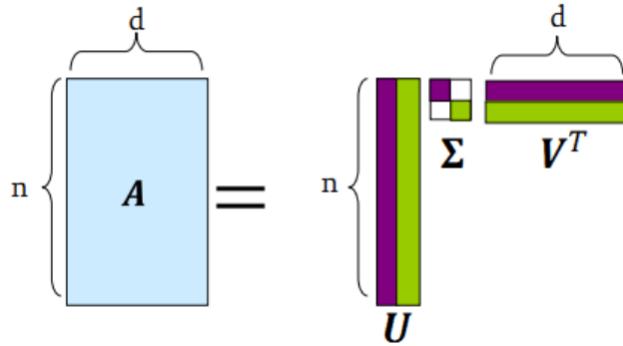
三、SVD——矩阵因式分解

矩阵的秩可以理解为数据的维度

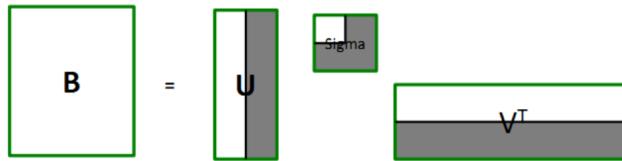
任何实数矩阵 $\mathbf{A} \in \mathbb{R}^{n \times d}$ 都可以分解为 $\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^T = \sum_i^r \sigma_i \cdot \mathbf{u}_i \cdot \mathbf{v}_i^T$, 其中 \mathbf{U}, \mathbf{V} 均为正交阵 ($\mathbf{U}\mathbf{U}^T = \mathbf{I}$)

- \mathbf{U} 称为left singular vectors, \mathbf{V} 为right ~

- $\Sigma \in \mathbb{R}^{r \times r}$ 对角阵，阶数 r 为矩阵 A 的秩， $\therefore r \leq d$ ，其元素全正且从小到大排列(
 $\sigma_1 \geq \sigma_2 \geq \dots \geq 0$)



在 Σ 的 r 个元素中，再砍去末尾几个数值小的量，就减少了 Σ 的秩， $A \approx B \in \mathbb{R}^{n \times d}$ ，根据 Frebenius Norm， B 是 A 的最佳近似



Projection: 获得裁剪并投影后的数据 $P = U' \cdot \Sigma' = A \cdot V'$ ，矩阵 $B \approx A$ 但仍留在 A 的坐标系下，仍需要用 A 的坐标轴来表示（维度没有显式地减少），经过旋转矩阵 V 后， B 转到 P 矩阵，多余的坐标轴即可删去。

计算 U, V, Σ

U 是 XX^T 的特征向量 $\in \mathbb{R}^{n \times n}$ ， V 是 $X^T X$ 的特征向量 $\in \mathbb{R}^{d \times d}$ ， $A = U^{(n \times n)} \Sigma^{(n \times d)} V^{T(d \times d)}$

$$X \cdot X^T = U \Sigma V (U \Sigma V)^T = U \Sigma \Sigma^T U^T, \text{ 同理 } X^T \cdot X = V \Sigma^2 V$$

Σ 为 U 或 V 对应的特征值开根号，并从大到小排列

四、矩阵因式分解 Factorization

$$\text{SVD: } R = \underbrace{Q}_{U \cdot \Sigma} \cdot \underbrace{P^T}_{V^T}$$

但不适用于稀疏矩阵，因为不能处理缺失数据，表现为：

1. 若在数据缺失位置置零，则 SVD 会尝试还原这些零，然而在 e.g. 评分系统中缺失的数据 $\neq 0$ (极低评分)
2. U 和 V 矩阵在数据非常稀疏时仍非常稠密，就不太对劲

\Rightarrow 实际上不是非要让 V ， U 为正交阵

$$\begin{aligned} SSD &= \min \sum_{i,j} (A_{ij} - [U \Sigma V^T])^2 \\ &= \min_{P,Q} \sum_{\substack{x \in R \\ \text{只计算} \\ \text{数据中} \\ \text{出现的}}} (r_{xi} - \underbrace{q_x p_i^T}_{\substack{Q,P \text{ 不必} \\ \text{为正交}}})^2 \\ &\triangleq \min_{P,Q} f(P, Q) \end{aligned}$$

