

Machine Learning Cookbook

EscapeTHU: GITHUB SITE

June 2022

数学从来不是一种工具，而是一种思想。

算者无用，道其中也。

Math is never a mean, but thought.

Mathematica non est instrumentum sed ideum.

——EscapeTHU

目录

1 前言与总述	6
1.1 前言	6
1.2 机器学习方法图谱	7
1.2.1 有监督学习	7
1.2.2 无监督学习	8
1.2.3 深度学习	8
1.3 特征工程方法图谱	9
1.4 其他方法	10
1.5 整体图谱	11
2 贝叶斯决策理论	12
2.1 贝叶斯决策面对的问题假设	12
2.2 贝叶斯决策准则	12
2.2.1 最小错误率准则	12
2.2.2 最小风险准则	13
2.2.3 给定一类错误率优化另一类错误率	14

3 概率密度函数估计	15
3.1 概率密度函数估计问题的建立	15
3.2 概率密度函数估计的衡量标准	16
3.2.1 按照均值衡量	16
3.2.2 按照方差衡量	16
3.3 参数估计	16
3.3.1 贝叶斯最优参数估计	16
3.3.2 最大似然估计	17
3.3.3 不能求导的均匀分布例子	17
3.4 非参数估计	18
3.4.1 Parzen 窗估计	18
3.4.2 最近邻估计	19
3.5 概率密度函数估计中的关键问题与改进	19
3.5.1 概率密度函数估计的准确性与分类器性能的关系	19
3.5.2 维数问题	19
3.5.3 过拟合与模型选择	19
3.6 概率密度函数估计应用实例：错误率估计	20
4 混合高斯模型与 EM 算法	20
4.1 混合高斯模型	20
4.2 EM 算法	21
4.3 EM 算法在混合高斯模型上的应用	23
4.4 扩展的 EM 算法 (Generalized EM)	24
5 支持向量机	24
5.1 生成式模型与判别式模型	24
5.2 严格线性可分情形的支持向量机	25
5.3 线性不可分情形的支持向量机	26
5.4 非线性情形的支持向量机	27
5.5 VC 维、经验风险最小化准则与 SVM 的可解释性	31
5.6 支持向量机的正则化	32
6 近邻法	32
6.1 K 近邻法	32

6.1.1 算法	32
6.1.2 以最近邻为例的错误率推导	33
6.1.3 K 近邻的错误率	36
6.2 压缩近邻法	36
6.3 距离度量	37
7 有监督数据降维 (特征选择)	37
7.1 特征选择问题描述	37
7.2 二分类的 Fisher 准则	38
7.3 多分类的 Fisher 准则	40
7.4 基于 Fisher 准则的分类器设计	41
7.5 RELIEF 算法	41
7.6 其他算法	42
8 神经网络	43
8.1 反向传播 BP 算法	43
8.1.1 神经网络的常见函数和操作	43
8.1.2 向量求导链式法则	45
8.1.3 实际计算 BP 算法	46
8.2 多层感知机算法 MLP	48
8.3 优化器	48
8.4 批标准化 Batch Normalization	49
8.5 卷积神经网络 CNN	49
8.6 池化 Pooling	50
8.7 残差神经网络 Residual	50
8.8 循环神经网络 RNN	51
8.9 长短期记忆 LSTM	53
8.10 自动编解码器 En/De-coder	54
8.11 注意力机制 Attention	54
8.11.1 ATTENTION IS ALL YOU NEED?	54
8.11.2 软注意力机制 Soft-Attention	55
8.11.3 硬注意力机制 Hard-Attention	56
8.11.4 软注意力机制在 Encoder-Decoder 中的运用	57
8.12 图卷积神经网络 GCN	58

8.12.1 图卷积神经网络问题提出	58
8.12.2 图卷积神经网络原理	58
9 非监督数据降维	60
9.1 主成分分析 PCA	60
9.1.1 PCA 问题引入与最优子空间	60
9.1.2 PCA 算法推导	60
9.1.3 PCA 算法流程	62
9.2 多维尺度变换 MDS	62
9.2.1 多维尺度变换的问题提出	62
9.2.2 多维尺度变换的数学推导	63
9.3 非线性降维方法 ISOMAP	64
9.3.1 ISOMAP 问题引入	64
9.3.2 ISOMAP 的算法流程	65
9.3.3 关于 ISOMAP 算法的一些讨论	65
9.4 非线性降维方法 LLE	66
9.4.1 LLE 问题引入	66
9.4.2 LLE 算法推导	66
10 非度量方法与决策树	69
10.1 非度量方法引言	69
10.2 不纯度度量方法	70
10.2.1 不纯度度量方法在决策树中的作用	70
10.2.2 符号统一	70
10.2.3 信息增益 (ID3 决策树)	70
10.2.4 增益率 (C4.5 决策树)	71
10.2.5 基尼系数准则 (CART 决策树)	71
10.3 决策树模型基本算法	72
10.3.1 递归建树算法	72
10.3.2 分支停止条件	72
10.3.3 剪枝算法	72
10.3.4 缺失数据的处理	74
10.4 CART 决策树算法	75
10.5 ID3 决策树算法	75

10.6 C4.5 决策树算法	75
11 集成学习 Ensemble	76
11.1 集成学习问题引入	76
11.2 基于训练样本的分类器构造	76
11.2.1 Bagging 方法	76
11.2.2 Boosting 方法的代表 Adaboost	76
11.2.3 二分类问题的 Adaboost 推导	77
11.3 基于样本特征的分类器构造	79
11.4 分类器输出的融合	79
11.4.1 决策层输出	79
11.4.2 排序层输出	79
11.4.3 度量层输出	80
11.5 集成学习可解释性	80
12 聚类分析与无监督学习	81
12.1 聚类分析与无监督学习引言	81
12.2 K-Means 聚类方法	81
12.3 分级聚类方法	82
12.3.1 两类之间的相似性度量方法	82
12.3.2 分级聚类算法	83
12.4 谱聚类	83
13 计算学习理论	84
14 概率图模型	84
15 强化学习理论	84
16 后记	84

1 前言与总述

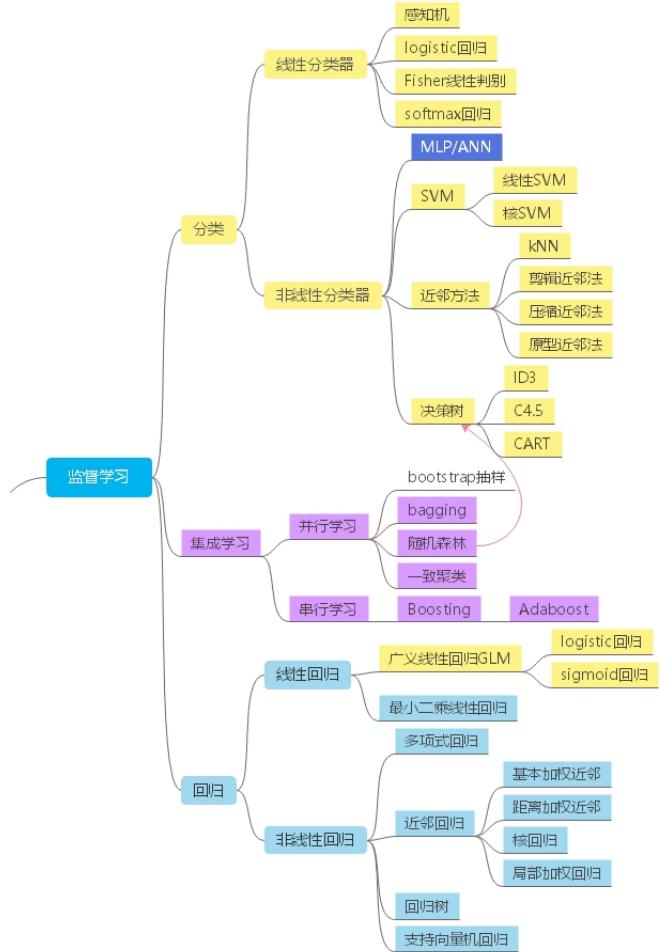
1.1 前言

琢磨多遍，本文的题目还是定为 Machine Learning Cookbook 较好。和著名的 Matrix Cookbook 类似，我编写这个文档的关键目的在于：一方面自己把所有已经掌握的机器学习方法手推一遍；另一方面则是关注每一种算法面对的基本问题、以及提出的基本假设、基本思想，而不在于把所有的算法都枚举出来、逐一对比。

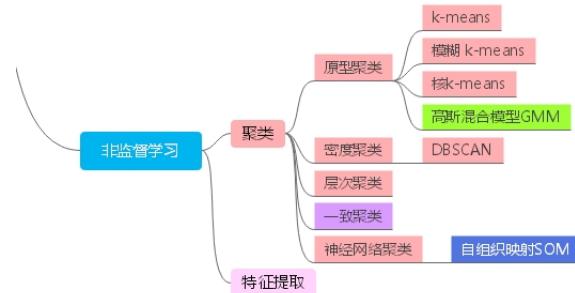
下面给出一份我的朋友整理的机器学习知识架构作为开始。图中的架构我基本都是认同的，但是在特征工程的部分，我认为他图中的“特征提取”应当是指“无监督数据降维”、他图中的“特征筛选”应当是指“有监督数据降维”。

1.2 机器学习方法图谱

1.2.1 有监督学习



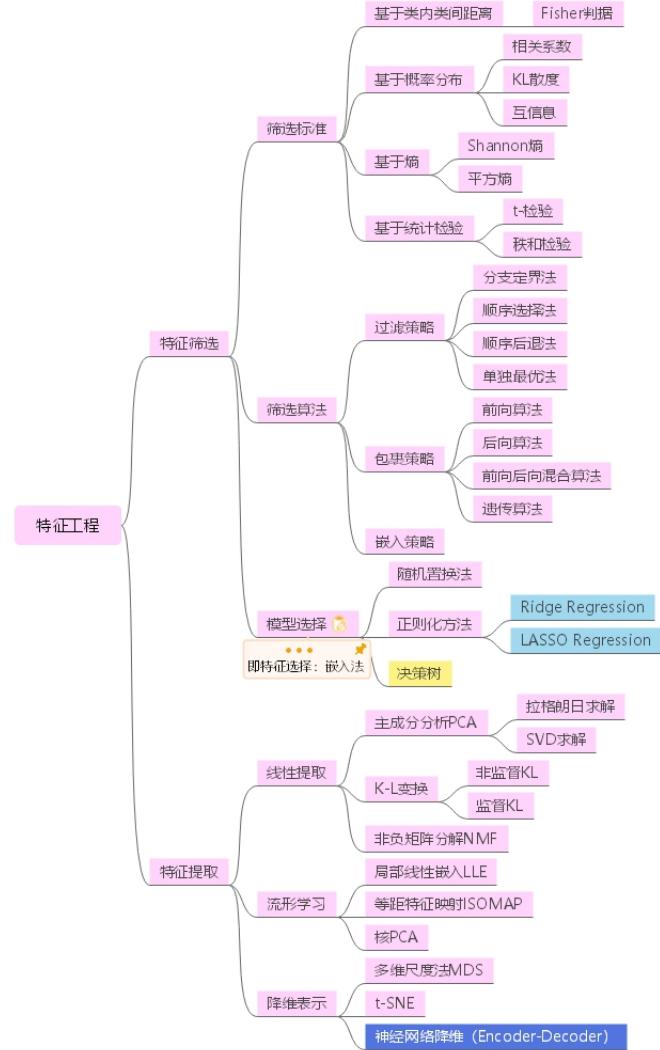
1.2.2 无监督学习



1.2.3 深度学习



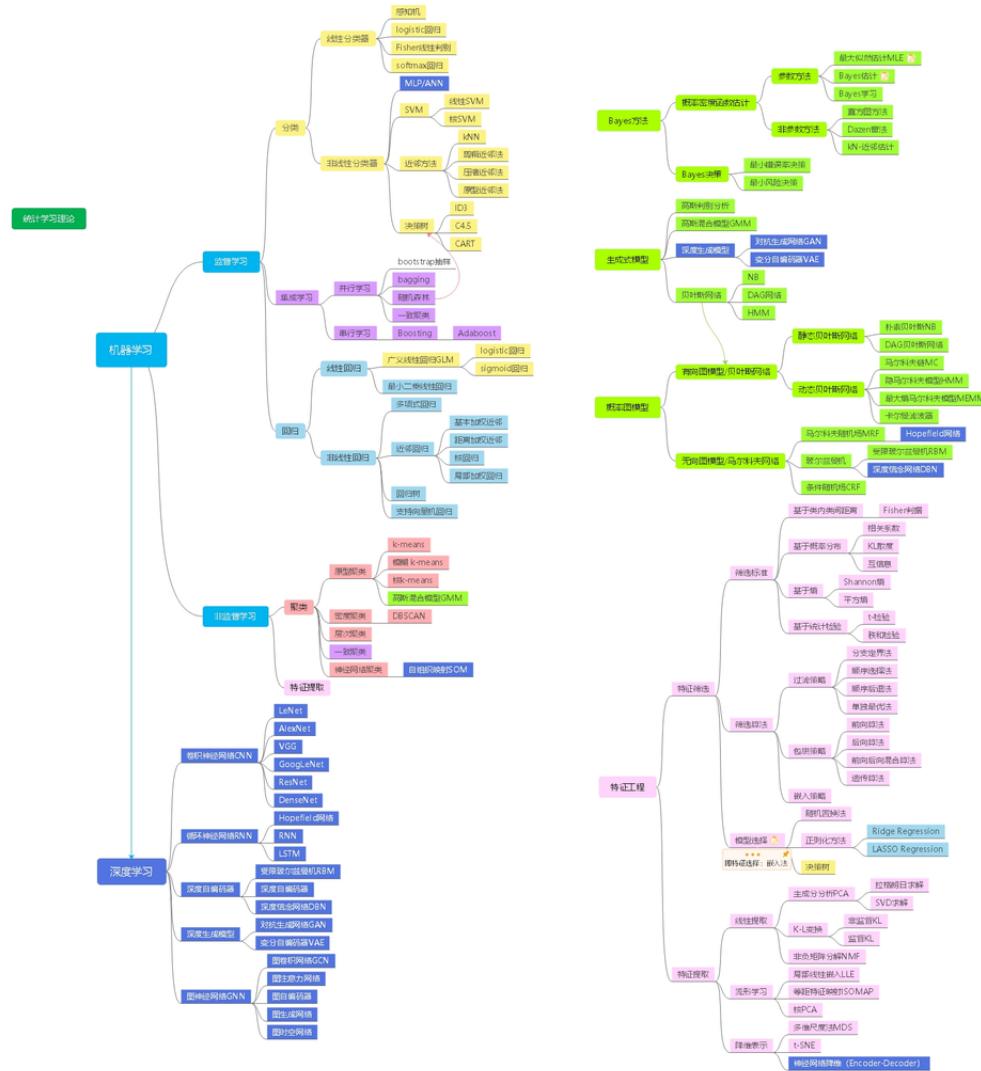
1.3 特征工程方法图谱



1.4 其他方法



1.5 整体图谱



2 贝叶斯决策理论

2.1 贝叶斯决策面对的问题假设

贝叶斯决策算法面对的问题主要有两点假设：第一，认为各个类别的总体概率分布已知；第二，认为分类数已知。

2.2 贝叶斯决策准则

2.2.1 最小错误率准则

在判决规则上，规则是：在某一段区间内，总是选择概率密度函数最大的类别作为预期类别。

在错误率方面，首先针对两类的情形进行推导。我们假设第一类的分布函数为 $p(x|w_1)$ ，第二类分布函数为 $p(x|w_2)$ ，那么错误率为：

$$P(e) = \int_{x \in \mathcal{R}_1} P(w_2|x)p(x)dx + \int_{x \in \mathcal{R}_2} P(w_1|x)p(x)dx$$

带入贝叶斯准则 $P(w_i|x)p(x) = p(x|w_i)P(w_i)$ 得到：

$$P(e) = \int_{x \in \mathcal{R}_1} p(x|w_2)P(w_2)dx + \int_{x \in \mathcal{R}_2} p(x|w_1)P(w_1)dx$$

$$P(e) = P(w_2)P_2(e) + P(w_1)P_1(e)$$

上式之中 $P_i(e)$ 为第 i 类错误，含义是本身应该是第 i 类但是被误判为其他类别的概率。一般认为第一类错误是指本身是阴性但是被判定为阳性（例如课本上的男人怀孕）；第二类错误是指本身是阳性，但是被判定为阴性（怀孕的女士被告知没有怀孕）。

那么在更多类别的情形下，第 i 类错误为：

$$P_i(e) = \sum_{j=1, j \neq i}^c P(x \in \mathcal{R}_j | w_i)$$

进而整体错误率为：

$$P(e) = \sum_{i=1}^c P_i(e)P(w_i)$$

我们可以发现此时整个式子之中的唯一变量就是积分边界，给定特殊的分布函数形式就可以通过对 $P(e)$ 求导的方式得到最优边界。我们不妨以两类

高斯分布为例。

$$p(x|w_1) = \frac{1}{\sqrt{2\pi}\sigma_1} e^{-\frac{(x-\mu_1)^2}{2\sigma_1^2}}$$

$$p(x|w_2) = \frac{1}{\sqrt{2\pi}\sigma_2} e^{-\frac{(x-\mu_2)^2}{2\sigma_2^2}}$$

我们考虑两种高斯分布的概率密度取值相同的位置：

$$p(x|w_1) = p(x|w_2)$$

$$(\sigma_2^2 - \sigma_1^2)x^2 - 2(\mu_1\sigma_2^2 - \mu_2\sigma_1^2)x + \sigma_2^2\mu_1^2 - \sigma_1^2\mu_2^2 - 2\sigma_1^2\sigma_2^2 \ln\left(\frac{\sigma_2}{\sigma_1}\right) = 0$$

如果 $\sigma_1 = \sigma_2$, 分界线为 $x = \frac{\mu_1 + \mu_2}{2}$ 。如果标准差不同，则会产生两个交点。在每一段之中，为了保证错误率最小化，应当选择分类为概率密度更大的一类。进一步进行求解即可（第一次作业）。

2.2.2 最小风险准则

首先构建损失函数，给定 \hat{w} 是预期的类别， w 是真实的类别， $\lambda(\hat{w}, w)$ 是在真实类别为 w 是预测为 \hat{w} 的损失函数，那么可以求出在 x 处的损失期望函数 $R(\hat{w}|x)$ ，也被称为条件平均风险：

$$R(\hat{w}|x) = \int \lambda(\hat{w}, w) P(w|x) dw$$

如果是离散决策的情形，那么：

$$R(\hat{w}_j|x) = \sum_{i=1}^c \lambda(\hat{w}_j, w_i) P(w_i|x)$$

于是针对每一个 x 都可以求出对应的条件平均风险。不论是离散决策还是连续决策，最终选择的分类是使得 $R(\hat{w}|x)$ 最小的类别。

我们还是以两分类的高斯问题为例，为了简化处理，我们用两个均值不同但是标准差相同的高斯分布作为研究对象。给定的风险函数如下所示：

$$\Lambda = \begin{bmatrix} & \hat{w}_1 & \hat{w}_2 \\ w_1 & 0 & \lambda_1 \\ w_2 & \lambda_2 & 0 \end{bmatrix}$$

也就是针对第一类错误的惩罚为 λ_1 ，针对第二类错误的惩罚为 λ_2 。那么计算过程如下：

$$R(\hat{w}_1|x) = \sum_{i=1}^2 \Lambda[\hat{w}_1, w_i] P(w_i|x) = \lambda_2 P(w_2|x)$$

$$R(\hat{w}_2|x) = \sum_{i=1}^2 \Lambda[\hat{w}_2, w_i] P(w_i|x) = \lambda_1 P(w_1|x)$$

那么决策策略就是两种风险相等时：

$$\frac{\lambda_2}{\lambda_1} \cdot \frac{P(w_1)}{P(w_2)} = \frac{P(x|w_2)}{P(x|w_1)}$$

也就是如果 $\frac{P(x|w_2)}{P(x|w_1)} > \frac{\lambda_2}{\lambda_1} \cdot \frac{P(w_1)}{P(w_2)}$, 选择最优分类为 \hat{w}_1 ; 否则选择 \hat{w}_2 。带入高斯函数进一步计算得到分类面：

$$x = \frac{1}{2(\mu_2 - \mu_1)} \cdot [\mu_2^2 - \mu_1^2 + 2\sigma^2 \ln(\frac{\lambda_2 P(w_2)}{\lambda_1 P(w_1)})]$$

进一步换算可以得到：

$$x = \frac{\mu_1 + \mu_2}{2} + \frac{\sigma^2}{\mu_2 - \mu_1} \cdot \ln(\frac{\lambda_2 P(w_2)}{\lambda_1 P(w_1)})$$

也就是会产生分类面的偏移。

2.2.3 给定一类错误率优化另一类错误率

规定第 i 类错误率的数值不能大于某一个数值 ϵ , 进而优化另一个错误率。因此可以构建拉格朗日函数：

$$\mathcal{L} = P_1(e) + \lambda(P_2(e) - \epsilon)$$

将上述的函数对于分类边界求导就能够得到最优的分类面。这里我们还是为了简单化, 选择均值不同方差相同的高斯函数进行具体的推导如下所示 (不妨假设 $\mu_1 < \mu_2$)：

$$P_1(e) = \int_{x \in \mathcal{R}_2} p(x|w_1) dx = \int_t^{+\infty} \frac{1}{\sqrt{2\pi}\sigma} \exp[-\frac{(x-\mu_1)^2}{2\sigma^2}]$$

$$P_2(e) = \int_{x \in \mathcal{R}_1} p(x|w_2) dx = \int_{-\infty}^t \frac{1}{\sqrt{2\pi}\sigma} \exp[-\frac{(x-\mu_2)^2}{2\sigma^2}]$$

带入上面的拉格朗日函数并对 t 求导可以得到：

$$\frac{\partial P_1(e)}{\partial t} = -\frac{1}{\sqrt{2\pi}\sigma} \exp[-\frac{(t-\mu_1)^2}{2\sigma^2}]$$

$$\frac{\partial P_2(e)}{\partial t} = \frac{1}{\sqrt{2\pi}\sigma} \exp[-\frac{(t-\mu_2)^2}{2\sigma^2}]$$

$$\frac{\partial \mathcal{L}}{\partial t} = \frac{1}{\sqrt{2\pi}\sigma} \left\{ \lambda \exp\left[-\frac{(t - \mu_2)^2}{2\sigma^2}\right] - \exp\left[-\frac{(t - \mu_1)^2}{2\sigma^2}\right] \right\} = 0$$

进一步化简可以得到分类面：

$$t = \frac{\mu_1 + \mu_2}{2} + \frac{\sigma^2}{\mu_1 - \mu_2} \cdot \ln(\lambda)$$

同样产生了分类面的偏移 (λ 通过上式带入 $P_2(e) = \epsilon$ 得到)。

3 概率密度函数估计

3.1 概率密度函数估计问题的建立

第一个章节的内容主要是讲述了朴素贝叶斯决策理论，其建立在已知各个类别的总体概率分布已知、分类数已知的两个基础上。但是事实上在大部分的现实问题中我们并不清楚真实的各个类别概率分布，也因此需要使用数据的方式对于概率分布进行估计。这也就引入了概率密度函数估计问题。

那么既然引入了概率密度函数估计问题，我们首先需要知道的是：这个估计能不能做出来，能不能找到最优的一组参数。因此引入一个概念：可识别性。可识别性的严谨定义是：对于参数空间的两个取值 θ 和 θ' ，如果存在一个 x 使得 $p(x|\theta) \neq p(x|\theta')$ ，那么称这个概率分布函数是可识别的。

一般而言，离散随机变量的混合密度函数是不可识别的；大部分连续随机变量的混合密度函数是可识别的。下面给出一个离散随机变量的混合密度函数不可识别的例子：

$$p(x|\theta) = \frac{1}{2}\theta_1^x(1-\theta_1)^{1-x} + \frac{1}{2}\theta_2^x(1-\theta_2)^{1-x}$$

上式之中的 $x \in \{0, 1\}$ 。清楚地看到，上面的分布函数只有两种情况：

$$p(x|\theta) = \begin{cases} \frac{1}{2}(\theta_1 + \theta_2) & x = 1 \\ 1 - \frac{1}{2}(\theta_1 + \theta_2) & x = 0 \end{cases}$$

针对一对 $\theta_1 + \theta_2 = \theta'_1 + \theta'_2$ 的参数，是不存在 x 能够区分两种情形的，因此不可分辨。

【发散性思维】 其实可分辨这个问题蛮有意思的，我觉得可分辨这个条件和函数的线性相关性之间可能有一定的关系。函数在区间 (a, b) 内线性相关的定义是存在一组不全为 0 的 $\{\alpha_1, \alpha_2, \dots, \alpha_m\}$ 使得 $\sum_{i=1}^m \alpha_i y_i(x) = 0, \forall x \in (a, b)$ 。不可分辨意味着在参数空间内的概率密度函数族是线性不相关的（不可分辨其实就是 $\alpha_1 = 1$ 且 $\alpha_2 = -1$ 的特殊情形）。

