

MA234 Introduction to Big Data Notes

MA234 Introduction to Big Data Notes

1. Introduction

- 机器学习 (Machine Learning) 的分类
- 数据表示 (PPT说的比较简洁清晰, 直接放截图了)
- 一些相关概念
- 风险最小化策略 (Risk Minimization Strategy)
- 模型评估 (Model Assessment)
- 过拟合 (Overfitting)

2. 数据预处理 (Data Preprocessing)

- 基本的统计概念
- 距离 (Distance)
 - 有关布尔变量的距离
 - 闵可夫斯基 (Minkowski) 距离
 - 余弦相似度 (Cosine Similarity)
- 数据缩放 (Data Scaling)
 - 数据缩放的原因
 - 4种数据放缩
- 数据离散化 (Data Discretization)
 - 离散化的原因
 - 分类
- 数据冗余 (Data Redundancy)
- 数据缺失 (Missing Data)
 - 数据缺失的原因
 - 应对方法
 - 哑变量 (Dummy Variables)
- 离群值 (Outlier)
 - 离群值检测
 - 相关概念

3. 分类1 (Classification)

- kNN (k-Nearest Neighbour)
 - 调参k —— 交叉验证 M-fold Cross-validation
 - 分析
- 决策树 (Decision Tree)
 - 不纯度 (Impurity) 度量 - GINI 指数 (GINI Index)
 - 不纯度度量 - 信息增益 (Information Gain)
 - 不纯度度量 - 误分类误差 (Misclassification Error)
 - 决策树相关算法
 - 优缺点
- 朴素贝叶斯 (Naive Bayes)
 - 贝叶斯定理
 - Naive Bayes Method
 - 优缺点
- 分类算法的模型评估
 - 混淆矩阵 (Confusion Matrix)
 - Receiver Operating Characteristic (ROC) and AUC
 - Kappa Coefficient
- 多分类问题

4. 回归 (Regression)

- 线性回归 (Linear Regression)
 - (平凡)最小二乘 [(Ordinary) Least Square, OLS]
 - 评价指标
- 多元共线性 (Multicollinearity)

Bias-Variance Decomposition
正则化 (Regularization)
Best-subset selection
使用惩罚项 (Penalties) 正则化
岭回归 (Ridge Regression)
从贝叶斯视角看岭回归
岭轨迹 (Ridge Trace)
LASSO 回归
最大后验估计 [Max A Posterior (MAP) Estimation]
回归模型的评价

5. 分类2

逻辑回归 (Logistic Regression)
线性判别分析 [Linear Discriminant Analysis (LDA)]
神经网络 (Neural Network)
深度学习 (Deep Learning)
感知器：正向传播 (The Perceptron : Forward Propagation)
神经网络算法
反向传播过程 (Back Propagation Procedure)
梯度下降算法 (Gradient Descent)
支持向量机 (Support Vector Machine, SVM)
线性 SVM
使用拉格朗日乘子法 (Method of Lagrange Multipliers) 解决
KKT 条件
SMO (Sequential Minimal Optimization) 算法
软边距
非线性 SVM
SVM 的优缺点

6. 集成学习 (Ensemble Learning)

集成学习的分类
随机森林 (Random Forest)
决策树的缺点
随机森林算法
随机森林的模型评估
随机森林的 Feature Importance
随机森林的优缺点
AdaBoost
使用 Boost 拟合叠加模型
AdaBoost 算法
AdaBoost 优缺点
Gradient Boosting Decision Tree (GBDT)

Boosting Tree
GBDT
Feature importance
GBDT 优缺点

XGBoost
节点分割：贪心算法

集成学习总结

7. 聚类 (Clustering)

K-Means
转化为优化问题
不相似度
K-Means (as Central Voronoi Tessellation)
K 的选取
K-Means 优缺点
K-Means 的变体
Bisecting K-means
K-medoids
其他变种

分层群聚 (Hierarchical Clustering)

凝聚聚类

分散聚类

分层聚类优缺点

DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

概念

DBSCAN算法

DBSCAN 优缺点

DBSCAN vs K-Means

最大期望算法 [Expectation-Maximization (EM) Algorithm]

谱聚类 (Spectral Clustering)

谱聚类

聚类模型评估

Purity

Confusion Matrix

Jaccard Coefficient

Rand index

Mutual Information (简略)

Davies-Bouldin Index

Silhouette Coefficient

8. 降维 Dimensionality Reduction

PCA

PCA

样本协方差矩阵的特征分解

PCA 算法

LDA

PCA 与 LDA 对比

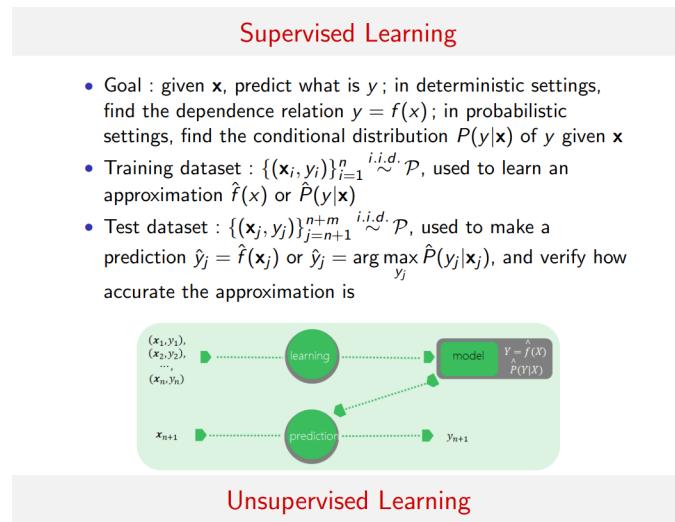
非线性降维

1. Introduction

机器学习 (Machine Learning) 的分类

- 监督学习 (supervised learning): 利用一组已知类别的样本调整分类器的参数, 使其达到所要求性能
 - 分类 (classification): 输出是离散的
 - 回归 (regression): 输出是连续的
- 无监督学习 (unsupervised learning): 根据类别未知(没有被标记)的训练样本解决模式识别
 - 密度估计 (density estimation)
 - 聚类 (clustering): 比如 K-Means, SOM
 - 降维 (dimensionality reduction)
- 半监督学习 (semi-supervised learning): 存在缺失数据
 - 比如填补有缺失的图像等
- 强化学习 (reinforcement learning)
 - 对抗类游戏的人工智能

数据表示 (PPT说的比较简洁清晰, 直接放截图了)



- Goal : in probabilistic settings, find the distribution $P(\mathbf{x})$ of \mathbf{x} and approximate it ; there is no y
- Training dataset : $\{\mathbf{x}_i\}_{i=1}^n \stackrel{i.i.d.}{\sim} \mathcal{P}$, used to learn an approximation $\hat{P}(\mathbf{x})$; no test data in general



一些相关概念

- Decision function (hypothesis) space :

$$\mathcal{F} = \{f_\theta | f_\theta = f_\theta(\mathbf{x}), \theta \in \Theta\} \text{ or } \mathcal{F} = \{P_\theta | P_\theta = P_\theta(y|\mathbf{x}), \theta \in \Theta\}$$
- Loss function : a measure for the “goodness” of the prediction, $L(y, f(\mathbf{x}))$
 - 0-1 loss : $L(y, f(\mathbf{x})) = I_{y \neq f(\mathbf{x})} = 1 - I_{y = f(\mathbf{x})}$
 - Square loss : $L(y, f(\mathbf{x})) = (y - f(\mathbf{x}))^2$
 - Absolute loss : $L(y, f(\mathbf{x})) = |y - f(\mathbf{x})|$
 - Cross-entropy loss :

$$L(y, f(\mathbf{x})) = -y \log f(\mathbf{x}) - (1 - y) \log(1 - f(\mathbf{x}))$$
- Risk : in average sense,

$$R(f) = E_P[L(y, f(\mathbf{x}))] = \int_{\mathcal{X} \times \mathcal{Y}} L(y, f(\mathbf{x})) P(\mathbf{x}, y) d\mathbf{x} dy$$
- Target of learning : choose the best f^* to minimize $R_{exp}(f)$,

$$f^* = \min_f R_{exp}(f)$$

风险最小化策略 (Risk Minimization Strategy)

- Empirical risk minimization (ERM) : given training set $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, $R_{emp}(f) = \frac{1}{n} \sum_{i=1}^n L(y_i, f(\mathbf{x}_i))$
 - By law of large number, $\lim_{n \rightarrow \infty} R_{emp}(f) = R_{exp}(f)$
 - Optimization problem : $\min_{f \in F} \frac{1}{n} \sum_{i=1}^n L(y_i, f(\mathbf{x}_i))$
- Structural risk minimization (SRM) : given training set $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ and a complexity functional $J = J(f)$,
$$R_{srm}(f) = \frac{1}{n} \sum_{i=1}^n L(y_i, f(\mathbf{x}_i)) + \lambda J(f)$$
 - $J(f)$ measures how complex the model f is, typically the degree of complexity
 - $\lambda \geq 0$ is a tradeoff between the empirical risk and model complexity
 - Optimization problem : $\min_{f \in F} \frac{1}{n} \sum_{i=1}^n L(y_i, f(\mathbf{x}_i)) + \lambda J(f)$

模型评估 (Model Assessment)

假设我们预测出了 $y = \hat{f}(\mathbf{x})$, 那么有以下 3 种误差 (error), 其中 $L(y, f(\mathbf{x}))$ 代表损失函数 (见上面) :

- 训练误差 (Training error): $R_{emp}(\hat{f}) = \frac{1}{n} \sum_{i=1}^n L(y_i, \hat{f}(\mathbf{x}_i))$, 说明了该问题的学习难度
- 测试误差 (Test error): $e_{test}(\hat{f}) = \frac{1}{m} \sum_{j=n+1}^{n+m} L(y_j, \hat{f}(\mathbf{x}_j))$, 说明了问题的预测能力
 - error rate: e_{test}
 - accuracy: r_{test}
 - $e_{test} + r_{test} = 1$
- 泛化误差 (Generalization error): $R_{exp}(\hat{f}) = E_p[L(y, \hat{f}(\mathbf{x}))] = \int_{\mathcal{X} \times \mathcal{X}} L(y, \hat{f}(\mathbf{x})) P(\mathbf{x}, y) d\mathbf{x} dy$, 刻画了学习算法的经验风险与期望风险之间偏差和收敛速度

过拟合 (Overfitting)

过多模型参数导致, 对训练集更好, 但对测试集更差

应对方法:

- 正则化 (Regularization): $\min_{f \in F} \frac{1}{n} \sum_{i=1}^n L(y_i, f(\mathbf{x}_i)) + \lambda J(f)$, 加入惩罚项 $J(f)$, 模型参数越多, $J(f)$ 越大, 选择合适的 λ 可同时将经验风险和模型复杂性降至最低
- 交叉验证 (Cross-validation, CV): 将训练集拆分为训练子集和验证子集, 使用训练集重复训练不同的模型, 使用验证集选择验证误差最小的模型
 - Simple CV: 将数据随机拆分为两个子集
 - K-fold CV: 将数据随机拆分为大小相同的 K 个不相交子集, 将 $K - 1$ 个子集的并集视为训练集, 将另一个子集视为验证集, 重复执行此操作并选择平均验证误差最小的模型

- Leave-one-out CV: 在上一种情况下取 $K = n$

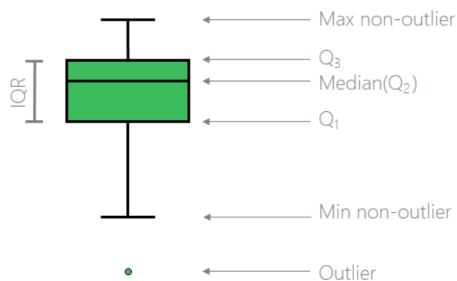
2. 数据预处理 (Data Preprocessing)

数据种类: 表格数据 (Tabular data) (矩阵、向量、对象、关系), 图形数据 (Graphical data) (网络、图形), 多媒体数据 (Multi-media data) (文本、图像、视频、音频)

属性 (Attributes) 类型: 离散的、连续的

基本的统计概念

- 平均数 (Mean)
- 中位数 (Median)
- 最大值 (Maximum), 最小值 (Minimum)
- 分位数 (Quantile): 中位数的推广, k^{th} q-分位数 x_q : $P[\mathbf{X} < x_q] \leq k/q$
四分位距 (interquartile range): $IQR = Q3(75\%) - Q1(25\%)$
- 方差 (Variance) $Var(\mathbf{X})$, 标准差 (Standard deviation)
- 众数 (Mode)
- 箱形图 (Box Plot): 如下图示例



距离 (Distance)

有关布尔变量的距离

对于布尔变量, 有 4 种距离

- symmetric distance $d(\mathbf{x}_i, \mathbf{x}_j) = \frac{r+s}{q+r+s+t}$
- Rand index $Sim_{Rand}(\mathbf{x}_i, \mathbf{x}_j) = \frac{q+t}{q+r+s+t}$
- non-symmetric distance $d(\mathbf{x}_i, \mathbf{x}_j) = \frac{r+s}{q+r+s}$
- Jaccard index $Sim_{Jaccard}(\mathbf{x}_i, \mathbf{x}_j) = \frac{q}{q+r+s}$

		Sample j		
		1	0	sum
Sample i	1	q	r	$q+r$
	0	s	t	$s+t$
		$q+s$	$r+t$	p

闵可夫斯基 (Minkowski) 距离

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt[h]{\sum_{k=1}^p |\mathbf{x}_{ik} - \mathbf{x}_{jk}|^h}$$

上述距离被称作 L_h 范数 (norm)

- 正定: $d(\mathbf{x}_i, \mathbf{x}_j) \geq 0$ 等号成立当且仅当 $i = j$
- 对称: $d(\mathbf{x}_i, \mathbf{x}_j) = d(\mathbf{x}_j, \mathbf{x}_i)$
- 三角不等式: $d(\mathbf{x}_i, \mathbf{x}_j) \leq d(\mathbf{x}_i, \mathbf{x}_k) + d(\mathbf{x}_k, \mathbf{x}_j)$

h	名称
1	曼哈顿距离 (Manhattan distance)
2	欧式距离 (Euclidean distance)
∞	Supremum distance

余弦相似度 (Cosine Similarity)

$$\cos(\mathbf{x}_i, \mathbf{x}_j) = \frac{\mathbf{x}_i \cdot \mathbf{x}_j}{\|\mathbf{x}_i\| \|\mathbf{x}_j\|}$$

数据缩放 (Data Scaling)

数据缩放的原因

- 为更好的性能: 比如 SVM 中的 RBF 和 Lasso/岭回归中的惩罚项, 需假设均值为 0 方差为 1
- 规范化不同的维度: 身高 (1.75m) 和体重 (70kg)

4种数据放缩

- Z-score scaling: $\mathbf{x}_i^* = \frac{\mathbf{x}_i - \hat{\mu}}{\hat{\sigma}}$, $\hat{\mu}$ 样本均值, $\hat{\sigma}$ 样本方差, 适用于最大值和最小值未知且数据分布良好
- 0-1 scaling: $\mathbf{x}_i^* = \frac{\mathbf{x}_i - \min_k \mathbf{x}_k}{\max_k \mathbf{x}_k - \min_k \mathbf{x}_k} \in [0, 1]$, 适用于有界数据集, 新加入数据要重新计算最大值最小值
- Decimal scaling: $\mathbf{x}_i^* = \frac{\mathbf{x}_i}{10^k}$, 适用于适用于不同幅度的数据
- Logistic scaling: $\mathbf{x}_i^* = \frac{1}{1+e^{-\mathbf{x}_i}}$, 适用于数据集中在原点附近, 但会改变数据的分布

数据离散化 (Data Discretization)

离散化的原因

- 提高鲁棒性 (robustness): 通过将异常值放入某些区间来消除异常值
- 能够对算法提供更合理的解释
- 降低存储和计算消耗

分类

- 无监督离散化 (Unsupervised discretization)
 - 等距离离散化 (Equal-distance discretization)
 - 等频离散化 (Equal-frequency discretization)
 - 基于聚类的离散化 (Clustering-based discretization): 进行分层聚类并得到分层结构 (如使用 K-Means)，并将同层中的样本放入相同的区间 (比如家谱)
 - 基于 3σ 的离散化 (3σ -based discretization): 将样本分成 8 个间隔，需要先取对数
- 监督离散化 (Supervised discretization)
 - 基于信息增益的离散化 (Information Gain): 使用决策树进行分类
 - 卡方分箱离散化 (Chi Merge): 比较复杂一点，[参考链接](#)

数据冗余 (Data Redundancy)

- 关联性强的变量 (Attribute): 比如“出生日期”和“年龄”
- 通过关联分析确定数据冗余

- 对于连续变量 A 和 B，计算相关系数 $\rho_{A,B} = \frac{\sum_{i=1}^k (a_i - \bar{A})(b_i - \bar{B})}{k\hat{\sigma}_A\hat{\sigma}_B} \in [-1, 1]$
- 对于离散变量 A 和 B，计算 χ^2 ，越大说明关联性越小

