监督学习
Supervised Learning

徐 君

提纲

- 背景介绍
- 常用的监督学习算法
  - 决策树
  - 支持向量机/Perceptron
  - AdaBoost
- 结构化输出预测
- 总结

机器学习
Machine Learning

- 研究一类算法，使之
  - 在某些任务上(task)
  - 通过已有的观测经验(数据)(experience)
  - 提升算法效果(performance)
- 任务举例：网页分类

f

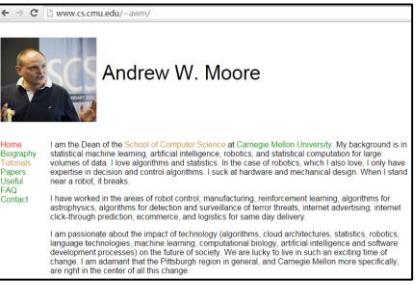
类别
个人主页
公司主页
科研机构主页
新闻网站主页
其它网页

网页分类

训练样本

人工标注

科研机构



个人

公司

科研机构

分类训练算法

网页分类模型f

未观测样本

分类预测算法

科研机构？
个人？
公司？

机器学习分类

- 监督学习(本次课程)
- 无监督学习(下次课程)
- 半监督学习(semi-supervised learning)
- 强化学习(reinforcement learning)
- ……

学习资料：Andrew W. Moore教授编写的课件

http://www.autonlab.org/tutorials/list.html

监督学习形式化描述

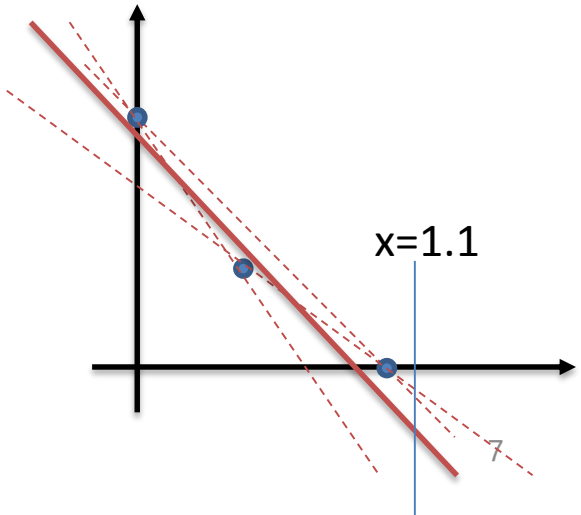
- 问题描述
  - 样本空间X (如X为所有的网页集合) 和目标空间Y (如Y={个人主页, 公司主页, 科研机构主页, 其他网页})
  - 预测函数$f: X \to Y$
  - 假设空间$H = \{h | h: X \to Y\}$
- 算法输入
  - 训练集合$\mathrm{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, 其中$(\mathbf{x}_i, y_i) \in X \times Y$
- 算法输出
  - 根据D中的样本估计“最优”的函数$h \in H$, 使之能够对未出现在D中的样本的标签进行预测

  反例：机械学习！

举例：单输入函数拟合

- 样本空间X=R；目标空间Y=R
- 预测函数$f: X \to Y$
- 假设空间$H = \{h | h = ax + b\}$, 线性函数
- 算法输入：$D = \{(0,1),(1,0),(0.5,0.4)\}$
- 算法输出：$f(x) = ax + b$, 其中a和b(模型参数)的值由算法给出，不同的算法策略估计出不同的值

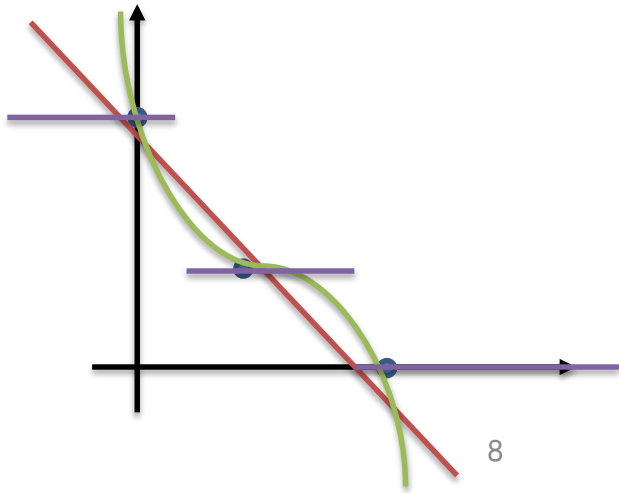
- 模型预测：x=1.1, y=?
  - 返回$1.1 \times a + b$



一个重要的问题

- 算法从未见过x=1.1的样本，为何能对其对应的y值做出预测?
  - 根据观测D中的数据 (经验)推断(猜)出其它x所对应的y值
- 基于假设
  - 模型为某类型函数（本例子中：f为线性函数）
    - 倾向于更加简单的函数（良好的泛化能力）
  - D中的训练样本与测试样本独立同分布(I.I.D.)
  - D中的训练数据足够多
- 能够100%肯定预测出的y值吗？
  - 不能! 预测会有错误!

具体参考机器学习中PAC学习理论



二值分类问题举例（训练数据）

是否批准放贷？

ID	Age	Has_Job	Own_House	Credit_Rating	Class
1	young	false	false	fair	**No**
2	young	false	false	good	**No**
3	young	true	false	good	**Yes**
4	young	true	true	fair	**Yes**
5	young	false	false	fair	**No**
6	middle	false	false	fair	**No**
7	middle	false	false	good	**No**
8	middle	true	true	good	**Yes**
9	middle	false	true	excellent	**Yes**
10	middle	false	true	excellent	**Yes**
11	old	false	true	excellent	**Yes**
12	old	false	true	good	**Yes**
13	old	true	false	good	**Yes**
14	old	true	false	excellent	**Yes**
15	old	false	false	fair	**No**

二值分类问题举例（学习任务）

- 样本空间X为4个特征组成的空间；目标空间Y={Yes, No}
- 预测函数$f: X \mapsto Y$
- 假设空间$H = \{h | h$为树决策函数$\}$
- 算法输入：D为15个观测样本及其类别标签
- 算法输出：$f(x)$
  - $f(\mathbf{x})$=*Yes*: 发放贷款
  - $f(\mathbf{x})$=*No*: 不发放贷款
- 测试过程：对新的测试样例，预测是否对其发放贷款

Age	Has_Job	Own_house	Credit-Rating	Class
young	false	false	good	?

监督学习算法分类

任务

模型函数	(二值)分类 Y为类别集合	回归 Y为实数集合	结构预测/排序 Y为结构类型/排列
线性函数 $f(x)=\langle w, x\rangle$	Perceptron, SVM	Ridge regression SVR	Structured SVM, Ranking SVM
树	Decision tree	Regression tree	LambdaMART
神经网络	神经网络	神经网络	RankNet
叠加函数 $f(x)=\sum_t h_t(x)$ **(additive model)**	AdaBoost	Boosted Regression Tree	RankBoost

本次课程关注（二值）分类和结构输出预测

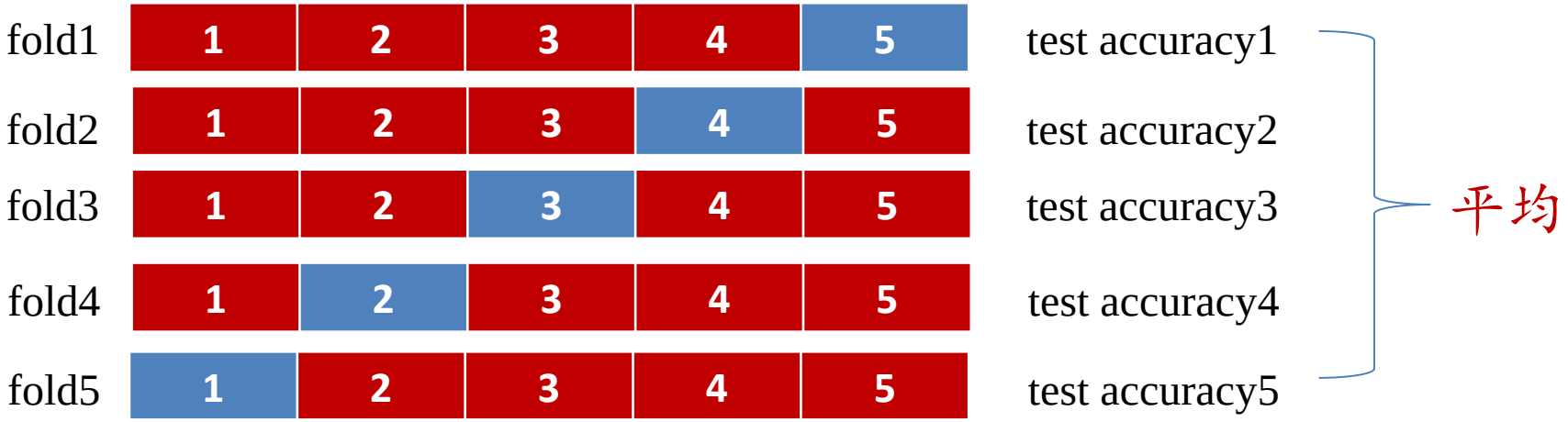
(Binary) Classification、Structured Output Prediction

模型评价

- 目的
  - 评估不同机器学习模型、不同参数设置的优劣
  - 在线应用模型前对预测精度进行估计
- 方法
  - 在预留的带标签数据集合（测试集合）上进行测试
- 需要解决的问题
  - 如何在有限的测试集合上得到更准确的预测精度估计

分类模型评价方法

- 将标注的数据集合随机分成两份
  - 训练数据(training data):用于学习分类模型
  - 测试数据(test data):用于测试分类模型
  - 测试数据必须和训练数据分开，保证测试结果准确
- n折交叉验证(n-fold cross-validation)



分类模型评价—评价方法

- Leave-one-out cross validation:
  - N个训练样本，N-1个样本作为训练集合，在1个样本上测试，重复N次。
  - 是n-fold cross-validation的一种特殊情况，样本量很少时才采用。
- 验证集合(validation set)
  - 从训练集合中再分一部分验证数据，用于调试模型的参数
  - 在验证集合上取得最好效果的参数设置被用于最终的模型训练
  - 模型的最终评价在测试集合(test set)上得出

分类模型评价—评价准则

	预测为正样本	预测为负样本
标注为正样本	TP (true positive)	FN (false negative)
标注为负样本	FP (false positive)	TN (true negative)

- 常用的分类评价准则
  - $\text{Accuracy}=\frac{\text{正确分类的样本数}}{\text{总样本数}}=\frac{TP+TN}{TP+FN+FP+TN}$
  - $\text{Precision}=\frac{\text{正确预测的正样本数}}{\text{预测为正例的样本数}}=\frac{TP}{TP+FP}$
  - $\text{Recall}=\frac{\text{正确预测的正样本数}}{\text{标注的正样本数}}=\frac{TP}{TP+FN}$
  - $\text{F1}=2\times\frac{\text{Precision}\times\text{Recall}}{\text{Precision}+\text{Recall}}$

问题：在什么情形下不能使用accuracy作为评价指标？

Accuracy存在的问题

	预测为正样本	预测为负样本	
标注为正样本	0	10	10
标注为负样本	0	990	990

$$\text{Accuracy} = \frac{0+990}{0+10+0+990} = 0.99$$

$$\text{Precision} = \frac{0}{0+0} = 0$$

$$\text{Recall} = \frac{0}{0+10} = 0$$

1. 数据不平衡，正负比例为1:99
2. 分类器将所有的样本预测为负例!

	预测为正样本	预测为负样本	
标注为正样本	490	5	495
标注为负样本	5	500	505

$$\text{Accuracy} = \frac{490+500}{490+5+5+500} = 0.99$$

$$\text{Precision} = \frac{490}{490+5} = 0.9899$$

$$\text{Recall} = \frac{490}{490+5} = 0.9899$$

Precision/Recall/F1评价对目标类别（正例）的分类效果。

提纲

- 背景介绍
- 常用的监督学习算法
  - 决策树
  - 支持向量机/Perceptron
  - AdaBoost
- 总结

监督学习算法分类

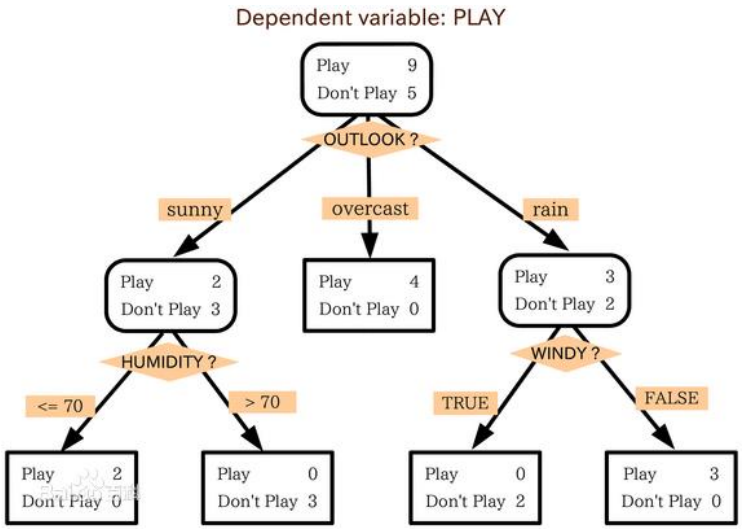
任务

模型函数

	(二值)分类 Y为类别集合	回归 Y为实数集合	结构预测/排序 Y为结构类型/排列
线性函数 **f(x)=⟨w, x⟩**	Perceptron, SVM	Ridge regression SVR	Structured SVM, Ranking SVM
树	Decision tree	Regression tree	LambdaMART
神经网络	神经网络	神经网络	RankNet
叠加函数 **f(x)=$\sum_t h_t(x)$** **(additive model)**	AdaBoost	Boosted Regression Tree	RankBoost

决策树

- 决策树是分类问题中最常用的模型之一
  - 能够接受类别型的特征
  - 分类效果与其它分类算法相当
  - 训练/测试效率高
- 分类函数f为一棵树，称为决策树



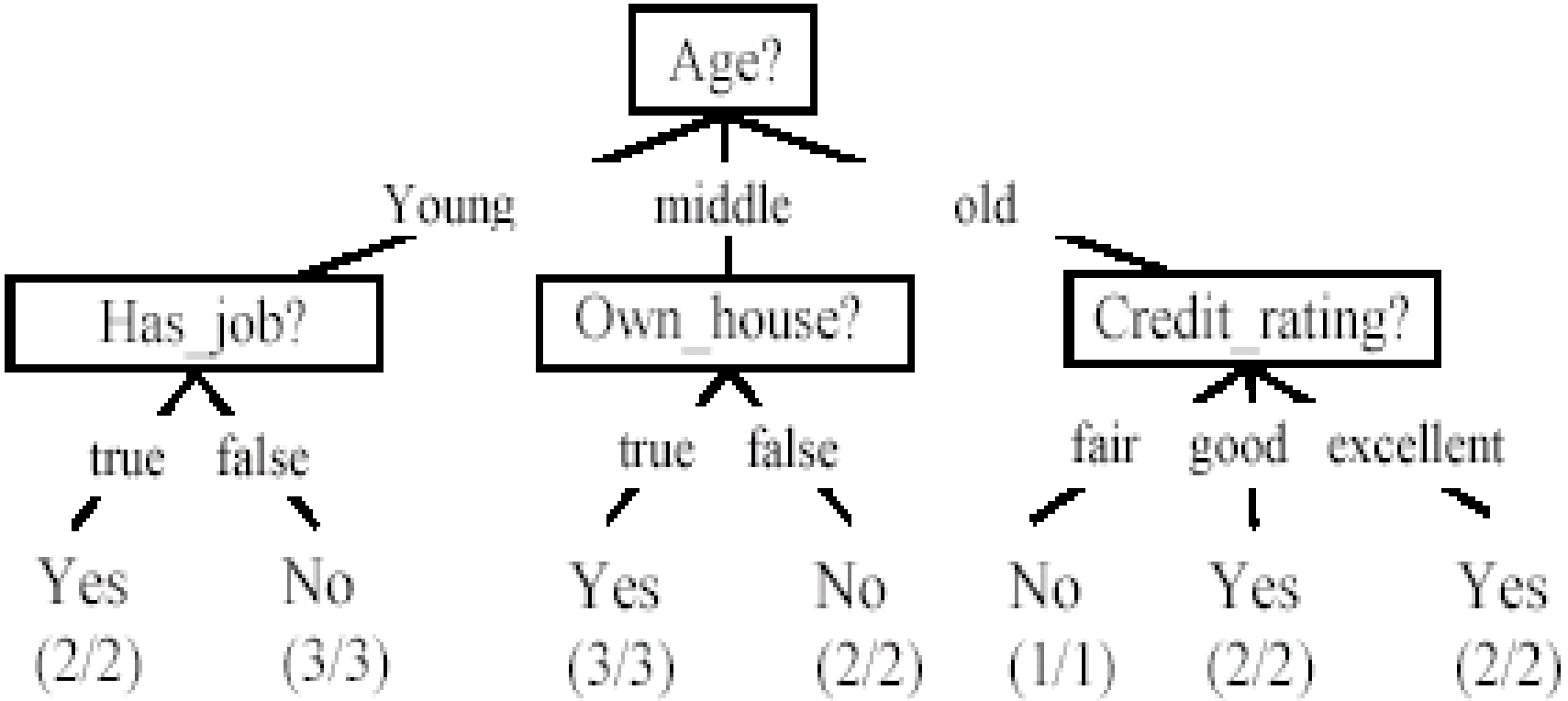
二值分类数据

是否批准放贷？

ID	Age	Has_Job	Own_House	Credit_Rating	Class
1	young	false	false	fair	**No**
2	young	false	false	good	**No**
3	young	true	false	good	**Yes**
4	young	true	true	fair	**Yes**
5	young	false	false	fair	**No**
6	middle	false	false	fair	**No**
7	middle	false	false	good	**No**
8	middle	true	true	good	**Yes**
9	middle	false	true	excellent	**Yes**
10	middle	false	true	excellent	**Yes**
11	old	false	true	excellent	**Yes**
12	old	false	true	good	**Yes**
13	old	true	false	good	**Yes**
14	old	true	false	excellent	**Yes**
15	old	false	false	fair	**No**