3.2 概率密度函数估计的衡量标准

【注意】：衡量标准都是针对确定性参数估计来说的！非参数估计或者把参数视为随机变量的估计方法是不能够使用这种方式衡量的，因为不存在“参数的真值”。

3.2.1 按照均值衡量

按照均值衡量主要分为三种：无偏估计、渐进无偏估计和有偏估计。无偏估计是 $E(\hat{\theta}) = \theta$ ，渐进无偏估计是 $\lim_{N \rightarrow \infty} E(\hat{\theta}) = \theta$ 。这里需要注意：使用同一种方法估计出来的参数不一定都是同样的偏差方式！好比最大似然估计针对高斯函数的均值估计是无偏的、但是针对高斯函数方差的估计是有偏的，需要进行贝塞尔修正。

3.2.2 按照方差衡量

方差主要衡量估计参数的稳定性，方差越小，说明参数估计的稳定性越高。

3.3 参数估计

3.3.1 贝叶斯最优参数估计

贝叶斯最优参数估计的前提假设是：待估计的参数 θ 是一个随机变量，并且这个参数的先验分布是已知的。

贝叶斯最优参数估计的优化目标是：

$$\hat{\theta} = \arg \max_{\theta} p(\theta | \{x_1, x_2, \dots, x_N\})$$

这就是本质了。优化的方式一般选择直接求导，但是也会遇到不能求导的情况，好比估计均匀分布的最优参数，放在本部分的最后一个小节举例计算。

首先对于上面的原始式进行推导，带入贝叶斯公式、在各个样本互相独立时可以得到：

$$\hat{\theta} = \arg \max_{\theta} \frac{p(\{x_1, x_2, \dots, x_N\} | \theta) p(\theta)}{p(\{x_1, x_2, \dots, x_N\})}$$

$$\hat{\theta} = \arg \max_{\theta} p(\{x_1, x_2, \dots, x_N\} | \theta) p(\theta)$$

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

由于贝叶斯估计的假设中已知 θ 的先验分布并且已知 x 概率密度模型，因此上面的式子可以写成解析形式然后求导。

这里我们以高斯分布为例进行求解，不妨假设我们需要估计 $x \sim N(\mu, \sigma^2)$ 分布之中的 μ ，同时 $\mu \sim N(\mu_0, \sigma_0^2)$ ，那么：

$$\begin{aligned}\hat{\mu} &= \arg \max_{\mu} \frac{1}{\sqrt{2\pi}\sigma_0} \exp\left[-\frac{(\mu - \mu_0)^2}{2\sigma_0^2}\right] \cdot \prod_{i=1}^N \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(x_i - \mu)^2}{2\sigma^2}\right] \\ \hat{\mu} &= \arg \max_{\mu} A \cdot \exp\left\{-\frac{1}{2} \left[\left(\frac{\mu - \mu_0}{\sigma_0}\right)^2 + \sum_{i=1}^N \left(\frac{x_i - \mu}{\sigma}\right)^2 \right]\right\} \\ \hat{\mu} &= \arg \max_{\mu} B \cdot \exp\left[-\frac{1}{2} \left(\frac{\mu - \mu_N}{\sigma_N}\right)^2\right] \\ \mu_N &= \frac{N\sigma_0^2}{N\sigma_0^2 + \sigma^2} m_N + \frac{\sigma^2}{N\sigma_0^2 + \sigma^2} \mu_0; \quad \sigma_N = \frac{\sigma^2 \sigma_0^2}{N\sigma_0^2 + \sigma^2}\end{aligned}$$

显然，当 $\hat{\mu} = \mu_N$ 时可以取得最大值。从上式之中可以看到实际上贝叶斯最优参数估计是在直接使用样本的最大似然估计的最优值 $\hat{\mu}_{MLE} = \frac{1}{N} \sum_{i=1}^N x_i$ 和先验估计的最优值 $\hat{\mu}_{pre} = \mu_0$ 之间进行了加权。需要注意的是：如果先验函数是可靠的，那么贝叶斯最优估计的参数是优于最大似然的；但是在对于先验一无所知时，最大似然显然是最优选择。

3.3.2 最大似然估计

最大似然估计的前提假设是：待估计的参数 θ 是一个确定的未知量。

最大似然估计的优化目标是：

$$\hat{\theta} = \arg \max_{\theta} p(\{x_1, x_2, \dots, x_N\} | \theta)$$

特殊地，上式之中的 $p(\{x_1, x_2, \dots, x_N\} | \theta) = l(\theta)$ ，也被称为似然函数。由于往往遇到指数型分布，为了使得对于指数型分布处理方便，常用对数似然函数表示：

$$\hat{\theta} = \arg \max_{\theta} \ln [p(\{x_1, x_2, \dots, x_N\} | \theta)] = \arg \max_{\theta} H(\theta)$$

3.3.3 不能求导的均匀分布例子

对于均匀分布如下所示：

$$p(x|\theta) = \begin{cases} \frac{1}{\theta_2 - \theta_1} & \theta_1 < x < \theta_2 \\ 0 & Otherwise \end{cases}$$

似然函数是：

$$l(x) = \begin{cases} \frac{1}{(\theta_2 - \theta_1)^N} & \{x_1, x_2, \dots, x_N\} \in (\theta_1, \theta_2) \\ 0 & \text{Otherwise} \end{cases}$$

对于 θ_1 求偏导得到：

$$\frac{\partial H(\theta_1, \theta_2)}{\partial \theta_1} = N \cdot \frac{1}{\theta_2 - \theta_1} \neq 0$$

因此不能求导来求取最大，只能按照定义找 $\frac{1}{(\theta_2 - \theta_1)^N}$ 最大的情况，这种情况是很好找的：

$$\begin{aligned}\hat{\theta}_1 &= \min\{x_1, x_2, \dots, x_N\} \\ \hat{\theta}_2 &= \max\{x_1, x_2, \dots, x_N\}\end{aligned}$$

3.4 非参数估计

3.4.1 Parzen 窗估计

Parzen 窗估计方法的前提假设是：不知道分布函数形式、 N 个样本独立同分布。

基本思路是在某一个小体积范围内，用样本出现的频率来代替概率，这样计算得到的概率在除以区间体积，就是概率密度函数。因此最基本的表达式是：

$$\begin{aligned}\hat{p}(x) &= \lim_{V \rightarrow 0} \frac{1}{NV} k(x) \\ \lim_{N \rightarrow \infty} V &= 0; \quad \lim_{N \rightarrow \infty} k_N = \infty; \quad \lim_{N \rightarrow \infty} \frac{k_N}{N} = 0\end{aligned}$$

上式之中 N 为样本总量， $k(x)$ 是在 x 处体积为 V 邻域内的样本点个数。

上面的基本表达式显然是存在一定问题的，一方面是选取区间长度较小时容易出现“毛刺”，另一方面是只考虑区间内的点对于分布的影响，而不考虑其他点的影响。针对这些，我们对点进行“加窗”操作。如果一个样本 $x_i = [u_1^{(i)} \ u_2^{(i)} \ \dots \ u_d^{(i)}]$ ，我们选取的邻域是空间内以 h 为边长的超立方体，窗函数为 ϕ ，那么概率密度函数的估计就是：

$$\hat{p}(x) = \frac{1}{NV} \sum_{i=1}^N \phi\left(\frac{x - x_i}{h}\right)$$

其中窗函数应当满足：

$$\int \phi(v) dv = 1$$

上式中 v 是零均值的归一化样本。

Parzen 窗的窗宽选择应当满足下式：

$$\lim_{N \rightarrow \infty} V_N = 0; \quad \lim_{N \rightarrow \infty} NV_N = \infty$$

含义是一方面要尽可能小一些；另一方面不能太小，要能够使得样本在每个区间内都有充足的点数。

3.4.2 最近邻估计

最近邻估计就是上一个部分之中的朴素形式。

3.5 概率密度函数估计中的关键问题与改进

3.5.1 概率密度函数估计的准确性与分类器性能的关系

分类器的错误主要有三种：贝叶斯错误、模型错误与估计误差。贝叶斯错误是分类器的固有属性，就是第一章之中我们推导的模型错误率；模型错误是指进行概率密度函数估计时选择了错误的模型；估计误差是由概率密度函数估计时样本量不足导致参数估计并不精确导致的。

贝叶斯错误取决于分类器，是不能消除的；模型错误要求我们尝试多种模型的估计效果；估计误差可以通过增加样本量的方式缓解，但是对于固定参数的估计方法是存在上界的：克拉美罗界（Cramer-Rao Lower Bound, CLB）。

3.5.2 维数问题

在进行估计时，对于一个维度的概率分布函数，往往采用数百个样本效果就已经较好。但是对于 d 个维度的概率密度函数估计，需要 100^d 个样本，这个数量是相当大的。我们可以通过假设各个维度之间的概率密度分布函数之间互相独立来缓解，此时只需要对每一个维度进行估计，这种假设被称为朴素贝叶斯方法：

$$\hat{p}(x) = \hat{p}(u_1)\hat{p}(u_2) \cdots \hat{p}(u_d)$$

3.5.3 过拟合与模型选择

在进行函数估计时最常面对的问题就是过拟合，过拟合的本质原因在于模型待定参数数目大于样本量。主要的缓解方法时：增大数据样本量；引入正则化惩罚；在算法泛化能力类似的情形下选择参数量少的模型（“如非必要

无增实体”)。为了衡量模型的效果，引入两个信息衡量准则：赤池信息准则 (AIC) 和贝叶斯信息量准则 (BIC)：

$$AIC = \frac{2}{N}(k - L)$$

$$BIC = k \ln(N) - 2L$$

上面两个式子之中 N 代表样本量， k 为模型参数量， L 是模型对数似然值。AIC 越低，模型越好；BIC 越低，模型越好。

3.6 概率密度函数估计应用实例：错误率估计

理论上直接推导错误率存在一定的困难，尤其是对于深度神经网络。因此从测试集的表现来估计错误率就是十分重要的问题。

针对一个先验概率未知的二分类问题，直接估计总错误率为 $\hat{\epsilon} = \frac{k}{N}$ ，其中 k 是分错的样本数。这是一个无偏估计，这个估计的来源如下所示，首先给出 N 个样本中出现 k 个分错样本的概率：

$$P(k) = C_N^k \epsilon^k (1 - \epsilon)^{N-k}$$

似然函数为：

$$\frac{\partial \ln[P(k)]}{\partial \epsilon} = \frac{k}{\epsilon} - \frac{N-k}{1-\epsilon} = 0$$

解出 $\hat{\epsilon} = \frac{k}{N}$ 。这里我们认为样本被分错的数量 k 服从一个二项分布：

$$k \sim B(N, \epsilon)$$

也因此 $E(\hat{\epsilon}) = \epsilon$ 且 $Var(\hat{\epsilon}) = \frac{N\epsilon(1-\epsilon)}{N^2} = \frac{\epsilon(1-\epsilon)}{N}$ ，是无偏估计。

针对两种类别先验概率已知的二分类问题，直接估计总错误率为 $\hat{\epsilon} = P(w_1)\hat{\epsilon}_1 + P(w_2)\hat{\epsilon}_2$ ，这是十分显然的，同样是无偏估计，方差：

$$Var(\hat{\epsilon}) = \frac{1}{N}[P(w_1)\epsilon_1(1-\epsilon_1) + P(w_2)\epsilon_2(1-\epsilon_2)]$$

4 混合高斯模型与 EM 算法

4.1 混合高斯模型

混合高斯模型是指如下式所示的概率分布模型：

$$p(x|\Theta) = \sum_{i=1}^c \alpha_i p_i(x|\mu_i, \Sigma_i)$$

$$\Theta = \{\alpha_1, \alpha_2, \dots, \alpha_c, \mu_1, \mu_2, \dots, \mu_c, \Sigma_1, \Sigma_2, \dots, \Sigma_c\}$$

如果直接对于混合高斯模型求取最大似然估计的话，将会是如下形式：

$$l(\Theta) = \prod_{j=1}^N \sum_{i=1}^c \alpha_i p_i(x|\mu_i, \Sigma_i)$$

$$\ln[l(\Theta)] = \sum_{j=1}^N \ln \left[\sum_{i=1}^c \alpha_i p_i(x|\mu_i, \Sigma_i) \right]$$

进而对上式求导得到：

$$\frac{\partial \ln[l(\Theta)]}{\partial \mu_k} = \sum_{i=1}^N \frac{\alpha_k p(x_i|\mu_k, \Sigma_k)}{\sum_{j=1}^c \alpha_j p(x_i|\mu_j, \Sigma_j)} \Sigma_k^{-1} (x_i - \mu_k) = 0$$

可以有如下记法：

$$P(w_k|x_i, \mu_k, \Sigma_k) = \frac{p(x_i|\mu_k, \Sigma_k)}{\sum_{j=1}^c \alpha_j p(x_i|\mu_j, \Sigma_j)}$$

于是：

$$\mu_k = \frac{\sum_{i=1}^N P(w_k|x_i, \mu_k, \Sigma_k) x_i}{\sum_{i=1}^N P(w_k|x_i, \mu_k, \Sigma_k)}$$

同样的方法对于方差 Σ_k 求导可以得到：

$$\Sigma_k = \frac{\sum_{i=1}^N P(w_k|x_i, \mu_k, \Sigma_k) (x_i - \mu_k)(x_i - \mu_k)^T}{\sum_{i=1}^N P(w_k|x_i, \mu_k, \Sigma_k)}$$

4.2 EM 算法

EM 算法的本质是使用迭代求解最大似然估计或贝叶斯最优估计的算法。

EM 算法的应用场景是：第一种，数据之中存在数据缺失；第二种，似然函数并不能够解析求解。

EM 算法的基本思想是通过最大化似然函数的下界，从而“提升”最大似然函数的取值。由于式子中存在隐含变量，因此在优化时采用分步骤优化的准则：首先固定参数 Θ 、并调整 z 隐含参数；之后固定隐含参数 z 、调整 Θ 从而达到整体最大值。这样的思路事实上在通信领域中也是很常见的。如下式所示是能够这样做的理由：

$$l(\Theta) = \prod_{i=1}^N p(x_i|\theta)$$

$$H(\Theta) = \sum_{i=1}^N \ln[p(x_i|\theta)] = \sum_{i=1}^N \ln \left[\sum_{z_i} p(x_i, z_i|\theta) \right]$$

$$H(\Theta) = \sum_{i=1}^N \ln[p(x_i|\theta)] = \sum_{i=1}^N \ln \left[\int_{z_i} p(x_i, z_i|\theta) dz_i \right]$$

上式中的第二个等号是通过边缘分布求取保证成立的。那么由琴生不等式：

$$H(\Theta) = \sum_{i=1}^N \ln \left[\sum_{z_i} q(z_i) \frac{p(x_i, z_i|\theta)}{q(z_i)} \right] \geq \sum_{i=1}^N \sum_{z_i} q(z_i) \ln \left[\frac{p(x_i, z_i|\theta)}{q(z_i)} \right] = F(q, \theta)$$

在上述琴生不等式之中 $\sum_{z_i} q(z_i) = 1$ 。下面我们来讨论琴生不等式的取等条件：

$$\frac{p(x_i, z_i|\theta)}{q(z_i)} = c, \quad \forall z_i$$

那么对上式进一步求取和化简：

$$p(x_i, z_i|\theta) = c \cdot q(z_i)$$

$$\sum_{z_i} p(x_i, z_i|\theta) = c \sum_{z_i} q(z_i)$$

$$c = p(x_i|\theta)$$

于是 $\hat{q}(z_i) = \frac{p(x_i, z_i|\theta)}{p(x_i|\theta)} = p(z_i|x_i, \theta)$ 时取等，同时 $F(\hat{q}, \theta)$ 取为最大值为 $H(\Theta)$ 。以上就是 E 步骤的全部内容，本质就是固定 θ ，通过调整 $q(z_i)$ 优化 $F(q, \theta)$ 。下面是 M 步骤，固定刚刚求出来的 $q(z_i)$ ，通过调整 θ 优化 $F(q, \theta)$ ：

$$F(q, \theta) = \sum_{i=1}^N \sum_{z_i} q(z_i) \ln \left[\frac{p(x_i, z_i|\theta)}{q(z_i)} \right]$$

由于我们的优化目标是通过调整 θ 来优化 $F(q, \theta)$ 也就有如下式：

$$F(q, \theta) = \sum_{i=1}^N \sum_{z_i} q(z_i) \ln[p(x_i, z_i|\theta)] - \sum_{i=1}^N \sum_{z_i} q(z_i) \ln[q(z_i)]$$

$$G(q, \theta) = \sum_{i=1}^N \sum_{z_i} q(z_i) \ln[p(x_i, z_i|\theta)]$$

即此时优化 $F(q, \theta)$ 与优化 $G(q, \theta)$ 是等价的（因为后面那一项之中与 θ 是无关的）。

于是至此可以完整地写下 EM 算法的迭代过程：

- E-Step:

$$\hat{q}_{[k+1]}(z_i) = p(z_i|x_i, \theta_{[k]})$$

- M-Step:

$$\hat{\theta}_{[k+1]} = \arg \max_{\theta} \sum_{i=1}^N \sum_{z_i} \hat{q}_{[k+1]}(z_i) \ln [p(x_i, z_i | \theta)]$$

4.3 EM 算法在混合高斯模型上的应用

- E-Step:

$$\begin{aligned}\hat{q}_{[k+1]}(z_i) &= p(z_i|x_i, \theta_{[k]}) = \frac{p(x_i, z_i | \theta_{[k]})}{\sum_{z_i} p(x_i, z_i | \theta_{[k]})} \\ \hat{q}_{[k+1]}(z_i) &= \frac{\alpha_{z_i}[k] \cdot N(x_i | \theta_{[k]}^{(z_i)})}{\sum_{j=1}^c \alpha_j[k] \cdot N(x_i | \theta_{[k]}^{(j)})}\end{aligned}$$

- M-Step:

$$\begin{aligned}G(q, \theta) &= \sum_{i=1}^N \sum_{z_i} \frac{\alpha_{z_i}[k] \cdot N(x_i | \theta_{[k]}^{(z_i)})}{\sum_{j=1}^c \alpha_j[k] \cdot N(x_i | \theta_{[k]}^{(j)})} \cdot \ln [\alpha_{z_i} \cdot N(x_i | \theta^{(z_i)})] \\ G(q, \theta) &= \sum_{i=1}^N \sum_{l=1}^c \frac{\alpha_l[k] \cdot N(x_i | \theta_{[k]}^{(l)})}{\sum_{j=1}^c \alpha_j[k] \cdot N(x_i | \theta_{[k]}^{(j)})} \cdot \ln [\alpha_l \cdot N(x_i | \theta^{(l)})]\end{aligned}$$

那么首先为了求取 $\hat{\alpha}_l[k+1]$ 的最优值，由于需要满足 $\sum_{l=1}^c \alpha_l = 1$ ，因此需要对 $G(q, \theta)$ 构建拉格朗日函数然后再求导：

$$\begin{aligned}\mathcal{L}(G, \alpha) &= G(q, \theta) + \lambda \left(\sum_{l=1}^c \alpha_l - 1 \right) \\ \frac{\partial \mathcal{L}(G, \alpha)}{\partial \alpha_l} &= \sum_{i=1}^N \frac{\alpha_l[k] \cdot N(x_i | \theta_{[k]}^{(l)})}{\sum_{j=1}^c \alpha_j[k] \cdot N(x_i | \theta_{[k]}^{(j)})} \cdot \frac{1}{\hat{\alpha}_l[k+1]} + \lambda = 0 \\ \lambda \hat{\alpha}_l[k+1] &= - \sum_{i=1}^N \frac{\alpha_l[k] \cdot N(x_i | \theta_{[k]}^{(l)})}{\sum_{j=1}^c \alpha_j[k] \cdot N(x_i | \theta_{[k]}^{(j)})} \\ \lambda &= - \sum_{l=1}^c \sum_{i=1}^N \frac{\alpha_l[k] \cdot N(x_i | \theta_{[k]}^{(l)})}{\sum_{j=1}^c \alpha_j[k] \cdot N(x_i | \theta_{[k]}^{(j)})} = - \sum_{i=1}^N \frac{\sum_{l=1}^c \alpha_l[k] \cdot N(x_i | \theta_{[k]}^{(l)})}{\sum_{j=1}^c \alpha_j[k] \cdot N(x_i | \theta_{[k]}^{(j)})} = -N\end{aligned}$$

$$\hat{\alpha}_l[k+1] = \frac{1}{N} \sum_{i=1}^N \frac{\alpha_l[k] \cdot N(x_i | \theta_{[k]}^{(l)})}{\sum_{j=1}^c \alpha_j[k] \cdot N(x_i | \theta_{[k]}^{(j)})}$$

接着是求 $\hat{\mu}_l[k+1]$, 直接对于 $G(q, \theta)$ 求导:

$$\begin{aligned}\frac{\partial G(q, \theta)}{\partial \mu_l} &= \sum_{i=1}^N \hat{\Sigma}_l^{-1}(x_i - \hat{\mu}_l) \cdot \frac{\alpha_l[k] \cdot N(x_i | \theta_{[k]}^{(l)})}{\sum_{j=1}^c \alpha_j[k] \cdot N(x_i | \theta_{[k]}^{(j)})} = 0 \\ \hat{\mu}_l[k+1] &= \frac{\sum_{i=1}^N x_i p(l|x_i, \theta_{[k]})}{\sum_{i=1}^N p(l|x_i, \theta_{[k]})}\end{aligned}$$

最后求 $\hat{\Sigma}_l[k+1]$, 直接对于 $G(q, \theta)$ 求导得到:

$$\hat{\Sigma}_l[k+1] = \frac{\sum_{i=1}^N (x_i - \mu_l[k+1])(x_i - \mu_l[k+1])^T p(l|x_i, \theta_{[k]})}{\sum_{i=1}^N p(l|x_i, \theta_{[k]})}$$

4.4 扩展的 EM 算法 (Generalized EM)

很简单, 意思就是不一定非要找到 $\hat{\theta}[k+1]$ 使得 $G(q, \theta)$ 最大化, 而是只要找到一个 $\theta[k+1]$ 使得 $G(q, \theta[k+1]) > G(q, \theta[k])$ 成立即可。

5 支持向量机

5.1 生成式模型与判别式模型

本节主要介绍支持向量机理论。那么第一个问题是: 为什么从上面的朴素贝叶斯、到概率密度函数估计、再到 EM 算法和混合高斯模型, 突然到这里蹦出来一个似乎并不基于“概率”表示的方法呢? 这个问题的根源在于, 前文的主要分类模式是“**生成式模型**”, 此后要着重讲述的分类模式是“**判别式模型**”。

生成式模型的特点在于对后验概率进行建模, 使用数据估计出变量的联合分布, 再从联合分布推导出条件概率从而分类。包括朴素贝叶斯、混合高斯模型、贝叶斯信念网络、马尔可夫随机场等。优点在于: 具备更加丰富的分布信息、能用于数据不完整的情况 (例如使用 GAN 进行数据扩充等等); 缺点在于: 训练难度大、学习效果不好 (是指模型分辨率不高, 容易把相似的分布统一化)。

判别式模型的特点在于不对分布建模, 而是直接估计条件概率进行分类。包

括：**支持向量机**、多层感知机、神经网络、最近邻、决策树等方法。优点在于：分类边界更加灵活、分辨率更高（特征提取能力强，能够发掘细微处的特征差距）、任务简单容易学习；缺点在于：不能反映数据真实的分布模型、可解释性弱、鲁棒性低（不一定能够达到最好效果，由数据决定）。

那么就让支持向量机模型成为学习判别式模型的开始，探索判别式模型的数学原理。

5.2 严格线性可分情形的支持向量机

在严格线性可分情形下，支持向量机的基本思路是找到一个分类面 G ，使得分类面两侧的支持向量到分类面的垂直距离是最远的。对于一个分类面和不是支持向量的样本 (\mathbf{x}_i, y_i) ，我们可以给出下式成立：

$$\begin{cases} \mathbf{w}_0^T \mathbf{x}_i + b_0 > 0 & y_i = 1 \\ \mathbf{w}_0^T \mathbf{x}_i + b_0 < 0 & y_i = -1 \end{cases}$$

进而可以找到一个 $\alpha > 0$ 使得下式成立：

$$\begin{cases} \mathbf{w}_0^T \mathbf{x}_i + b_0 > \alpha & y_i = 1 \\ \mathbf{w}_0^T \mathbf{x}_i + b_0 < -\alpha & y_i = -1 \end{cases}$$

将上式归一化得到：

$$\begin{cases} \mathbf{w}^T \mathbf{x}_i + b > 1 & y_i = 1 \\ \mathbf{w}^T \mathbf{x}_i + b < -1 & y_i = -1 \end{cases}$$

$$y_i \cdot (\mathbf{w}^T \mathbf{x}_i + b) > 1$$

如果 x_i 是支持向量，那么：

$$y_i \cdot (\mathbf{w}^T \mathbf{x}_i + b) = 1$$

下面求取支持向量到分类面的距离如下：

$$d_s = \frac{|\mathbf{w}^T \mathbf{x}_s + b|}{\|\mathbf{w}\|} = \frac{1}{\|\mathbf{w}\|}$$

