

数据挖掘与应用

分类

授课教师：周晟

浙江大学 软件学院

2025.10.28



数据挖掘十大算法 [1]

- ① C4.5 ✓
- ② CART ✓
- ③ AdaBoost
- ④ SVM[1]
- ⑤ Naive Bayes
- ⑥ EM
- ⑦ Apriori (频繁项挖掘)
- ⑧ k-Means
- ⑨ PageRank
- ⑩ kNN



课程内容

① 支持向量机 SVM

- 函数间隔与几何间隔
- 支持向量机优化

② 贝叶斯分类器

- 贝叶斯决策论
- 贝叶斯分类器
- 拉普拉斯修正

③ 集成学习

- 集成学习概述
- Bagging, Stacking
- Boosting

④ 深度学习中的分类

- 损失函数
- 类别不平衡分类
- 带噪声标签的分类



课程内容

① 支持向量机 SVM

- 函数间隔与几何间隔
- 支持向量机优化

② 贝叶斯分类器

- 贝叶斯决策论
- 贝叶斯分类器
- 拉普拉斯修正

③ 集成学习

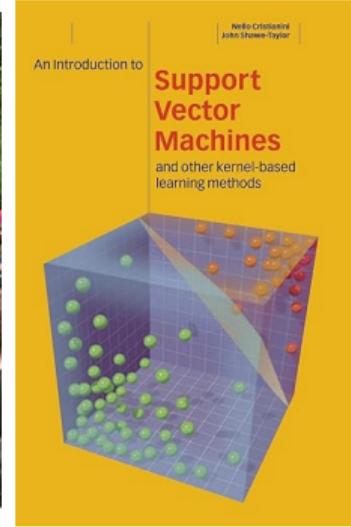
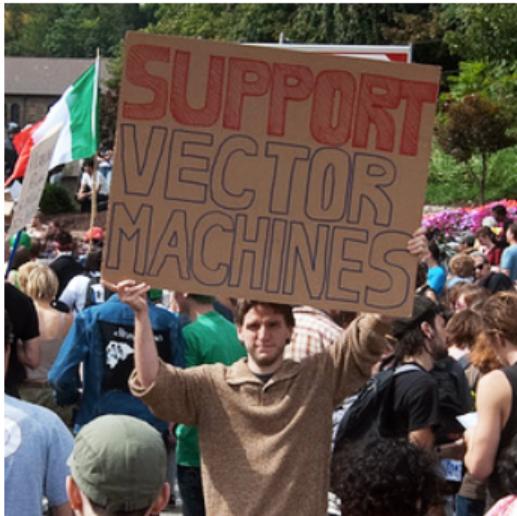
- 集成学习概述
- Bagging, Stacking
- Boosting

④ 深度学习中的分类

- 损失函数
- 类别不平衡分类
- 带噪声标签的分类



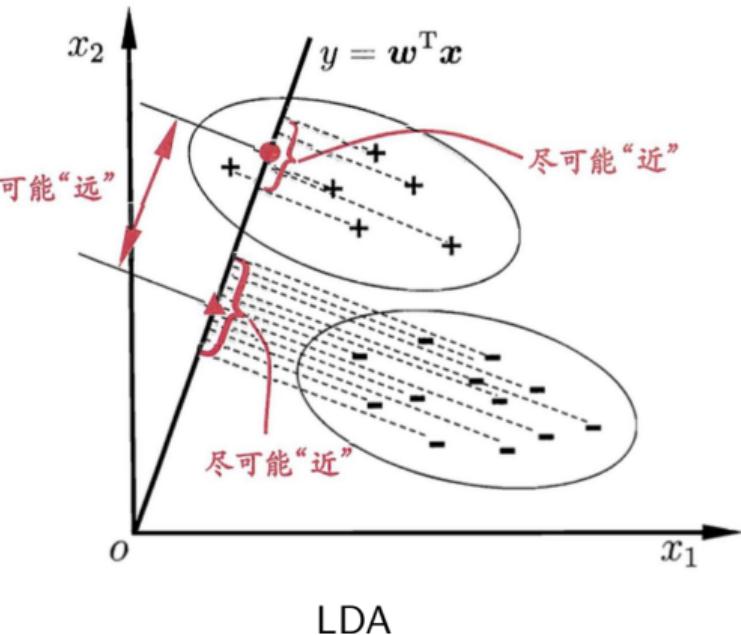
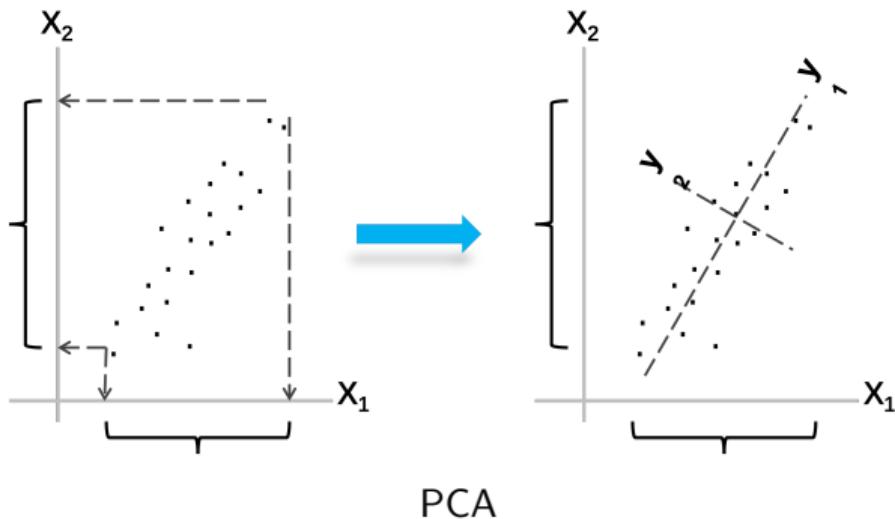
SVM



一直以来学术界和工业界甚至只是学术界里做理论的和做应用的之间，都有一种“鸿沟”
而 SVM 则正好是一个特例，在两边都混得开¹。

¹<https://blog.pluskid.org/archives/632>

向量空间的分类器



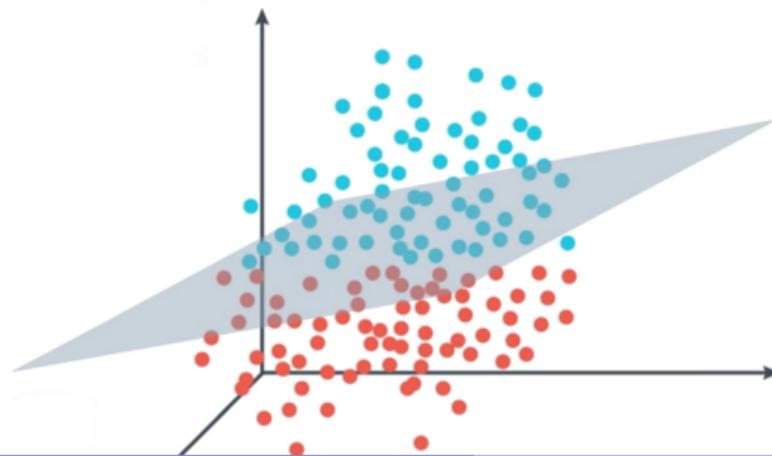
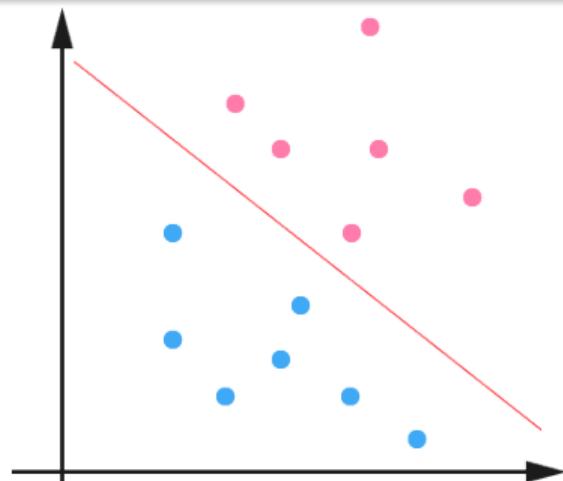
PCA 和 LDA 等方法主要在二维向量空间中展开讨论，而真实的数据空间往往是高维的

分类与超平面

超平面

n 维特征空间中的线性分类器就是要在特征空间中找到一个超平面，使得两类数据尽可能分布在超平面的两侧。超平面可以表示为：

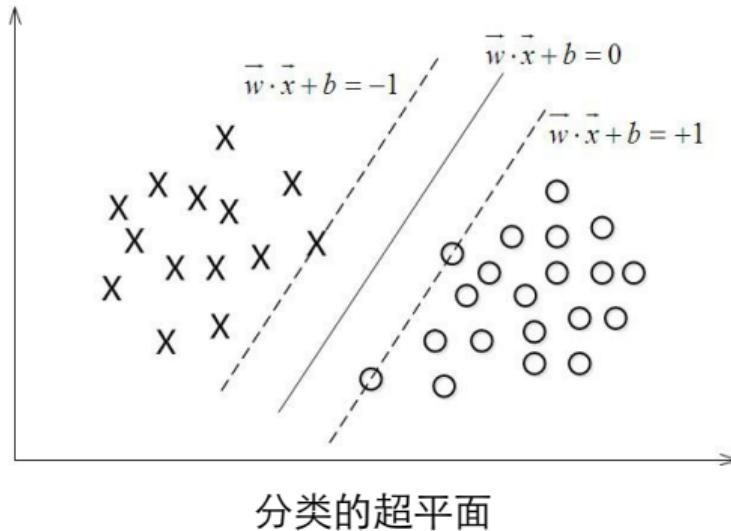
$$w^T x + b = 0$$



分类与超平面

为了计算方便，首先将类别信息数值化。

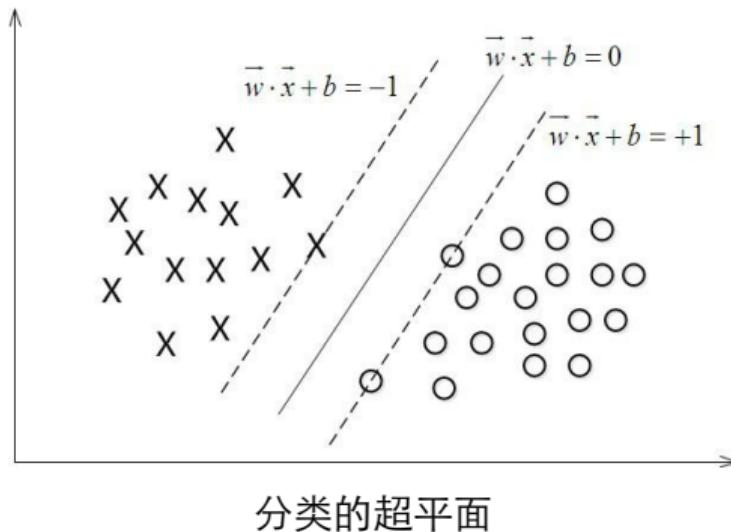
$$f(x) = w^T x + b \begin{cases} > 0 & y = 1 \\ = 0 & \text{超平面} \\ < 0 & y = -1 \end{cases}$$



