机器学习

Jianqi Huang

November 2022

1 导论

1.1 什么是机器学习

HerBert A.Simon:“若一个系统能够通过执行某个过程改进它的性能，这就是学习.”

定义 1.1 (机器学习)**.** 一门能够让编程计算机从数据中学习的计算机技术. **形式定义**: P 评判计算机程序在某任务类 T 上的性能，若一个程序通过经验 E 在 T 中获得性能的改善，就说这个通过 T 和 P 进行了学习. 学习的本质就是模仿和创新.

例子 1.1. 乌鸦吃核桃: 红绿灯观察到的现象，称为数据，为了实现任务，观测数据，通过规律来完成得更好. 而有可能学的好，又可能学的不好.

有一些问题是难以预判的. 算法能够学到规律并将规律应用在新问题的技术.

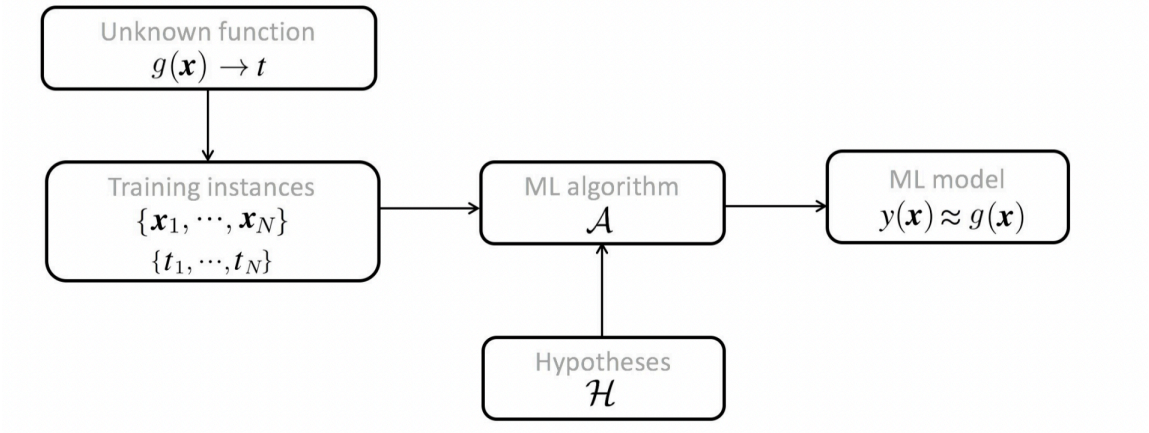


图 1: 机器学习框架

1.2 机器学习分类

定义 1.2 (标记 (label))**.** 一个标记是对学习样本的潜在规律的识别过程中的信息处理. 标记所在的空间为标记空间/输出空间.

我们根据标记的不同，可将机器学习的问题分为:

- 离散标记取值时候是分类问题.
- 连续标记取值时候是回归问题.

对于是否用到标记还可以将学习任务划分为:

- 监督学习：分类、回归.
- 无监督学习
- 半监督学习：两者进行结合，先以无监督进行学习（聚类）

再根据标记信息类型还可以将监督问题划分为:

- 分类问题：标记是离散值;
- 回归问题：标记是连续值;

若没有标签，则会使用预测性的，对数据的要求降低而预测性降低.（存在一个 trade off）

定义 1.3 (输入和输出空间)**.** 在监督学习中将输入与输出的所有取值的集合分别称为输入空间 *(input space)* 与输出空间（*output space*）而对于输入输出空间是否是同一个空间没有要求. 可以相同也可以不同.

每个具体的输入是一个实例，通常以特征向量来表示 (feature vector)，所有特征向量的空间就是特征空间. 特征向量写作 $x = (x_1, x_2, \cdots x_n)$ ，其中的 x_i 就是特征向量的第 i 个特征. 而因此训练集的表示:

$$T = \{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$$

其他的训练策略

- 强化学习：以试错方法训练;
- 迁移学习：将其他领域算法经验应用到当前问题上. 加速训练过程.

定义 1.4 (假设空间)**.** 监督学习就是实现从输入到输出的一个映射，这样的映射通过模型表示，学习的目的在于找到这样的一个模型. 模型属于输入到输出的**映射的集合**，这个集合就是假设空间 *(hypothesis space)*. 假设空间确定了学习范围. 数学表示模型 f 与假设空间 $\mathcal{F}$ 的关系：

$$f \in \mathcal{F}$$
$$\cup_i f \in \mathcal{F}$$

1.3 模型

机器学习三要素：方法 = 模型 + 策略 + 算法.

定义 1.5 (模型)**.** 也就是输入到机器中训练的数据集（包括训练集和验证集）然后通过训练得到模型（学习器）最后测试集测试. 模型的假设空间包含所有的条件概率分布或决策函数.

定义 1.6 (参数空间)**.** 对于 X 和 Y 是定义在 $\mathcal{X}$ 和 $\mathcal{Y}$ 的变量，这时 $\mathcal{F}$ 认为是由一个参数向量决定的函数族：

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

这时候参数取决于 n 维欧式空间 R^n 为参数空间.

定义 1.7. 概率模型与非概率模型或确定性模型，监督学习中，概率模型取条件概率形式 $P(y|x)$，非概率模型取 $y = f(x)$.

优化算法：需要对于一些难解的问题（求参）等问题进行优化求解. 对于一个机器学习需要最后考虑使用怎么样的计算方法求解模型.

1.3.1 生成模型与判别模型

监督学习的人物就是从数据中学习一个模型，利用这个模型，对给定的输入预测相应的输出. 这个模型一般是一个决策函数：

$$Y = f(X)$$

或者是一个条件分布函数：

$$P(Y|X)$$

而监督学习又可分为生成学习方法 (generative approach) 和判别方法 (discriminative approach). 所学习到的模型分别为生成模型 (generative model) 和判别模型 (discriminative model).
生成方法是由数据学习联合分布 $P(X, Y)$，之后求得条件概率分布 $P(Y|X)$ 作为预测模型.

$$P(Y|X) = \frac{P(X, Y)}{P(X)}$$

1.4 模型评估与选择

给定一个训练集合 $T = \{(x_1, y_1), (x_2, y_2), \cdots (x_n, y_n)\}$ 模型 $f(X)$ 关于训练集的平均损失称为经验风险，记为 R_{emp}

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

损失函数（策略）和风险函数：通过不同的损失函数可能最终得到的模型参数是不一致的．预测值与真实值之间的偏差用一个损失函数 (loss function) 来度量偏差程度．

定义 1.8 (经验风险最小化与结构风险最小化)**.** 经验风险最小化认为就是最优的模型．结构风险最小化为了防止过拟合而提出的策略．结构风险最小化等价于正则化．

1.4.1 训练误差与测试误差

训练误差 (training error) 就是平均损失:

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

其中 N 表示的是训练容量．
测试误差 (test error) 是模型 $Y = \hat{f}(X)$ 一个测试验证集的平均损失:

$$e_{test} = \frac{1}{N'} \sum_{i=1}^{N'} L(y_i, \hat{f}(x_i))$$

其中 N' 表示测试样本容量．

注释 1.1. 测试误差反映了学习方法对未知测试集的预测能力．一般将这个能力称为泛化能力 *(generalization ability)*

1.4.2 过拟合与模型选择 (model section)

定义 1.9. 过拟合：模型把样本学的过像，捕捉了数据的所有特征，所有样本的一般性质，导致泛化性能下降．进一步就可以使用正则技术、早停技术实现．**欠拟合**：就是过拟合的对立面，对一些特征的捕捉过少．

注释 1.2. 随着模型复杂度的上升，训练误差会不断下降，而测试误差会经历一个先下降后上升的过程．

定义 1.10 (奥卡姆剃刀定律)**.** 若有多个假设与观测值一致，选择最简单的一个．

定义 1.11 (No Free Lunch)**.** 当我们选择 A 算法时候，一定会在一些时候表现不好，一定不能所有囊括下来，一个算法也不能放之四海而皆准．

1.5 正则化及交叉验证

通过正则化进行模型选择是一个经典方法，正则化是结构风险最小化的策略．在经验风险上加入一个惩罚项 (penalty term)．模型越复杂，惩罚项越大．一般形式：

$$\min_{f\in\mathcal{F}} \frac{1}{N}\sum_{i=1}^{N} L(y_i, f(x_i)) + \lambda J(f)$$

注释 1.3. 正则化很好地体现了奥卡姆剃刀原理．

1.5.1 交叉验证 (cross validation)

对于样本量充足的模型选择问题，可以将数据集简单分为三部分：

- 训练集 (training set) 用于训练模型；
- 验证集 (validation set)：用于模型的选择；
- 测试集 (test set)：用于模型的评估．

注释 1.4. 交叉验证在于不是选出最佳的参数，而是对每一个可能的模型，用训练集最小化 *loss function* 的误差从而得到最佳参数后，运用验证集对繁华误差的评估来选择最好的模型．（先在一堆模型中选出最优参数，再一堆模型中选出最能泛化的模型）

1.6 模型评价

模型评估: 不同的机器学习有不同的评价指标，同一种机器学习任务也有不同的评价指标．每个指标的分类重点不同．比如回归的一个评价指标可以用上 RMSE．评估方法有不同的维度: 泛化性能、时间开销、存储开销、可解释性．

1.6.1 泛化误差

学习方法的泛化能力 (generalization ability) 是指由该方法学习到的对未知预测的能力．

$$R_{exp}(\hat{f}) = E_p[L(Y, \hat{f}(X))] = \int_{\chi\times\gamma} L(y, \hat{f}(x))P(x, y)dxdy$$

用泛化误差来表示一个泛化的能力．(泛化误差也表示所学习到的模型的一个风险期望)．本质上与测试误差/训练误差的形式是相同的．

1.6.2 偏差-方差分解

泛化偏差的构成：训练集 d 上，训练后的模型 $f_d(x)$.

- 对数据 x 的预测输出为 $f(\overline{x}) = E_d[f_d(x)]$;
- 验证集样本的真实值 y;
- 噪声，样本的标签值与真实值存在出入，设标签值为 y_d;
- 偏差：$bias = y - f(\overline{x})$
- 噪声：$\epsilon = y - y_d$
- 方差：$var = E_d[(f_d(x) - f(\overline{x}))^2]$
- 泛化误差：$E_d[y_d - f_d(x)^2]$

我们假定偏差 (bias) 是服从高斯分布（也就是说输入 x 是独立同分布的），方差的定义上看，预测输出与不同测试集差的离散程度.
进一步我们可以推导出泛化误差可分解为偏差、方差及噪声之和.

$$E_d[(y_d - f_d(x))^2] = \epsilon^2 + bias^2 + var$$

1.7 监督学习应用

监督学习从输入一个分类决策函数（分类器 classifier），分类器对于新的输入进行预测，称为分类.

