机器学习

Machine learning

第四章 非线性分类

Nonlinear Classifier

授课人：周晓飞
zhouxiaofei@iie.ac.cn
2020-11-5

第四章 非线性分类

4.1 概述

4.2 决策树

4.3 最近邻方法

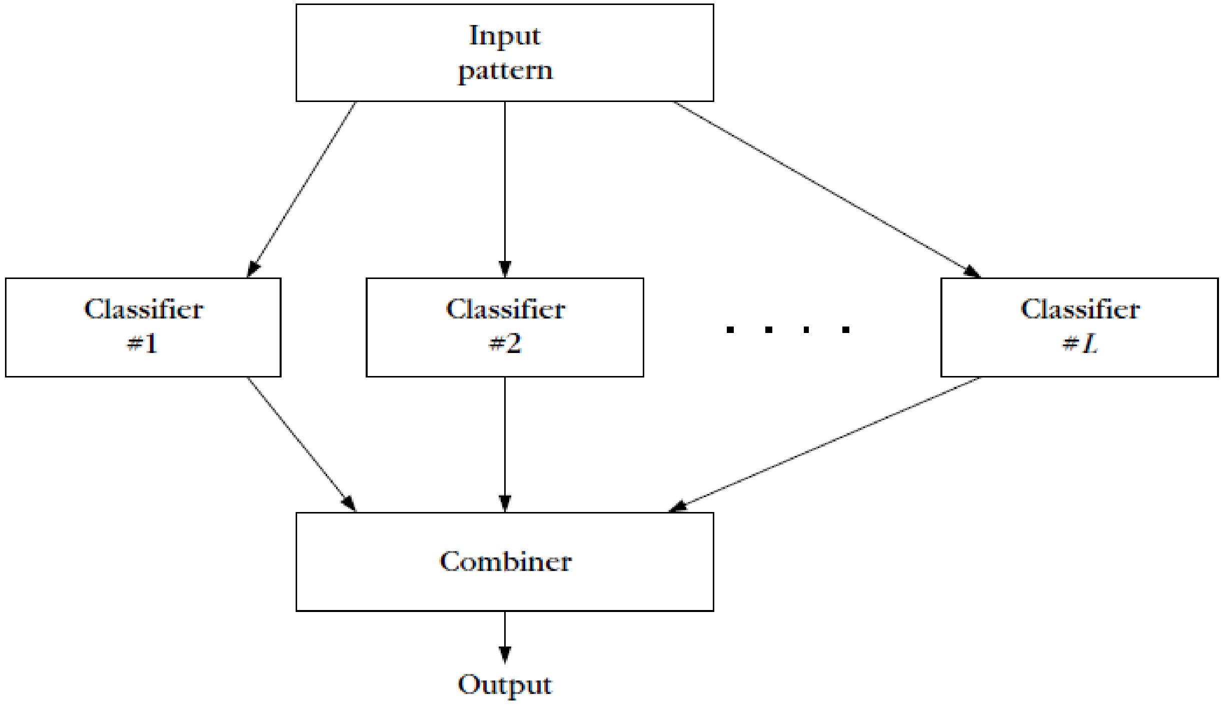
4.4 集成学习

4.5 非线性 SVM

集成学习

结合策略

原理：不同的分类器对样本有不同的鉴别力；综合优势，使错误率最小。



集成学习

结合策略

问题描述

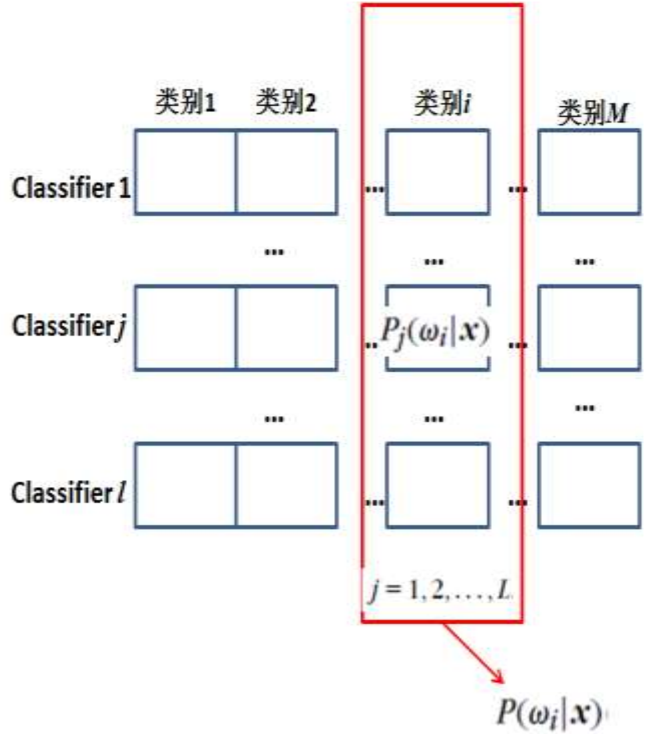
- 已知：一组训练分类器,分类器的类别后验为

$$P_j(\omega_i|\boldsymbol{x}),\quad i=1,2,\ldots,M,\quad j=1,2,\ldots,L.$$

(其中，i 索引类别，j 索引分类器.)

- 目标：对 x 进行分类，求

$$P(\omega_i|\boldsymbol{x}) \qquad i=1,2,\ldots,M$$



结合： $P(\omega_1|\boldsymbol{x})\ldots$ $P(\omega_i|\boldsymbol{x})\ldots$ $P(\omega_l|\boldsymbol{x})$

集成学习

结合策略

几种集成学习准则

- Geometric Average Rule
- Arithmetic Average Rule
- Majority Voting Rule

集成学习

Geometric Average Rule

目标函数：最小化 KL 距离平均

集成方法：

$$P(\omega_i|x) = \prod_{j=1}^{L}(P_j(\omega_i|x))$$

决策规则：

$$\max_{\omega_i} \prod_{j=1}^{L} P_j(\omega_i|x)$$

- 每个分类器的类别后验

$$P_j(\omega_i|x),\ i=1,2,\ldots,M,\ j=1,2,\ldots,L$$

- 目标组合分类器类别后验：$P(\omega_i|x)$
- 每个分类器的 KL 距离

$$D_j = \sum_{i=1}^{M} P(\omega_i|x)\ln\frac{P(\omega_i|x)}{P_j(\omega_i|x)}, \qquad \sum_{i=1}^{M} P_j(\omega_i|x) = 1$$

- 平均 KL 距离

$$D_{av} = \frac{1}{L}\sum_{j=1}^{L} D_j = \frac{1}{L}\sum_{j=1}^{L}\sum_{i=1}^{M} P(\omega_i|x)\ln\frac{P(\omega_i|x)}{P_j(\omega_i|x)}$$

- 最小化 D_{av}

$$D_{av} = \frac{1}{L}\sum_{j=1}^{L}\sum_{i=1}^{M} P(\omega_i|x)\ln\frac{P(\omega_i|x)}{P_j(\omega_i|x)}, \qquad \sum_{i=1}^{M} P_j(\omega_i|x) = 1$$

- Langrange Multipliers:

$$P(\omega_i|x) = \frac{1}{C}\prod_{j=1}^{L}(P_j(\omega_i|x))^{\frac{1}{L}}, \qquad C = \sum_{i=1}^{M}\prod_{j=1}^{L}(P_j(\omega_i|x))^{\frac{1}{L}}$$

集成学习

Arithmetic Average Rule

目标函数：最小化 Alternative KL 距离平均

集成方法：

$$P(\omega_i|x) = \sum_{j=1}^{L} P_j(\omega_i|x)$$

决策规则：

$$\max_{\omega_i} \sum_{j=1}^{L} P_j(\omega_i|x)$$

- 每个分类器的 Alternative KL 距离

$$D_j = \sum_{i=1}^{M} P_j(\omega_i|x) \ln \frac{P_j(\omega_i|x)}{P(\omega_i|x)}, \quad \sum_{i=1}^{M} P_j(\omega_i|x) = 1$$

- 平均 KL 距离

$$D_{av} = \frac{1}{L} \sum_{j=1}^{L} D_j$$

- 最小化 Dav

$$D_{av} = \frac{1}{L} \sum_{j=1}^{L} \sum_{i=1}^{M} P_j(\omega_i|x) \ln \frac{P_j(\omega_i|x)}{P(\omega_i|x)}, \quad \sum_{i=1}^{M} P_j(\omega_i|x) = 1$$