分类与超平面

为了计算方便，首先将类别信息数值化。

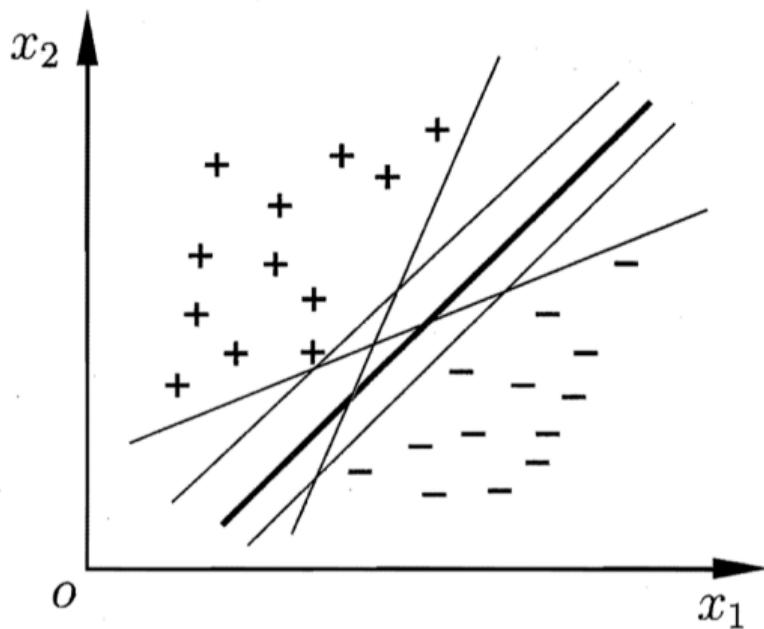
$$f(x) = w^T x + b \begin{cases} > 0 & y = 1 \\ = 0 & \text{超平面} \\ < 0 & y = -1 \end{cases}$$



优点：

- ① $|w^T x + b|$ 表示样本距离超平面的远近
 - ② $f(x) = w^T x + b$ 的正负号可以直接转化为类别
 - ③ $y \cdot f(x) > 0$

超平面的选择



超平面一定存在嘛？超平面如何选择？



超平面的选择



安全行车



弹幕游戏

自然界中的很多场景都是在利用超平面

SVM 的优化目标

SVM 的优化目标 1

SVM 的优化目标是寻找空间中的一个超平面，使得两类节点分布在超平面的两侧且距离超平面尽量远。

① 样本到超平面的距离如何定义？

- ① 函数间隔
- ② 几何间隔

② 如何定义尽量远？

- ① 距离超平面最近的节点定义
- ② 较远节点可以忽略



函数间隔与几何间隔

函数间隔

样本 x_i 的函数间隔 (Functional margin) 定义为：

$$\hat{\gamma}_i = y_i f(x_i) = y_i (w^T x_i + b)$$

用函数值表示分类的置信度

几何间隔

样本 x_i 几何间隔 (Geometrical margin) 定义为点到超平面的距离：

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

函数间隔与几何间隔

定义样本 x 在超平面上的投影为 x_0 , w 是垂直于超平面的向量 (通常取 $y=1$ 的方向), 易得:

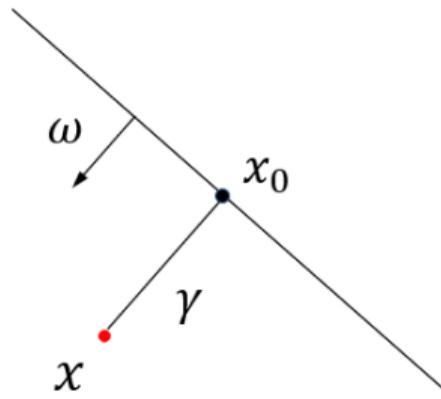
$$x = x_0 + \gamma \frac{yw}{\|w\|}$$

由于 x_0 是超平面上的点, 满足 $f(x_0) = 0$, 易得:

$$\gamma = \frac{y(w^T x + b)}{\|w\|} = \frac{yf(x)}{\|w\|}$$

因此, 函数间隔与几何间隔满足关系:

$$\gamma = \frac{\hat{\gamma}}{\|w\|}$$



SVM 的优化目标

SVM 的优化目标 2

SVM 的优化目标是寻找空间中的一个超平面，使得节点到超平面的函数/几何间隔足够大。

函数间隔的缺陷

通过等比例缩放 $\|w\|$ ，函数间隔可以在超平面不变的情况下被取得任意大，因此在同一超平面下，函数间隔的取值可以无限大，不适合作为优化目标。

最终选择几何间隔为优化目标

$$\max \gamma_i$$

SVM 的优化目标

由于样本中数据非常多，而真正影响决策平面的是距离平面最近的样本点，因此进一步将每个样本的几何间隔 γ_i 转化为全局的几何间隔 γ :

SVM 的优化目标 3

$$\begin{aligned} \max \gamma &= \max \frac{\hat{\gamma}}{\|w\|} = \max \frac{y(w^T x + b)}{\|w\|} \\ s.t. \quad \gamma_i &\geq \gamma, \quad i = 1, \dots, n \end{aligned}$$

最优超平面

为了计算方便，令函数间隔为 1(距离超平面最近的点)：

$$\hat{\gamma} = y (w^T x + b) = yf(x) = 1$$

目标函数转变为：

$$\max \frac{1}{\|w\|} \quad \text{s.t.} \quad y_i(w^T x_i + b) \geq 1, i = 1, \dots, n$$

支持向量

距离超平面最近的点，称之为**支持向量**：

$$y(w^T x + b) = 1$$

特征空间中的其他点 ($y(w^T x + b) > 1$) 点对最有超平面的学习没有实质贡献，因此可以最大程度地提升存储和计算的效率。

支持向量机的优化

支持向量机的优化目标 4

$$\max \frac{1}{\|w\|} \quad \text{s.t. , } y_i (w^T x_i + b) \geq 1, i = 1, \dots, n$$

为了计算方便，将优化目标等价变换为带约束的二次线性优化：

支持向量机的优化目标 5

$$\min \frac{1}{2} \|w\|^2 \quad \text{s.t. , } y_i (w^T x_i + b) \geq 1, i = 1, \dots, n$$

优化方法：

- ① 拉格朗日乘子



使用拉格朗日乘子优化 SVM

利用拉格朗日乘子法，将带约束的优化问题转化为不带约束的优化问题：

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

其中 α_i 是每个约束条件对应的拉格朗日乘子。

令

$$\theta(\mathbf{w}) = \max_{\alpha_i \geq 0} \mathcal{L}(\mathbf{w}, b, \alpha)$$

易证带约束的最小化 $\frac{1}{2} \|\mathbf{w}\|^2$ 等价于最小化 $\theta(\mathbf{w})$



使用拉格朗日乘子优化 SVM

SVM 优化目标

$$\theta(w) = \max_{\alpha_i \geq 0} \mathcal{L}(w, b, \alpha) = \max \frac{1}{2} \|w\|^2 - \sum_{i=1}^n \alpha_i (y_i (\mathbf{w}^T \mathbf{x}_i + b) - 1)$$

为何等价性成立?

- ① 若样本不满足约束条件: $y_i(\mathbf{w}^T \mathbf{x}_i + b) < 1$, 则 $\theta(w) = +\infty$
- ② 若样本满足约束条件, 则 $\alpha_i = 0$, 因此 $\theta(w) = \frac{1}{2} \|w\|^2$

Support Vector!



使用拉格朗日乘子优化 SVM

利用拉格朗日乘子，SVM 的优化目标转化为：

SVM 的优化目标

$$\min_{w,b} \theta(w) = \min_{w,b} \max_{\alpha_i \geq 0} \mathcal{L}(w, b, \alpha) = p^*$$

由于 SVM 优化目标满足 KKT 条件，因此将 SVM 的优化进一步转化为拉格朗日乘子的对偶问题：

$$\max_{\alpha_i \geq 0} \min_{w,b} \mathcal{L}(w, b, \alpha) = d^*$$



SVM 优化目标

SVM 优化目标

$$\theta(w) = \max_{\alpha_i \geq 0} \mathcal{L}(w, b, \alpha) = \max \frac{1}{2} \|w\|^2 - \sum_{i=1}^n \alpha_i (y_i (\mathbf{w}^T \mathbf{x}_i + b) - 1)$$

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

$$\frac{\partial \mathcal{L}}{\partial b} = 0 \Rightarrow \sum_{i=1}^n \alpha_i y_i = 0$$



SVM 的拉格朗日对偶问题求解

代入可得

$$\begin{aligned}
 \mathcal{L}(w, b, \alpha) &= \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j x_i^T x_j - \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j x_i^T x_j \\
 &\quad - b \sum_{i=1}^n \alpha_i y_i + \sum_{i=1}^n \alpha_i \\
 &= \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j x_i^T x_j
 \end{aligned}$$

最终转化为：

$$\begin{aligned}
 &\max_{\alpha} \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j x_i^T x_j \\
 \text{s.t. } &\alpha_i \geq 0, i = 1, \dots, n \\
 &\sum_{i=1}^n \alpha_i y_i = 0
 \end{aligned}$$



SVM 的拉格朗日对偶问题求解

由于在梯度为 0 时，需满足

$$w = \sum_{i=1}^n \alpha_i y_i x_i$$

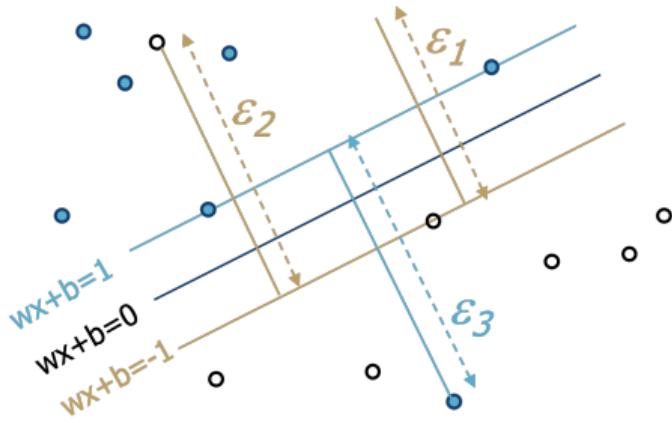
因此对于未知标签的节点，可得：

$$\begin{aligned} f(x) &= \left(\sum_{i=1}^n \alpha_i y_i x_i \right)^T x + b \\ &= \sum_{i=1}^n \alpha_i y_i \langle x_i, x \rangle + b \end{aligned}$$

BONUS: Support Vector 有没有优势?