混淆矩阵：对于一个预测与真实之间的对应情况的表示. 对于一个二分类问题一个常用的评价指标就是查准率和查全率. 查准率 (precision)

$$P = \frac{TP}{TP + FP}$$

查全率 (recall)

$$R = \frac{TP}{TP + FN}$$

ROC(receiveer operating characteristic curve) 曲线：表示的是受试者工作特征曲线. 曲线上的每一个点都是对同一信号刺激相同的感受性.
ROC 图的绘制：给定 $m+$ 个正例和 $m-$ 个负例，根据学习器预测结果对样例进行排序，将分类阈值设为每个样例的预测值，当前标记点坐标为, 当前若为真正例，则对应标记点的坐标为; 当前若为假正例，则对应标记点的坐标为, 然后用线段连接相邻点.
AUC 值：若某个学习器的 ROC 曲线被另一个学习器的曲线“包住”, 则后者性能优于前者；否则如果曲线交叉，可以根据 ROC 曲线下面积大小进行比较，也即 AUC 值.
再对于一些数据 test data 在实践测试中的表现能力.（数据划分为训练集和验证集）来求泛化误差，所谓的交叉验证.

2 线性模型

规定一个数据集合 $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N)\}$ 输入矩阵：$X = (x_1, x_2, \cdots, x_n)$ 输出矩阵 $Y = (y_1, y_2, \cdots y_n)$

模型的形式

$$f(x) = w_1 x_1 + w_2 x_2 + \cdots + w_d x_d + b$$

一般以向量形式表示

$$f(x) = \omega^T x + b$$

其中规定 $\omega = (w_1; w_2; \cdots; w_d)$，$\omega$ 和 b 学习之后，模型得到确定.

2.1 一般线性回归

参数向量 $w = (w_1, w_2, \cdots, w_d)^T$
模型输入：参数标记方法，我们将 $\hat{w} = (w, b)$ 归纳入 w 中. 相应的，X 数据中也要并上一个 $(m \times 1) = 1$ 的列向量. $X = (x_1, x_2, \cdots x_p, 1)^T$ 从而推出

$$y = f(x) = \omega^T x$$

线性回归试图去学习得到：$f(x_i) = w x_i + b$ 使得 $f(x_i) \simeq y_i$

回归模型的优点：

1. 形式简单，便于建模
2. 可解释性强
3. 非线性模型的基础：引入层级结构或高维映射

2.1.1 损失函数

一般常用的是平方损失函数作为性能度量手段.

$$L(w) = \sum_{i=1}^{N} \|w^T X_i - y_i\|_2^2 \tag{1}$$

优化算法：使用最小二乘法进行估计

- 对于欧式距离来说，能够找到最小的参数点.
- 优化平方损失函数一般采用最小二乘法

$$w^* = \arg\min_{\omega} \sum_{i=1}^{N} (f(x_i) - y_i)^2 \tag{2}$$

对(??) w 参数求偏导就可以得到：

$$\frac{\partial E_{\hat{w}}}{\partial \hat{w}} = 2X^T(X\hat{w} - y)$$

令上式为零可得到 w 的闭式解．不考虑 X 非满秩的情况．我们就可以得到

$$\hat{w} = (X^T X)^{-1} X^T y$$

2.2 Ridge 回归模型

基本模型保持不变，在损失函数中加入二范数：

$$L_{r2} = L(w) + \lambda \|w\|_2^2, \quad \lambda > 0$$

求导可得到

$$\begin{aligned} \hat{w} = \arg\min_{w} L(w) + \lambda w^T w \rightarrow \frac{\partial L(w)}{\partial w} + 2\lambda w = 0 \\ \rightarrow 2X^T X\hat{w} - 2X^T Y + 2\lambda\hat{w} = 0 \\ \rightarrow \hat{w} = (X^T X + \lambda I)X^T Y \end{aligned}$$

在贝叶斯概率视角下：

$$\begin{aligned} \hat{w} &= \arg\max_{w} \prod P(w|y) = \arg\max \prod_{i=1}^{N} P(y|w) \cdot P(w) \\ &= \arg\max_{w} \sum_{i=1}^{N} \log[P(y|w) \cdot P(w)] \\ &= \arg\max \sum_{i=1}^{N} \log\left(\frac{1}{\sqrt{2\pi}\sigma} \cdot \frac{1}{2\pi^{(p+1)/2}\,|\Sigma|^{1/2}}\right) + \sum_{i=1}^{N} \log exp\left\{-\frac{(y - w^T x)^2}{2\sigma^2} - \frac{w^T \Sigma^{-1} w}{2}\right\} \\ &= \arg\min \sum_{i=1}^{N} \frac{(y - w^T x)^2}{2\sigma^2} + N\frac{w^T \Sigma^{-1} w}{2} \\ &= \arg\min \sum_{i=1}^{N} (y - w^T x)^2 + N\sigma^2 w^T \Sigma^{-1} w \end{aligned}$$

使用贝叶斯学派的概率视角得到的最优参数与使用最小二乘法求解得到的岭回归模型最优参数相同．

2.2.1 凸优化角度

从凸优化的视角看待该正则项，上述的目标函数会等价于一下约束式．

$$\min_{w} \frac{1}{n} \|y - Xw\|^2$$
$$s.t. \|w\| \leq C$$

其中的 C 是与 λ 一一对应的常数．也就是我们通过限制了 w 的范数的大小，实现了对模型空间的限制，从而在一定程度上避开了 overfitting．而 Ridge 仍然需要通过数据的输入才会得到参数的输出，并不能得到稀疏解，得到的系数 w 仍然需要数据的所有特征才能计算预测结果，从计算量来看并没有得到改观．

2.3 高斯、正态分布

一维向量：

$$p(X) = p(X = x) = \mathcal{N}\left(X \mid \mu, \sigma^2\right) = \frac{1}{\sigma\sqrt{2\pi}} \exp^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

k 维向量：

$$p(X) = \mathcal{N}(X \mid \mu, \Sigma) = (2\pi)^{-k/2}|\Sigma|^{-\frac{1}{2}} \exp^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)}$$

$$Y = \begin{bmatrix} y_1 \\ \cdots \\ y_d \end{bmatrix} = \begin{bmatrix} \mathbf{e}_1^T \\ \cdots \\ \mathbf{e}_d^T \end{bmatrix} (x - \mu) = E^T(x - \mu) \tag{3}$$

而对于高斯分布的讨论更多内容参见该篇推导．

2.4 LASSO 回归模型

同样和 Ridge 类似，加入的是一范数．损失函数为

$$L_{r2} = L(w) + \lambda \|w\|_1, \quad \lambda > 0 \tag{4}$$

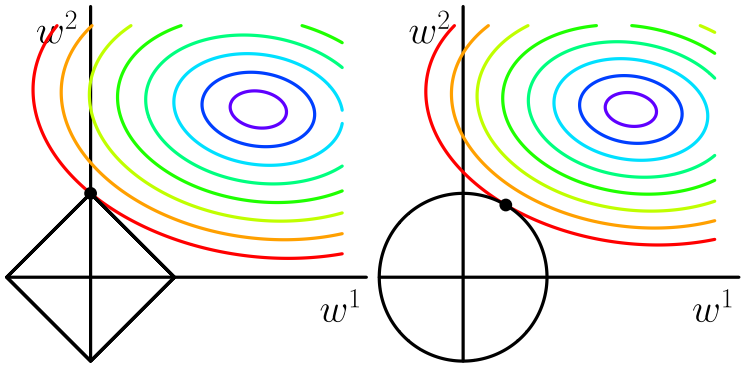
而一范式并没有如二范数一般的较好的性质，因此没有一个闭式解．LASSO 仍然是一个凸优化问题，但不再有解析解，其优良性质可以得到稀疏解，所谓稀疏解是指解向量 w 中有多个 0 分量的解，因此 LASSO 回归还可以进行特征选择．

2.4.1 稀疏解

为何 LASSO 可以产生稀疏解，和上述的 RiDGE 方法类似，可将其转换为

$$\min_w \frac{1}{n}\|y - Xw\|^2$$
$$s.t. \quad \|w\|_1 \le C$$

其中的 C 为一个与 λ 相对应的常数．在数学上，就是从一个梯度的跃迁，往往发生在 $w_j = 0$ 的时候．从一个简单的二维图像进行理解．



可从图中看出，l_1 基本上在坐标轴上相交．而严谨的数学推导要涉及凸优化的相关内容．

该博客对此有很好的解释．

2.5 对数几率回归

定义 2.1. 设 X 是连续随机变量，X 服从逻辑分布是指 X 具有以下分布函数和密度函数：

$$F(x) = P(X \le x)\frac{1}{1 + e^{-(x-\mu)/\gamma}}$$
$$f(x) = F'(x) = \frac{e^{-(x-\mu)/\gamma}}{\gamma(1 + e^{-(x-\mu)/\gamma})^2}$$

其中 μ 为位置参数，$\gamma > 0$ 是形状参数

2.5.1 二项逻辑回归

二项逻辑回归是一个分类模型，由条件概率分布 $P(Y|X)$ 表示，形式为参数化的 logistic 分布．这里的随机变量 X 为实数，随机变量 Y 为 0 或 1．我们通过监督学习来估计参数模型．

一个 sigmoid 函数：

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

进一步可对 x 求 sigmoid 函数的导数：

$$d\sigma(x)/dx = \frac{1}{exp(-x) + exp(x) + 2} = \frac{exp(-x)}{1 + exp(-x)}$$

sigmoid 函数的重要性质：

1. 关于原点中心对称；
2. 单调递增；
3. 处处可导，且导数的满足 $\sigma(x)' = \sigma(x)(1 - \sigma(x))$；

sigmoid 函数的模型形式：

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

再将上式进行转换，可得

$$\ln \frac{y}{1 - y} = w^T x + b$$

因此将 y 视为是样本 x 的正例可能性，则 $1 - y$ 表示的是其反例的可能性．二者的比值

$$\frac{y}{1 - y}$$

则称为几率 (odds)，反映了 x 作为正例的相对可能性．对几率取对数得到的是对数几率 (log odds, 或称 logit)

$$\ln \frac{y}{1 - y}$$

考虑一个输入 x 进行分类的线性函数 $w \cdot x$，其值域为实数域．通过逻辑斯蒂回归模型可将线性函数转换为概率:

$$P(Y = 1|x) = \frac{exp(w \cdot x)}{1 + exp(w \cdot x)}$$

对模型参数进行估计，仍然是使用 MLE 的方法：其似然函数为

$$\prod_{i=1}^{N} [\pi(x_i)]^{y_i} [1 - \pi(x_i)]^{1 - y_i}$$

对数似然函数为：

$$\begin{aligned} L(w) &= \sum_{i=1}^{N} [y_i \log \pi(x_i) + (1 - y_i) \log(1 - \pi(x_i))] \\ &= \sum_{i=1}^{N} \left[y_i \log \frac{\pi(x_i)}{1 - \pi(x_i)} + \log(1 - \pi(x_i)) \right] \\ &= \sum_{i=1}^{N} [y_i (w \cdot x_i) - \log(1 + exp(w \cdot x_i))] \end{aligned}$$

2.5.2 多项逻辑回归

假设响应变量 y 的取值为 K 类．$y \in \{1, 2, \cdots, K\}$．给定特征向量 x_i，假设事件 $y_i = k$ 的条件概率是

$$P(y_i = k|x_i) = \frac{exp(x_i'\beta_k)}{\sum_{i=1}^{K} exp(x_i'\beta_i)} \quad (k = 1, 2, \cdots, K) \tag{5}$$