- Langrange Multipliers:

$$P(\omega_i|x) = \frac{1}{L} \sum_{j=1}^{L} P_j(\omega_i|x)$$

集成学习

Majority Voting Rule

原理: 对两类问题，多个分类器进行决策投票，票数过半的类别为样本最终标签.

$$l_c = \begin{cases} \frac{L}{2} + 1, & L \text{ even} \\ \frac{L+1}{2}, & L \text{ odd} \end{cases}$$

什么样的基分类器组合比较好：

(1)多样性　(2)性能不太差

Majority Voting Rule

讨论：分类器彼此独立，并有相同的正确识别概率 p。L 个分类器的联合，正确识别概率 ？

$$P_c(L) = \sum_{m=l_c}^{L} \binom{L}{m} p^m (1-p)^{L-m}$$

$$\boxed{\text{If } p > 0.5, P_c(L) \text{ is monotonically increasing in } L \text{ and } P_c(L) \to 1 \text{ as } L \to \infty}.$$

If $p < 0.5$, $P_c(L)$ is monotonically decreasing in L and $P_c(L) \to 0$ as $L \to \infty$.

If $p = 0.5$, $P_c(L) = 0.5$ for all L.

要求基分类器：(1)相互独立， (2) 正确率 p>50%

集成学习

Bagging 和随机森林

1. **Bagging**：通过随机采样，训练分类器，保证分类器的差异。
   - 从训练集中不断随机抽取样本构造分类器，分类时通过投票进行类别判断。

```
For t = 1, 2, …, T Do
        从数据集 S 中取样（放回选样）；
        训练得到模型 Ht;
 End For
未知样本 X，每个模型 Ht 都得出一个分类，得票最高的即为 X 的分类。
```

   - 算法复杂度：基分类器复杂度 ＋ 投票复杂度：

集成学习

Bagging 和随机森林

2. 随机森林： 多决策树的 Bagging；决策树随机属性选择；

- 从训练集中不断随机构造决策树分类器，分类时通过投票进行类别判断。

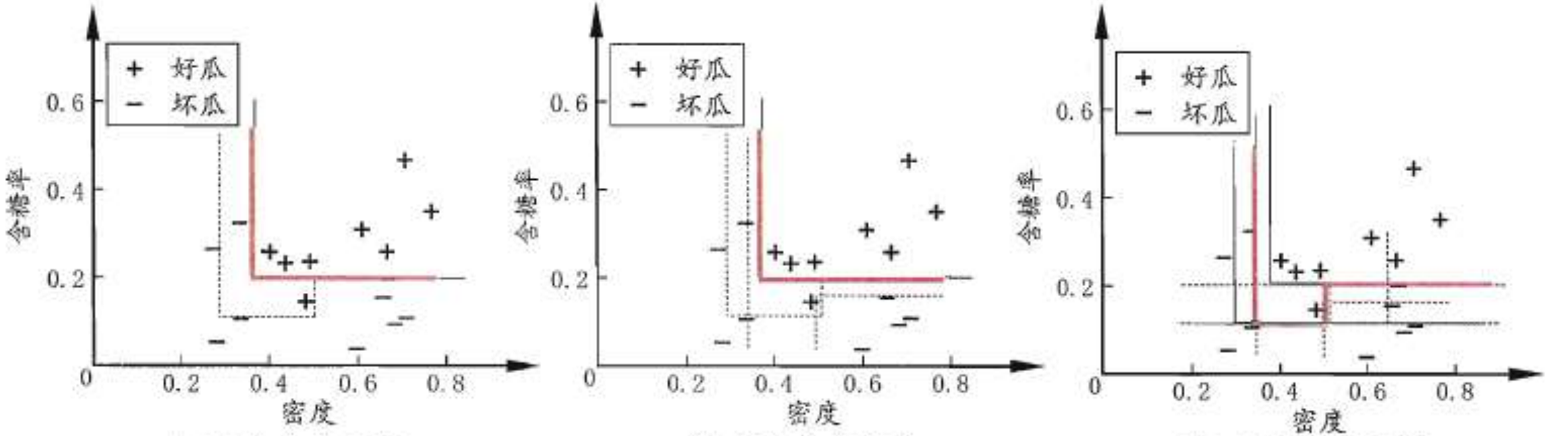
```
For t = 1, 2, ···, T Do
    从数据集 S 中取样（放回选样）；
    决策树模型 Ht:
        For each node  通过随机选择 k 个属性子集，构建决策节点
End For
未知样本 X,每个模型 Ht 都得出一个分类，得票最高的即为 X 的分类。
```

集成学习

Bagging 和随机森林

例子：

密度	含糖率	好瓜
0.697	0.460	是
0.774	0.376	是
0.634	0.264	是
0.608	0.318	是
0.556	0.215	是
0.403	0.237	是
0.481	0.149	是
0.437	0.211	是
0.666	0.091	否
0.243	0.267	否
0.245	0.057	否
0.343	0.099	否
0.639	0.161	否
0.657	0.198	否
0.360	0.370	否
0.593	0.042	否
0.719	0.103	否



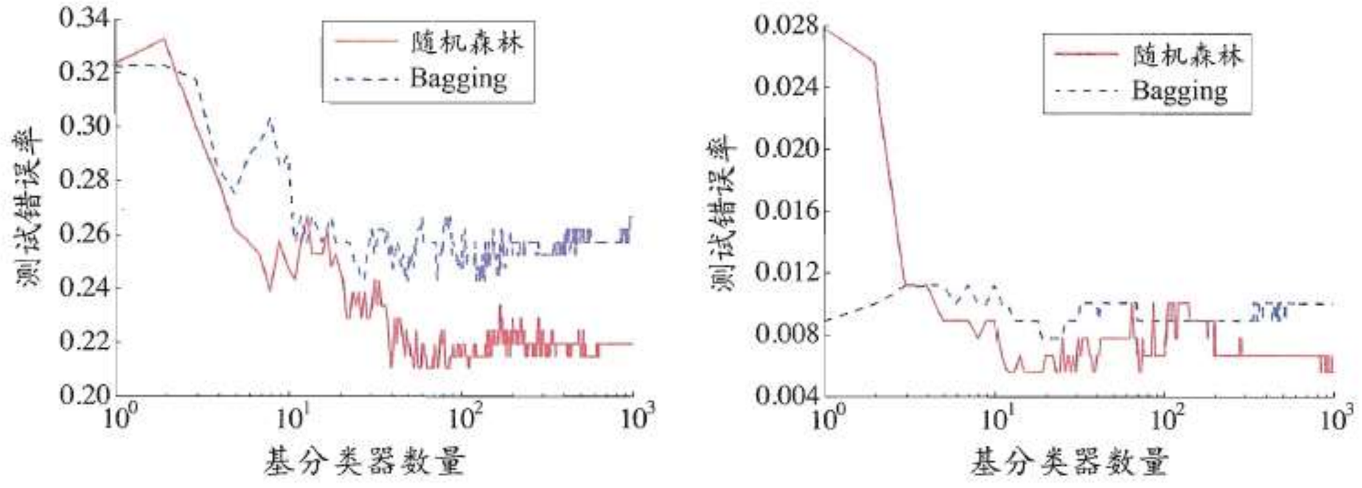
(a) 3个基学习器 (b) 5个基学习器 (b) 11个基学习器

问题数：17*2=34

每棵树的每节点候选问题是 34 个问题的随机子集。

Bagging 和随机森林

随机森林较一般 Bagging 效果好



(a) glass 数据集 (b) auto-mpg 数据集

集成学习

Boosting: AdaBoost

Boosting 原理：一系列弱分类器，在不同子集上学习，得到增强分类器。

original work [Vali 84, Kear 94]：a “weak” learning algorithm can be boosted into a “strong” algorithm with good error performance?

[Schapire 98]: It turns out that given a sufficient number of iterations the classification error of the final combination measured on the training set can become arbitrarily low.