非线性可分下的 SVM



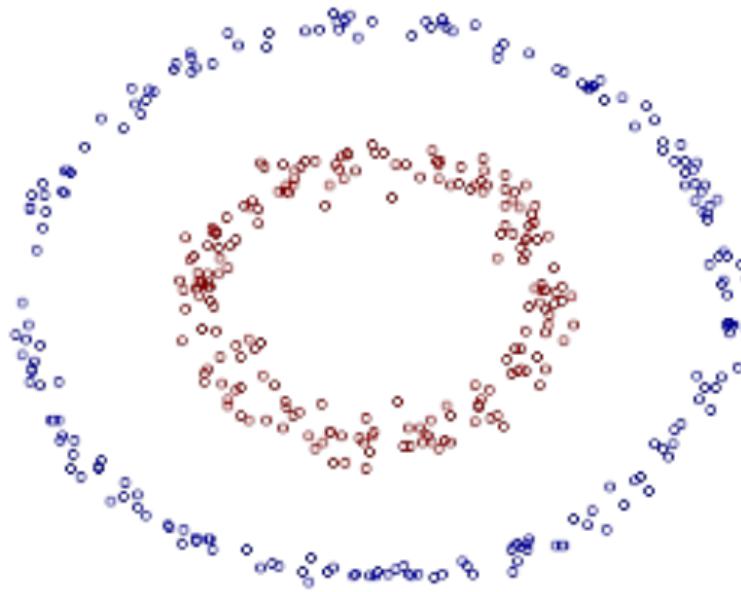
$$\begin{aligned} & \min_{w,b,\xi} \frac{1}{2} w^T w + C \left(\sum_{i=1}^l \xi_i \right) \\ & y_i ((w^T x_i) + b) \geq 1 - \xi_i \\ & \xi_i \geq 0, i = 1, \dots, l \end{aligned}$$

非线性可分下的 SVM

$$L_P \equiv \frac{1}{2} \|w\|^2 + C \sum_{i=1}^l \xi_i - \sum_{i=1}^l \alpha_i [y_i (w \cdot x_i + b) - 1 + \xi_i] - \sum_{i=1}^l \mu_i \xi_i$$

$$L_D \equiv \sum_{i=1}^l \alpha_i - \frac{1}{2} \alpha^T H \alpha \quad \text{s.t. } 0 \leq \alpha_i \leq C \quad \text{and} \quad \sum_{i=1}^l \alpha_i y_i = 0$$

非线性可分下的 SVM



二次曲线的表达形式：

$$a_1X_1 + a_2X_1^2 + a_3X_2 + a_4X_2^2 + a_5X_1X_2 + a_6 = 0$$



非线性可分下的 SVM

在新空间（假设线性可分）中的对偶优化问题：

$$\begin{aligned} \max_{\alpha} & \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j \langle \phi(x_i), \phi(x_j) \rangle \\ \text{s.t.} & \alpha_i \geq 0, i = 1, \dots, n \end{aligned}$$

$$\sum_{i=1}^n \alpha_i y_i = 0$$

在新空间（假设线性可分）中的 SVM：

$$f(x) = \sum_{i=1}^n \alpha_i y_i \langle \phi(x_i), \phi(x) \rangle + b$$



核函数

核函数

计算两个向量在映射过后的空间中的内积的函数叫做核函数 (Kernel Function)，它能简化映射空间中的内积运算

常见的核函数包括：

- ① 线性核 $\kappa(x_1, x_2) = \langle x_1, x_2 \rangle$
- ② 多项式核 $\kappa(x_1, x_2) = (\langle x_1, x_2 \rangle + R)^d$
- ③ 高斯核 $\kappa(x_1, x_2) = \exp\left(-\frac{\|x_1-x_2\|^2}{2\sigma^2}\right)$



SVM 总结

- ① Support Vector 和 Margin
- ② 使用拉格朗日乘子优化 SVM
- ③ 转化为拉格朗日乘子的对偶问题
- ④ 非线性可分下的 SVM
- ⑤ Kernel 函数



1 支持向量机 SVM

- 函数间隔与几何间隔
- 支持向量机优化

2 贝叶斯分类器

- 贝叶斯决策论
- 贝叶斯分类器
- 拉普拉斯修正

3 集成学习

- 集成学习概述
- Bagging, Stacking
- Boosting

4 深度学习中的分类

- 损失函数
- 类别不平衡分类
- 带噪声标签的分类



贝叶斯决策论

贝叶斯决策论

在所有相关概率都已知的理想情形下，贝叶斯决策论考虑如何基于这些概率和误判所造成的损失来选择最优的类别标签。

基于后验概率 $P(c_i|x)$ ，可以获得将样本 x 分类为 c_i 所产生的期望损失，即在样本 x 上的“条件风险”：

$$R(c_i|x) = \sum_{j=1}^K \lambda_{ij} P(c_j|x)$$

$\lambda_{i,j}$ 是将一个真实标签为 c_j 的样本误分类为 c_i 所产生的损失。



贝叶斯决策论

优化目标是最小化分类错误率，则误判损失 λ_{ij} 可以写做：

$$\lambda_{ij} = \begin{cases} 0, & if \quad i = j; \\ 1, & otherwise, \end{cases}$$

此时条件风险为：

$$R(c|x) = 1 - P(c|x) \quad (P(c_j|x) \text{ 为 } 0 \text{ 或 } 1)$$

因此，最小化分类错误率的贝叶斯最优分类器为：

$$h^*(x) = \arg \max_{c \in Y} P(c|x)$$

即对于每个样本 x ，选择能使其后验概率 $P(c|x)$ 最大的类别标签



贝叶斯分类器

贝叶斯分类器

贝叶斯分类器的目标是基于**有限的**训练样本集来**尽可能准确**地估计出后验概率 $P(c|x)$

大体来说，主要有两种策略：

- 给定 x ，可通过直接建模 $P(c|x)$ 来预测 c ，这样得到的是“**判别式模型**”（决策树、逻辑回归、SVM）
- 先对联合概率分布 $P(x, c)$ 建模，然后再由此获得 $P(c|x)$ ，这样得到的是“**生成式模型**”



贝叶斯公式

$$P(c|x) = \frac{P(x,c)}{P(x)} = \frac{P(c)P(x|c)}{P(x)}$$

类先验概率 似然
 ↑
 证据因子

$P(c)$ 是类“先验”概率；

$P(x|c)$ 是样本 x 相对于类标记 c 的类条件概率，或称为“似然”

；

$P(x)$ 是用于归一化的“证据”因子。



Thomas Bayes
1702 - 1761



贝叶斯分类器

$$P(c|x) = \frac{P(c)P(x|c)}{P(x)}$$

证据因子 $P(x)$ 与类标签无关 \Rightarrow 如何基于训练数据 D 来估计先验 $P(c)$ 和似然 $P(x|c)$

根据大数定律，当训练集包含充足的独立同分布样本时，类先验概率 $P(c)$ 可通过各类样本出现的概率进行估计。

对于类条件概率（似然） $P(x|c)$ 来说，直接用频率来估计是不可行的，因为“未被观测到” \neq “出现概率为零”



极大似然估计

统计学界的两个学派分别提供了不同的解决方案：

- 频率主义学派认为参数虽然未知，但却是客观存在的**固定值**。因此，可通过优化似然函数等准则来确定参数值
- 贝叶斯学派则认为参数是未观察到的随机变量，其本身也可有**分布**。因此，可假定参数服从一个先验分布，然后基于观测到的数据来计算参数的后验分布。

极大似然估计源自频率主义学派，是根据数据采样来估计概率分布参数的经典方法。



极大似然估计

极大似然估计

假设 $P(x|c)$ 具有确定的形式并被参数向量 θ_c 唯一确定，令 D_c 表示训练集 D 中第 c 类样本组成的集合，且假设这些样本是独立同分布的。参数 θ_c 对于数据集 D_c 的似然是：

$$P(D_c|\theta_c) = \prod_{x \in D_c} P(x|\theta_c)$$

对 θ_c 进行极大似然估计，就是去寻找能最大化似然 $P(D_c|\theta_c)$ 的参数值 $\hat{\theta}_c$ 。



极大似然估计

上式中的连乘操作易造成下溢，通常使用对数似然（log-likelihood）：

$$\begin{aligned} LL(\theta_c) &= \log P(D_c|\theta_c) \\ &= \sum_{x \in D_c} \log P(x|\theta_c) \end{aligned}$$

此时参数 θ_c 的极大似然估计 $\hat{\theta}_c$ 为：

$$\hat{\theta}_c = \arg \max_{\theta_c} LL(\theta_c)$$



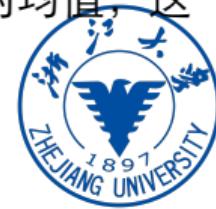
极大似然估计

假设似然函数 $p(x|c) \sim \mathcal{N}(\mu_c, \sigma_c^2)$, 则参数 μ_c 和 σ_c^2 的极大似然估计为:

$$\hat{\mu}_c = \frac{1}{|D_c|} \sum_{x \in D_c} x$$

$$\hat{\sigma}_c^2 = \frac{1}{|D_c|} \sum_{x \in D_c} (x - \hat{\mu}_c)(x - \hat{\mu}_c)^T$$

通过极大似然法得到正态分布均值就是样本均值, 方差就是 $(x - \hat{\mu}_c)(x - \hat{\mu}_c)^T$ 的均值, 这显然是一个符合直觉的结果。



朴素贝叶斯分类器

极大似然估计的问题

类条件概率 $P(x|c)$ 是所有属性上的联合概率，难以从有限的训练样本直接估计而得到。

朴素贝叶斯分类器 (naive Bayes classifier) 采用了**属性条件独立性假设**：对已知类别，假设所有属性相互独立。即假设每个属性独立地对分类结果发生影响。

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

其中 d 为属性数目， x_i 为 x 在第 i 个属性上的取值。



朴素贝叶斯分类器

由于对于所有类别来说 $P(x)$ 相同，因此对应的贝叶斯判定准则为：

$$h_{nb}(x) = \arg \max_{c \in Y} P(c) \prod_{i=1}^d P(x_i|c)$$

这也就是朴素贝叶斯分类器的表达式。

由上式可知，朴素贝叶斯分类器的训练过程就是基于训练集 D 来估计类先验概率 $P(c)$ ，并为每个属性估计条件概率 $P(x_i|c)$ 。



朴素贝叶斯分类器

令 D_c 表示训练集 D 中第 c 类样本组成的集合，若有充足的独立同分布样本，则可容易地估计出类先验概率：

$$P(c) = \frac{|D_c|}{|D|}$$

对离散属性而言，令 D_{c,x_i} 表示 D_c 中在第 i 个属性上取值为 x_i 的样本组成的集合，则条件概率 $P(x_i|c)$ 可估计为：

$$P(x_i|c) = \frac{|D_{c,x_i}|}{|D_c|}$$



朴素贝叶斯分类器

而对于**连续属性**，可以考虑概率密度函数。假定 $p(x_i|c) \sim \mathcal{N}(\mu_{c,i}, \sigma_{c,i}^2)$ ，其中 $\mu_{c,i}$ 和 $\sigma_{c,i}^2$ 分别为第 c 类样本在第 i 个属性上取值的均值和方差，则有：

$$p(x_i|c) = \frac{1}{\sqrt{2\pi}\sigma_{c,i}} \exp\left(-\frac{(x_i - \mu_{c,i})^2}{2\sigma_{c,i}^2}\right)$$