于是支持向量机的优化问题可以直接被写为如下标准形态：

$$\hat{\mathbf{w}} = \arg \min_{\mathbf{w}} \frac{1}{2} \mathbf{w}^T \mathbf{w}$$

$$\text{s.t. } y_i \cdot (\mathbf{w}^T \mathbf{x}_i + b) \geq 1, \forall (\mathbf{x}_i, y_i)$$

为了求解这样的优化问题，我们写出拉格朗日函数形式：

$$\mathcal{L} = \frac{1}{2} \mathbf{w}^T \mathbf{w} - \sum_{i=1}^N \lambda_i [y_i \cdot (\mathbf{w}^T \mathbf{x}_i + b) - 1]$$

求导可以得到：

$$\begin{cases} \mathbf{w} = \sum_{i=1}^N \lambda_i y_i \mathbf{x}_i \\ \sum_{i=1}^N \lambda_i y_i = 0 \end{cases}$$

可以进一步求取 $\frac{1}{2} \mathbf{w}^T \mathbf{w}$ 为：

$$\mathbf{w}^T \mathbf{w} = \left(\sum_{i=1}^N \lambda_i y_i \mathbf{x}_i \right)^T \left(\sum_{i=1}^N \lambda_i y_i \mathbf{x}_i \right) = \sum_{i=1}^N \sum_{j=1}^N \lambda_i \lambda_j y_i y_j \mathbf{x}_i^T \cdot \mathbf{x}_j$$

于是转换为对偶问题：

$$\begin{aligned} \hat{\alpha}_i &= \arg \max_{\alpha_i} \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \lambda_i \lambda_j y_i y_j \mathbf{x}_i^T \cdot \mathbf{x}_j \\ \text{s.t. } & \sum_{i=1}^N \alpha_i y_i = 0, \quad \alpha_i \geq 0, \quad \forall (\mathbf{x}_i, y_i) \end{aligned}$$

于是分类器构造为：

$$f(x) = \operatorname{sgn} \left(\sum_{i=1}^N \lambda_i y_i \mathbf{x}_i^T \cdot \mathbf{x} + b \right)$$

5.3 线性不可分情形的支持向量机

线性不可分情形与非线性情形存在很大的区别，线性不可分并不是真正的线性不可分，而是其实也是线性可分的，但是只不过认为某些样本的错分是松弛。这样的做法一定程度上能够提升模型的泛化能力、提升鲁棒性。

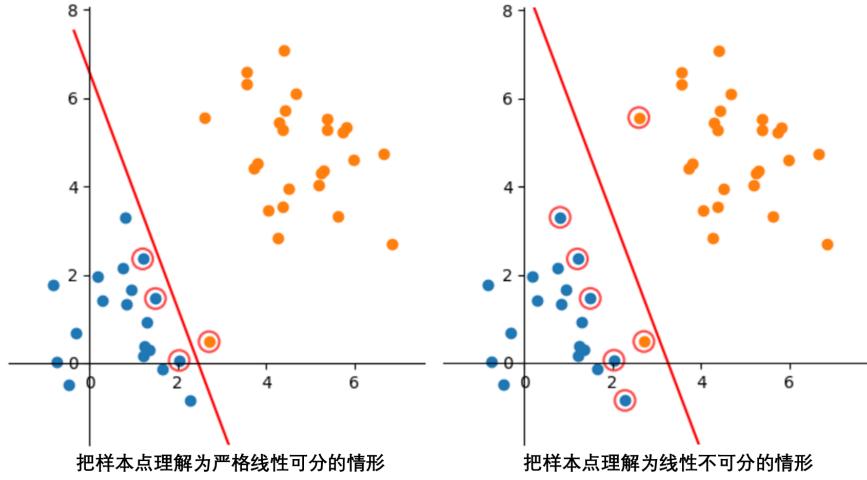


Fig 1. 严格线性可分与线性不可分

为了在优化目标之中体现这种松弛思想，引入 C 和 ξ_i :

$$\hat{\mathbf{w}} = \arg \min_{\mathbf{w}} \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^N \xi_i$$

$$\text{s.t. } y_i \cdot (\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \forall (\mathbf{x}_i, y_i)$$

上式之中的 $\sum_{i=1}^N \xi_i$ 是错分样本率的上界； C 是模型复杂程度与能够容忍的错分数目的 trade-off。

其对偶问题为：

$$\hat{\alpha}_i = \arg \max_{\alpha_i} \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \lambda_i \lambda_j y_i y_j \mathbf{x}_i^T \cdot \mathbf{x}_j$$

$$\text{s.t. } \sum_{i=1}^N \alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq C, \quad \forall (\mathbf{x}_i, y_i)$$

5.4 非线性情形的支持向量机

这里我们假设真实的最优分类面是一个圆： $x_1^2 + x_2^2 = 1$ ，那么我们进行如下的空间映射：

$$\Phi : \mathcal{R}^2 \mapsto \mathcal{H}$$

$$\Phi \left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \right) = \begin{bmatrix} x_1^2 \\ x_1 x_2 \\ x_2^2 \end{bmatrix}$$

既然有了上述的映射关系，我们观察原本分类面在这一组映射作用下，映射到新空间之中的函数，假设我们记新空间中的三个维度坐标为 $\begin{bmatrix} \xi_1 & \xi_2 & \xi_3 \end{bmatrix}$ ，那么分类面为：

$$\xi_1 + \xi_3 = 1$$

这个 Φ 映射把原空间 \mathcal{R}^2 中的圆映射为 \mathcal{H} 之中的一面，因此完成了非线性分类问题的线性化。

上面的变换 Φ 函数的数学形式实际上并不能够用矩阵表示出来，也就是说找不到一组矩阵能够使得下面的式子成立。这个原因是很简单的，因为矩阵算子对应的变换是线性变换，此处的变换是非线性的，因此并不好处理。事实上，大部分的非线性映射，我们甚至不容易知道 Φ 的表示形式。

$$\begin{bmatrix} x_1^2 \\ x_1 x_2 \\ x_2^2 \end{bmatrix} = \mathbf{A} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \cdot \mathbf{B} \cdot \begin{bmatrix} x_1 & x_2 \end{bmatrix} \cdot \mathbf{C}$$

我们真正能够自己设计的参数只有：我们希望如何把分类面映射成线性。因此如何映射空间中每一个坐标点、获知 Φ 函数并不重要，重要的是这个 Φ 如何把分类面变成线性的。现在我们来观察上文给出的分类面形式：

$$\begin{aligned} \hat{\alpha}_i &= \arg \max_{\alpha_i} \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \lambda_i \lambda_j y_i y_j \mathbf{x}_i^T \cdot \mathbf{x}_j \\ \text{s.t. } & \sum_{i=1}^N \alpha_i y_i = 0, \quad \alpha_i \geq 0, \quad \forall (\mathbf{x}_i, y_i) \\ f(x) &= \operatorname{sgn} \left(\sum_{i=1}^N \lambda_i y_i \mathbf{x}_i^T \cdot \mathbf{x} + b \right) \end{aligned}$$

不难发现，上述分类面求取过程中与样本有关的所有运算都是内积运算：不论是优化目标之中的 $\mathbf{x}_i^T \cdot \mathbf{x}_j$ 还是分类面之中的 $\mathbf{x}_i^T \cdot \mathbf{x}$ 。因此我们的关键问题变为如何设计原空间内的样本内积使其在高维空间内产生线性分类面。

假设我们记原空间之中 \mathbf{x}_1 与 \mathbf{x}_2 的内积映射到高维空间内 $\Phi(\mathbf{x}_1)$ 和 $\Phi(\mathbf{x}_2)$ 内积的映射为：

$$K : \langle \mathbf{x}_1, \mathbf{x}_2 \rangle \mapsto \langle \Phi(\mathbf{x}_1), \Phi(\mathbf{x}_2) \rangle$$

那么在进行非线性映射之后的 SVM 问题变为如下形式：

$$\begin{aligned}\hat{\alpha}_i &= \arg \max_{\alpha_i} \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \lambda_i \lambda_j y_i y_j \cdot K(\mathbf{x}_i, \mathbf{x}_j) \\ \text{s.t. } & \sum_{i=1}^N \alpha_i y_i = 0, \quad \alpha_i \geq 0, \quad \forall (\mathbf{x}_i, y_i) \\ f(x) &= \operatorname{sgn} \left(\sum_{i=1}^N \lambda_i y_i \cdot K(\mathbf{x}_i, \mathbf{x}) + b \right)\end{aligned}$$

这就是对于非线性问题的 SVM。还是很简单的，不过为了使得阐述更加明确，这里还是详细介绍一下上面引入的核函数 K 。首先从空间的角度讲，我们刚刚引入的 K 是一个从实数空间到实数空间的映射，因为 $\langle \mathbf{x}_1, \mathbf{x}_2 \rangle$ 和 $\langle \Phi(\mathbf{x}_1), \Phi(\mathbf{x}_2) \rangle$ 都是内积，本质上就是两个实数。（事实上如果写成更加广泛的形式， $K(\mathbf{x}, \mathbf{y})$ 是一个双变量的、从原空间映射到实数空间的函数）。接着从形式上来讲，上面的映射形式看起来仿佛比较复杂，实际上也可以简写为 $K(\mathbf{x}_1, \mathbf{x}_2) = \langle \Phi(\mathbf{x}_1), \Phi(\mathbf{x}_2) \rangle$ 。

既然介绍了核函数，就需要进一步说明核函数并不是随意选取的，一方面核函数需要具备把原空间内的非线性分类面近似映射为高维空间内的线性分类面的能力；另一方面核函数必须满足泛函的一些性质，即 Mercer's Condition。

- **Mercer's Condition:** 对任意满足 L_2 条件（即函数的勒贝格积分有限）的函数 $g(\mathbf{x})$ 都有下式成立：

$$\int_{-\infty}^{+\infty} K(\mathbf{x}, \mathbf{y}) g(\mathbf{x}) g(\mathbf{y}) d\mathbf{x} d\mathbf{y} \geq 0$$

$\forall g(\mathbf{x}) \text{ that satisfies : } \int_{-\infty}^{+\infty} |g(\mathbf{x})|^2 d\mathbf{x} \text{ is finite}$

下面给出一些具体的核函数：

- **多项式核函数**

$$K(\mathbf{x}, \mathbf{y}) = (\gamma \mathbf{x}^T \cdot \mathbf{y} + r)^p$$

- **RBF 核函数（径向基核函数、高斯核函数）**

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

特殊地，由于高斯核函数十分重要，因此这里对高斯核函数的映射函数 Φ 进行推导为：

由高斯核函数的表示形式：

$$K(\mathbf{x}, \mathbf{y}) = \exp\left[-\frac{\|\mathbf{x} - \mathbf{y}\|^2}{2\sigma^2}\right] = \exp\left[-\frac{\mathbf{x}^T \mathbf{x}}{2\sigma^2}\right] \cdot \exp\left[-\frac{\mathbf{y}^T \mathbf{y}}{2\sigma^2}\right] \cdot \exp\left[\frac{\mathbf{x}^T \mathbf{y}}{2\sigma^2}\right]$$

假设 \mathbf{x} 和 \mathbf{y} 的第 i 维度分量分别是 x_i 和 y_i ，那么：

$$K(\mathbf{x}, \mathbf{y}) = \prod_{i=1}^d \exp\left[-\frac{x_i^2}{2\sigma^2}\right] \cdot \prod_{i=1}^d \exp\left[-\frac{y_i^2}{2\sigma^2}\right] \cdot \prod_{i=1}^d \exp\left[-\frac{x_i y_i}{2\sigma^2}\right]$$

$$K(\mathbf{x}, \mathbf{y}) = \prod_{i=1}^d \exp\left[-\frac{x_i^2}{2\sigma^2}\right] \cdot \exp\left[-\frac{y_i^2}{2\sigma^2}\right] \cdot \exp\left[-\frac{x_i y_i}{2\sigma^2}\right]$$

我们假设上式 $K(\mathbf{x}, \mathbf{y})$ 之中的第 i 个乘子为 $K_i(\mathbf{x}, \mathbf{y})$ ，于是：

$$K_i(\mathbf{x}, \mathbf{y}) = \exp\left[-\frac{x_i^2}{2\sigma^2}\right] \cdot \exp\left[-\frac{y_i^2}{2\sigma^2}\right] \cdot \exp\left[-\frac{x_i y_i}{2\sigma^2}\right]$$

$$K_i(\mathbf{x}, \mathbf{y}) = \exp\left[-\frac{x_i^2}{2\sigma^2}\right] \cdot \exp\left[-\frac{y_i^2}{2\sigma^2}\right] \cdot \sum_{j=0}^{+\infty} \frac{1}{j!} \left(\frac{x_i y_i}{\sigma^2}\right)^j$$

$$K_i(\mathbf{x}, \mathbf{y}) = \sum_{j=0}^{+\infty} \left[\left(\sqrt{\frac{1}{j! \sigma^{2j}}} e^{-\frac{x_i^2}{2\sigma^2}} x_i^j \right) \cdot \left(\sqrt{\frac{1}{j! \sigma^{2j}}} e^{-\frac{y_i^2}{2\sigma^2}} y_i^j \right) \right]$$

于是对于每一个维度的映射 Φ_i 就是：

$$\Phi_i(x_i) = \sqrt{\frac{1}{j! \sigma^{2j}}} e^{-\frac{x_i^2}{2\sigma^2}} x_i^j, \forall \text{integer } j \in [0, +\infty)$$

这只是一个维度上的映射，我们不难发现单单一个维度就能映射出无穷维……如果加上所有维度，那应该是不可数无穷维的空间映射了，RBF 函数是很强的。至于每一个维度上的映射（就是如何从 K_i 过渡到 $K = \prod_{i=1}^d K_i$ ），确实不会推，等学过泛函和张量回来填一下吧。

- **Sigmoid 核函数**

$$K(\mathbf{x}, \mathbf{y}) = \tanh(\gamma \mathbf{x}^T \mathbf{y} - r)$$

5.5 VC 维、经验风险最小化准则与 SVM 的可解释性

VC 维是一个很有趣的东西。本质上，模式识别分类问题就是一个函数族内的优化问题：给定一个函数族（理解为分类器），如何使用训练数据，选出最好的那一个函数（理解为选出函数族中最好的函数对应的参数，也就是调参）。那么我们总会首先想到：你给我的这一族函数所具备的能力的上限在哪里？这个上限就可以用 VC 维来描述。

- **VC 维**: 一个函数族的 VC 维是能够被这个函数族“打散”(shatter)的最大数据集大小。

换句话说：假设一族函数 \mathcal{H} 的 VC 维是 N ，那么意味着任意给定 N 个样本，随便给它们分配标签，都能从 \mathcal{H} 之中找到函数来把这 N 个样本完美分类。

对于一个二维空间内的线性分类器，我们很容易知道其 VC 维是 3，因为线性分类器的上限是异或问题。但是对于指数函数族或者正弦函数族，其 VC 维是无穷，因为任意函数都可以使用指数函数或正弦函数在无穷维级数意义上完美拟合。对于 N 维空间内的 SVM，满足 $\|\mathbf{w}\| \leq A$ 条件的标准超平面族的 VC 维满足下面的界：

$$VC_{SVM} \leq \min \left\{ N + 1, [RA]^2 + 1 \right\}$$

R 是包含所有样本的最小超球的半径。

我们讲了这么长时间 SVM，只说了他的原则是最大化分类间隔，但是为什么最大化分类间隔就是最好的呢？原因其实就在于 VC 维。这里我们引入经验风险最小化准则 (Empirical Risk Minimization, ERM)。

- **ERM**: 假设样本的分布函数是 $F(\mathbf{x}, y)$ ，需要学习的函数族是 Q ，针对样本集 (\mathbf{x}, y) 和函数参数 α ，得到的函数族之中的具体函数是 $Q((\mathbf{x}, y), \alpha)$ 。那么风险泛函是 $R_{emp}(\alpha) = \int Q((\mathbf{x}, y), \alpha) dF(\mathbf{x}, y)$ 。

对于实际的风险，关于 ERM 存在上界：

$$R(\alpha) \leq R_{emp}(\alpha) + \Phi \left(\frac{h}{l} \right)$$

上式之中 h 是函数族的 VC 维， l 是样本量。针对现实分类问题，我们的目标都是优化实际的风险 $R(\alpha)$ 。

那么针对 SVM 问题，为了优化真实风险，我们需要使得风险上界更低。为

了使得上界更低，需要使得 $\Phi\left(\frac{h}{l}\right)$ 降低；由于 Φ 是单调增函数，因此需要 $\frac{h}{l}$ 更低；在样本量不变的情况下要求 h 更低。SVM 的分类间隔是 $\frac{1}{\|\mathbf{w}\|^2} = \frac{1}{A^2}$ ，于是分类间隔越大，意味着 SVM 的 VC 维越低，从而使得风险更小。

这就是 SVM 的统计学解释。当然写得很粗糙、不严谨、单纯是讲概念性的理解。真的希望认真推一边的话，非常推荐 Vapnik 的几本书：*Statistical Learning Theory* 和 *The Nature of Statistical Learning Theory*。我也在研究，很有趣！

5.6 支持向量机的正则化

本质上上面讲述的线性不可分支持向量机就是一种正则化。我们给出支持向量机的三种正则化的优化目标：

- General Loss:

$$\hat{\mathbf{w}} = \arg \min \mathbf{w} \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^N L(\mathbf{w}, \mathbf{x}_i, y_i)$$

- Hinge Loss:

$$\hat{\mathbf{w}} = \arg \min \mathbf{w} \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^N \max\{0, \xi_i\}$$

- Differentiable Loss:

$$\hat{\mathbf{w}} = \arg \min \mathbf{w} \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^N \max\{0, \xi_i\}^2$$

6 近邻法

6.1 K 近邻法

6.1.1 算法

- 保存所有输入的训练样本
- 对于每一个输入的测试样本，遍历所有训练样本，并求取这个测试样本到每一个训练样本的距离（这个距离可以是欧氏距离、马氏距离、切比雪夫距离、闵氏距离等等，本质上这个距离反映的是样本在空间中分布的流型，后面再说）

- 对于上一步骤求出的所有距离进行排序，求出与测试样本距离最近的前 K 个训练样本
- 针对上一步骤选出的 K 个训练样本，统计每个类别的个数，进行投票
- 把测试样本标记为票数最多的类别

K 近邻算法准确性相对高，对于异常值和噪声容忍度也较高，但是计算量很大、对于内存需求极高。

不过我认为，K 近邻最大的作用并不在于分类，而是在于为后面的高维流型降维提供思路，因为在非线性流型上的每一个样本都可以使用其周围的样本近似地线性表示，也就因此可以把高维流型“展平”。后面讲数据降维的时候再具体讨论吧。

6.1.2 以最近邻为例的错误率推导

对于每一个测试集样本是否错误，是由其最近邻决定的：如果其最近邻的类别是 θ' ，那么推测出这个样本的类别应该是 $\hat{\theta} = \theta'$ ，那么在 $\theta = \hat{\theta}$ 时分类正确，否则错误。

这就是错误率最为普遍的描述，因此写成概率形式就是：

$$P(e|\mathbf{x}, \mathbf{x}') = 1 - P(\theta = \theta'|\mathbf{x}, \mathbf{x}') = 1 - \sum_{i=1}^c P(\theta = \theta' = i|\mathbf{x}, \mathbf{x}')$$

由于本质上 \mathbf{x} 测试样本与 \mathbf{x}' 训练样本是无关的，因此上式可以进一步写成：

$$P(e|\mathbf{x}, \mathbf{x}') = 1 - \sum_{i=1}^c P(\theta = i|\mathbf{x})P(\theta' = i|\mathbf{x}')$$

上式之中 θ 是测试样本 \mathbf{x} 的真实类别标签； θ' 是训练样本 \mathbf{x}' 的真实类别标签。在这个过程中，测试样本找到的训练样本是 \mathbf{x}' ，或者说，在这种情形下 \mathbf{x}' 是 \mathbf{x} 的最近邻。最近邻本质上在确定一个训练集之后，对于某一个测试样本就已经是确定的了。但是我们分析错误率并不是仅仅分析这一种训练集下的错误率，而是所有可能出现的训练集下的错误率期望值。因此我们需要对上面的式子针对 $p(\mathbf{x}'|\mathbf{x})$ 求取期望。也就是：

$$P(e|\mathbf{x}) = \int_{\mathbf{x}'} P(e|\mathbf{x}, \mathbf{x}')p(\mathbf{x}'|\mathbf{x})d\mathbf{x}'$$

上式之中 $p(\mathbf{x}'|\mathbf{x})$ 针对于不同的训练集是变化的，我们并不知道。但是我们假设极限的情形，也就是 $N \rightarrow +\infty$ 的情形， N 是训练集样本量。在这种情

形下，我们可以联想整个空间内充斥着样本点，于是对于每一个测试集样本点 \mathbf{x} ，其对应的训练集最近邻趋近于 \mathbf{x} ，于是有如下形式：

$$\lim_{N \rightarrow \infty} p(\mathbf{x}'|\mathbf{x}) = \delta(\mathbf{x}' - \mathbf{x})$$

同时当样本量趋于无穷时， $P(\theta = i|\mathbf{x}) = P(\theta' = i|\mathbf{x}')$ 。

那么我们可以进一步推导：

$$\begin{aligned}\lim_{N \rightarrow \infty} P(e|\mathbf{x}) &= \lim_{N \rightarrow \infty} \int_{\mathbf{x}'} P(e|\mathbf{x}, \mathbf{x}') p(\mathbf{x}'|\mathbf{x}) d\mathbf{x}' \\ \lim_{N \rightarrow \infty} P(e|\mathbf{x}) &= \int_{\mathbf{x}'} \left[1 - \sum_{i=1}^c P^2(\theta = i|\mathbf{x}) \right] \delta(\mathbf{x}' - \mathbf{x}) d\mathbf{x}' \\ \lim_{N \rightarrow \infty} P(e|\mathbf{x}) &= 1 - \sum_{i=1}^c P^2(\theta = i|\mathbf{x})\end{aligned}$$

为了对比最近邻和贝叶斯，我们下面给出贝叶斯的错误率 $P^*(e|\mathbf{x})$ ：

$$P^*(e|\mathbf{x}) = 1 - \max_i \{P(\theta = i|\mathbf{x})\}$$

贝叶斯错误率是上面的形式的主要原因是朴素贝叶斯分类器就是最小化错误率的分类器，朴素贝叶斯分类器是知道每一种类别的分布函数的，会把每一段区间内概率密度最大的类别认为是正确类别，因此错误率就是 1 减去这个类别的概率。

我们观察 $P(e|\mathbf{x}) = 1 - \sum_{i=1}^c P^2(\theta = i|\mathbf{x})$ 和 $P^*(e|\mathbf{x}) = 1 - \max_i \{P(\theta = i|\mathbf{x})\}$ 两个式子，根据归纳法，在满足 $\sum_{i=1}^c P(\theta = i|\mathbf{x}) = 1$ 的情形下，下式是恒成立的：

$$\max_i \{P(\theta = i|\mathbf{x})\} \geq \sum_{i=1}^c P^2(\theta = i|\mathbf{x})$$

因此：

$$\int_{\mathbf{x}} \left[1 - \sum_{i=1}^c P^2(\theta = i|\mathbf{x}) \right] p(\mathbf{x}) d\mathbf{x} \geq \int_{\mathbf{x}} \left[1 - \max_i \{P(\theta = i|\mathbf{x})\} \right] p(\mathbf{x}) d\mathbf{x}$$

$$P(e) \geq P^*(e)$$

下面求最近邻法的错误率上界：

$$\sum_{i=1}^c P^2(\theta = i|\mathbf{x}) = P^2(\theta = m|\mathbf{x}) + \sum_{j=1, j \neq m}^c P^2(\theta = j|\mathbf{x})$$

$$m = \arg \max_i P(\theta = i | \mathbf{x})$$

$$\sum_{i=1}^c P^2(\theta = i | \mathbf{x}) = (1 - P^*(e | \mathbf{x}))^2 + \sum_{j=1, j \neq m}^c P^2(\theta = j | \mathbf{x})$$

在固定最大类别概率 $1 - P^*(e | \mathbf{x})$ 时, 剩余的概率加起来的数值存在最小值, 在其余各个类别概率相等时取得, 其余各个类别概率分别是:

$$P(\theta = j | \mathbf{x}) = \frac{P^*(e | \mathbf{x})}{c-1}, \forall j \neq m$$

最小值是:

$$\sum_{j=1, j \neq m}^c P^2(\theta = j | \mathbf{x}) \geq \frac{(P^*(e | \mathbf{x}))^2}{c-1}$$

$$\sum_{i=1}^c P^2(\theta = i | \mathbf{x}) \geq (1 - P^*(e | \mathbf{x}))^2 + \frac{(P^*(e | \mathbf{x}))^2}{c-1}$$