决策函数

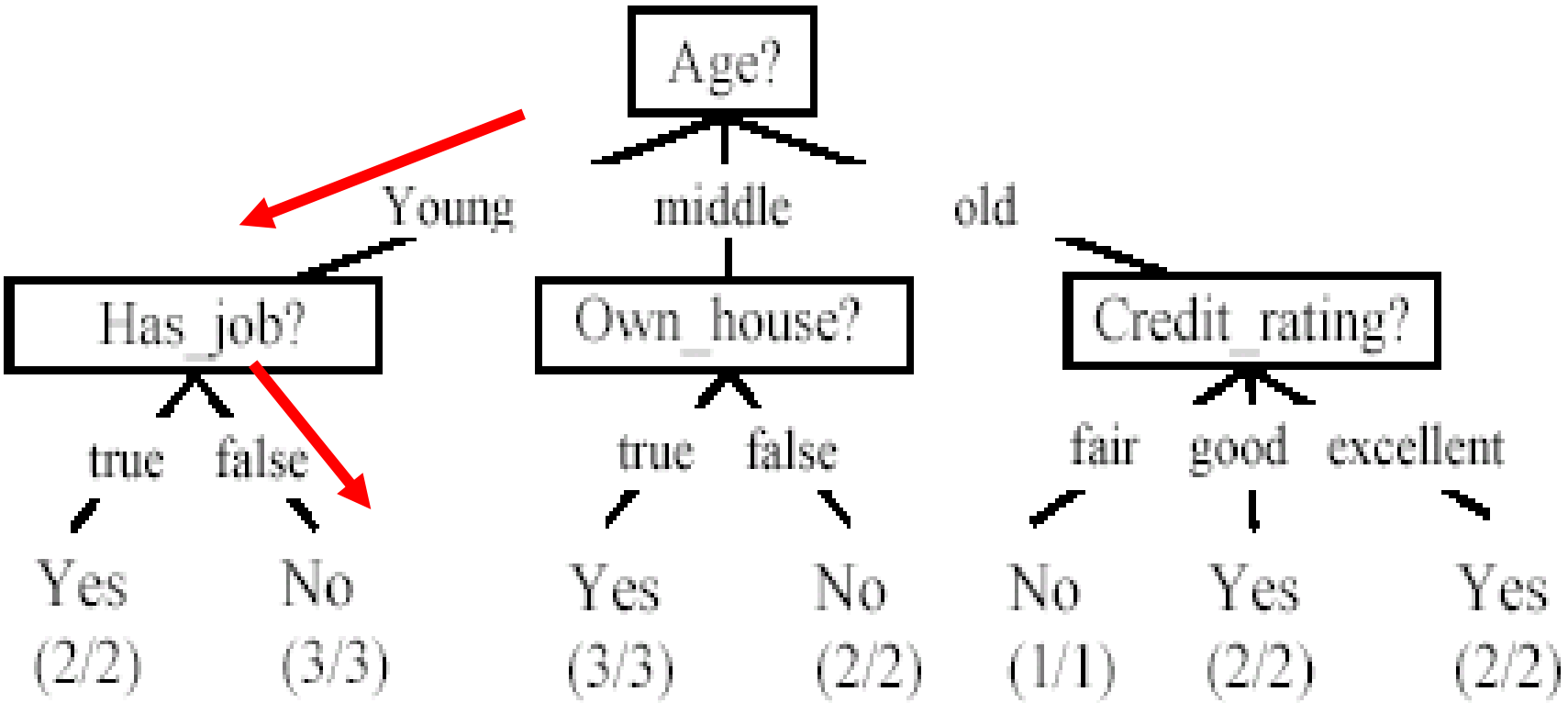
- 中间节点：决策步骤
- 叶子节点：决策结果/类别标签



决策函数测试

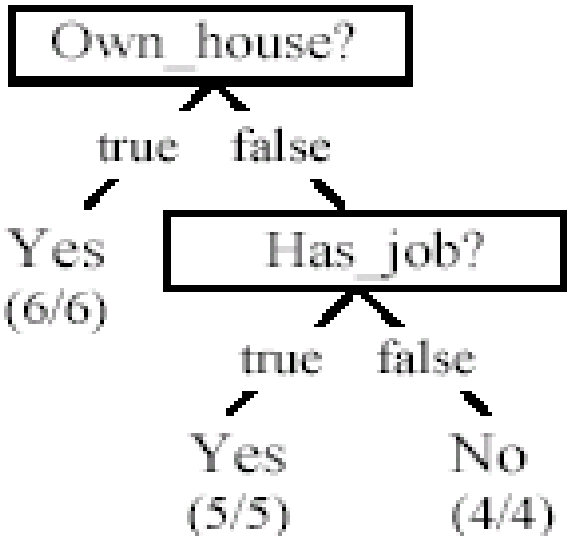
Age	Has_Job	Own_house	Credit-Rating	Class
young	false	false	good	?

No



决策函数的选择

- 给定训练数据，可能存在多棵能够拟合数据的决策树
- 如何选择？
  - 小(简单)的树
  - 拟合精度高的树
- 决策树学习过程
  - 全局最优：NP-hard
  - 利用启发式规则建立次优决策树



决策树训练算法

- 贪心分治算法
  - 假设所有的特征都是类别类型
  - 自顶向下递归建树
  - 初始状态：所有的训练数据都在根节点上
  - 根据选择的特征，对训练数据进行划分
  - 选择特征的依据：信息增益(information gain)
- 终止条件
  - 节点上的所有训练样本都属于一个类别
  - 节点上没有训练样本
  - 没有更多的特征可以供选择

决策树算法

递归出口：数据属于一个类/没有更多的特征供选择，生成叶子节点

选择最优划分特征：
p_0: 划分前的混杂度
p_i: 以特征A_i划分后的混杂度

递归出口：数据划分收益太低

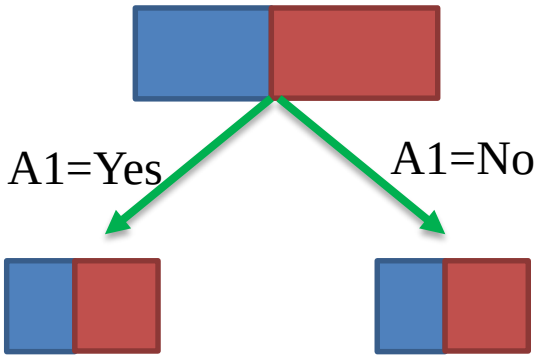
划分数据

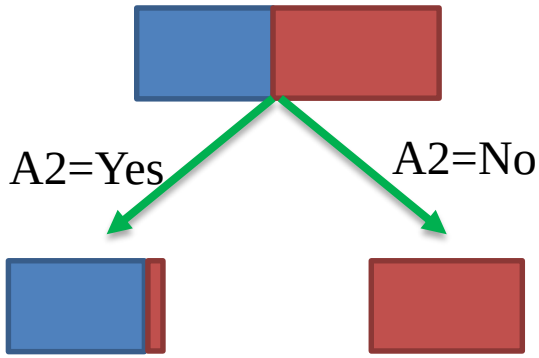
向下递归：逐个递归生成每一棵子树

```
. Algorithm decisionTree(D, A, T)
1    if D contains only training examples of the same class c_j ∈ C then
2        make T a leaf node labeled with class c_j;
3    elseif A = ∅ then
4        make T a leaf node labeled with c_j, which is the most frequent class in D
5    else  // D contains examples belonging to a mixture of classes. We select a single
6          // attribute to partition D into subsets so that each subset is purer
7        p_0 = impurityEval-1(D);
8        for each attribute A_i ∈ {A_1, A_2, ..., A_k} do
9            p_i = impurityEval-2(A_i, D)
10       end
11       Select A_g ∈ {A_1, A_2, ..., A_k} that gives the biggest impurity reduction,
             computed using p_0 − p_i;
12       if p_0 − p_g < threshold then     // A_g does not significantly reduce impurity p_0
13         make T a leaf node labeled with c_j, the most frequent class in D.
14       else                              // A_g is able to reduce impurity p_0
15           Make T a decision node on A_g;
16           Let the possible values of A_g be v_1, v_2, ..., v_m. Partition D into m
                 disjoint subsets D_1, D_2, ..., D_m based on the m values of A_g.
17           for each D_j in {D_1, D_2, ..., D_m} do
18               if D_j ≠ ∅ then
19                   create a branch (edge) node T_j for v_j as a child node of T;
20                   decisionTree(D_j, A-{A_g}, T_j)// A_g is removed
21               end
22           end
23       end
24   end
```

选择决策特征

- 核心步骤：选择合适的特征进行决策，划分数据，生成子节点
- 合适：尽量大的减少划分后子数据集的混杂度

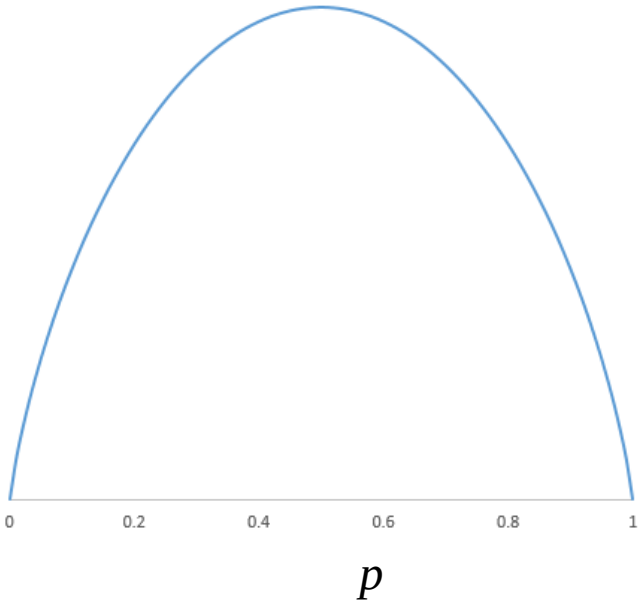




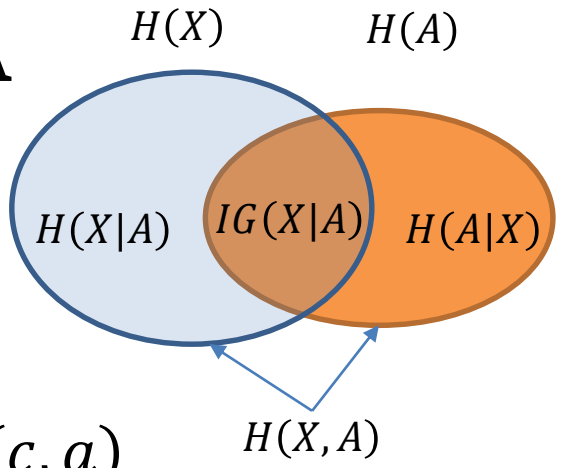
混杂度度量工具—熵

- 熵(entropy)

  给定一个随机变量X，其取值为$\{P(X = V_1) = P_1, \cdots, P(X =$



两个随机变量X和A



- 联合熵(joint entropy)

$$H(X,A) = -\sum_{c=1}^{|C|}\sum_{a=1}^{|A|} P(c,a)\log P(c,a)$$

  - 如X与A独立：$P(c,a) = P(c)P(a) \Rightarrow H(X,A) = H(X) + H(A)$
- 条件熵(conditional entropy): 给定随机变量A后，X剩余的不确定性(熵)

$$H(X|A) = \sum_{a=1}^{|A|} P(A=a)H(X|A=a)$$

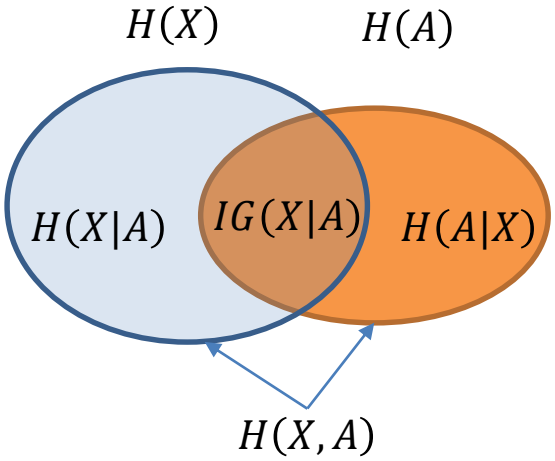
  - 如X与A独立$\Rightarrow H(X|A) = H(X)$
- Chain Rule

$$H(A,X) = H(A) + H(X|A)$$

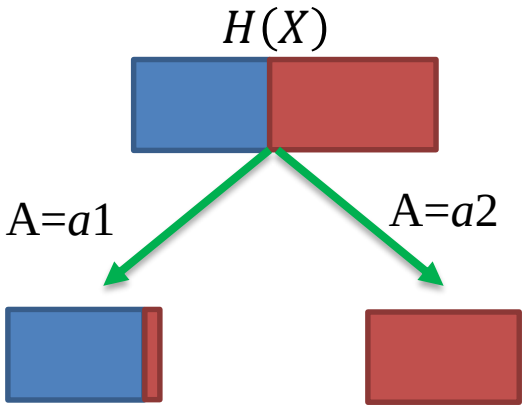
条件熵与信息增益

- 信息增益(information gain, IG)定义为给定随机变量A后，X所减少的不确定性

$$IG(X|A) = H(X) - H(X|A)$$
$$= H(X) - \sum_j P(A = a_j) H(X|A = a_j)$$



ImpurityEval-1:



信息增益性质：

$$IG(X|A) = IG(A|X)$$
$$IG(X|X) = H(X)$$
$$IG(X|A) = H(X) + H(A) - H(X,A)$$

ImpurityEval-2:

$$H(X|A) = H(X|A = a1)P(A = a1) + H(X|A = a2)P(A = a2)$$

举例

X	Y
Math	Yes
History	No
CS	Yes
Math	No
Math	No
CS	Yes
History	No
Math	Yes

- X:学生专业；Y:是否喜欢玩游戏
- X和Y的分布
  - $\Pr(X = Math) = 0.5, \Pr(X = \text{CS}) = 0.25,$ $\Pr(X = \text{History}) = 0.25$
  - $\Pr(Y = Yes) = 0.5,\ \Pr(Y = No) = 0.5$
  - 以2为底，计算出H(X)=1.5; H(Y)=1.
- P(Y|X)
  - X=Math(4项): P(Y=yes|X=Math)=0.5, P(Y=No|X=Math)=0.5 **H(Y|X=Math)=1**
  - X=CS(2项): P(Y=yes|X=Math)=1; P(Y=No|X=Math)=0 **H(Y|X=CS)=0**
  - X=History(2项): P(Y=Yes|X=History)=0; P(Y=No|X=History)=1 **H(Y|X=History)=0**
- $IG(Y|X) = H(Y) - H(Y|X) = 1 - 0.5 \times 1 - 0 - 0 = 0.5$

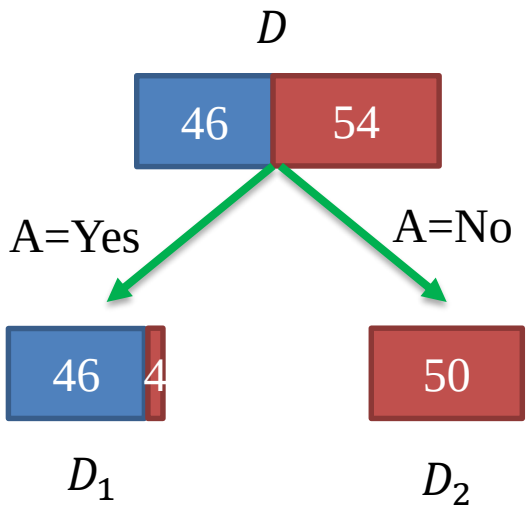
练习：计算IG(X|Y)