朴素贝叶斯分类器——以西瓜数据集为例

编号	色泽	根蒂	敲声	纹理	脐部	触感	密度	含糖率	好瓜
1	青绿	蜷缩	浊响	清晰	凹陷	硬滑	0.697	0.460	是
2	乌黑	蜷缩	沉闷	清晰	凹陷	硬滑	0.774	0.376	是
3	乌黑	蜷缩	浊响	清晰	凹陷	硬滑	0.634	0.264	是
4	青绿	蜷缩	沉闷	清晰	凹陷	硬滑	0.608	0.318	是
5	浅白	蜷缩	浊响	清晰	凹陷	硬滑	0.556	0.215	是
6	青绿	稍蜷	浊响	清晰	稍凹	软粘	0.403	0.237	是
7	乌黑	稍蜷	浊响	稍糊	稍凹	软粘	0.481	0.149	是
8	乌黑	稍蜷	浊响	清晰	稍凹	硬滑	0.437	0.211	是
9	乌黑	稍蜷	沉闷	稍糊	稍凹	硬滑	0.666	0.091	否
10	青绿	硬挺	清脆	清晰	平坦	软粘	0.243	0.267	否
11	浅白	硬挺	清脆	模糊	平坦	硬滑	0.245	0.057	否
12	浅白	蜷缩	浊响	模糊	平坦	软粘	0.343	0.099	否
13	青绿	稍蜷	浊响	稍糊	凹陷	硬滑	0.639	0.161	否
14	浅白	稍蜷	沉闷	稍糊	凹陷	硬滑	0.657	0.198	否
15	乌黑	稍蜷	浊响	清晰	稍凹	软粘	0.360	0.370	否
16	浅白	蜷缩	浊响	模糊	平坦	硬滑	0.593	0.042	否
17	青绿	蜷缩	沉闷	稍糊	稍凹	硬滑	0.719	0.103	否

朴素贝叶斯分类器——以西瓜数据集为例

编号	色泽	根蒂	敲声	纹理	脐部	触感	密度	含糖率	好瓜
测 1	青绿	蜷缩	浊响	清晰	凹陷	硬滑	0.697	0.460	?

测试数据

首先，我们来估计类先验概率 $P(c)$ ，显然有：

$$P(\text{好瓜} = \text{是}) = \frac{8}{17} \approx 0.471$$

$$P(\text{好瓜} = \text{否}) = \frac{9}{17} \approx 0.529$$



朴素贝叶斯分类器——以西瓜数据集为例

接着，我们为每个属性估计条件概率 $P(x_i|c)$ ：

$$P_{\text{青绿} | \text{是}} = P(\text{色泽} = \text{青绿} | \text{好瓜} = \text{是}) = \frac{3}{8} = 0.375$$

$$P_{\text{青绿} | \text{否}} = P(\text{色泽} = \text{青绿} | \text{好瓜} = \text{否}) = \frac{3}{9} \approx 0.333$$

$$P_{\text{蜷缩} | \text{是}} = P(\text{根蒂} = \text{蜷缩} | \text{好瓜} = \text{是}) = \frac{5}{8} = 0.625$$

$$P_{\text{蜷缩} | \text{否}} = P(\text{根蒂} = \text{蜷缩} | \text{好瓜} = \text{否}) = \frac{3}{9} \approx 0.333$$

$$P_{\text{浊响} | \text{是}} = P(\text{敲声} = \text{浊响} | \text{好瓜} = \text{是}) = \frac{6}{8} = 0.750$$

$$P_{\text{浊响} | \text{否}} = P(\text{敲声} = \text{浊响} | \text{好瓜} = \text{否}) = \frac{4}{8} \approx 0.444$$



朴素贝叶斯分类器——以西瓜数据集为例

$$P_{\text{清晰} | \text{是}} = P(\text{纹理} = \text{清晰} | \text{好瓜} = \text{是}) = \frac{7}{8} = 0.875$$

$$P_{\text{清晰} | \text{否}} = P(\text{纹理} = \text{清晰} | \text{好瓜} = \text{否}) = \frac{2}{9} \approx 0.222$$

$$P_{\text{凹陷} | \text{是}} = P(\text{脐部} = \text{凹陷} | \text{好瓜} = \text{是}) = \frac{6}{8} = 0.750$$

$$P_{\text{凹陷} | \text{否}} = P(\text{脐部} = \text{凹陷} | \text{好瓜} = \text{否}) = \frac{2}{9} \approx 0.222$$

$$P_{\text{硬滑} | \text{是}} = P(\text{触感} = \text{硬滑} | \text{好瓜} = \text{是}) = \frac{6}{8} = 0.750$$

$$P_{\text{硬滑} | \text{否}} = P(\text{触感} = \text{硬滑} | \text{好瓜} = \text{否}) = \frac{6}{8} \approx 0.667$$



朴素贝叶斯分类器——以西瓜数据集为例

$$p_{\text{密度}:0.697} \mid \text{是} = p(\text{密度} = 0.697 \mid \text{好瓜} = \text{是})$$

$$= \frac{1}{\sqrt{2\pi} \cdot 0.129} \exp\left(-\frac{(0.697 - 0.574)^2}{2 \cdot 0.129^2}\right) \approx 1.959$$

$$p_{\text{密度}:0.697} \mid \text{否} = p(\text{密度} = 0.697 \mid \text{好瓜} = \text{否})$$

$$= \frac{1}{\sqrt{2\pi} \cdot 0.195} \exp\left(-\frac{(0.697 - 0.496)^2}{2 \cdot 0.195^2}\right) \approx 1.203$$

$$p_{\text{含糖}:0.460} \mid \text{是} = p(\text{含糖} = 0.460 \mid \text{好瓜} = \text{是})$$

$$= \frac{1}{\sqrt{2\pi} \cdot 0.101} \exp\left(-\frac{(0.460 - 0.279)^2}{2 \cdot 0.101^2}\right) \approx 0.788$$

$$p_{\text{含糖}:0.460} \mid \text{否} = p(\text{含糖} = 0.460 \mid \text{好瓜} = \text{否})$$

$$= \frac{1}{\sqrt{2\pi} \cdot 0.102} \exp\left(-\frac{(0.460 - 0.154)^2}{2 \cdot 0.102^2}\right) \approx 0.066$$



朴素贝叶斯分类器——以西瓜数据集为例

于是，有

$$\begin{aligned} P(\text{好瓜} = \text{是}) &\times P_{\text{青绿} | \text{是}} \times P_{\text{蜷缩} | \text{是}} \times P_{\text{浊响} | \text{是}} \times P_{\text{清晰} | \text{是}} \times P_{\text{凹陷} | \text{是}} \\ &\times P_{\text{硬滑} | \text{是}} \times p_{\text{密度}:0.697 | \text{是}} \times p_{\text{含糖}:0.460 | \text{是}} \approx 0.063 \end{aligned}$$

$$\begin{aligned} P(\text{好瓜} = \text{否}) &\times P_{\text{青绿} | \text{否}} \times P_{\text{蜷缩} | \text{否}} \times P_{\text{浊响} | \text{否}} \times P_{\text{清晰} | \text{否}} \times P_{\text{凹陷} | \text{否}} \\ &\times P_{\text{硬滑} | \text{否}} \times p_{\text{密度}:0.697 | \text{否}} \times p_{\text{含糖}:0.460 | \text{否}} \approx 6.80 \times 10^{-5} \end{aligned}$$

由于 $0.063 > 6.80 \times 10^{-5}$ ，因此，朴素贝叶斯分类器将测试样本“测 1”判别为“好瓜”。



朴素贝叶斯分类器——拉普拉斯修正

需注意，若某个属性值在训练集中没有与某个类同时出现过，则基于以上公式进行概率估计和判别将会出现问题。

例如，在使用西瓜数据集训练朴素贝叶斯分类器时，对一个“敲声 = 清脆”的测试用例，有

$$P_{\text{清脆} \mid \text{是}} = P(\text{敲声} = \text{清脆} \mid \text{好瓜} = \text{是}) = \frac{0}{8} = 0$$

因为朴素贝叶斯表达式的连乘结果为零，因此，无论该样本的其他属性是什么，哪怕在其他属性上明显像好瓜，分类的结果都将是“好瓜 = 否”，这显然不太合理。



朴素贝叶斯分类器——拉普拉斯修正

为了避免其他属性携带的信息被训练集中未出现的属性值“抹去”，在估计概率值时通常要进行“平滑”，常用**“拉普拉斯修正”**。

具体来说，令 K 表示训练集 D 中可能的类别数， N_i 表示第 i 个属性可能的取值数，则可将估计概率的公式修正为：

$$\hat{P}(c) = \frac{|D_c| + 1}{|D| + K}$$

$$\hat{P}(x_i|c) = \frac{|D_{c,x_i} + 1|}{|D_c| + N_i}$$



朴素贝叶斯分类器——拉普拉斯修正

例如，在上述例子中，类先验概率可估计为：

$$\hat{P}(\text{好瓜} = \text{是}) = \frac{8+1}{17+2} \approx 0.474, \quad \hat{P}(\text{好瓜} = \text{否}) = \frac{9+1}{17+2} \approx 0.526$$

类似的， $P_{\text{青绿} | \text{是}}$ 可估计为：

$$\hat{P}_{\text{青绿} | \text{是}} = \hat{P}(\text{色泽} = \text{青绿} | \text{好瓜} = \text{是}) = \frac{3+1}{8+3} \approx 0.364$$

同时，上文提到的概率 $P_{\text{清脆} | \text{是}}$ 可估计为：

$$\hat{P}_{\text{清脆} | \text{是}} = \hat{P}(\text{敲声} = \text{清脆} | \text{好瓜} = \text{是}) = \frac{0+1}{8+3} \approx 0.091$$



1 支持向量机 SVM

- 函数间隔与几何间隔
- 支持向量机优化

2 贝叶斯分类器

- 贝叶斯决策论
- 贝叶斯分类器
- 拉普拉斯修正

3 集成学习

- 集成学习概述
- Bagging, Stacking
- Boosting

4 深度学习中的分类

- 损失函数
- 类别不平衡分类
- 带噪声标签的分类



Ensemble Methods

当单一分类模型表现不佳的时候，有没有办法聚合多个模型的决策以达到更好的分类效果呢？

集成方法

为了更好地解决特定的机器学习问题，带有策略地生成和组合多个模型的过程。

- 提升单一模型的性能。
 - 减少选择不佳模型的可能性。
-
- 这类方法被称为模型聚合，常见的方法有：
 - Bagging：将若干分类器预测的结果做平均
 - Random Forest
 - Boosting：将若干分类器预测的结果加权