其中的 β_k 为对应第 k 类回归系数．方程(**??**) RHS 为软极值函数．广泛用于分类问题的神经网络模型．同时各个条件概率之和为 1．

$$\sum_{k=1}^{K} P(y_i = k|x_i) = 1$$

而在方程(**??**) 无法识别出所有的系数 β_i，这是因为

$$\frac{exp(x_i'\beta_k + \alpha)}{\sum_{i=1}^{K} exp(x_i'\beta_i + \alpha)} = \frac{exp(x_i'\beta_k) \cdot \cancel{exp(x_i'\alpha)}}{\cancel{exp(x_i'\alpha)} \cdot exp(x_i'\beta_k)} = \frac{exp(x_i'\beta_k)}{\sum_{i=1}^{K} exp(x_i'\beta_i)} \tag{6}$$

通常会选择将某一个类作为参照类别 (base category)，然后令其系数为 0．之后的条件概率为

$$P(y_i = k|x_i) = \begin{cases} \dfrac{1}{1 + \sum_{i=2}^{K} exp(x_i'\beta)} & k = 1 \\ \dfrac{exp(x_i'\beta)}{1 + \sum_{i=2}^{K} exp(x_i'\beta)} & k = 2, \cdots, k \end{cases}$$

显然，当 $K = 2$ 时，就是我们的逻辑回归．

2.6 类内不平衡问题 (Class imbalance)

在训练数据的时候，可能会存在不同类别训练样例数相差很大情况（正类为小类）．

$$\frac{y}{1-y} > 1 \Rightarrow \frac{y}{1-y} > \frac{m^+}{m^-}$$

再对其进行缩放

- 首先可以采用欠采样的方式，去除一些反例使正反例数目接近（EasyEnsemble）欠采样就是把比较多的一类少采集一点，核心问题是怎么防止因为忽略了一些样本导致的信息缺失．

一个方法是 Tomek Links，是从数据集中的两个样本彼此是对方的最近邻，同时他们的类别不同进行采样，进一步删除那些样本比例更多的那一类的点．这样在数量上就平衡了两类的出现的比例．但这种做法可能产生的问题是信息的一定的缺失．

- 其次可以使用过采样的方式，即增加一些正例使正反例数目接近（SMOTE ）在少数类样本中选择一个样本 x_i，得到它的 K 近邻，然后随机选择一个样本 $\hat{x}_i$ ，在样本 x_i 和样本 $\hat{x}_i$ 之间随机选择一个点，生成新的样本．直到总样本数达到要求的采样倍率 N 为止．
- 最后可以考虑使用阈值移动（也称为调整权重）．

2.7 梯度下降

正如上面所述，一些问题无法通过求导进行求解，可以选择采用迭代的算法来实现．迭代中一个较为常用的方式是梯度下降法，其一个基本的思想是设置一个固定的步长，再将函数梯度的反方向作为下降的方向，由此不断的迭代，最后收敛到一定水平后即可找到我们想要的解．其数学表达形式是

$$w^{i+1} = w^i - \alpha \nabla L(X) \tag{7}$$

为何选择梯度的反方向作为下降方向，因为梯度的反方向是下降的最快方向．

证明．$f(x)$ 进行泰勒展开．

$$\begin{aligned} & f(x+\Delta x) = f(x) + (\nabla f(x))^T \Delta x + o\|\Delta x\| \\ \Leftrightarrow \quad & f(x+\Delta x) - f(x) = (\nabla f(x))^T \Delta x + o\|\Delta x\| \\ \Leftrightarrow (\Delta x \text{同向} \nabla f(x), \lambda > 0) \quad & f(x+\Delta x) - f(x) = (\nabla f(x))^T (\lambda \nabla f(x)) + o\|\Delta x\| \end{aligned}$$

若 Δx 无穷小，高阶部分忽略，可根据很显然看出 $LHS = RHS \geq 0$，若 Δx 与 $\nabla f(x)$ 取反向，$LHS = RHS \leq 0$．因此我们就可以根据此发现，迭代的增长与否与我们想要求解的目标有关．进一步证明沿着梯度反方向函数值下降最快．根据内积的定义可知

$$(\nabla f(x))^T \Delta x = \|\nabla f(x)\| \, \|\Delta x\| \cos\theta$$

显然在 $\theta = -1$ 时候，我们可以取得最大值．即 $\theta = \pi$．证毕． □

2.8 线性判别分析

在给定数据集 $D = \{(x_i, y_i)\}_{i=1}^m$ 下，令 X_i, μ_i, Σ_i 分别为第 i 类的集合、均值、协方差矩阵．若将数据投影到直线 w 上，两类样本的协方差矩阵为 $w^T\Sigma_0 w$ 和 $w^T\Sigma_1 w$．
而我们对线性判别的要求是类内的间距尽可能的小，类外的间距尽可能的大．因此可以让同类的投影点的方差尽可能小，即 $w^T\Sigma_0 w + w^T\Sigma_1 w$ 尽可能小（当然可能会问，为何不使用乘积的形式，加法会比乘法在运算上有优势）．同样，我们希望对某个类的判断尽可能的正确，也就是让不同类之间尽可能的距离远一点，因此要使得类间距，也即 $\left\|w^T\mu_0 - w^T\mu_1\right\|$ 尽可能的大．
同时在考虑这两者，也就是

$$J = -\frac{\left\|w^T\mu_0 - w^T\mu_1\right\|}{w^T\Sigma_0 w + w^T\Sigma_1 w}$$

尽可能的大．
定义类内离散度矩阵 (within-class scatter matrix)

$$S_w = \Sigma_0 + \Sigma_1 = \sum_{x \in X_0} (x - \mu_0)(x - \mu_0)^T + \sum_{x \in X_1} (x - \mu_1)(x - \mu_1)^T$$

类间离散度矩阵

$$S_b = (\mu_0 - \mu_1)(\mu_0 - \mu_1)^T$$

对 J 进行修正

$$J = \frac{w^T S_b w}{w^T S_w w}$$

S_b 与 S_w 就可称为是广义瑞利商．(generalized Rayleigh quotient)

进一步如何对估计参数 w 进行求解：回到最初的目标：最大化 $J = \frac{w^T S_b w}{w^T S_w w}$ 也就是

$$\min_w -w^T S_b w$$

$$s.t. w^T S_w w = 1$$

在约束 $w^T S_w w = 1$ 下最小化 $\min_w -w^T S_b w$，也就是对于 J 来说，控制分母不变，使得分子的相反数最小化．将这个问题转换为无约束问题，乘上拉格朗日乘子．

$$S_b w = \lambda S_w w$$

同时发现 $S_b w$ 的方向为 $\mu_0 - \mu_1$

$$S_b w = \lambda(\mu_0 - \mu_1)$$

带入得到

$$\hat{w} = S_w^{-1}(\mu_0 - \mu_1)$$

严格来说，应该需要将 $\hat{w}$ 进行标准化，使得 $\hat{w} S_w \hat{w} = 1$．
由此得到的最佳投影 $z_i = \hat{w}' x_i$，即为线性判元 (linear discriminant) 或线性判别得分．而最佳投影方向 $\hat{w}$ 称为是线性判别载荷 (linear discriminant loading)，线性判别系数或判别坐标．
对于新的样本点 x，可根据其线性判别得分 $z_0 = \hat{w}' x_0$ 与两类数据投影的中心位置 $\bar{z}_0 = \hat{w}' \mu_0$ 与 $\bar{z}_1 = \hat{w}' \mu_1$ 的距离远近进行分类．[1]

3 决策树

决策树是一种基本的分类与回归的算法．分类中，对于基于特征对实例进行分类的过程．可以认为是 'if then' 规则的集合．也可以是定义在特征空间与类空间上的概率分布．

3.1 决策树模型与学习

定义 3.1. *分类决策树模型是一种描述分类的树形结构．决策树由 node 和 directed edge 组成．结点有两种：内部结点和叶节点．内部节点表示一个特征或属性，叶节点表示一个类．*

在每一条从根节点到叶节点上构建一个规则，内部节点对应的是规则的条件，叶节点是规则的结论，决策树的路径或其对应的 if-then 规则集合具有：互斥且完备性．每一个实例是被一条规则所覆盖．

[1] 这里参考了陈强的机器学习-线性判别分析课件

3.2 决策树与条件概率分布

决策树是给定特征下的类的条件概率分布，定义在特征空间中的一个划分 (partition) 上，将特征空间分为几个不相交的单元 (cell) 或区域 (region). 同时在每一个单元定义一个类的概率分布就构成了一个条件概率分布．假设 X 为随机变量，Y 为类的随机变量．这个概率分布表示为 $P(Y|X)$．X 的取值为给定划分单元的集合．Y 取值为类的集合．各个的叶结点上的条件概率往往会偏向于某一个类，即属于某一个类的概率更大，决策树的分类时候就会将该结点的实例以概率“投票”的方式选择类．

3.3 决策树学习

在给定的训练数据集:

$$D = \{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$$

其中 $x_i = (x_i^{(1)}, x_i^{(2)}, \cdots, x_i^{(n)})$ 为输入实例，n 为特征个数．$y \in \{1, 2, 3, \cdots, K\}$ 为类标记．$i = 1, 2, \cdots, N$，N 为样本容量．学习的目标是根据给定的训练数据集构建一个决策树模型，使得得到正确的分类．

注释 3.1. 决策树学习的本质是从训练数据集中学习关于一条分类规则．这些分类规则能够帮助进行预测和标记等作用．

决策树算法，递归的选择最优特征，并根据该特征对训练集进行分割，使得对各个子训练集有一个最好的分类过程．最终每个实例都能被分到叶节点上，都有了明确的类，生成了一个决策树．

3.4 特征选择

特征的选择在于对训练数据具有分类能力的特征，可以提高决策树的学习效率．特征的选择往往是信息增益或信息增益比．

3.5 信息熵

信息熵是接收每条信息所带来的信息平均量．

1. 信息量：越不可能的事件发生的，信息量越大 $I(x_0) = -\log(p(x_0))$
2. 信息熵：越大的期望表示的是系统越混乱．
3. 相对熵：又称为 KL 散度，衡量两个概率分布的差异程度

$$KLD = D_{KL}(p||q) = \sum p(x_i) - \log(\frac{p(x_i)}{q(x_i)})$$

4. 交叉熵：衡量差异的量

假设 px 是真实的分布，qx 是算法计算的分布．逻辑回归就是用交叉熵度量损失函数．
用交叉熵来衡量：

$$L(w) = -\sum_{i=1}^{N}(y_i \log(p_1(x_i)) + (1 - y_i)\log(p_0(x_i)))$$

熵只依赖于 X 的分布与 X 的取值无关．熵越大，随机变量的不确定性越大，从定义可验证：

$$0 \le H(X) \le \log n$$

3.5.1 信息增益

定义 3.2 (信息增益 (Information Divergence))**.** *信息增益是知道特征 X 的信息，能够使得 Y 的信息的不确定性降低的程度．特征 A 对训练数据集 D 的信息增益 $g(D,A)$，定义集合 D 的经验熵 $H(D)$ 与特征 A 给定条件下 D 的经验条件熵 $H(D|A)$ 之差，即*

$$g(D,A) = H(D) - H(D|A)$$

最终降低的不确定性 $g(D,A)$ 称为互信息 (mutual infomation)，决策树学习中的信息增益等价于训练集中类与特征的互信息．

信息增益是在原有的经验熵的基础上加上经验条件熵（也就是被降低的部分）最后而得到的．可以理解为在自身熵的条件下与外界的合力作用下的信息熵的状态．
一定程度上是可描述为测量某种“意外”，这个意外越大，所带来的信息越多．信息熵也就越大．