AdaBoost (Adaptive Boosting)算法:

Boosting 族算法最著名的代表是 AdaBoost [Freund and Schapire, 1997]

Boosting: AdaBoost

AdaBoost 加权分类器

$$F(\boldsymbol{x}) = \sum_{k=1}^{K} \alpha_k \phi(\boldsymbol{x}; \boldsymbol{\theta}_k)$$

参数向量

$$\phi(\boldsymbol{x}; \boldsymbol{\theta}_k) \in \{-1, 1\}$$

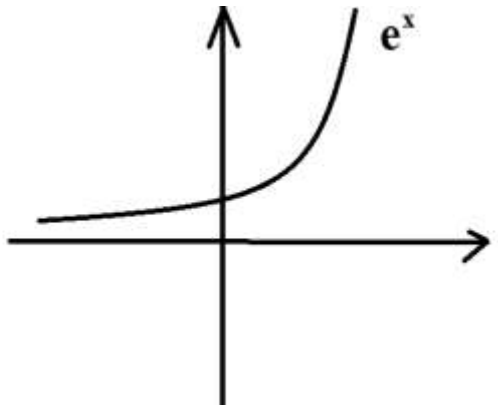
决策:

$$f(\boldsymbol{x}) = sign\{F(x)\}$$

Boosting: AdaBoost

AdaBoost 目标函数

$$\arg\min_{\alpha_k;\boldsymbol{\theta}_k,k:1...,K}\sum_{i=1}^{N}\exp(-y_i F(\boldsymbol{x}_i))$$



错误分类时， $y_i F(x_i) < 0$， $-y_i F(x_i) > 0$

正确分类时， $y_i F(x_i) > 0$， $-y_i F(x_i) < 0$

目标函数更强调修正错误样本

其中，$\{(x_1, y_1), (x_2, y\ )..., (x_N, y_N)\}\ with\ y \in \{-1, 1\}, i = 1, 2..., N$

集成学习

Boosting: AdaBoost

AdaBoost 求解推理过程

- 前 m 个分类器结果

$$F_m(\boldsymbol{x}) = \sum_{k=1}^{m} \alpha_k \phi(\boldsymbol{x}; \boldsymbol{\theta}_k), m = 1, 2, \ldots, K$$

- 增量表示法

$$F_m(\boldsymbol{x}) = F_{m-1}(\boldsymbol{x}) + \alpha_m \phi(\boldsymbol{x}; \boldsymbol{\theta}_m)$$

{m 个分类器组合结果} = {m-1 个结果} + {第 m}

Boosting: AdaBoost

AdaBoost 求解推理过程

- 前 m 个分类器的目标函数

$$J(\alpha,\boldsymbol{\theta})=\sum_{i=1}^{N}\exp(-y_i(F_{m-1}(\boldsymbol{x})+\alpha\phi(\boldsymbol{x};\boldsymbol{\theta})))$$

$$(\alpha_m,\boldsymbol{\theta}_m)=\arg\min_{\alpha,\boldsymbol{\theta}} J(\alpha,\boldsymbol{\theta})$$

- 转化目标函数形式

$$J(\alpha,\boldsymbol{\theta})=\sum_{i=1}^{N}w_i^{(m)}\exp(-y_i\alpha\phi(\boldsymbol{x};\boldsymbol{\theta}))$$

$$\boxed{w_i^{(m)}\equiv\exp(-y_iF_{m-1}(\boldsymbol{x}_i))}$$

定义样本权重：前 m-1 个分类器对 x_i 的错分指数，与第 m 个分类器的参数 $\alpha,\boldsymbol{\theta}$ 无关。

集成学习

Boosting: AdaBoost

（1）求 θ_m （确定选择的基分类器）

$$\theta_m = \arg\min_{\theta} \sum_{i=1}^{N} w_i^{(m)} \exp(-y_i \alpha \phi(x;\theta))$$

$$\theta_m = \arg\min_{\theta} \left\{ \underbrace{P_m}_{\text{第 m 个分类器错误识别样本的平均权重}} = \sum_{i=1}^{N} w_i^{(m)} \underbrace{I(1 - y_i \phi(x;\theta))}_{\text{正确分类为 0}} \right\}$$

Function $I(\cdot)$ is either 0 or 1

第 m 个分类器错误识别样本的平均权重　正确分类为 0

- 重新理解目标：最优化第 m 个分类器，使得错分样本所对应平均权重最小化，

即第 m 个分类器正确分类的样本，尽可能是前 m-1 个组合分类器错分指数高的样本（$w_i^{(m)}$大）。

求 θ_m 这一过程，通常在选择第 m 个分类器时实现，选择阈值 P_m<0.5。

Boosting: AdaBoost

(2) 求 α_m (分类器权重)

$$J(\alpha,\boldsymbol{\theta})=\sum_{i=1}^{N} w_i^{(m)}\exp(-y_i\alpha\phi(\boldsymbol{x};\boldsymbol{\theta}))$$

由于

$$\sum_{y_i\phi(\boldsymbol{x}_i;\boldsymbol{\theta}_m)<0} w_i^{(m)}=P_m\ , \qquad y_i\phi(\boldsymbol{x}_i;\boldsymbol{\theta}_m)=-1$$

$$\sum_{y_i\phi(\boldsymbol{x}_i;\boldsymbol{\theta}_m)>0} w_i^{(m)}=1-P_m\ , \quad y_i\phi(\boldsymbol{x}_i;\boldsymbol{\theta}_m)=1$$

含义：1-p 表示正确分类样本的比重。第 m 个分类器能够正确分类样本的“比重”决定该分类器的权重。

$$\alpha_m=\arg\min_{\alpha}\{\exp(-\alpha)(1-P_m)+\exp(\alpha)P_m\} \Rightarrow \boxed{\alpha_m=\frac{1}{2}\ln\frac{1-P_m}{P_m}}$$

集成学习

Boosting: AdaBoost

(3）样本权重更新

$$w_i^{(m+1)} \equiv \exp(-y_i F_m(\boldsymbol{x}_i)) = w_i^{(m)} \exp(-y_i \alpha_m \phi(\boldsymbol{x}_i; \boldsymbol{\theta}_m))$$

将 $\alpha_m = \frac{1}{2} \ln \frac{1-P_m}{P_m}$ 代入，

被第 m 个分类器正确分类的样本权值为：

$y_i \phi(\boldsymbol{x}_i; \boldsymbol{\theta}_m) = 1$， $w_i^{(m+1)} = w_i^{(m)} \exp(-\alpha_m) = w_i^{(m)} \sqrt{\frac{P_m}{1-P_m}}$

被第 m 个分类器错误分类的样本权值为：

$y_i \phi(\boldsymbol{x}_i; \boldsymbol{\theta}_m) = -1$， $w_i^{(m+1)} = w_i^{(m)} \exp(\alpha_m) = w_i^{(m)} \sqrt{\frac{1-P_m}{P_m}}$