Ensemble Methods

如何组合分类器的输出结果？

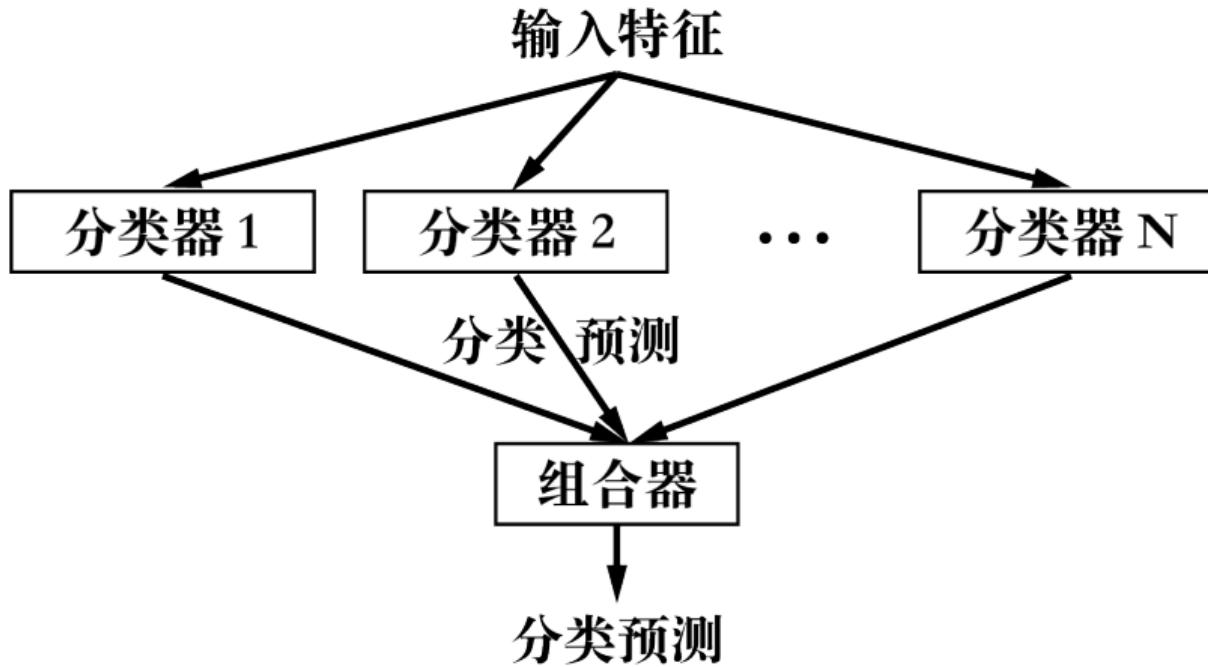
- 求平均
- 投票
 - 多数投票
 - 随机森林
 - 加权多数投票
 - AdaBoost
- 可学习组合器

集成学习的关键

- 分类器之间能够互相纠正彼此的错误。
- 不同分类器有不同的输出结果，否则集成没有意义。



集成流程



Bootstrap

Bootstrap 起源

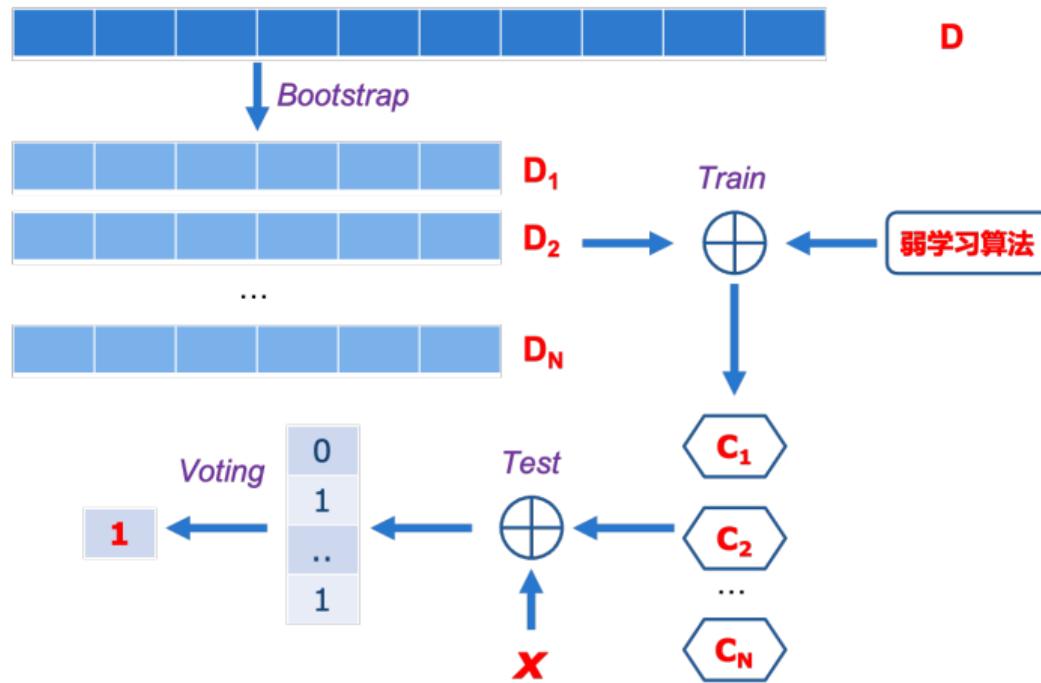
《吹牛大王历险记》“pull yourself up by your bootstraps”，常指自力更生。

集成学习中的 Bootstrap

在机器学习和集成学习中，Bootstrap 往往指对原始数据集进行有放回重采样，形成一系列子样本。

由于很多分类器的精度并不高，因此往往希望通过构造不同的分类器并进行投票来提升整体精度。在模型框架有限的情况下，使用 bootstrap 生成不同的训练数据，可以最低成本地构造不同的分类器。

Bagging: Bootstrap Aggregation



随机森林

随机森林是决策树的扩展，其基本流程如下：

- ① 有放回的抽取 N 次，每次抽取 1 个，最终形成了 N 个样本用来训练一个决策树。
- ② 当每个样本有 M 个属性时，在决策树的每个节点需要分裂时，随机从这 M 个属性中选取出 m ($m \ll M$) 个属性，然后从这 m 个属性中选择 1 个属性作为该节点的分裂属性。
- ③ 重复数据选择和分裂过程，一直到不能够再分裂为止。
- ④ 按照步骤 1, 2, 3 建立大量的决策树，构成随机森林了。



随机森林的优缺点

① 优点：

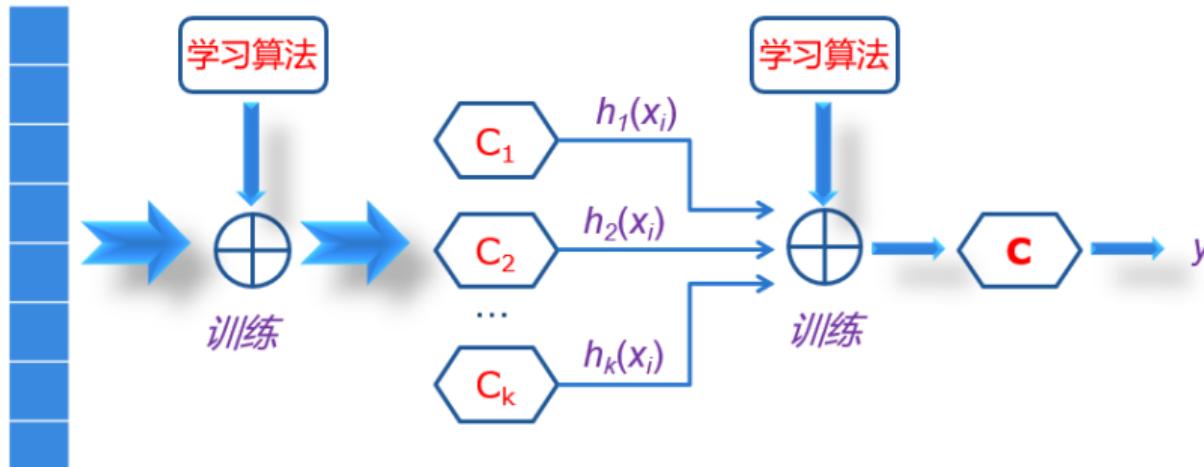
- 1、可以应对高维度（特征很多）的数据，并且不用降维，无需做特征选择。
- 2、可以通过投正确票的决策树的结构判断特征的重要程度。
- 3、不容易过拟合，学习速度快，可以并行训练整个随机森林。
- 4、如果有很大一部分的特征遗失，仍可以维持准确度

② 缺点：

- 1、随机森林无法控制模型内部的运行，只能在不同的参数和随机种子之间进行尝试。
- 2、可能有很多相似的决策树，多数人的暴政掩盖了真实的结果。



Stacking



Boost

Boosting 算法的工作机制：

- ① 从训练集用初始权重训练出一个弱学习器 1，根据弱学习的学习误差率表现来更新训练样本的权重，使得之前弱学习器 1 学习误差率高的训练样本点的权重变高
- ② 基于调整权重后的训练集来训练弱学习器 2.
- ③ 如此重复进行，直到弱学习器数达到事先指定的数目 T ，最终将这 T 个弱学习器通过集合策略进行整合，得到最终的强学习器。

Boosting 算法

AdaBoost (Adaptive Boosting) 是由 Freund 和 Schapire 于 1995 年提出的第一个实用的 Boosting 算法，也是最经典的 Boosting 方法之一。

- 通过迭代，每一轮训练中，提高错误率低的样本的权重，降低错误率高的样本的权重。
- 在每一轮训练中，训练一个弱学习器，并计算其错误率。
- 将所有弱学习器进行加权组合，得到最终的强学习器。

AdaBoost

输入输出

- 输入: 训练数据集 $T = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$, 其中 $x_i \in \mathcal{X} \subseteq R^n, y_i \in \mathcal{Y} = \{+1, -1\}$
- 输出: 分类器 $G(x)$

1. 初始化训练数据的权值分布

$$D_1 = (w_{11}, w_{12}, \dots, w_{1N}), \quad w_{1i} = \frac{1}{N}, \quad i = 1, 2, \dots, N$$

w_{mi} 表示第 m 轮迭代中第 i 个样本的权值

2. 初始化 $m = 1, 2, \dots, M$ 基本分类器

$$G_m(x) : \mathcal{X} \rightarrow \{-1, +1\}$$



AdaBoost

2.2 计算 $G_m(x)$ 在训练数据集上的分类误差率

$$\begin{aligned} e_m &= \sum_{i=1}^N P(G_m(x_i) \neq y_i) \\ &= \sum_{i=1}^N w_{mi} I(G_m(x_i) \neq y_i) \end{aligned}$$

2.3 计算 $G_m(x)$ 的系数

$$\alpha_m = \frac{1}{2} \log \frac{1 - e_m}{e_m}$$



AdaBoost

2.4 更新训练数据集的权值分布

$$\begin{aligned}
 D_{m+1} &= (w_{m+1,1}, \dots, w_{m+1,i}, \dots, w_{m+1,N}) \\
 w_{m+1,i} &= \frac{w_{mi}}{Z_m} \exp(-\alpha_m y_i G_m(x_i)), \\
 &= \begin{cases} \frac{w_{mi}}{Z_m} \exp(-\alpha_m), & G_m(x_i) = y_i \\ \frac{w_{mi}}{Z_m} \exp(\alpha_m), & G_m(x_i) \neq y_i \end{cases} \quad i = 1, 2, \dots, N
 \end{aligned}$$

其中, Z_m 是归一化因子, 为了使概率分布和为 1

$$Z_m = \sum_{i=1}^N w_{mi} \exp(-\alpha_m y_i G_m(x_i))$$

3. 构建基本分类器的线性组合

$$f(x) = \sum_{m=1}^M \alpha_m G_m(x)$$



AdaBoost

得到最终分类器

$$G(x) = \text{sign}(f(x)) = \text{sign} \left(\sum_{m=1}^M \alpha_m G_m(x) \right)$$