选择决策特征

- 数据集合D的熵定义为

$$H(X) = -\sum_{c=1}^{|C|} P_c \log P_c$$

其中 $P_c = \frac{\text{属于类别}c\text{的样本数}}{D\text{中的样本数}}$，$X$对应类别标签



- 选择特征A所带来的信息增益

$$\mathrm{IG}(X|A) = H(X) - \left(\frac{|D_1|}{|D|}H(X|A=Yes) + \frac{|D_2|}{|D|}H(X|A=\mathrm{No})\right)$$

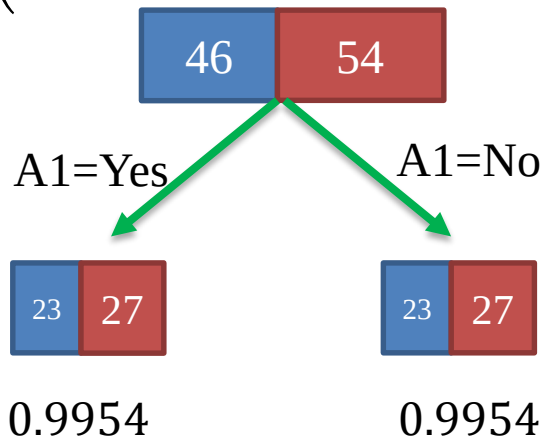
- 注意：特征A可以有多个取值

选择决策特征

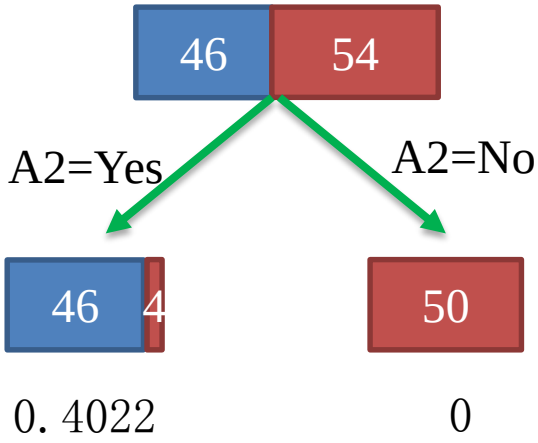
$$\mathrm{H(X)} = -\sum_{c=1}^{|C|} P_c \log P_c$$

$$\mathrm{IG}(X|A) = H(X) - \left(\frac{|D_1|}{|D|} H(X|A = \mathrm{Yes}) + \frac{|D_2|}{|D|} H(X|A = \mathrm{No})\right)$$

$$H(X) = -\left(\frac{46}{100} * \log\frac{46}{100} + \frac{54}{100} * \log\frac{54}{100}\right) = 0.9954$$



$H(X) = 0.9954$



$$IG(X|A1) = 0.9954 - \left(\frac{0.9954}{2} + \frac{0.9954}{2}\right)$$

$$= 0$$

$$IG(X|A2) = 0.9954 - \left(\frac{0.4022}{2} + \frac{0}{2}\right)$$

$$= 0.7943$$

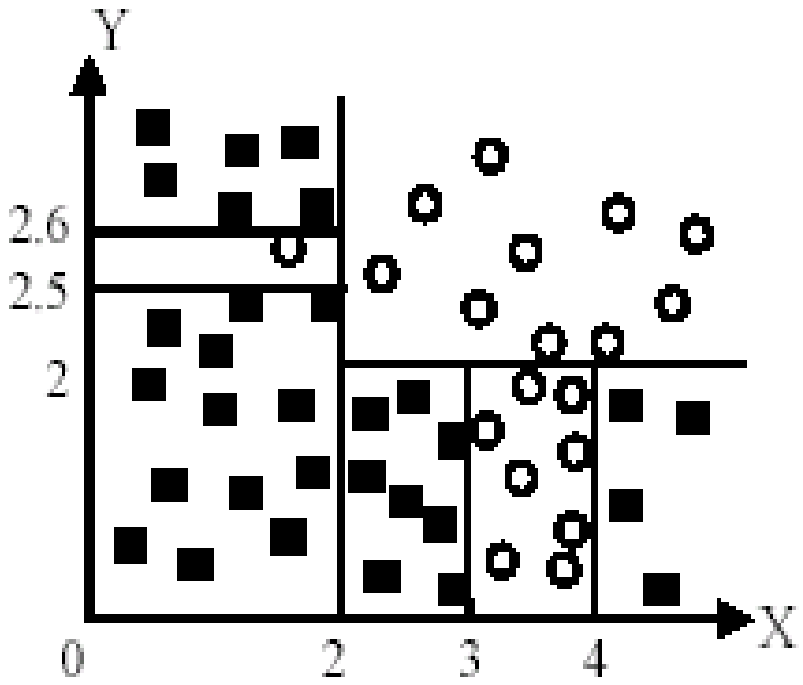
关于决策树算法

- 贪心策略
  - 从根节点开始扩张，已建立的树不再改变
  - 数据划分后不再改变
  - 每次选择局部信息增益最大的特征划分数据
  - 局部最优解
- 递归算法
  - 每一个决策节点用相同的策略扩张
- IG不是唯一的选择
  - IG ratio

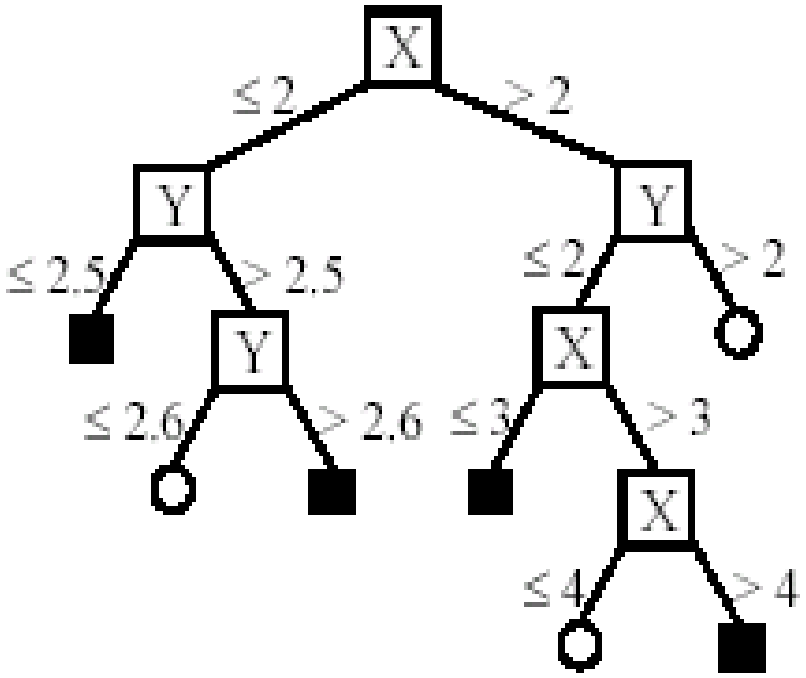
决策树算法改进1:数值型特征

- 上述算法只能处理类别型特征
  - 选择的特征取K个类别→分裂成K个子节点
- 数值型特征
  - 寻找阈值，将数值划分为(两个)多个区间
- 如何确定阈值
  - 将数据中出现过特征值从大到小排序$\{v_1, v_2, \cdots, v_N\}$
  - 可能的阈值有$\left\{\frac{v_1+v_2}{2}, \frac{v_2+v_3}{2}, \cdots, \frac{v_{N-1}+v_N}{2}\right\}$
  - 按照最大化信息增益原则选择一个阈值

数值型特征建立的决策树



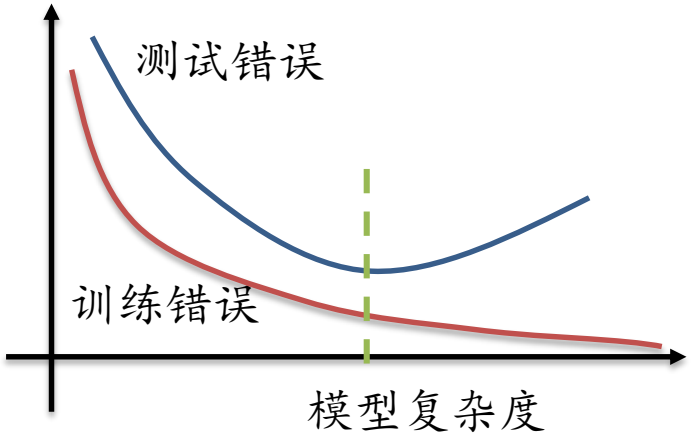
(A) A partition of the data space



(B). The decision tree

过拟合问题

- 过拟合：机器学习算法拟合噪声数据，而不是数据中的有意义规律
- 产生的现象：在训练数据上得到的精度远大于在测试数据上得到的精度
- 过拟合问题不仅仅存在于决策树学习，是机器学习算法面临的共同问题
- 原因
  - 训练数据量不够
  - 模型过于复杂，模型参数过多
    - 决策树：太深、节点太多。有些节点是在噪声和离群数据的干扰下建立
- 解决方案
  - 降低模型复杂度
  - 决策树：减少节点数目，剪枝

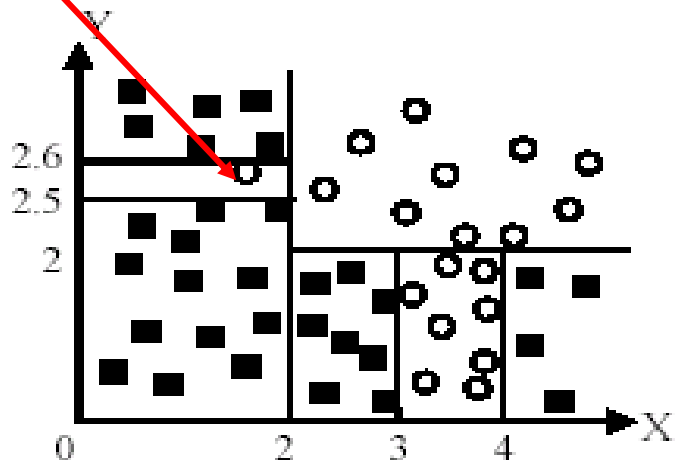


两种剪枝方案

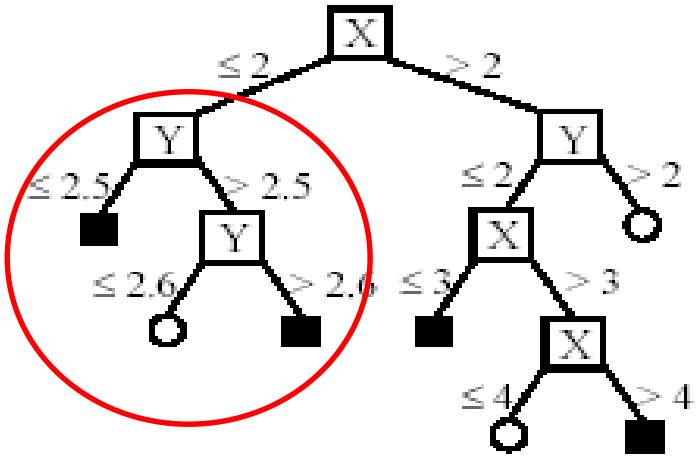
- Pre-pruning
  - 提前结束分裂节点
  - 难以操作：无法预知分裂节点的下一步动作
- Post-pruning
  - 完成决策树构建后，剪除部分节点
  - 常用的方法，可以建立不同的策略进行剪枝（如χ^2测试）
  - 可以利用验证集合进行剪枝（将所有的数据随机分为训练(training), 验证(validation)和测试(test)三部分）

可能过拟合

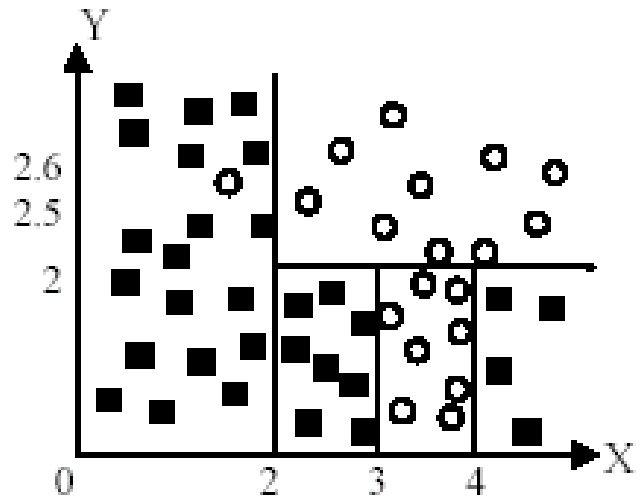
举例:决策树剪枝

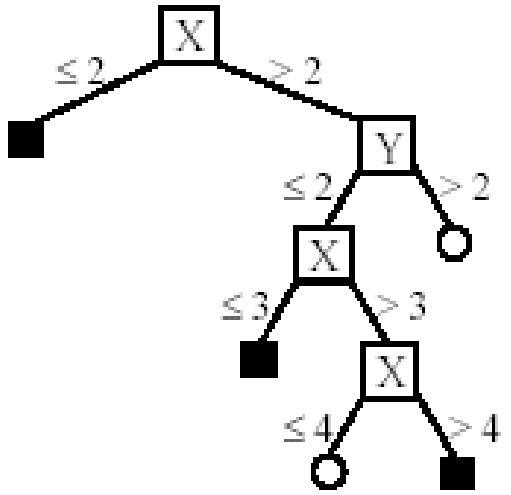


(A) A partition of the data space



(B). The decision tree





关于决策树的其他话题

- 输出变量也为数值型：回归问题
  - Regression tree
- 计算复杂度
- IG以外的数据划分原则
- 决策树可以和Boosting相结合
  - Multiple Additive Regression Trees (MART)

提纲

- 背景介绍
- 常用的监督学习算法
  - 决策树
  - 支持向量机/Perceptron
  - AdaBoost
- 总结

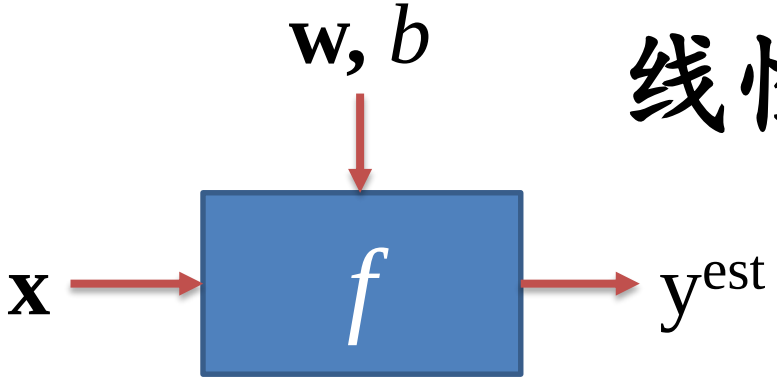
监督学习算法分类

任务

模型函数

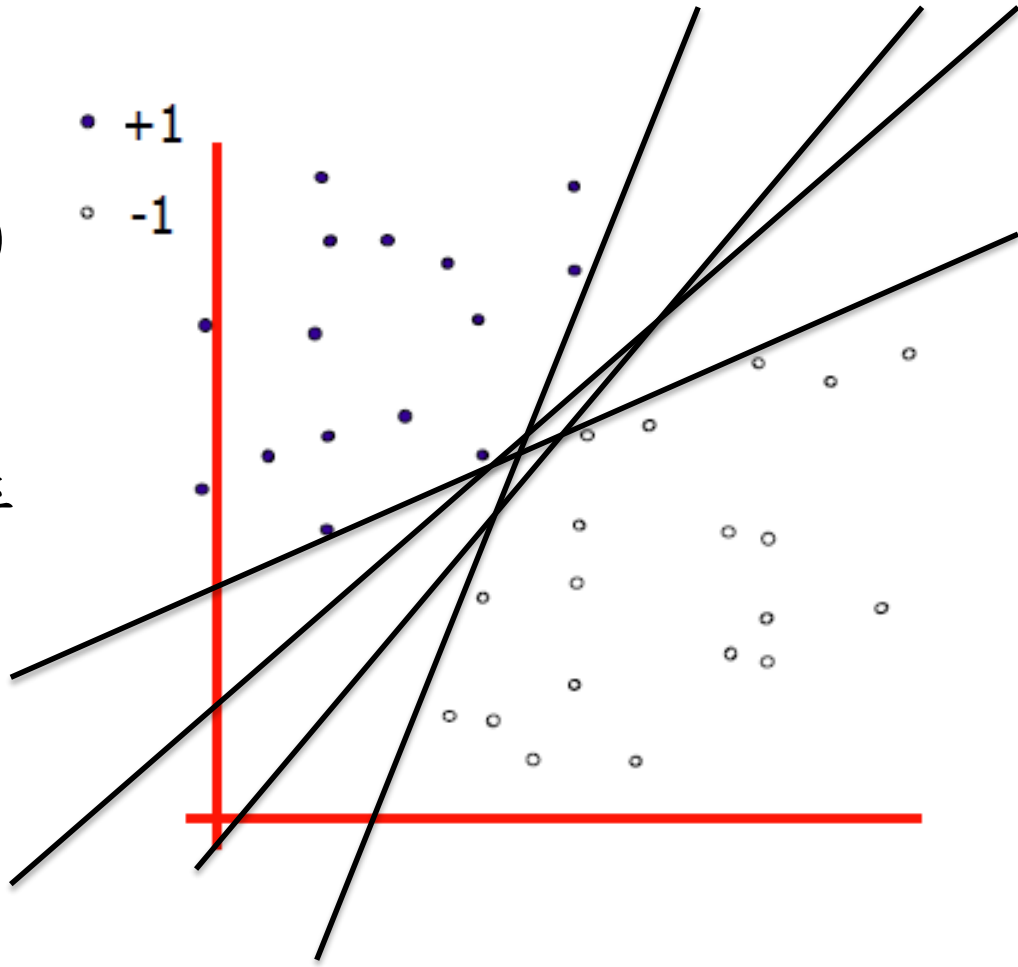
	(二值)分类 Y为类别集合	回归 Y为实数集合	结构预测/排序 Y为结构类型/排列
线性函数 **f(x)=⟨w, x⟩**	Perceptron, SVM	Ridge regression SVR	Structured SVM, Ranking SVM
树	Decision tree	Regression tree	LambdaMART
神经网络	神经网络	神经网络	RankNet
叠加函数 **f(x)=$\sum_t h_t$(x)** **(additive model)**	AdaBoost	Boosted Regression Tree	RankBoost

线性分类器



$$f(\mathbf{x};\mathbf{w},b) = \text{sgn}(\langle \mathbf{w},\mathbf{x}\rangle + b)$$

所有的模型(分类面)错误率为0, 哪个最好?