3.5.2 信息增益比

信息增益值是相对于训练数据集而言的．没有一个绝对意义，分类困难时候，也即是在经验上（自身先决条件）下较大，信息增益值也会偏大；同样反过来，信息增益值也会偏小．这时候使用信息增益比能够很好的对这样的问题进行矫正．

定义 3.3. *特征 A 对训练数据集 D 的信息增益比 $g_R(D,A)$ 是其信息增益 $g(D,A)$ 与训练数据集 D 的经验熵之比*

$$g_R(D,A) = \frac{g(D,A)}{H(D)}$$

从该定义可看出，信息增益比在先前的信息增益 $g(D,A)$ 的基础上，以整体的信息规模（经验熵）$H(D)$ 的关系．也就是刻画的是对整体的贡献比率．

why Information Gain instead of just Entropy Note that all the examples below will have the same entropy for the parents since they (each of the attributes) are using entire data of the same table. However, when considering nodes of different branches, they may not have the same entropy. It is also time to discuss the advantage of using information GAIN instead of just looking at the Entropy alone.

We can consider diminishing returns here. For example, when a node already has very low entropy (from which we can already make decisions), it seems unnecessary to split it further. So this makes GAIN a very suitable metric for deciding whether a node should be split.

3.6 ID3 算法

3.6.1 决策树的生成

在决策树的各个节点使用信息增益选择特征，递归地构建.
算法：从根结点开始，计算所有可能的信息增益，选择信息增益最大的特征作为结点的特征.

- 输入：训练数据 D，特征集 A，阈值 ε；
- 输出：决策树 T；
- 若 D 中所有的实例属于同一类 C_k，T 为单结点数，并将类 C_k 作为该结点的类标记，返回 T；
- 若 $A = \phi$ 则 T 为单结点数，并将 D 中实例最大的类 C_k 作为该结点的类标记.
- 否则计算信息增益，选择信息增益最大的特征 A_g；
- 若 A_g 的信息增益小于阈值，这让其为单结点数，并将 D 中实例最大的类 C_k 作为该结点的类标记，返回 T；

3.7 决策树的剪枝

在对决策树进行训练的过程中，很容易发生过拟合，因此需要简化已生成的决策树. 决策树的剪枝往往通过极小化决策树整体的损失函数 (loss function) 或代价函数 (cost function) 来实现.

$$C_\alpha(T) = \sum_{t=1}^{|T|} N_t H_t(T) + \alpha |T|$$

其中经验熵为

$$H_t(T) = -\sum_k \frac{N_{tk}}{N_t} \log \frac{N_{tk}}{N}$$

这时候：

$$C_\alpha(T) = C(T) + \alpha |T|$$

$C(T)$ 表示的是模型对训练数据的预测误差，即模型与训练数据的拟合度. $|T|$ 表示的是模型复杂度，参数 $\alpha \geq 0$ 表示的是控制两者之间的影响关系，也就是一个惩罚系数. 剪枝：同样也是优化损失函数，使得其最小的参数为我们想要的模型. 往往存在一个 trade-off：子树越大，训练数据拟合越好而模型的复杂度越高；而子树越小，训练数据拟合差而复杂度低.

3.8 CART 算法

CART 算法所使用的决策树是二叉树，即每次将母节点一分为二，分裂得到两个子节点，直到终点．本质上，二叉树是根据特征空间进行递归分割 (recursive partitioning)，每次沿着与某个特征变量平行的方向进行分割．被切割为矩形或超矩形．算法步骤：

1. 决策树生成：基于训练集生成
2. 决策树剪枝：用验证集进行剪枝并选择最优的

3.8.1 CART 生成

1. 回归树的生成：假设 X 与 Y 为输入和输出变量，且 Y 是连续变量，给定训练集

$$D = \{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$$

一个回归树对应输入空间的一个划分以及在划分单元上的输出值．

$$f(x) = \sum_{m=1}^{M} c_m I(x \in R_m)$$

输入空间给定时候，可以用平方误差 $\sum_{x_i \in R_m}(y_i - f(x_i))^2$ 进行表示回归树对于训练集的预测误差．对于如何切分，主要采用的是启发式算法，随机选择第 j 个变量 $x_j^{(1)}$ 和其值 s，作为切分变量 (splitting variable) 和切分点 (splitting point) 并定义两个区域：

$$R_1(j, s) = \{x|x^{(j)} \leq s\} R_2(j, s) = \{x|x^{(j)} > s\}$$

这样就得到了一个划分．进一步需要遍历所有可能的切分然后找到最优的切分点，具体的实现方式是

$$\min\left[\min_{c_1} \sum (y_i - c_1)^2 + \min_{c_2} \sum (y_i - c_2)^2\right]$$

对固定输入变量 j 可以找到最优切分点 s．

2. 分类树的生成：根据前文所述，分类树是通过分割特质空间进行分类，而我们的 CART 分类法，分类树采用 Gini 最优特征选取，同时决定该特征的最优二值切分点．对于 CART 算法的分类二叉树，在每个节点进行分裂时候，需要选择什么分裂变量 (split variable) 进行分裂，以及选择怎么样的临界值进行分裂．同时，我们希望在每一个叶节点，其纯度 (purity) 总体达到最大．对于 CART 来说更为简化的来说是希望两个子节点内部的纯度最高．也就是分裂的不纯度下降最快．由此我们去定义一个节点不纯度函数，常见的不纯度函数为 Gini 指数和熵．

定义 3.4 (Gini index)**.** *假设有 k 类，样本点属于第 k 类的概率为 p_k，则该样本的 Gini 指数为*

$$Gini(p) = \sum_{k=1}^{K} p_k(1 - p_k) = 1 - \sum_{k=1}^{K} p_k^2$$

基尼指数同样可以表示不确定性，$Gini(D)$ 表示集合 D 的不确定性，$Gini(D,A)$ 表示经过 $A=a$ 分割之后，D 的不确定性．Gini 指数越大不确定越大．

特别的对于一个二分类问题，样本点属于第 1 类的概率为 p，则该概率的 Gini 指数为

$$Gini(p_1,p_2)=1-p_1^2-(1-p_1)^2=2p_1(1-p_1)$$

其中的 $p_1(1-p_1)$ 可视为是两点分布的方差．

3.9 随机森林

定义 3.5. 随机森林是在集成学习的思想框架下，将我们先前的一棵棵树集聚为一个森林．它的基本单元为决策树．

随机森林的分类错误率与森林中的两颗树的相关性有关，也就是相关性越高，所能够看到的差异越小，错误率越大．另一个是森林的每棵树的分类能力．每棵树的分类能力越强，整个森林的错误率越低．

3.10 总结

1. 分类决策树是表示基于特征的对特征树形结构，决策树可以转换为一个 if-then 集合．也可以看作是一个特征空间的划分的条件分布．
2. 决策树的学习最终希望在得到一个与训练集数据拟合较好，并且能够将复杂度降低的决策树．
3. 决策树的生成：通常会使用最大的信息增益、信息增益比、Gini 系数最小的作为准则．

4 SVM

4.1 感知机模型

定义 4.1. 假设输入空间是 $\mathcal{X}\subset R^n$，输出空间是 $\mathcal{Y}=\{+1,-1\}$，输入 $x\in\mathcal{X}$ 表示实例的特征向量．对应于输入空间（特征空间）的点；输出 $y\in\mathcal{Y}$ 表示实例的类别．输入空间到输出空间的函数

$$f(x)=sign(w\cdot x+b)$$

称为感知机，其中 w 和 b 为模型参数，w 称为权值或权值向量 *(weight vector)*，$b\in R$ 叫做偏置 *(bias)*．$w\cdot x$ 表示 w 和 x 的内积．$sign$ 是符号函数．

$$sign(x)\begin{cases}+1, x>0\\-1, x<0\end{cases}$$

感知机实际上为寻找一个能够将两种不同的类别进行分离的模型．当我们找到了这个分类线，此时距离这条分类线最近的点，称为支持向量．

4.1.1 感知机学习策略

定义 4.2 (数据集的线性可分性)**.** 给定一个数据集

$$T = \{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$$

若存在一个超平面

$$w \cdot x + b = 0$$

将数据集分为两个平面，若存在，则认为这个是可分的，若不存在则称为不可分的．

为了找到这样一个完全正确分离的超平面，需要定义一个学习策略．损失函数的一个选择是误分类点的总数，用其进行最值优化．但这样的损失函数在函数性质上并不可导，不容易优化求解．我们假定距离:

$$\frac{1}{\|w\|} |w \cdot x + 0 + b|$$

对于误分类的数据来说

$$-y_i(w \cdot x_i + b) > 0$$

成立．因此我们的损失函数可以定义为

$$L(w, b) = -\sum_{x_i \in M} y_i(wx_i + b)$$

具体的算法在这并不展开．

4.1.2 算法的收敛性

定理 4.1. 假设训练数据集 T 是线性可分的，则存在满足条件的 $\left\|\hat{w}_{opt}\right\| = 1$ 的超平面，是的将训练数据集完全正确的分开，且存在 $\gamma > 0$，对有的 $i = 1, 2, \cdots, N$:

$$y_i(w_{opt} \cdot x_i + b_{opt}) \geq \gamma$$

4.2 SVM 概述

SVM 也称为支持向量机，是一个二分类模型，具体是在特征空间中寻找间隔最大的线性分类器，间隔能够使得其有别于感知器．SVM 还有别于核技巧，使得其为一个非线性分类器．SVM 其学习策略是在使得两个类别间的间隔最大，具体来说，就是使得在映射空间内的求解一个超平面的参数（凸二次规划问题）

定义 4.3. 凸函数：对凸函数来说，数学表达为

$$f([x_1 + x_2]/2) \leq [f(x_1) + f(x_2)]/2$$

$f(x) = x$ 也是一个凸函数，但非一个严格的凸函数，严格的凸函数需要不等式严格小于．比如 $f(x)$ 就是一个凸函数．

定义 4.4 (仿射函数)**.** 仿射函数是最高次项为 *1* 的函数．$f(X) = w_1x_1 + w_2x_2 + \cdots + w_nx_n$ 仿射函数同样也是非严格的凸函数．常数项为 *0* 的函数常常被称为线性函数，线性函数是过原点的函数．