于是化简得到:

$$1 - \sum_{i=1}^c P^2(\theta = i | \mathbf{x}) \leq 2P^*(e | \mathbf{x}) - \frac{c}{c-1} P^{*2}(e | \mathbf{x})$$

于是积分可以得到:

$$P(e) \leq P^*(e) \left(2 - \frac{c}{c-1} P^*(e) \right)$$

综合上面的所有推导:

$$P^*(e) \leq P(e) \leq P^*(e) \left(2 - \frac{c}{c-1} P^*(e) \right)$$

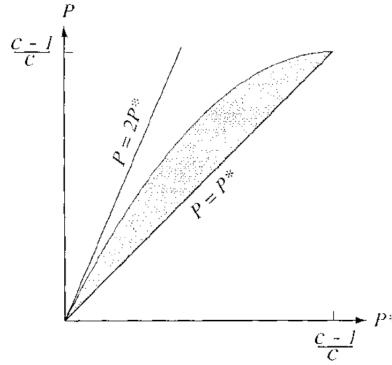


Fig 2. 最近邻错误率

6.1.3 K 近邻的错误率

在 K 个近邻之中，看似是一个多分类问题，其实是一个二分类：只有正确与否。从 K 个近邻的统计结果之中，如果存在多于 $\frac{k}{2}$ 个点被分对，那么结果就是正确的。为了表述这个说法，可以引入下面的数学形式：

$$P_{N \rightarrow \infty}^{KNN}(e|\mathbf{x}) = P(\theta = i|\mathbf{x}) \sum_{j=0}^{\frac{k-1}{2}} C_k^j P^j(\theta = i|\mathbf{x}) [1 - P(\theta = i|\mathbf{x})]^{k-j} \\ + [1 - P(\theta = i|\mathbf{x})] \sum_{j=\frac{k-1}{2}}^k C_k^j P^j(\theta = i|\mathbf{x}) [1 - P(\theta = i|\mathbf{x})]^{k-j}$$

上式之中前一项是判决并不是第 i 类但是实际上是第 i 类的概率；后一项是判决为第 i 类，但是实际上不是第 i 类的概率。

在这种情况下贝叶斯错误率为：

$$P^*(e|\mathbf{x}) = \min [P(\theta = i|\mathbf{x}), 1 - P(\theta = i|\mathbf{x})]$$

那么进一步转写上式为：

$$P_{N \rightarrow \infty}^{KNN}(e|\mathbf{x}) = P^*(e|\mathbf{x}) \sum_{j=0}^{\frac{k-1}{2}} C_k^j P^{*j}(e|\mathbf{x}) [1 - P^*(e|\mathbf{x})]^{k-j} \\ + [1 - P^*(e|\mathbf{x})] \sum_{j=\frac{k-1}{2}}^k C_k^j P^{*j}(e|\mathbf{x}) [1 - P^*(e|\mathbf{x})]^{k-j}$$

6.2 压缩近邻法

提出压缩近邻法实际上是为了降低 K 近邻法的庞大存储量而提出的，基本思路非常简单，就是把距离分类面近的训练样本保存下来，距离分类面远的样本扔掉。有点点像 SVM 找支撑集，但是本质是不同的，支撑集是通过数学方式确定的，但是压缩近邻法是用贪婪的思想随机选取的，并不相同。

- 构建两个集合，Grabbag 和 Store 集合；并将 Grabbag 集合初始化为训练集、Store 初始化为空集
- 从 Grabbag 取出一个样本放入 Store
- 用 Store 中全部样本以 K 近邻方法测试 Grabbag 中的全部样本，如果分错，则将分错的样本放入 Store
- 重复上述过程，直到 Grabbag 中没有样本转入 Store

6.3 距离度量

- 闵氏距离:

$$D_{Min}(\mathbf{x}, \mathbf{y}) = \left[\sum_{j=1}^d |x_j - y_j|^s \right]^{\frac{1}{s}}$$

很简单没啥说的，有一点值得提，就是闵可夫斯基不等式。在泛函 \mathcal{L}_p 空间内如果闵氏不等式成立，说明这个空间是一个赋范线性空间。

- 切比雪夫距离:

$$D_C(\mathbf{x}, \mathbf{y}) = \max_j |x_j - y_j|$$

也很简单，是闵氏不等式在 $s \rightarrow \infty$ 的极限。

- 马氏距离:

$$D_M(\mathbf{x}, \mathbf{y}) = (\mathbf{x} - \mathbf{y})^H \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \mathbf{y})$$

本质上是各个维度放缩和旋转之后的欧氏距离，因为直接对 $\boldsymbol{\Sigma}^{-1}$ 进行 Cholesky 分解之后得到：

$$D_M(\mathbf{x}, \mathbf{y}) = (\mathbf{x} - \mathbf{y})^H \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \mathbf{y}) = (\mathbf{L}\mathbf{z})^H \cdot \mathbf{L}\mathbf{z}$$

- KL 距离:

$$D_{KL}(P, Q) = \sum_{x \in X} P(x) \log \left[\frac{P(x)}{Q(x)} \right]$$

用来度量不同分布之间的相似程度熵的距离，放在这里凑个数。

- 切线距离:

$$D_{tg}(\mathbf{x}, \mathbf{y}) = \min_a \|(\mathbf{x} + a\mathbf{T}) - \mathbf{y}\|$$

对于一个样本 \mathbf{x} ，可构造矩阵 \mathbf{T} ，代表 $\mathbf{x} \in \mathbf{X}$ 的流型在 \mathbf{x} 处的切线。这样的距离对于手写数字识别有很好的效果，其实我觉得就是把高维流型展平了。

7 有监督数据降维（特征选择）

7.1 特征选择问题描述

特征选择也可以称为数据降维。我们仿佛又一次突然从 SVM 和 KNN 分类器设计跳跃到了特征提取与选择。特征提取与选择这个问题是为了降低特

征维度，避免维数灾难而引入的。维数灾难这个事情我们在前面的生成式模型：概率密度函数估计部分提到过了。

事实上，如果是学过 Fisher 准则和 LDA 的朋友们可能觉得是不是 Fisher 和 PCA 很像，都是在找某个协方差矩阵的最大特征值做投影（这里说某个，是因为两种方法找的矩阵不是同样的矩阵，如下式所示）。但是本质上不一样。最大的不同在于 Fisher 是有监督的特征提取，而 PCA 是无监督的特征提取。

$$\begin{aligned} \text{Cov}_{PCA} &= \sum_{i=1}^N (\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^H \\ \text{Cov}_{Fisher} &= \mathbf{S}_w^{-1} \mathbf{S}_b \end{aligned}$$

这里还是讲一句，别的总结之中有很多人把 Fisher 算法认为是一种分类器，这也是成立的，因为针对 M 类分类的 Fisher 算法将样本投影到 M-1 维空间内，就是在 M-1 维空间内找到使得类间距离最大、类内距离最小的 M-1 维最优超平面作为分类面。

最后澄清一下 Fisher 和 LDA 这两个名词的意义。Fisher 准则是一种准则，用于衡量类内距离最小化和类间距离最大化的程度；LDA 是线性判别分析，是指基于 Fisher 准则的线性分类器。这只是我的理解，你也可以认为二者是一样的，其实确实，只要思想是相同的，无所谓叫什么。

7.2 二分类的 Fisher 准则

Fisher 准则的基本优化目标是最大化类间距离同时最小化类内距离。衡量类间和类内距离的方式是采用类间和类内散度矩阵。首先是两类的情况，目标就具体地变为：把 d 维空间内的样本投影到一条直线上，使得在这条直线上能够达到类内距离最小、类间距离最大。

$$y_i = \mathbf{w}^T \mathbf{x}_i, \forall i \in \{1, 2, \dots, N\}$$

在这条直线上，两类的中心分别为：

$$\begin{aligned} \tilde{\mathbf{m}}_1 &= \frac{1}{N_1} \sum_{y \in \mathcal{Y}_1} y = \mathbf{w}^T \left(\frac{1}{N_1} \sum_{y \in \mathcal{Y}_1} \mathbf{x} \right) \\ \tilde{\mathbf{m}}_2 &= \frac{1}{N_2} \sum_{y \in \mathcal{Y}_2} y = \mathbf{w}^T \left(\frac{1}{N_2} \sum_{y \in \mathcal{Y}_2} \mathbf{x} \right) \end{aligned}$$

在这条直线上，两类的类内方差分别为：

$$\tilde{S}_1^2 = \sum_{y \in \mathcal{Y}_1} (y - \tilde{m}_1)^2 = \mathbf{w}^T \left[\sum_{y \in \mathcal{Y}_1} (\mathbf{x} - \mathbf{m}_1)(\mathbf{x} - \mathbf{m}_1)^H \right] \mathbf{w} = \mathbf{w}^T \mathbf{S}_1 \mathbf{w}$$

$$\tilde{S}_2^2 = \sum_{y \in \mathcal{Y}_2} (y - \tilde{m}_2)^2 = \mathbf{w}^T \left[\sum_{y \in \mathcal{Y}_2} (\mathbf{x} - \mathbf{m}_2)(\mathbf{x} - \mathbf{m}_2)^H \right] \mathbf{w} = \mathbf{w}^T \mathbf{S}_2 \mathbf{w}$$

于是类内距离可以定义为最大化 J_F ：

$$\tilde{S}_w^2 = \tilde{S}_1^2 + \tilde{S}_2^2 = \mathbf{w}^T (\mathbf{S}_1 + \mathbf{S}_2) \mathbf{w} = \mathbf{w}^T \mathbf{S}_w \mathbf{w}$$

类间距离定义为：

$$\tilde{S}_b^2 = (\tilde{m}_1 - \tilde{m}_2)^2 = \mathbf{w}^T (\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^H \mathbf{w} = \mathbf{w}^T \mathbf{S}_b \mathbf{w}$$

从而优化目标定义为：

$$J_F(\mathbf{w}) = \frac{\tilde{S}_b^2}{\tilde{S}_w^2} = \frac{(\tilde{m}_1 - \tilde{m}_2)^2}{\tilde{S}_1^2 + \tilde{S}_2^2} = \frac{\mathbf{w}^T \mathbf{S}_b \mathbf{w}}{\mathbf{w}^T \mathbf{S}_w \mathbf{w}}$$

这样的形式其实就是广义瑞利商，按照广义瑞利商的相关定理可以直接知道当 \mathbf{w} 取为 $\mathbf{S}_w^{-1} \mathbf{S}_b$ 的最大特征值对应的特征向量时 J_F 取得最大值，即类内距离最小同时类间距离最大。

为了使得证明完整，我们推导一下广义瑞利商的最大值和最小值，首先给出广义瑞利商最大值和最小值的定义：

- 对于如下形式的广义瑞利商：

$$R(\mathbf{A}, \mathbf{B}, \mathbf{x}) = \frac{\mathbf{x}^H \mathbf{A} \mathbf{x}}{\mathbf{x}^H \mathbf{B} \mathbf{x}}$$

其最大值为 $\mathbf{B}^{-1} \mathbf{A}$ 矩阵的最大特征值，最小值为该矩阵的最小特征值，分别在对应于 \mathbf{x} 是最大、最小特征值对应的特征向量处取得。

下面对于上述定理进行证明：

$$\begin{aligned} \tilde{\mathbf{x}} &= \mathbf{B}^{\frac{1}{2}} \mathbf{x} \\ R(\mathbf{A}, \mathbf{B}, \mathbf{x}) &= \frac{\tilde{\mathbf{x}}^H (\mathbf{B}^{-\frac{1}{2}} \mathbf{A} \mathbf{B}^{-\frac{1}{2}}) \tilde{\mathbf{x}}}{\tilde{\mathbf{x}}^H \tilde{\mathbf{x}}} \end{aligned}$$

由于对于普通的瑞利商 $R(\mathbf{A}, \mathbf{x}) = \frac{\mathbf{x}^H \mathbf{A} \mathbf{x}}{\mathbf{x}^H \mathbf{x}} \in [\lambda_{min}, \lambda_{max}]$, 其中 λ 是矩阵 \mathbf{A} 的特征值, 且最大/最小值在 \mathbf{x} 是 \mathbf{A} 最大/最小特征值对应的特征向量处取等。于是带入上面的式子证毕:

$$\mathbf{B}^{-\frac{1}{2}} \mathbf{A} \mathbf{B}^{-\frac{1}{2}} \tilde{\mathbf{x}} = \lambda \tilde{\mathbf{x}}$$

$$\mathbf{B}^{-1} \mathbf{A} \mathbf{x} = \lambda \mathbf{x}$$

推导到了这里, 我们给出一个 Fisher 准则的结论。对于二分类的 Fisher 准则, 有如下式子成立:

$$\mathbf{S}_w^{-1} \mathbf{S}_b \hat{\mathbf{w}} = \lambda_{max} \hat{\mathbf{w}}$$

$$\mathbf{S}_b \hat{\mathbf{w}} = (\mathbf{m}_1 - \mathbf{m}_2) [(\mathbf{m}_1 - \mathbf{m}_2)^H \mathbf{w}] = k(\mathbf{m}_1 - \mathbf{m}_2)$$

于是:

$$\hat{\mathbf{w}} = k \mathbf{S}_w^{-1} (\mathbf{m}_1 - \mathbf{m}_2)$$

上式之中 k 仅仅是比例系数, 不一定互相相等。

接着我们对于上述的结果进行一个小讨论:

- **$\hat{\mathbf{w}}$ 一定只有一个吗?** 对, 必然只有一个, 因为它是 $\mathbf{S}_w^{-1} \mathbf{S}_b$ 矩阵的特征向量, $\mathbf{S}_b = (\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^H$ 是一个秩为 1 的矩阵, 因此 $\mathbf{S}_w^{-1} \mathbf{S}_b$ 秩至多是 1, 因此也就只有一个非 0 特征值对应的特征向量。
- **类内散度矩阵一定可逆吗?** 不一定可逆, 但是在样本量充足时往往都是可逆的。不可逆的时候取广义逆即可。

7.3 多分类的 Fisher 准则

还是按照上面一个小节给出对应的参量:

$$\begin{aligned} \tilde{\mathbf{m}}_k &= \frac{1}{N_k} \sum_{y \in \mathcal{Y}_k} \mathbf{y} = \mathbf{w}^T \left(\frac{1}{N_k} \sum_{y \in \mathcal{Y}_k} \mathbf{x} \right) = \mathbf{w}^H \mathbf{m}_k \\ \mathbf{m} &= \frac{1}{\sum_{k=1}^K N_k} \sum_{i=1}^N \mathbf{x}_i = \frac{1}{N} \sum_{k=1}^K N_k \mathbf{m}_k \\ \tilde{\mathbf{m}} &= \mathbf{w}^H \mathbf{m} \\ \tilde{\mathbf{S}}_b &= \sum_{k=1}^K N_k \|\tilde{\mathbf{m}}_k - \tilde{\mathbf{m}}\|^2 = \mathbf{w}^H \left[\sum_{k=1}^K N_k (\mathbf{m}_k - \mathbf{m})(\mathbf{m}_k - \mathbf{m})^H \right] \mathbf{w} = \mathbf{w}^H \mathbf{S}_b \mathbf{w} \end{aligned}$$

$$\tilde{\mathbf{S}}_{\mathbf{w}} = \sum_{k=1}^K \sum_{y \in \mathcal{Y}_k} \|\mathbf{y} - \tilde{\mathbf{m}}_k\|^2 = \mathbf{w}^H \left[\sum_{k=1}^K \sum_{y \in \mathcal{Y}_k} (\mathbf{x} - \mathbf{m}_k)(\mathbf{x} - \mathbf{m}_k)^H \right] \mathbf{w} = \mathbf{w}^H \mathbf{S}_{\mathbf{w}} \mathbf{w}$$

下面是设计目标优化函数，有很多种设计方法，介绍两种常用的，分别是：

$$J_F^{(1)}(\mathbf{w}) = \frac{\prod_{diag} \tilde{\mathbf{S}}_{\mathbf{b}}}{\prod_{diag} \tilde{\mathbf{S}}_{\mathbf{w}}}$$

$$J_F^{(2)}(\mathbf{w}) = \frac{tr(\tilde{\mathbf{S}}_{\mathbf{b}})}{tr(\tilde{\mathbf{S}}_{\mathbf{w}})}$$

在上面的表述中值得注意的是：此时 $\mathbf{w} \in \mathcal{R}^{d \times (K-1)}$ 。我们主要仔细推导 $J_F^{(1)}(\mathbf{w})$ ，因为 $\tilde{\mathbf{S}}_{\mathbf{b}}$ 的对角线元素应分别为： $\mathbf{w}_i^H \mathbf{S}_{\mathbf{b}} \mathbf{w}_i$ ，其中 \mathbf{w}_i 是 \mathbf{w} 的第 i 列；同样的 $\tilde{\mathbf{S}}_{\mathbf{w}}$ 的对角线元素应分别为： $\mathbf{w}_i^H \mathbf{S}_{\mathbf{w}} \mathbf{w}_i$ 。于是上面的 $J_F^{(1)}(\mathbf{w})$ 可以写为：

$$J_F^{(1)}(\mathbf{w}) = \prod_{i=1}^{K-1} \frac{\mathbf{w}_i^H \mathbf{S}_{\mathbf{b}} \mathbf{w}_i}{\mathbf{w}_i^H \mathbf{S}_{\mathbf{w}} \mathbf{w}_i}$$

对上面那个式子求最大值也很简单，就是让 \mathbf{w}_i 分别是 $\mathbf{S}_{\mathbf{w}}^{-1} \mathbf{S}_{\mathbf{b}}$ 的前 $K-1$ 大个特征向量就行了。针对 $J_F^{(2)}(\mathbf{w})$ 也可以得到相同的结论，很显然，就不费周章了。

7.4 基于 Fisher 准则的分类器设计

既然 Fisher 也可以被理解为是一种分类器，我们下面给出基于 Fisher 准则可以如何设计分类器。给出如下三种常用的设计方法：

- 类别均匀条件下的最大似然准则： $y^{(1)} = \frac{\tilde{m}_1 + \tilde{m}_2}{2}$
- 贝叶斯准则： $y^{(2)} = \frac{\tilde{m}_1 + \tilde{m}_2}{2} + \frac{1}{N_1 + N_2 - 2} \ln \left[\frac{P(w_1)}{P(w_2)} \right]$
- 类别不均匀条件下的最大似然准则： $y^{(3)} = \frac{N_2 \tilde{m}_1 + N_1 \tilde{m}_2}{N_1 + N_2}$

7.5 RELIEF 算法

直接给出算法，看完算法再讨论：

- 首先设定各个维度的初始化权重 $\mathbf{w} = [w_1 \quad w_2 \quad \cdots \quad w_d]^T = \mathbf{0}$
- 从训练集中随机选出一个样本 \mathbf{x}

- 计算训练集之中距离样本 \mathbf{x} 最近的同类样本 \mathbf{h} 和不同类样本 \mathbf{m}
- 分别更新每个维度的权重为 $w_j = w_j - \frac{1}{n} \text{diff}(j, \mathbf{x}, \mathbf{h}) + \frac{1}{n} \text{diff}(j, \mathbf{x}, \mathbf{m})$, 其中 n 是总共重复选取样本的次数
- 返回第二步, 一直迭代 n 轮
- 返回权值向量 \mathbf{w} , 输出权重最大的前 k 个特征

上面算法过程中有一个 diff 函数, 对于连续变量:

$$\text{diff}(j, \mathbf{x}, \mathbf{h}) = \frac{|\mathbf{x}_j - \mathbf{h}_j|}{\mathbf{x}_{jmax} - \mathbf{x}_{jmin}}$$

上式之中 \mathbf{x}_{jmax} 是所有样本 \mathbf{x} 中第 j 维上的最大值。

对于离散变量:

$$\text{diff}(j, \mathbf{x}, \mathbf{h}) = \begin{cases} 0 & \mathbf{x}_j = \mathbf{h}_j \\ 1 & \text{otherwise} \end{cases}$$

下面对于 RELIEF 算法进行简单的讨论, 其实 RELIEF 算法的本质就是: 如果一个维度上有很多的样本都是不同的, 那么认为这个维度上信息量很大, 需要保留; 而如果一个维度上很多样本都是相同的, 那么认为这个维度差距不大, 扔掉这个维度。

7.6 其他算法

除了上面介绍的 Fisher 和 RELIEF 这两种有监督特征选择算法, 还有别的很多算法:

- **最优搜索法:** 分支定界法
- **次优搜索法:** 顺序前进法、顺序后退法、增 1 减 r 法、模拟退火法、Tabu 搜索法、遗传算法等

上述的这些其他算法中很多也十分有趣, 但是限于篇幅暂且不表。

8 神经网络

8.1 反向传播 BP 算法

8.1.1 神经网络的常见函数和操作

- Softmax 层:

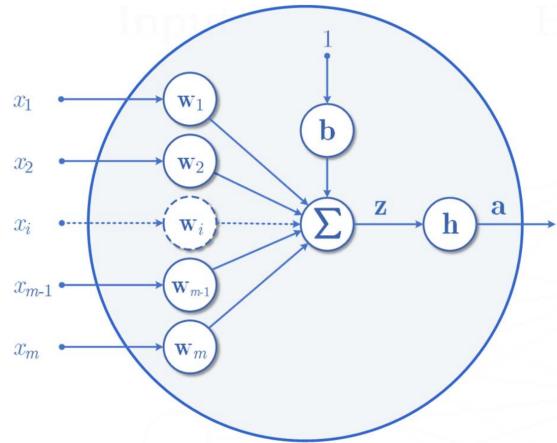


Fig 3. Softmax 单元

$$\mathbf{W} = [\mathbf{w}_1 \quad \mathbf{w}_2 \quad \cdots \quad \mathbf{w}_m] \in \mathcal{R}^{l \times m}$$

$$\mathbf{x} = [x_1 \quad x_2 \quad \cdots \quad x_m]^T \in \mathcal{R}^{m \times 1}$$

$$\mathbf{b} = [b_1 \quad b_2 \quad \cdots \quad b_m]^T \in \mathcal{R}^{l \times 1}$$

$$\mathbf{a} = [a_1 \quad a_2 \quad \cdots \quad a_m]^T \in \mathcal{R}^{l \times 1}$$

$$z = \mathbf{W}\mathbf{x} + \mathbf{b}$$

$$\mathbf{a} = \frac{\exp(z)}{\|\exp(z)\|_1}$$

- Logistic 层:

$$z = \mathbf{w}^T \mathbf{x} + b$$

$$a = \frac{1}{1 + e^{-z}}$$

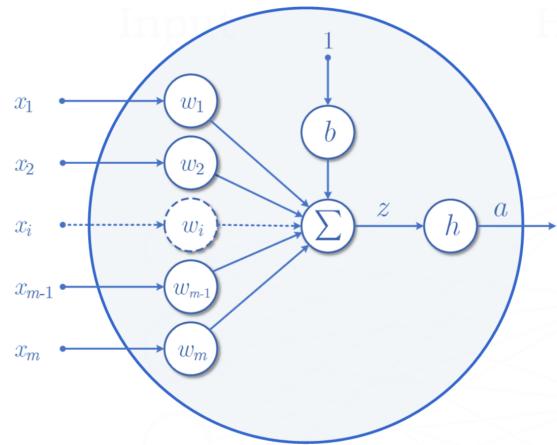


Fig 4. Logistic 单元

$$\frac{\partial a}{\partial b} = \frac{\partial a}{\partial z} \cdot \frac{\partial z}{\partial b} = a(1 - a)$$

$$\frac{\partial a}{\partial \mathbf{w}} = \frac{\partial a}{\partial z} \cdot \frac{\partial z}{\partial \mathbf{w}} = a(1 - a)\mathbf{x}^T$$

- 线性单元:

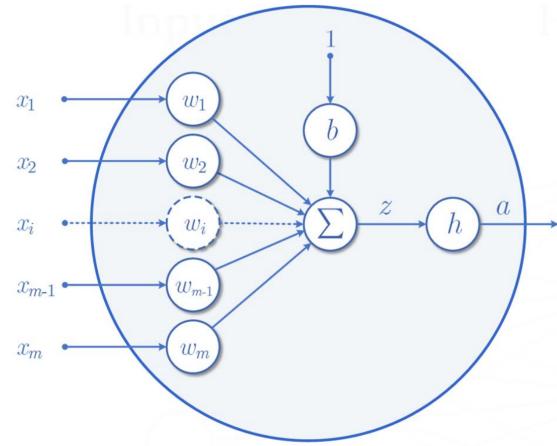


Fig 5. 线性单元

$$a = z = \mathbf{w}^T \mathbf{x} + b$$

$$\frac{\partial a}{\partial b} = 1$$

$$\frac{\partial a}{\partial \mathbf{w}} = \mathbf{x}^T$$

- ReLU 单元:

$$a = \max\{0, z\}$$

$$\begin{aligned} z &= \mathbf{w}^T \mathbf{x} + b \\ \frac{\partial a}{\partial z} &= \begin{cases} 1 & z \geq 0 \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