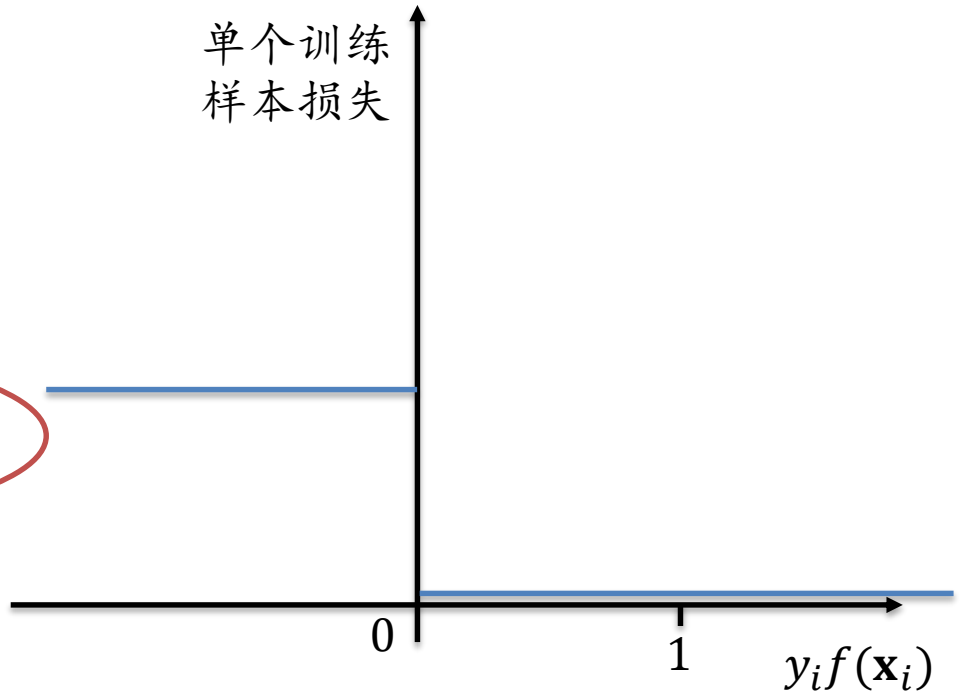


简单的学习规则

- 线性感知机(Perceptron)算法

- 输入: 训练数据$D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, 学习步长$\eta > 0$
- 输出: 感知机模型$\mathbf{w}, b$

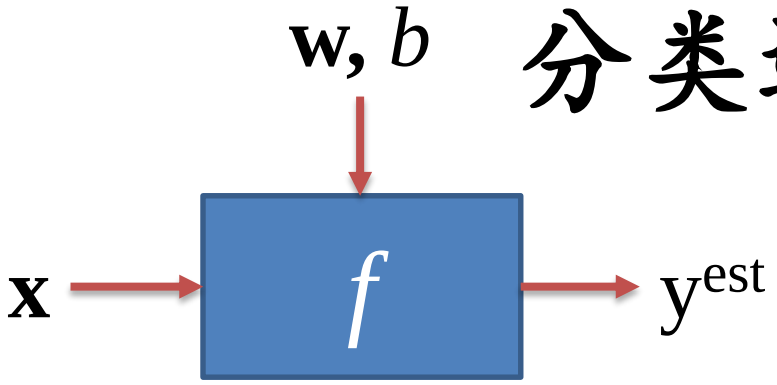
1. $(\mathbf{w}, b) \leftarrow$ 随机值
2. $R \leftarrow \max_i |\mathbf{x}_i|$
3. **repeat**
4. 选取数据$(x_i, y_i) \in D$
5. **if** $y_i f(\mathbf{x}_i) \leq 0$ **then**
6. $\mathbf{w} \leftarrow \mathbf{w} + \eta y_i \mathbf{x}_i$
7. $b \leftarrow b + \eta R y_i$
8. **end if**
9. **until** convergence



关于更新规则的说明

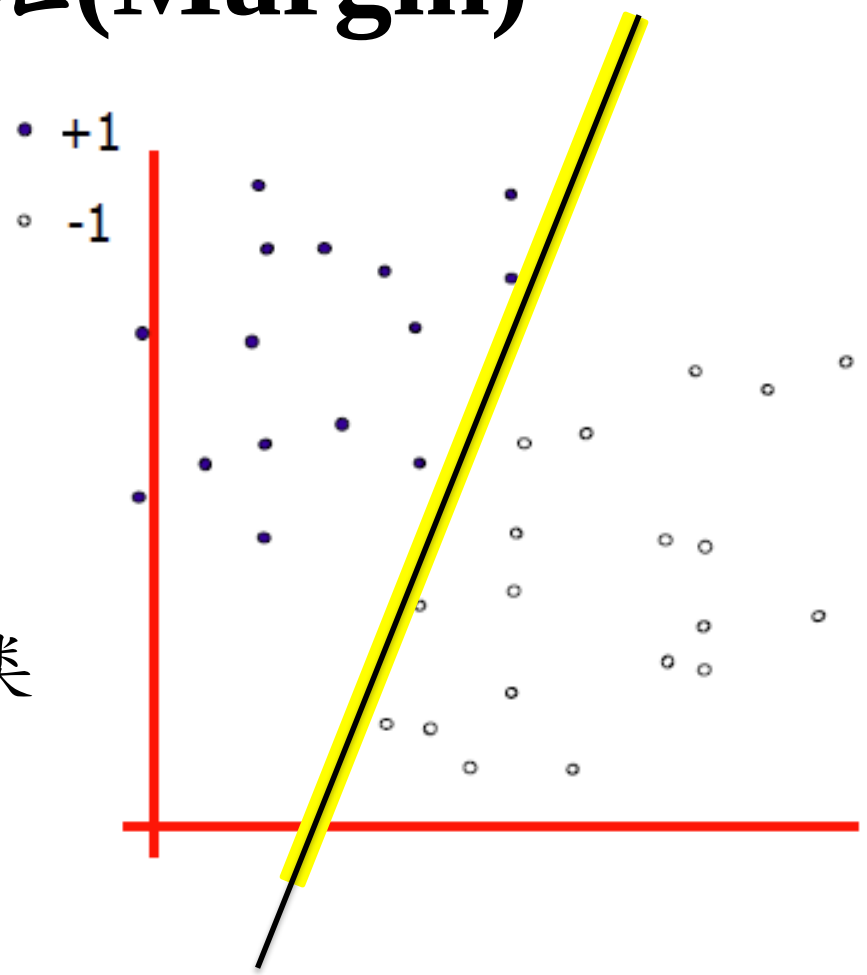
- $\mathbf{w}^{t+1} = \mathbf{w}^{t} + \eta y_i \mathbf{x}_i, b^{t+1} \leftarrow b^{t} + \eta R y_i$
- $y_i f^{\mathrm{t+1}}(\mathbf{x}_i)$
  $= y_i(\langle \mathbf{w}^{t+1}, \mathbf{x}_i \rangle + b^{t+1})$
  $= y_i(\langle \mathbf{w}^{t} + \eta y_i \mathbf{x}_i, \mathbf{x}_i \rangle + b^{t} + \eta R y_i)$
  $= y_i \langle \mathbf{w}^{t}, \mathbf{x}_i \rangle + y_i \langle \eta y_i \mathbf{x}_i, \mathbf{x}_i \rangle + y_i b^{t} + \eta R y_i y_i$
  $= y_i f^{\mathrm{t}}(\mathbf{x}_i) + \eta(\langle \mathbf{x}_i, \mathbf{x}_i \rangle + R)$
  $> y_i f^{\mathrm{t}}(\mathbf{x}_i)$
- 每次更新都将减少当前的错误
- 收敛性：在线性可分的数据上，在有限步更新后收敛(Novikoff (1962))

分类边距(Margin)



$$f(\mathbf{x};\mathbf{w},b) = \text{sgn}(\langle \mathbf{w},\mathbf{x}\rangle + b)$$

分类边距(Margin):对线性分类器而言，边距是从分类面到最近的训练样本的距离。

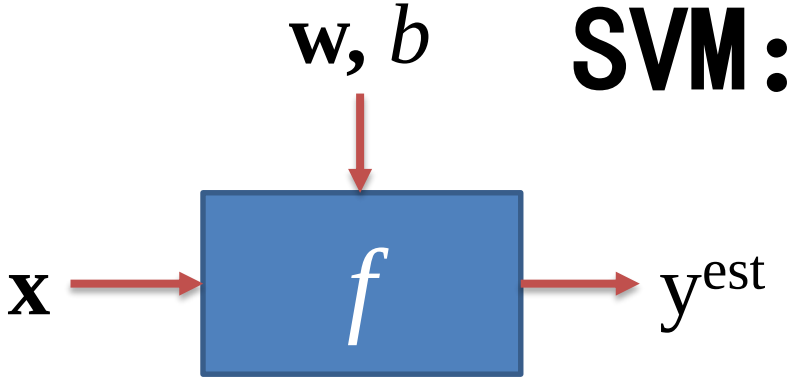


带边距感知机
(Perceptron with Margin)

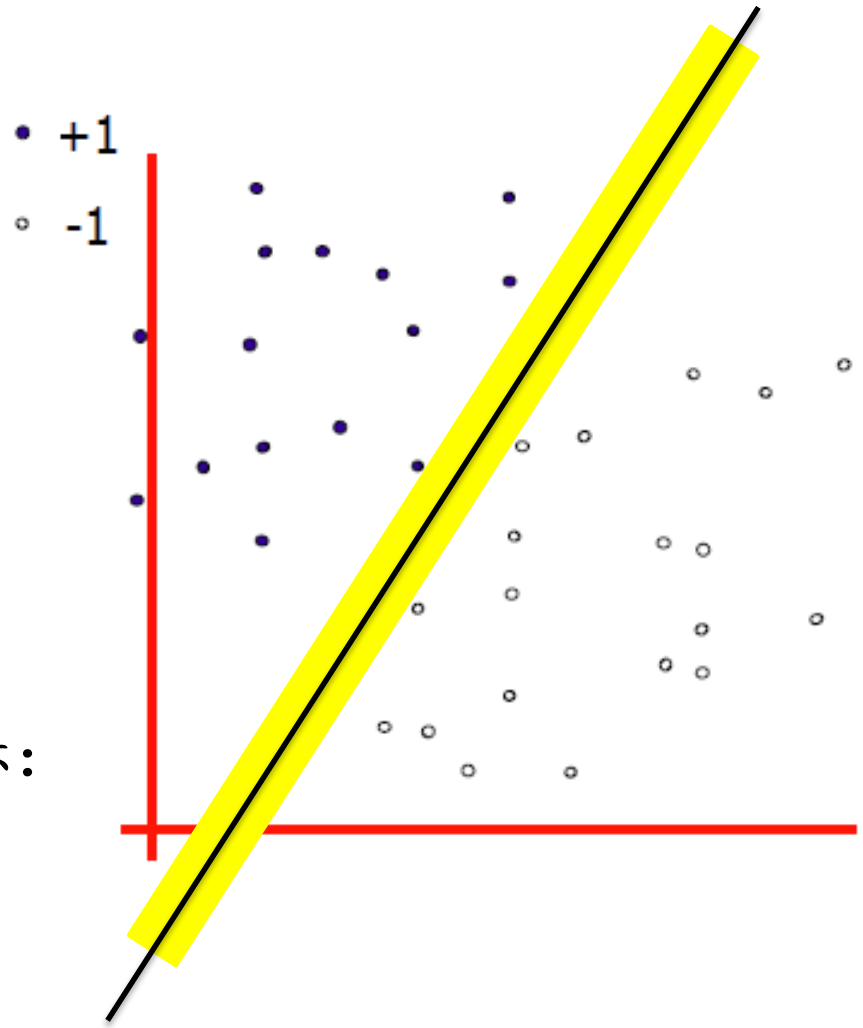
- 输入: 训练数据$D = \{(x_i, y_i)\}_{i=1}^{N}$, 学习步长$\eta > 0$, 边距$\tau > 0$
- 输出: 感知机模型$\mathbf{w}, b$

1. $(\mathbf{w}, b) \leftarrow$ 随机值
2. $R \leftarrow \max_i |\mathbf{x}_i|$
3. **repeat**
4. 选取数据$(x_i, y_i) \in D$
5. **if** $y_i f(\mathbf{x}_i) \leq \tau$ **then**
6. $\mathbf{w} \leftarrow \mathbf{w} + \eta y_i \mathbf{x}_i$
7. $b \leftarrow b + \eta R y_i$
8. **end if**
9. **until** convergence

SVM：最大边距分类器



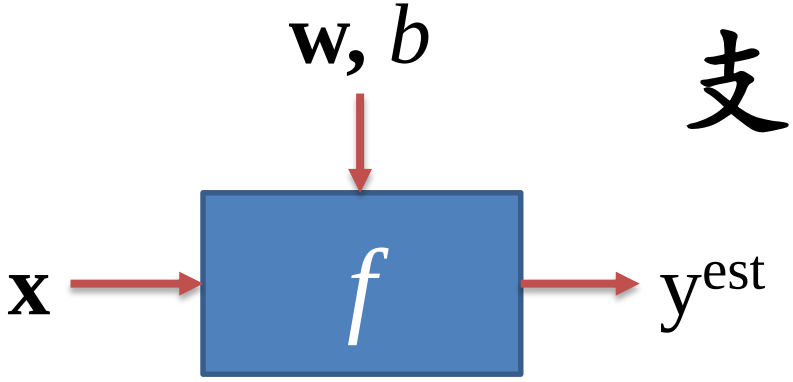
$$f(\mathbf{x}; \mathbf{w}, b) = \text{sgn}(\langle \mathbf{w}, \mathbf{x} \rangle + b)$$