归一化： $w_i^{(m+1)} = w_i^{(m+1)} \Big/ \sum_{i=1}^{N} w_i^{(m+1)}$

有的程序中，只修改了正确样本的权，或只修改错误样本权。

错分样本较正确分类样本放大的倍数（两者相除）：

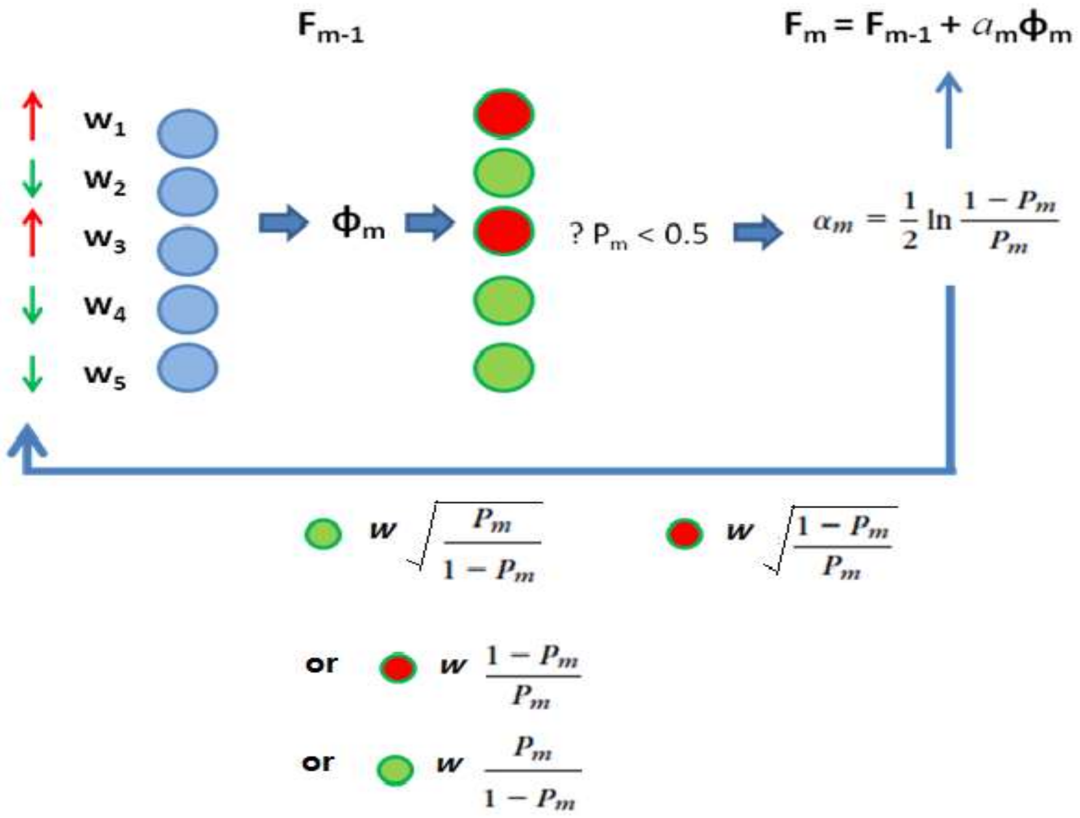
$$\exp(2\alpha_m) = \frac{1-P_m}{P_m}$$

所以，可以只修改错分样本权： $w_i^{(m+1)} = w_i^{(m)} \frac{1-P_m}{P_m}$

或者，之修正确分类样本权： $w_i^{(m+1)} = w_i^{(m)} \frac{P_m}{1-P_m}$

Boosting: AdaBoost

AdaBoost 流程：



集成学习

示例

样本权值更新：

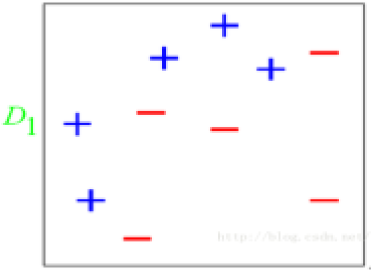
正确分类的样本权值不变

错误分类的样本权值为：

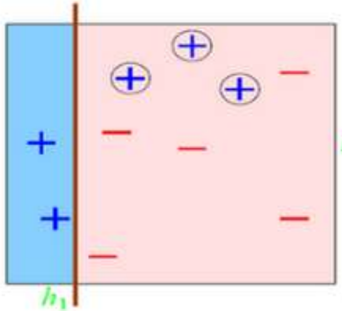
$$w_i^{(m+1)} = w_i^{(m)} \frac{1-P_m}{P_m}$$

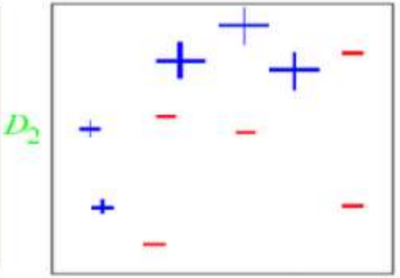
分类器权值：

$$\alpha_m = \frac{1}{2}\ln\frac{1-P_m}{P_m}$$



$all\ x_i，\ w_{\mathrm{i}} = 0.1$





$p_{m=}0.3, \quad 1\text{-}p_{m=}0.7$

$$\alpha_1 = \frac{1}{2}\ln\left(\frac{1-\varepsilon_1}{\varepsilon_1}\right) = \frac{1}{2}\ln\left(\frac{1-0.3}{0.3}\right) = 0.42$$

正确分类不变 **wi=0.1;**

错误分类 **wi=0.233;** 对 **wi** 归一化

集成学习

示例

样本权值更新：

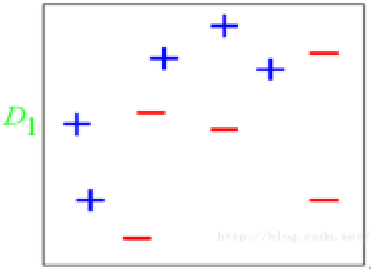
正确分类的样本权值不变

错误分类的样本权值为:

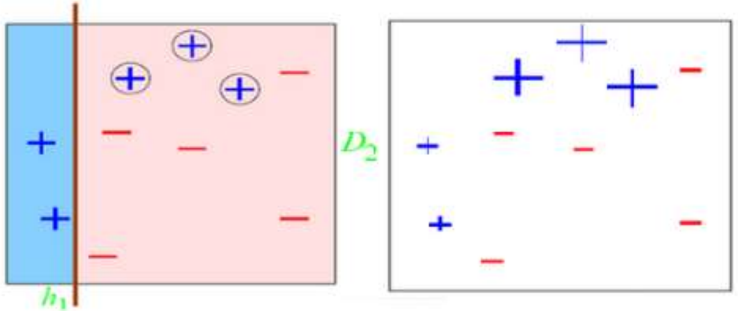
$$w_i^{(m+1)} = w_i^{(m)} \frac{1-P_m}{P_m}$$

分类器权值：

$$\alpha_m = \frac{1}{2} \ln \frac{1-P_m}{P_m}$$



all x_i， $w_i = 0.1$

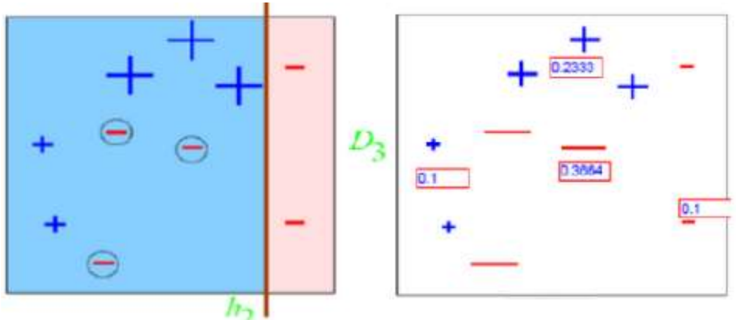


$p_{m=}0.3$, $1\text{-}p_{m=}0.7$

$$\alpha_1 = \frac{1}{2}\ln\left(\frac{1-\varepsilon_1}{\varepsilon_1}\right) = \frac{1}{2}\ln\left(\frac{1-0.3}{0.3}\right) = 0.42$$

正确分类不变 w_i=0.1;

错误分类 w_i=0.233; 对 w_i 归一化



p_{m}=3* (0.1/(0.1*7+0.233*3)) = 0.2144

$$\alpha_2 = \frac{1}{2}\ln\left(\frac{1-\varepsilon_2}{\varepsilon_2}\right) = \frac{1}{2}\ln\left(\frac{1-0.2144}{0.2144}\right) = 0.6493$$

错误分类 $0.1\left(\frac{1-0.2144}{0.2144}\right) = 0.3644$

对 wi 归一化

集成学习

示例

样本权值更新：

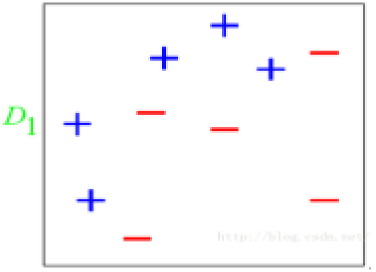
正确分类的样本权值不变

错误分类的样本权值为:

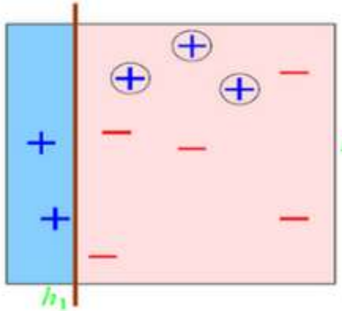
$$w_i^{(m+1)} = w_i^{(m)} \frac{1-P_m}{P_m}$$