注：对于 Adaboost 多元分类算法，其实原理和二元分类类似，最主要区别在弱分类器的系数上。比如 Adaboost SAMME 算法，它的弱分类器的系数：

$$\alpha_m = \frac{1}{2} \log \frac{1 - e_m}{e_m} + \log(R - 1)$$

其中 R 为类别数。从上式可以看出，如果是二元分类， $R = 2$



1 支持向量机 SVM

- 函数间隔与几何间隔
- 支持向量机优化

2 贝叶斯分类器

- 贝叶斯决策论
- 贝叶斯分类器
- 拉普拉斯修正

3 集成学习

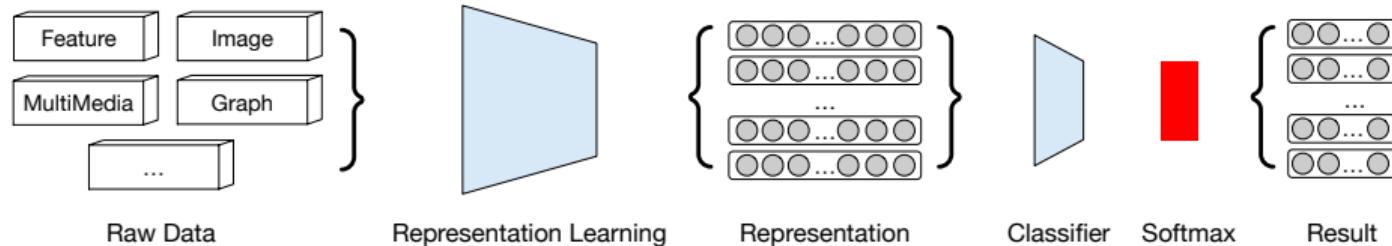
- 集成学习概述
- Bagging, Stacking
- Boosting

4 深度学习中的分类

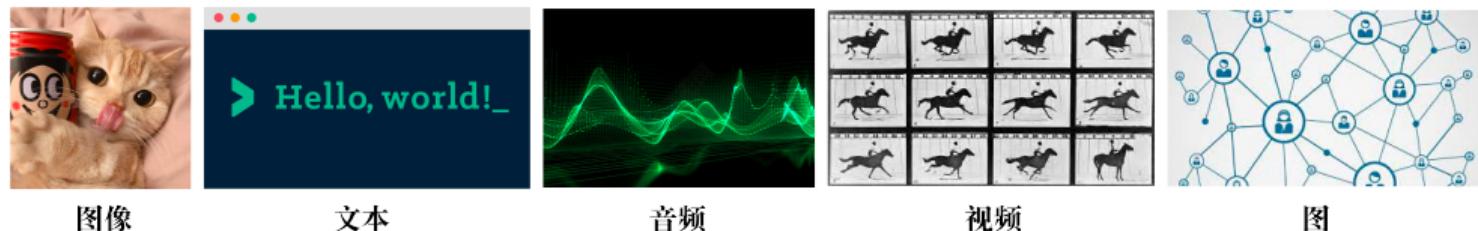
- 损失函数
- 类别不平衡分类
- 带噪声标签的分类



深度学习时代下的分类



深度学习时代的分类框架



图像

文本

音频

视频

图

卷积神经网络
自动编码器

词嵌入
循环神经网络

梅尔频谱倒谱系数
循环神经网络

3D卷积神经网络
时空特征金字塔

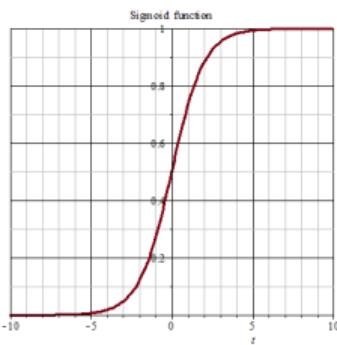
图卷积神经网络
图注意力网络

从 Sigmoid 到 Softmax

在二分类任务中，常常使用 Sigmoid 函数来将模型的输出 t （数值）映射到 $\phi(t) \in [0, 1]$ ，即概率分布的有效实数空间。

$$\phi(t) = \frac{1}{1 + e^{-t}}$$

此时 $\phi(t)$ 为样本属于某一类的概率，而 $1 - \phi(t)$ 则是另一类的概率。



扩展到多分类，此时模型的输出应该为 K 维的向量。

Sigmoid 虽然可以把输出映射到 $[0, 1]$ ，但是要变成概率分布还有一个条件：和为 1。



Softmax

定义 & 作用

Softmax 函数，又名归一化指数函数，能够将一个含任意实数的 K 维向量 \mathbf{z} “压缩” 到另一个 K 维向量 $\sigma(\mathbf{z})$ 中，使得其中每一个元素的取值范围都在 $(0,1)$ 之间，且所有元素的和为 1。

$$\sigma(\mathbf{z})_j = \frac{e^{\mathbf{z}_j}}{\sum_{k=1}^K e^{\mathbf{z}_k}} \text{ for } j = 1, \dots, K.$$

$\sigma(\mathbf{z})$ 通常也被称为 logits，可以表示样本属于每个类的概率。



Softmax 的来源

Softmax，顾名思义，就是 soft + max。

对于多分类，Softmax 的目标不是让模型只输出一个类，而是这个类的概率要比其他类大

$$\mathcal{L} = \sum_{i=1, i \neq y}^C \max(z_i - z_y, 0) \quad (1)$$

借助 LogSumExp trick，可以得到：

$$\mathcal{L}_{\text{lse}} = \max \left(\log \left(\sum_{i=1, i \neq y}^C e^{z_i} \right) - z_y, 0 \right)$$



Softmax 的来源

$\max(x, 0)$ 有一个 smooth 版本，即 softplus: $\log(1 + e^x)$

$$\begin{aligned}\mathcal{L}_{\text{softmax}} &= \log \left(1 + e^{\log(\sum_{i=1, i \neq y}^K e^{z_i}) - z_y} \right) \\ &= \log \left(1 + \frac{\sum_{i=1, i \neq y}^K e^{z_i}}{e^{z_y}} \right) \\ &= \log \frac{\sum_{i=1}^K e^{z_i}}{e^{z_y}} \\ &= -\log \frac{e^{z_y}}{\sum_{i=1}^K e^{z_i}}\end{aligned}\tag{3}$$



Softmax 的优缺点

优点

- 指数函数拉大区别
- 离散-> 连续
- 偏导数计算简便
- 可以表现出不同种类类别之间的关系

缺点

- 指数函数可能导致数值溢出



Softmax 的偏导数

现有的神经网络需要借助梯度进行优化，因此，了解 Softmax 的导数也很有必要。

Softmax

$$y_i = \text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_{k=1}^K e^{z_k}}$$

从公式中可以看出，所有的 z_j 都对 y_i 的值有影响，可以分为两类： $j = i$ 和 $j \neq i$ 。

$j \neq i$ 时，不同的 j 只是下标不同。



Softmax 的偏导数

$j = i$

$$\frac{\partial y_i}{\partial z_j} = \frac{\partial}{\partial z_j} \left(\frac{e^{z_i}}{\sum_{k=1}^K e^{z_k}} \right) = \frac{(e^{z_i})' \sum_{k=1}^K e^{z_k} - e^{z_i} (\sum_{k=1}^K e^{z_k})'}{(\sum_{k=1}^K e^{z_k})^2}$$

$$\frac{\partial y_i}{\partial z_j} = \frac{e^{z_i} \sum_{k=1}^K e^{z_k} - e^{z_i} e^{z_j}}{(\sum_{k=1}^K e^{z_k})^2} = \frac{e^{z_i} (\sum_{k=1}^K e^{z_k} - e^{z_j})}{(\sum_{k=1}^K e^{z_k})^2}$$

$$\frac{\partial y_i}{\partial z_j} = \frac{e^{z_i}}{\sum_{k=1}^K e^{z_k}} \times \frac{\sum_{k=1}^K e^{z_k} - e^{z_j}}{\sum_{k=1}^K e^{z_k}} = y_i (1 - y_j)$$



Softmax 的偏导数

$j \neq i$

$$\frac{\partial y_i}{\partial z_j} = \frac{\partial}{\partial z_j} \left(\frac{e^{z_i}}{\sum_{k=1}^K e^{z_k}} \right) = \frac{(e^{z_i})' \sum_{k=1}^K e^{z_k} - e^{z_i} (\sum_{k=1}^K e^{z_k})'}{(\sum_{k=1}^K e^{z_k})^2}$$

$$\frac{\partial y_i}{\partial z_j} = \frac{0 - e^{z_i} e^{z_j}}{(\sum_{k=1}^K e^{z_k})^2} = \frac{-e^{z_i}}{\sum_{k=1}^K e^{z_k}} \times \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}}$$

$$\frac{\partial y_i}{\partial z_j} = -y_i y_j$$

最终的偏导数：

$$\frac{\partial y_i}{\partial z_j} = \begin{cases} y_i(1 - y_j) & j = i \\ -y_i y_j & j \neq i \end{cases}$$

每次计算 softmax 偏导值，只需做一个减法和一个乘法



Cross Entropy

假设第 i 类为正确类别， y_i 表示样本分类正确的概率。

$$y_i = \text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_{k=1}^K e^{z_k}}$$

我们希望 y_i 越大越好，因此可以借助损失函数来进行优化：

$$l_i = -y_i$$

此时损失函数越小越好。为了加快梯度下降，引入 \log 操作，不影响函数的单调性，且取值范围扩大到了 $(0, \infty)$ ：

$$l_i = -\log y_i = -\log \frac{e^{z_i}}{\sum_{k=1}^K e^{z_k}}$$



Cross Entropy

我们希望的损失函数：

$$l_i = -\log y_i = -\log \frac{e^{z_i}}{\sum_{k=1}^K e^{z_k}}$$

实际上，Cross Entropy 就起到这个作用。

$$\mathcal{L} = - \sum_{i=1}^K \bar{y}_i \log(y_i)$$

其中 \bar{y} 是一个 one-hot 向量，表示真实的标签。

$\bar{y}_i = 0$ ，对应的 loss 值为 0，而 $\bar{y}_i = 1$ ，即上述损失函数。



Softmax 与 CrossEntropy 的联动

回到损失函数本身

$$l_i = -\log y_i = -\log \frac{e_i^z}{\sum_{k=1}^K e^{z_k}} = -z_i + \log \sum_{k=1}^K e^{z_k}$$

因此，当 Softmax 和 Cross Entropy loss 同时使用时，借助 logSoftmax 可以省去一个指数计算和一次除法，在数值上相对稳定一些。