数据缺失 (Missing Data)

数据缺失的原因

- Missing Completely At Random (MCAR): 缺失数据的出现是随机事件
- Missing At Random (MAR): 取决于一些控制变量，例如，在青少年调查中，年龄大于 20 的样本会被排除
- Missing Not At Random (MNAR): 例如，表现不佳的员工被解雇后缺少数据

应对方法

- 简单的方法：删除样本（适用于少量样本有数据缺失），删除变量（适用于某一变量缺失值较多）
- 填补
 - 填补 0
 - 用数字类型的均值填充，用非数字类型的模数填充，**适用于MCAR**
 - 缺点：集中于均值而低估方差
 - 解决方案：填写不同的组
 - 用相似变量填充：引入自相关 (auto-correlation)
 - 用历史数据填充
 - K-Means 填充：使用完善的变量（无缺失值）计算数据的成对距离，然后用前 K 个最相似的完善数据的平均值填充缺失值，将自相关引入
 - 用期望最大化 (EM) 填充：引入隐藏变量并使用 MLE 估计缺失值
 - 随机填充 (Random filling)
 - Bayesian Bootstrap, Approximate Bayesian Bootstrap (略)

- 基于模型的方法 (Model based methods): 将缺失变量视为 y , 将其他变量视为 x , 将没有缺失值的数据作为我们的训练集来训练分类或回归模型, 将缺失值的数据作为测试集来预测缺失值
- 插值填充 (Filling by Interpolation)

哑变量 (Dummy Variables)

哑变量, 也称为虚拟变量或名义变量, 是一种用于表示分类变量的变量类型。哑变量通常取值为 0 或 1, 用来反映某个变量的不同属性。在一个有 n 个分类属性的自变量中, 通常需要选择一个分类作为参照, 从而产生 $n-1$ 个哑变量。

离群值 (Outlier)

数据点看似来自不同的分布, 或噪声数据, 通常采用无监督检测

离群值检测

- 上下 α 分位数之外的样本 (取较小的 α , 通常是 1%)
- 从箱形图观察
- 局部异常值因子 (LOF): 是一种基于密度的方法

计算每个点 x 的密度, 将每个点 x 的密度与其相邻点的密度进行比较

相关概念

Computing Density by Distance

Some definitions :

- $d(A, B)$: distance between A and B;
- $d_k(A)$: k -distance of A, or the distance between A and the k -th nearest point from A
- $N_k(A)$: k -distance neighborhood of A, or the points within $d_k(A)$ from A;
- $rd_k(B, A)$: k -reach-distance from A to B, the repulsive distance from A to B as if A has a hard-core with radius $d_k(A)$,
 $rd_k(B, A) = \max\{d_k(A), d(A, B)\}$; note that $rd_k(A, B) \neq rd_k(B, A)$, which implies that k -reach-distance is not symmetric.

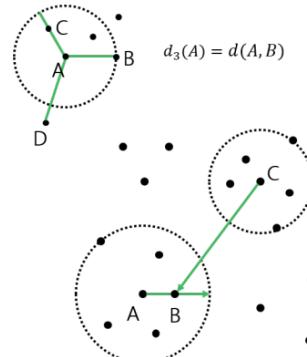
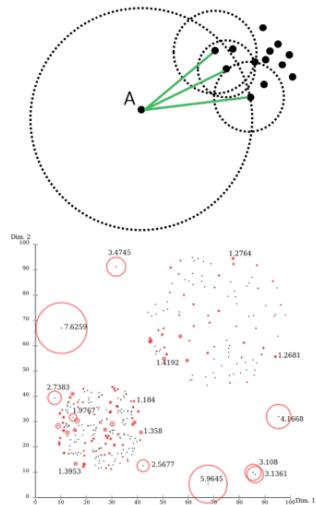


FIGURE: $rd_5(B, A) = d_5(A)$ and $rd_5(B, C) = d(B, C)$

Local Outlier Factor

Some definitions :

- $lrd_k(A)$: local reachability density is inversely proportional to the average distance,
$$lrd_k(A) = 1 / \left(\frac{\sum_{O \in N_k(A)} rd_k(A, O)}{|N_k(A)|} \right)$$
; intuitively, if for most $O \in N_k(A)$, more than k points are closer to O than A is, then the denominator is much larger than $d_k(A)$ and $lrd_k(A)$ is small; e.g., $k = 3$ in the figure
- $LOF_k(A)$: local outlier factor,
$$LOF_k(A) = \frac{\sum_{O \in N_k(A)} lrd_k(O)}{lrd_k(A)}$$
;
- $LOF_k(A) \ll 1$, the density of A is locally higher, dense point;
 $LOF_k(A) \gg 1$, the density of A is locally lower, probably outlier



3. 分类1 (Classification)

分类是一种监督学习，简而言之是对样本 \mathbf{x} 预测它的标签 y

训练阶段：给定数据集 $D = \{(\mathbf{x}, y)\}$, 分割为 $D = D_{train} \cup D_{test}$, 找一个能最佳的关联 \mathbf{x}_{train} 和 y_{train} 的函数 $y = f(\mathbf{x})$, 然后测试这个函数能在多大程度上适配 \mathbf{x}_{test} 和 y_{test}

预测阶段：将训练得到的函数用于没有标签的样本 \mathbf{x}_{pred} , 得到预测值 $y_{pred} = f(\mathbf{x}_{pred})$

kNN (k-Nearest Neighbour)

对于待预测的 \mathbf{x} , 找和它相邻的 k 个点, 统计它们的标签, 数量最多的标签作为 \mathbf{x} 的标签。

```
Compute d(x, x_j) for each (x_j, y_j) in D_train
Sort the distances in an ascending order, choose the first k samples (x_1, y_1), ..., (x_k, y_k)
Make majority vote y_pred = Mode(y_1, ..., y_k)
```

特点: 是最简单的监督学习算法, 同时进行训练和测试, 低偏差 (bias, 预测值和真实值之间的误差)、高方差 (variance, 预测值之间的离散程度)

优点: 对异常值不敏感, 易于实现和并行化, 适用于大型训练集, 适用于对数据的先验知识非常有限的情况

缺点: 需要调参 k , 占用储存空间大, 计算量大且密集

调参 k —— 交叉验证 M-fold Cross-validation

将数据集分为 M 折 (通常取 $M = 5$ 或 10) , 设 $\kappa : \{1, \dots, N\} \rightarrow \{1, \dots, M\}$ 是随即分区索引映射, 那么预测误差的 CV 估计值为 $CV(\hat{f}, k) = \frac{1}{N} \sum_{i=1}^N L(y_i, \hat{f}^{-\kappa_i}(\mathbf{x}_i, k))$

分析

时间复杂度: $O(mndK)$, 其中 n 是训练样本的数量, m 是测试样本的数量, d 是维度, K 是最近邻参数

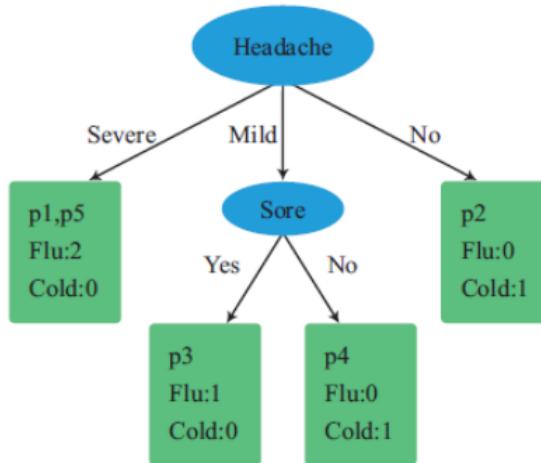
误差: 假设样本是 i.i.d (独立同分布) 的, 对于任何测试样本 \mathbf{x} 和足够小的 δ , 总存在一个训练样本 $\mathbf{z} \in B(\mathbf{x}, \delta)$ (\mathbf{x} 的标签与 \mathbf{z} 的标签相同) , 则 1NN 误差为

$$\epsilon = \sum_{c=1}^C p_c(x)(1 - p_c(z)) \xrightarrow{\delta \rightarrow 0} 1 - \sum_{c=1}^C p_c^2(x)$$

决策树 (Decision Tree)

简介略, 如图

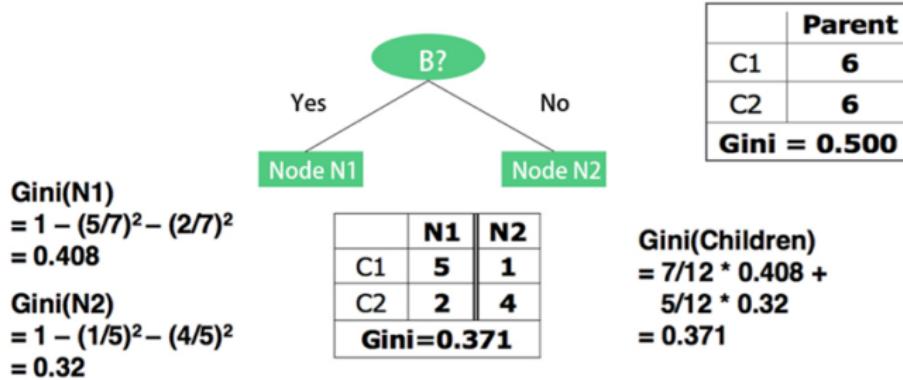
但要注意, 尽量将整个数据集划分成的每个部分所含杂质尽量少, 下面介绍 3 种不纯度度量



不纯度 (Impurity) 度量 - GINI 指数 (GINI Index)

- 节点 (Node) t 的 Gini 指数: $Gini(t) = 1 - \sum_{c=1}^C (p(c|t))^2$, $p(c|t)$ 是节点 t 中 c 类数据的比例
- $Gini(t)$ 的最大值是 $1 - \frac{1}{C}$, 在 $p(c|t) = \frac{1}{C}$ 时取到; 最小值是 0, 在对于某些 c 有 $p(c|t) = 1$ 时取到

- 一条分支 (Split) 的 Gini 指数: $Gini_{split} = \sum_{k=1}^K \frac{n_k}{n} Gini(k)$, 其中 n_k 是子节点 k 的样本数量,
$$n = \sum_{k=1}^K n_k$$
- 选择使得 $Gini(t) - Gini_{split}$ 最大的分支



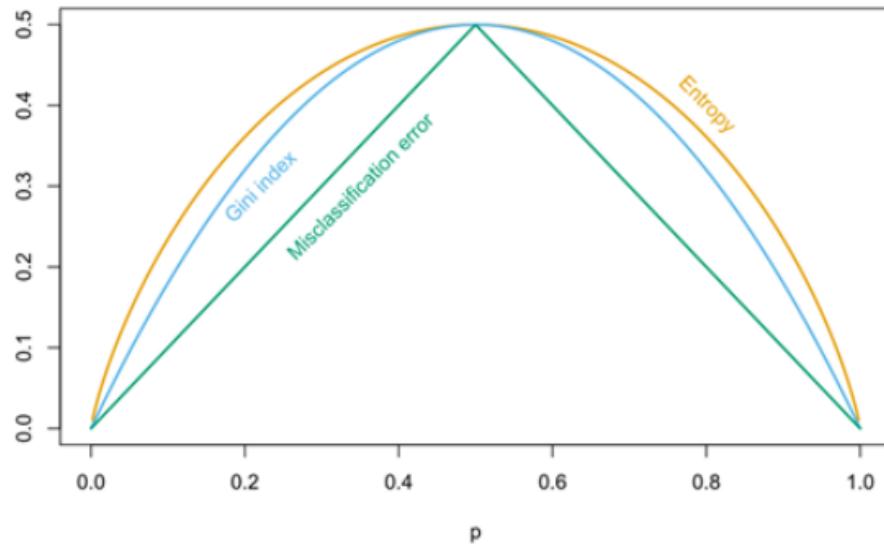
不纯度度量 - 信息增益 (Information Gain)

- 节点 t 的熵 (Entropy): $H(t) = - \sum_{c=1}^C p(c|t) \log_2 p(c|t)$
- $H(t)$ 的最大值是 $\log_2 C$, 在 $p(c|t) = \frac{1}{C}$ 时取到; 最小值是 0, 在对于某些 c 有 $p(c|t) = 1$ 时取到
- 信息增益: $InfoGain_{split} = H(t) - \sum_{k=1}^K \frac{n_k}{n} H(k)$, 其中 n_k 是子节点 k 的样本数量, $n = \sum_{k=1}^K n_k$
- 引入信息增益比 (Introduce information gain ratio): $SplitINFO = - \sum_{k=1}^K \frac{n_k}{n} \log_2 \frac{n_k}{n}$,
 $InfoGainRatio = \frac{InfoGain_{split}}{SplitINFO}$ (C4.5 算法)
- 选择使得 $InfoGain_{split}$ 最大的分支 (ID3 算法)
- 缺点: 容易生成过多的子节点导致过拟合

不纯度度量 - 误分类误差 (Misclassification Error)

- 节点 t 的误分类误差: $Error(t) = 1 - \max_c p(c|t)$
- 最大值是 $1 - \frac{1}{C}$, 在 $p(c|t) = \frac{1}{C}$ 时取到; 最小值是 0, 在对于某些 c 有 $p(c|t) = 1$ 时取到

例: 对于二分类问题, 3 种不纯度度量如下图



决策树相关算法

算法	属性类型	不纯度度量	分割的子节点数量	目标属性类型
Iterative Dichotomiser 3 (ID3)	离散型	信息增益	$k \geq 2$	离散型
C4.5	离散型、连续型	信息增益率	$k \geq 2$	离散型
C5.0	离散型、连续型	信息增益率	$k \geq 2$	离散型
Classification and Regression Tree (CART)	离散型、连续型	Gini 指数	$k = 2$	离散型、连续型

ID3 Algorithm

- Input : training set $D = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$, $Y = \{y_1 \dots, y_n\}$, set of features $F = \{\text{column variables of } X = (\mathbf{x}_1 \dots \mathbf{x}_n)^T\}$
 - Output : decision tree T
1. Create a root node
 2. Check Y : if all are positive, then return a single node tree T with label "+" ; if all are negative, then return a single node tree T with label "-"
 3. Check F : if empty, then return a single node tree T with label as majority vote of Y
 4. For each feature in F , compute information gain, choose the feature $A \in F$ which maximizes information gain as root
 5. For $A = i$, let $D(i) = \{(\mathbf{x}_j, y_j) \in D | x_{jA} = i\}$:
 - 5.1 If $D(i) = \emptyset$, then create a leaf node and make majority vote of D as the label
 - 5.2 Else, let $D = D(i)$, go back to step 1 iteratively

对于较为复杂的树，可以采用剪枝 (Pruning) 的方法减小其复杂程度

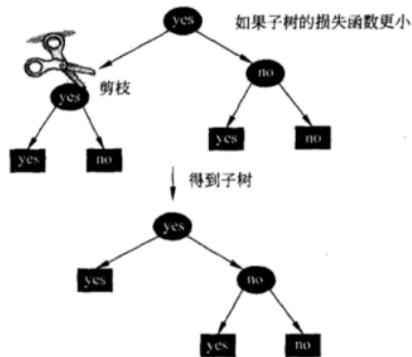
Tree Pruning

- Too complex tree structure easily leads to overfitting
- Prepruning : set threshold δ for impurity decrease in splitting a node ; if $\Delta Impurity_{split} > \delta$, do splitting, otherwise stop

- Postpruning : based on cost function

$$Cost_\alpha(T) = \underbrace{\sum_{t=1}^{|T|} n_t Impurity(t)}_{\text{data fidelity}} + \underbrace{\alpha |T|}_{\text{model complexity}}$$

- Input : a complete tree T , α
- Output : postpruning tree T_α
 1. Compute $Impurity(t)$ for $\forall t$
 2. Iteratively merge child nodes bottom-up : T_A and T_B are the trees before and after merging, do merging if $Cost_\alpha(T_A) \geq Cost_\alpha(T_B)$



优缺点

优点：

- 容易解释和可视化：广泛应用于金融、医疗健康、生物等领域
- 易于处理缺失值（视为新数据类型）
- 可以扩展到回归

缺点：

- 由于采用贪心算法，容易得到局部最小值
- 决策的边界过于简单：例如与轴平行的线

朴素贝叶斯 (Naive Bayes)

基于贝叶斯定理和样本的条件独立假设而进行分类的算法

贝叶斯定理

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

$P(Y)$ 先验 (prior) 概率分布, $P(X|Y)$ 似然 (likelihood) 函数, $P(X)$ 论据 (evidence), $P(Y|X)$ 后验 (posterior) 概率分布

Naive Bayes Method

机器学习的目的是估计 $P(Y|X)$

假设 $X = \{X_1, \dots, X_d\}$ (X 是 d 维的)