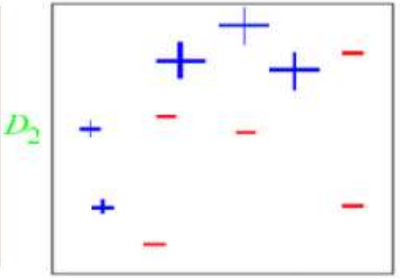
分类器权值：

$$\alpha_m = \frac{1}{2}\ln\frac{1-P_m}{P_m}$$



all x_i， $w_i = 0.1$



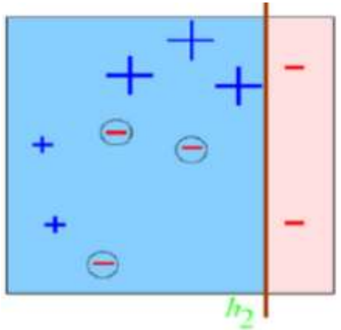


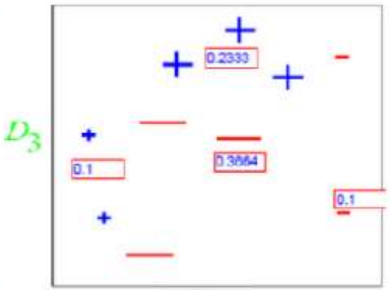
$p_{m=}0.3$, $1\text{-}p_{m=}0.7$

$$\alpha_1 = \frac{1}{2}\ln\left(\frac{1-\varepsilon_1}{\varepsilon_1}\right) = \frac{1}{2}\ln\left(\frac{1-0.3}{0.3}\right) = 0.42$$

正确分类不变 **w_i=0.1;**

错误分类 **w_i=0.233;** 对 w_i 归一化



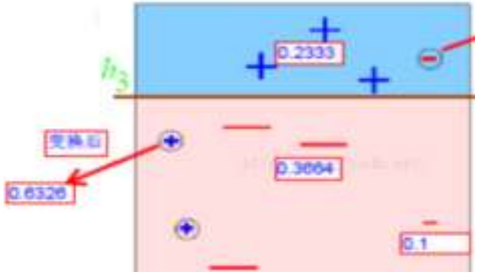


$$p_{m}=3*(0.1/(0.1*7+0.233*3)) = 0.2144$$

$$\alpha_2 = \frac{1}{2}\ln\left(\frac{1-\varepsilon_2}{\varepsilon_2}\right) = \frac{1}{2}\ln\left(\frac{1-0.2144}{0.2144}\right) = 0.6493$$

错误分类 $0.1\left(\frac{1-0.2144}{0.2144}\right) = 0.3644$

对 wi 归一化



$$p_{m}=3*(0.1/(0.1*4+0.233*3+0.3664*3))= 0.1365$$

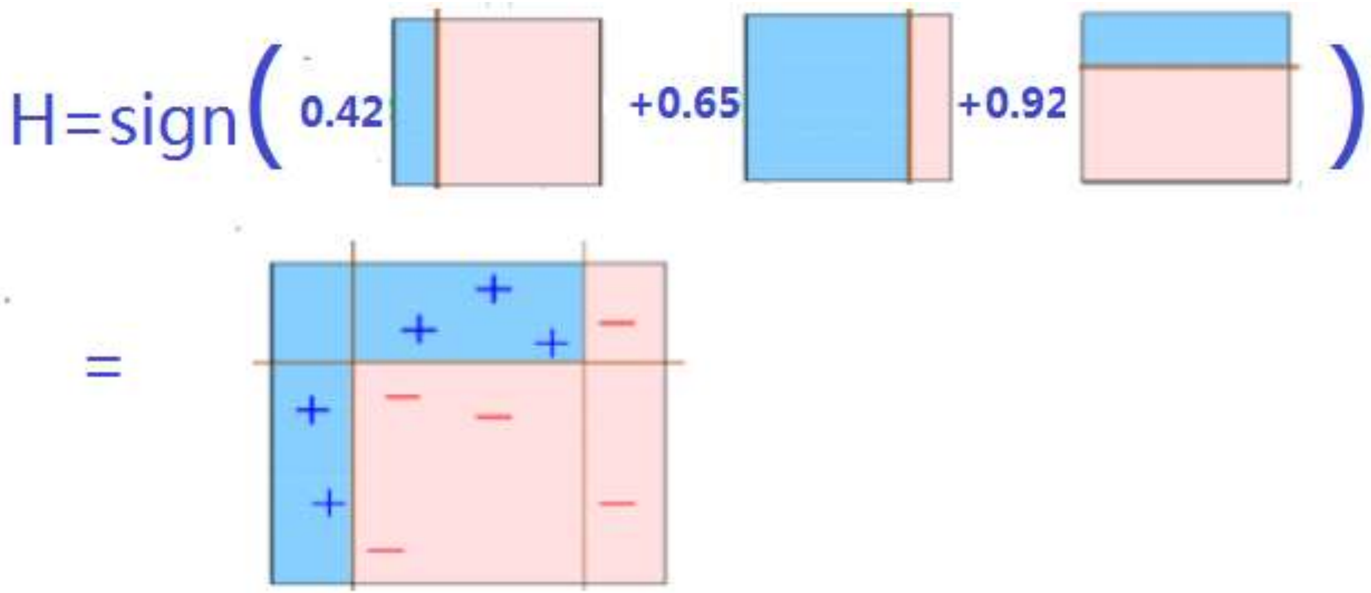
$$\alpha_3 = \frac{1}{2}\ln\left(\frac{1-\varepsilon_3}{\varepsilon_3}\right) = \frac{1}{2}\ln\left(\frac{1-0.1365}{0.1365}\right) = 0.9223$$

错误分类 $0.1\left(\frac{1-0.1365}{0.1365}\right) = 0.6326$

对 wi 归一化

集成学习

示例



第四章 非线性分类

SVM 原理

两个核心思想

- **最大间隔**

  找到最大间隔分类超平面；

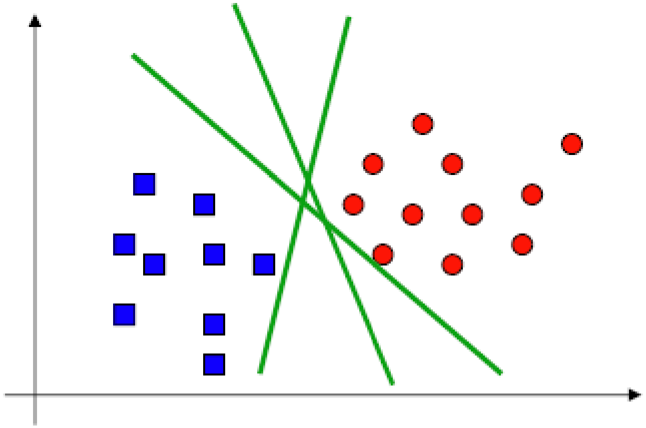
- **核函数方法**

  样本升维映射到高维空间后，采用线性决策。升维映射由核技巧实现。

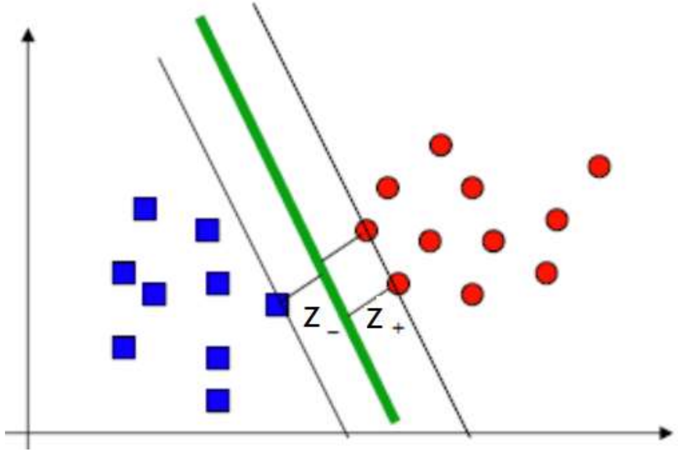
非线性 SVM

最大间隔

目标：找到最大间隔分类超平面



有多个超平面将数据分开，哪个更好?