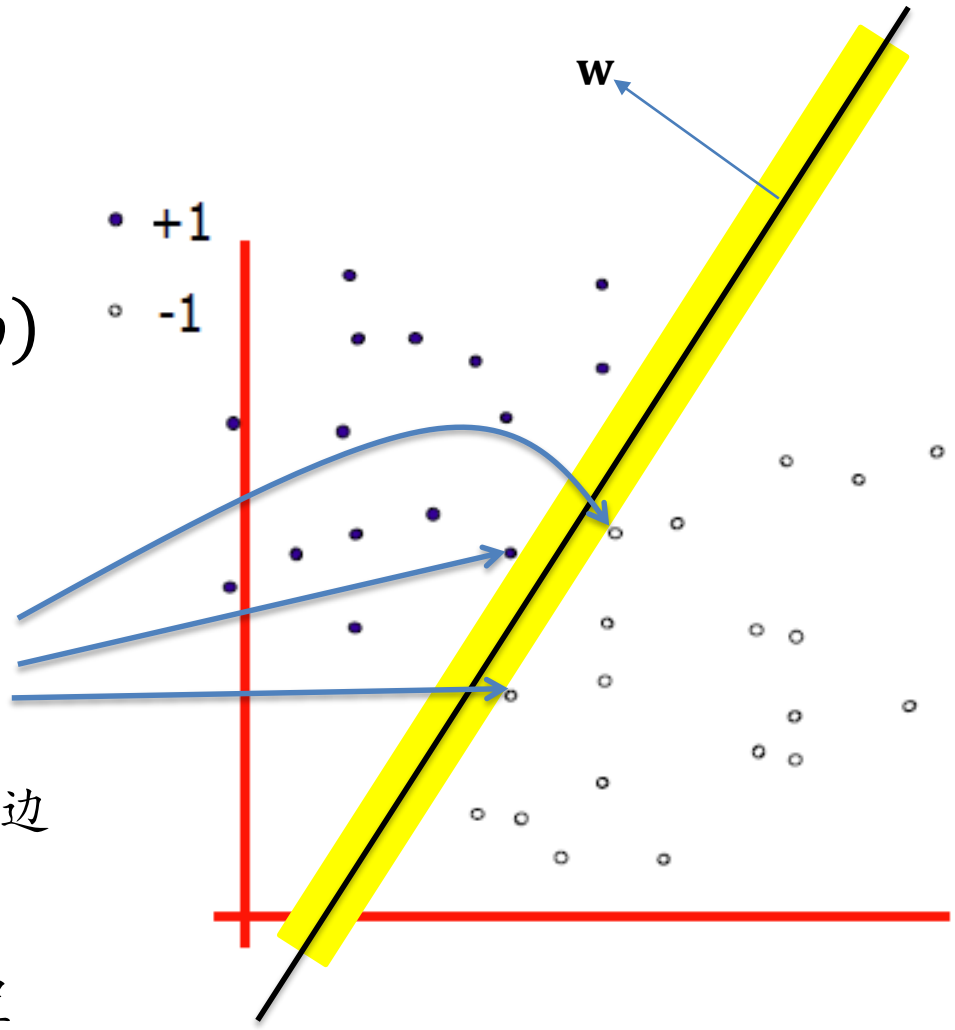
不同的模型有不同大小的边距。

SVM (support vector machine)的目标：
最大化分类边距

支持向量



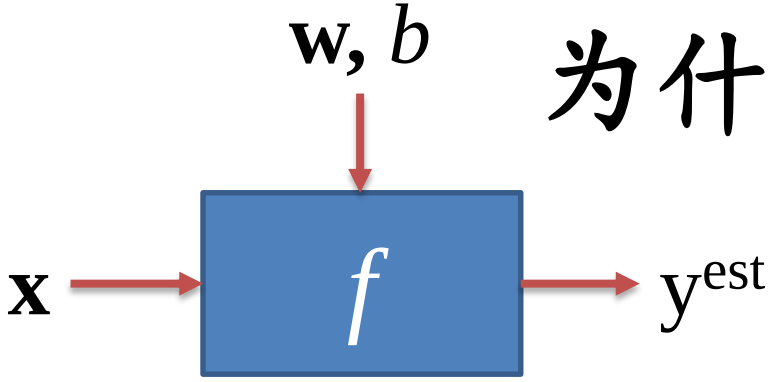
$$f(\mathbf{x};\mathbf{w},b) = \text{sgn}(\langle \mathbf{w},\mathbf{x}\rangle + b)$$



支持向量(support vector):位于分类边距上的数据点

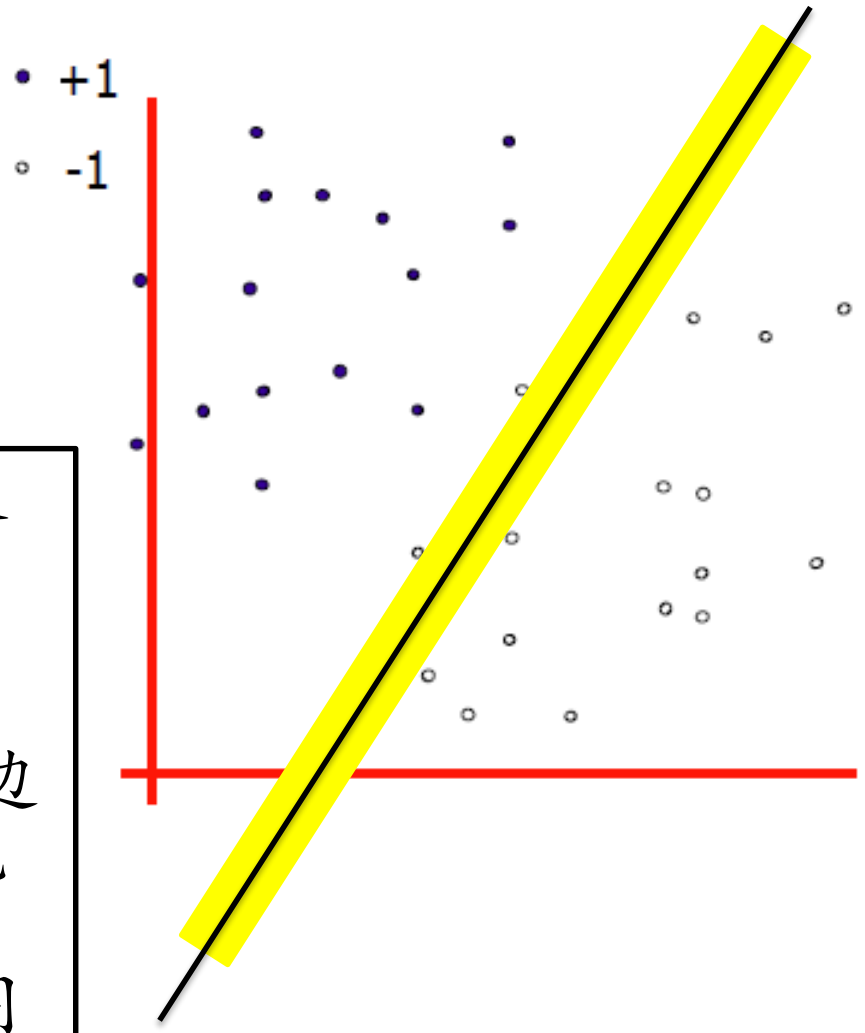
支持向量决定了边距的大小和位置
给定训练数据集合，w的方向决定了边距大小

为什么最大化边距？

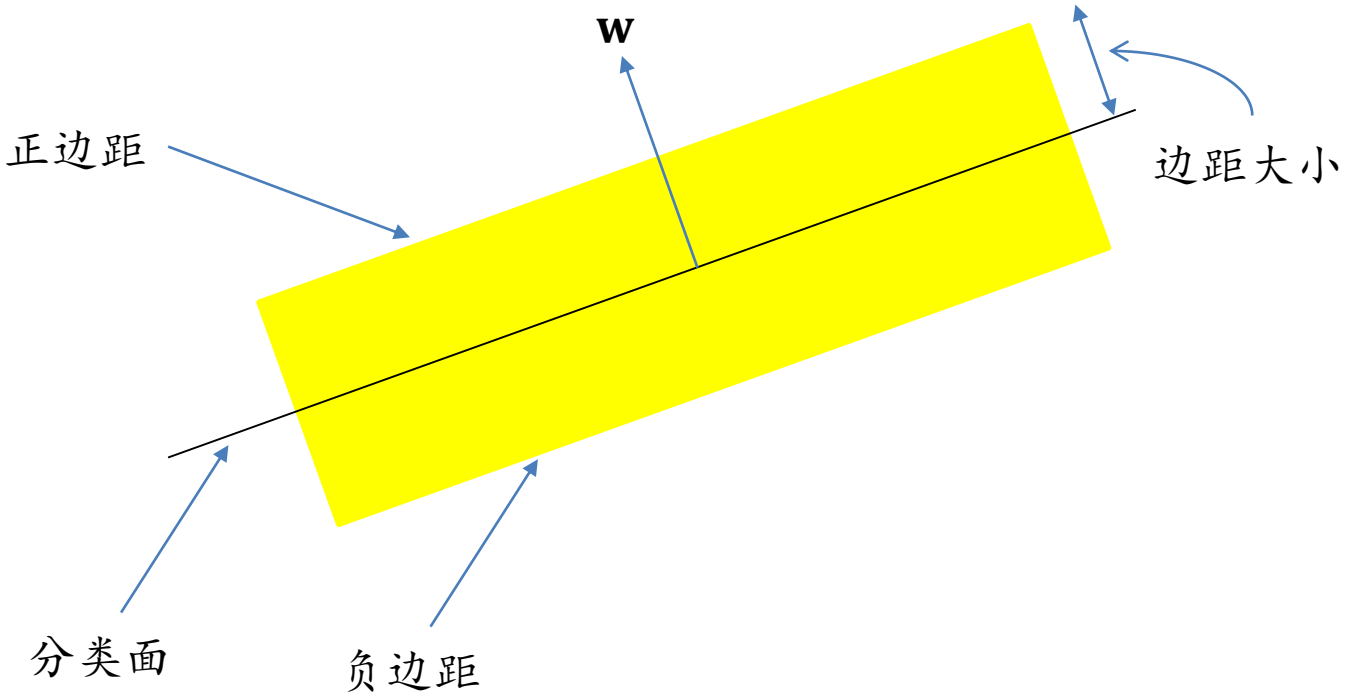


$$f(\mathbf{x};\mathbf{w},b) = \mathrm{sgn}(\langle \mathbf{w},\mathbf{x}\rangle + b)$$

- 直觉上，比其它分类界面更不容易出错
  - 分类面的微小扰动不会导致错误发生
- 理论上，VC理论证明最大边距分类器具有较好的泛化能力(generalization ability)
- 实际应用：SVM在很多应用上取得了很好的效果

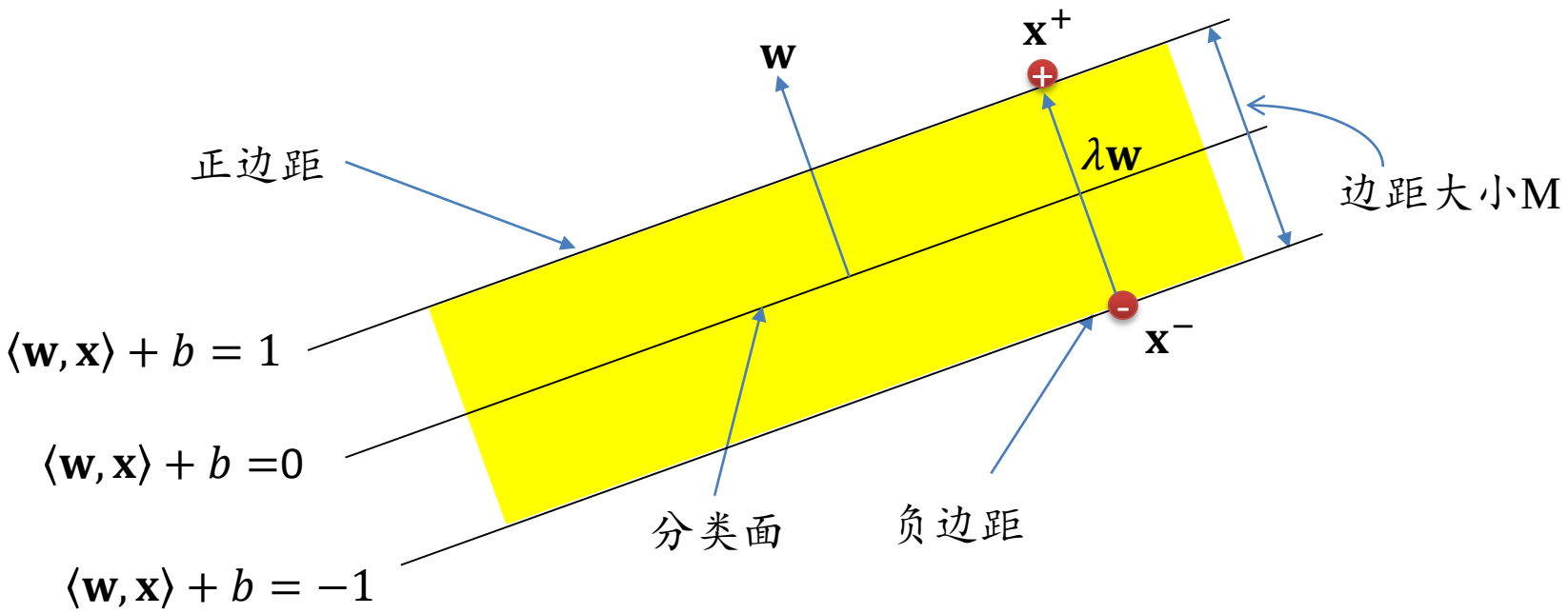


边距的表示



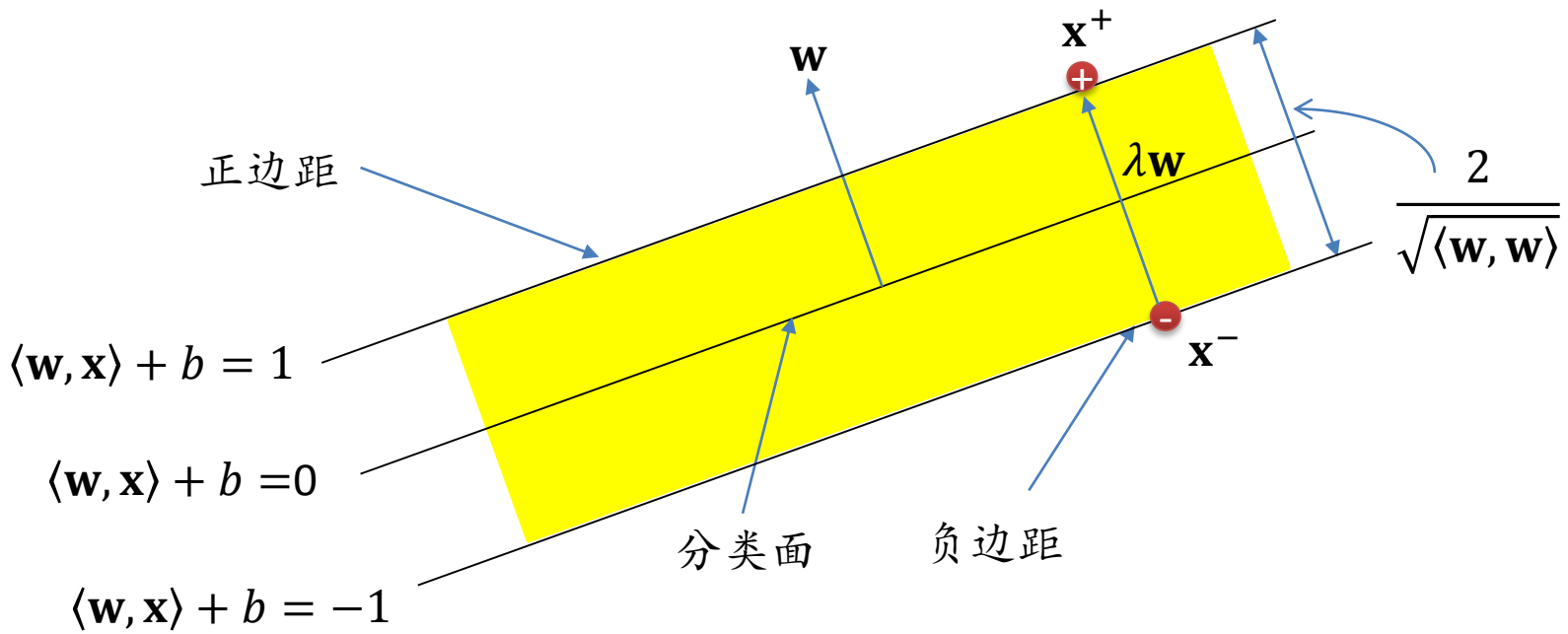
如何（在高维空间中）计算边距的大小？

边距的表示



- $\langle \mathbf{w}, \mathbf{x}^+ \rangle + b = 1, \langle \mathbf{w}, \mathbf{x}^- \rangle + b = -1 \Rightarrow \langle \mathbf{w}, \mathbf{x}^+ - \mathbf{x}^- \rangle = 2$
  - $\mathbf{x}^+ = \mathbf{x}^- + \lambda \mathbf{w}$
  - $\lambda \langle \mathbf{w}, \mathbf{w} \rangle = 2 \Rightarrow \lambda = \frac{2}{\langle \mathbf{w}, \mathbf{w} \rangle}$
- $\mathrm{M} = |\mathbf{x}^+ - \mathbf{x}^-| = |\lambda \mathbf{w}| = \frac{2}{\langle \mathbf{w}, \mathbf{w} \rangle} \sqrt{\langle \mathbf{w}, \mathbf{w} \rangle} = \frac{2}{\sqrt{\langle \mathbf{w}, \mathbf{w} \rangle}}$