定义 4.5. 若目标函数为凸函数，约束为仿射函数，这个就称为是一个凸优化问题．实际上更为广义的凸优化问题的定义是凸集中的凸函数的最小化问题．
进一步的对于目标函数为二次函数，不等式约束为仿射函数的问题称为是凸二次规划问题．

定义 4.6. 超平面：是从所建立的空间中所减少的一维的子空间，能够将原有的空间进行得到划分．在 n 维空间中，设 x_0 为超平面上一点，w 为超平面上的法向量．则

$$\omega(x - x_0) = \omega x - \omega x_0 = 0 \text{法向量与平面任意一条线垂直}$$

令

$$\omega x_0 = -b$$

得到超平面的表达式

$$\omega x + b = 0$$

对于一个超平面，ω 确定了超平面的方向，而 b 确定了超平面上的一点，即超平面在空间的位置．实际上在空间的位置是在 ω 的基础上确定的．

回到我们想要研究的 SVM 问题上，在给定的空间和空间内的数据，我们想要使得数据集中不同类尽可能地分隔开，也即使得类间隔最大．但如何度量这种间隔．

定义 4.7. 对于给定的数据集 T 和超平面 $w \cdot x + b = 0$，定义超平面关于样本点 (x_i, y_i) 的几何间隔为

$$\gamma_i = y_i(\frac{w}{\|w\|} \cdot x_i + \frac{b}{\|w\|})$$

在这里我们利用了欧氏空间点到平面距离公式计算

$$\frac{|AX + BY + C|}{\sqrt{A^2 + B^2}} = \frac{y_i(w^T x_i + b)}{\|w\|}$$

y_i 恰好能够反映距离的正负性．而我们要寻找的最近的点才是真正意义上的间隔，进一步对于所有的点求解距离得到最小值才是间隔，数学表示为

$$\gamma = \min_{i=1,2,\cdots,N} \gamma_i$$

4.3 回顾拉格朗日

给定一个目标 $f:\mathbb{R}^{\varkappa}\to\mathbb{R}$，对于一个等式约束优化问题，我们希望找到的是 $\mathbf{x}\in\mathbb{R}^n$，在满足 $g(X)=0$ 的前提下，使得 $f(x)$ 有最小值．这个约束记为

$$\begin{aligned}&\min f(x)\\ &s.t. g(x)=0\end{aligned}$$

定义拉格朗日函数

$$L(x,\lambda)=f(x)+\lambda g(x) \tag{8}$$

其中的 *lambda* 就称为是拉格朗日乘子，这样也就使得我们将一个约束问题转换为求解无约束最优化问题．分别对 x 和 λ 求偏导，其极值的必要条件为

$$\begin{aligned}\nabla_x L=\frac{\partial L}{\partial x}=\nabla f+\lambda\nabla g=0\\ \nabla_\lambda L=\frac{\partial L}{\partial \lambda}=g(x)=0\end{aligned}$$

其中第一式为定常方程式 (stationary equation)，第二式为约束条件．上述有 n 个 x 向量以及一个参数 λ，n+1 个方程可以最终求解未知数．

进一步，我们想将约束条件得到放松，$g(x)\le 0$，使得

$$\begin{aligned}&\min f(x)\\ &s.t.\quad g(x)\le 0\end{aligned}$$

在 $g(x)\le 0$ 的约束下的 $K=x,\in\mathbb{R}^n|g(x)\le 0$．假设 x^* 满足条件，需要进行分类讨论：

- 当 x 恰好使得 $g(x^*)=0$ 成立，称为边界解，约束条件是有效的；
- 当 x 使得 $g(X^*)<0$，称为内点解，约束条件是无效的．

在内部解下，我们需要回到原有的问题极值条件下，约束不发生作用，此时 $f(x)=0,\lambda=0$ 若为边界解，不等式变为等式约束，如为我们一开始讨论的那样

$$\nabla_x L=\frac{\partial L}{\partial x}=\nabla f+\lambda\nabla g=0 \tag{9}$$

即 $\nabla f=-\lambda\nabla g$

综合前面所述，$\lambda \nabla g(x) = 0$ 在约束条件下恒成立，这称为互补松弛性定理．因此这些条件可以列出

$$\begin{aligned} \nabla_x L = \nabla f + \lambda \nabla g &= 0 \\ g(X) &\leq 0 \\ \lambda &\geq 0 \\ \lambda g(X) &= 0 \end{aligned}$$

这些条件合称为 KKT 条件，若是最大化 $f(x)$ 问题，我们还需要考虑 $g(X) < 0$ 其对偶可行性为 $\lambda < 0$.

将不等式与等式约束一同写出，其标准形式为

$$\begin{aligned} \min \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & g_j(\mathbf{x}) = 0, \quad j = 1, \ldots, m, \\ & h_k(\mathbf{x}) \leq 0, \quad k = 1, \ldots, p. \end{aligned}$$

一个标准的定义拉格朗日函数：

$$L(X, \{\lambda_j\}, \{\mu_k\}) = f(X) + \sum_{j=1}^{m} \lambda_j g_j(X) + \sum_{k=1}^{p} \mu_k h_k(x) \tag{10}$$

回到我们的 SVM 中，在之前的定义下：

$$\begin{aligned} & \max_{\boldsymbol{w},b} \gamma \\ & s.t. \quad y_i \left(\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|} \cdot \boldsymbol{x}_i + \frac{b}{\|\boldsymbol{w}\|} \right) \geq \gamma, i = 1, 2, \ldots, N \end{aligned}$$

将等式两侧同时除以 γ 得到

$$y_i \left(\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|\gamma} \cdot \boldsymbol{x}_i + \frac{b}{\|\boldsymbol{w}\|\gamma} \right) \geq 1 \tag{11}$$

$\|w\|$ 和 b 都是标量，对于超平面 $wx + b$ 来说，我们只需要关注于其方向，比如一个二维平面的超平面 $ax + b$，其大小比例关系给定下，大小的变化只是会影响在超平面上的对应点的变化．

$$w = \frac{w}{\|w\|\gamma} b = \frac{b}{\|w\|\gamma} \tag{12}$$

得到

$$y_i(wx_i + b) \geq 1, i = 1, 2 \cdots, N \tag{13}$$

目标为最大化 γ 从而等价转换为 $\frac{1}{\|w\|}$，等价于一个更好表达的 $\frac{1}{2}\|w\|^2$，因此 SVM 可以转换为一个在约

束条件的下的最优化问题，而目标函数为二次，因此也就是上文提到的凸二次优化问题.

而这个优化问题，我们转换为拉格朗日对偶问题，代入公式中，其目标函数为

$$L(w,b,\alpha)=\frac{1}{2}\|w\|^2-\sum_{i=1}^{N}\alpha_i(y_i(w\cdot x_i+b)-1) \tag{14}$$

其中的 α_i 为拉格朗日乘子，且根据我们前面的推导可知 $\alpha\geq 0$，现在我们令

$$\theta(w)=\max L(w,b,\alpha) \tag{15}$$

这里的 $\theta(w)$ 表示的是超平面 $wx+b$ 与点集 $\{y_i,y_i\in Y\}$ 的所有距离中，对参数 w,b 优化实现对所有距离的最大化，而进一步我们需要找到使得最大化距离下的最近距离，这个距离就是间隔距离.

$$\min\theta(w)=\min_{\alpha\geq 0}\max_{w,b}L(w,b,\alpha) \tag{16}$$

实际上，这里的所有的点对于 SVM 的超平面进行了“投票”，根据总体的距离进行选择，根据不同点的分布实际上将超平面的参数选出. 而这里仍然是一个硬间隔，因在一开始我们的约束条件已经得到相关的说明.

而若需要转换为无约束问题即式可能会一些点 (x_i,y_i) 并不满足，即:

$$y_i(\boldsymbol{w}\cdot\boldsymbol{x}+b)<1$$

则需要将这些可能落入其中的点进行“惩罚”进一步需要考虑一些落入在间隔内的点，通常让这个惩罚项极大反优化方向呈现：我们的优化是 min 问题，因此将 α_i 设为无穷大，则 $\theta(\boldsymbol{w})$ 也为无穷大.

此时的 $\theta(\boldsymbol{w})$

$$\theta(w)=\begin{cases}\frac{1}{2}\|w\|^2 & x\in\text{可行区域}\\ +\infty & x\in\text{不可行区域}\end{cases}$$

$$\min\theta(w)=\min_{w,b}\max_{\alpha\geq 0}L(w,b,\alpha)=\frac{1}{2}\|w\|^2-\sum_{i=1}^{N}\alpha_i(y_i(w\cdot x_i+b)-1) \tag{17}$$

而这个问题仍然并不是一个好求解的问题，进一步考虑其对偶问题

$$\min\theta(w)=\max_{\alpha\geq 0}\min_{w,b}L(w,b,\alpha)=\frac{1}{2}\|w\|^2-\sum_{i=1}^{N}\alpha_i(y_i(w\cdot x_i+b)-1)$$

对偶问题满足的可能是需要几个条件. 优化问题是一个凸优化问题（这里显然满足）、满足 KKT 条件; KKT 条件:

$$\begin{cases}\alpha_i \geq 0 \\ y_i(w_i \cdot x_i + b) - 1 \geq 0 \\ \alpha_i(y_i(\boldsymbol{w_i} \cdot \boldsymbol{x_i} + b) - 1) = 0\end{cases}$$

令 $L(\omega, b, \alpha)$ 求偏导：

$$w = \sum_{i=1}^{m} \alpha_i y_i x_i \tag{18}$$

$$0 = \sum_{i=1} \alpha_i y_i \tag{19}$$

将这些代入消去 w, b 可得到

$$L(w, b, \alpha) = -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j (\boldsymbol{x_i} \boldsymbol{x_j}) + \sum_{i=1}^{N} \alpha_i$$

而以上的假设都还是基于线性可分的情况下做出的，对于线性不可分的情况，也即

$$y_i(\boldsymbol{w} \cdot \boldsymbol{x} + b) < 1$$

我们对于损失函数进行改进，在之前的情况下对于落入间隔内的点采用的是极大化的策略．实际上也就是将之前的不可行区域转变为可行区域，同时损失函数以不同的形式呈现．这里使用的损失函数为 hinge 损失

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

$$s.t. y_i(w_i \cdot x_i + b) \geq 1 - \xi, \xi_i \geq 0, i = 1, 2, \cdots, N$$

其中的 ξ_i 为线性优化中的松弛变量．即那些并不满足硬性约束的程度，再乘以系数即为一个损失．$C > 0$ 也称为损失参数．

SMO 算法的基本思路在固定 α_i 之外的所有参数，再求出 α_i 上的极值．由于存在约束 $\sum_{i=1}^{m} \alpha_i y_i = 0$ ，若固定之后，不断直到收敛．

4.4 核方法

核技巧的基本思想：从原有将特征向量的特征转换受到特征空间维度约束，考虑特征转换与内积计算结合．

定义 4.8 (在数学上的定义)**.** 我们通过定义一个二元函数 $k : E \times E \to \mathbb{R}$ 来表达对元素的相似性. 这种思想同样对于处理各种信息任务是有效的. 其中, 根据我们的定义, 若 $k(x, y)$ 越大, x, y 的相似性越强. 因此我们将这个函数 $k : E \times E \to \mathbb{R}$ 称为核.