类别集合到分类超平面的**最小距离最大化**

最大间隔

支撑向量（SV）：支撑最小距离最大化的样本

支撑超平面：通过支持向量，平行于分类面的超平面

间隔：支撑向量到分类面的距离

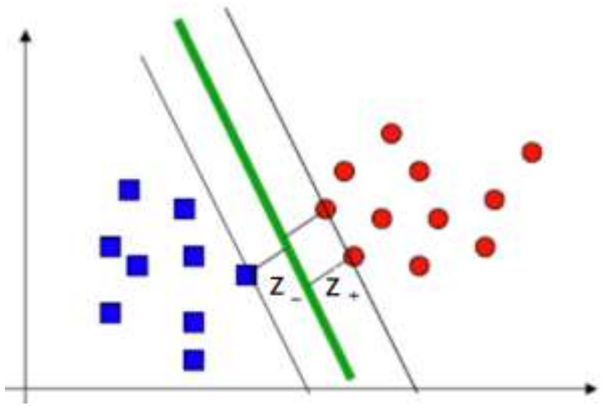
$$z=\frac{g(x)}{\|w\|},$$

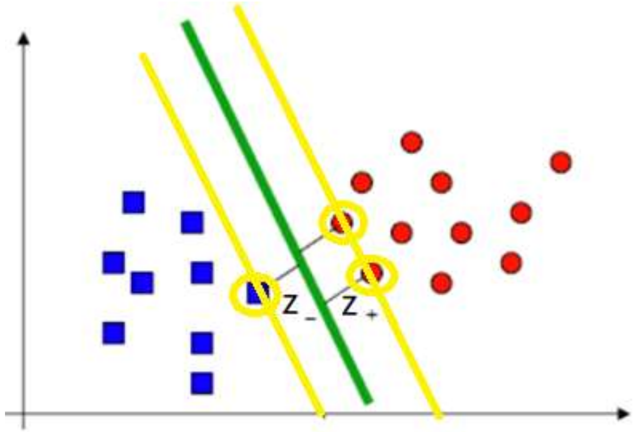
其中，$g(\mathbf{x})=\mathbf{w}^T\mathbf{x}+b$，

为找到最大间隔方向，且保证优化解唯一

令 $|g(\mathbf{x})|=c$ 取 $c=1$，则 $g(\mathbf{x})=\pm 1$

$$z_{\pm}=\frac{\pm 1}{\|w\|}$$





非线性 SVM

最大间隔

正确类别的样本：

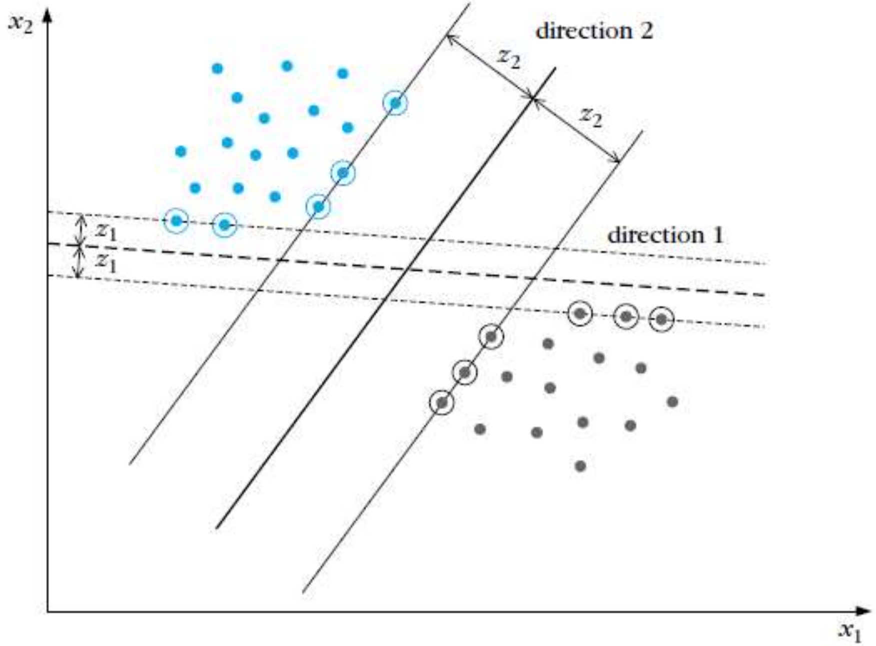
支撑超平面$\boldsymbol{w}^T\boldsymbol{x}+b=\pm1$，等价表达：$y(\boldsymbol{w}^T\boldsymbol{x}+b)-1=0$

正确类别的样本：$y_i(\boldsymbol{w}^T\boldsymbol{x}_i+b)-1\geq0$

最大间隔问题： $\max(z_+-z_-)=\max\limits_{w}\frac{2}{\|w\|}$

目标函数

$$\min_{w,b}\frac{1}{2}\|\boldsymbol{w}\|^2$$
$$s.t\quad y_i\left(\boldsymbol{w}^T\boldsymbol{x}_i+b\right)-1\geq0,\quad i=1,2,\ldots l \qquad (1)$$



最大间隔

优化问题

The dual problem

$$
\begin{aligned}
\min_{\alpha} \quad & \frac{1}{2}\sum_{i,j=1}^{n} \alpha_i\alpha_j y_i y_j x_i^\top x_j - \sum_{i=1}^{n}\alpha_i \\
s.t. \quad & \alpha_i \geq 0 \;\; i = 1\ldots n \\
& \sum_{i=1}^{n} \alpha_i y_i = 0
\end{aligned}
$$

非线性 SVM

最大间隔

对偶问题的推导：

目标函数的拉格朗日乘子法:

$$L(\omega,b,\alpha)=\frac{1}{2}\|\omega\|^2-\sum_{i=1}^{n}\alpha_i(y_i(\omega^T x_i+b)-1) \qquad (2)$$

令 $\theta_p(\omega,b)=\max_{\alpha_i\ge 0} L(\omega,b,\alpha)$ ， 原始问题:

$$\min_{\omega,b}\theta_p(\omega,b)=\min_{\omega,b}\max_{\alpha_i\ge 0} L(\omega,b,\alpha)=p^*$$

对偶问题: $\theta_d(\alpha)=\min_{\omega,b} L(\omega,b,\alpha)$

$$\max_{\alpha}\theta_d(\alpha)=\max_{\alpha_i\ge 0}\min_{\omega,b} L(\omega,b,\alpha)=\mathrm{d}^*$$

最大间隔

当满足 KKT 条件时两者等价：

注意式（2）已经不包括约束项了。

$$\frac{\partial L(\omega,b,\alpha)}{\partial\omega}=0$$

$$\alpha_i(y_i(\omega^T x_i+b)-1)=0,\ \ i=1,2\ldots N$$

$$y_i(\omega^T x_i+b)\geq 1,\ \ i=1,2\ldots N$$

$$\alpha_i\geq 0,\ \ i=1,2\ldots N$$

已经加到 Lagrange 函数中了，即改条件已经满足。

与此同时，上段说"在满足某些条件的情况下"，这所谓的"满足某些条件"就是要满足KKT条件。那KKT条件的表现形式是什么呢？据维基百科：KKT 条件的介绍，一般地，一个最优化数学模型能够表示成下列标准形式：

$$\min.\ f(\mathbf{x})$$

$$\text{s.t.}\quad h_j(\mathbf{x})=0, j=1,\ldots,p,$$

$$g_k(\mathbf{x})\leq 0, k=1,\ldots,q,$$

$$\mathbf{x}\in\mathbf{X}\subset\Re^n$$

其中，f(x)是需要最小化的函数，h(x)是等式约束，g(x)是不等式约束，p和q分别为等式约束和不等式约束的数量。同时，我们得明白以下两个定理：

- 凸优化的概念：$\mathcal{X}\subset\mathbb{R}^n$ 为一凸集，$f:\mathcal{X}\to\mathbb{R}$ 为一凸函数。凸优化就是要找出一点 $x^*\in\mathcal{X}$，使得每一 $x\in\mathcal{X}$ 满足 $f(x^*)\leq f(x)$。
- KKT条件的意义：它是一个非线性规划（Nonlinear Programming）问题能有最优化解法的必要和充分条件。

那到底什么是所谓Karush-Kuhn-Tucker条件呢？KKT条件就是指上面最优化数学模型的标准形式中的最小点 x* 必须满足下面的条件：

1. $h_j(\mathbf{x}_*)=0,\ j=1,\ldots,p,\ g_k(\mathbf{x}_*)\leq 0,\ k=1,\ldots,q,$
2. $\nabla f(\mathbf{x}_*)+\sum_{j=1}^{p}\lambda_j\nabla h_j(\mathbf{x}_*)+\sum_{k=1}^{q}\mu_k\nabla g_k(\mathbf{x}_*)=\mathbf{0},$