边距的表示



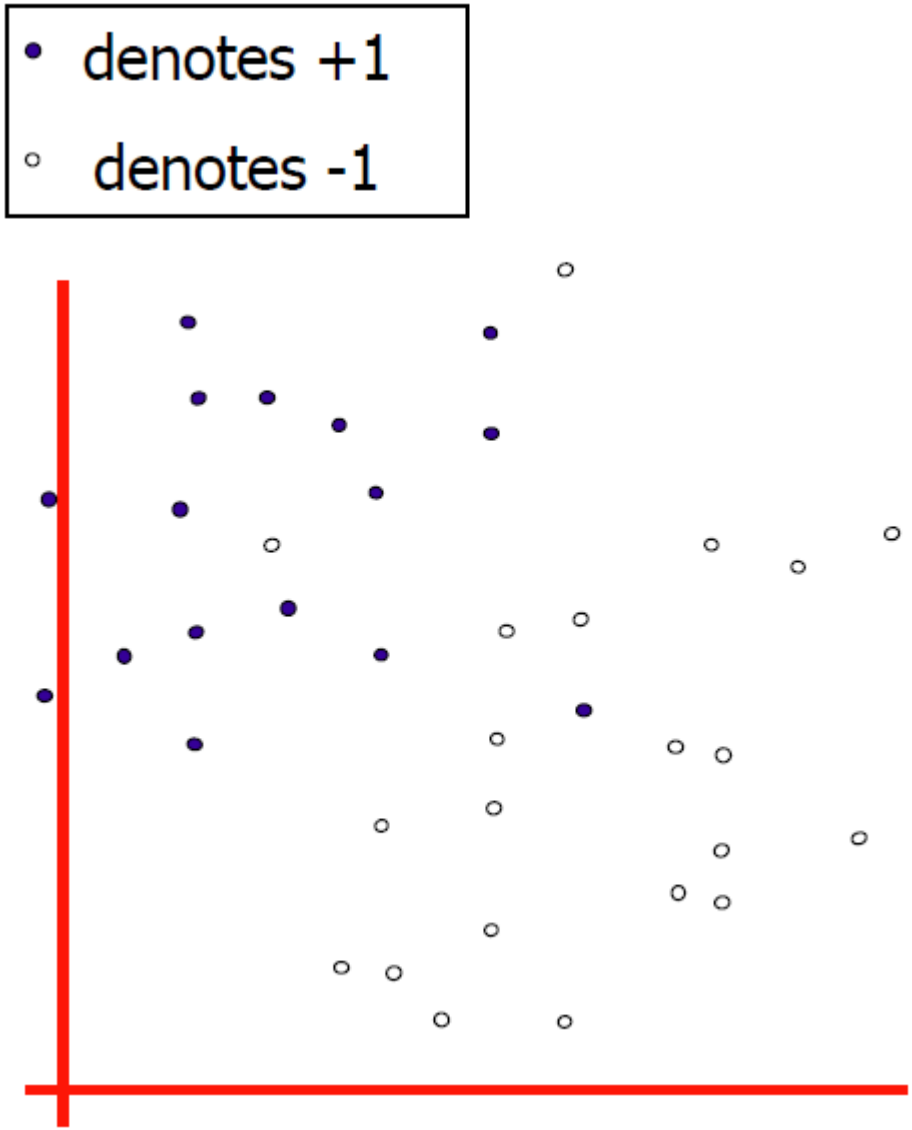
- SVM学习过程归结为寻找合适的$\mathbf{w}$和b
  - 所有的训练数据都在正确的分类区域
    - $y_i(\langle \mathbf{w},\mathbf{x}_i\rangle + b) \geq 1$, 其中$y_i \in \{+1,-1\}$
  - 最大化边距
    - $\max \frac{2}{\sqrt{\langle \mathbf{w},\mathbf{w}\rangle}} \Leftrightarrow \min |\mathbf{w}|^2$

SVM优化问题

- 优化目标: $\min_{\mathbf{w},b} |\mathbf{w}|^2$
- 约束条件: $y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1$, for i=1, … ,N

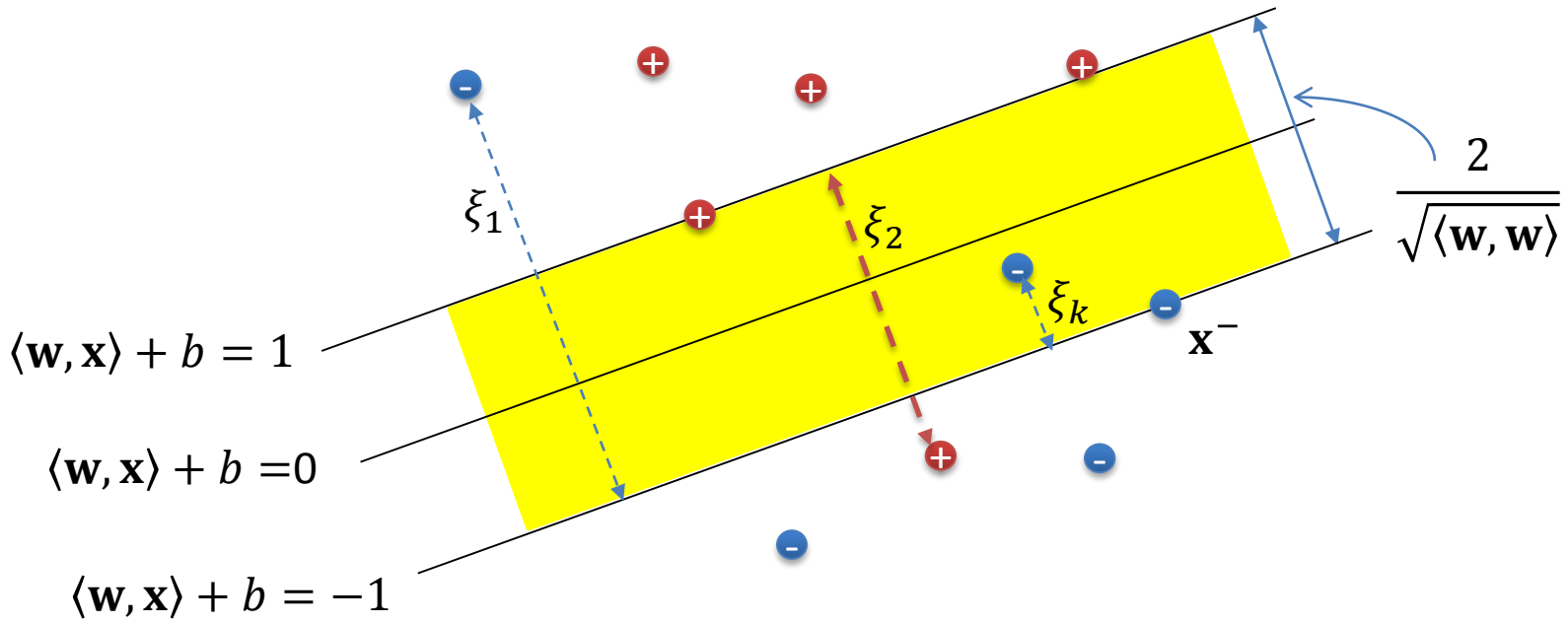
- 二次规划(quadratic programming, QP)问题
  - 二次的目标函数，N个线性约束条件
  - 有成熟的解法
- 只适应训练数据可以完全可分的情况！

上述理想模型存在的问题
——训练数据不可分



- 不存在一个分类面使得训练数据能够被完美分开
- 解决方案：容忍错误的发生
  - $\min_{\mathbf{w},b} |\mathbf{w}|^2 +$ 错误惩罚，
  - 错误惩罚：出错的数据点与其正确位置的距离
  - 边距不再是硬性限制条件(软边距)

软边距



- 优化目标: $\min_{\mathbf{w},b} |\mathbf{w}|^2 + C \cdot \sum_{i=1}^{N} \xi_i$
- 约束条件: $y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i, \xi_i \geq 0,$ for i=1, … ,N
- C：权衡边距大小与错误容忍度
  - C变大：牺牲边距，减少训练集合上的错误

优化目标求解

将b融入$\mathbf{w}$和$\mathbf{x}$

$$\min_{\mathbf{w},b} |\mathbf{w}|^2 + C \cdot \sum_{i=1}^{N} \xi_i$$

原问题：

$$\text{s.t.} \quad y_i \langle \mathbf{w}, \mathbf{x}_i \rangle \geq 1 - \xi_i,$$
$$\xi_i \geq 0, \text{ for } i=1, \ldots, N$$

$$\max_{\alpha_1,\cdots,\alpha_N} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{k=1}^{N} \sum_{l=1}^{N} \alpha_k \alpha_l y_k y_l \langle \mathbf{x}_k, \mathbf{x}_l \rangle$$

对偶问题：

$$\text{s.t.} \quad \forall i: 0 \leq \alpha_i \leq C, \sum_{i=1}^{N} y_i \alpha_i = 0$$

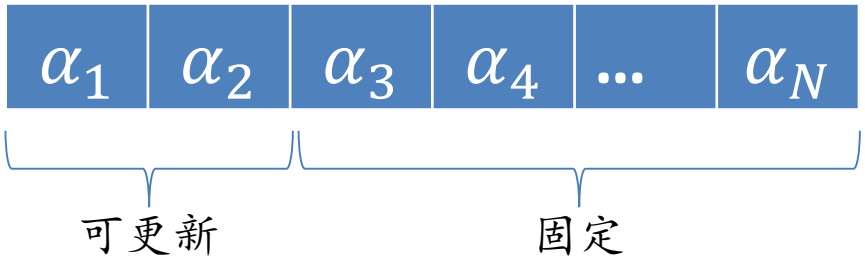
$\alpha_i > 0$: $\mathbf{x}_i$为支持向量

求解对偶问题

$$\max_{\alpha_1,\cdots,\alpha_N} \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{k=1}^{N}\sum_{l=1}^{N} \alpha_k \alpha_l y_k y_l \langle \mathbf{x}_k, \mathbf{x}_l \rangle$$

$$\text{s.t.} \quad \forall i: 0 \le \alpha_i \le C, \sum_{i=1}^{N} y_i \alpha_i = 0$$

- SMO (sequential minimal optimization)
  - 多维变量优化→一系列单变量优化问题
  - 一次放开两个(注意$\sum_{i=1}^{N} y_i \alpha_i = 0$)，固定其他



$$\min_{\alpha_1} \alpha_1^2 + b\alpha_1 + c$$

$$\text{s.t. } 0 \le \alpha_1 \le C$$

  - 求解后更新α_1和α_2
  - 选择另外两个变量继续求解，重复。

学习到的分类函数

- $f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle$, 其中$\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i$
  - b融合在$\mathbf{w}$中: $\mathbf{x}$→($\mathbf{x}$, 1); $\mathbf{w}$→($\mathbf{w}$, b)
- 分类决策

$$\text{sgn}(\langle \mathbf{w}, \mathbf{x} \rangle)$$

- 关于参数C
  - 可以通过交叉验证或者验证集合确定C的值。
  - C值越大，越可能出现过拟合。

损失函数

- 很多机器学习/数据挖掘算法都可以归结为优化损失函数
- 步骤1：损失函数的定义

$$L(f) = \sum_{i=1}^{N} l(f(\mathbf{x}_i), y_i) + \lambda\Omega(f)$$

权衡参数

在每一个训练样本上的“损失”

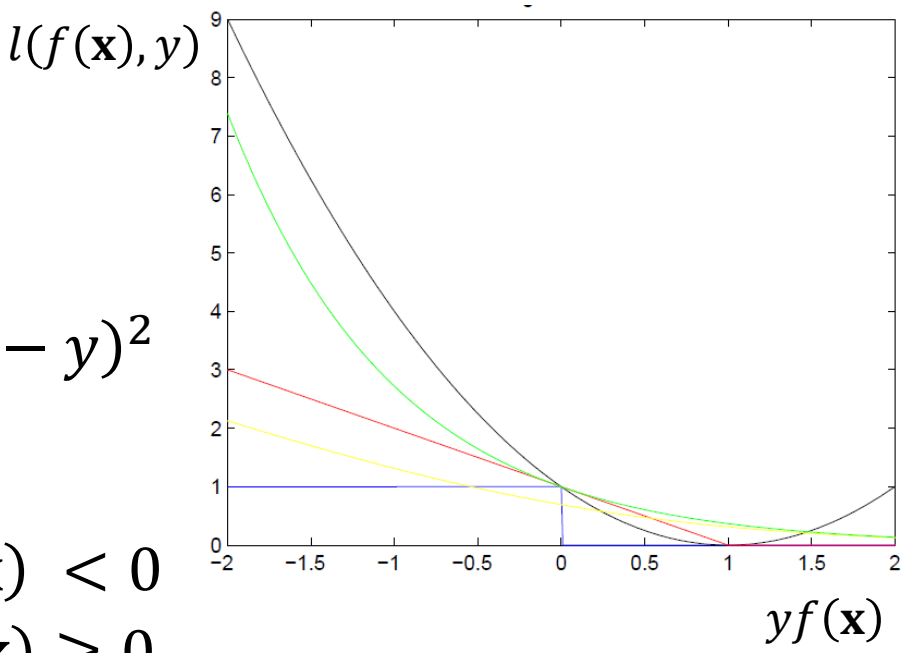
对复杂函数f进行惩罚

- 步骤2：损失函数优化

$$f^* = \text{argmin}_{f \in F} L(f)$$

- 优点
  - 目标定义和优化算法分开
  - 是分析/提出新算法的有效手段

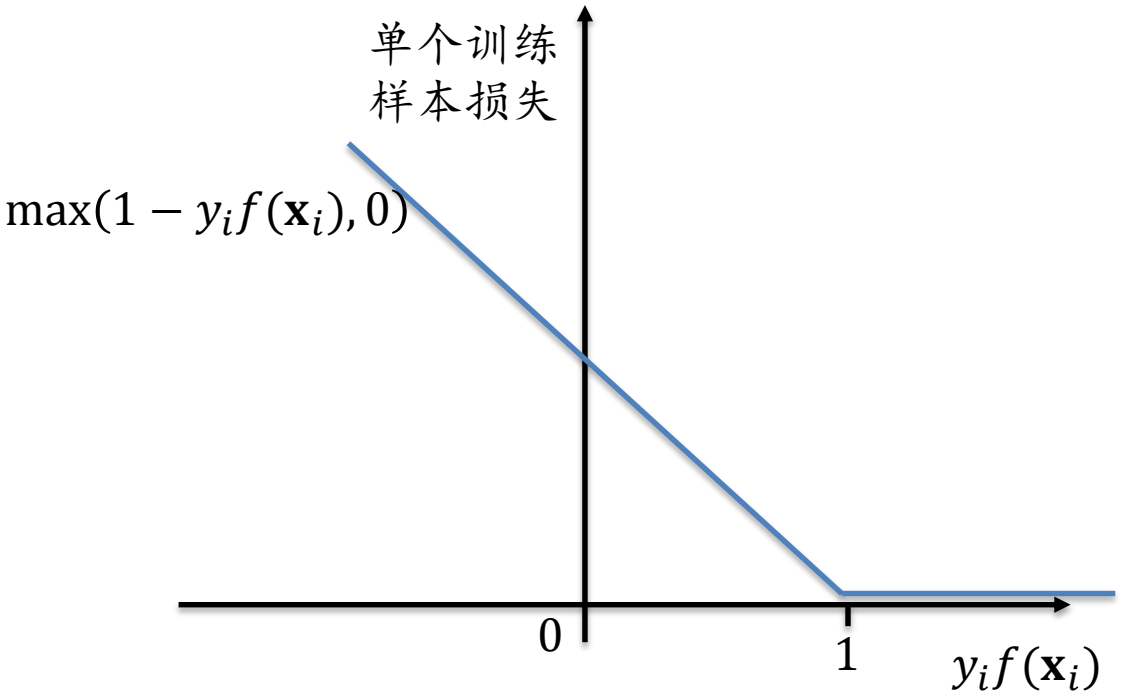
常用的$l(f(\mathbf{x}), y)$



- 回归
  - Square loss: $l(f(\mathbf{x}), y) = (f(\mathbf{x}) - y)^2$
- 二值分类
  - 0-1 loss: $l(f(\mathbf{x}), y) = \begin{cases} 1, & yf(\mathbf{x}) < 0 \\ 0, & yf(\mathbf{x}) \geq 0 \end{cases}$
  - Hinge loss: $l(f(\mathbf{x}), y) = \max(0, 1 - yf(\mathbf{x}))$
  - Exp loss: $l(f(\mathbf{x}), y) = \exp(-yf(\mathbf{x}))$
  - Logistic loss: $l(f(\mathbf{x}), y) = \log(1 + \exp(-yf(\mathbf{x})))$

SVM损失函数

$$L(\mathbf{w}) = \sum_{i=1}^{N} \max(1 - y_i f(\mathbf{x}_i), 0) + \lambda |\mathbf{w}|^2$$