(一)、Alternating Optimierung

求解上式可用Alternating Optimierung: 分别依次求P,Q梯度下降

$$P^{(t+1)} = \underset{p}{\operatorname{argmin}} f(P, Q^{(t)})$$

$$Q^{(t+1)} = \underset{q}{\operatorname{argmin}} f(P^{(t+1)}, Q)$$

$t+1 = 1$, 直到收敛

直到收敛, 即no improvement on the validation loss for a certain amount of training steps

由于计算时Q已经固定, 求P最优解可以分割为对每一个 p_i 求最优解再求和

$$\min_P f(P, Q^{(t)}) = \sum_i^d \min_{p_i} \sum_{x \in R} (r_{xi} - q_x p_i^T)^2$$

如此就将问题简化为一系列常规最小平方和 (Ordinary Least Square) 优化,

1. 最优值解析解已知 \Rightarrow 算得快
2. R中缺失值不用算 \Rightarrow 算得更快
3. 不保证算出最优解P,Q

(二)、Regularization

变化后的维度为k, 然而k太大会过拟合。 (n个样本, k维就有 $n \times k$ 个参数)

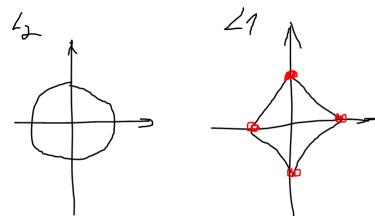
$$\min_{P,Q} \underbrace{\sum_{x \in R} (r_{xi} - q_x p_i^T)^2}_{\text{数据误差项}} + \underbrace{\left[\lambda_1 \sum_x \|p_x\|^2 + \lambda_2 \sum_i \|q_i\|^2 \right]}_{\text{惩罚项}}$$

数据为累加项, 当数据充足时, 数据误差项占比大, 惩罚项可忽略; 数据稀疏时, 惩罚项起作用, 减小权重防止过拟合

比较L1与L2-Regularization

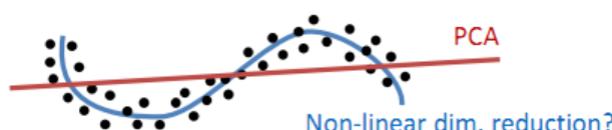
L2平均缩小每个维度上的权重; L1会将一些权重置零 \Rightarrow 获得稀疏矩阵

原理: 最优解永远在凸集的顶点上, L1凸集形状为菱形, 因此最优解在菱形的顶点上, 意味着某些维度权重为0

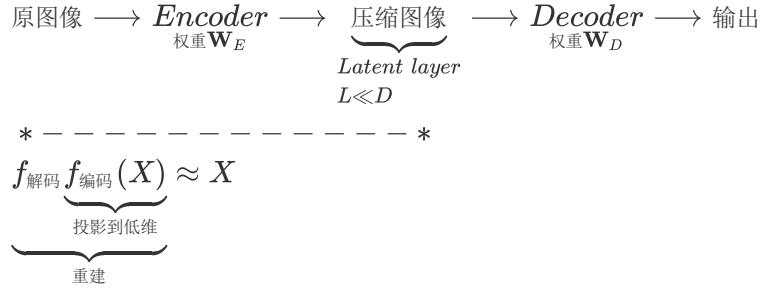


五、Autoencoder

PCA / SVD不能拟合沿曲线的一维数据



路径图:



编码器维度 $L \ll$ 输入图像维度 D : 迫使模型学习最主要的特征, 实现降维; 若 $L > D$ 则会直接学习一个identity function, 降维失败

- 当Autoencoder为线性, 即 $f_{enc}/f_{dec} = \mathbf{x}\mathbf{W}$ 时, 输出 $\hat{\mathbf{y}} = \mathbf{x}\mathbf{W}_1\mathbf{W}_2 \triangleq \mathbf{x} \cdot \mathbf{W}$ (令 \mathbf{W} 秩为R), 则就相当于PCA或SVD

聚类

有 $z_i \in K$ 类, 使得组内差异小, 组间差异大。 $\min_{\mathbf{Z}, \mu} J(\mathbf{X}, \mathbf{Z}, \mu) = \min_{\mathbf{Z}, \mu} \sum_i \sum_k z_{ik} \|x_i - \mu_k\|^2$, 另外必须保证每个 x_i 均分到类 $\rightarrow \forall i \sum_k z_{ik} = 1$