- tanh 单元:

$$\begin{aligned} a &= \frac{e^z - e^{-z}}{e^z + e^{-z}} \\ z &= \mathbf{w}^T \mathbf{x} + b \\ \frac{\partial a}{\partial z} &= 1 - a^2 \end{aligned}$$

8.1.2 向量求导链式法则

- 标量对向量求导的链式法则:

假设向量 \mathbf{x} 和 \mathbf{y} , 其维度分别为 m 和 n , 以及一个标量 z , 依赖关系为 $\mathbf{x} \rightarrow \mathbf{y} \rightarrow z$, 则有如下链式求导法则:

$$\begin{aligned} \frac{\partial z}{\partial \mathbf{x}} &= \left(\frac{\partial \mathbf{y}}{\partial \mathbf{x}} \right)^T \frac{\partial z}{\partial \mathbf{y}} \\ \frac{\partial z}{\partial \mathbf{x}} &\in \mathcal{R}^{m \times 1}; \quad \frac{\partial \mathbf{y}}{\partial \mathbf{x}} \in \mathcal{R}^{n \times m}; \quad \frac{\partial z}{\partial \mathbf{y}} \in \mathcal{R}^{n \times 1} \end{aligned}$$

- 向量对向量求导的链式法则:

假设向量 \mathbf{x} 和 \mathbf{y} 以及 \mathbf{z} , 其维度分别为 m 和 n 以及 p , 依赖关系为 $\mathbf{x} \rightarrow \mathbf{y} \rightarrow \mathbf{z}$, 则有如下链式求导法则:

$$\begin{aligned} \frac{\partial \mathbf{z}}{\partial \mathbf{x}} &= \frac{\partial \mathbf{z}}{\partial \mathbf{y}} \frac{\partial \mathbf{y}}{\partial \mathbf{x}} \\ \frac{\partial \mathbf{z}}{\partial \mathbf{x}} &\in \mathcal{R}^{p \times m}; \quad \frac{\partial \mathbf{z}}{\partial \mathbf{y}} \in \mathcal{R}^{p \times n}; \quad \frac{\partial \mathbf{y}}{\partial \mathbf{x}} \in \mathcal{R}^{n \times m} \end{aligned}$$

- 标量对多个向量求导的链式法则:

假设多个向量 $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$, 分别对应于维度 $\{p_1, p_2, \dots, p_n\}$ 以及一个标量 y , 依赖关系为 $\mathbf{x}_1 \rightarrow \mathbf{x}_2 \rightarrow \dots \rightarrow \mathbf{x}_n \rightarrow y$, 则有如下链式求导法则:

$$\begin{aligned} \frac{\partial y}{\partial \mathbf{x}_1} &= \left(\frac{\partial \mathbf{x}_n}{\partial \mathbf{x}_1} \right)^T \frac{\partial y}{\partial \mathbf{x}_n} \\ \frac{\partial \mathbf{x}_n}{\partial \mathbf{x}_1} &= \frac{\partial \mathbf{x}_n}{\partial \mathbf{x}_{n-1}} \frac{\partial \mathbf{x}_{n-1}}{\partial \mathbf{x}_{n-2}} \dots \frac{\partial \mathbf{x}_2}{\partial \mathbf{x}_1} \end{aligned}$$

- 标量对多个矩阵求导的链式法则:

假设矩阵 \mathbf{X} 和 \mathbf{Y} 以及标量 z , 那么:

$$\frac{\partial z}{\partial X_{ij}} = \sum_{k,l} \frac{\partial z}{\partial Y_{kl}} \frac{\partial Y_{kl}}{\partial X_{ij}} = \text{tr} \left[\left(\frac{\partial z}{\partial \mathbf{Y}} \right)^T \frac{\partial \mathbf{Y}}{\partial X_{ij}} \right]$$

- 机器学习常用的标量对矩阵的链式法则:

第一种情况:

$$z = f(\mathbf{Y}); \quad \mathbf{Y} = \mathbf{AX} + \mathbf{B}$$

$$\frac{\partial z}{\partial \mathbf{X}} = \mathbf{A}^T \frac{\partial z}{\partial \mathbf{Y}}$$

第二种情况:

$$z = f(\mathbf{y}); \quad \mathbf{y} = \mathbf{Ax} + \mathbf{b}$$

$$\frac{\partial z}{\partial \mathbf{x}} = \mathbf{A}^T \frac{\partial z}{\partial \mathbf{y}}$$

第三种情况:

$$z = f(\mathbf{Y}); \quad \mathbf{Y} = \mathbf{XA} + \mathbf{B}$$

$$\frac{\partial z}{\partial \mathbf{X}} = \frac{\partial z}{\partial \mathbf{Y}} \mathbf{A}^T$$

第四种情况:

$$z = f(\mathbf{y}); \quad \mathbf{y} = \mathbf{Xa} + \mathbf{b}$$

$$\frac{\partial z}{\partial \mathbf{X}} = \frac{\partial z}{\partial \mathbf{y}} \mathbf{a}^T$$

8.1.3 实际计算 BP 算法

好的既然我们已经知道了基本方法, 我们举一个例子实际计算一下。如下图所示是一个简单的神经网络, 计算其反向传播算法。

Backward Propagation Algorithm

We have a k -class classification problem. The input \mathbf{x} is a d -dimensional vector, and the corresponding label \mathbf{y} is a one-hot k -dimensional vector with one element being 1 and others being 0. We define the following neural network $\hat{\mathbf{y}} = f(\mathbf{x}; \mathbf{w}_1, \mathbf{b}_1, \mathbf{w}_2, \mathbf{b}_2)$:

$$\mathbf{h}_1 = \mathbf{w}_1^\top \cdot \mathbf{x} + \mathbf{b}_1, \quad \mathbf{h}_1 \in \mathbb{R}^s, \quad (1)$$

$$\mathbf{a}_1 = \text{ReLU}(\mathbf{h}_1), \quad \mathbf{a}_1 \in \mathbb{R}^s, \quad (2)$$

$$\mathbf{h}_2 = \text{Concat}(\mathbf{a}_1, \mathbf{x}), \quad \mathbf{h}_2 \in \mathbb{R}^{s+d}, \quad (3)$$

$$\mathbf{m} \sim \text{Bernoulli}(p), \quad \mathbf{m} \in \mathbb{R}^{s+d}, \quad (4)$$

$$\mathbf{a}_2 = \mathbf{h}_2 \odot \mathbf{m}, \quad \mathbf{a}_2 \in \mathbb{R}^{s+d}, \quad (5)$$

$$\mathbf{h}_3 = \mathbf{w}_2^\top \cdot \mathbf{a}_2 + \mathbf{b}_2, \quad \mathbf{h}_3 \in \mathbb{R}^k, \quad (6)$$

$$\hat{\mathbf{y}} = \text{Softmax}(\mathbf{h}_3), \quad \hat{\mathbf{y}} \in \mathbb{R}^k, \quad (7)$$

where $\text{ReLU}(\cdot)$ is element-wise ReLU activation function, $\text{Concat}(\mathbf{a}, \mathbf{b})$ means concatenating vector \mathbf{b} after vector \mathbf{a} to make a vector with larger dimensionality, \mathbf{m} is current dropout mask, \odot means element-wise multiplication (Specifically, equation(5) randomly zeroes some of the elements of the input tensor \mathbf{h}_2 with probability p using \mathbf{m} sampled from a Bernoulli distribution. For example, if $\mathbf{h}_2 = [a, b, c, d]$, and $\mathbf{m} = [e, f, g, h]$, then $\mathbf{a}_2 = [ae, bf, cg, dh]$ according to equation(5)), and $\text{Softmax}(\cdot)$ is softmax activation function. The dimensionalities of parameters and internal variables could be inferred from the neural network definition.

We use the following cross-entropy loss:

$$L = -\mathbf{y}^\top \log \hat{\mathbf{y}}. \quad (8)$$

Please use backward propagation algorithm to find the following derivatives:

$$\frac{\partial L}{\partial \hat{\mathbf{y}}}, \quad \frac{\partial L}{\partial \mathbf{h}_3}, \quad \frac{\partial L}{\partial \mathbf{w}_2}, \quad \frac{\partial L}{\partial \mathbf{b}_2}, \quad \frac{\partial L}{\partial \mathbf{a}_2}, \quad \frac{\partial L}{\partial \mathbf{h}_2}, \quad \frac{\partial L}{\partial \mathbf{a}_1}, \quad \frac{\partial L}{\partial \mathbf{h}_1}, \quad \frac{\partial L}{\partial \mathbf{w}_1}, \quad \frac{\partial L}{\partial \mathbf{b}_1}, \quad \frac{\partial L}{\partial \mathbf{x}}.$$

Please express your results in matrices and vectors instead of sum of individual elements.

Fig 6. 实际计算题面

- $\frac{\partial \mathcal{L}}{\partial \hat{\mathbf{y}}} :$

$$\frac{\partial \mathcal{L}}{\partial \hat{\mathbf{y}}} = \begin{bmatrix} -\frac{y_i}{\hat{y}_i} \end{bmatrix}_i \in \mathcal{R}^{k \times 1}$$

- $\frac{\partial \mathcal{L}}{\partial \mathbf{h}_3} :$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{h}_3} = \left(\frac{\hat{\mathbf{y}}}{\mathbf{h}_3} \right)^T \frac{\partial \mathcal{L}}{\partial \hat{\mathbf{y}}} = \left[\hat{y}_i \sum_{j=1}^k y_j - y_i \right]_i \in \mathcal{R}^{k \times 1}$$

- $\frac{\partial \mathcal{L}}{\partial \mathbf{w}_2} :$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}_2} = \mathbf{a}_2 \frac{\partial \mathcal{L}}{\partial \mathbf{h}_3} \in \mathcal{R}^{(s+d) \times k}$$

- $\frac{\partial \mathcal{L}}{\partial \mathbf{b}_2} :$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{b}_2} = \frac{\partial \mathcal{L}}{\partial \mathbf{h}_3} \in \mathcal{R}^{k \times 1}$$

- $\frac{\partial \mathcal{L}}{\partial \mathbf{a}_2} :$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{a}_2} = \mathbf{w}_2 \frac{\partial \mathcal{L}}{\partial \mathbf{h}_3} \in \mathcal{R}^{(s+d) \times 1}$$

- $\frac{\partial \mathcal{L}}{\partial \mathbf{h}_2}$:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{h}_2} = \left(\frac{\partial \mathbf{a}_2}{\partial \mathbf{h}_2} \right)^T \frac{\partial \mathcal{L}}{\partial \mathbf{a}_2} = \text{diag}\{m_1, m_2, \dots, m_{(s+d)}\} \cdot \frac{\partial \mathcal{L}}{\partial \mathbf{a}_2} \in \mathcal{R}^{(s+d) \times 1}$$

- $\frac{\partial \mathcal{L}}{\partial \mathbf{a}_1}$:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{a}_1} = \left(\frac{\partial \mathbf{h}_2}{\partial \mathbf{a}_1} \right)^T \frac{\partial \mathcal{L}}{\partial \mathbf{h}_2} = [\mathbf{I}_{s \times s} \quad \mathbf{0}_{s \times d}] \cdot \frac{\partial \mathcal{L}}{\partial \mathbf{h}_2} \in \mathcal{R}^{s \times 1}$$

- $\frac{\partial \mathcal{L}}{\partial \mathbf{h}_1}$:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{h}_1} = \left(\frac{\partial \mathbf{a}_1}{\partial \mathbf{h}_1} \right)^T \frac{\partial \mathcal{L}}{\partial \mathbf{a}_1} = \text{diag}\{sgn(\mathbf{a}_{11}), sgn(\mathbf{a}_{12}), \dots, sgn(\mathbf{a}_{1s})\} \cdot \frac{\partial \mathcal{L}}{\partial \mathbf{a}_1} \in \mathcal{R}^{s \times 1}$$

- $\frac{\partial \mathcal{L}}{\partial \mathbf{w}_1}$:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}_1} = \mathbf{x} \frac{\partial \mathcal{L}}{\partial \mathbf{h}_1} \in \mathcal{R}^{d \times s}$$

- $\frac{\partial \mathcal{L}}{\partial \mathbf{b}_1}$:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{b}_1} = \frac{\partial \mathcal{L}}{\partial \mathbf{h}_1} \in \mathcal{R}^{s \times 1}$$

- $\frac{\partial \mathcal{L}}{\partial \mathbf{x}}$:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{x}} = \mathbf{w}_1 \frac{\partial \mathcal{L}}{\partial \mathbf{h}_1} \in \mathcal{R}^{d \times 1}$$

8.2 多层感知机算法 MLP

太简单了不想讲了，本质上就是很多层全连接，全连接中间随便插入几层 Dropout。标准模型里面没有 Dropout，但是实际使用和设计 MLP 的时候最好自己在全连接层间随便放置一些 Dropout，Dropout 的参数大概在 0.1 到 0.5 范围内效果比较好。

关于 MLP 的层数选取，当然越深的全连接网络的表示能力是更强的，但是需要注意，随着层数增加，网络更有可能出现梯度消失的现象，因此加深层数有利有弊。

8.3 优化器

简单说一说，主要介绍三种，都是梯度下降方法：

- **Batch Gradient Descent**:

$$\theta = \theta - \eta \cdot \nabla_{\theta} J(\theta)$$

- **Stochastic Gradient Descent**:

$$\theta = \theta - \eta \cdot \nabla_{\theta} J(\theta, \mathbf{x}^{(i)}, \mathbf{y}^{(i)})$$

- **Mini-Batch Gradient Descent**:

$$\theta = \theta - \eta \cdot \nabla_{\theta} J(\theta, \mathbf{x}^{(i:i+k)}, \mathbf{y}^{(i:i+k)})$$

8.4 批标准化 Batch Normalization

假设 BN 层的输入是一个 batch 之中的 m 个样本 $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$, 那么 BN 层的输出是 $\{\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_m\}$, 输入和输出之间满足:

$$\begin{aligned} \mathbf{z}_i &= \gamma \hat{\mathbf{x}}_i + \beta \\ \hat{\mathbf{x}}_i &= \frac{\mathbf{x}_i - \boldsymbol{\mu}}{\sqrt{\sigma^2 + \epsilon}} \\ \boldsymbol{\mu} &= \frac{1}{m} \sum_{i=1}^m \mathbf{x}_i; \quad \sigma = \frac{1}{m} \sum_{i=1}^m \|\mathbf{x}_i - \boldsymbol{\mu}\|^2 \end{aligned}$$

批标准化的主要作用有两点: 第一, 可以加速神经网络的收敛; 第二, 提升神经网络的泛化能力。加速网络的收敛这个很容易理解, 本质上就是因为各层参数随着迭代轮次是变化的, 因此各个层输出的数据内部的方差变化很大, 如果进行标准化, 每层的输入和输出的方差就是一致的, 更好学习。(本质上就是对每一层进行了噪声的白化, 学过通信的都知道, 白化后的噪声更容易使用基础函数进行滤波)。提升网络的泛化能力这一点有争议, 不表。

8.5 卷积神经网络 CNN

假设我们的输入矩阵 $\mathbf{X} \in \mathcal{R}^{h \times w \times d}$, 那么卷积神经网络的含义是选取一个 $\mathbf{K} \in \mathcal{R}^{k \times k \times d}$ 的卷积核, 这个卷积核按照步长 s 从左至右、从上至下遍历整个 \mathbf{X} , 在经过的小区间之中进行元素相乘 (element-wise multiply) 从而得到输出矩阵。注意, 各个通道分别进行卷积之后, 对于每一个像素处, 多个通道数值相加。

首先讨论输出矩阵的大小：输出矩阵 $\mathbf{Y} \in \mathcal{R}^{[\frac{1}{s}(h-k+2p)+1] \times [\frac{1}{s}(w-k+2p)+1] \times f}$ 。这个表述之中 p 是填充 (padding)，好比如果 $p = 1$ ，那输入矩阵变为 $\mathbf{X}_{pad} \in \mathcal{R}^{(k+2) \times (k+2) \times d}$ ，也就是矩阵上下左右各自长出一溜。上式中 f 是这次卷积采用的卷积核个数，卷积核的数量就是输出的通道数。

下面给出卷积过程的数学表示：

$$\mathbf{Y}[i : (i+k)][j : (j+k)][l] = \sum_{m=1}^d \mathbf{X}[i : (i+k)][j : (j+k)][m] \odot \mathbf{K}_l[:, :, m]$$

8.6 池化 Pooling

对于池化操作，假设池化层步长为 s ，那么：

$$\mathbf{Y}[i][j][:] = P(\mathbf{X}[i : (i+s)][j : (j+s)][:])$$

- 最大池化：

$$P(\mathbf{X}[i : (i+s)][j : (j+s)][:]) = \max_{x \in [i, i+s], y \in [j, j+s]} \{\mathbf{X}[x][y][:]\}$$

- 平均池化：

$$P(\mathbf{X}[i : (i+s)][j : (j+s)][:]) = avg(\mathbf{X}[i : (i+s)][j : (j+s)][:])$$

- 随机池化：

$$P(\mathbf{X}[i : (i+s)][j : (j+s)][:]) = \frac{\mathbf{X}[x][y][:]}{\sum_{l=i}^{i+s} \sum_{m=j}^{j+s} \mathbf{X}[l][m][:]}$$

上式中 x 和 y 是从 $[i, i+s]$ 和 $[j, j+s]$ 之中随便选的。

8.7 残差神经网络 Residual

ResNet 提出者孙剑博士于本文档编撰期间（2022 年 06 月 14 日）因突发疾病逝世，谨此缅怀。

残差运算如下式所示：

$$\mathcal{F}(\mathbf{x}) = \mathbf{W}_2 \cdot \sigma(\mathbf{W}_1 \mathbf{x})$$

$$\mathbf{z} = \mathcal{F}(\mathbf{x}) + \mathbf{x}$$

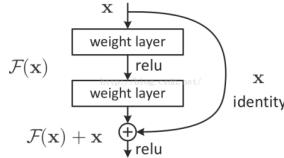


Fig 7. 残差层

这里简单解释一下残差层的意义，意义在于跳层连接能够保护原始信息不会因梯度消失而丢失，同时只需要学习输入和输出之间有差别的部分，简化了学习难度。

8.8 循环神经网络 RNN

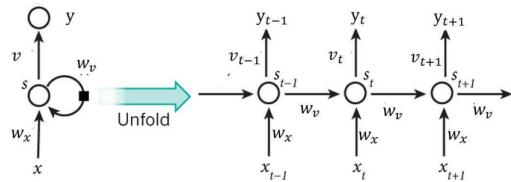


Fig 8. RNN 神经网络

如上图所示展示的是 RNN 的基本模型图，基本的数学公式如下所示：

$$\mathbf{y}_t = \sigma_y(\mathbf{W}_y \mathbf{v}_t + \mathbf{b}_y)$$

$$\mathbf{v}_t = \sigma_v(\mathbf{W}_x \mathbf{x}_t + \mathbf{W}_v \mathbf{v}_{t-1} + \mathbf{b}_v)$$

上式之中的两个激活函数是分别对于输入的每一个分量进行激活，因此输出是一个向量。我们给出一个最为常用的具体例子，不妨假设 $\mathbf{x}_t, \mathbf{y}_t \in \mathcal{R}^{d \times 1}$ 并且 $\mathbf{v}_t \in \mathcal{R}^{m \times 1}$ ，于是可以推导出各个参数的维度是：

$$\mathbf{W}_x \in \mathcal{R}^{m \times d}; \mathbf{W}_v \in \mathcal{R}^{m \times m}; \mathbf{W}_y \in \mathcal{R}^{d \times m}$$

$$\mathbf{b}_v \in \mathcal{R}^{m \times 1}; \mathbf{b}_y \in \mathcal{R}^{d \times 1}$$

进而一种最常用的 RNN 实例是：

$$\mathbf{y}_t = \text{Softmax}(\mathbf{W}_y \mathbf{v}_t + \mathbf{b}_y)$$

$$\mathbf{v}_t = \tanh(\mathbf{W}_x \mathbf{x}_t + \mathbf{W}_v \mathbf{v}_{t-1} + \mathbf{b}_v)$$

于是我们可以求取其 BP 算法如下（随便演示一个比较难的，其他的用前面讲的链式法则就可以求，选这个主要是演示一下向量对矩阵的求导）：

- $\frac{\partial \mathbf{y}_t}{\partial \mathbf{W}_y}$:

$$\frac{\partial \mathbf{y}_t}{\partial \mathbf{W}_y} = \left[\frac{\partial \mathbf{y}_t[i]}{\partial \mathbf{W}_y} \right]_i \in \mathcal{R}^{d^2 \times m}$$

上式之中的 $\mathbf{y}_t[i]$ 意思是向量 \mathbf{y}_t 的第 i 维上的分量; \cdot_i 的含义是这个向量/矩阵的第 i 个元素, 这个第 i 个元素可以是矩阵, 是矩阵的情况那就按照从上到下顺次排列 (这是向量对矩阵求导的基本知识)。接着由于:

$$\mathbf{y}_t[i] = \text{Softmax} \left(\sum_{j=1}^m \mathbf{W}_y[i][j] \cdot \mathbf{v}_t[j] + \mathbf{b}_y[i] \right)$$

于是 (下式之中 $k \neq i$):

$$\begin{cases} \frac{\partial \mathbf{y}_t[i]}{\partial \mathbf{W}_y[i][j]} = (\mathbf{y}_t[i] - \mathbf{y}_t^2[i]) \cdot \mathbf{v}_t[j] \\ \frac{\partial \mathbf{y}_t[i]}{\partial \mathbf{W}_y[k][j]} = 0 \end{cases}$$

$$\frac{\partial \mathbf{y}_t[i]}{\partial \mathbf{W}_y}[p][q] = \begin{cases} (\mathbf{y}_t[i] - \mathbf{y}_t^2[i]) \cdot \mathbf{v}_t[q] & p = i \\ 0 & p \neq i \end{cases} \in \mathcal{R}^{d \times m}$$

最终:

$$\frac{\partial \mathbf{y}_t}{\partial \mathbf{W}_y} = \begin{bmatrix} \frac{\partial \mathbf{y}_t[1]}{\partial \mathbf{W}_y} \\ \frac{\partial \mathbf{y}_t[2]}{\partial \mathbf{W}_y} \\ \vdots \\ \frac{\partial \mathbf{y}_t[d]}{\partial \mathbf{W}_y} \end{bmatrix} \in \mathcal{R}^{d^2 \times m}$$

上面的一个例子其实没有演示完全, RNN 做反向传播的时候比较麻烦的是需要求和。为了演示, 我们再给出分别对于 \mathbf{W}_x 和 \mathbf{W}_v 求导的例子, 但是就不详细推导了。

$$\begin{aligned} \frac{\partial \mathbf{y}_t}{\partial \mathbf{W}_x} &= \sum_{k=1}^t \frac{\partial \mathbf{y}_t}{\partial \mathbf{v}_t} \frac{\partial \mathbf{v}_t}{\partial \mathbf{v}_k} \frac{\partial \mathbf{v}_k}{\partial \mathbf{W}_x} \\ \frac{\partial \mathbf{y}_t}{\partial \mathbf{W}_v} &= \sum_{k=1}^t \frac{\partial \mathbf{y}_t}{\partial \mathbf{v}_t} \frac{\partial \mathbf{v}_t}{\partial \mathbf{v}_k} \frac{\partial \mathbf{v}_k}{\partial \mathbf{W}_v} \end{aligned}$$

上式之中:

$$\frac{\partial \mathbf{v}_t}{\partial \mathbf{v}_k} = \prod_{i=k+1}^t \frac{\partial \mathbf{v}_i}{\partial \mathbf{v}_{i-1}}$$

从上面的推导之中我们不难发现, RNN 必然面对着梯度消失或梯度爆炸的问题! 主要原因是 RNN 之中的权重是级联相乘的, 如果一个元素比较

大/小，那么必然会顺着矩阵传导到下一个单元，从而使其进一步变大/变小！处理的方法有很多，最简单的一种是设置截断机制，就是梯度在 $[\theta_1, \theta_2]$ 区间之外则截断。

8.9 长短时期记忆 LSTM

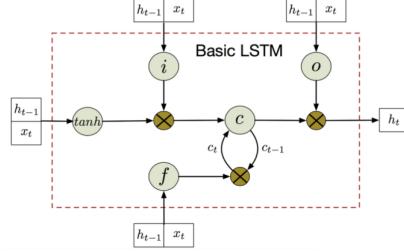


Fig 9. LSTM 神经网络

如上图所示展示的是 LSTM 的基本模型图，基本的数学公式如下所示：

$$i_t = \text{Softmax}(\mathbf{W}_{xi}x_t + \mathbf{W}_{hi}h_{t-1} + \mathbf{b}_i)$$

$$f_t = \text{Softmax}(\mathbf{W}_{xf}x_t + \mathbf{W}_{hf}h_{t-1} + \mathbf{b}_f)$$

$$o_t = \text{Softmax}(\mathbf{W}_{xo}x_t + \mathbf{W}_{ho}h_{t-1} + \mathbf{b}_o)$$

$$g_t = \tanh(\mathbf{W}_{xg}x_t + \mathbf{W}_{hg}h_{t-1} + \mathbf{b}_g)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t$$

$$h_t = o_t \odot \tanh(c_t)$$

上式之中 \odot 算子表示元素相乘 (element-wise multiply)。LSTM 被提出的主要原因是并不像 RNN 一样记住所有时刻的输入；而是引入了门机制对于记忆的程度进行控制。上述推导过程中的 f_t 也被称为遗忘门，是最重要的门，主要作用是控制记忆，它的每一个元素都在 $[0,1]$ 区间内。 i_t 被称为输入门， o_t 被称为输出门。三个门的重要性排序是遗忘门大于输入门大于输出门。