例子 4.1 (Epanechnikov kernel)**.** 我们使用一个核

$$k(x, y) = D(\frac{|x - y|}{\lambda})$$

$$D(t) = \begin{cases} \frac{3}{4}(1 - t^2), & |t| \leq 1 \\ 0, & Otherwise \end{cases}$$

对于 $lambda > 0$, 我们可以建立以下的函数

$$\hat{f}(x) = \frac{\sum_{i=1}^{N} k(x, x_i) y_i}{\sum_{j=1}^{N}} k(x, x_j)$$

其中每一个观测值 $(x_1, y_1), \cdots, (x_n, y_n) \in E \times R$. 我们再对于 $x* \in E$ 从 N 对中输入, $y_1, \cdots, y_N$ 加权得到

$$\frac{k(x*, x_1)}{\sum_{j=1}^{N} k(x*, x_j)}, \cdots, \frac{k(x*, x_N)}{\sum_{j=1}^{N} k(x*, x_j)}$$

作为估计值 $\hat{f(x)}$, 同时我们有: 若 $k(x, y)$ 的值越大会得到 $x, y \in E$ 越相似. 也即 $x*$ 和 x_i 越相近, y_i 权重越大. 给定输入 $x* \in Efori = 1, 2, \cdots, N$, 权重 y_i 使得 $x_i - \lambda \leq x* \leq x_i + \lambda$ 与 $k(x*, x_i)$ 成比例. 若构成一个逼近, 即 λ 越小, 使得 $x_* \to x_i$, 因此使得最终的估计越靠近.

在这里就不得不引入泛函分析: 在这个空间内上的点的函数. 我们对不同的空间的定义: 在这个空间上, 我们通过一组基底和运算能够支撑起整个空间.

- 线性空间定义了的是数乘和加法的运算空间;
- 线性度量空间: 在线性空间的基础上加入了度量;
- 线性赋范空间: 在线性空间内加入范数; 其中一个著名的空间为 Banach 空间: 一个完备的赋范线性空间;
- 内积线性空间 (inner product): 内积的赋范线性空间;
- 欧几里得空间: 有限维的实内积线性空间;
- 希尔伯特空间: 完备的内积线性空间;

再生希尔伯特空间: 将 $\{\sqrt{\lambda_i}\psi_i\}_{i=1}^{\infty}$ 作为一组正交基, 生成希尔伯特空间

$$f = \sum_{i=1}^{\infty} f_i \sqrt{\lambda}\psi$$

f 看为 $\mathcal{H}$ 向量 $f = (f_1, f_2, \cdots, f_n)^T_{\mathcal{H}}$ 另外的一个函数 $g = (g_1, g_2, \cdots, g_n)^T_{\mathcal{H}}$

$$< f, g >_{\mathcal{H}} = \sum f_i g_i$$

对于一个函数来说：我们可以看出

$$\min_{\alpha} \alpha^T Q\alpha - e^T\alpha$$
$$0 \le \alpha_i \le C_i$$

对于这个问题来说，除了 α 以外，其他的参数是已知的．其中

$$Q_{ij} = y_i y_j \Phi(x_i)^T \Phi(x_j) = (y_1\Phi(x_i), \cdots, y_N\Phi(x_i))^T (y_1 x_1, \cdots, y_N\Phi(x_N))$$

定义核函数

$$K(x_i, x_j) = \Phi(x_i)\Phi(x_j)$$

进一步代入得到

$$Q_{ij} = y_i y_j K(x_i, x_j)$$

核函数形式：若是一个对称矩阵，满足 $k(x_i, x_j) = k(x_j, x_i)$ 的形式，关于它的核函数

$$K = \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_1, x_j) & \cdots & k(x_1, x_m) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ k(x_i, x_1) & \cdots & k(x_i, x_j) & \cdots & k(x_i, x_m) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ k(x_m, x_i) & \cdots & k(x_m, x_j) & \cdots & k(x_m, x_m) \end{bmatrix}$$

那么这个矩阵就是半正定的，可以作为核函数使用．

定理 4.2 (正定核的充要条件)**.** *假设 $K : \mathcal{X} \times \mathcal{X} \to R$ 是对称函数，则 $K(x, z)$ 为正定核函数的充要条件是 $\forall x_i \in \mathcal{X}, i = 1, 2, \cdots, m$，$K(x, z)$ 对应的 Gram 矩阵：*

$$K = [K(x_i, x_j)] \tag{20}$$

是半正定矩阵．

我们试图去证明上面这个命题：必要性：由于是正定核，所以存在 $\mathcal{X}$ 到希尔伯特空间 $\mathcal{H}$ 的映射 ϕ，使得

$$K(x, z) = \phi(x) \cdot \phi(z)$$

因此，对任意的 $x_1, x_2, \cdots, x_m$ 构造一个关于 $x_1, x_2, \cdots, x_m$ 的 $K(x, z)$ 矩阵：

$$[K_{ij}]_{m\times m} = [K(x_i, x_j)]_{m\times m}$$

对任意的 $c_1, c_2, \cdots, c_m \in R$ 有:

$$\sum_{i,j=1}^{m} c_i c_j K(x_i, x_j) = \sum_{i,j=1}^{m} c_i c_j (\phi(x_i) \cdot \phi(x_j))$$
$$= \left(\sum_i c_i \phi(x_i) \right) \cdot \left(\sum_j c_j \phi(x_j) \right) = \left\| \sum_i c_i \phi(x_i) \right\|^2 \geq 0$$

表明是半正定的. 充分性: 根据前面的结果, 对于给定的 $K(x, z)$, 可以构造一个希尔伯特空间 $\mathcal{H}$ 的映射:

$$\phi : x \to K(\cdot, x)$$

根据 $K(\cdot, x) \cdot f = f(x)$ 和 $K(\cdot, x) \cdot K(\cdot, z) = K(x, z)$ 可知

$$K(x, z) = \phi(x) \cdot \phi(z)$$

表示是在 $\mathcal{X} \times \mathcal{X}$ 上的核函数. 因此可以得到一个定理: 若是对称函数, 这对于任意的 $x_i \in \mathcal{X}$, $K(x, z)$ 对应的 Gram 矩阵:

$$K = [K(x_i, x_j)]_{m \times m}$$

是半正定矩阵, 则称 $K(x, z)$ 是正定核.

4.5 序列最小最优化算法

简要概述: 当要求解的凸二次规划的训练样本容量较大时候, 之前所提到的一些算法较为低效, 因此我们提出序列最小最优化算法 (Sequential minimal optimization,SMO). SMO 主要解凸二次规划的对偶问题:

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2} \sum_{i=1} \sum_{i=1} \alpha_1 \alpha_j y_i y_j K(x_i, x_j) - \sum_{i=1} \alpha_i \\ s.t. \quad & \sum_{i=1}^{N} \alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq C, i = 1, 2, \cdots, N \end{aligned}$$

SMO 算法是一个启发式算法. 基本的核心在于若所有的变量都满足 KKT 条件. 这个解是可求的. 这时候, 当我们有一些变量, 选择两个变量, 来构建一个二次规划问题. 这个二次规划是关于这两个变量的更接近于原始解的.

$$\alpha_1 = -y_1 \sum_{i=2}^{N} \alpha_i y_i$$

若 α_2 确定, 我们将其余变量视为常数以后, 那么 α_1 也跟着确定了. 所以子问题中同时更新了这两个变量.

4.5.1 常用的一些核函数

线性核函数：令 $E := \mathbb{R}^d$，kernel$k(x,y) = x^T Ay =< Bx, By >_H, x, y \in \mathbb{R}^d$ 基于有限非负矩阵 $A = B^T B \in \mathbb{R}^{d\times d}, B \in \mathbb{R}^{d\times d}$．其正有限核是一个内积的扩展：

$$K_1(x_i, x_j) = x_i^T x_j$$

多项式核函数：

$$K_{m,d}(x,y) := (x^T y + 1)^m$$

需优化的参数为 γ, ζ, n

高斯核函数：

$$K_p(x_i, x_j) = exp\{-\gamma \left\|x_i - x_j\right\|^2\}$$

高斯核函数对应的是一个无穷维的转换．一个核函数与对应一个特征转换，核函数的值与转换之后的内积相等．

$$K_g(x_i, x_j) = exp\{-\left\|x_i - x_j\right\|^2\} = exp\{-x_i^2\}exp\{-x_j^2\}exp\{2x_i x_j\}$$

4.6 SVR

在处理一些没有分布在模型上的数据时候，支持向量回归设立一个容忍度，若支持向量容忍 ε 之下的记为 0，在二维空间中，支持向量机可以用蓝色带状区域展示，称为决策带．黄色数据点称为训练误差为 0 的数据．红色的数据点表示有训练误差的数据．

模型的形式

$$\min_{w,b,\xi_1,\xi_2} \frac{1}{2} w^T w + C \sum_{i=1}^{N} (\xi_i + \hat{\xi}_i)$$

约束条件为我们的预测误差需要小于定义的误差

$$s.t. \begin{cases} f(x_i) - y_i \le \epsilon + \xi_i \\ y_i - f(x_i) \le \epsilon + \xi_i \\ \xi_i \ge 0, \hat{\xi}_i \ge 0, i = 1, 2, \cdots, N \end{cases}$$

进一步引入拉格朗日乘子，我们可以得到

$$L(w, b, \hat{\alpha}, \alpha, \xi, \hat{\xi}, \mu, \hat{\mu}) = \frac{1}{2} w^T w + C \sum_{i=1}^{N} (\xi_i + \hat{\xi}_i) - \sum_{i=1}^{N} \mu_i \xi_i - \sum_{i=1}^{N} \hat{\mu} \hat{\xi}$$

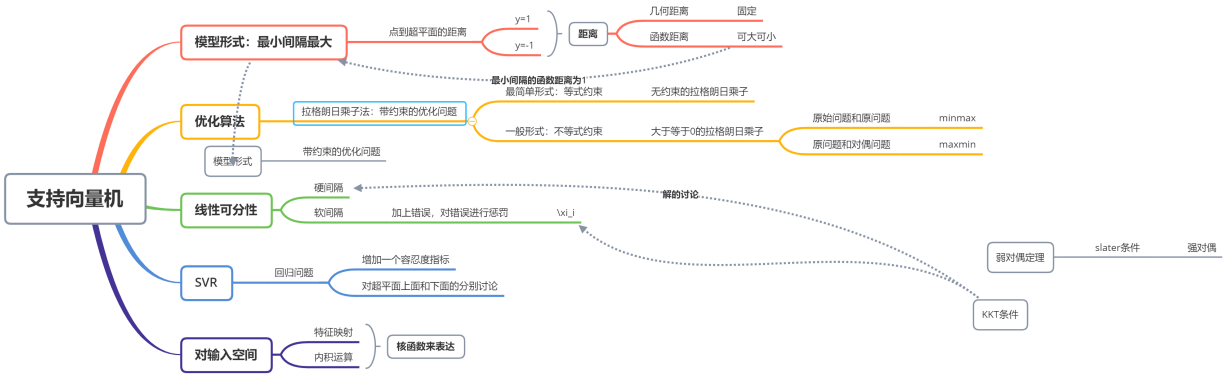


图 2: 支持向量机整体框架 (来自于李国文)