类别不平衡分类

类别不平衡

许多现实世界的分类问题都具有类别不平衡问题：
某些类别比其他类别更普遍

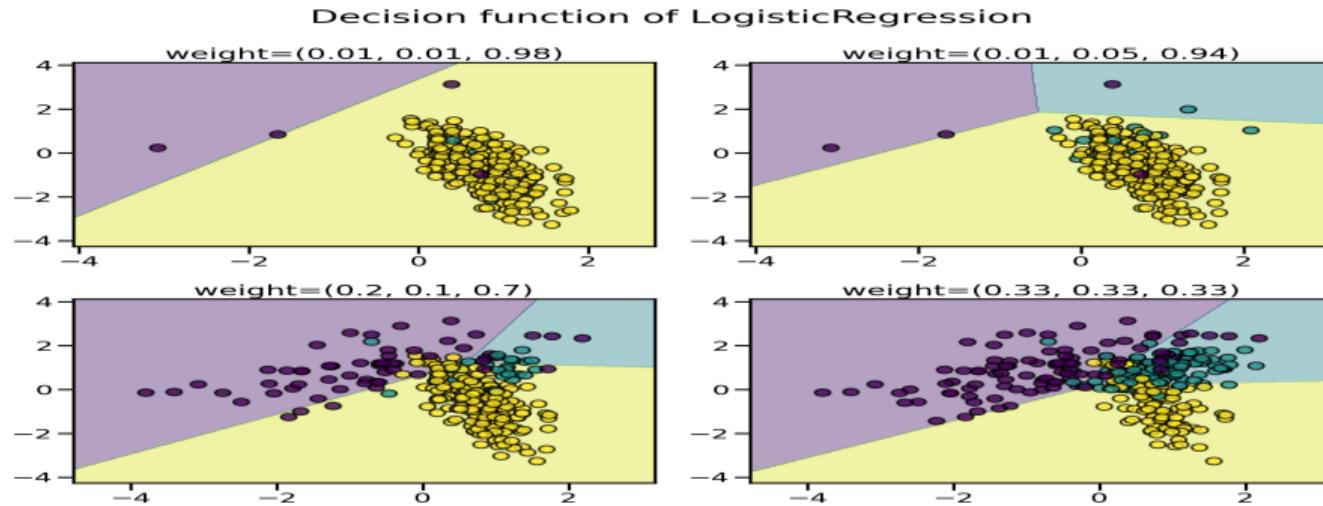
- ① 新冠病毒感染：所有发热患者中，只有 10% 可能患有新冠病毒
- ② 欺诈检测：在所有信用卡交易中，欺诈可能占交易的 0.2%
- ③ 制造缺陷分类：不同类型的制造缺陷可能有不同的发生率
- ④ 自动驾驶：根据驾驶场景，不同类型的物体出现概率不同
(高速路场景轿车、卡车较多、街道场景行人较多)



类别不平衡分类

类别不平衡的影响

类别不平衡问题严重影响机器学习模型的泛化性



类别不平衡的不良影响

类别不平衡对评价指标的影响

- 对于不平衡的数据，准确性 (ACC) 等标准指标可能没有意义。例如，始终预测“非欺诈”的分类器在检测信用卡欺诈方面的准确率可达 99.8%，然而欺诈样本一个都没有被准确分类。

法国士兵：
我们的马奇诺防线
有效率高达 99%



德国士兵：
阿登森林的 1%
并不设防



类别不平衡的不良影响

类别不平衡的潜在社会风险

- 加纳裔科学家 Joy Buolamwini 一次偶然发现，IBM、微软和旷视 Face++ 三家的人脸识别产品，均存在不同程度的女性和深色人种“歧视”（即女性和深色人种的识别正确率均显著低于男性和浅色人种），最大差距可达 34.3%。
- 其根本原因在于模型训练集中深色人种样本数量少



类别不平衡-评价指标侧解决方案

评价指标：F-Beta score

$$F_{\beta} = (1 + \beta^2) \frac{\text{precision} * \text{recall}}{\beta^2 \text{precision} + \text{recall}} \quad (4)$$

选择评估指标没有一刀切的解决方案，选择应该取决于问题。例如，信用卡欺诈检测的评估指标可以是精确度和召回率分数（F-beta 分数）的加权平均值，权重是通过权衡未能阻止欺诈交易和错误阻止欺诈交易的相对成本来确定的。



类别不平衡-数据侧解决方案

采样

- 对多数类进行降采样

平衡数据集的另一种方法是从多数类中删除数据样本。在某些情况下，当数据集高度不平衡时，可能会导致丢弃大量数据，从而导致性能不佳。

- 对少数类进行上采样

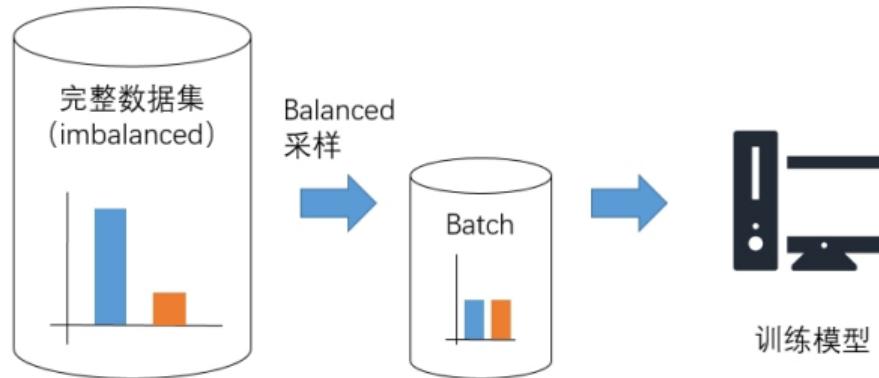
SMOTE (合成少数过采样技术)。通过组合或扰动少数样本来创建少数类别的新样本，从而增加少数类的占比。



类别不平衡-模型侧解决方案

Balanced mini-batch training

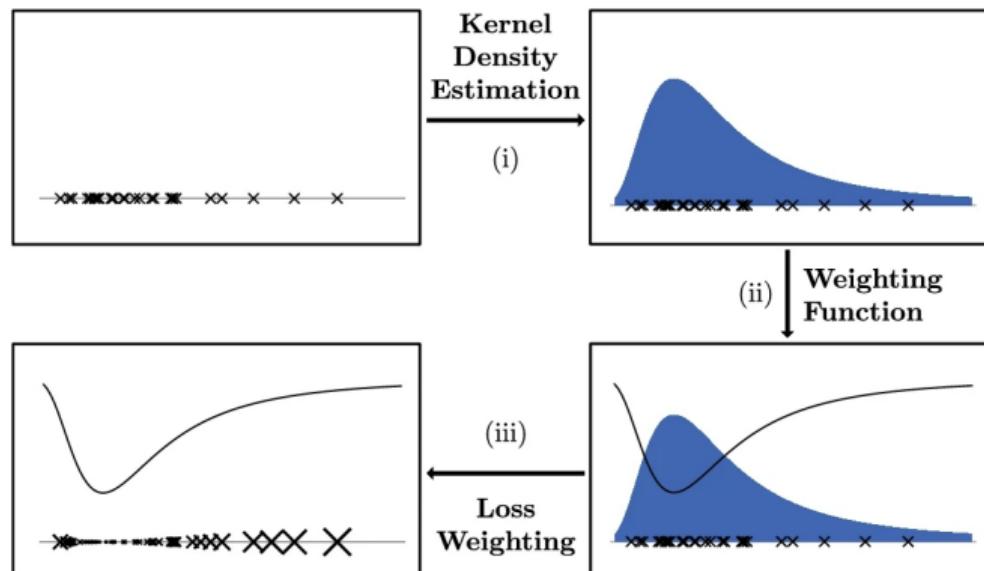
对于使用小 batch 训练模型，在划分每个小 batch 的随机数据子集时，可以包含更多的少数类的数据点，从而使小 batch 中的数据保持平衡。这种方法类似于过采样，并且不会丢弃数据。



类别不平衡-模型侧解决方案

Sample Re-weighting

在训练过程中对少数类（密度低）样本赋予更高的权重。通过优化损失的加权平均值使得模型更加重视某些数据点。



类别不平衡-模型侧解决方案

当样本分布失衡时，在损失函数 L 的分布也会发生倾斜，模型训练过程中会倾向于样本多的类别，造成模型对少样本类别的性能较差。

Balanced Cross entropy

一个直观的做法就是在损失函数中添加权重因子，提高少数类别在损失函数中的权重，平衡损失函数的分布。

$$\text{CELoss} = L(y, \hat{p}) = -y \log(\hat{p}) - (1 - y) \log(1 - \hat{p})$$

$$L_{BCE}(y, \hat{p}) = -(\beta * y \log(\hat{p}) + (1 - \beta) * (1 - y) \log(1 - \hat{p}))$$

$\frac{\beta}{1-\beta}$ = 负样本 / 正样本，正样本指本类样本数量，负样本指其他类样本数量。



类别不平衡-模型侧解决方案

Focal Loss 的具体形式为：

$$L_{fl} = \begin{cases} -(1 - \hat{p})^\gamma \log(\hat{p}) & \text{if } y = 1 \\ -\hat{p}^\gamma \log(1 - \hat{p}) & \text{if } y = 0 \end{cases}$$

令 $p_t = \begin{cases} \hat{p} & \text{if } y = 1 \\ 1 - \hat{p} & \text{otherwise} \end{cases}$

将 focal loss 表达式统一为一个表达式：

$$L_{fl} = -(1 - p_t)^\gamma \log(p_t)$$

同理可将交叉熵表达式统一为一个表达式：

$$L_{ce} = -\log(p_t) \quad (4)$$

类别不平衡-模型侧解决方案

Focal loss

focal loss 相比交叉熵多了一个 modulating factor 即 $(1 - p_t)^\gamma$ 。对于分类准确的样本 $p_t \rightarrow 1$ ， $(1 - p_t)^\gamma$ 趋近于 0。对于分类不准确的样本 $1 - p_t \rightarrow 1$ ， $(1 - p_t)^\gamma$ 趋近于 1。即相比交叉熵损失，focal loss 对于分类不准确的样本，损失没有改变，对于分类准确的样本，损失会变小。整体而言，相当于增加了分类不准确样本在损失函数中的权重。



背景

标签噪声：现实数据的常见问题

- 标签噪声是指训练数据中存在错误标签的现象，这些错误标签源自于人工标注时不可避免的失误或对抗样本攻击。
- 据统计，源自于真实世界的数据集中有 8.0% – 38.5% 的错误标签。

Images labelled as cat



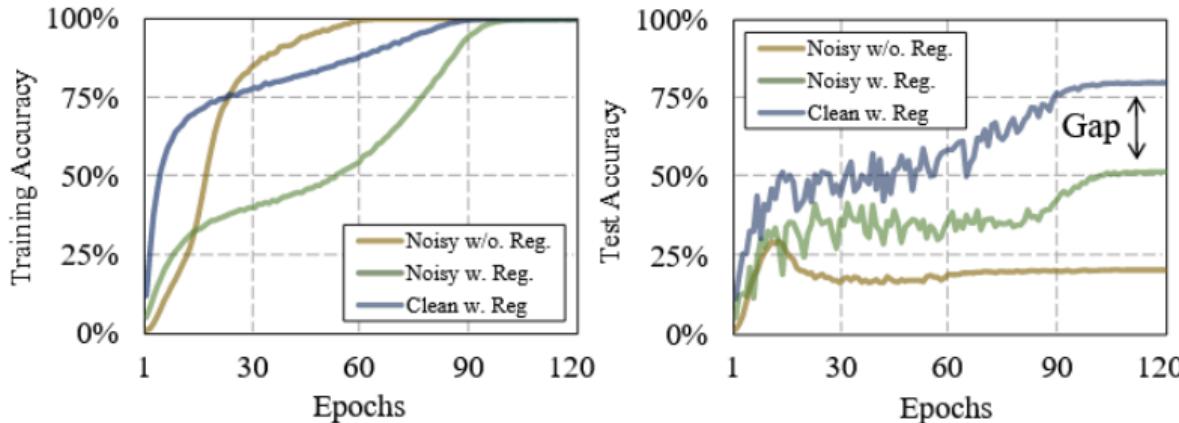
Images labelled as dog



背景

标签噪声对模型的影响

- 标签噪声的危害主要体现在降低模型的泛化性能。
- DNN 可以轻易地过拟合任何程度的噪声标签，这种过拟合使得模型在未知数据上给出错误的答案。
- 如下图，标签噪声不会导致训练精度的变化，但会极大降低模型的测试精度。