除了上述的优点之外，LSTM 缓解了 RNN 的梯度爆炸或梯度消失问题。主要原因是使用遗忘门来控制长期记忆的权重，如果长期记忆比较大（爆炸），那么遗忘门的输出会倾向于记忆这个长期效应；如果长期记忆偏小（消失），那么遗忘门会倾向于抛弃这个长期效应。不论是记住还是抛弃，遗忘门的输出都是 $[0,1]$ 之间的，因此并不会产生消失或爆炸。

8.10 自动编解码器 En/De-coder

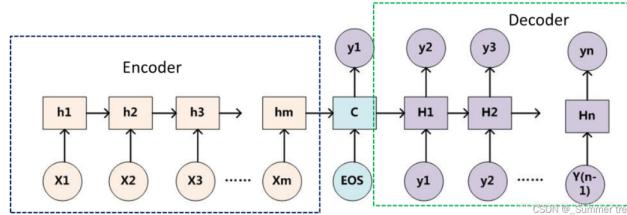


Fig 10. Encoder-Decoder 神经网络

刚刚引入的 RNN 和 LSTM 其实可以被理解为是一种有监督编码，因为事实上 LSTM 和 RNN 都把输入参量 $\{x_1, x_2, \dots, x_N\}$ 在输出参量 $\{y_1, y_2, \dots, y_N\}$ 的监督下转换为了一组隐含状态变量 $\{(h_1, c_1), (h_2, c_2), \dots, (h_N, c_N)\}$ (LSTM) 或 $\{v_1, v_2, \dots, v_N\}$ (RNN)。

Encoder-Decoder 的数学公式如下表示：

$$C = \mathcal{F}(x_1, x_2, \dots, x_N)$$

$$y_i = \mathcal{G}(C, y_1, y_2, \dots, y_{i-1})$$

8.11 注意力机制 Attention

8.11.1 ATTENTION IS ALL YOU NEED?

在引入注意力机制之前，我们先看一看这个论文的标题。这个梗感觉像是“PHP 是世界上最好的语言”或者“python import this: The Zen of Python”一样。不过我们提出一个小疑问，注意力很重要吗？

从我们日常生活中来看，注意力很重要。脑科学表示，人类的模式识别能力很大程度上依赖于图像的部分区域，这些区域对于人脑的特定神经元起到了独特的刺激作用，从而使得人类能够进行识别。事实上，“管中窥豹，可见一斑”说的就是注意力机制，因为豹子身上的花纹能够激活人类的特定神经元，因此只看到豹子的花纹对于识别“这个动物是否是豹子”这样一个二分类问题就已经是足够的了。

在上面的论述中我们不难得到引入注意力的优点：第一，样本筛选从而提升泛化能力，即对于重要的样本倾注更多的学习资源，这样的做法能够从样本中选出最有价值的信息，这往往就是更为普遍的知识；第二，解决信息过载，扔掉一些不重要的信息，能够使得网络更集中于特定任务，提升鲁棒性，避免冗余或无关信息干扰。

既然已经从生物学的角度看到了注意力机制，我们将其转化为数学语言来说就是：对于统计学习算法的输入进行赋权，所赋的权值与这个输入的“重要性”有关。

8.11.2 软注意力机制 Soft-Attention

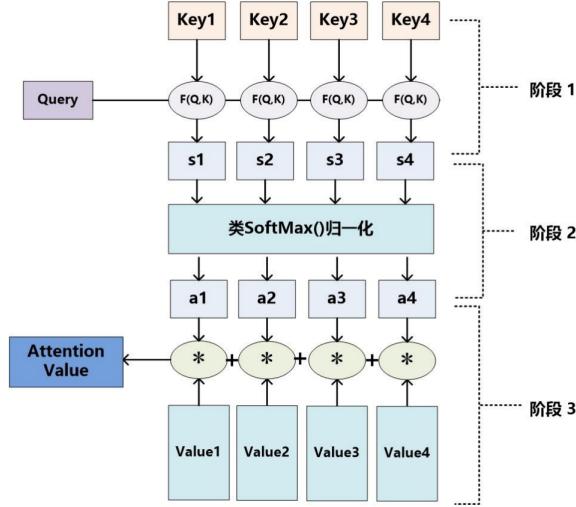


Fig 11. 注意力机制

软注意力机制就是根据重要性加权。数学表示有三个式子：

$$Sim_i = \text{Similarity}(Query, Key_i)$$

$$a_i = \text{Softmax}(Sim_i) = \frac{e^{Sim_i}}{\sum_{j=1}^N e^{Sim_j}}$$

$$\text{Attention}(Query, Source) = \sum_{i=1}^N a_i \cdot Value_i$$

上面的式子之中主要提到了三种量，Query、Key 和 Value；以及一种函数 Similarity。接下来分别介绍：

- **Value**: $Value_i$ 的本质就是输入的数据 x_i 。上式之中 $Source = \{x_1, x_2, \dots, x_N\}$ 。
 $Value_i$ 就是输入量，可以是向量、矩阵等等。
- **Key**: Key_i 的本质是对于输入数据 $Value_i$ 的一种向量表示（或者说是一种编码）。在很多的情况下 Key_i 就是 $Value_i$ ，当然也可以是不同的。有点像是在 $Value$ 样本空间内的“坐标”的概念。

- **Query**: *Query* 的本质是用于作为一个“标准向量”，物理含义是值得注意的事物，用于引导注意力。
- **Similarity**: *Similarity* 的本质是一个用于衡量相似程度的函数。在上面的式子之中， $Similarity(Query, Key_i)$ 用于衡量 *Query* 和 Key_i 的相似程度。由于 *Query* 是我们认为网络需要关注的标准，因此如果 Key_i 越贴近于这个标准，与 *Query* 的相似程度也就越高，也因此 $Similarity(Query, Key_i)$ 对应的函数数值也就越大，产生的权重 a_i 也就越大。

上面对于各个参数进行了解释，这里我按照白话再讲一遍。注意力机制本质就是对于样本集 $Source = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\} = \{Value_1, Value_2, \dots, Value_N\}$ 加权重。这个权重来源于对于某个样本应当施加关注的程度。那么如何衡量这个关注程度，我们提出：给出我们认为神经网络应当关注的目标标准 *Query*，如果输入的每个样本 $Value_i$ 对应的编码 Key_i 与这个 *Query* 越相似，就给这个 $Value_i$ 更大的权重。衡量相似度由 *Similarity* 函数给出。
下面给出几种比较常用的相似度函数：

- 投影/内积相似度：

$$Similarity(Query, Key_i) = Query \cdot Key_i$$

- 余弦相似度：

$$Similarity(Query, Key_i) = \frac{Query \cdot Key_i}{\|Query\| \cdot \|Key_i\|}$$

- 神经网络相似度：

$$Similarity(Query, Key_i) = Neural(Query, Key_i)$$

至此，软注意力的基本原理介绍完毕，不过其实并不具体，在 7.11.4 小节之中给出注意力机制在 Encoder-Decoder 之中的实际应用就更加明确了。

8.11.3 硬注意力机制 Hard-Attention

硬注意力机制就是只选择最重要的样本，其他样本直接抛弃。（这就是真·管中窥豹了）。用的少且简单，不说了。

8.11.4 软注意力机制在 Encoder-Decoder 中的运用

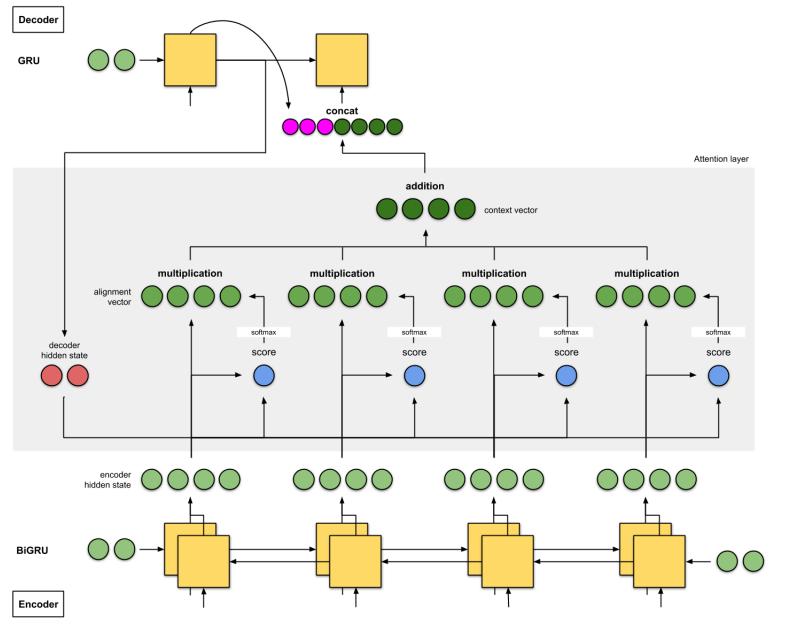


Fig 12. Encoder-Decoder 神经网络引入注意力机制

上图是参考网站之中给出的实例介绍，写得是比较完整的了。下面用我自己的话再介绍一遍。

在上图之中，注意力机制是在 Encoder 之后、Decoder 之前引入的，在上图的计算过程中认为 Decoder 的第一个 GRU 之中的隐变量 $h_d^{(0)}$ 是 $Query$ ，认为 Encoder 的每一个 GRU 的隐变量 $h_e^{(j)}$ 是 $Value_j$ ，也是 Key_j （在这个例子之中 $Value_j$ 和 Key_j 是一样的）。在上图的例子之中求取相似性分数的 $Similarity$ 函数是投影相似度（就是点乘）。于是可以使用 7.11.2 小节之中的数学公式计算出本实例之中的注意力机制：

$$\begin{aligned} Sim_i &= h_d^{(0)} \cdot h_e^{(i)} \\ a_i &= \frac{e^{h_d^{(0)} \cdot h_e^{(i)}}}{\sum_{j=0}^3 e^{h_d^{(0)} \cdot h_e^{(j)}}} \\ \text{Attention} &= \sum_{i=0}^3 a_i h_e^{(i)} = \sum_{i=0}^3 \frac{e^{h_d^{(0)} \cdot h_e^{(i)}}}{\sum_{j=0}^3 e^{h_d^{(0)} \cdot h_e^{(j)}}} h_e^{(i)} \end{aligned}$$

上图之中给出了 Decoder 的第二个 GRU 输入就是 $\mathbf{h}_d^{(0)}$ 与上面求出的 *Attention* 进行拼接：

$$\mathbf{x}_d^{(1)} = \text{Concat}(\mathbf{h}_d^{(0)}, \mathbf{Attention})$$

这里需要提醒大家！*Query* 和 *Key* 本质上是向量！*Attention* 本质上是和 *Value* 一样的元素，可以是向量、矩阵等等！

8.12 图卷积神经网络 GCN

8.12.1 图卷积神经网络问题提出

为什么需要图卷积神经网络，这是由数据的数学形态决定的。在医药化学分子性能预测、社交网络关系检测等的问题之中，建立的模型从形态上来讲更像是一个图结构。因此针对这种特殊的数据结构，设计出适合于数据结构的神经网络算法就是十分必要的。

8.12.2 图卷积神经网络原理

对于一个给定的无向图 $\mathcal{G} = (V, E)$ ， V 是无向图的节点集合； E 是无向图边的集合。一般而言为了对于无向图进行数学表示，常常把每一个节点使用一个向量来进行描述。因此 V 和 E 的数学形式如下所示：

$$V = \{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_N\}$$

$$E = \{(\mathbf{v}_i, \mathbf{v}_j)\}, \forall i \neq j$$

那么为了对于节点以及对应的边和邻居的信息进行提取，我们采用如下的方式进行：

$$\mathbf{H}_{l+1} = \sigma(\mathbf{A}\mathbf{H}_l\mathbf{W}_l)$$

这就是一层图卷积层。上式之中的 \mathbf{A} 是原图 \mathcal{G} 产生的邻接矩阵； \mathbf{H}_l 是第 l 层图卷积的输入； \mathbf{W}_l 是第 l 层图卷积的可训练权重矩阵； \mathbf{H}_{l+1} 是第 $l+1$ 层图卷积的输出（也是第 $l+1$ 层图卷积的输入）。

我们简要解释一下为何如此设计，如此设计的主要原因是希望能够提取出每个节点周围的邻接节点的特征；逐层迭代后每个节点的特征（就是最后一层卷积层的输出矩阵 \mathbf{H}_l 的第 i 行） $\hat{\mathbf{v}}_i = \mathbf{H}_l^{(i)}$ 就是原图各个节点按照远近亲疏关系加权的和，从而完成特征提取。或许会有疑问为什么是矩阵点乘，

如果是矩阵加法的话，很容易出现梯度爆炸的现象。

但是上述设计并不很好，因为邻接矩阵实际上忽略了每个节点自己的信息，于是就有如下形式：

$$\mathbf{H}_{l+1} = \sigma(\tilde{\mathbf{A}}\mathbf{H}_l\mathbf{W}_l)$$

$$\tilde{\mathbf{A}} = \mathbf{A} + \lambda \mathbf{I}$$

λ 是一个可训练的参数，代表自己节点的重要性。

经过上述改进后图卷积能够学到自己节点的特征。但是仍然存在梯度消失和梯度爆炸的可能性，因此继续引入矩阵标准化。从上面介绍中我们知道 \mathbf{A} 是邻接矩阵，邻接矩阵每一行加起来是这个节点的“度”，为了进行矩阵的标准化，我们需要使得其行和为 1，因此用于标准化矩阵应当是度矩阵的逆：

$$\begin{aligned}\mathbf{H}_{l+1} &= \sigma(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}_l\mathbf{W}_l) \\ \tilde{\mathbf{A}} &= \mathbf{A} + \lambda \mathbf{I} \in \mathcal{R}^{N \times N} \\ \tilde{\mathbf{D}} &= \text{diag} \left\{ \sum_{j=1}^N \tilde{\mathbf{A}}[1][j], \sum_{j=1}^N \tilde{\mathbf{A}}[2][j], \dots, \sum_{j=1}^N \tilde{\mathbf{A}}[N][j] \right\} \in \mathcal{R}^{N \times N} \\ \mathbf{H}_l &\in \mathcal{R}^{N \times d_{l+1}}; \quad \mathbf{H}_l \in \mathcal{R}^{N \times d_l}; \quad \mathbf{W}_l \in \mathcal{R}^{d_l \times d_{l+1}}; \quad \mathbf{H}_0 = \mathbf{X} \in \mathcal{R}^{N \times C}\end{aligned}$$

上式之中， \mathbf{X} 是如下表示， C 是每个节点的特征维度：

$$\mathbf{X} = [\mathbf{v}_1 \quad \mathbf{v}_2 \quad \cdots \quad \mathbf{v}_N]^T \in \mathcal{R}^{N \times C}$$

我们发现第 1 层的输出矩阵 $\mathbf{H}_{l+1} \in \mathcal{R}^{N \times d_{l+1}}$ ，如果图卷积一共是 m 层，最后一层的输出就可以是 $\mathbf{H}_{m+1} \in \mathcal{R}^{N \times d_{m+1}}$ ，当 $d_{m+1} = K$ 时就可以用作对每个节点 K 分类问题的训练；当 $d_{m+1} = K$ 时，添加一个全连接层，令 $\hat{\mathbf{H}}_{m+1} = \mathbf{w}_{out}^T \cdot \mathbf{H}_{m+1} \in \mathcal{R}^{1 \times d_{m+1}}$ ，就可以用作对整个图 K 分类问题的训练；当 $d_{m+1} = 1$ 时，同样添加一个全连接层，就可以用作对回归问题的训练。

这里我举一个例子，在化学制药领域，我们往往希望通过观察一个化学分子的化学结构来预测这个分子对于某种病是否有治愈效果（即对整个图的有效-无效的 0-1 二分类）。训练集是每一个分子的无向图结构、以及这个分子有效或无效的 0-1 标签。于是我们可以构建图卷积神经网络，令最后一层的输出是 $\mathbf{H}_{m+1} \in \mathcal{R}^{N \times 2}$ ，再添加一层全连接转化为 $\hat{\mathbf{H}}_{m+1} \in \mathcal{R}^{1 \times 2}$ ，就可以对这个问题进行学习。

9 非监督数据降维

9.1 主成分分析 PCA

9.1.1 PCA 问题引入与最优子空间

主成分分析面对的关键问题是维数灾难，针对一个维度为 D （很大）的样本空间生成的样本序列进行学习，往往需要更多的神经元、更深的网络层数以及更加精巧设计的算法以避免梯度消失和梯度爆炸，而且很难得到较好的训练效果。既然如此，降维就成为了十分重要的一点。

当我们并不知道这些数据我们将会用于什么问题时，并不能够使用有监督的数据降维。此时需要引入无监督的数据降维。既然是无监督的数据降维，也就不存在有监督的“优化目标”（例如 Fisher 准则）来衡量降维效果，于是就需要重新创造一个无监督的“优化目标”。这个优化目标就是最优子空间。

最优子空间是原 D 维空间的维度为 d 的子空间，这个子空间能够使得原空间以最小均方误差被 d 维空间内的基线性表出。这一句话其实信息量很丰富：第一，确定了一个无监督“优化目标”，子空间以最小均方误差线性表出原空间；第二，子空间的维度仅仅是 $d < D$ ，可以任意选取，具有普适性。

9.1.2 PCA 算法推导

上面一个小节给出了基本目标，本小节进行推导。从 D 维原空间内随机产生了一个样本 \mathbf{x} ，在这个 D 维空间内存在一组完备的标准正交基 $\{\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_D\}$ 。那么：

$$\mathbf{x} = \sum_{i=1}^D c_i \mathbf{u}_i$$

我们取这组正交基的前 d 个张成最优子空间，满足最小化均方误差，那么现在推导一下这前 d 个标准正交基应当满足的选取条件。首先给出在 d 维最优子空间内 \mathbf{x} 对应的元素 $\hat{\mathbf{x}}$ ：

$$\hat{\mathbf{x}} = \sum_{i=1}^d c_i \mathbf{u}_i$$

于是均方误差是：

$$\xi = E [(\mathbf{x} - \hat{\mathbf{x}})^T (\mathbf{x} - \hat{\mathbf{x}})]$$

$$\begin{aligned}\xi &= E \left[\left(\sum_{i=d+1}^D c_i \mathbf{u}_i \right)^T \left(\sum_{i=d+1}^D c_i \mathbf{u}_i \right) \right] \\ \xi &= E \left[\sum_{i=d+1}^D c_i^2 \right]\end{aligned}$$

又由于：

$$c_i = \mathbf{x}^T \mathbf{u}_i = \mathbf{u}_i^T \mathbf{x}$$

因此：

$$\begin{aligned}\xi &= E \left[\sum_{i=d+1}^D \mathbf{u}_i^T \mathbf{x} \mathbf{x}^T \mathbf{u}_i \right] \\ \xi &= \sum_{i=d+1}^D \mathbf{u}_i^T E[\mathbf{x} \mathbf{x}^T] \mathbf{u}_i\end{aligned}$$

我们引入所有样本的协方差矩阵 $\Psi = E[\mathbf{x} \mathbf{x}^T]$, 于是：

$$\xi = \sum_{i=d+1}^D \mathbf{u}_i^T \Psi \mathbf{u}_i$$

所以完整的最小化均方误差的完整表示形式就是：

$$\begin{aligned}\{\hat{\mathbf{u}}_i\} &= \arg \min_{\{\mathbf{u}_i\}} \sum_{i=d+1}^D \mathbf{u}_i^T \Psi \mathbf{u}_i \\ \text{s.t. } \mathbf{u}_i^T \mathbf{u}_i &= 1, \forall \text{ integer } i \in [d+1, D]\end{aligned}$$

又到了熟悉的拉格朗日函数构造环节：

$$\begin{aligned}\mathcal{L} &= \sum_{i=d+1}^D \mathbf{u}_i^T \Psi \mathbf{u}_i - \sum_{i=d+1}^D \lambda_i (\mathbf{u}_i^T \mathbf{u}_i - 1) \\ \frac{\partial \mathcal{L}}{\partial \mathbf{u}_j} &= 2(\Psi - \lambda_j \mathbf{I}) \mathbf{u}_j = \mathbf{0} \\ \xi_{min} &= \sum_{i=d+1}^D \lambda_i\end{aligned}$$

从上面的求解形式上来看, $\{\mathbf{u}_i\}, i \in [d+1, D]$ 分别是 Ψ 矩阵的特征向量, ξ 是第 $d+1$ 到第 D 个特征值的求和。为了使得 ξ 真正取到最小值, $\{\lambda_{d+1}, \lambda_{d+2}, \dots, \lambda_D\}$ 应当是 Ψ 矩阵的最小的 $D-d$ 个特征值, $\{\mathbf{u}_i\}, i \in [d+1, D]$ 是对应于最小的 $D-d$ 个特征值的特征向量。

从而, 用于生成最优子空间的特征向量就对应于矩阵 Ψ 的前 d 大特征值的特征向量。顺便, 本质上 PCA 就是 KL 展开、也是 SVD 分解。

9.1.3 PCA 算法流程

输入数据是 $\mathbf{X} = [\mathbf{x}_1 \ \mathbf{x}_2 \ \cdots \ \mathbf{x}_N] \in \mathcal{R}^{D \times N}$ 的 N 个 D 维样本:

- 计算样本均值:

$$\boldsymbol{\mu} = \sum_{i=1}^N \mathbf{x}_i$$

- 对数据进行中心化:

$$\mathbf{x}_i = \mathbf{x}_i - \boldsymbol{\mu}$$

- 计算协方差矩阵:

$$\Psi = \frac{1}{N} \mathbf{X} \mathbf{X}^T$$

- 对协方差矩阵特征值分解

- 取前 d 大的特征值对应的特征向量构成投影矩阵:

$$\mathbf{U} = [\mathbf{u}_1 \ \mathbf{u}_2 \ \cdots \ \mathbf{u}_d]$$

- 将样本投影到最优子空间:

$$\hat{\mathbf{X}} = \mathbf{U}^T \mathbf{X}$$

请务必注意中心化，非常重要！原因很简单，如下图。

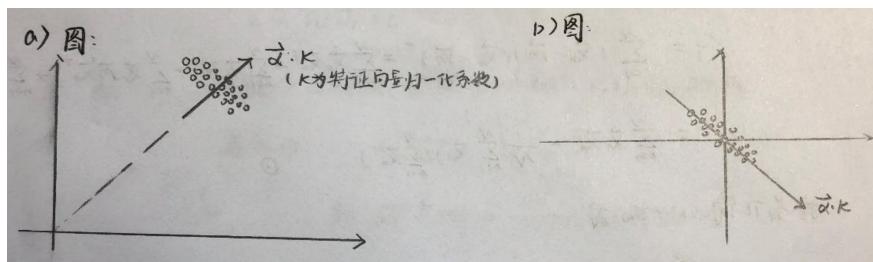


Fig 13. PCA 不中心化和中心化提取的最大特征值对应特征向量

9.2 多维尺度变换 MDS

9.2.1 多维尺度变换的问题提出

多维尺度变换面对的基本问题是：当我们已知一些样本之间的相似性的时候，我们可以根据这些相似性把数据样本投影到二维或三维欧氏空间，投影

的时候还能够基本上保证这些样本的相似性，在投影后的二维或三维空间上可视化地直观分析这些样本。

这种寻找一种数据变换使得样本点在新空间中的距离反映原空间数据之间相似性的过程被称为多维尺度变换。一般用于数据可视化。

9.2.2 多维尺度变换的数学推导

单纯从算法层面上来讲，MDS 就是变相 PCA。至于为什么这么说，看完推导就明白了。

首先明确我们的已知量，我们仅仅已知各个样本之间的距离。不妨假设 D 维原空间之中 N 个样本对应的样本矩阵（这个是未知的，仅仅是假设）是：

$$\mathbf{X} = [\mathbf{x}_1 \quad \mathbf{x}_2 \quad \cdots \mathbf{x}_N] \in \mathcal{R}^{D \times N}$$