对其分别求偏导可得

$$
\begin{aligned}
w &= \sum_{i=1}^{N}(\hat{\alpha}-\alpha)x_i \\
0 &= \sum_{i=1}^{N}(\hat{\alpha}-\alpha) \\
C &= \alpha_i + \mu_i \\
C &= \hat{\alpha} + \hat{\mu}_i
\end{aligned}
$$

最终的 lagrange 函数可写成

$$
\max_{\alpha,\hat{\alpha}} \sum_{i=1}^{N}(\hat{\alpha}_i-\alpha_i) - \epsilon(\hat{\alpha}_i+\alpha_i) - \frac{1}{2}\sum_{i=1}\sum_{j=1}(\hat{\alpha}_i+\alpha_i)(\hat{\alpha}_j+\alpha_j)x_i x_j
$$

$$
s.t. = \begin{cases} \sum_{i=1}(\alpha-\alpha) = 0 \\ 0 \le \alpha_i, \alpha_j \le C \end{cases}
$$

分析：因为 $f(x_i) - y_i - \epsilon - \varepsilon = 0$ 和 $y_i - f(x_i) - \epsilon - \hat{\xi}_i = 0$ 不能成立，因此 $\hat{\alpha}$ 和 α_i 至少有一个为 0．而当 $\alpha \neq 0$ 时候，$f(x_i) - y_i - \epsilon - \xi_i = 0$ 时候对应的数据点是支持向量．而当，为自由变量．

5 神经元模型

5.1 感知机和多层网络

感知机（Perceptron）也是一个神经网络，是一个简单的由两层神经元组成的．在输入层接收到输入信号之后，传递给输出层．输出层对信号通过阈值的原理进行处理．因此我们可通过图可发现，感知机模

型是容易实现与或非的逻辑运算.
迭代的思路:

$$w_i := w + i + \Delta w_i$$

$$\Delta w_i = \eta(y - \hat{y})x_i$$

其中 $\eta \in (0, 1)$ 称为是学习率，也就是对应梯度下降的步长. 若 Perceptron 对已有的预测是准确的，则迭代不发生变化；反之则发生迭代. 同时感知机只有通过输出层进行激活函数的处理，即拥有一层功能的神经元，其学习能力是非常的有限的.

进一步，这样的一个简单的神经网络只是解决了一个与或非问题（非线性可分问题），而对于异或问题仍然悬而未决. “异或”是一个排他性的“或”，即当二者取值不同时为真 (TRUE)，当二者的取值相同时为假 (FLASE).

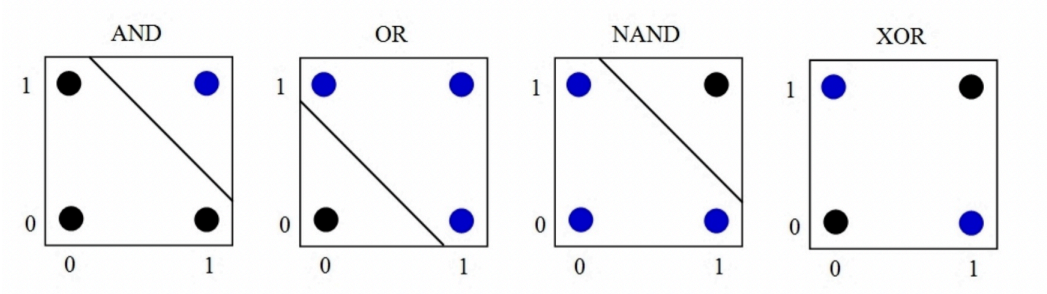


图 3: 异或函数的非线性决策边界

要解决非线性可分问题，需要考虑的是使用多层功能的神经元，通过中间层的神经元，经过两次以上的迭代就可以得到非线性的决策边界.

注释 5.1. 非线性的激活函数是关键：如果使用线性的激活函数，则无论叠加或嵌套多少次 *(*相当于复合函数*)*，所得结果还是线性函数.

5.1.1 激活函数

为何使用激活函数? 不使用激活函数，相当于线性组合，表达能力有限. 通过激活函数处理产生神经元的输出，能够将不同的权重进行更加多样化的表示. 一个常用的激活函数是 sigmoid 函数，可以在较大的范围内的数据压缩至 $y \in [0, 1]$ 范围内. 因此也称为是挤压函数.
而从纯数学的角度进行理解：只是将多个输入映射到同一个函数处理不同的参数 w_i. 比如 $f(\sum_i w_i x_i - \theta)$ 其中的 θ 即为阈值 (threshold)，即是否能激发神经元的下限. 一些激活函数类型：S 型激活函数

$$\Lambda(z) \equiv \frac{1}{1 + e^{-z}} \tag{21}$$

当其输出介于 0-1 分布，可以解释为概率分布. 其数学性质会好于跃迁函数. 但当输入靠近与两端 $(|z| \to \infty)$ 时，可能会使得其导数 $\Lambda(z)' \to 0$. 在实际训练中也就是产生所谓的”梯度消失“.
双曲正切函数:

$$Tanh(z) \equiv \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

Tanh 函数可看作是将 logistic 函数拉伸到 (-1,1) 区间，二者的关系如下：

$$Tanh(z) = 2\Lambda(2z) - 1$$

Tanh 的两端仍然是饱和的，依然可能发生梯度消失.

5.2 误差传播算法

我们的目标是为寻找一个计算前馈神经网络的误差函数 $E(\omega)$ 的梯度的一种高效方法. 我们可以看到可以使用局部信息传递的思想完成这一点. 局部信息传递中，信息在神经网络中交替向前、向后传播. 这种方法就称为误差反向传播 (error backpropagation).

5.2.1 误差函数导数的计算能力

多层的网络学习能力强于单层感知机的能力.
输入和输出： $D = \{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$, 其中 $x \in \mathbb{R}^d, y \in \mathbb{R}^l$ 输入实例是有 d 个属性描述，输出 l 维的实值向量.

- 输出层阈值：θ_j
- 输入层阈值：γ_h
- 输入层 i 与隐层 h 的连接权 v_{ih}
- 隐层 h 与输出层 j 的连接权为 w_{hj}
- 隐层 h 收到的信号：$\alpha_h = \sum_{i=1}^{h} v_{ih} x_i$
- 输出层 j 收到的信号：$\beta_j = \sum_{h=1}^{q} w_{hj} b_h$

在施加激活函数之前，记“净输入”为

$$z_i^{(l)} \equiv \sum_j w_{ji}^{(l)} a_j^{(l-1)}$$

其中的 a_j^{l-1} 表示的是第 l 层的第 i 的神经元的输出值

$$a_i^{(l)} = f\left(\sum_j w_{ji}^{(l)} a_j^{(l-1)}\right) \equiv f(z_j^{(l)})$$

所以其中的 a_j^{l-1} 上标-1 表示的是上一个隐层.

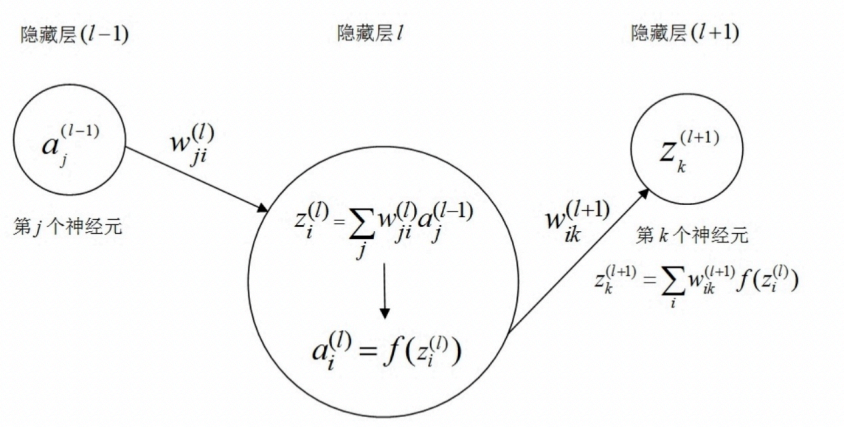


图 4: 误差反向传播的求导示意图

记神经网络的损失函数为 L．考虑损失值对 $w_{ji}^{(l)}$ 求导

$$\frac{\partial L}{\partial w_{ji}^{(l)}} = \frac{\partial L}{\partial z_i^{(l)}} \cdot \frac{\partial z_i^{(l)}}{\partial w_{ji}^{(l)}} \equiv \delta_i^{(l)} a_j^{(l)}$$

其中的 $\delta_i^{(l)} \equiv \frac{\partial L}{\partial z_i^{(l)}}$ 表示的是误差 (error)．若最终达到局部最小值即 $\delta_i^{(l)} = 0$，则不再更新．

具体而言，$z_i^{(l)}$ 影响到损失函数的途径是通过将传递到下一层 $(l+1)$ 来影响最终的输出．再次使用链式法则：

$$\delta_i^{(l)} \equiv \frac{\partial L}{\partial z_i^{(l)}} = \sum_k \frac{\partial L}{\partial z_i^{(l+1)}} \cdot \frac{\partial z_i^{(l+1)}}{\partial w_{ji}^{(l)}} = \sum_k \delta_k^{(l+1)} \cdot \frac{\partial z_k^{(l+1)}}{\partial z_i^{(l+1)}} \tag{22}$$

进一步由于 $z_k^{(l+1)} = \sum_i w_{ik}^{(l+1)} f(z_i^{(l)})$，使得

$$\frac{\partial z_k^{(l+1)}}{\partial z_i^{(l)}} = w_{ik}^{(l+1)} f'(z_i^{(l)})$$

带入上述公式中，可将求偏导进行消除得到

$$\delta_i^{(l)} = \sum_k \delta_k^{(l+1)} \cdot w_{ik}^{(l+1)} f'(z_i^{(l)}) = f'(z_i^{(l)}) \sum_k \delta_k^{(l+1)} \cdot w_{ik}^{(l+1)}$$

这样也就将第 l 层的误差用 l+1 层的误差来表示．这是一种反向递推的方式，最后再将其带入 w 对 L 求偏导的公式中可以得到 $\frac{\partial L}{\partial w_{ji}^{(l)}}$

计算误差时候需要先计算最后一层的误差，之后再计算倒数第二层．以此类推就可求得所有的神经元．

对于训练集 (x_k, y_k)，假定神经网络的输出为 $\hat{y_k} = (\hat{y_1^k}, \hat{y_2^k}, \cdots, \hat{y_n^k})$．
网络的 RMSE 为

$$E_k = \frac{1}{2} \sum_{j=1}^{l} (\hat{y_k^k} - y_j^k)^2$$

BP 采用的是传统的 GD 算法，给定学习率 η 下的迭代策略为：

$$\Delta w_{hj} = -\eta \frac{\partial E_k}{\partial w_{hj}}$$

而其中的

对于一个 *sigmoid* 函数来说，求导有一个良好的性质：

$$f'(x) = f(x)(1 - f(x))$$

根据上述式子我们可以得到：

$$\Delta w_{hj} = \eta g_j b_h$$

类似可得到

$$\Delta \theta_j = -\eta g_j$$
$$\Delta v_{ih} = \eta e_h x_i$$
$$\Delta \gamma_h = -\eta e_h$$

5.2.2 神经网络的正则化

定义 5.1. *在训练样本时候，所有的样本都使用了一遍称为轮 (one epoch). 经过一轮之后参数就会得到更新.*

包含多个隐藏层的深度神经网络是表达能力很强的模型 (very expressive models)，可学习输入与输出之间非常复杂的函数关系. 而若进行多轮可能会造成过拟合问题. 常用的方法是进行正则化.