对于给定的 $X = x$, $P(X = x)$ 对于 Y 是独立的, 由贝叶斯定理:

$$P(Y|X = x) \propto P(X = x|Y)P(Y)$$

假设 X_1, \dots, X_d 是独立的, 对于给定的 $Y = c$:

$$P(X = x|Y = c) = \prod_{i=1}^d P(X_i = x_i|Y = c)$$

那么, 朴素贝叶斯算法就是以下面这个方式求 y 的预测值 \hat{y}

$$\hat{y} = \arg \max_c P(Y = c) \prod_{i=1}^d P(X_i = x_i|Y = c)$$

对于数据集 $D = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$ 需要估计 $P(Y = c)$ 和 $P(X_i = x_i|Y = c)$, 使用极大似然估计 (MLE)

MLE 估计 $P(Y = c)$: $P(Y = c) = \frac{\sum_{i=1}^n I(y_i=c)}{n}$

当 X_i 是离散的, 且值域为 $\{v_1, \dots, v_k\}$, $P(X_i = v_k|Y = c) = \frac{\sum_{i=1}^n I(x_i=v_k, y_i=c)}{\sum_{i=1}^n I(y_i=c)}$

当 X_i 是连续的, 假设服从 $N(\mu, \sigma^2)$, $P(X_i = x|Y = c) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$, MLE 估计参数 μ 和 σ

优缺点

优点

- 对于离群值和缺失值表现较为稳定
- 鲁棒性: 用于不相关的变量, $P(X|Y)$ 与 Y 无关, 因此对后验概率没有影响
- 即使不满足条件独立性假设, 也可能优于更复杂的替代方案

缺点

- 违反条件独立性假设时, 朴素贝叶斯的性能可能会更差
- 很大程度上取决于参数估计值的准确性

分类算法的模型评估

混淆矩阵 (Confusion Matrix)

二分类问题的混淆矩阵示例

真实标签	预测结果	
	1 (正例)	0 (反例)
1 (正例)	TP (真正例)	FN (假反例)
0 (反例)	FP (假正例)	TN (真反例)

TP: true positive, TN: true negative, FP: false positive, FN: false negative

Accuracy = $\frac{TP+TN}{TP+FP+FN+TN}$, 当样本分布不平衡时不是一个很好的参考指标

$$\text{Precision} = \frac{TP}{TP+FP}$$

$$\text{Recall} = \frac{TP}{TP+FN}, \text{ 在医学领域很重要}$$

$$\text{Specificity} = \frac{TN}{FP+TN}, \text{ 负样本的 Recall}$$

$$\text{F Score: } F_{\beta} = \frac{(1+\beta^2)\text{Precision} \times \text{Recall}}{\beta^2 \times \text{Precision} + \text{Recall}}$$

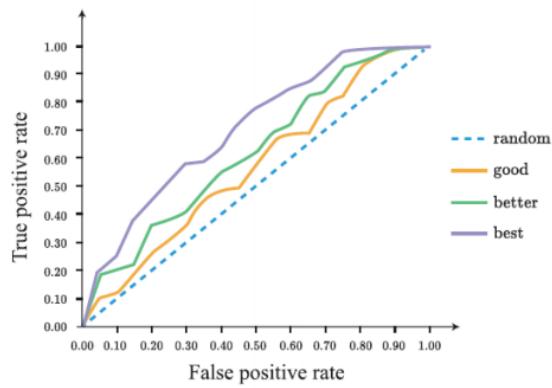
Receiver Operating Characteristic (ROC) and AUC

目的是解决不同的类的分布不均衡。

为连续的预测值设置不同的阈值 t , 比如若 $P(Y=1|X=X_i) > t$, 则 $\hat{y}_i = 1$

对不同的 t 值计算 $TPR = \frac{TP}{TP+FN}$, $FPR = \frac{FP}{FP+TN}$ 并绘制 ROC 曲线, ROC 越大, 说明表现越好

AUC: ROC 曲线下的面积, 越大越好, > 0.75 说明性能极佳



Kappa Coefficient

Cohen's Kappa Coefficient

- $\kappa = \frac{p_o - p_e}{1 - p_e} = 1 - \frac{1 - p_0}{1 - p_e}$ measures the agreement between two raters
- p_o is the accuracy (or the relative observed agreement)
- p_e is the hypothetical probability of chance agreement,
$$p_e = \sum_{c=1}^C \frac{n_c^{pred}}{N} \frac{n_c^{true}}{N}$$
, where n_c^{pred} is the number of samples predicted in class c , n_c^{true} is the true number of samples in class c , N is the total number of samples
- Eg : $p_o = \frac{20+15}{50} = 0.7$, $p_e = \frac{25}{50} \times \frac{20}{50} + \frac{25}{50} \times \frac{30}{50} = 0.5$, $\kappa = 0.4$

		Predicted Label		
		1	0	Total
True Label	1	20 TP	10 FN	30 C
	0	5 FP	15 TN	20 D
	Total	25 A	25 B	50 N

- $\kappa \in [-1, 1]$
- $\kappa = 1$: perfect agreement between two raters
- $\kappa = -1$: completely disagreement
- $\kappa = 0$: no agreement among the raters other than what would be expected by chance
- $\kappa < 0$: worse than random
- $\kappa > 0$: the result is meaningful, agree more as κ gets larger
- $\kappa \geq 0.75$: good performance
- $\kappa < 0.4$: bad performance

多分类问题

- ROC 和 AUC 无法生效
- 混淆矩阵变为 $C \times C$, 每个格子表示在预测类 i 和真实类 j 相交处的样本数量
- 多分类转化为多个二分类问题

4. 回归 (Regression)

从自变量 x 中预测因变量 y : $y = f(x)$ 或 $y = E(y|x)$

线性回归 (Linear Regression)

对于 n 个样本的数据集 $\{(x_i, y_i)\}_{i=1}^n$, 每个 x_i 是 p 维的, 设 $\mathbf{y} = (y_1, \dots, y_n)^T$, $\mathbf{w} = (w_1, \dots, w_p)$ 是待求的参数, $\mathbf{X} = [\mathbf{1}_n, (x_1, \dots, x_n)^T] \in \mathbb{R}^{n \times (p+1)}$, $\epsilon = (\epsilon_1, \dots, \epsilon_n)^T \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_n)$ 为误差, 则有

$$\mathbf{y} = \mathbf{X}\mathbf{w} + \epsilon$$

(平凡)最小二乘 [(Ordinary) Least Square, OLS]

最小化残差平方和 (residual sum-of-squares):

$$RSS(\mathbf{w}) = \sum_{i=1}^n (y_i - w_0 - w_1 x_1 - \dots - w_p x_p)^2 = \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2$$

$$\nabla_{\mathbf{w}} RSS(\hat{\mathbf{w}}) = 0 \Rightarrow \hat{\mathbf{w}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

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

其中, \mathbf{P} 是一个投影矩阵, 即 $\mathbf{P}^2 = \mathbf{P}$

评价指标

参数 $R^2 = 1 - \frac{SS_{res}}{SS_{tot}} = \left(\frac{SS_{reg}}{SS_{tot}} \text{ for linear regression} \right)$, 越大说明回归效果越好

总平方和 (total sum of squares): $SS_{tot} = \sum_{i=1}^n (y_i - \bar{y})^2$

回归平方和 (regression sum of squares): $SS_{reg} = \sum_{i=1}^n (\hat{y}_i - \bar{y})^2$

残差平方和 (residual sum of squares): $SS_{res} = \sum_{i=1}^n (y_i - \hat{y}_i)^2$

多元共线性 (Multicollinearity)

若 \mathbf{X} 的列几乎线性相关 (即多元共线), 那么 $\det(\mathbf{X}^T \mathbf{X}) \approx 0$, 则 $(\mathbf{X}^T \mathbf{X})^{-1}$ 会非常大, 那么得到的 $\hat{\mathbf{w}}$ 将会不准确。

Bias-Variance Decomposition

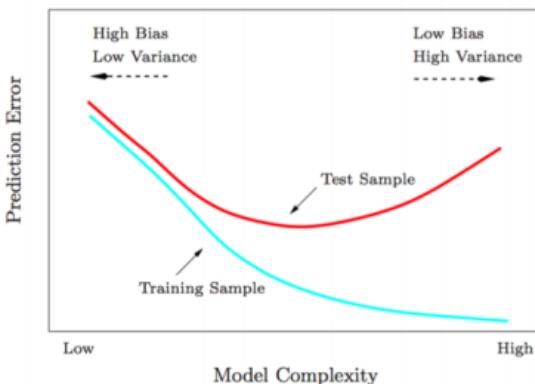
L^2 损失中泛化误差 (generalization error) 的偏置方差分解 (bias-variance decomposition):

$$E_{train} R_{exp}(\hat{f}(x)) = E_{train} E_P[(y - \hat{f}(x))^2 | x] = Var(\hat{f}(x)) + Bias^2(\hat{f}(x)) + \sigma^2$$

其中 $P = P(y|x)$

偏差 (Bias): $Bias^2(\hat{f}(x)) = E_{train} \hat{f}(x) - f(x)$, 是对该模型进行预测的平均精度 (accuracy)

方差 (Variance): $Var(\hat{f}(x)) = E_{train}(\hat{f}(x) - E_{train} \hat{f}(x))^2$, 是由于不同数据集而导致的模型预测的可变性 (稳定性)



对于 kNN 回归 (将 kNN 分类的众数改为平均数) 有:

- 对于较小的 k , 过拟合, 低偏差, 高方差
- 对于较大的 k , 不足拟合, 高偏差, 低方差

正则化 (Regularization)

在维度过高，列很多的时候，方差会很大。减小一些系数或将其设置为零可以减少过拟合。选择更少的变量同样可以达到更高的效果。

Best-subset selection

对于所有 $k \in \{0, 1, \dots, p\}$ ，大小为 k 的子集 $S_k \subset \{0, 1, \dots, p\}$ ，使得

$$RSS(\mathbf{w}) = \sum_{i=1}^n (y_i - w_0 - \sum_{j \in S_k} w_j x_{ij})^2$$

使用惩罚项 (Penalties) 正则化

$$\sum_{i=1}^n (y_i - w_0 - w_1 x_1 - \dots - w_p x_p)^2 + \lambda \|\mathbf{w}\|_q^q = \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_q^q$$

$q = 2$ 时，被称为岭回归； $q = 1$ 时，被称为 LASSO 回归

岭回归 (Ridge Regression)

$$\hat{\mathbf{w}} = \arg \min_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_2^2$$

$\lambda \geq 0$ 是可以通过交叉验证 (CV) 调参的参数

该问题等价于下面的优化问题：

$$\hat{\mathbf{w}} = \arg \min_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2, \quad \text{subject to } \|\mathbf{w}\|_2^2 \leq \mu$$

$\mu \geq 0$ 是一个参数，较大的 λ 对应较小的 μ

岭回归的解如下

$$\begin{aligned} \hat{\mathbf{w}}^{ridge} &= (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I}_{p+1})^{-1} \mathbf{X}^T \mathbf{y} \\ \hat{\mathbf{y}}^{ridge} &= \mathbf{X} \hat{\mathbf{w}}^{ridge} \end{aligned}$$

也可以使用 SVD 先将矩阵 \mathbf{X} 分解，得到 $\mathbf{X} = \mathbf{P} \mathbf{D} \mathbf{Q}^T$ ，其中 $\mathbf{P} \in \mathbb{R}^{n \times (p+1)}$ ， $\mathbf{Q} \in \mathbb{R}^{(p+1) \times (p+1)}$ ，都是正交矩阵（即 $\mathbf{P}^T \mathbf{P} = \mathbf{I}$ ， $\mathbf{Q}^T \mathbf{Q} = \mathbf{I}$ ） \mathbf{D} 是对角矩阵， $\mathbf{D} = diag(v_1, \dots, v_{p+1})$ ，那么得到岭回归的解为

$$\hat{\mathbf{y}}^{ridge} = \mathbf{P} diag\left(\frac{v_1}{v_1^2 + \lambda}, \dots, \frac{v_{p+1}}{v_{p+1}^2 + \lambda}\right) \mathbf{P}^T \mathbf{y}$$

(可以对比一下 OLS (ordinary least square) 得到的解 $\hat{\mathbf{y}}^{OLS} = \mathbf{P} \mathbf{P}^T \mathbf{y}$)

从贝叶斯视角看岭回归

Bayesian Viewpoint of Ridge Regression

- Given \mathbf{X} and \mathbf{w} , the conditional distribution of \mathbf{y} is
$$P(\mathbf{y}|\mathbf{X}, \mathbf{w}) = \mathcal{N}(\mathbf{X}\mathbf{w}, \sigma^2 \mathbf{I}) \propto \exp\left(-\frac{1}{2\sigma^2}(\mathbf{y} - \mathbf{X}\mathbf{w})^T(\mathbf{y} - \mathbf{X}\mathbf{w})\right)$$
- In addition, assume \mathbf{w} has a prior distribution
$$P(\mathbf{w}) = \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Lambda}_0) \propto \exp\left(-\frac{1}{2}(\mathbf{w} - \boldsymbol{\mu}_0)^T \boldsymbol{\Lambda}_0^{-1}(\mathbf{w} - \boldsymbol{\mu}_0)\right)$$
- By Bayes theorem, the posterior distribution of \mathbf{w} given the data \mathbf{X} and \mathbf{y} is

$$\begin{aligned} P(\mathbf{w}|\mathbf{X}, \mathbf{y}) &\propto P(\mathbf{y}|\mathbf{X}, \mathbf{w})P(\mathbf{w}) \\ &\propto \exp\left(-\frac{1}{2\sigma^2}(\mathbf{w}^T \mathbf{X}^T \mathbf{X}\mathbf{w} - 2\mathbf{y}^T \mathbf{X}\mathbf{w})\right) \\ &\quad - \frac{1}{2}(\mathbf{w}^T \boldsymbol{\Lambda}_0^{-1} \mathbf{w} - 2\boldsymbol{\mu}_0^T \boldsymbol{\Lambda}_0^{-1} \mathbf{w}) \\ &\propto \exp\left(-\frac{1}{2}(\mathbf{w} - \boldsymbol{\mu}_m)^T \boldsymbol{\Lambda}_m^{-1}(\mathbf{w} - \boldsymbol{\mu}_m)\right) \end{aligned}$$

where $\boldsymbol{\Lambda}_m = (\frac{1}{\sigma^2} \mathbf{X}^T \mathbf{X} + \boldsymbol{\Lambda}_0^{-1})^{-1}$ and $\boldsymbol{\mu}_m = \boldsymbol{\Lambda}_m (\frac{1}{\sigma^2} \mathbf{X}^T \mathbf{y} + \boldsymbol{\Lambda}_0^{-1} \boldsymbol{\mu}_0)$

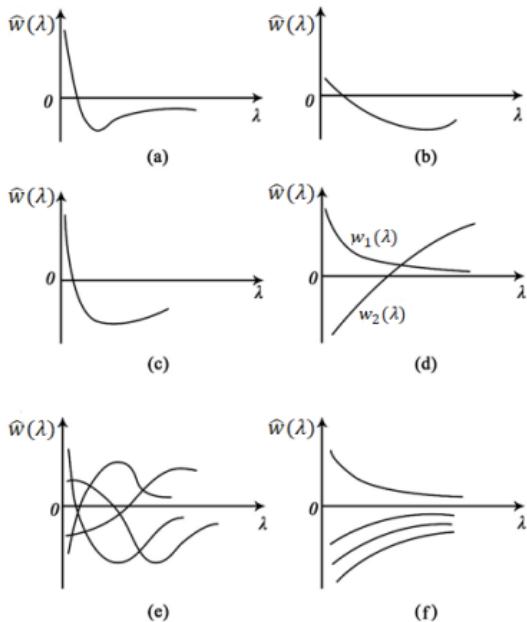
- If $\boldsymbol{\mu}_0 = 0$ and $\boldsymbol{\Lambda}_0 = \frac{\sigma^2}{\lambda} \mathbf{I}_{p+1}$, then $\hat{\mathbf{w}} = \boldsymbol{\mu}_m = (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I}_{p+1})^{-1} \mathbf{X}^T \mathbf{y}$ maximizes the posterior probability $P(\mathbf{w}|\mathbf{X}, \mathbf{y})$

岭轨迹 (Ridge Trace)

以 λ 为自变量, $\hat{\mathbf{w}}^{ridge}(\lambda)$ 为因变量的函数图像被称为岭轨迹

当 $\lambda \in (0, 0.5)$ 时, 岭轨迹很不稳定, 一般取 $\lambda = 1$

下图是一些岭轨迹图像:



- 轨迹稳定、绝对值较小的系数对 y 的影响不大, 如 (a)
- 轨迹稳定、绝对值较大的系数对 y 的影响较大, 如 (b)
- 两个变量的岭轨迹不稳定, 但和是稳定的, 说明这两个变量有多重共线性, 如 (d)
- 所有变量的岭轨迹稳定, 说明使用 OLS 具有良好的性能, 如 (f)