在上面的矩阵之中，我们认为原空间之中的样本都是已经经过中心化之后的！这样的假设是成立的，因为我们真正已知的只有原空间之中样本之间的距离，是否进行中心化并不影响距离的计算。因此对于所有的样本点满足如下式子：

$$\sum_{i=1}^N \mathbf{x}_i = \mathbf{0}$$

我们的已知量就是距离矩阵 ***Distance***：

$$\text{Distance} = [d_{rs}]_{rs}$$

其中：

$$d_{rs}^2 = (\mathbf{x}_r - \mathbf{x}_s)^T (\mathbf{x}_r - \mathbf{x}_s)$$

$$d_{rs}^2 = \mathbf{x}_r^T \mathbf{x}_r + \mathbf{x}_s^T \mathbf{x}_s - 2\mathbf{x}_r^T \mathbf{x}_s$$

接下来的推导目的是求出一个矩阵 $\mathbf{B} = \mathbf{X}\mathbf{X}^T$ ，于是带入上面提到的样本中心化条件：

$$\frac{1}{N} \sum_{r=1}^N d_{rs}^2 = \frac{1}{N} \sum_{r=1}^N \mathbf{x}_r^T \mathbf{x}_r + \mathbf{x}_s^T \mathbf{x}_s$$

$$\frac{1}{N} \sum_{s=1}^N d_{rs}^2 = \mathbf{x}_r^T \mathbf{x}_r + \frac{1}{N} \sum_{s=1}^N \mathbf{x}_s^T \mathbf{x}_s$$

$$\frac{1}{N^2} \sum_{r=1}^N \sum_{s=1}^N d_{rs}^2 = \frac{2}{N} \sum_{r=1}^N \mathbf{x}_r^T \mathbf{x}_r$$

对于 $\mathbf{B} = \mathbf{XX}^T$ 矩阵之中的 [r][s] 位置上的元素：

$$b_{rs} = \mathbf{x}_r^T \mathbf{x}_s = -\frac{1}{2} \left(d_{rs}^2 - \frac{1}{N} \sum_{r=1}^N d_{rs}^2 - \frac{1}{N} \sum_{s=1}^N d_{rs}^2 + \frac{1}{N^2} \sum_{r=1}^N \sum_{s=1}^N d_{rs}^2 \right)$$

那么如果我们有如下记法：

$$\begin{aligned} \mathbf{A} &= [a_{rs}]_{rs} \\ a_{rs} &= -\frac{1}{2} d_{rs}^2 \\ \mathbf{l} &= \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}^T \\ \mathbf{H} &= \mathbf{I} - \frac{1}{N} \mathbf{l} \mathbf{l}^T \end{aligned}$$

于是：

$$\mathbf{B} = \mathbf{H} \mathbf{A} \mathbf{H} = \Psi$$

至此我们已经求出了原空间内数据样本的协方差矩阵，下一步就是按照 PCA 的要求，求取矩阵 \mathbf{B} 的前 d 大的特征值对应的特征向量作为降维后的最优子空间的一组基，取每个样本在这组基上的坐标作为新空间内样本点的坐标。

最后举用一个 MDS 最经典的城市相对位置估计的例子作为结束。

	北京	天津	上海	重庆	呼市	乌市	拉萨	银川	南宁	哈尔滨
北京	0	125	1239	3026	480	3300	3736	1192	2373	1230
天津	125	0	1150	1954	604	3330	3740	1316	2389	1207
上海	1239	1150	0	1945	1717	3929	4157	2092	1892	2342
重庆	3006	1954	1945	0	1847	3202	2457	1570	993	3156
呼市	480	604	1717	1847	0	2825	3260	716	2657	1710
乌市	3300	3330	3929	3202	2825	0	2668	2111	4279	4531
拉萨	3736	3740	4157	2457	3260	2668	0	2547	3431	4967
银川	1192	1316	2092	1570	716	2111	2547	0	2673	2422
南宁	2373	2389	1892	993	2657	4279	3431	2673	0	3592
哈尔滨	1230	1207	2342	3156	1710	4531	4967	2422	3592	0

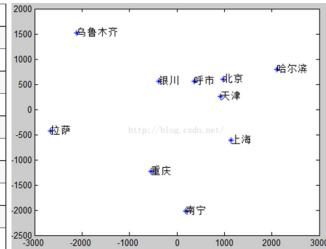


Fig 14. MDS 的城市计算实例（左侧为距离矩阵；右侧为二维可视化效果）

9.3 非线性降维方法 ISOMAP

9.3.1 ISOMAP 问题引入

ISOMAP 的全称是 Isometric Feature Mapping，保距特征变换（或者保距特征映射）。ISOMAP 面对的问题其实和刚刚讲的 MDS 十分相似，最为关键的区别在于：MDS 使用的距离矩阵是样本在高维空间之中的欧氏距离计算得到的；而 ISOMAP 使用的距离矩阵是使用样本在高维空间之中的测地

距离计算得到的。使用测地距离的原因是：ISOMAP 认为样本在高维空间内密集分布，并且构成了特定的“流型”（manifold）。能够说明这一点的最著名的例子就是“瑞士卷”数据集（Swiss Roll）。如下图所示是瑞士卷数据集的例子，下图 A 子图之中蓝色的虚线代表的是 MDS 使用欧氏距离衡量高维空间内两个样本的距离，而蓝色实线是 ISOMAP 的测地距离。

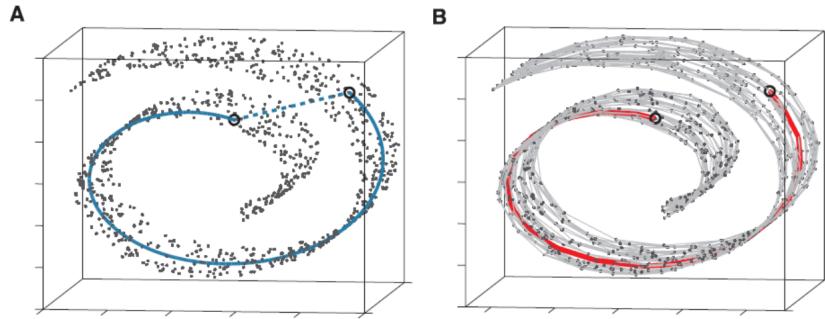


Fig 15. ISOMAP 测地距离在 Swiss Roll 上的表示

9.3.2 ISOMAP 的算法流程

ISOMAP 算法具体可以由下面的三个步骤构成：

- 第一步：在所有样本点上定义无向图 \mathcal{G} ，如果样本点 x_i 和样本点 x_j 之间的欧氏距离小于 ϵ ，那么认为样本点 x_i 和样本点 x_j 互为近邻点，连接两个样本。连线的长度定义为他们之间的欧氏距离 $d_X(i, j)$
- 第二步：计算无向图 \mathcal{G} 上每两个点之间的最短距离。初始化时如果样本点 x_i 和样本点 x_j 互为近邻，令 $d_{\mathcal{G}}(i, j) = d_X(i, j)$ ，否则 $d_{\mathcal{G}}(i, j) = +\infty$ 。然后对于距离矩阵之中的每个元素 $d_{\mathcal{G}}(i, j)$ 循环更新其数值为 $d_{\mathcal{G}}(i, j) = \min\{d_{\mathcal{G}}(i, j), d_{\mathcal{G}}(i, k) + d_{\mathcal{G}}(k, j)\}, \forall k = 1, 2, \dots, N$ 。最终得到测地距离矩阵为：

$$\mathbf{D}_{\mathcal{G}} = [d_{\mathcal{G}}(i, j)]_{ij}$$

- 第三步：在这个 $\mathbf{D}_{\mathcal{G}}$ 上使用 MDS 算法进行数据降维。

9.3.3 关于 ISOMAP 算法的一些讨论

ISOMAP 本质是什么？我认为是展平。整体上来讲，MDS 是一种“保距变换”，其所“保”的“距”就是作为 MDS 输入的距离矩阵。原始的 MDS 算

法所“保”的是“欧氏距离”，ISOMAP 所“保”的是“测地距离”。直接理解就是把一张纸摊平，保证“沿着流型”上的点的相对距离是不变的。

9.4 非线性降维方法 LLE

9.4.1 LLE 问题引入

既然我们已经在 ISOMAP 引入了样本分布在高维流型的思想，下面我们猜想真实的高维数据应当是集中分布在一个潜在的平滑流型上，如果我们“进入”这个“流型”的空间，实际上是一个更低维的空间。

这一段话说得很绕，不过道理很简单，还是以“瑞士卷”数据集为例子。瑞士卷数据集在三维空间之中是一个流型，但是仔细想想这个瑞士卷本质上是几维的？二维！再好比，手边的一张 A4 纸，随便卷一卷，我们认为 A4 纸其实还是二维的……因此我们认为流型空间的维度是较低的。

既然上面的说法是成立的，LLE 希望找到高维数据在流型空间内的坐标。LLE 算法采用的基本思路和 ISOMAP 的第一步很相似，是使用一个样本点周围的近邻将这个样本点近似线性表出。

9.4.2 LLE 算法推导

LLE 的输入是高维空间内部的一组样本点：

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_N \end{bmatrix} \in \mathcal{R}^{D \times N}$$

LLE 期望的输出是低维空间内的一组降维后的样本点：

$$\mathbf{Y} = \begin{bmatrix} \mathbf{y}_1 & \mathbf{y}_2 & \cdots & \mathbf{y}_N \end{bmatrix} \in \mathcal{R}^{d \times N}$$

上式之中 \mathbf{x}_i 与 \mathbf{y}_i 一一对应。

LLE 的第一步：对于样本 \mathbf{x}_i ，找到其 K 近邻。然后找到一组参数 w_{ij} ，将样本 \mathbf{x}_i 使用其近邻 $\{\mathbf{x}_1^{(i)}, \mathbf{x}_2^{(i)}, \dots, \mathbf{x}_K^{(i)}\}$ 以最小均方误差线性表出：

$$\xi = \sum_{i=1}^N \left\| \mathbf{x}_i - \sum_{j=1}^K w_{ij} \mathbf{x}_j^{(i)} \right\|_2^2$$

$$\text{s.t. } \sum_{j=1}^K w_{ij} = 1$$

我们为了对于上式进行推导，需要进行矩阵化：

$$\begin{aligned}\xi &= \sum_{i=1}^N \left\| \mathbf{x}_i - \sum_{j=1}^K w_{ij} \mathbf{x}_j^{(i)} \right\|_2^2 \\ \xi &= \sum_{i=1}^N \left\| \sum_{j=1}^K w_{ij} (\mathbf{x}_i - \mathbf{x}_j^{(i)}) \right\|_2^2 \\ \xi &= \sum_{i=1}^N \left\| (\mathbf{X}_i - \mathbf{N}_i) \cdot \mathbf{w}_i \right\|_2^2 \\ \mathbf{X}_i &= \begin{bmatrix} \mathbf{x}_i & \mathbf{x}_i & \cdots & \mathbf{x}_i \end{bmatrix} \in \mathcal{R}^{D \times K} \\ \mathbf{N}_i &= \begin{bmatrix} \mathbf{x}_1^{(i)} & \mathbf{x}_2^{(i)} & \cdots & \mathbf{x}_K^{(i)} \end{bmatrix} \in \mathcal{R}^{D \times K} \\ \mathbf{w}_i &= \begin{bmatrix} w_{i1} & w_{i2} & \cdots & w_{iK} \end{bmatrix}^T \in \mathcal{R}^{K \times 1}\end{aligned}$$

于是进一步推导：

$$\xi = \sum_{i=1}^N \mathbf{w}_i^T (\mathbf{X}_i - \mathbf{N}_i)^T (\mathbf{X}_i - \mathbf{N}_i) \mathbf{w}_i^T$$

假设有如下记法：

$$\begin{aligned}\mathbf{S}_i &= (\mathbf{X}_i - \mathbf{N}_i)^T (\mathbf{X}_i - \mathbf{N}_i) \\ \mathbf{l} &= \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}^T \in \mathcal{R}^{K \times 1}\end{aligned}$$

那么：

$$\xi = \sum_{i=1}^N \mathbf{w}_i^T \mathbf{S}_i \mathbf{w}_i$$

于是优化问题变为：

$$\begin{aligned}\hat{\mathbf{w}}_i &= \arg \min_{\mathbf{w}_i} \sum_{i=1}^N \mathbf{w}_i^T \mathbf{S}_i \mathbf{w}_i \\ \text{s.t. } \mathbf{w}_i^T \mathbf{l} &= 1, \forall \text{ integer } i \in [1, N]\end{aligned}$$

为了求解上述优化问题，构建拉氏算子：

$$\begin{aligned}\mathcal{L} &= \xi - \sum_{i=1}^N \lambda_i (\mathbf{w}_i^T \mathbf{l} - 1) \\ \frac{\partial \mathcal{L}}{\partial \mathbf{w}_i} &= 2 \mathbf{S}_i \mathbf{w}_i - \lambda_i \mathbf{l} = \mathbf{0}\end{aligned}$$

$$\mathbf{w}_i = \frac{1}{2} \lambda_i \mathbf{S}_i^{-1} \mathbf{l}$$

带入 $\mathbf{w}_i^T \mathbf{l} = 1$ 得到:

$$\lambda_i = \frac{2}{\mathbf{l}^T \mathbf{S}_i^{-1} \mathbf{l}}$$

于是:

$$\mathbf{w}_i = \frac{\mathbf{S}_i^{-1} \mathbf{l}}{\mathbf{l}^T \mathbf{S}_i^{-1} \mathbf{l}}$$

LLE 的第一步至此结束!

LLE 的第二步: 保存好上面求出的 \mathbf{w}_i , 找到一组中心化且协方差标准化的低维样本空间之中的 \mathbf{Y} , 使得如下优化目标成立:

$$\begin{aligned} \hat{\mathbf{y}}_i &= \arg \min_{\mathbf{y}_i} \sum_{i=1}^N \|\mathbf{y}_i - \sum_{j=1}^K w_{ij} \mathbf{y}_j^{(i)}\|_2^2 \\ \text{s.t. } & \sum_{i=1}^N \mathbf{y}_i = \mathbf{0}; \quad \frac{1}{N} \sum_{i=1}^N \mathbf{y}_i \mathbf{y}_i^T = \mathbf{I} \end{aligned}$$

这里解释一下上述两个条件。事实上仅仅满足保测地距条件并不能够固定唯一的一个 \mathbf{Y} , 但是满足上面的中心化和标准化条件就可以唯一确定一个最优的 $\hat{\mathbf{Y}}$ 。因为我们认为这一组 $\hat{\mathbf{Y}}$ 中心在原点, 并且样本在空间各个维度上均匀分布 (方差相等)。

如果不满足上面两个条件, 求出的 \mathbf{Y} 应该是在 $\hat{\mathbf{Y}}$ 随意平移 + 旋转的结果。
下面我们继续推导:

$$\xi_y = \sum_{i=1}^N \|\mathbf{y}_i - \sum_{j=1}^K w_{ij} \mathbf{y}_j^{(i)}\|_2^2$$

构造 $\mathbf{W}' \in \mathcal{R}^{N \times N}$, 其中每个元素 w'_{rs} 是:

$$w'_{rs} = \begin{cases} w_{rs} & \mathbf{y}_s \text{是} \mathbf{y}_r \text{的近邻} \\ 0 & \text{otherwise} \end{cases}$$

上式之中 w_{rs} 就是 LLE 算法第一步之中计算得到的那个 \mathbf{w}_r 向量之中的第 s 个分量。同时引入如下记法:

$$\mathbf{e}_i = [0 \ 0 \ \dots \ 0 \ 1 \ 0 \ \dots \ 0]^T \in \mathcal{R}^{N \times 1}$$

就是只有第 i 个分量是 1, 其他 $N-1$ 个分量都是 0。进一步推导:

$$\xi_y = \sum_{i=1}^N \|\mathbf{Y}(\mathbf{e}_i - \mathbf{w}'_i)\|_2^2$$

上式之中 \mathbf{w}'_i 是矩阵 \mathbf{W}' 的第 i 列，于是进一步有：

$$\xi_y = \sum_{i=1}^N (\mathbf{e}_i - \mathbf{w}'_i)^T \mathbf{Y}^T \mathbf{Y} (\mathbf{e}_i - \mathbf{w}'_i)$$

下面给出一个引理：

- **Lemma:** 对于任意 $\forall \mathbf{A} \in \mathcal{R}^{m \times n}$:

$$\begin{aligned} \mathbf{A} &= \begin{bmatrix} \mathbf{a}_1^{(1)} & \mathbf{a}_2^{(1)} & \cdots & \mathbf{a}_n^{(1)} \end{bmatrix} \\ \mathbf{A} &= \begin{bmatrix} \mathbf{a}_1^{(2)} & \mathbf{a}_2^{(2)} & \cdots & \mathbf{a}_m^{(2)} \end{bmatrix}^T \\ \mathbf{a}_i^{(1)} &\in \mathcal{R}^{m \times 1}, \forall \text{ integer } i \in [1, n]; \quad \mathbf{a}_j^{(2)} \in \mathcal{R}^{n \times 1}, \forall \text{ integer } j \in [1, m] \end{aligned}$$

(也就是 $\mathbf{a}_i^{(1)}$ 是 \mathbf{A} 的列向量； $\mathbf{a}_j^{(2)}$ 是 \mathbf{A} 的行向量)

有如下式子成立：

$$\sum_{i=1}^m \sum_{j=1}^n a_{ij}^2 = \sum_{i=1}^n \mathbf{a}_i^{(1)T} \mathbf{a}_i^{(1)} = \sum_{j=1}^m \mathbf{a}_j^{(2)T} \mathbf{a}_j^{(2)} = \text{tr}(\mathbf{A}^T \mathbf{A}) = \text{tr}(\mathbf{A} \mathbf{A}^T)$$

这个引理的证明很简单，一眼法。把这个引理应用在上面的推导之中：

$$\begin{aligned} \xi_y &= \sum_{i=1}^N (\mathbf{e}_i - \mathbf{w}'_i)^T \mathbf{Y}^T \mathbf{Y} (\mathbf{e}_i - \mathbf{w}'_i) \\ \xi_y &= \text{tr} [\mathbf{Y} (\mathbf{I} - \mathbf{W}') (\mathbf{I} - \mathbf{W}')^T \mathbf{Y}^T] \end{aligned}$$

如果引入如下记法：

$$\mathbf{M} = (\mathbf{I} - \mathbf{W}') (\mathbf{I} - \mathbf{W}')^T$$

那么：

$$\xi_y = \text{tr} (\mathbf{Y} \mathbf{M} \mathbf{Y}^T)$$

这个式子老演员了，直接由 Rayleigh-Ritz 定理（广义瑞利商）知道， \mathbf{Y} 取为 \mathbf{M} 的前 d 小的特征值对应的特征向量即可。

10 非度量方法与决策树

10.1 非度量方法引言

非度量方法主要指的是：样本的特征无法被一组实数数据结构表出时使用的模式识别方法。这是很好理解的，例如我们希望描述一个苹果，选取了

三种特征（颜色，光泽，甜味水平）。“颜色”维度存在两种取值：“红”与“青”；“光泽”维度存在两种取值：“有光泽”和“皱巴巴”；甜味水平存在三种取值：“甜”、“一般”和“不甜”。在这种情形下，虽然也有将三种特征量化的方法：例如构建一组 one-hot 向量对三个特征进行描述。不过也可以直接使用这样的非度量数据。

非度量数据的特点是：取值一般离散、没有自然语义下的相似性概念，甚至可能没有次序关系（例如水果的颜色就不存在相似性度量、也不存在次序关系）。

10.2 不纯度度量方法

10.2.1 不纯度度量方法在决策树中的作用

不纯度度量方法在决策树中用于衡量：使用某一个属性对样本集分类是不是最好的选择。

10.2.2 符号统一

为了使得如下论述之中的符号是一致的，首先进行符号的统一化。

假设我们的决策树面对的是一个 $|\mathcal{Y}|$ 分类问题，也就是说，对于样本集 D 之中的每一个样本，标签都是 $k = 1, 2, \dots, |\mathcal{Y}|$ 之中的某一个。

假设我们现在讨论的离散属性是 a ，这个 a 属性一共有 V 种取值，分别为 $\{a^1, a^2, \dots, a^V\}$ 。如果我们采用这个离散属性对于样本集 D 进行划分，显然一共会产生 V 个分支，其中的第 v 个分支包含了样本集 D 之中所有在 a 属性上取值为 a^v 的样本子集，这个子集我们记为 D^v 。

我们使用 $|\cdot|$ 算符表示集合之中元素的数量，例如 $|D|$ 是样本集 D 的大小。

10.2.3 信息增益 (ID3 决策树)

首先给出信息熵 (information entropy) 的定义。假设当前样本集合 D 之中的第 k 类样本所占的比例是 p_k ，则针对样本集 D 的信息熵定义为：

$$Ent(D) = - \sum_{k=1}^{|\mathcal{Y}|} p_k \log_2(p_k)$$

一般来说， $Ent(D)$ 的值越小，说明样本集 D 的纯度越高。

于是按照符号统一的定义，对于当前按照 a 属性划分的节点计算出的信息

增益就是：

$$Gain(D, a) = Ent(D) - \sum_{v=1}^V \frac{|D^v|}{|D|} Ent(D^v)$$

一般会选择信息增益最大的属性进行划分，也就是：

$$\hat{a} = \arg \max_a Gain(D, a)$$

这个准则被 ID3 决策树算法采用。

10.2.4 增益率 (C4.5 决策树)

采用信息增益的衡量方式实际上对于属性取值数量 V 更大的属性具有喜好，原因是很简单的，因为属性的取值数量 V 越多，说明每一种取值之中包含的样本量也就越少（抽屉原理，抽屉多的话，每个抽屉里球的数量就期望性地变少）。为了对于这种倾向进行平衡，进一步引入了增益率准则。

首先给出属性的熵 $IV(a)$ ：

$$IV(a) = - \sum_{v=1}^V \frac{|D^v|}{|D|} \log_2 \left(\frac{|D^v|}{|D|} \right)$$

进而给出增益率：

$$Gain_ratio(D, a) = \frac{Gain(D, a)}{IV(a)}$$

一般会选择增益率最大的属性进行划分，也就是：

$$\hat{a} = \arg \max_a Gain_ratio(D, a)$$

这个准则被 C4.5 决策树算法采用。

10.2.5 基尼系数准则 (CART 决策树)

基尼系数的含义是：从数据集 D 中随机抽取两个样本，样本的类别不同的概率，于是数学形式是：

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

基尼系数越小说明样本集纯度越高。

进一步引入针对属性 a 的基尼系数准则：

$$Gini_index(D, a) = \sum_{v=1}^V \frac{|D^v|}{|D|} Gini(D^v)$$

一般会选择基尼系数准则最小的属性进行划分，也就是：

$$\hat{a} = \arg \min_a Gini_index(D, a)$$

这个准则被 CART 决策树算法采用。

10.3 决策树模型基本算法

10.3.1 递归建树算法

如算法1所示。

10.3.2 分支停止条件

上面建树的过程中的停止条件总共是三种情况：第一，如果此节点处样本集已经是同一类别的全纯样本，停止；第二，如果已经没有属性能够使用了，停止；第三，如果属性 \hat{a} 的某一个分支 \hat{a}^v 取值下的样本集为空，停止。

这样的停止条件实际上是要求整个决策树算法走到不能走为止，往往会导致过拟合。为了避免过拟合，采用的方法可以是比较朴素的方法例如：某个节点的样本集数量小于全部训练样本数的 $\theta\%$ 时停止；叶子节点的总数大于 N 时强制停止，等。

还有一些做法是用复杂度换准确率，例如有一种做法是基于优化函数来判定分支停止的：

$$\alpha \cdot size + \sum_{\text{叶子节点}} Ent(D)$$

即在所有节点的纯度与整个树的大小之间以 α 系数做平衡。如果上述指标达到最小、再继续分支会导致上述指标变大，那么就停止分支。

10.3.3 剪枝算法

剪枝算法主要分为两种，预剪枝和后剪枝。下面分别介绍两种剪枝方法：

Algorithm 1 递归建树算法 $TreeGenerate(D, A)$

Require: 训练集: $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$; 属性集: $A = \{a_1, a_2, \dots, a_d\}$

Ensure: 以当前节点 node 为根节点的一棵决策树

- 1: 生成节点 node
- 2: **if** D 中样本都是同一类别 C **then**
- 3: 将 node 标记为 C 类叶子节点; **return**
- 4: **end if**
- 5: **if** $A = \emptyset$ **OR** D 中样本在 A 上取值相同 **then**
- 6: 将 node 标记为叶子节点, 类别标记为 D 中样本数最多的类; **return**
- 7: **end if**
- 8: 从 A 之中按照**不纯度度量方法**选择最优划分属性 \hat{a} ;
- 9: **for** \hat{a} 的每一种取值 \hat{a}^v **do**
- 10: 为 node 新产生一个分支; 令 D_v 表示 D 中在 \hat{a} 上取值为 \hat{a}^v 的子集;
- 11: **if** D_v 为空 **then**
- 12: 将分支节点标记为叶子节点, 其类别标记为 D 中样本最多的类;
- 13: **return**
- 14: **else**
- 15: 递归调用 $TreeGenerate(D_v, A - \{\hat{a}\})$ 生成分支节点
- 16: **end if**
- 17: **end for**