$\lambda_j\neq 0,\ \mu_k\geq 0,\ \mu_k g_k(\mathbf{x}_*)=0.$

非线性 SVM

最大间隔

对偶问题： $\max\limits_{\alpha_i \geq 0} \min\limits_{\omega,b} L(\omega,b,\alpha)$

求解 $\theta_d(\alpha) = \min\limits_{\omega,b} L(\omega,b,\alpha)$

$$\frac{\partial L(\omega,b,\alpha)}{\partial \omega} = \omega - \sum_{i=1}^{n} \alpha_i y_i x_i = 0$$

$$\frac{\partial L(\omega,b,\alpha)}{\partial b} = \sum_{i=1}^{n} \alpha_i y_i = 0$$

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

$$\frac{1}{2}\|\omega\|^2 = \frac{1}{2}\sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$-\sum_{i=1}^{n} \alpha_i \left(y_i \left(\omega^T x_i + b\right) - 1\right) = -\left(\sum_{i=1}^{n} \alpha_i y_i \omega^T x_i + \sum_{i=1}^{n} \alpha_i y_i b - \sum_{i=1}^{n} \alpha_i\right) = -\sum_{i=1}^{n} \alpha_i y_i \omega^T x_i + \sum_{i=1}^{n} \alpha_i$$

非线性 SVM

最大间隔

得到：

$$\theta_d(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

目标函数：

$$\max_{\alpha_i \geq 0} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

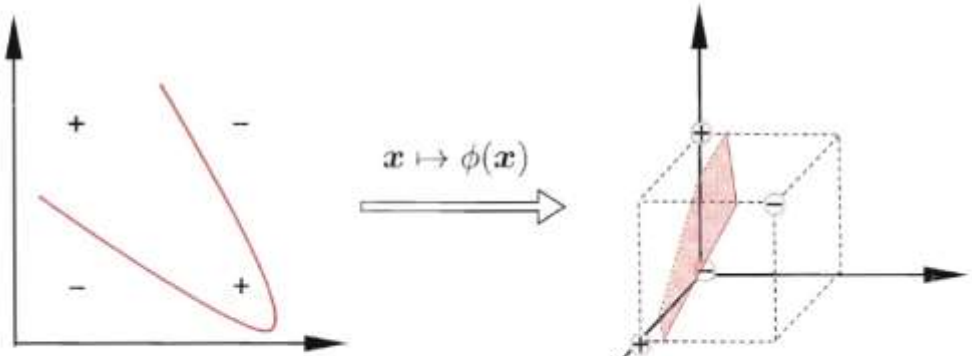
$$s.t. \quad \alpha_i \geq 0, \ i = 1, 2, \ldots n$$

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

非线性 SVM

核函数方法

非线性映射



扩展的线性模型 $f(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\phi(\boldsymbol{x}) + b$ $\Rightarrow$

$$f(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\phi(\boldsymbol{x}) + b$$

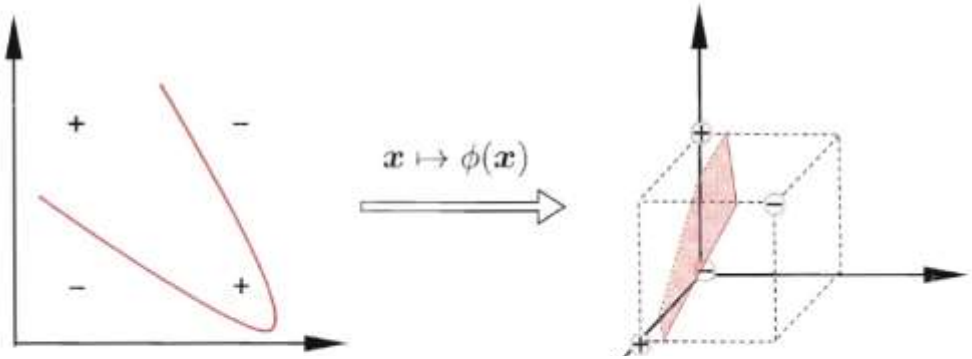
$$\boldsymbol{w} = \sum_{i=1}^{m} \alpha_i \phi(\boldsymbol{x}_i)$$

$$= \sum_{i=1}^{m} \alpha_i \phi(\boldsymbol{x}_i)^{\mathrm{T}}\phi(\boldsymbol{x}) + b$$

非线性 SVM

核函数方法

非线性映射



扩展的线性模型 $f(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\phi(\boldsymbol{x}) + b$ $\Rightarrow$

$$f(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\phi(\boldsymbol{x}) + b$$

$$\boldsymbol{w} = \sum_{i=1}^{m} \alpha_i \phi(\boldsymbol{x}_i)$$

$$= \sum_{i=1}^{m} \alpha_i \phi(\boldsymbol{x}_i)^{\mathrm{T}}\phi(\boldsymbol{x}) + b$$

核函数方法:避免直接求非线性映射，由核函数替代内积运算

$$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle = \phi(\boldsymbol{x}_i)^{\mathrm{T}}\phi(\boldsymbol{x}_j)$$

$\Rightarrow$

$$= \sum_{i=1}^{m} \alpha_i \kappa(\boldsymbol{x}, \boldsymbol{x}_i) + b$$

非线性 SVM

核函数方法

定理 (核函数) 令 $\mathcal{X}$ 为输入空间, $\kappa(\cdot,\cdot)$ 是定义在 $\mathcal{X}\times\mathcal{X}$ 上的对称函数, 则 κ 是核函数当且仅当对于任意数据 $D=\{\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_m\}$, "核矩阵" (kernel matrix) $\mathbf{K}$ 总是半正定的:

$$\mathbf{K}=\begin{bmatrix}\kappa(\boldsymbol{x}_1,\boldsymbol{x}_1) & \cdots & \kappa(\boldsymbol{x}_1,\boldsymbol{x}_j) & \cdots & \kappa(\boldsymbol{x}_1,\boldsymbol{x}_m)\\ \vdots & \ddots & \vdots & \ddots & \vdots\\ \kappa(\boldsymbol{x}_i,\boldsymbol{x}_1) & \cdots & \kappa(\boldsymbol{x}_i,\boldsymbol{x}_j) & \cdots & \kappa(\boldsymbol{x}_i,\boldsymbol{x}_m)\\ \vdots & \ddots & \vdots & \ddots & \vdots\\ \kappa(\boldsymbol{x}_m,\boldsymbol{x}_1) & \cdots & \kappa(\boldsymbol{x}_m,\boldsymbol{x}_j) & \cdots & \kappa(\boldsymbol{x}_m,\boldsymbol{x}_m)\end{bmatrix}.$$

定理表明, 只要一个对称函数所对应的核矩阵半正定, 它就能作为核函数使用. 事实上, 对于一个半正定核矩阵, 总能找到一个与之对应的映射 ϕ. 换言之, 任何一个核函数都隐式地定义了一个称为"再生核希尔伯特空间" (Reproducing Kernel Hilbert Space, 简称 RKHS)的特征空间.