LASSO 回归

$$\hat{\mathbf{w}} = \arg \min_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_1$$

该问题等价于下面的优化问题：

$$\hat{\mathbf{w}} = \arg \min_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2, \quad \text{subject to } \|\mathbf{w}\|_1 \leq \mu$$

$\mu \geq 0$ 是一个参数，较大的 λ 对应较小的 μ

LASSO 回归的解为：

$$\hat{w}_i^{lasso} = (|\hat{w}_i^{OLS}| - \lambda)_+ sign(\hat{w}_i^{OLS})$$

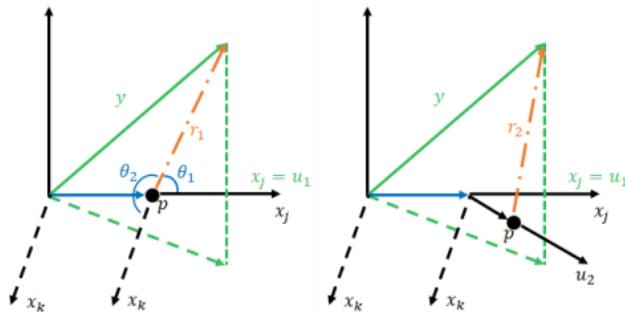
\hat{w}_i^{lasso} 被称为 \hat{w}_i^{OLS} 的软阈值 (soft thresholding)，其中记号 $(a)_+ = \max(a, 0)$ 代表 a 的正部

LASSO 轨迹与岭轨迹定义相同，这些路径是分段线性的，可能多次穿过 x 轴。在实际应用中，通常会通过交叉验证 (CV) 的方式选择参数 λ 。

下面是一种解 LASSO 回归的方法：

Solving LASSO by LARS (Hastie and Efron)

1. Start with all coefficients w_i equal to zero
2. Find the predictor x_i most correlated with y
3. Increase the coefficient w_i in the direction of the sign of its correlation with y . Take residuals $r = y - \hat{y}$ along the way. Stop when some other predictor x_k has as much correlation with r as x_i has
4. Increase (w_i, w_k) in their joint least squares direction, until some other predictor x_m has as much correlation with the residual r
5. Continue until all predictors are in the model



与 LASSO 相关的一些模型：

- Elastic net: $\hat{\mathbf{w}} = \arg \min_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda_1 \|\mathbf{w}\|_2^2 + \lambda_2 \|\mathbf{w}\|_1$
- Group LASSO: $\hat{\mathbf{w}} = \arg \min_{\mathbf{w}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \sum_{g=1}^G \lambda_g \|\mathbf{w}_g\|_2$, 其中 $\mathbf{w} = (\mathbf{w}_1, \dots, \mathbf{w}_g)$ 是 \mathbf{w} 的一个分组

最大后验估计 [Max A Posterior (MAP) Estimation]

给定参数 θ , y 的条件分布为 $P(y|\theta)$, 且参数 θ 有先验分布 $P(\theta)$

那么 θ 对于给定 y 的后验分布 $P(\theta|y) \propto P(y|\theta)P(\theta)$

MAP 选择后验最大的参数估计:

$$\hat{\theta}^{MAP} = \arg \max_{\theta} P(\theta|y) = \arg \max_{\theta} (\log P(y|\theta) + \log P(\theta))$$

不同的对数先验导致不同的惩罚 (正则化), 但一般情况不是这样: 有些惩罚可能不是概率分布的对数, 其他一些惩罚取决于数据 (先验独立于数据)

回归模型的评价

- Mean absolute error (MAE): $MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$
- Mean square error (MSE): $MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$
- Root mean square error (RMSE): $RMSE = \sqrt{MSE}$
- 相关系数 (Coefficient of Determination) $R^2 = 1 - \frac{SS_{res}}{SS_{tot}} = \left(\frac{SS_{reg}}{SS_{tot}} \text{ for linear regression} \right)$, 越大说明回归效果越好

总平方和 (total sum of squares): $SS_{tot} = \sum_{i=1}^n (y_i - \bar{y})^2$

回归平方和 (regression sum of squares): $SS_{reg} = \sum_{i=1}^n (\hat{y}_i - \bar{y})^2$

残差平方和 (residual sum of squares): $SS_{res} = \sum_{i=1}^n (y_i - \hat{y}_i)^2$

- 校正系数的决定系数 (Adjusted Coefficient of Determination): $R_{adj}^2 = 1 - \frac{(1-R^2)(n-1)}{n-p-1}$, 其中 n 代表样本数量, p 代表样本的维数, 越大说明回归效果越好

当加入重要样本时, R_{adj}^2 变大而 SS_{res} 变小; 当加入不重要样本时, R_{adj}^2 可能变小而 SS_{res} 可能变大

实际上, $1 - R_{adj}^2 = \frac{\hat{\sigma}^2}{S^2}$, 其中 $\hat{\sigma}^2 = \frac{1}{n-p-1} \sum_{i=1}^n (y_i - \hat{y}_i)^2$, $S^2 = \frac{1}{n-1} \sum_{i=1}^n (y_i - \bar{y})^2$ (若 $\mathbf{w} = 0$, 则有 $(n-p-1) \frac{\hat{\sigma}^2}{\sigma^2} \sim \chi^2_{n-p-1}$ 和 $(n-1) \frac{S^2}{\sigma^2} \sim \chi^2_{n-1}$)

5. 分类2

逻辑回归 (Logistic Regression)

对于二分类问题, 逻辑回归是

$$P(y=1|\mathbf{X}=\mathbf{x}) = \frac{\exp(\mathbf{w}^T \mathbf{x})}{1 + \exp(\mathbf{w}^T \mathbf{x})}$$
$$P(y=0|\mathbf{X}=\mathbf{x}) = \frac{1}{1 + \exp(\mathbf{w}^T \mathbf{x})}$$

假设 $P(y=k|\mathbf{X}=\mathbf{x}) = p_k(\mathbf{x}; \mathbf{w}) \quad k = 0, 1$, 似然函数为 $L(\mathbf{w}) = \prod_{i=1}^n p_{y_i}(\mathbf{x}_i; \mathbf{w})$

MLE 估计 \mathbf{w} : $\hat{\mathbf{w}} = \arg \max_{\mathbf{w}} L(\mathbf{w})$

最后使用 Newton-Raphson 方法解 $\nabla_{\mathbf{w}} \log L(\mathbf{w}) = 0$

线性判别分析 [Linear Discriminant Analysis (LDA)]

假设 $\pi_k = P(Y = k)$ 是先验概率, $f_k(\mathbf{x}) = P(\mathbf{X} = \mathbf{x}|Y = k)$ 是 $Y = k$ 这一类的密度函数

由贝叶斯定理: $P(Y|\mathbf{X} = \mathbf{x}) \propto f_k(\mathbf{x})\pi_k$

假设 $f_k(\mathbf{x})$ 是多元高斯分布, 那么

$$f_k(\mathbf{x}) = \frac{1}{(2\pi)^{p/2} |\Sigma_k|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\mu_k)^T \Sigma_k^{-1} (\mathbf{x}-\mu_k)}$$

协方差矩阵是统一的, $\Sigma_k = \Sigma$, 从对数角度来看, 也就是

$$\log \frac{P(Y = k|\mathbf{X} = \mathbf{x})}{P(Y = l|\mathbf{X} = \mathbf{x})} = \log \frac{\pi_k}{\pi_l} - \frac{1}{2}(\mu_k + \mu_l)^T \Sigma^{-1}(\mu_k - \mu_l) + \mathbf{x}^T \Sigma^{-1}(\mu_k - \mu_l)$$

记 $\delta_k(\mathbf{x}) = \log \pi_k - \frac{1}{2}\mu_k^T \Sigma^{-1} \mu_k + \mathbf{x}^T \Sigma^{-1} \mu_k$, 那么 $\log \frac{P(Y=k|\mathbf{X}=\mathbf{x})}{P(Y=l|\mathbf{X}=\mathbf{x})} = \delta_k(\mathbf{x}) - \delta_l(\mathbf{x})$

分类依据: $k^* = \arg \max_k \delta_k(\mathbf{x})$

对未知信息的样本的参数估计:

- $\hat{\pi}_k = N_k/N$, 其中 $N = \sum_{k=1}^K N_k$
- $\hat{\mu}_k = \frac{1}{N_k} \sum_{y_i=k} \mathbf{x}_i$
- $\hat{\Sigma} = \frac{1}{N-K} \sum_{k=1}^K \sum_{y_i=k} (\mathbf{x}_i - \hat{\mu}_k)(\mathbf{x}_i - \hat{\mu}_k)^T$

对于二分类问题, LDA 将样本分类为类别 2 如果满足下列不等式:

$$(\mathbf{x} - \frac{\hat{\mu}_1 + \hat{\mu}_2}{2})^T \Sigma^{-1}(\hat{\mu}_1 - \hat{\mu}_2) + \log \frac{\hat{\pi}_2}{\hat{\pi}_1} > 0$$

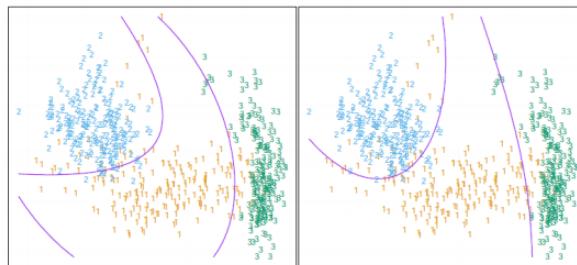
判别的方向为 $\beta = \Sigma^{-1}(\hat{\mu}_1 - \hat{\mu}_2)$, 贝叶斯错误分类比例为 $1 - \Phi(\beta^T(\mu_2 - \mu_1)/(\beta^T \Sigma \beta)^{\frac{1}{2}})$, 其中 $\Phi(x)$ 是正态分布的累计函数

下面是一个其他的判别分析:

Other Variants

- Quadratic discriminant analysis (QDA) :

$$\delta_k(\mathbf{x}) = -\frac{1}{2} \log |\Sigma_k| - \frac{1}{2} (\mathbf{x} - \mu_k)^T \Sigma_k^{-1} (\mathbf{x} - \mu_k) + \log \pi_k$$
- Regularized discriminant analysis : $\hat{\Sigma}_k(\alpha) = \alpha \hat{\Sigma}_k + (1 - \alpha) \hat{\Sigma}$
- Computations for LDA :
 1. Sphere the data with respect to $\hat{\Sigma} = \mathbf{U} \mathbf{D} \mathbf{U}^T$: $\mathbf{X}^* = \mathbf{D}^{-\frac{1}{2}} \mathbf{U}^T \mathbf{X}$.
Then the common covariance estimate of \mathbf{X}^* is \mathbf{I}_p
 2. Classify to the closest class centroid in the transformed space, taking into account of the class prior probabilities π_k 's
- Reduced-Rank LDA : see dimensionality reduction



神经网络 (Neural Network)

深度学习 (Deep Learning)

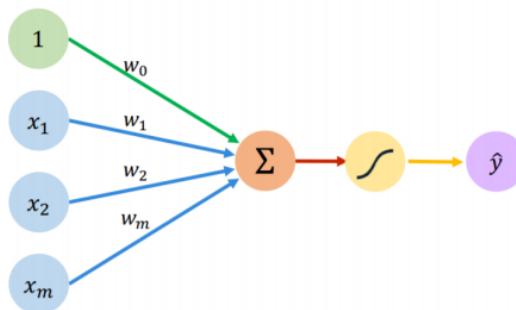
深度学习是机器学习的一个子领域，试图模仿人类大脑使用神经元的工作

深度学习专注于使用几个隐藏层来构建人工神经网络 (Artificial Neural Networks, ANN)

有各种各样的深度学习网络，如多层感知器 (Multilayer Perceptron, MLP)、自动编码器 (Autoencoders, AE)、卷积神经网络 (Convolution Neural Network, CNN)、递归神经网络 (Recurrent Neural Network, RNN)

近年来，软件硬件及大数据飞速发展，为深度学习的发展提供了良好的条件。还有研究表明，数据量更大时，深度学习的效果比传统机器学习算法更优

感知器：正向传播 (The Perceptron : Forward Propagation)

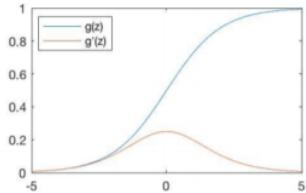


Inputs Weights Sum Non-Linearity Output

$$\hat{y} = g(w_0 + \mathbf{X}^T \mathbf{W}), \quad \text{where} \quad \mathbf{X} = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix}, \mathbf{W} = \begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix}$$

常见的激活函数 (Activation functions)

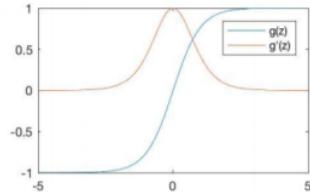
Sigmoid Function



$$g(z) = \frac{1}{1 + e^{-z}}$$

$$g'(z) = g(z)(1 - g(z))$$

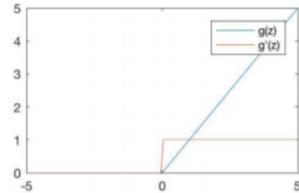
Hyperbolic Tangent



$$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

$$g'(z) = 1 - g(z)^2$$

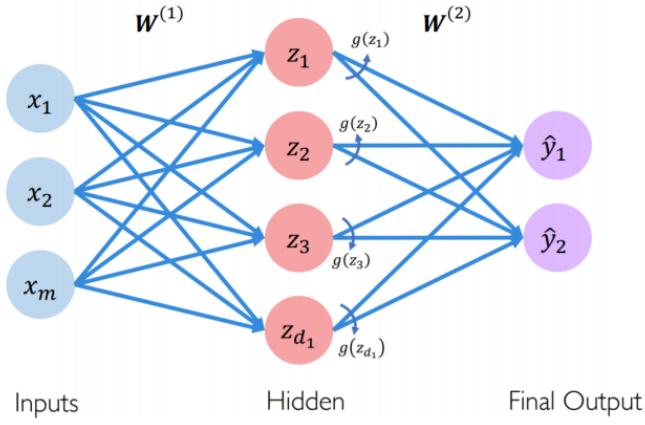
Rectified Linear Unit (ReLU)



$$g(z) = \max(0, z)$$

$$g'(z) = \begin{cases} 1, & z > 0 \\ 0, & \text{otherwise} \end{cases}$$

单层神经网络举例：



$$z_i = w_{0,i}^{(1)} + \sum_{j=1}^m x_j w_{j,i}^{(1)}, \quad \hat{y}_i = g(w_{0,i}^{(2)} + \sum_{j=1}^{d_1} z_j w_{j,i}^{(2)})$$

神经网络算法

记 $C(\mathbf{W}) = \frac{1}{n} \sum_{i=1}^n \mathcal{L}(f(x^{(i)}; \mathbf{W}), y^{(i)})$, 需要求损失函数最小的参数 $\mathbf{W}^* = \arg \min_{\mathbf{W}} C(\mathbf{W})$, 其中 $\mathbf{W} = \{\mathbf{W}^{(0)}, \mathbf{W}^{(1)}, \dots\}$

使用梯度下降法 (Gradient Decent) 优化参数 \mathbf{W} , 计算 $\frac{\partial C}{\partial \mathbf{W}}$

符号说明：

- w_{jk}^l 是从第 $(l-1)$ 层第 k 个神经元到第 l 层第 j 个神经元的权重
- $b_j^l = w_{j0}^l$ 是第 l 层第 j 个神经元的偏差 (bias)
- a_j^l 表示第 l 层第 j 个神经元 z_j^l 的激活, 即 $a_j^l = g(z_j^l) = g\left(\sum_k w_{jk}^l a_k^{l-1} + b_j^l\right)$

4 个基本结论及推导：

Four fundamental equations

We first define the error δ_j^l of neuron j in layer by

$$\delta_j^l \equiv \frac{\partial C}{\partial z_j^l},$$

and we give the four fundamental equations of back propagation :

$$\delta^L = \nabla_a C \odot \sigma'(z^L) \quad (BP1)$$

$$\delta^l = \left((w^{l+1})^T \delta^{l+1} \right) \odot \sigma'(z^l) \quad (BP2)$$

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l \quad (BP3)$$

$$\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l \quad (BP4)$$

An equation for the error in the output layer (BP1)

An equation for the error in the hidden layer (BP2)

The components of δ^l are given by

$$\delta^L = \nabla_a C \odot \sigma'(z^L) \quad (BP1)$$

$$\delta^l = \left((w^{l+1})^T \delta^{l+1} \right) \odot \sigma'(z^l) \quad (BP2)$$

Démonstration.