- 可以证明，当$\lambda = \frac{2}{C}$时，最小化$L(\mathbf{w})$与线性SVM学习等价

提纲

- 背景介绍
- 常用的监督学习算法
  - 决策树
  - 支持向量机/Perceptron
  - AdaBoost
- 总结

监督学习算法分类

任务

模型函数

	(二值)分类 Y为类别集合	回归 Y为实数集合	结构预测/排序 Y为结构类型/排列
线性函数 **f(x)=⟨w, x⟩**	Perceptron, SVM	Ridge regression SVR	Structured SVM, Ranking SVM
树	Decision tree	Regression tree	LambdaMART
神经网络	神经网络	神经网络	RankNet
叠加函数 **f(x)=$\sum_t h_t(x)$** **(additive model)**	AdaBoost	Boosted Regression Tree	RankBoost

Boosting

- 定义： general method of converting rough rules of thumb into highly accurate prediction rule.
  - 组合弱分类器可以得到强分类器
  - 不同的弱分类器可以通过不同的算法、特征、数据训练而得到
  - 弱分类器的预测精度强于随机预测 (rough rules of thumb)

对Boosting的研究

- [Freund & Schapire '95]:
  - introduced "AdaBoost" algorithm
  - strong practical advantages over previous boosting algorithms
- experiments using AdaBoost:

[Drucker & Cortes '95]

[Jackson & Craven '96]

[Freund & Schapire '96]

[Quinlan '96]

[Breiman '96]

[Schapire & Singer '98]

[Maclin & Opitz '97]

[Bauer & Kohavi '97]

[Schwenk & Bengio '98]

[Dietterich '98]

⋮

- continuing development of theory and algorithms:

[Schapire, Freund, Bartlett & Lee '97]

[Breiman '97]

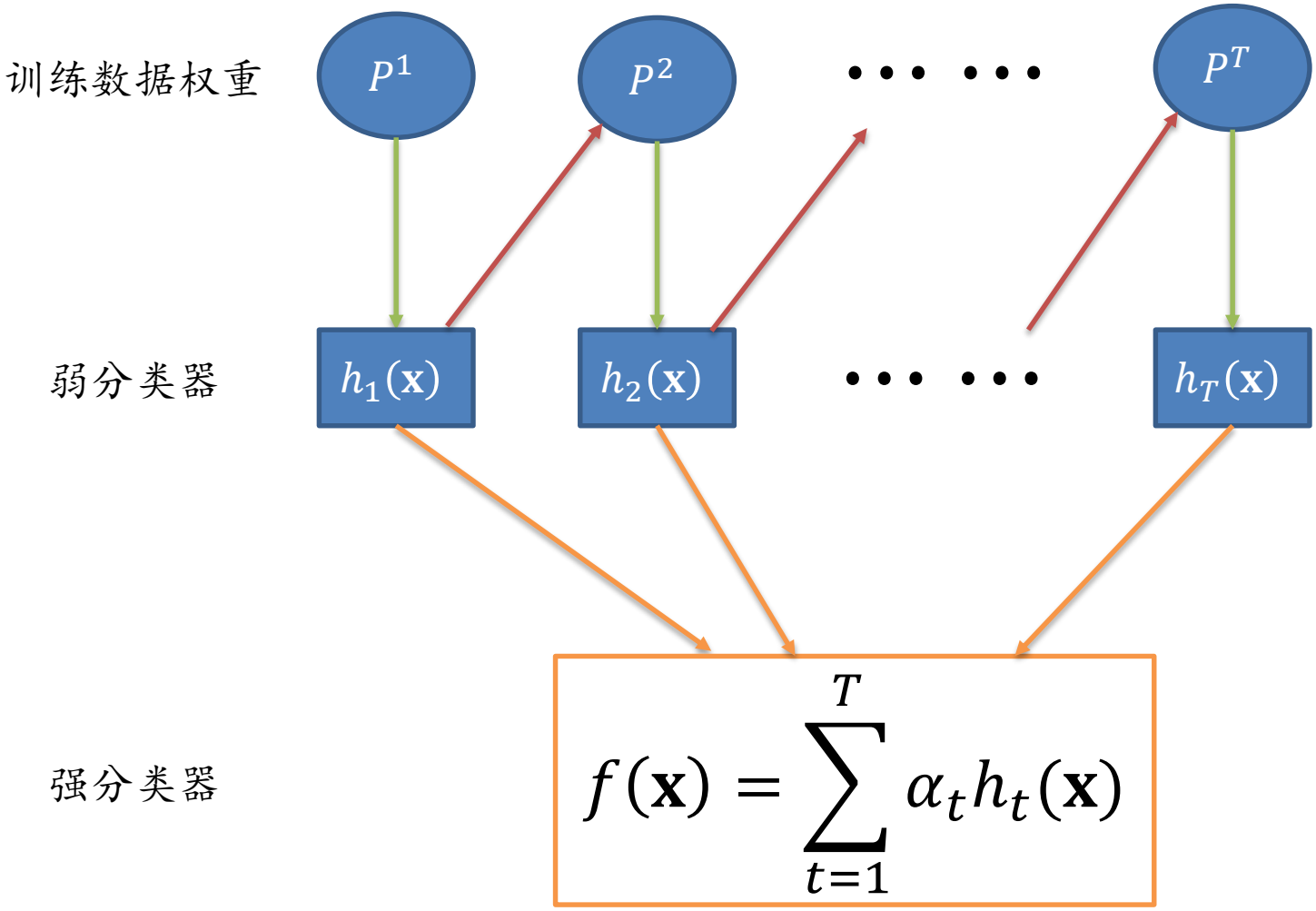
[Grove & Schuurmans '98]

[Schapire & Singer '98]

[Mason, Bartlett & Baxter '98]

[Friedman, Hastie & Tibshirani '98]

AdaBoost的基本训练过程



AdaBoost算法

- AdaBoost: **Ada**ptive **Boost**ing

- 输入: 训练数据$D=\{(\mathbf{x}_i,y_i)\}_{i=1}^N, y_i\in\{+1,-1\}$
- 输出: 分类模型$F(\mathbf{x})$

1. 初始化训练数据的权重$P^1=\left\{\frac{1}{N},\cdots,\frac{1}{N}\right\}$
2. **for** t=1 **to** T
3. 基于权重P^t训练弱分类器$h_t(\mathbf{x})$
4. 计算弱分类器的分类错误率$\epsilon_t=\sum_{i=1}^N P^t(i)[\![h_t(\mathbf{x}_i)\neq y_i]\!]$
5. 计算弱分类器的权重$\alpha_t=\frac{1}{2}\ln\frac{1-\epsilon_t}{\epsilon_t}$
6. 重新估计训练$P^{t+1}(i)=\frac{P^t(i)\exp\{-\alpha_t y_i h_t(\mathbf{x}_i)\}}{Z_{t+1}}$
7. **end for**
8. **return** $F(\mathbf{x})=\mathrm{sgn}\big(f_T(\mathbf{x})\big)=\mathrm{sgn}(\sum_{t=1}^T\alpha_t h_t(\mathbf{x}))$

开始时所有数据一样重要

可为任何分类器，$h_t(\mathbf{x})\in\{+1,-1\}$

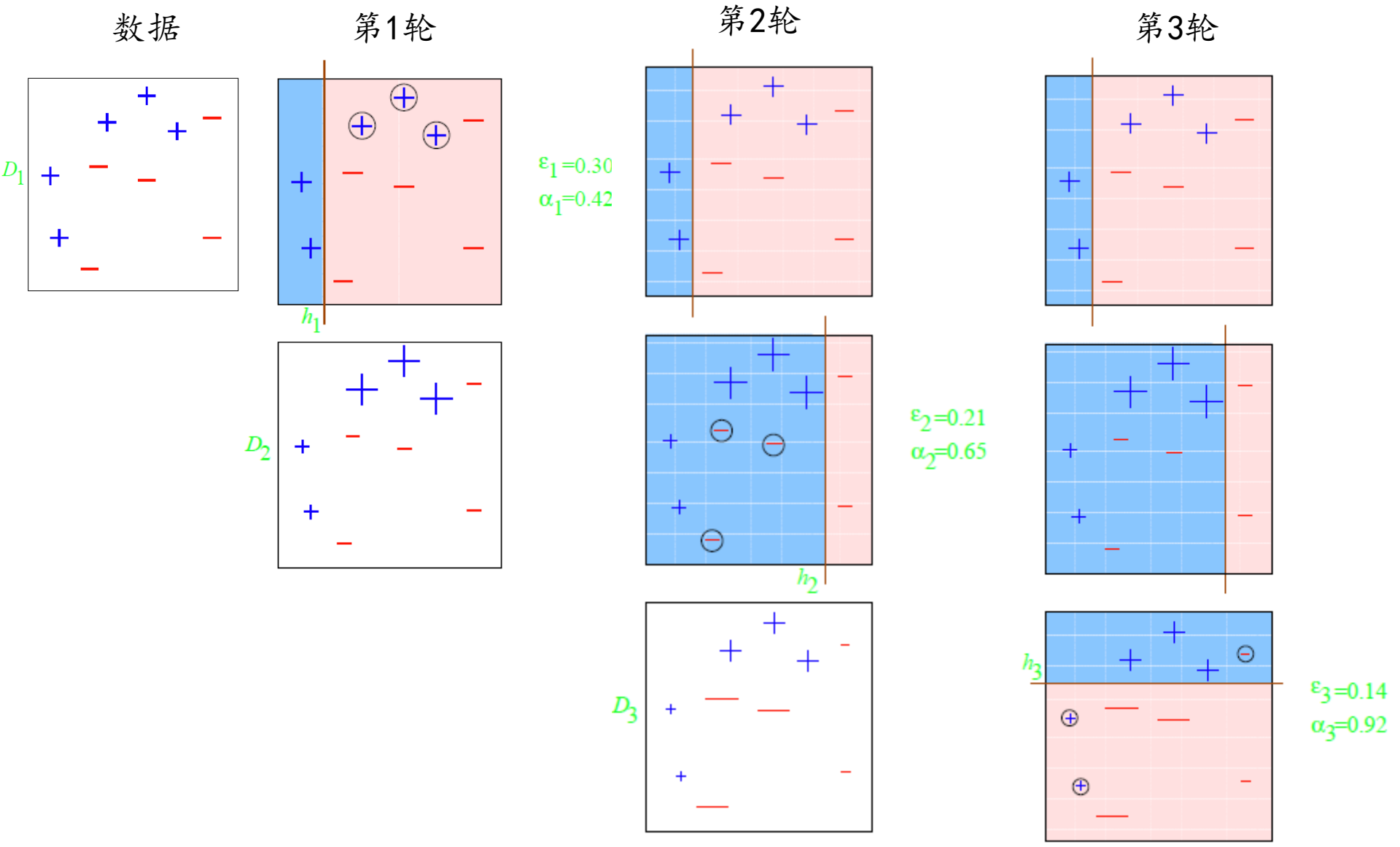
弱分类器在P^t意义下的错误率ϵ_t<0.5

分类器越强 α_t 越大。注意：$\epsilon_t<0.5$

对$(\mathbf{x}_i,y_i)$预测不准$P^{t+1}(i)$增大，下一轮重点关注本轮出错的数据点！

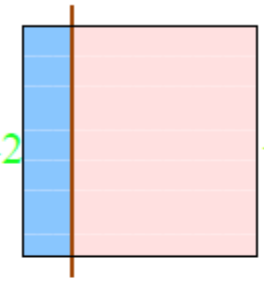
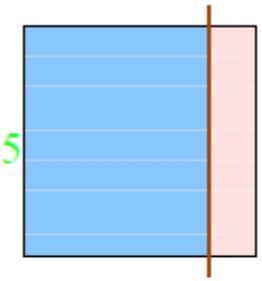
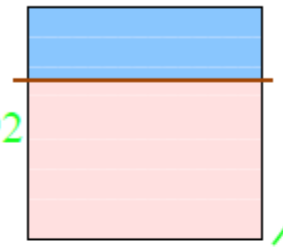
$f(\mathbf{x})$为加性函数(additive model)

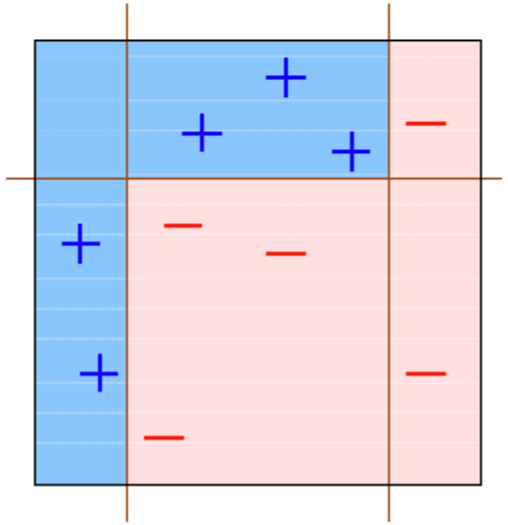
算法运行举例


数据
第1轮
第2轮
第3轮
D_1
h_1
D_2
$\varepsilon_1=0.30$
$\alpha_1=0.42$
h_2
D_3
$\varepsilon_2=0.21$
$\alpha_2=0.65$
h_3
$\varepsilon_3=0.14$
$\alpha_3=0.92$

学习到的分类器

H_{final}

$= \text{sign}\left(0.42 \right.$  $+ 0.65$  $+ 0.92$  $\left. \right)$

$=$ 

AdaBoost在训练集上错误率

令$\epsilon_t = \frac{1}{2} - \gamma_t (0 < \gamma_t \leq \frac{1}{2})$,则AdaBoost算法得到的分类模型$\mathrm{sgn}\big(f_T(\mathbf{x})\big)$在训练数据上的错误率

$$E = \frac{1}{N}\sum_{i=1}^{N} [\![f_T(\mathbf{x}_i) \neq y_i]\!]$$

$$\leq \prod_{t=1}^{T} \left[2\sqrt{\epsilon_t(1-\epsilon_t)} \right]$$

$$= \prod_{t=1}^{T} \sqrt{1 - 4\gamma_t^2}$$

$$\leq \exp\left\{ -2\sum_{t=1}^{T} \gamma_t^2 \right\}$$

随着t增大，训练集合上的错误率逐渐减少

证明步骤

- 令$f_T(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$
- Step 1: 展开最后的数据权重分布

$$P^{T+1}(i) = P^{T}(i)\frac{\exp\{-\alpha_T y_i h_T(\mathbf{x}_i)\}}{Z_T}$$

$$= \cdots = \frac{1}{N}\frac{\exp\{-y_i \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}_i)\}}{\prod_{t=1}^{T} Z_t}$$

- Step 2：训练集错误率$E \le \prod_{t=1}^{T} Z_t$

$$E = \frac{1}{N}\sum_{i=1}^{N} [\![f_T(\mathbf{x}_i) \neq y_i]\!] \le \frac{1}{N}\sum_{i=1}^{N} \exp\{-y_i f_T(\mathbf{x}_i)\}$$

$$= \sum_{i=1}^{N} P^{T+1}(i) \prod_{t=1}^{T} Z_t = \prod_{t=1}^{T} Z_t$$

证明步骤

- Step 3:$Z_t = 2\sqrt{\epsilon_t(1-\epsilon_t)}$

$$Z_t = \sum_{i=1}^{N} P_{t-1}(i)\exp\{-\alpha_t y_i h_t(\mathbf{x}_i)\}$$

$$= \sum_{i:y_i=h_t(\mathbf{x}_i)} P_{t-1}(i)\exp\{-\alpha_t\} + \sum_{i:y_i\neq h_t(\mathbf{x}_i)} P_{t-1}(i)\exp\{\alpha_t\}$$

$$= \epsilon_t \exp\{-\alpha_t\} + (1-\epsilon_t)\exp\{\alpha_t\}$$

$$= \epsilon_t \exp\left\{-\frac{1}{2}\ln\frac{1-\epsilon_t}{\epsilon_t}\right\} + (1-\epsilon_t)\exp\left\{\frac{1}{2}\ln\frac{1-\epsilon_t}{\epsilon_t}\right\}$$

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

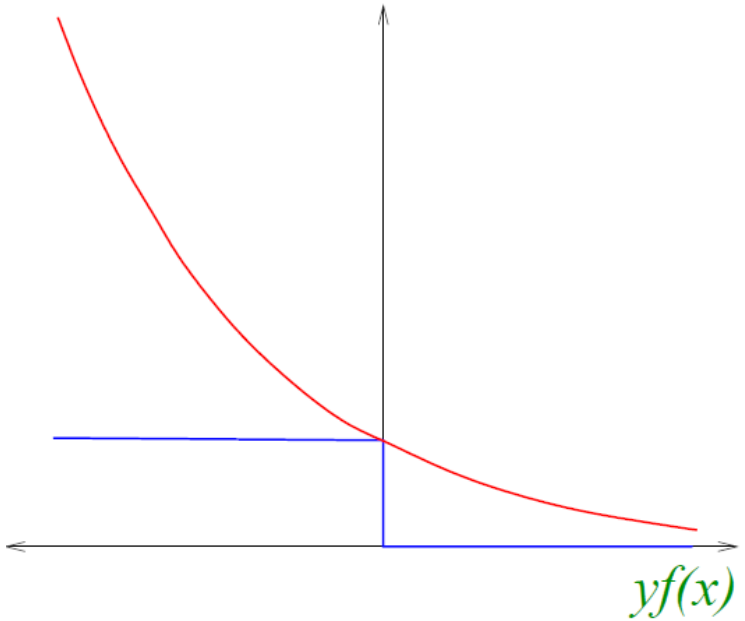
AdaBoost的损失函数

- 指数损失函数

$$E_{0-1} = \frac{1}{N}\sum_{i=1}^{N}[\![f_T(\mathbf{x}_i) \neq y_i]\!]$$

$$\leq \frac{1}{N}\sum_{i=1}^{N}\exp\{-y_i f_T(\mathbf{x}_i)\} = L(f_T)$$

- 指数损失是0-1损失的上界(upper bound)
- 在训练的每一个步骤，AdaBoost贪心选择h_t和α_t最小化损失函数
- $f_{t+1} = f_t + \alpha_t h_t$，加性模型，不再改变$f_t$



关于Boosting的其他话题

- 为多类分类、回归、排序等任务设计的Boosting算法
- Boosting为一种算法框架，其弱分类器可以为任何其他分类模型
  - MART (回归)
  - LambdaMART (排序)
- Boosting可以看成函数空间的梯度下降，由此可以导出多种不同的Boosting算法

  Mason et al., Boosting Algorithms as Gradient Descent, NIPS 1999.