背景

定义

- 噪声标签: \tilde{y} ; 真实标签: y^* ; 类别数量: c
- 噪声率: ϵ ; 样本特征: x

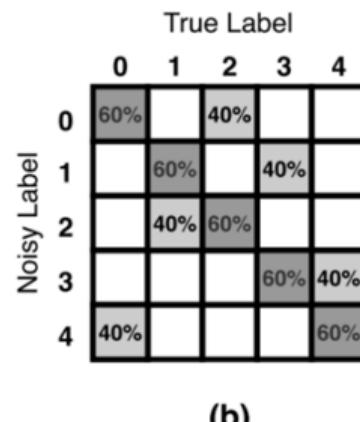
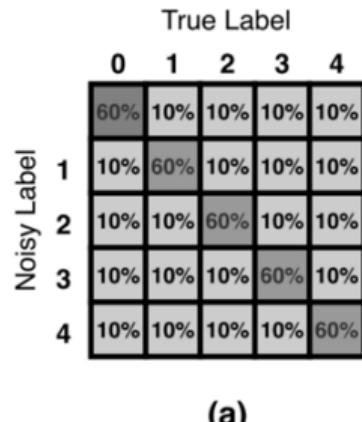
标签噪声的种类

- 实例无关噪声 (Instance independent noise): 假设标签的损坏过程与样本特征无关, 而仅仅与样本的真实标签有关。通常可以用一个标签转移矩阵 $T_{ij} = P(\tilde{y} = j | y^* = i)$ 表示, 其中 $P(\tilde{y} = j | y^* = i)$ 为真实标签为 i 时实际观测标签为 j 的概率。
- 实例相关噪声 (Instance dependent noise): 假设标签转移概率同样与样本特征有关, 用 $P(\tilde{y} = j | y^* = i, x)$ 表示。这种假设更贴近现实, 但也更复杂。
- 为便于研究, 通常假设标签噪声是实例无关的。

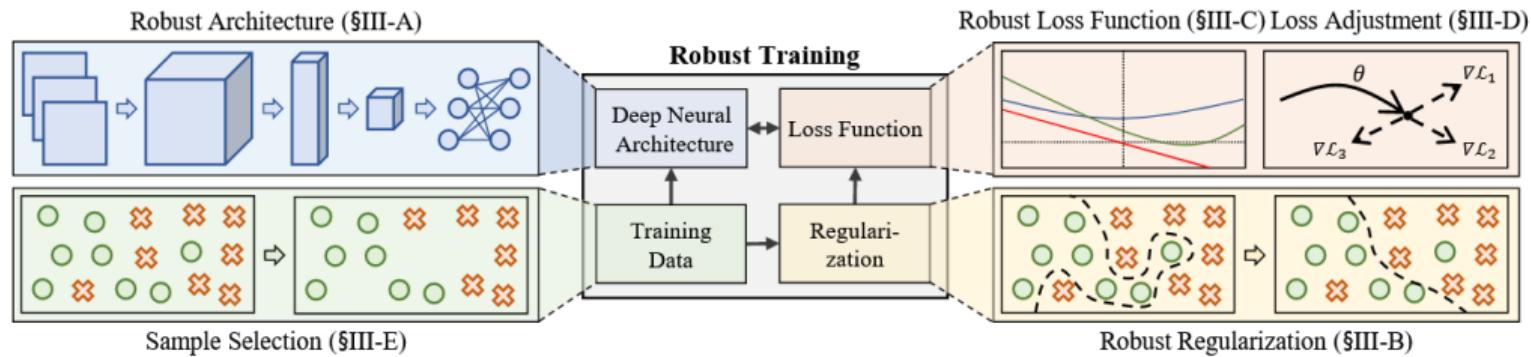
背景

两种与实例无关的标签噪声

- 对称噪声 (Symmetric Noise): 一个标签被翻转到其它任意一个类别的概率相同, 即:
 $\forall_{j \neq i} T_{ij} = P(\tilde{y} = j | y^* = i) = \frac{\epsilon}{c-1}$ 。
- 非对称噪声 (Asymmetric Noise): 真实标签更有可能被错误地归类到某个特定的标签中, 即: $\exists_{i \neq j, i \neq k, j \neq k} T_{ij} > T_{ik}$



标签噪声下的学习策略



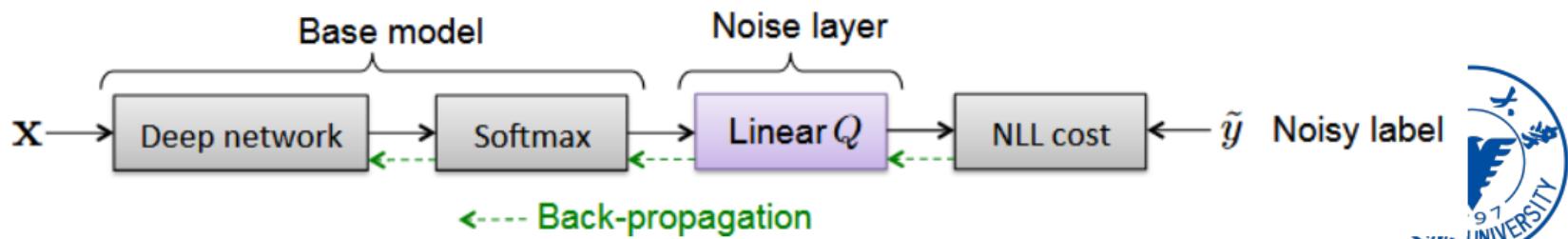
四类方法

- **Robust Architecture**: 旨在改进模型结构，使之适用于标签噪声。
- **Robust Regularization**: 约束学习过程，减少对噪声标签过拟合。
- **Robust Loss Function**: 设计鲁棒的损失函数，使得模型在噪声数据下有接近干净数据的表现。
- **Sample Selection**: 从噪声标签中寻找正确的标签用于训练。

标签噪声下的学习策略

Robust Architecture

- 一种典型的方案是在模型的输出层之后再加一个噪声适应层（Noise Adaptation Layer）。
- 噪声适应层的作用类似于近视镜片，学习过程可以表示为 $p(\tilde{y} = j | \mathbf{x}, \theta) = \sum_i p(\tilde{y} = j | y^* = i)p(y^* = i | \mathbf{x}, \theta)$ 。
- $p(\tilde{y} = j | y^* = i)$ 将噪声标签的分布复原回真实标签的分布。将复原后的标签用于训练，使得模型能够学习到正确的分布 $p(y^* = i | \mathbf{x}, \theta)$ 。



标签噪声下的学习策略

Robust Regularization

- 标签平滑是一种典型约束策略。
- 假设标签的先验服从分布 $P(Y)$, 标签平滑策略通常可以表示为在 NLL 损失函数上增加一个约束项:

$$\mathcal{L} = - \sum \log P_{\theta}(Y|X) - \mathbb{D}_{KL}(P(Y)||P_{\theta}(Y|X)) \quad (5)$$

- 通常假设 $P(Y)$ 服从均匀分布, 最小化上述损失函数使得模型的输出接近均匀分布。
- 从而防止模型将全概率分配给噪声标签, 减少对噪声标签过拟合。



标签噪声下的学习策略

Robust Loss Function

- 此类方法期望设计一种鲁棒的损失函数，使得模型在噪声标签和干净标签下训练具有相近的表现。
- 通常在风险最小化（Risk Minimization）的框架下设计鲁棒的损失函数。对于任意的分类器 f ，其在无噪声数据下的期望风险为：

$$R_L(f) = E_{\mathcal{D}}(L(f(\mathbf{x}), y_{\mathbf{x}})) \quad (6)$$

假设 f^* 为上述 $R_L(f)$ 的 global minimizer。当数据中存在标签噪声时，对于任意的分类器 f ，其在噪声数据下的期望风险为：

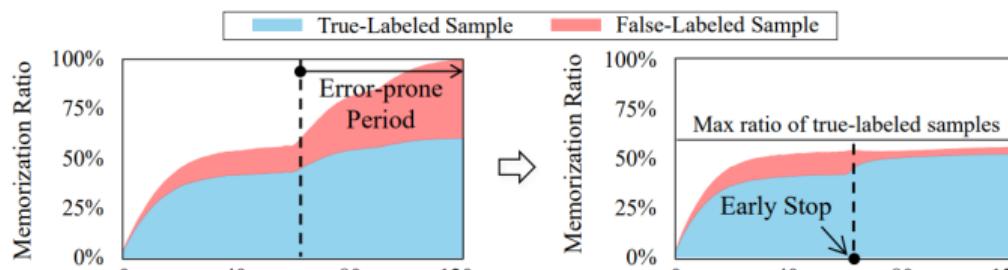
$$R_L^\eta(f) = E_{\mathcal{D}_\eta}(L(f(\mathbf{x}), \tilde{y}_{\mathbf{x}})) \quad (7)$$

假设 f_η^* 为上述 $R_L^\eta(f)$ 的 global minimizer。一个鲁棒的损失函数 L 满足 $R_L^\eta(f^*) - R_L^\eta(f) \leq 0$ ，满足该条件则说明在噪声数据上优化 $R_L^\eta(f)$ 可以使得 f 接近于

标签噪声下的学习策略

Sample Selection

- 这类方法通常选取一部分更可能有干净标签的样本，只用这些样本进行训练，从而避免学习错误的标签。
- 实证研究表明，DNN 倾向于先学习服从简单和广义模式的样本，这部分样本通常具有较小的损失。
- 在对这部分样本充分学习后，DNN 则开始学习难以学习的样本，其中就包括了错误标注样本，通常具有较大的损失。
- 因此，Sample Selection 方法通常选择具有较小损失的样本（Small-loss instances）学习。



总结

① 支持向量机 SVM

- 函数间隔与几何间隔
- 支持向量机优化

② 贝叶斯分类器

- 贝叶斯决策论
- 贝叶斯分类器
- 拉普拉斯修正

③ 集成学习

- 集成学习概述
- Bagging, Stacking
- Boosting

④ 深度学习中的分类

- 损失函数
- 类别不平衡分类
- 带噪声标签的分类



参考文献

-  WU, X., KUMAR, V., ROSS QUINLAN, J., GHOSH, J., YANG, Q., MOTODA, H., McLACHLAN, G. J., NG, A., LIU, B., YU, P. S., ET AL.
Top 10 algorithms in data mining.
Knowledge and information systems 14, 1 (2008), 1–37.