$$\begin{aligned} \delta_j^l &= \frac{\partial C}{\partial z_j^l} \\ &= \sum_k \frac{\partial C}{\partial a_k^l} \frac{\partial a_k^l}{\partial z_j^l} \\ &= \frac{\partial C}{\partial a_j^l} \frac{\partial a_j^l}{\partial z_j^l} = \frac{\partial C}{\partial a_j^l} \frac{\partial \sigma(z_j^l)}{\partial z_j^l} \\ &= \frac{\partial C}{\partial a_j^l} \sigma'(z_j^l) \end{aligned}$$

□

$$\begin{cases} \delta_j^l = \frac{\partial C}{\partial z_j^l} = \sum_k \frac{\partial C}{\partial z_i^{l+1}} \frac{\partial z_i^{l+1}}{\partial z_j^l} = \sum_k \delta_k^{l+1} \frac{\partial z_k^{l+1}}{\partial z_j^l} \\ z_k^{l+1} = \left(\sum_i w_{ki}^{l+1} a_i^l \right) + b_k^{l+1} = \left(\sum_i w_{ki}^{l+1} \sigma(z_i^l) \right) + b_k^{l+1} \end{cases}$$

$$\Rightarrow \begin{cases} \delta_j^l = \sum_k \delta_k^{l+1} \frac{\partial z_k^{l+1}}{\partial z_j^l} \\ \frac{\partial z_k^{l+1}}{\partial z_j^l} = w_{kj}^{l+1} \sigma'(z_j^l) \end{cases} \Rightarrow \delta_j^l = \sum_k \delta_k^{l+1} w_{kj}^{l+1} \sigma'(z_j^l) \quad \square$$

The change of the cost with respect to any bias (BP3)

The change of the cost with respect to any weight (BP4)

$$\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l \quad (BP4)$$

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l \quad (BP3)$$

Démonstration.

$$\begin{cases} \frac{\partial C}{\partial b_j^l} = \sum_k \frac{\partial C}{\partial z_j^l} \frac{\partial z_j^l}{\partial b_j^l} = \frac{\partial C}{\partial z_j^l} \frac{\partial z_j^l}{\partial b_j^l} \\ z_j^l = \left(\sum_k w_{jk}^l a_k^{l-1} \right) + b_j^l \Rightarrow \frac{\partial z_j^l}{\partial b_j^l} = 1 \Rightarrow \frac{\partial C}{\partial b_j^l} = \frac{\partial C}{\partial z_j^l} \cdot 1 = \delta_j^l. \end{cases} \quad \square$$

$$\begin{cases} \frac{\partial C}{\partial w_{jk}^l} = \sum_i \frac{\partial C}{\partial z_i^l} \frac{\partial z_i^l}{\partial w_{jk}^l} = \frac{\partial C}{\partial z_j^l} \frac{\partial z_j^l}{\partial w_{jk}^l} \\ z_j^l = \left(\sum_m w_{jm}^l a_m^{l-1} \right) + b_j^l \Rightarrow \frac{\partial z_j^l}{\partial w_{jk}^l} = a_k^{l-1} \\ \Rightarrow \frac{\partial C}{\partial w_{jk}^l} = \frac{\partial C}{\partial z_j^l} \frac{\partial z_j^l}{\partial w_{jk}^l} = \delta_j^l a_k^{l-1} \end{cases}$$

□

反向传播过程 (Back Propagation Procedure)

1. 输入 x : 为输入层设置相应的激活值 a^1
2. 向前反馈: 对于每个 $l = 2, 3, \dots, L$, 计算 $z^l = w^l a^{l-1} + b^l$ 和 $a^l = \sigma(z^l)$
3. 输出误差 δ^L : 计算 $\delta^L = \nabla_a C \odot \sigma'(z^L)$
4. 反向传播误差: 对于每个 $l = L-1, L-2, \dots, 2$, 计算 $\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$
5. 输出: 函数 C 的输出梯度为 $\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l$ 和 $\frac{\partial C}{\partial b_j^l} = \delta_j^l$

梯度下降算法 (Gradient Descent)

1. 随机初始化权值 $\sim \mathcal{N}(0, \sigma^2)$
2. 循环直到结果收敛
3. 计算梯度 $\frac{\partial J(\mathbf{W})}{\partial \mathbf{W}}$

4. 更新权值 $\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{\partial J(\mathbf{W})}{\partial \mathbf{W}}$

5. 返回权值结果

支持向量机 (Support Vector Machine, SVM)

使用超平面划分数据以达到分类效果，最大化边距 (Margin)

线性 SVM

训练数据: $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$, $y_i \in \{-1, +1\}$

超平面: $S = \mathbf{w}^T \mathbf{x} + b$

决策函数: $f(\mathbf{x}) = sign(\mathbf{w}^T \mathbf{x} + b)$

点与超平面间的距离: $r_i = \frac{y_i(\mathbf{w}^T \mathbf{x}_i + b)}{\|\mathbf{w}\|_2}$

数据集与超平面间的边距: $\min_i r_i$

最大化边距: $\max_{\mathbf{w}, b} \min_i \frac{y_i(\mathbf{w}^T \mathbf{x}_i + b)}{\|\mathbf{w}\|_2}$

化为一下约束规划问题:

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|_2^2, \quad s.t. \quad y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1, i = 1, \dots, n$$

这是一个具有线性约束的二次规划 (quadratical programming) 问题, 计算复杂度为 $O(p^3)$, 其中 p 为维数

使用拉格朗日乘子法 (Method of Lagrange Multipliers) 解决

假设 $\alpha_i \geq 0$ 是约束条件 $y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1$ 拉格朗日乘子因数, 那么拉格朗日函数如下:

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2} \|\mathbf{w}\|_2^2 - \sum_{i=1}^n \alpha_i [y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1]$$

那么,

$$\max_{\alpha} L(\mathbf{w}, b, \alpha) = \begin{cases} \frac{1}{2} \|\mathbf{w}\|_2^2, & y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 \geq 0 \\ +\infty, & y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 < 0 \end{cases}$$

那么问题转化为求 $\min_{\mathbf{w}, b} \max_{\alpha} L(\mathbf{w}, b, \alpha)$, 对偶问题为 $\max_{\alpha} \min_{\mathbf{w}, b} L(\mathbf{w}, b, \alpha)$

先解 $\min_{\mathbf{w}, b} L(\mathbf{w}, b, \alpha)$:

$$\nabla_{\mathbf{w}} L = 0 \Rightarrow \mathbf{w}^* = \sum_i \alpha_i y_i \mathbf{x}_i$$

$$\frac{\partial L}{\partial b} = 0 \Rightarrow \sum_i \alpha_i y_i = 0$$

代入 L 中: $L(\mathbf{w}^*, b^*, \alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j)$

问题转化为:

$$\max_{\alpha} \left[\frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j) - \sum_i \alpha_i \right] \quad s.t. \quad \alpha_i \geq 0, \sum_i \alpha_i y_i = 0, i = 1, \dots, n$$

KKT 条件

$$\begin{cases} \alpha_i^* \geq 0 \\ y_i((\mathbf{w}^*)^T \mathbf{x}_i + b^*) - 1 \geq 0 \\ \alpha_i^* [y_i((\mathbf{w}^*)^T \mathbf{x}_i + b^*) - 1] = 0 \end{cases}$$

支持向量的指标集 $S = \{i | \alpha_i > 0\}$

$$b = y_s - \mathbf{w}^T \mathbf{x}_s = y_s - \sum_{i \in S} \alpha_i y_i \mathbf{x}_i^T \mathbf{x}_s$$

$$\text{更稳定的解: } b = \frac{1}{|S|} \sum_{s \in S} \left(y_s - \sum_{i \in S} \alpha_i y_i \mathbf{x}_i^T \mathbf{x}_s \right)$$

SMO (Sequential Minimal Optimization) 算法

1. 选择离 KKT 条件最远的一对 α_i 和 α_j
2. 假设其他参数固定不变, 求解 α_i 和 α_j
3. 更新 α_i 和 α_j , 返回第 1 步
4. 直到参数收敛, 退出

计算复杂度为 $O(n^3)$

软边距

当数据集不线性可分, 那么可以引入一些松弛变量 $\xi_i \geq 0$, 将约束条件变为 $y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i$

那么优化问题为:

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|_2^2 + C \sum_{i=1}^n \xi_i, \quad s.t. \quad y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i, \xi_i \geq 0, i = 1, \dots, n$$

对偶问题为:

$$\max_{\alpha} \left[\frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j) - \sum_i \alpha_i \right] \quad s.t. \quad 0 \leq \alpha_i \leq C, \sum_i \alpha_i y_i = 0, i = 1, \dots, n$$

非线性 SVM

非线性 SVM 采用核函数 (Kernel function) 来替代 \mathbf{x}_i 和 \mathbf{x}_j 的内积, 优化问题的对偶问题为:

$$\max_{\alpha} \left[\frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) - \sum_i \alpha_i \right]$$

一般选用的核函数有以下几种

Kernel	Definition	Parameters
Polynomial	$(\mathbf{x}_1^T \mathbf{x}_2 + 1)^d$	d 是正整数
Gaussian	$e^{-\frac{\ \mathbf{x}_1 - \mathbf{x}_2\ ^2}{2\delta^2}}$	$\delta > 0$
Laplacian	$e^{-\frac{\ \mathbf{x}_1 - \mathbf{x}_2\ }{2\delta}}$	$\delta > 0$
Fisher	$\tanh(\beta \mathbf{x}_1^T \mathbf{x}_2 + \theta)$	$\beta > 0, \theta < 0$

SVM 的优缺点

优点

- 在图像识别方面表现很好
- 易于使用核函数处理维度高的数据集
- 鲁棒性好，并易于推广到新数据集

缺点

- 对超高维数据处理能力不好
- 当样本量较大时，非线性 SVM 的计算效率较低
- 没有概率的参与所以可解释性较差

6. 集成学习 (Ensemble Learning)

多个弱学习模型 (basic learner, 可能是异源的 (heterogenous)) 可以提高学习效果

减少误分类率：假设一个分类器的误分类率为 p , 选用 N 个独立同分布的分类器, 它们投票后的误分类率
 $\sum_{k>N/2}^N \binom{N}{k} p^k (1-p)^{N-k}$, 当 $N = 5$, $p = 0.1$ 时, 误分类率小于 1%

集成学习的分类

- Bagging (引导聚合 (bootstrap aggregation) 的缩写)
 - 随机抽样：生成独立的模型，并对回归进行平均（对分类或回归进行多数投票）
 - 算法：

输入：数据集 $D = \{(x_1, y_1), \dots, (x_n, y_n)\}$

输出：叠加模型 $\hat{f}_{bag} = (x)$

```
for m = 1 to M:  
    Sample from D with replacement to obtain D_m  
    Train a model f_m(x) from the dataset D_m : for classification, f_m(x)  
    returns a K-class 0-1 vector e_k ; for regression, it is just a value  
    Compute bagging estimate f_bag(x) = 1/M \sum_{m=1}^M f_m(x) : for  
    classification, make majority vote G_bag(x) = argmax_k f_k(x); for regression,  
    just return the average value
```

- 误差分析

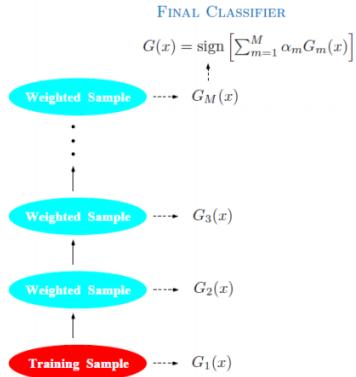
假设 $\{\hat{f}_m(x)\}_{m=1}^M$ 的方差均为 $\sigma^2(x)$, 每一对之间的相关性为 $\rho(x)$

$$Var(\hat{f}_{bag}(x)) = \frac{1}{M^2} \left(\sum_{m=1}^M Var(\hat{f}_m(x)) + \sum_{t \neq m} Cov(\hat{f}_t(x), \hat{f}_m(x)) \right) = \rho(x)\sigma^2(x) + \frac{1 - \rho(x)}{M}\sigma^2(x)$$

方差小于原来的 $\sigma^2(x)$

- Boosting
 - 顺序训练：根据以往模型的误差对后续模型进行训练
 - 减少偏差

- 比如 AdaBoost, GBDT



随机森林 (Random Forest)

决策树的缺点

- 受限于局部最优：贪婪算法使它停止在局部最优，因为它在每棵树的分裂中寻求最大的信息增益
 - 决策边界：在每个分割中仅使用一个特征，决策边界平行于坐标轴
 - 可描述性不好，稳定性不好

随机森林算法

Random Forest Algorithm

- Input : training set $D = \{(x_1, y_1), \dots, (x_N, y_N)\}$
 - Output : additive model $\hat{f}_{rf}(x)$
 1. For $m = 1$ to M :
 - 1.1 Sample from D with replacement to obtain D_m
 - 1.2 Grow a random-forest tree T_m to the dataset D_m : by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached
 - 1.2.1 Select q features at random from the p features
 - 1.2.2 Pick the best feature/split-point among the q
 - 1.2.3 Split the node into two daughter nodes
 2. Output the ensemble of trees $\{T_m\}_{m=1}^M$: for regression,

$$\hat{f}_{rf}(x) = \frac{1}{M} \sum_{m=1}^M T_m(x)$$
 : for classification, make majority vote
 - Small value of q increases the independency of trees ; empirically, $q = \log_2 p + 1$

随机森林的模型评估

Out-of-bag (OOB) errors 袋外样本误差:

对于每个观察结果 (x_i, y_i) , 找到将其作为 OOB 样本的树: $\{\hat{T}_m(\mathbf{x}) : (\mathbf{x}_i, y) \notin D_m\}$

使用这些树对这些观察结果进行分类，并将多数投票作为该观察结果的标签：

$$\hat{f}_{oob}(\mathbf{x}_i) = \arg \max_{y \in \mathcal{Y}} \sum_{m=1}^M I(\hat{f}_m(\mathbf{x}_i) = y) I(\mathbf{x}_i \notin D_m)$$

计算错误分类样本的数量，并以该数量与样本总数的比值作为OOB误差： $Err_{oob} = \frac{1}{N} \sum_{i=1}^N I(\hat{f}_{oob}(\mathbf{x}_i) \neq y_i)$

随机森林的 Feature Importance

[链接](#)

随机森林的优缺点

优点

- Bagging 或随机森林 (RF) 适用于具有高方差但低偏差的模型
- 对于非线性估计更好
- RF 适用于非常高维的数据，并且不需要做特征选择，因为 RF 赋予了 Feature Importance
- 易于进行并行计算

缺点

- 当样本规模大、噪声大或数据维度较低时容易过拟合
- 与单棵树相比，计算较慢
- 难以解释

AdaBoost

使用 Boost 拟合叠加模型

对于叠加模型 (Additive Model): $f(x) = \sum_{m=1}^M \beta_m b(x; \gamma_m)$

参数选择: $\min_{\{\beta_m, \gamma_m\}} \sum_{i=1}^N L(y_i, f(x))$

损失函数: 平方误差或 likelihood-based loss

算法:

- Input : training set $D = \{(x_1, y_1), \dots, (x_N, y_N)\}$
 - Output : additive model $f_M(x)$
1. Initialize $f_0(x) = 0$
 2. For $m = 1$ to M :
 - 2.1 Compute $(\beta_m, \gamma_m) = \arg \min_{\beta, \gamma} \sum_{i=1}^N L(y_i, f_{m-1}(x_i) + \beta b(x_i; \gamma))$
 - 2.2 Update $f_m(x) = f_{m-1}(x) + \beta_m b(x; \gamma_m)$
 - Squared error loss : in step 2.1,
 $L(y_i, f_{m-1}(x_i) + \beta b(x_i; \gamma)) = (\underbrace{y_i - f_{m-1}(x_i)}_{\text{residual}} - \beta b(x_i; \gamma))^2$

AdaBoost 算法

指数损失: $L(y, f(x)) = \exp(-yf(x))$

取 $b(x; \gamma_m)$ 为 $G(x)$

假设 $w_i^{(m)} = \exp(-y_i f_{m-1}(x_i))$, 上述算法的步骤 2.1 为

$$(\beta_m, G_m) = \arg \min_{\beta, G} \sum_{m=1}^M w_i^{(m)} \exp(-\beta_m y_i G(x_i)) = \arg \min_{\beta, G} \left[\sum_{y_i \neq G(x_i)} w_i^{(m)} (e^\beta - e^{-\beta}) + e^{-\beta} \sum_{m=1}^M w_i^{(m)} \right]$$

$$G_m = \arg \min_G \sum_{i=1}^n w_i^{(m)} I(y_i \neq G(x_i))$$

$$\beta_m = \arg \min_\beta [\epsilon_m (e^\beta - e^{-\beta}) + e^{-\beta}] = \frac{1}{2} \log \frac{1-\epsilon_m}{\epsilon_m}, \text{ 其中 } \epsilon_m = \frac{\sum_{i=1}^n w_i^{(m)} I(y_i \neq G(x_i))}{\sum_{i=1}^n w_i^{(m)}} \text{ 为 weighted error}$$

rate

AdaBoost Algorithm

- Input : training set $D = \{(x_1, y_1), \dots, (x_N, y_N)\}$, loss function $L(y, f(x))$
- Output : Weighted classifier $G(x)$