一、K-means

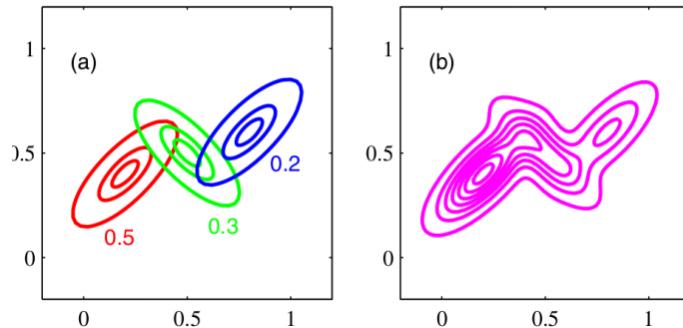
- 初始化聚类中心centroids $\mu_1, \mu_2, \dots, \mu_k$
- 更新分类结果 $z_{ik} = \begin{cases} 1 & \text{如果 } k = \arg \min_j \|x_i - \mu_j\|^2 \\ 0 & \text{否则} \end{cases}$
- 更新聚类中心 $\mu_k = \frac{1}{N_k} \sum_i z_{ik} \mathbf{x}_i$
- 重复步骤2、3直至收敛

二、混合高斯

K-means每次将一个数据点归到最近的类, 混合高斯可以给出一个数据到多个类的概率。

$$p(\mathbf{x}|\pi, \mu, \Sigma) = \sum_k \underbrace{p(z_k|\pi)}_{\text{分类先验概率}} \cdot \underbrace{p(\mathbf{x}|z_k, \mu_k, \Sigma_k)}_{\text{以聚类中心建模的高斯}} = \sum_k \pi_k \cdot \mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k)$$

即几个高斯模型的线性组合, 几个高斯模型完全无联系, Σ_k 任意取



当数据分布不呈椭圆形, 用单一高斯模型来建模就不精准。混合高斯模型即多个椭圆叠加更好。

- 推理: 数据点 x_i 聚类到第k类的概率即为

$$p(z_{ik} | \mathbf{x}_i, \pi, \mu, \Sigma) = \frac{p(x|z) \cdot p(z)}{p(x)} = \frac{\mathcal{N}(x_i | \mu_k, \Sigma_k) \cdot \pi_k}{\sum_j \pi_j \cdot \mathcal{N}(\mathbf{x}_i | \mu_j, \Sigma_j)} \triangleq \gamma(z_{ik})$$

2. 学习：如何获得参数 μ_k, Σ_k, π_k ？

答：最大化 x 出现的概率，即最大化 $\log p(x|\pi, \mu, \Sigma) = \max \log \left[\sum_k \pi_k \cdot \mathcal{N}(x_i | \mu_k, \Sigma_k) \right]$

如果是分类问题，有标签 z_i 可以设损失函数 $Loss = \sum \mu_i - z_i = \hat{z}_i - z_i$ 。然而在聚类问题中，根本没有 z 。没有依据来更新参数。

回顾监督学习 LDA，标签 $z \sim CAT(\vec{\pi})$, $\pi_k = \frac{N_k}{N}$; 数据 $x \sim \mathcal{N}(\mu_k, \Sigma_{\text{共享}})$

3. EM 算法：Expectation+Maximization

$$\boxed{\begin{array}{l} E \rightarrow \text{用初始先验 (initial guess) 先做一次聚类, 生成标签 } \vec{z}_i \\ M \rightarrow \text{更新参数 } \pi, \mu, \Sigma \\ \left. \begin{array}{l} \gamma_t(z_{ik}) = p(z_{ik} | \mathbf{x}_i, \pi, \mu, \Sigma) \\ \arg \max_{\pi, \mu, \Sigma} \mathbb{E}_{\mathbf{Z} \sim \gamma_t} \log p(\mathbf{X}, \mathbf{Z} | \pi, \mu, \Sigma) \\ \mu_k^{(t+1)} = \frac{1}{N_k} \sum_i \gamma_t(z_{ik}) \cdot x_i: \text{加权平均} \\ \pi_k^{(t+1)} = \frac{N_k}{N}, N_k \sum_i \gamma_t(z_{ik}) \\ \Sigma_k^{(t+1)} = \frac{1}{N_k} \sum_i \gamma_t(z_{ik})(x_i - \mu_k^{(t+1)})(x_i - \mu_k^{(t+1)})^T \end{array} \right. \end{array}}$$

E 减少 KL 散度，M 把下限往上推