- **预剪枝**: 预剪枝是指在决策树生成过程中, 对每个节点在使用 a 属性划分之前, 先使用验证集估计划分前后验证集准确率的变化, 如果划分后验证集准确率下降, 则不进行这个划分, 分支停止, 并将当前节点标记为叶子节点, 类别标记为当前样本集中最多的类别。
- **后剪枝**: 后剪枝是指在决策树生成之后, 自底向上对非叶子节点进行考察, 如果将这个非叶子节点替换为叶子节点后, 验证集准确率提升, 那么删掉这个分支, 并将当前节点标记为叶子节点, 类别标记为当前样本集中最多的类别。

10.3.4 缺失数据的处理

针对样本 D 和属性 a , 如果令 \tilde{D} 表示 D 在 a 属性上没有缺失 (全部完整) 的样本子集。沿用前面 “符号统一” 的规则, \tilde{D}^v 是 \tilde{D} 中 a 属性取值为 a^v 的样本子集, \tilde{D}_k 是 \tilde{D} 中类别标签为 k 的样本子集。为了对于缺失数据进行处理, 我们对 D 样本集中每一个样本 (\mathbf{x}, y) 引入权重 $w_{\mathbf{x}}$, 并定义:

$$\begin{aligned}\rho &= \frac{\sum_{\mathbf{x} \in \tilde{D}} w_{\mathbf{x}}}{\sum_{\mathbf{x} \in D} w_{\mathbf{x}}} \\ \tilde{p}_k &= \frac{\sum_{\mathbf{x} \in \tilde{D}_k} w_{\mathbf{x}}}{\sum_{\mathbf{x} \in \tilde{D}} w_{\mathbf{x}}} \\ \tilde{r}_v &= \frac{\sum_{\mathbf{x} \in \tilde{D}^v} w_{\mathbf{x}}}{\sum_{\mathbf{x} \in \tilde{D}} w_{\mathbf{x}}}\end{aligned}$$

对于样本集 D 和属性 a , 上式中的 ρ 代表无缺失样本的比例, \tilde{p}_k 代表无缺失样本中第 k 类的占比, \tilde{r}_v 代表无缺失样本中属性 a 取值为 a^v 的占比。显然 $\sum_{k=1}^{|\mathcal{Y}|} \tilde{p}_k = 1$ 且 $\sum_{v=1}^V \tilde{r}_v = 1$ 。

基于上述定义, 我们给出扩展后的信息增益:

$$\begin{aligned}Gain(D, a) &= \rho \cdot Gain(\tilde{D}, a) \\ Gain(\tilde{D}, a) &= Ent(\tilde{D}) - \sum_{v=1}^V \tilde{r}_v Ent(\tilde{D}) \\ Ent(\tilde{D}) &= - \sum_{k=1}^{|\mathcal{Y}|} \tilde{p}_k \log_2(\tilde{p}_k)\end{aligned}$$

基于上述所有说明, 我们在构建决策树时, 如果样本 \mathbf{x} 在 a 属性上的取值是已知的, 直接正常划分这个样本, 且其权值在子节点之中保持为 $w_{\mathbf{x}}$ 。如

果样本 \mathbf{x} 在 a 属性上取值缺失，那么将这个样本同时划入所有子节点，并在各个子节点中的权重变更为 $\tilde{r}_v \cdot w_{\mathbf{x}}$ 。

从理解的角度来解释这样的做法就是：缺失数据可以按照无缺失数据在属性 a 上的分布进行概率加权。这种方法是 C4.5 决策树采用的方法，CART 决策树采用了一种不同的“替代分支法”和“虚拟值法”处理缺失数据。这种方法在 CART 决策树算法之中介绍。

10.4 CART 决策树算法

CART 决策树在建树时采用基尼系数准则。原始的 CART 算法没有剪枝，改进的 CART 则进行剪枝。针对缺失数据的处理方法是“替代分支法”或“虚拟值法”，下面详细介绍这种方法。

- **虚拟值法**：用这个节点缺失的属性的最大似然值补全缺失。
- **替代分支法**：首先仅仅按照无缺失样本集构建针对 a 属性的分支，称为“主分支”。因为 a 这个属性之中存在缺失，我们就从别的属性之中寻找替代。衡量替代次序的方式是衡量分类相似度，分类相似度是指采用某个属性 b 分支后，每个子节点之中包含的样本与使用 a 分支后子节点包含的样本的相同比例的加权平均。将主分支直接使用最优的替代分支进行替代即可。

10.5 ID3 决策树算法

CART 决策树在建树时采用信息增益。原始的 ID3 算法没有剪枝、也没有缺失值处理。

10.6 C4.5 决策树算法

C4.5 决策树在建树时采用信息增益率。有剪枝、也有缺失数据处理，处理方法就是上面讲过的概率加权法。

11 集成学习 Ensemble

11.1 集成学习问题引入

前面的所有章节之中我们讨论了很多分类算法，在实践中我们发现把多个模型或者分类器结合在一起的效果比单独使用一个的效果更好。这样的方法称为集成学习，也成为多分类器方法等等。

11.2 基于训练样本的分类器构造

11.2.1 Bagging 方法

假设训练集之中有 N 个样本，每次随机抽取 n 个样本用于训练一个分类器 h ，这样的过程重复 L 次，于是得到了 L 个分类器 $\{h_1, h_2, \dots, h_L\}$ 。随机抽取 n 个样本这个过程称为 Bootstrap。这些分类器的融合方法是投票。也就是说，对于一个测试样本 \mathbf{x} ，其类别是这 L 个分类器分类结果进行投票，票数最多的类别。

11.2.2 Boosting 方法的代表 Adaboost

Boosting 算法有很多种，我们介绍最多使用的方法 Adaboost，算法如下3

Algorithm 2 Adaboost 算法 $Adaboost(D, \mathcal{L}, T)$

Require: 训练集: $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$; 基学习算法: \mathcal{L} ;

训练轮数: T

Ensure: 分类器: $H(\mathbf{x}) = sign\left(\sum_{t=1}^T \alpha_t h_t(\mathbf{x})\right)$

1: 对于所有样本加权，并将第 i 个样本的权重都初始化为 $w_1(i) = \frac{1}{m}$

2: **for** $t = 1, 2, \dots, T$ **do**

3: $p_t(i) = \frac{w_t(i)}{\sum_{i=1}^m w_t(i)}$

4: 按照 $p_t(i)$ 的概率对每一个样本进行采样，生成样本子集 D_t

5: 使用 D_t 训练基学习算法得到 $h_t = \mathcal{L}(D_t)$

6: 对 D_t 中的样本测试，计算 $\epsilon_t = \sum_{i=1}^m p_t(i) \cdot \mathbf{I}[h_t(\mathbf{x}_i) \neq y_i]$

7: $\alpha_t = \frac{1}{2} \ln \left[\frac{1-\epsilon_t}{\epsilon_t} \right]$

8: $w_{t+1}(i) = w_t(i) \cdot \begin{cases} e^{-\alpha_t} & \mathbf{I}[h_t(\mathbf{x}_i) \neq y_i] = 1 \\ e^{\alpha_t} & \mathbf{I}[h_t(\mathbf{x}_i) \neq y_i] = 0 \end{cases}$

9: **end for**

11.2.3 二分类问题的 Adaboost 推导

对于一个 $y_i = \{-1, 1\}$ 的二分类问题，经过上述的 T 轮迭代，得到的样本概率分布是：

$$p_{T+1}(i) = p_1(i) \cdot \frac{e^{-y_i \sum_{j=1}^T \alpha_j h_j(\mathbf{x}_i)}}{\prod_{t=1}^T Z_t}$$

上式之中的 Z_t 是第 t 轮迭代后的归一化因子，有如下表示：

$$Z_t = \sum_{i=1}^m p_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)}$$

然后引入如下记法：

$$\begin{aligned}\boldsymbol{\alpha} &= \begin{bmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_T \end{bmatrix}^T \\ \mathbf{h} &= \begin{bmatrix} h_1 & h_2 & \cdots & h_T \end{bmatrix}^T\end{aligned}$$

将上式以及 $p_1(i) = \frac{1}{m}$ 带入 $p_{T+1}(i)$ 的计算式中得到：

$$p_{T+1}(i) = \frac{e^{-y_i \boldsymbol{\alpha}^T \mathbf{h}(\mathbf{x}_i)}}{m \prod_{t=1}^T Z_t}$$

由于这是归一化之后的概率，必须满足 $\sum_{i=1}^m p_{T+1}(i) = 1$ ，因此带入上式有：

$$\frac{\sum_{i=1}^m e^{-y_i \boldsymbol{\alpha}^T \mathbf{h}(\mathbf{x}_i)}}{m \prod_{t=1}^T Z_t} = 1$$

$$\frac{1}{N} \sum_{i=1}^m e^{-y_i \boldsymbol{\alpha}^T \mathbf{h}(\mathbf{x}_i)} = \prod_{t=1}^T Z_t$$

对于所有的训练集样本的判决应当有如下表示：

$$error = \frac{1}{m} \sum_{i=1}^m \mathbf{I}[h_t(\mathbf{x}_i) \neq y_i]$$

在上面引入的表述的基础上，我们现在来讨论随着训练轮次逐渐上升时，Adaboost 在训练集上的错误率变化情况；并最终推导出为什么系数 $\alpha_t = \frac{1}{2} \ln \left[\frac{1-\epsilon_t}{\epsilon_t} \right]$

首先讨论一下 $\mathbf{I}[h_t(\mathbf{x}_i) \neq y_i]$ 与 $e^{-y_i \boldsymbol{\alpha}^T \mathbf{h}(\mathbf{x}_i)}$ 的大小关系。我们给出 $y_i \boldsymbol{\alpha}^T \mathbf{h}(\mathbf{x}_i)$ 的取值：

$$y_i \boldsymbol{\alpha}^T \mathbf{h}(\mathbf{x}_i) = \begin{cases} c \geq 0 & \mathbf{I}[h_t(\mathbf{x}_i) \neq y_i] = 1 \\ -c \leq 0 & \mathbf{I}[h_t(\mathbf{x}_i) \neq y_i] = 0 \end{cases}$$

于是：

$$e^{-y_i \alpha^T h(\mathbf{x}_i)} = \begin{cases} e^{-c} \geq 0 & \mathbf{I}[h_t(\mathbf{x}_i) \neq y_i] = 1 \\ e^c \geq 1 & \mathbf{I}[h_t(\mathbf{x}_i) \neq y_i] = 0 \end{cases}$$

所以： $\mathbf{I}[h_t(\mathbf{x}_i) \neq y_i] \leq e^{-y_i \alpha^T h(\mathbf{x}_i)}$ 必然恒成立。于是下式成立：

$$\text{error} = \frac{1}{m} \sum_{i=1}^m \mathbf{I}[h_t(\mathbf{x}_i) \neq y_i] \leq \frac{1}{N} \sum_{i=1}^m e^{-y_i \alpha^T h(\mathbf{x}_i)} = \prod_{t=1}^T Z_t$$

那么为了最小化错误率，可以最小化其上界（这个思想类似于 EM 算法）。既然如此，如何找到一组分类器线性组合的系数 α 使得 $\prod_{t=1}^T Z_t$ 最小就是我们的关键问题。不妨对于 $\prod_{t=1}^T Z_t$ 求导：

$$\frac{\partial \prod_{t=1}^T Z_t}{\partial \alpha} = \mathbf{0}$$

这个计算显然十分困难，于是我们采用一种贪婪的思路，就是使得 $\prod_{t=1}^T Z_t$ 之中的每一项都是针对于当前迭代轮次 α_t 最小的，也就是令下式成立：

$$\frac{\partial Z_t}{\partial \alpha_t} = 0$$

于是带入 Z_t 的表达式得到：

$$\frac{\partial Z_t}{\partial \alpha_t} = - \sum_{i=1}^m y_i h_t(\mathbf{x}_i) p_t(i) e^{-\alpha_t y_i h_t(\mathbf{x}_i)} = 0$$

我们研究上式中 $y_i h_t(\mathbf{x}_i)$ ：

$$y_i h_t(\mathbf{x}_i) = \begin{cases} 1 & \mathbf{I}[h_t(\mathbf{x}_i) \neq y_i] = 1 \\ -1 & \mathbf{I}[h_t(\mathbf{x}_i) \neq y_i] = 0 \end{cases}$$

带入得到：

$$\begin{aligned} - \sum_{i \in \text{Correct}} p_t(i) e^{-\alpha_t} + \sum_{i \in \text{False}} p_t(i) e^{\alpha_t} &= 0 \\ -\epsilon_t e^{-\alpha_t} + (1 - \epsilon_t) e^{\alpha_t} &= 0 \\ \alpha_t &= \frac{1}{2} \ln \left[\frac{1 - \epsilon_t}{\epsilon_t} \right] \end{aligned}$$

于是进一步可以带入到 Z_t 式子中得到：

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

如果令 $\epsilon_t = \frac{1}{2} - \gamma_t$, $\gamma_t \in [-\frac{1}{2}, \frac{1}{2}]$, 那么:

$$Z_t = \sqrt{1 - 4\gamma_t^2}$$

$$\text{error} \leq \prod_{t=1}^T Z_t = \prod_{t=1}^T \sqrt{1 - 4\gamma_t^2} \leq \prod_{t=1}^T 1 - 2\gamma_t^2 \leq \prod_{t=1}^T e^{-2\gamma_t^2} = e^{-2 \sum_{t=1}^T \gamma_t^2} \leq e^{-2T\gamma^2}$$

上式之中 $\gamma = \max\{\gamma_1, \gamma_2, \dots, \gamma_T\}$ 。

上面的式子意思是, 每迭代一次 (t 每 +1 一次), 训练集上的错误率都会以指数形式衰减。

11.3 基于样本特征的分类器构造

基于样本特征的分类器构造方法的含义是对于样本集的每个样本, 每次随机选 d 个特征训练分类器, 然后输出的分类器是这些分类器投票。传统方法是选样本, 但是基于特征的分类器构造是选特征。具体的算法之一是“随机子空间法”, 算法如下:

Algorithm 3 随机子空间算法 $\text{RandomSubspace}(D, \mathcal{L}, T)$

Require: 训练集: $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$; 基学习算法: \mathcal{L} ;
训练轮数: L
Ensure: 分类器: $H(\mathbf{x}) = \arg \max_{k \in |\mathcal{Y}|} \sum_{l=1}^L \mathcal{I}[h_l(\mathbf{x}) = y]$

- 1: **for** $l = 1, 2, \dots, L$ **do**
- 2: 随便选 d 个特征, 应用于整个样本集 D , 从而构建样本子集 S_l
- 3: 用 S_l 训练分类器 $h_l = \mathcal{L}(S_l)$
- 4: **end for**

11.4 分类器输出的融合

11.4.1 决策层输出

如果基分类器的输出是样本类别标号的时候采用决策层输出。最常用的就是投票。

11.4.2 排序层输出

基分类器的输出是输入样本可能属于的类别列表的时候, 使用排序层输出, 例如下表所示的输出结果:

分类器	h_1	h_2	h_3	h_4	h_5	h_6
排第一的类别号	1	1	3	1	2	2
排第二的类别号	2	2	2	3	1	3
排第三的类别号	3	3	1	2	3	1

假如我们按照第一名得 3 分，第二名得 2 分，第三名得 1 分的规则计算，第一类共得到 13 分。第二类共得到 14 分。第三类共得到 10 分。因此，最终判决样本 x 为第二类。

Fig 16. 集成学习的排序层输出

11.4.3 度量层输出

基分类器的输出是输入样本可能属于的类别候选号以及每个候选类别的可能性（似然值）的时候，使用排序层输出，例如下表所示的输出结果：

分类器	h_1	h_2	h_3	h_4	h_5	h_6
排第一的类别号	1, 0. 8	1, 0. 8	3, 0. 5	1, 0. 5	2, 0. 9	2, 0. 6
排第二的类别号	2, 0. 1	2, 0. 15	2, 0. 3	3, 0. 05	1, 0. 3	3, 0. 2
排第三的类别号	3, 0. 1	3, 0. 05	1, 0. 2	2, 0. 05	3, 0. 1	1, 0. 2

按照求和规则，第一类的相似性度量值: $M_1=0.8+0.8+0.2+0.5+0.3+0.2=2.8$;

第二类的相似性度量值 $M_2=0.1+0.15+0.3+0.05+0.9+0.6=2.1$;

第三类的相似性度量值 $M_3=0.1+0.8+0.2+0.5+0.1+0.2=1.9$;

因此，最终判决样本 x 为第一类。

Fig 17. 集成学习的度量层输出

11.5 集成学习可解释性

事实上很令人好奇的是为什么集成学习是有意义的。下面从三个简单的角度进行不太理论化的解释：

- **统计学意义上**：对于多种简单的分类器加权，每一种分类器本质上是针对这个分类问题的一种泛化；构建了多个简单分类器有一种：让每个分类器产生一种泛化，最后合并、统筹所有的泛化，就能够得到多维度意义上的泛化。

- **计算学意义上**: 由于最优分类函数的搜索空间很大，采用一个分类器往往是具有倾向性的落到某个局部最优值之中。而使用多个分类器进行组合，能够使得算法跳出各个分类器的局部最优、并共同寻找全局最优。
- **表示学意义上**: 统计学习理论表明，当分类器结构过于复杂时（即 VC 维很高），泛化性能很差。而集成学习采用很多个 VC 维小的分类器合并成大分类器的思想，使得算法简化的同时使用多个分类器也能够分类高维样本，从而实现 VC 维较低、泛化较好的情况下处理复杂样本，“四两拨千斤”。

12 聚类分析与无监督学习

12.1 聚类分析与无监督学习引言

聚类分析是什么？是无监督地在样本空间之中把相似的样本结合为一个“集体”。自然的我们会问，这个“集体”是什么意思？我认为可以把“集体”理解为一种“标签”，也因此聚类分析本质上就是一种无监督分类器。这与“物以类聚”这个成语有一曲同工之妙。把“集体”理解为一种“标签”也是合乎常理的，例如我的朋友们很多都是清本，那么我是清本的可能性也就很大（这个例子是 K-Means 算法的例子）。

12.2 K-Means 聚类方法

既然聚类是按照类别内部样本的相似程度来进行计算的，因此首先需要找到一种准则来衡量这种“类内相似性”。我们假设一共聚类得到的类别数必须是 C 类，每一类的样本集合我们记为 Γ_i ，那么我们的衡量方法如下式所示：

$$J_e = \sum_{i=1}^C \sum_{x_j^{(i)} \in \Gamma_i} \|x_j^{(i)} - m_i\|_2^2$$

$$m_i = \frac{1}{|\Gamma_i|} \sum_{j=1}^{|\Gamma_i|} x_j^{(i)}$$

上式之中 $x_j^{(i)}$ 表示 Γ_i 样本集合之中的第 j 个样本， $|\Gamma_i|$ 表示 Γ_i 样本集合的大小。

K-Means 算法的优化目标就是使得 J_e 最小化。

K-Means 算法如下所示：

- 第一步：把原始样本随机划分为 C 类
- 第二步：从全部样本集中任选一个样本 \mathbf{x} ，假设这个样本是 Γ_i 集合中的
- 第三步：如果 $|\Gamma_i| = 1$ ，重新选择样本
- 第四步：对于所有的类别计算下式：

$$\rho_j = \begin{cases} \frac{|\Gamma_j|}{|\Gamma_j|+1} \|\mathbf{x} - \mathbf{m}_j\|_2^2 & j \neq i \\ \frac{|\Gamma_i|}{|\Gamma_i|-1} \|\mathbf{x} - \mathbf{m}_i\|_2^2 & j = i \end{cases} \quad \forall \text{integer } j \in [1, C]$$

- 第五步：如果对于任意的 j ，存在一个 k 使得 $\rho_k < \rho_j, \forall \text{integer } j \in [1, C], j \neq k$ ，那么把样本 \mathbf{x} 从 Γ_i 移动到 Γ_k 中去
- 第六步：重新计算 \mathbf{m}_i 与 \mathbf{m}_k 以及 J_e
- 第七步：如果连续迭代 L 次，聚类情况都不变，终止；否则回到第二步。

上面的步骤之中，我们省略了一步证明：只要存在一个 k 使得 $\rho_k < \rho_j, \forall \text{integer } j \in [1, C], j \neq k$ ，那么把样本 \mathbf{x} 从 Γ_i 移动到 Γ_k 中，一定会使 J_e 下降。我们下面进行证明：

$$\begin{aligned} \tilde{\mathbf{m}}_i &= \mathbf{m}_i + \frac{1}{|\Gamma_i|-1} (\mathbf{m}_i - \mathbf{x}) \\ \tilde{\mathbf{m}}_k &= \mathbf{m}_k + \frac{1}{|\Gamma_k|+1} (\mathbf{x} - \mathbf{m}_k) \\ \tilde{J}_e &= J_e - \frac{|\Gamma_i|}{|\Gamma_i|-1} \|\mathbf{x} - \mathbf{m}_i\|_2^2 + \frac{|\Gamma_k|}{|\Gamma_k|+1} \|\mathbf{x} - \mathbf{m}_k\|_2^2 \end{aligned}$$

证毕。

12.3 分级聚类方法

12.3.1 两类之间的相似性度量方法

总共有三种方法，如下所示：

- 最近距离:

$$\Delta(\Gamma_i, \Gamma_j) = \min_{\mathbf{x}^{(i)} \in \Gamma_i, \mathbf{x}^{(j)} \in \Gamma_j} Dist(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$$

- 最远距离:

$$\Delta(\Gamma_i, \Gamma_j) = \max_{\mathbf{x}^{(i)} \in \Gamma_i, \mathbf{x}^{(j)} \in \Gamma_j} Dist(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$$

- 均值距离:

$$\Delta(\Gamma_i, \Gamma_j) = \frac{1}{|\Gamma_i| \cdot |\Gamma_j|} \sum_{\mathbf{x}^{(i)} \in \Gamma_i} \sum_{\mathbf{x}^{(j)} \in \Gamma_j} Dist(\mathbf{x}^{(i)}, \mathbf{x}^{(j)})$$

12.3.2 分级聚类算法

- 第一步: 每个样本自成一类
- 第二步: 相似性最大的两个类聚为一类
- 第三步: 重复上述步骤直到聚类数为 C

12.4 谱聚类

给定样本集 $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$, 谱聚类本质上就是对于样本构建相似性图, 相似性图的邻接矩阵为 \mathbf{W} 。

$$\mathbf{W} = [w_{ij}] \in \mathcal{R}^{m \times m}$$

每个元素 w_{ij} 都是代表 \mathbf{x}_i 与 \mathbf{x}_j 之间的相似度, 常用的相似度定义是高斯相似度:

$$w_{ij} = \exp \left[-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma^2} \right]$$

上式之中, σ 是参数, 用于控制某一个元素的近邻影响范围。

再使用这个邻接矩阵构建图拉普拉斯矩阵 (可以归一化也可以不归一化, 是两种方法)。如果是归一化拉普拉斯矩阵, 那就是 $\tilde{\mathbf{L}}$, 未进行归一化则是 \mathbf{L} 。

$$\mathbf{L} = \mathbf{D} - \mathbf{W}$$

$$\tilde{\mathbf{L}} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{W}$$

$$\mathbf{D} = diag(d_1, d_2, \dots, d_m)$$

$$d_i = \sum_{j=1}^m w_{ij}$$

计算拉普拉斯矩阵的前 C 个特征值(这个 C 就是聚类数)为 $\{\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_C\}$, 构成矩阵:

$$\mathbf{U} = [\mathbf{u}_1 \quad \mathbf{u}_2 \quad \dots \quad \mathbf{u}_C] \in \mathcal{R}^{m \times C}$$

假设

$$\mathbf{y}_k \in \mathcal{R}^{C \times 1}, \forall i = 1, 2, \dots, m$$

那么谱聚类就是对于这一族 $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_m\}$ 进行的 K-Means 聚类。

13 计算学习理论

14 概率图模型

15 强化学习理论

16 后记

本文还留了计算学习理论、概率图模型和强化学习理论三个部分，本人对这些算法推导仍然存在一定的生疏或不明白的地方。待后面补齐。

本科期间学到机器学习理论时，我就曾经设想过将所有的算法自己手动推导一遍，并加以整理。时至大四的最后几天，在其他朋友们都在拍摄毕业照、享受青春时，我因准备研究生课程模式识别的考试为机，偶得此静心无扰的清净时光，将曾经的设想付诸实现。从提笔到落稿，不过短短一周不到。本文初成，激动之情溢于言表。我读过的机器学习著作时刻提醒着我，我所撰写的文章仅仅是拙笔浅墨，但这是我自己的亲手推出的，意义还是不同的。

数学是什么？很多人说它是工具，也有人将其捧上神坛、认其为科学之母。我认为数学不仅仅是一种方法、但也并非神明，数学是人类思想的总结与凝练。若是对应到中华文化之中，或许使用道家思想的“无用”和“道”最为合适于描述数学。

算者无用，道其中也。

——EscapeTHU

2022 年 06 月 15 日于清华园