1. Initialize $w_i = 1/N, i = 1, \dots, N$
2. For $m = 1$ to M :
 - 2.1 Fit a classifier $G_m(x)$ to the training data D with weight $\{w_i\}$
 - 2.2 Compute the error $\epsilon_m = (\sum_{i=1}^n w_i^{(m)} I(y_i \neq G(x_i))) / \sum_{i=1}^n w_i^{(m)}$
 - 2.3 Compute $\alpha_m = \log \frac{1-\epsilon_m}{\epsilon_m}$ ($\alpha_m = 2\beta_m > 1$)
 - 2.4 Update the weight $w_i^{(m+1)} = w_i^{(m)} \exp(\alpha_m I(y_i \neq G_m(x_i))),$ for $i = 1, \dots, N$
3. Output $G(x) = \text{sign} \left[\sum_{m=1}^M \alpha_m G_m(x) \right]$

弱分类器的权重：分类器越好，其权重就越大

样本权重：每一步后重新加权，增加错误分类样本的权重

损失函数的选择：对于分类，exponential loss 也就是 $\exp(-yf(x))$ 和 binomial negative log-likelihood (deviance) loss $\log(1 + \exp(-2yf))$ 一样好；对于回归，squared error loss 不好，exponential loss 好，binomial deviance 更好

AdaBoost 优缺点

优点

- 与弱分类器相比，AdaBoost 提高了分类性能
- 弱分类器有许多选择：树、支持向量机、kNN 等
- 只有一个需要调的参数 M : 弱分类器的数量
- 防止单一弱分类器（如复杂决策树）造成的过拟合

缺点

- 可解释性弱

- 在使用很差的弱分类器时容易过拟合
- 对异常值敏感
- 不易并行计算

Gradient Boosting Decision Tree (GBDT)

Boosting Tree

使用分类树或回归树作为基础学习单元

$$f_M(x) = \sum_{m=1}^M T(x; \Theta_m), \text{ 其中 } T(x; \Theta) = \sum_{j=1}^J \gamma_j I(x \in R_j), \text{ 参数集 } \Theta = \{R_j, \gamma_j\}_{j=1}^J$$

对参数进行估计是一个组合优化问题: $\hat{\Theta} = \arg \min_{\Theta} \sum_{j=1}^J \sum_{x_i \in R_j} L(y_i, \gamma_j)$

如果使用前向分步算法 (Forward Stagewise Algorithm),

$$\hat{\Theta}_m = \arg \min_{\Theta_m} \sum_{i=1}^N L(y_i, f_{m-1}(x_i) + T(x_i; \Theta_m))$$

- $\hat{\gamma}_{jm} = \arg \min_{\gamma_{jm}} \sum_{x_i \in R_{jm}} L(y_i, f_{m-1}(x_i) + \gamma_{jm})$

- 但是计算 R_{jm} 比在单颗树计算困难很多

GBDT

监督学习等同于优化问题: $\min_f L(f) = \min_f \sum_{i=1}^N L(y_i, f(x_i))$

采用数值优化方法: $\hat{\mathbf{f}} = \arg \min_{\mathbf{f}} L(\mathbf{f}), \text{ 其中 } \mathbf{f} = \{f(x_1), \dots, f(x_N)\}$

$$\mathbf{f}_M = \sum_{m=1}^M \mathbf{h}_m, \quad \mathbf{f}_0 = \mathbf{h}_0 \text{ 是随机初始值}$$

$$\mathbf{f}_m = \mathbf{f}_{m-1} - \rho_m \mathbf{g}_m, \quad \mathbf{h}_m = -\rho_m \mathbf{g}_m, \quad g_{im} = \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)} \right]_{f(x_i)=f_{m-1}(x_i)}$$

$$\text{GBDT 要找一个树 } T(x; \Theta_m) \text{ 满足 } \tilde{\Theta}_m = \arg \min_{\Theta_m} \sum_{i=1}^N (-g_{im} - T(x; \Theta_m))^2$$

GBDT Algorithm

- Input : training set $D = \{(x_1, y_1), \dots, (x_N, y_N)\}$, loss function $L(y, f(x))$
- Output : boosting tree $\hat{f}(x)$

1. Initialize $f_0(x) = \arg \min_{\gamma} \sum_{i=1}^N L(y_i, \gamma)$
2. For $m = 1$ to M :
 - 2.1 For $i = 1, 2, \dots, N$ compute $r_{im} = \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)} \right]_{f=f_{m-1}}$
 - 2.2 Fit a regression tree to the target (residual) r_{im} , giving terminal regions $R_{jm}, j = 1, \dots, J_m$
 - 2.3 For $j = 1, \dots, J_m$, compute
$$\gamma_{jm} = \arg \min_{\gamma} \sum_{x_i \in R_{jm}} L(y_i, f_{m-1}(x_i) + \gamma)$$
 - 2.4 Update $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x_i \in R_{jm})$
3. $\hat{f}(x) = f_M(x)$

正则化的方法：

- 收缩 Shrinkage: 步骤 2.4 变为 $f_m(x) = f_{m-1}(x) + \nu \sum_{j=1}^{J_m} \gamma_{jm} I(x_i \in R_{jm})$
- 二次抽样 Subsampling: 在每次迭代中，采样训练集的占比为 η 的一部分，并使用子样本生成下一个树

Feature importance

- Feature importance
 - When fitting a single tree T , at each node t , one feature $X_{v(t)}$ and one separate value $X_{v(t)} = c_{v(t)}$ are chosen to improve a certain quantity of criterion (e.g. GINI, entropy, squared error, etc.)
 - Sum all these improvements i_t brought by each feature X_k over all internal nodes : $I_k(T) = \sum_{t=1}^{J-1} i_t I(v(t) = k)$
 - Average the improvements of all trees \Rightarrow importance of that feature : $I_k = \frac{1}{M} \sum_{m=1}^M I_k(T_m)$
- Partial Dependence Plots
 - Partial dependence of $f(X)$ on X_S : $f_S(X_S) = \mathbb{E}_{X_C} f(X_S, X_C)$
 - Estimate by empirical mean : $\bar{f}_S(X_S) = \frac{1}{N} \sum_{i=1}^N f(X_S, x_{iC})$

GBDT 优缺点

优点

- 适用于所有回归问题
- 更适用于两分类，可适用于多分类问题（不建议）
- 具有多种非线性性，强表征性

缺点

- 顺序流程，不易于并行计算

- 计算复杂度高，不适合用于具有稀疏特征的高维问题

XGBoost

Cost Function: $F(\Theta_m) = \sum_{i=1}^N L(y_i, f_{m-1}(x_i) + T(x_i; \Theta_m)) + R(\Theta_m)$

二阶泰勒展开得到 $F(\Theta_m) \approx \sum_{i=1}^N \left[L(y_i, f_{m-1}(x_i)) + g_i^{(m)} T(x_i; \Theta_m) + \frac{1}{2} h_{ii}^{(m)} T(x_i; \Theta_m)^2 \right] + R(\Theta_m)$
 , 其中 $g_i^{(m)} = \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)} \right]_{f(x_i)=f_{m-1}(x_i)}$ 是损失函数的梯度, $h_{ii}^{(m)} = \left[\frac{\partial^2 L(y_i, f(x_i))}{\partial f(x_i)^2} \right]_{f(x_i)=f_{m-1}(x_i)}$ 是损失函数的 Haisen 矩阵的对角线元素 (非对角线全为0)

惩罚项的选择, 以回归树为例

设 J_m 为叶子节点的数量 (分区中的矩形数量), γ_{jm} 为叶子节点 (区域) R_{jm} 中的近似常数 (权重 w)

树的复杂度为 $\{\gamma_{jm}\}$ 的 L^0 范数与 L^2 范数的和: $R(\Theta_m) = \frac{1}{2} \lambda \sum_{j=1}^{J_m} \gamma_{jm} + \mu J_m$

对 cost function 进行求解:

$$F(\Theta_m) \approx \sum_{i=1}^N L(y_i, f_{m-1}(x_i)) + \sum_{j=1}^{J_m} \left[\left(\sum_{x_i \in R_{jm}} g_i^{(m)} \right) \gamma_{jm} + \frac{1}{2} \left(\sum_{x_i \in R_{jm}} h_{ii}^{(m)} + \lambda \right) \gamma_{jm}^2 \right] + \mu J_m$$

$$F(\Theta_m) \approx \sum_{j=1}^{J_m} [G_j^{(m)} \gamma_{jm} + \frac{1}{2} (H_j^{(m)} + \lambda) \gamma_{jm}^2] + \mu J_m + \text{constant}, \text{ 其中 } G_j^{(m)} = \sum_{x_i \in R_{jm}} g_i^{(m)},$$

$$H_j^{(m)} = \sum_{x_i \in R_{jm}} h_{ii}^{(m)}$$

对 γ_{jm} 求导可得: $\hat{\gamma}_{jm} = -\frac{G_j^{(m)}}{H_j^{(m)} + \lambda}$

化简: $F(\Theta_m) = -\frac{1}{2} \sum_{j=1}^{J_m} \frac{(G_j^{(m)})^2}{H_j^{(m)} + \lambda} + \mu J_m + \text{constant}$

忽略常数项, 我们得到了结构分数 (Structure Score): $SS = -\frac{1}{2} \sum_{j=1}^{J_m} \frac{(G_j^{(m)})^2}{H_j^{(m)} + \lambda} + \mu J_m$, 它类似于信息增益:

最小化结构分数会得到最好的树

损失函数:

- Square loss $L(y, f) = (y - f)^2$: $g_i^{(m)} = 2(f_i - y_i) = 2 \times \text{residue}$, $h_{ii}^{(m)} = 2$
- Logistic loss $L(y, f) = y \ln(1 + e^{-f}) + (1 - y) \ln(1 + e^f)$:
 $g_i^{(m)} = -y_i \left(1 - \frac{1}{1+e^{-f_{m-1}(x_i)}} \right) + (1 - y_i) \frac{1}{1+e^{-f_{m-1}(x_i)}} = \text{Pred} - \text{Label},$
 $h_{ii}^{(m)} = \frac{e^{-f_{m-1}(x_i)}}{(1+e^{-f_{m-1}(x_i)})^2} = \text{Pred} \times (1 - \text{Pred})$

节点分割: 贪心算法

当将一个节点分割为左 (L) 和右 (R) 子节点时, 最大化 $Gain = \frac{1}{2} \left[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda} \right] - \gamma$

- Input : training set $D = \{(x_1, y_1), \dots, (x_N, y_N)\}$, loss function $L(y, f(x))$, the index set $I = \{i | x_i \in R_{jm}\}$ of current node R_{jm} , feature dimension d
- Output : best split
 1. Initialize $gain = 0$, $G = \sum_{i \in I} g_i$, $H = \sum_{i \in I} h_{ii}$
 2. For $k = 1$ to K :
 - 2.1 $G_L = 0$, $H_L = 0$
 - 2.2 For j in sorted(I , by x_{jk}), do
 - 2.2.1 $G_L = G_L + g_j$, $H_L = H_L + h_{jj}$, $G_R = G - G_L$, $H_R = H - H_L$
 - 2.2.2 $score = \max(score, \frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{G^2}{H + \lambda})$
 3. Output split with max score

集成学习总结

- 集成方法具有单个模型的可叠加能力，具有更好的性能
- 易于推广到新的数据
- 噪声强时，容易过拟合（对噪声敏感）
- 计算密集型

7. 聚类 (Clustering)

也称为数据分割，将对象集合分组为子集或集群，得到的结果为每个集群中的对象比不同集群中的对象更相似

聚类不同于分类的点在于，它是**无监督学习**，没有输出或标签

聚类的最终目标：

- 在集群内样本相似性更高
- 在集群之间样本相似性更低

Cost Function：与输出无关，但与相似性有关

2 种类型的输入：

- $n \times n$ 相似度（不相似度）矩阵
 - 只有样本对之间的距离
 - 可能会丢失一些数据信息
- 具有 d 个特征的原始数据 $X \in \mathbb{R}^{n \times d}$

聚类的过程：

- 数据预处理，特别是标准化预处理
- 得到相似矩阵
- 选择要运用聚类算法
- 确定集群的最佳数量

聚类算法：

- 分区聚类 (Partitional clustering)
 - K-means
 - K-Medoids
 - Spectral clustering
 - DBSCAN
- 分层聚类 (Hierarchical clustering)

K-Means

将 n 个样本分组为 k 个集群，使每个样本属于最近的集群

数据集表示为 $\{x_i\}_{i=1}^n$, 其中 $x_i \in \mathbb{R}^d$

第 k 个集群 C_k 的质心为 c_k , 其中 $k = 1, 2, \dots, K$

K-Means 的大致思路是, x_i 属于集群 k 如果 $d(x_i, c_k) < d(x_i, c_m)$, $m \neq k$, 其中 $d(x_i, x_j)$ 是不相似度函数, 使质心位置良好, 从而使每个样本到其质心之间的平均距离尽可能小

转化为优化问题

聚类 $C : \{1, 2, \dots, n\} \rightarrow \{1, 2, \dots, k\}$

$$T = \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n d(x_i, x_j) = \frac{1}{2} \sum_{k=1}^K \sum_{C(i)=k} \left(\sum_{C(j)=k} d_{ij} + \sum_{C(j) \neq k} d_{ij} \right) = W(C) + B(C)$$

Loss Function:

- 聚类内点损失 $W(C) = \sum_{k=1}^K \sum_{C(i)=k} \sum_{C(j)=k} d_{ij}$
- 聚类间点损失 $B(C) = \sum_{k=1}^K \sum_{C(i)=k} \sum_{C(j) \neq k} d_{ij}$

最小化 $W(C)$ 相当于最大化 $B(C)$

不相似度

相似矩阵: $n \times n$ 的相似矩阵

点对之间的不相似度:

- $d(x_i, x_j) = \sum_{k=1}^p d_k(x_{ik}, x_{jk})$, d_k 可以取平方距离, 绝对值距离等
 - 加权平均: $d(x_i, x_j) = \sum_{k=1}^p w_k d_k(x_{ik}, x_{jk})$, 其中 $\sum_{k=1}^p w_k = 1$
- 设置 $w_k \sim 1/d_k$, $\overline{d_k} = \frac{1}{n^2} \sum_{i=1}^n \sum_{j=1}^n d_k(x_{ik}, x_{jk}) = 2\hat{Var}(X_k)$ 将对所有列产生同等影响
- 基于相关性: $d(x_i, x_j) \propto 1 - \rho(x_i, x_j)$

K-Means (as Central Voronoi Tessellation)

最小化 $W(C)$ 通常是不可行的, 因为这是一个贪婪的算法, 只适用于小的数据集, 因此会将其转化为优化问题

取平方不相似度，即 $W(C) = \sum_{k=1}^K n_k \sum_{c(i)=k} \|x_i - \bar{x}_k\|^2$ ，其中 n_k 表示集群 k 的样本数量，

$$\bar{x}_k = \frac{1}{n_k} \sum_{C(j)=k} x_j = \arg \min_{m_k} \sum_{C(j)=k} \|x_j - m_k\|^2$$

那么，转化的优化问题为

$$\min_C W(C) \Leftrightarrow \min_{C, m_k} \sum_{k=1}^K n_k \sum_{C(i)=k} \|x_i - m_k\|^2$$

轮换着迭代：

- 给定 C ，解 m_k ， $\Rightarrow m_k^* = \bar{x}_k$
- 给定 m_k ，解 C ， $\Rightarrow C(i) = \arg \min_{1 \leq k \leq K} \|x_i - m_k\|^2$

K-Means 的迭代结束条件是所有集群的质心都不再变化

各集群初始质心的选择：

- 随机猜测，选取猜测到的最小的 $W(C)$
- 基于其他聚类方法猜测 K-Means 的初始值

K 的选取

- 最小化贝叶斯信息准则 [Minimizing Bayesian Information Criterion (BIC)]
 $BIC(\mathcal{M}|\mathbf{X}) = -2 \log P(\mathbf{X}|\hat{\Theta}, \mathcal{M}) + p \log n$, 其中 \mathcal{M} 是模型本身， $\hat{\Theta}$ 是模型参数的最小似然估计， $P(\mathbf{X}|\hat{\Theta})$ 是似然函数， p 是模型的参数个数，是控制对数似然和模型复杂度之间的权衡
- 最小描述长度 [Minimum Description Length (MDL)]
从较大的 K 开始，减小 K 直到 $-\log P(\mathbf{X}|\hat{\Theta}, \mathcal{M}) - \log P(\Theta|\mathcal{M})$ 达到最小值，类似于最大后验估计 (MAP)
- 基于高斯分布的假设
从 $K = 1$ 开始，增加 K ，直到每个簇中的点遵循高斯分布