- 早停 (Early Stopping)，意味着提前停止训练数据，而不需要等到神经网络的损失函数的最小值才停止. 具体操作上，可以使用交叉验证，将样本分为训练集、测试集、验证集.
- 丢包 (dropout)，随机的丢掉一些神经元
- 惩罚 (penalization) 类似于岭回归方法在目标函数中加入一个惩罚项.

5.2.3 早停

误差函数是一个关于迭代次数不增函数. 由于模型开始过拟合而逐渐增大. 因此，训练过程可以在关于验证集误差最小的点停止. 这种条件下，网络的行为可以通过网络的自由度来描述. 自由度有效数量一开始很少，随着训练过程中不断增长，对应于模型复杂度的持续增长. 在训练误差达到最小值之前停止训练就表示一个限制模型复杂度的方式.

5.2.4 不变性

对于输入变量进行一个或多个的变换之后，预测不应该发生变化．也即是应该具备不变性 (invariant)．一个特定的图像的类别应该与图像的位置无关，也和大小无关．即使原始数据产生了巨大的影响，也会得到同样的输出．

6 特征选择

6.1 PCA

在对一些问题时候，会出现维度爆炸的问题出现，因此我们试图通过减少特征/特征选取的方式，来减少使用的特征从而避免该问题的出现．

因此在这里使用 PCA(principle component Analysis) 的方法．而 PCA 的核心思想：而对于一个 PCA 的方法，对原始输入的空间进行重构，使得原相关的特征，变得只有不相关的部分．

6.1.1 统计量的矩阵表示

$$\bar{\boldsymbol{x}} = \frac{1}{N}\boldsymbol{x}^T\mathbb{I}_N \tag{23}$$

其中的 $\boldsymbol{x}$ 表示为一组向量．样本方差的表示，在之前我们学过关于一个协方差矩阵的表示方法：

$$\Sigma = \boldsymbol{x}\boldsymbol{x}^T$$

而实际上我们将数据进行输入，以真实的样本进行计算时候会得到不同的表示方式：

$$\begin{aligned} S &= \frac{1}{N}\Sigma_{i=1}^{N}(\boldsymbol{x}_i - \bar{\boldsymbol{x}})(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^T \\ &= \frac{1}{N}(\boldsymbol{x_1} - \bar{\boldsymbol{x}}, \boldsymbol{x_2} - \bar{\boldsymbol{x}}, \cdots, \boldsymbol{x_N} - \bar{\boldsymbol{x}})(\boldsymbol{x_1} - \bar{\boldsymbol{x}}, \boldsymbol{x_2} - \bar{\boldsymbol{x}}, \cdots, \boldsymbol{x_N} - \bar{\boldsymbol{x}})^T \\ &= \frac{1}{N}(X^T - \frac{1}{N}X^T\mathbb{I}_{N1}\mathbb{I}_{N1}^T)(X^T - \frac{1}{N}X^T\mathbb{I}_{N1}\mathbb{I}_{N1}^T)^T \\ &= \frac{1}{N}X^T H_N H_N X = \frac{1}{N}X^T H X \end{aligned}$$

其中的 $H = X^T - \frac{1}{N}X^T\mathbb{I}_{N1}\mathbb{I}_{N1}^T$，同时可简单的看出 $H^T = H$

6.1.2 最大投影方差

记得在原先学习 LDA 方法时候，需要将一组数据投影到新的一组基中，而我们的基的使用，其最终的目标在于将这些的类别最大程度的分离开．同样也是如此，我们希望构建一个正定的基底，将这些数据投影到不同的维度．

$$\hat{x}_i = \sum_{i=1}^{p}(u_i^T x_i)u_i = \sum_{i=1}^{q}(u_i^T x_i)u_i + \sum_{i=q+1}^{p}(u_i x_i)u_i \tag{24}$$

而新坐标下的方差表示：

$$J = \frac{1}{N}\sum_{i=1}^{p}\sum_{j=1}^{q}((x_i - \bar{x})^T u_j)^2 = \sum_{j=1}^{q} u_j^T S u_j$$
$$s.t. \quad u_j^T u_j = 1$$

通过刻画方差，我们能够显现出不同数据的特征，因此使得在变换之后的方差最大，使用拉格朗日来刻画这个最大化的优化函数，约束条件为：

$$\arg\max_{u_j} L(u_j, \lambda) = \arg\max_{u_j} u_j^T S u_j + \lambda(1 - u_j^T u_j)$$
$$S u_j = \lambda u_j$$

6.1.3 最小重构距离

我们可以去换一个视角：对于原有的观测值，在重新构建的过程中，并非完全的正交变换，因此我们可以提取出那些被正交的和非正交的被“剔除”的部分．此时的目标优化参数就变为将完全重构和部分重构的差值作为损失函数

$$\arg\min_{u_j} L(u_j, \lambda) = \arg\min_{u_j} u_j^T S u_j + \lambda(1 - u_j^T u_j) \tag{25}$$

6.2 特征值分解

对于一个矩阵，若它有 n 个线性无关的特征向量．

$$A = Q\Lambda Q^{-1}$$

其中，Λ 为对角矩阵，Q 为 n 阶矩阵，每一列所对应的是特征向量．

$$Q = (x_1, x_2, \cdots, x_n)$$

而一个矩阵可以被特征值分解的充要条件为它有 n 个特征向量．

正交化我们的特征矩阵．

$$Q^T = Q^{-1}$$

根据特征分解，可计算出

$$A^n = Q\Lambda^n Q^{-1}$$

对于一个广义的特征值分解，我们想要去分解一个任意的矩阵 $A \in R^{n\times m}$ 而我们知道：

$$A^T A \in \mathbb{R}^{m\times m}, AA^T \in \mathbb{R}^{n\times n}$$

都是一个方阵．因此可对其进行分解．首先证明上述两个矩阵有相同的非零特征值，首先取 λ 为后者的特征值：

$$AA^T x = \lambda x$$

左乘 A^T 可得到

$$A^T AA^T x = A^T \lambda x \Rightarrow A^T A(A^T x) = \lambda(A^T x)$$

因此我们可将上述的特征值进行分解得到

$$A^T A = V\Sigma^T \Sigma V^T$$

$$AA^T = U\Sigma^T \Sigma U^T$$

在中间分别乘上 $U^T U$ 和 $V^T V$，得到：

$$A = U\Sigma V$$

因此我们通过 SVD 分解得到的数据：

$$\begin{aligned} HX &= U\Sigma V^T \\ U^T U &= E_N \\ V^T V &= E_p \end{aligned}$$

$$\Sigma_{N\times p} = \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \vdots & \\ & & & \sigma_n \end{bmatrix}$$

因此方差 S 可表示为：

$$S = \frac{1}{N} X^T HX = \frac{1}{N} X^T H^T HX = \frac{1}{N} V\Sigma^T \Sigma V^T$$

在转换之后的坐标表示为：

$$HX \cdot V$$

7 聚类

对于聚类来说，属于一种无标签的学习方式，聚类性能度量亦称聚类有效性指标，与监督学习中的性能度量作用相似．性能度量的作用

- 对聚类结果，使用性能度量来评估其好坏
- 直接将性能度量作为聚类过程的优化目标

定义 7.1. 原型是指样本空间中具有代表性的点．原型聚类也称基于原型的聚类，此类算法假设聚类结构能通过一组原型刻画，在现实聚类中极为常用．

一般流程为先对原型进行初始化，然后对原型进行迭代更新求解．采用不同的原型表示、不同的求解方式，将产生不同的算法．

性质 7.1. 使用的性能度量方式是聚类有效性指标．而同样使用性能度量指标，我们就可以评价模型的好坏．另一方面，若使用的性能度量，能直接的作为聚类过程的优化目标，得到想要的聚类结果．

- 外部指标：于某一个参考模型进行比较．
- 内部指标：直接考察聚类结果、不考察任何模型．

Jaccard 系数：

$$JC = \frac{a}{a+b+c}$$

FM 指数 (Fowlkes and Mallows Index)

$$FMI = \sqrt{\frac{a}{a+b} + \frac{a}{a+c}}$$

其中的 $a, b, c, d \in$

Rand 指数 (Rand Index)

$$RI = \frac{2(a+d)}{m(m-1)}$$

其中的 m 是总数．

7.0.1 距离计算

一个聚类，自然就不是一个点，是一个块，对于这些块的集合，我们试图去找出度量他们距离的方式常用的是 Minkowski Distance．

$$dis_{mk}(x_i, x_j) = \left(\sum_{w=1}^{n} |x_{iu} - x_{ju}| \right)^{1/p}$$

7.1 VDM

对于无序数据的处理，我们同样可将其映射到一个距离函数上，

$$VDM_p(a,b) = \sum_{i=1}^{k} \left| \frac{m_{\mu,a,i}}{m_{\mu,a}} - \frac{m_{\mu,b,i}}{m_{\mu,b}} \right|^p \tag{26}$$

其中的 $m_{\mu,a}$ 表示的是在 μ 上取值为样本 a 的样本数．$m_{\mu,a,i}$ 表示的是在 i 个样本簇中的属性 μ 上取值为 a 的样本数．k 为样本簇数．因此由公式可发现，对于两个属性 a,b 之间的距离计算，是根据同一个簇内的样本占比的绝对值加和来得到．考虑一个极端的例子，ab 之间的所有样本在所有簇中的数量等同，我们可根据该判断其距离为 0．
进一步，将 Minsky 距离与 VDM 结合就可以得到 MinkovDMp，能用于处理混合属性．

$$MinkovDM_p(x_i, x_j) = \left(\sum_{\mu=1}^{n_c} \left| x_{i\mu} - x_{j\mu} \right|^p + \sum_{\mu=n_c+1}^{n} VDM_p(x_{i\mu} x_{j\mu}) \right)^{1/p} \tag{27}$$

当样本空间中不同属性的重要性不同时，可使用” 加权距离”，以加权闵可夫斯基距离为例：

7.2 学习向量量化

学习向量量化 (Learning Vector Quantization,LVQ) 也是试图找到一组原型向量来刻画聚类结构，但是与 k 均值不同的是，LVQ 假设样本数据带有类别标记，学习过程利用样本的这些监督信息来辅助聚类．
给定样本集

$$D = \{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$$

7.3 高斯混合与 EM 算法

详细推导可见该博客