非线性 SVM

核函数方法

常用核函数

名称	表达式	参数
线性核	$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j$	
多项式核	$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j)^d$	$d \geqslant 1$ 为多项式的次数
高斯核	$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}\right)$	$\sigma > 0$ 为高斯核的带宽(width)
拉普拉斯核	$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|}{\sigma}\right)$	$\sigma > 0$
Sigmoid 核	$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \tanh(\beta \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j + \theta)$	tanh 为双曲正切函数, $\beta > 0$, $\theta < 0$

非线性 SVM

核函数方法

核函数化 SVM

线性 SVM

$$\min_{\alpha} \quad \frac{1}{2}\sum_{i,j=1}^{n} \alpha_i\alpha_j y_i y_j x_i^\top x_j - \sum_{i=1}^{n}\alpha_i$$

$$s.t. \quad \alpha_i \geq 0 \quad i = 1 \ldots n$$

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

决策函数：

$$f(x) = \sum_{i=1}^{n}\alpha_i y_i x_i^\top x + b$$

$\Rightarrow$

非线性 SVM

$$\min_{\alpha} \quad \frac{1}{2}\sum_{i,j=1}^{n} \alpha_i\alpha_j y_i y_j k(x_i, x_j) - \sum_{i=1}^{n}\alpha_i$$

$$s.t. \quad \alpha_i \geq 0 \quad i = 1 \ldots n$$

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

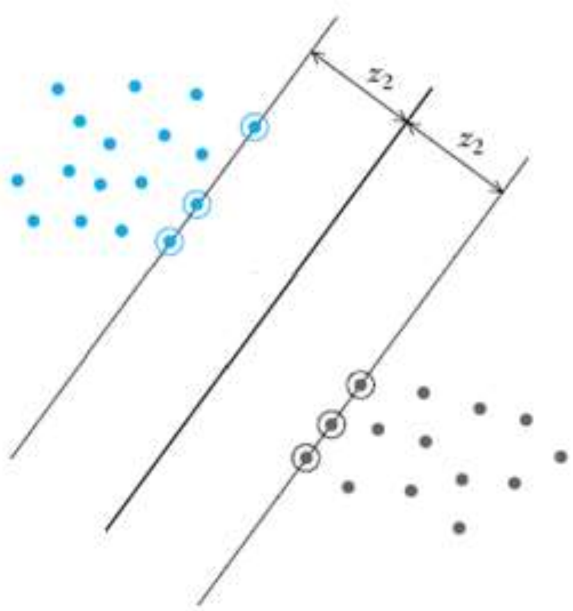
决策函数：

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

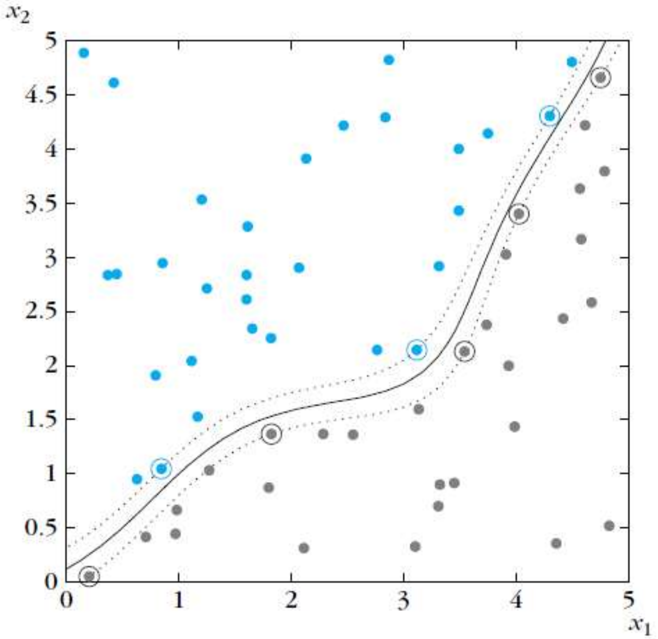
非线性 SVM

核函数方法

核函数化 SVM



线性 SVM 决策界



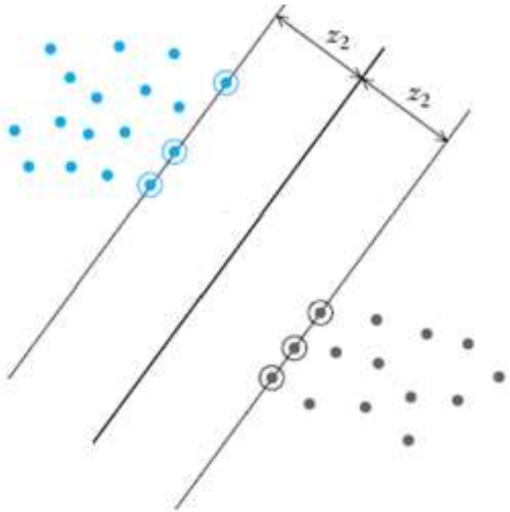
非线性 SVM 决策界

SVM 算法

硬间隔 SVM

$$\min_{w,b}\frac{1}{2}\|w\|^2$$

$$s.t \quad y_i\left(w^T x_i + b\right) - 1 \geq 0, \quad i=1,2,\ldots l$$



硬间隔 SVM 对偶问题：

$$\min_{\alpha} \quad \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j K(x_i, x_j) - \sum_{j=1}^{l}\alpha_j$$

$$\text{s.t.} \quad \sum_{i=1}^{l} y_i \alpha_i = 0,$$

$$\alpha_i \geqslant 0, \quad i = 1, \cdots, l.$$

(1)解对偶问题，得 $\boldsymbol{\alpha}=(\alpha_1,\ldots\alpha_l)^{\mathrm{T}}$

(2)求 w, b:

$$\boldsymbol{w}=\sum_{i=1}^{l} y_i \alpha_i \boldsymbol{x}_i$$

$$y_j\left(w^T x_j + b\right) - 1 = 0, j \in \{j | \alpha_j > 0\} \quad \Rightarrow \quad b = y_j - \sum_{i=1}^{l} y_i \alpha_i k(\boldsymbol{x}_i, \boldsymbol{x}_j), \quad j \in \{j | \alpha_j > 0\}$$

(3)决策函数 $f(\boldsymbol{x}) = \operatorname{sgn}\left(\sum_{i=1}^{l} \alpha_i y_i k(x_i, x_j) + b\right)$

SVM 算法

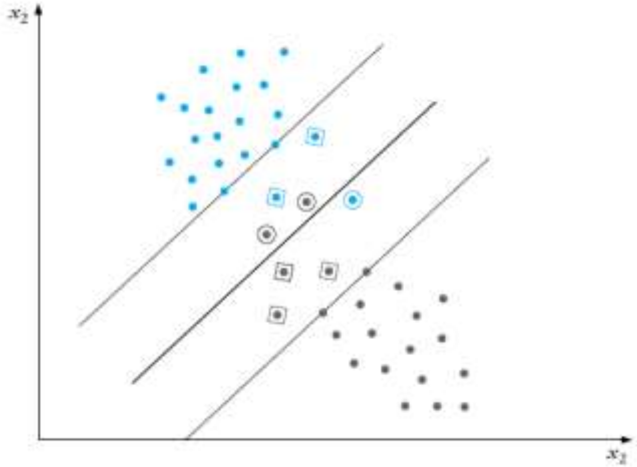
软间隔 SVM

$$\min_{\mathbf{w}\in\mathcal{H},b\in\mathbf{R},\xi\in\mathbf{R}^l} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{l}\xi_i$$

$$\text{s.t.} \quad y_i((\mathbf{w}\cdot\mathbf{x}_i)+b) \geqslant 1-\xi_i\ ,\ i=1,\cdots,l\ ,$$

$$\xi_i \geqslant 0\ ,\ i=1,\cdots,l.$$

$$C>0$$



软间隔 SVM 对偶问题

$$\min_{\boldsymbol{\alpha}} \quad \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j K(x_i, x_j) - \sum_{j=1}^{l}\alpha_j$$

$$\text{s.t.} \quad \sum_{i=1}^{l} y_i\alpha_i = 0\ ,$$

$$0 \leqslant \alpha_i \leqslant C,\quad i=1,\cdots,l\ .$$

(1)解对偶问题，得 $\boldsymbol{\alpha}=(\alpha_1,\ldots\alpha_l)^{\mathrm{T}}$

(2)求 w, b

$$\boldsymbol{w}=\sum_{i=1}^{l} y_i\alpha_i\boldsymbol{x}_i$$

$$b=y_j-\sum_{i=1}^{l} y_i\alpha_i k(\boldsymbol{x}_i,\boldsymbol{x}_j),\quad j\in\{j|0<\alpha_j<C\}$$

(3)决策函数 $f(\boldsymbol{x})=\operatorname{sgn}(\sum_{i=1}^{l}\alpha_i y_i k(x_i,x_j)+b)$

本章小结

掌握

1. 决策树
2. 集成学习

了解

1. 最近邻
2. SVM
3. 核函数方法

参考文献

1. 机器学习，周志华，清华大学，2016.
2. 统计学习方法，李航，清华大学，2012.
3. 模式识别（第二版），边肇祺，张学工等,清华大学出版社，2000.1
4. 数据挖掘中的新方法-支持向量机，邓乃扬，田英杰。