K-Means 优缺点

优点

- 直观，易于实现
- 时间复杂度较低，为 $O(tnpK)$ ，其中 t 为迭代次数

缺点

- 需要调参 K
- 强烈依赖于各集群质心的初始值
- 容易陷入局部最优
- 假设数据是球形的（数据经过了归一化预处理），难处理非球形数据
- 对异常值敏感

K-Means 的变体

Bisecting K-means

用于对各集群初始中心的选择的猜测，核心思想是依次将最坏的集群划分为两个子集群

算法过程：

1. 初始化将所有数据放在一个集群中
2. 循环：
 - 2.1 选择使集群内点最大分散 $\sum_{C(i)=k} \sum_{C(j)=k} \|x_i - x_j\|^2$ 的集群 k
 - 2.2 使用 2-means 将集群 k 划分为两个子集群，随机初始猜测两个中心
 - 2.3 重复步骤 2.2 p 次，选择能最小化集群点分散的最佳集群对
3. 有 K 个集群时停止，或在获得满意集群结果的时候停止

K-medoids

为克服异常值的影响而发明，可以处理更一般类型的数据，核心思想为每个集群的中心点仅限于分配给该集群的观测值之一（不是大话啊）

轮换迭代：

- 给定 C ，解 $m_k = x_{i_k^*}$ 使得集群中点的分散指标最小： $i_k^* = \arg \min_{\{i:C(i)=k\}} \sum_{C(j)=k} d(x_i, x_j)$
- 给定 m_k ，求解 C ： $C(i) = \arg \min_{1 \leq k \leq K} d(x_i, m_k)$

比 K-Means 健壮性更强

在轮换迭代的第 1 个步骤中，时间复杂度为 $O(n_k^2)$ ，比 K-Means 的 $O(n_k)$ 更高

其他变种

- K-Median：使用曼哈顿距离 (L^1 -距离) 代替；中心不是平均值，而是中位数
- K-Means++：选择彼此远离的初始中心
- Rough-set-based K-means：每个样本都可以被分配到多个集群中

分层群聚 (Hierarchical Clustering)

在不同的层次结构中进行聚类，生成树状结构

两种方式：

- 凝聚聚类 Agglomerate clustering：自下而上
- 分散聚类 Divisive clustering：自上而下

不足之处是，一旦合并或分割，该操作就不能被修改

凝聚聚类

给定 n 个样本和不相似度矩阵，执行以下步骤

1. 让每个观测结果都代表一个单例集群
2. 将两个最近的集群合并为一个集群
3. 计算新的不相似度矩阵（两个集群之间的差异）
4. 重复步骤 2 和步骤 3，直到将所有样本都合并到一个集群中

计算组间不相似度的三种方法：

- Single linkage: 最大相似或最不相似 $d_{SL}(C_i, C_j) = \min_{x \in C_i, y \in C_j} d(x, y)$
- Complete linkage: 最小相似性或最大差异性 $d_{CL}(C_i, C_j) = \max_{x \in C_i, y \in C_j} d(x, y)$
- Average linkage: 平均相似度或不相似度 $d_{AL}(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{x \in C_i, y \in C_j} d(x, y)$

凝聚聚类算法：

- Input : training set $D = \{(x_1), \dots, (x_n)\}$, dissimilarity function $d(C_i, C_j)$
 - Output : A dendrogram containing $\{\mathcal{R}_t\}_{t=0}^{n-1}$, where \mathcal{R}_t is the clustering result at time t
1. Initialize the clustering result $\mathcal{R}_0 = \{\{x_1\}, \{x_2\}, \dots, \{x_n\}\}$, $t = 0$
 2. Do iterations :
 - 2.1 $t = t + 1$
 - 2.2 Choose (C_i, C_j) from \mathcal{R}_{t-1} so that $d(C_i, C_j) = \min_{(r,s)} d(C_r, C_s)$
 - 2.3 $C_q = C_i \cup C_j$
 - 2.4 $\mathcal{R}_t = (\mathcal{R}_{t-1} \setminus \{C_i, C_j\}) \cup \{C_q\}$
 3. Stop at $t = n - 1$ when $|\mathcal{R}_{n-1}| = 1$, return $\{\mathcal{R}_t\}_{t=0}^{n-1}$

分散聚类

分散聚类算法

- Input : training set $D = \{(x_1), \dots, (x_n)\}$, dissimilarity function $d(C_i, C_j)$
 - Output : A dendrogram containing $\{\mathcal{R}_t\}_{t=0}^{n-1}$, where $\mathcal{R}_t = \{C_{t,i}\}_{i=1}^{t+1}$ is the clustering result at time t
1. Initialize $\mathcal{R}_0 = \{X\}$, $t = 0$
 2. Do iterations :
 - 2.1 $t = t + 1$
 - 2.2 For $i = 1$ to t , do :
 - 2.2.1 Choose $(C_{t-1,i}^1, C_{t-1,i}^2)$ from $C_{t-1,i}$ so that $d(C_{t-1,i}^1, C_{t-1,i}^2) = \max_{G \cup H = C_{t-1,i}} d(G, H)$
 - 2.2.2 Choose i_{t-1} so that $i_{t-1} = \arg \max_i d(C_{t-1,i}^1, C_{t-1,i}^2)$
 - 2.2.3 $\mathcal{R}_t = (\mathcal{R}_{t-1} \setminus \{C_{t-1,i_{t-1}}\}) \cup \{C_{t-1,i_{t-1}}^1, C_{t-1,i_{t-1}}^2\}$
 - 2.3 Stop at $t = n - 1$ when $|\mathcal{R}_{n-1}| = n$, return $\{\mathcal{R}_t\}_{t=0}^{n-1}$

分层聚类优缺点

优点

- 分层聚类可以一次性计算出整个聚类过程的树状结构
- SL 和 CL 对异常值很敏感, 而 AL 则给出了一个妥协

缺点

- 计算密集
- 一旦一个样本被错误地分组到一个分支中，无论如何对树设置参数或阈值，它永远会在与该分支对应的集群中

DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

分层聚类和 K-means 聚类的局限性：倾向于发现凸集群

基于密度的聚类：寻找由低密度区域分隔的高密度区域，可以发现任何形状的集群

概念

- Core point: 在其 ϵ -邻域内的样本点数量 $\geq \text{MinPts}$
- Boundary point: 在某些核心点的 ϵ -邻域的边界上，且它本身不是核心点
- Noise point: 不是核心点也不是边界点，位于稀疏区域
- ϵ -邻域: $N_\epsilon(x_i) = \{x_j \in D | d(x_i, x_j) \leq \epsilon\}$
- Directly density-reachable: 若 $x_j \in N_\epsilon(x_i)$, 且 x_i 是核心点，则称 x_j 从 x_i 直接密度可达
- Density-reachable: 对于 x_i 和 x_j , 若存在 p_1, p_2, \dots, p_m 使得 $p_1 = x_i, p_m = x_j, p_{k+1}$ 从 p_k 直接密度可达，那么称 x_j 从 x_i 密度可达
- Density-connected: 若存在 p 使得 x_i 和 x_j 都从 p 密度可达，那么称 x_i 和 x_j 密度连接

DBSCAN算法

- Input : training set $D = \{(x_1), \dots, (x_n)\}$, dissimilarity function $d(C_i, C_j)$, parameters MinPts, ϵ
 - Output : a set of clusters $\{C_t\}$
1. Mark all samples in D as non-processed
 2. For each sample $p \in D$, do :
 - 2.1 If p has been grouped into some cluster or marked as noise point, go to check next sample
 - 2.2 Else, if $|N_\epsilon(p)| < \text{MinPts}$, then mark p as boundary point or noise point
 - 2.3 Else, mark p as core point, construct cluster $C = N_\epsilon(p)$. For each $q \in N_\epsilon(p)$, do :
 - 2.3.1 If $|N_\epsilon(q)| \geq \text{MinPts}$, then put all un-clustered points in $N_\epsilon(q)$ into C
 3. Stop when all samples in D have been clustered

算法的时间复杂度为 $O(nt)$, t 为在 ϵ -邻域内搜寻目标的时间，最坏情况下为 $O(n^2)$

在低维数据集中，可以使用 KD-Tree 优化到 $O(n \log n)$

DBSCAN 优缺点

优点

- 能快速地实现聚类
- 能更好的处理噪音点
- 对任何形状的集群都有效

缺点

- 需占用大量内存
- 当密度分布不均匀且集群间距较大时，性能不佳

DBSCAN vs K-Means

DBSCAN	K-Means
聚类结果并不是对原始数据集的完全划分（噪声点被排除了）	聚类结果是对原始数据集的完整划分
可以处理任何形状和大小的集群吗	聚类的集群几乎是球形的
可处理噪声点和离群值	对离群值敏感
对密度的界定很重要	集群中心的选择很重要
在处理高维数据时效率低下	在处理高维数据时效率较好
对样本分布没有隐含的假设	这些样本隐式地遵循高斯分布的假设

最大期望算法 [Expectation-Maximization (EM) Algorithm]

省略，考试对这一部分不做要求

谱聚类 (Spectral Clustering)

相似性图

- ϵ -邻域图： v_i 和 v_j 相连如果 $d(v_i, v_j) < \epsilon$ ，无权图， $\epsilon \sim (\log n/n)^p$ ，需要调参 ϵ
- k 近邻图：如果 v_j 是 v_i 的 k 近邻之一，则连接 v_i 到 v_j ，有向图；如果 v_i 和 v_j 是彼此的 k 近邻之间，则连接 v_i 和 v_j ，相互 k 最近邻图，无向； $k \sim \log n$
- 全连通图：将所有具有正相似性的点连接起来；建模局部邻域关系；高斯相似度函数
 $s(x_i, x_j) = \exp(-\|x_i - x_j\|^2/(2\sigma^2))$ ，其中 σ 控制邻域的宽度；邻接矩阵不是稀疏的； $\sigma \sim \epsilon$

拉普拉斯图

- 非归一化拉普拉斯图： $L = D - W$
 - 有 $\mathbf{1}$ 作为特征向量对应 0 特征值
 - 对称和正定： $\mathbf{f}^T \mathbf{L} \mathbf{f} = \frac{1}{2} \sum_{i,j} w_{ij} (f_i - f_j)^2$
 - 非负的、实值的特征值： $0 = \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$
 - 特征值 0 的特征空间由向量 $\mathbf{1}_{A_1}, \dots, \mathbf{1}_{A_k}$ 张成，其中 A_1, \dots, A_k 是图中的 k 个连通分量
- 归一化拉普拉斯图
 - 对称拉普拉斯函数： $L_{sym} = D^{-1/2} L D^{-1/2}$
 - 随机游走拉普拉斯函数： $L_{rw} = D^{-1} L$
 - 两者都有与 L 相似的性质

谱聚类

Graph cut: 将 G 分为 K 组 A_1, \dots, A_K ，其中 $A_i \subset V$ ，这相当于最小化图切割函数

$cut(A_1, \dots, A_K) = \frac{1}{2} \sum_{k=1}^K W(A_k, \overline{A_k})$ ，其中 $W(A, B) = \sum_{i \in A, j \in B} w_{ij}$ 。简单的解由一个单例及其补组成。

RatioCut: $\text{RatioCut}(A_1, \dots, A_K) = \frac{1}{2} \sum_{k=1}^K \frac{W(A_k, \bar{A}_k)}{|A_k|}$, 其中 $|A|$ 是 A 中的点的数量

Normalized cut: $Ncut(A_1, \dots, A_K) = \frac{1}{2} \sum_{k=1}^K \frac{W(A_k, \bar{A}_k)}{\text{vol}(A_k)}$, 其中 $\text{vol}(A_k) = \sum_{i \in A} d_i$, 这个计算是 NP

困难的

Relaxation of RatioCut to Eigenvalue Problems with $K = 2$

- $\min_{A \subset V} \text{RatioCut}(A, \bar{A})$
- Binary vector $f = (f_1, \dots, f_n)^T$ as indicator function :

$$f_i = \begin{cases} \sqrt{|\bar{A}|}/|\bar{A}|, & \text{if } v_i \in A \\ -\sqrt{|\bar{A}|}/|\bar{A}|, & \text{if } v_i \in \bar{A} \end{cases}$$
- $f^T L f = |V| \cdot \text{RatioCut}(A, \bar{A})$, $\sum_{i=1}^n f_i = 0$, and $\|f\|_2^2 = n$
- Relax f to be real-valued : $\min_{f \in \mathbb{R}^n} f^T L f$, subject to $f \perp \mathbf{1}$ and $\|f\|_2 = \sqrt{n}$
- By Rayleigh-Ritz theorem, the solution f is the eigenvector corresponding to the second smallest eigenvalue of L
- Cluster $\{f_i\}_{i=1}^n$ to two groups C and \bar{C} : $v_i \in A$ if $f_i \in C$, and else $v_i \in \bar{A}$

Relaxation of RatioCut and Ncut with general K

- RatioCut
 - Binary vector $h_j = (h_{1j}, \dots, h_{nj})^T$, $j = 1, \dots, K$, as indicator function : $h_{ij} = \begin{cases} 1/\sqrt{|A_j|}, & \text{if } v_i \in A_j \\ 0, & \text{otherwise} \end{cases}$
 - $h_j^T L h_j = \text{Cut}(A_j, \bar{A}_j)/|A_j|$, $H = (h_1, \dots, h_K) \in \mathbb{R}^{n \times K}$, $\text{RatioCut}(A_1, \dots, A_K) = \text{Tr}(H^T L H)$, $H^T H = I$
 - Relax $H : \min_{H \in \mathbb{R}^{n \times K}} \text{Tr}(H^T L H)$, subject to $H^T H = I$
 - Solution : the first K eigenvectors of L as columns
 - Cluster the rows of H to K groups
- Ncut
 - Replacing $|A_j|$ by $\text{vol}(A_j)$, the same argument for the relaxation of Ncut : $\min_{H \in \mathbb{R}^{n \times K}} \text{Tr}(H^T L H)$, subject to $H^T D H = I$
 - Solution : the first K eigenvectors of L_{rw} as columns

谱聚类算法：

Spectral Clustering Algorithm

- Input : Similarity matrix $S \in \mathbb{R}^{n \times n}$, number k of clusters
- Output : Clusters A_1, \dots, A_K of indices of vertices
- Algorithm :
 1. Construct a similarity graph $G = (V, E)$ with weighted adjacency matrix W
 2. Compute the unnormalized graph Laplacian L or normalized graph Laplacian L_{sym} or L_{rw}
 3. Compute the first K eigenvectors $\mathbf{U} = [\mathbf{u}_1, \dots, \mathbf{u}_K] \in \mathbb{R}^{n \times K}$
 4. In the case of L_{sym} , normalize the rows of \mathbf{U} to norm 1; for the other two cases, skip this step
 5. Let $\mathbf{y}_i \in \mathbb{R}^K$ be the i -th row of \mathbf{U} , use K-means to cluster the point set $\{\mathbf{y}_i\}_{i=1}^n$ into clusters C_1, \dots, C_K
 6. $A_k = \{i | y_i \in C_k\}$

聚类模型评估

外部指标：通过真实标签或比较两个集群得到，比如 Purity, Jaccard coefficient and Rand index, Mutual information

内部指标：不通过外部信息得到，基于集群内相似性和集群间距离，比如 Davies-Bouldin index (DBI), Silhouette coefficient (SC)

Purity

n_{ij} 表示标签是 j 但聚类到集群 i 的样本个数

$$n_i = \sum_{j=1}^C n_{ij} \text{ 是集群 } i \text{ 的总样本数}$$

$p_{ij} = n_{ij}/n_i$ 是在集群 i 中的概率分布

集群 i 的纯度为 $p_i = \max_j p_{ij}$

总纯度定义为 $\sum_i \frac{n_i}{n} p_i$

Confusion Matrix

	Same Cluster	Different Cluster
Same Class	SS (True Positive or TP)	DS (False Negative or FN)
Different Class	SD (False Positive or FP)	DD (True Negative or TN)

Jaccard Coefficient

$$JC = \frac{SS}{SS + SD + DS} \in [0, 1]$$

Rand index

$$RI = \frac{SS + DD}{SS + SD + DS + DD} \in [0, 1]$$

Mutual Information (简略)

维基链接

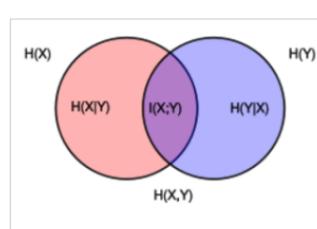
- Mutual information (MI) measures the uncertainty decrement of one random variable given another random variable
- Probability that a sample belongs to both cluster u_i and v_j : $p_{UV}(i, j) = \frac{|u_i \cap v_j|}{n}$
- Its marginal probabilities are : $p_U(i) = \frac{u_i}{n}$ and $p_V(j) = \frac{v_j}{n}$
- Mutual information : $I(U, V) = \sum_{i=1}^R \sum_{j=1}^C p_{UV}(i, j) \log \frac{p_{UV}(i, j)}{p_U(i)p_V(j)}$
- MI attains its maximum $\min\{H(U), H(V)\}$ only when we have many small clusters
- Normalized MI : $NMI(U, V) = \frac{I(U, V)}{(H(U) + H(V))/2}$

- Entropy : $H(X) = - \sum_x p(x) \log p(x)$

- Conditional entropy :

$$\begin{aligned} H(X|Y) &= \sum_y p(y) H(X|Y=y) \\ &= \sum_y p(y) \left(- \sum_x p(x|y) \log p(x|y) \right) \end{aligned}$$

- MI : $I(X; Y) = H(X) - H(X|Y)$



Davies-Bouldin Index

DBI 测量集群内发散性和集群间距离

$$DBI = \frac{1}{k} \sum_{i=1}^k \max_{j \neq i} \left(\frac{div(c_i) + div(c_j)}{d(\mu_i, \mu_j)} \right)$$

其中 $div(c_i)$ 表示集群 c_i 内样本的平均距离, μ_i 是 c_i 的质心

DBI 越小说明聚类效果越好

Silhouette Coefficient

$$SC = \frac{b_i - a_i}{\max(a_i, b_i)} \in [-1, 1]$$

其中, a_i 是第 i 个样本与同一集群中所有其他样本之间的平均距离, b_i 是从第 i 个样本到其他集群的最小距离

SC 越大, 说明聚类效果越好

8. 降维 Dimensionality Reduction

$f : \mathcal{X} \rightarrow \mathcal{Y}$, 其中 $\dim \mathcal{X} > \dim \mathcal{Y}$

降维的原因:

- 在实际应用中, 样本的数量是有限的
- 由于高维数据的稀疏性, 很容易进行过拟合
- 很难训练一个好的模型来对边界数据 (在高维更多) 进行分类

降维能做什么:

- 数据压缩 (Data compression)
- 去噪声
- 通过映射和特征选择来进行特征提取 (比如 LASSO)
- 降低空间和时间的复杂性, 从而需要更少的参数和更小的计算能力
- 可视化

分类:

- 线性降维
 - Principal component analysis (PCA)
 - Linear discriminant analysis (LDA)
 - Independent component analysis (ICA)
- 非线性降维
 - Kernel based methods (Kernel PCA)
 - Manifold learning (ISOMAP, Locally Linear Embedding (LLE), Multidimensional scaling (MDS), t-SNE)

PCA

方差: $Var(X) = \mathbb{E}(X - \mathbb{E}X)^2$

样本方差: $S^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2$

相关系数: $Cov(X, Y) = \mathbb{E}(X - \mathbb{E}X)(Y - \mathbb{E}Y)$

样本相关系数: $C = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})$

若 $X = (\mathbf{x}_1, \dots, \mathbf{x}_n)^T \in \mathbb{R}^{n \times p}$ 是样本矩阵,

$C = \frac{1}{n-1} (X - \mathbf{1}_n \bar{\mathbf{x}}^T)^T (X - \mathbf{1}_n \bar{\mathbf{x}}^T) = \frac{1}{n-1} (X - \frac{1}{n} \mathbf{1}_n \mathbf{1}_n^T X)^T (X - \frac{1}{n} \mathbf{1}_n \mathbf{1}_n^T X) = \frac{1}{n-1} X^T J X$, 其中 $J = \mathbf{I}_n - \frac{1}{n} \mathbf{1}_n \mathbf{1}_n^T$ 是一个秩为 $n-1$ 的投影矩阵

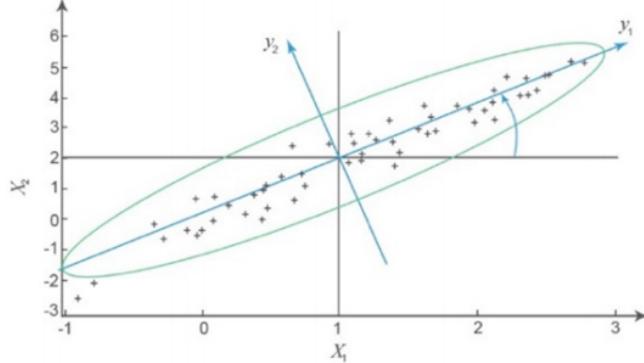
PCA

- PCA 通过使用正交变换, 将一组强相关变量转换为另一组 (通常要小得多) 的弱相关变量
- 这些新的变量被称为主成分 (principal components)
- 新的变量集是原始变量的线性组合, 其方差信息被尽可能地继承
- 是无监督学习

可视化解释

假设一组二维数据遵循高斯分布 (但不限于高斯分布!), 通过采用较大方差 (数据变异性较大) 的方向, 成功地降低到一维

主轴的方向比其他方向包含更多的信息, 因为较小的方差表明变量包含的信息几乎相同



样本协方差矩阵的特征分解

$\{e_i\}_{i=1}^p$ 是欧几里得空间的标准基, 想找到另一组正交基 $\{\tilde{e}_i\}_{i=1}^p$ 使得随机向量 $v = \sum_{i=1}^p x_i e_i$ 可以在新的基中被表示 $v = \sum_{i=1}^p \tilde{x}_i \tilde{e}_i$, 且 $Var(\tilde{x}_1) \geq \dots \geq Var(\tilde{x}_n)$ 且对于 $i \neq j$ 有 $Cov(\tilde{x}_i, \tilde{x}_j) \approx 0$

通过线性代数, 由线性变换得到坐标变换 $(\tilde{e}_1, \dots, \tilde{e}_p) = (e_1, \dots, e_p)W$, 其中 $W \in \mathbb{R}^{p \times p}$, 并对分量系数进行了相应的变换 $x = W\tilde{x}$

假设我们有 n 个中心化的样本 $\{x_i\}_{i=1}^n$ 且 $\frac{1}{n} \sum_{i=1}^n x_i = \mathbf{0}_p$

$$X^T = (x_1, \dots, x_n) = W(\tilde{x}_1, \dots, \tilde{x}_n) = W\tilde{X}^T$$

$$Cov(X) = \frac{1}{n-1} X^T X$$

$$Cov(\tilde{X}) = \frac{1}{n-1} \tilde{X}^T \tilde{X} = \frac{1}{n-1} W^T X^T X W = W^T Cov(X) W$$

它的对角线是 $Var(\tilde{x}_1), \dots, Var(\tilde{x}_p)$, 非对角线是 \tilde{x}_i 和 \tilde{x}_j 协方差

需要 $Cov(\tilde{X})$ 几乎是对角线的, 且对角线项递减, 相当于进行特征分解:

$Cov(X) = Odiag(\lambda_1, \dots, \lambda_p)O^T$, $O \in \mathbb{R}^{p \times p}$ 是正交矩阵且 $\lambda_1 \geq \dots \geq \lambda_p \geq 0$ 并让 $W = O$

解释:

- 转换后的变量中的方差: $Var(\tilde{x}_i) = \lambda_i$, $Cov(X)$ 的特征值
- 新的基, $W = O$ 的每一列
- 百分比 $\frac{\lambda_i}{\sum \lambda_i}$ 代表新变量 \tilde{x}_i 的重要性
- 对于任意向量 $x \in \mathbb{R}^p$, 对应的 r 个主成分是这样表示的 $w_1^T x, \dots, w_r^T x$

从 Best Reconstruction 的角度

Another Viewpoint - Best Reconstruction

- Note that the new basis $\{\tilde{\mathbf{e}}_j\}_{j=1}^p$ is given by $\tilde{\mathbf{e}}_j = \mathbf{w}_j$;
- After the projection (if we keep the first r components), the projected point of each sample \mathbf{x}_i is $\tilde{\mathbf{x}}_{i,1}\mathbf{w}_1 + \dots + \tilde{\mathbf{x}}_{i,r}\mathbf{w}_r$, where the coordinate is given by $\tilde{x}_{i,j} = \mathbf{w}_j^T \mathbf{x}_i$;
- The reconstruction error is the sum of all squared L^2 errors of all samples :

$$\begin{aligned} RE(W) &= \sum_{i=1}^n \left\| \sum_{j=1}^r \tilde{x}_{i,j} \mathbf{w}_j - \mathbf{x}_i \right\|_2^2 = \sum_{i=1}^n \| (W_r W_r^T - I) \mathbf{x}_i \|_2^2 \\ &= \sum_{i=1}^n \mathbf{x}_i^T (I - W_r W_r^T) \mathbf{x}_i = Tr \left(\sum_{i=1}^n \mathbf{x}_i \mathbf{x}_i^T (I - W_r W_r^T) \right) \\ &= Tr(X^T X (I - W_r W_r^T)) = Tr(X^T X) - Tr(W_r^T X^T X W_r) \end{aligned}$$

- Resulting in an optimization problem :

$$\min_{W_r} -Tr(W_r^T X^T X W_r), \quad \text{subject to } W_r^T W_r = I$$

PCA Algorithm

- Given the data matrix $X = (x_1, \dots, x_n)^T \in \mathbb{R}^{n \times p}$ and a threshold t (in some other cases, the number of principal components r) :
 - Centralize the data by their mean $\bar{x} = \frac{1}{n}\mathbf{1}_n^T X$, and compute the sample covariance matrix $C = \frac{1}{n-1}(X - \mathbf{1}_n\bar{x}^T)^T(X - \mathbf{1}_n\bar{x}^T)$
 - Compute the eigenvalues $\{\lambda_i\}_{i=1}^p$ and the corresponding eigenvectors $\{w_i\}_{i=1}^p$
 - Order the eigenvalues as $\lambda_{(1)} \geq \dots \geq \lambda_{(p)}$, and compose an orthogonal matrix W by the eigenvectors columnwise in the same order : $W = (w_1, \dots, w_p)$
 - Compute the variance contribution of the first r eigenvalues : $\sum_{i=1}^r \frac{\lambda_i}{\sum_{j=1}^p \lambda_j}$, find a suitable r such that this variance contribution is greater than the threshold t
 - Pick the first r columns in W and form a matrix $W_r = (w_1, \dots, w_r) \in \mathbb{R}^{p \times r}$
 - Output $\tilde{X}_r = XW_r \in \mathbb{R}^{n \times r}$ as the projected data matrix, whose rows consist of data points in r dimensional subspace

应用实例：

An Example

- The data : the monthly prices of three brands of vehicles (Jeep : x_1 , Toyota : x_2 , Benz : x_3)
- The covariance matrix is given by

$$C = \begin{pmatrix} 1 & \frac{2}{\sqrt{10}} & -\frac{2}{\sqrt{10}} \\ \frac{2}{\sqrt{10}} & 1 & -\frac{4}{5} \\ -\frac{2}{\sqrt{10}} & -\frac{4}{5} & 1 \end{pmatrix}$$

- Compute the characteristic polynomial :

$$\det(\lambda I - C) = \begin{vmatrix} \lambda - 1 & -\frac{2}{\sqrt{10}} & \frac{2}{\sqrt{10}} \\ -\frac{2}{\sqrt{10}} & \lambda - 1 & \frac{4}{5} \\ \frac{2}{\sqrt{10}} & \frac{4}{5} & \lambda - 1 \end{vmatrix}$$

- Solve for the eigenvalues : $\lambda_1 = 2.38$, $\lambda_2 = 0.42$, $\lambda_3 = 0.2$

An Example (Cont')

- Plug in each eigenvalues and solve for the corresponding eigenvectors, e.g., $(\lambda_1 I - C)w_1 = \mathbf{0}$, or equivalently,

$$\begin{cases} 1.38w_{11} - \frac{2}{\sqrt{10}}w_{12} + \frac{2}{\sqrt{10}}w_{13} = 0, \\ -\frac{2}{\sqrt{10}}w_{11} + 1.38w_{12} + 0.8w_{13} = 0, \\ \frac{2}{\sqrt{10}}w_{11} + 0.8w_{12} + 1.38w_{13} = 0. \end{cases}$$

- One can find three eigenvectors as $w_1 = (0.54, 0.59, -0.59)^T$, $w_2 = (0.84, -0.39, 0.39)^T$, $w_3 = (0, 0.71, 0.71)^T$
- The three components are

$$\begin{aligned} \tilde{x}_1 &= w_1^T x = 0.54x_1 + 0.59x_2 - 0.59x_3, \\ \tilde{x}_2 &= w_2^T x = 0.84x_1 - 0.39x_2 + 0.39x_3, \\ \tilde{x}_3 &= w_3^T x = 0.71x_2 + 0.71x_3. \end{aligned}$$

- As $\lambda_1 \gg \lambda_2, \lambda_3$, the first principal component \tilde{x}_1 reflects the change of prices in all three brands of vehicles

LDA

LDA 属于监督学习，在标签的基础上进行线性投影，以最大限度地提高低维类间点分散性（可变性）

每个类中的样本数量为 n_k , 总样本数量为 n

第 k 类样本的平均值为 $\mu_k = \frac{1}{n_k} \sum_{i:x_i \in C_k} x_i$, 所有样本的平均值为 μ

在投影之前, 类间点散度为 $S_b = \sum_{k=1}^K \frac{n_k}{n} (\mu_k - \mu)(\mu_k - \mu)^T$, 投影后, 投影矩阵为 $W_r \in \mathbb{R}^{p \times r}$, 类间点散度为 $\tilde{S}_b = W_r^T S_b W_r$

在投影之前, 每个类 C_k 的类内点散度(方差)为 $S_k = \frac{1}{n_k} \sum_{i:x_i \in C_k} (x_i - \mu_k)(x_i - \mu_k)^T$, 因此, 类内点总散度是 $S_w = \sum_{k=1}^K \frac{n_k}{n} S_k$

投影后, 每个类 C_k 的类内点散度(方差)为 $\tilde{S}_k = W_r^T S_k W_r$, 类内点总散度是 $\tilde{S}_w = W_r^T S_w W_r$

LDA 转化为优化问题即为, 要找到一个 W_r , 使得类间点散度 \tilde{S}_b 最大而类内点散度 \tilde{S}_w 最小, 即

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

这和以下等价

$$\max_w J_b(w) = w^T S_b w, \text{ subject to } w^T S_w w = 1$$

使用拉格朗日乘子法, 定义 $L(w, \lambda) = w^T S_b w - \lambda(w^T S_w w - 1)$

$$\nabla_w L = 2S_b - 2\lambda S_w w = 0 \Rightarrow S_w^{-1} S_b w = \lambda w$$

最优解的方向是 $S_w^{-1} S_b$ 的特征向量

应用实例:

An Example

- Given two sets of data : class 1 is $\{(4, 1)^T, (2, 4)^T, (2, 3)^T, (3, 6)^T, (4, 4)^T\}$, and class 2 is $\{(9, 10)^T, (6, 8)^T, (9, 3)^T, (8, 7)^T, (10, 8)^T\}$
- Class means : $\mu_1 = (3, 3.6)^T$, $\mu_2 = (8.4, 7.6)^T$, the point scatter metrics are

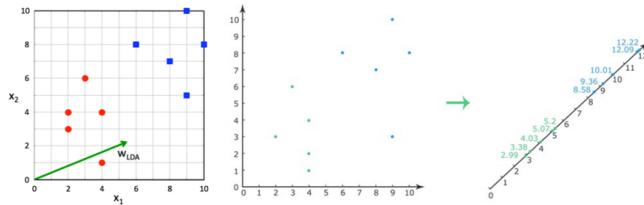
$$S_1 = \begin{pmatrix} 0.8 & -0.4 \\ -0.4 & 2.6 \end{pmatrix}, \quad S_2 = \begin{pmatrix} 1.84 & -0.28 \\ -0.28 & 5.36 \end{pmatrix},$$
$$S_b = \begin{pmatrix} 7.29 & 4.86 \\ 4.86 & 3.24 \end{pmatrix}, \quad S_w = \begin{pmatrix} 1.32 & -0.34 \\ -0.34 & 4 \end{pmatrix}.$$

- The eigenvalue of $S_w^{-1}S_b$ is solved from

$$0 = \det(\lambda I - S_w^{-1}S_b) = \begin{vmatrix} \lambda - 5.97 & -3.98 \\ -1.72 & \lambda - 1.15 \end{vmatrix} \Rightarrow \lambda = 7.11$$

An Example (Cont')

- The optimal directions is $w^* = (0.96, 0.28)^T$
- After projection, the data become 1D :
 - Class 1 : $\{4.12, 3.03, 2.75, 4.55, 4.95\}$
 - Class 2 : $\{11.42, 7.98, 9.48, 9.63, 11.83\}$



PCA 与 LDA 对比

- PCA
 - 从样本协方差矩阵出发，找到方差最大的方向
 - 无监督学习，作为训练前的步骤，必须与其他学习方法相结合
- LDA
 - 利用标签，找到投影，使之后分类变得更加明显
 - 监督学习，可以用作分类或与其他学习方法相结合

非线性降维

考试不涉及，因此省略了 QAQ