  Friedman, J. H. Greedy Function Approximation: A Gradient Boosting Machine. 1999.

提纲

- 背景介绍
- 常用的监督学习算法
  - 决策树
  - 支持向量机/Perceptron
  - AdaBoost
- 结构化输出预测
- 总结

监督学习算法分类

任务

模型函数	(二值)分类 Y为类别集合	回归 Y为实数集合	结构预测/排序 Y为结构类型/排列
线性函数 $f(x)=\langle w, x\rangle$	Perceptron, SVM	Ridge regression SVR	Structured SVM, Ranking SVM
树	Decision tree	Regression tree	LambdaMART
神经网络	神经网络	神经网络	RankNet
叠加函数 $f(x)=\sum_t h_t(x)$ **(additive model)**	AdaBoost	Boosted Regression Tree	RankBoost

结构化输出问题描述

$$y = f(\mathbf{x}), y \in \mathcal{Y}$$

- 分类或者回归：$\mathcal{Y} = \{1, -1\}$；$\mathcal{Y} = \{$红, 黄, 蓝$\}$；$\mathcal{Y} = R$
- 如何解决更加复杂的预测问题
  - 多标签多类别分类 (multi-label multi-class classification) $y \subseteq \mathcal{Y}$
  - 序列标注$y \in \mathcal{Y}^{|\mathbf{x}|}$, 其中$\mathcal{Y}$为所有可能的词标签

    词性标注：$\mathbf{x}$ =“This is a book”, y = “PRP VB AT NN”

    实体识别：$\mathbf{x}$ =“This is a text book”, y = “O O O B E”
  - 排序(将在排序学习中介绍)

  $\mathbf{x} = \{d_1, d_2, \cdots, d_N\}, y \in \Pi_N$，其中$\Pi_N$为N个元素的排列集合

简单做法：简化为更加细粒度的分类问题

- 简单归结为分类问题
  - 假设模型为：对于每一个$x_i \in \mathbf{x}, y_i = f(x_i)$
  - 学习过程：将训练数据$(\mathbf{x}, \mathbf{y})$打散为多个(x_i, y_i)的形式

    举例：$\mathbf{x}$="This is a text book", y = "O O O B E"

    =>{(this, O), (is, O), (a, O), (text, B), (book, E)}
  - （用SVM或者其它任何算法）训练（多类）分类模型f
  - 预测过程：对于新的输入$\mathbf{x}$，对其每一个位置i的标签用$f(x_i)$进行预测

    举例：$\mathbf{x}$="That is a blue pen", y = ？

$y_1 = f(\text{that}), y_2 = f(\text{is}), y_3 = f(\text{a}), y_4 = f(\text{blue}), y_5 = f(\text{pen})$

$y = (y_1 y_2 y_3 y_4 y_5)$

注意：在抽取每一个细粒度（位置）上的特征时，可以参考整体的$\mathbf{x}$，即$f(that) = \langle \mathbf{w}, \boldsymbol{\phi}(1, \text{That is a blue pen}) \rangle$

简单做法：简化为更加细粒度的分类问题

- 简单归结为分类问题

  $\mathbf{x}$ =“That is a blue pen”, $y =$ ？

  $\mathrm{y}_1 = f(\mathrm{that}), \mathrm{y}_2 = f(\mathrm{is}), \mathrm{y}_3 = f(\mathrm{a}), \mathrm{y}_4 = f(\mathrm{blue}), \mathrm{y}_5 = f(\mathrm{pen})$

  $y = (y_1 y_2 y_3 y_4 y_5)$

- 可能存在的问题
  - 产生不合法的结果：$\mathrm{y}_1 = O, \mathrm{y}_2 = E, \mathrm{y}_3 = O, \mathrm{y}_4 = B, \mathrm{y}_5 = E$

    （在左边不出现B的情况下，不能出现E）
  - 上述模型无法利用或者学习到这一类信息
- 原因
  - 分类模型 f 假设每一个训练/预测的对象(位置)都是独立的。例如：可以同时预测同一个句子不同位置上的标签
  - 在预测一个句子不同位置上的标签时，不考虑其它位置上的标签信息（注意：已经考虑到了其它位置上的输入单词信息）。因此当标签存在依赖关系时，简化的分类模型学习不到

结构化输出预测模型定义

- 解决办法：对任意输入**x**进行整体预测
- 怎么将**x**进行整体预测？

  $f(\mathbf{x}) = \text{argmax}_{y \in \mathcal{Y}_\mathbf{x}} F(\mathbf{x}, \mathrm{y}) = \text{argmax}_{y \in \mathcal{Y}_\mathbf{x}} \langle \mathbf{w}, \Psi(\mathbf{x}, y) \rangle$

  - $F(\mathbf{x}, \mathrm{y}) \in R$: $F(\mathbf{x}, \mathrm{y})$越大，表明$F$判断y是**x**的正确的结果（所有位置上标签都正确）的可能性越大, confidence score
  - $\mathcal{Y}_\mathbf{x}$: 根据x的输入生成的所有的合法的输出（未必正确）
  - 采用**比较**方式选取最优预测（F的绝对值不重要！传统的分类问题看f的绝对值）
- 优点
  - $\mathcal{Y}_\mathbf{x}$中的候选全都合法：不会产生非法输出
  - **x**进行整体预测：可以考虑更多信息label的依赖信息，预测更加准确

结构化输出预测模型定义

$$f(\mathbf{x}) = \text{argmax}_{y \in \mathcal{Y}_\mathbf{x}} F(\mathbf{x}, \mathrm{y}) = \text{argmax}_{y \in \mathcal{Y}_\mathbf{x}} \langle \mathbf{w}, \Psi(\mathbf{x}, y) \rangle$$

- 缺点
  - 需要首先生成所有 $\mathcal{Y}_\mathbf{x}$ 并且 $\mathcal{Y}_\mathbf{x}$ 可能含有很多个候选：根据不同的应用可以改进优化
  - $F(\mathbf{x}, \mathrm{y})$的定义比较复杂：同时将输入和输出编码为特征
  - argmax求解复杂度可能比较高（注意：此处为预测时复杂度，在实际应用中对在线效率影响较大），最差的情况下需要遍历$\mathcal{Y}_\mathbf{x}$

如何训练？

- 仍然将其转化为二值分类问题（我们可利用的机器学习模型非常有限）
  - 第一步：数据转换

    举例：($\mathbf{x}$ ="This is a text book", y^+ = "O O O B E")

    得到一个正样本(($\mathbf{x}$, y^+), +1)

    得到多个负样本{(($\mathbf{x}$, y_i), -1)},$\forall y_i \in \mathcal{Y}_{\mathbf{x}} \backslash \{y^+\}$
  - 第二步：训练二值分类模型
    - 任然采用比较方式对待每一个输入样本，即：对于每一个训练样本$\mathbf{x}$，我们希望y^+的分值大于其他y的分值。
    - 对于不同的输入样本$\mathbf{x}$，其不同标签分值之间的比较没有意义

Structured Perceptron

M. Collins. Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms. In EMNLP '02, pages 1–8, 2002.

- 输入: 训练样本$\{(\mathbf{x}_k, y_k)\}$
- 初始化: $\mathbf{w} = \mathbf{0}$
- 算法流程

for t=1 **TO** T //算法执行T轮

  for k=1 **TO** K //每一轮遍历所有的训练样本

    get $z_k = \text{argmax}_{z \in \mathcal{Y}_k} = \langle \mathbf{w}, \Psi(\mathbf{x}_k, z) \rangle$ //

    if $(y_k \neq z_k)$ **then**

      $\mathbf{w} \leftarrow \mathbf{w} + \Psi(\mathbf{x}_k, y_k) - \Psi(\mathbf{x}_k, z_k)$

    end if

  end for

end for

return w

Structured Perceptron

get $z_k = \text{argmax}_{z \in \mathcal{Y}_k} = \langle \mathbf{w}_{\text{old}}, \Psi(\mathbf{x}_k, z) \rangle$ //

if $(y_k \neq z_k)$ **then**

$$\mathbf{w}_{\text{new}} \leftarrow \mathbf{w}_{\text{old}} + \Psi(\mathbf{x}_k, y_k) - \Psi(\mathbf{x}_k, z_k)$$

end if

- 如果$\langle \mathbf{w}_{\text{old}}, \Psi(\mathbf{x}_k, z_k) \rangle > \langle \mathbf{w}_{\text{old}}, \Psi(\mathbf{x}_k, y_k) \rangle$,进入更新
- 更新效果:

$$\begin{aligned}
&\boldsymbol{\Delta}_{\mathbf{new}}(y_k, z_k) = \langle \mathbf{w}_{\text{new}}, \Psi(\mathbf{x}_k, y_k) \rangle - \langle \mathbf{w}_{\text{new}}, \Psi(\mathbf{x}_k, z_k) \rangle \\
&= \langle \mathbf{w}_{\text{old}} + \Psi(\mathbf{x}_k, y_k) - \Psi(\mathbf{x}_k, z_k), \Psi(\mathbf{x}_k, y_k) \rangle \\
&- \langle \mathbf{w}_{\text{old}} + \Psi(\mathbf{x}_k, y_k) - \Psi(\mathbf{x}_k, z_k), \Psi(\mathbf{x}_k, z_k) \rangle \\
&= \langle \mathbf{w}_{\text{old}}, \Psi(\mathbf{x}_k, y_k) \rangle - \langle \mathbf{w}_{\text{old}}, \Psi(\mathbf{x}_k, z_k) \rangle + \langle \Psi(\mathbf{x}_k, y_k) - \Psi(\mathbf{x}_k, z_k), \Psi(\mathbf{x}_k, y_k) \rangle \\
&- \langle \Psi(\mathbf{x}_k, y_k) - \Psi(\mathbf{x}_k, z_k), \Psi(\mathbf{x}_k, z_k) \rangle \\
&= \boldsymbol{\Delta}_{\boldsymbol{old}}(y_k, z_k) + \langle \Psi(\mathbf{x}_k, y_k) - \Psi(\mathbf{x}_k, z_k), \Psi(\mathbf{x}_k, y_k) - \Psi(\mathbf{x}_k, z_k) \rangle \geq \boldsymbol{\Delta}_{\boldsymbol{old}}(y_k, z_k)
\end{aligned}$$

Structured Perceptron理论分析

- [Collins 2002]证明
  - If there is a parameter vector **w** which makes zero errors on the training set, then after a finite number of iterations the training algorithm will have converged to parameter values with zero training error
  - If the training data is not separable, but "close" to being separable, then the algorithm will make a small number of mistakes (on the training data)
  - If the algorithm makes a small number of errors on the training data, it is likely to generalize well to unseen data

最大边距结构化输出模型

I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support Vector Learning for Interdependent and Structured Output Spaces, ICML, 2004

软件包下载：

http://www.cs.cornell.edu/People/tj/svm_light/svm_struct.html

- 基本思路：

  1. 对每一输入x，其正确标注y^+与其它错误标注y_i的边距为

$$\min_{y \in \mathcal{Y}_{\mathbf{x}} \setminus \{y^+\}} F(\mathbf{x}, y^+) - F(\mathbf{x}, y)$$

  – 注意：采用比较方式计算边距。区别于分类边距

  2. 训练过程中，用SVM最大化边距，期待能够取得更好的可泛化能力

  – 注意：独立同分布条件被打破，并**没有**理论保证

SVM-struct

- 给定训练集合$\{(\mathbf{x}_i, y_i)\}$，y_i为对$\mathbf{x}_i$的人工标注正确答案（目前还没有负例）
- 假设对每一个$\mathbf{x}_i$，其所有合法的标注集合为$\mathcal{Y}_{\mathbf{x}_i}$，显然$y_i \in \mathcal{Y}_{\mathbf{x}_i}$
- 硬边距SVM-struct

$$\min_{\mathbf{w}} \frac{1}{2}|\mathbf{w}|^2$$

$$s.t. \quad \forall i, \forall y \in \mathcal{Y}_{\mathbf{x}_i} \backslash \{y_i\}:$$

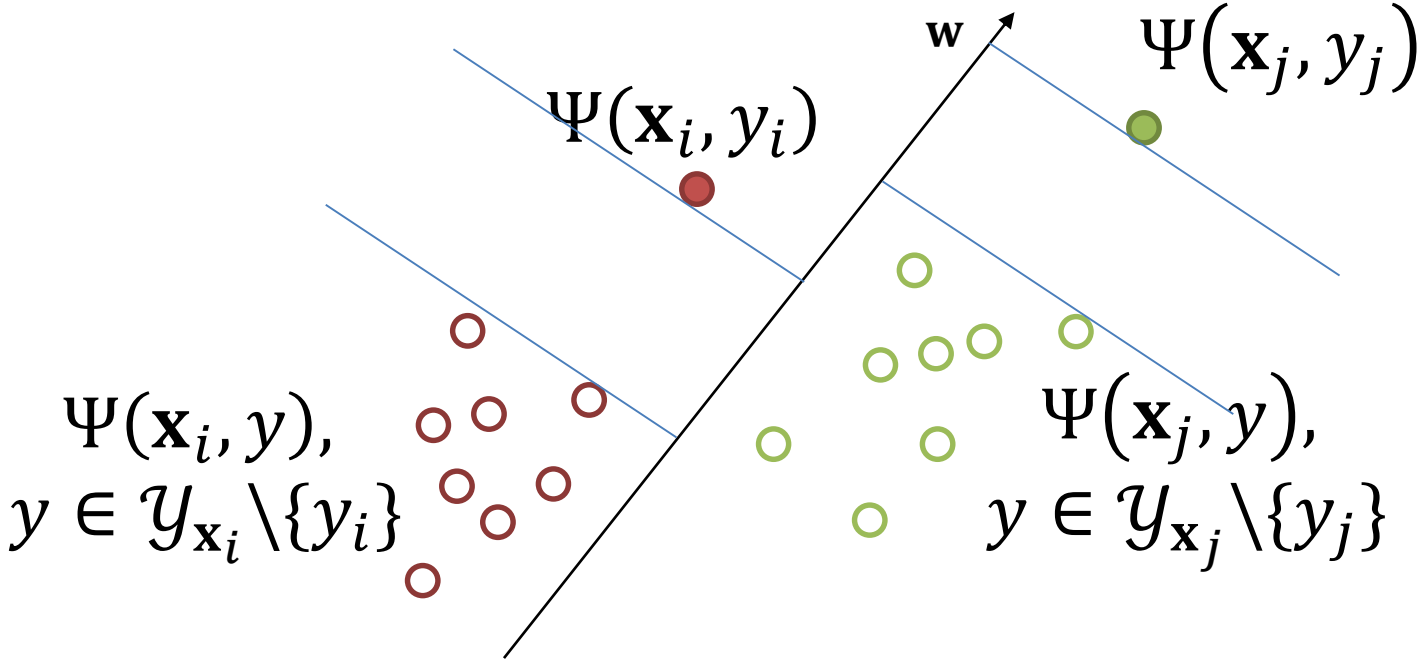
$$\langle \mathbf{w}, \Psi(\mathbf{x}_i, y_i) \rangle - \langle \mathbf{w}, \Psi(\mathbf{x}_i, y) \rangle \geq 1$$

SVM-struct

$$\min_{\mathbf{w}} \frac{1}{2}|\mathbf{w}|^2$$

$$s.t. \quad \forall i, \forall y \in \mathcal{Y}_{\mathbf{x}_i} \backslash \{y_i\}:$$

$$\langle \mathbf{w}, \Psi(\mathbf{x}_i, y_i) \rangle - \langle \mathbf{w}, \Psi(\mathbf{x}_i, y) \rangle \geq 1$$



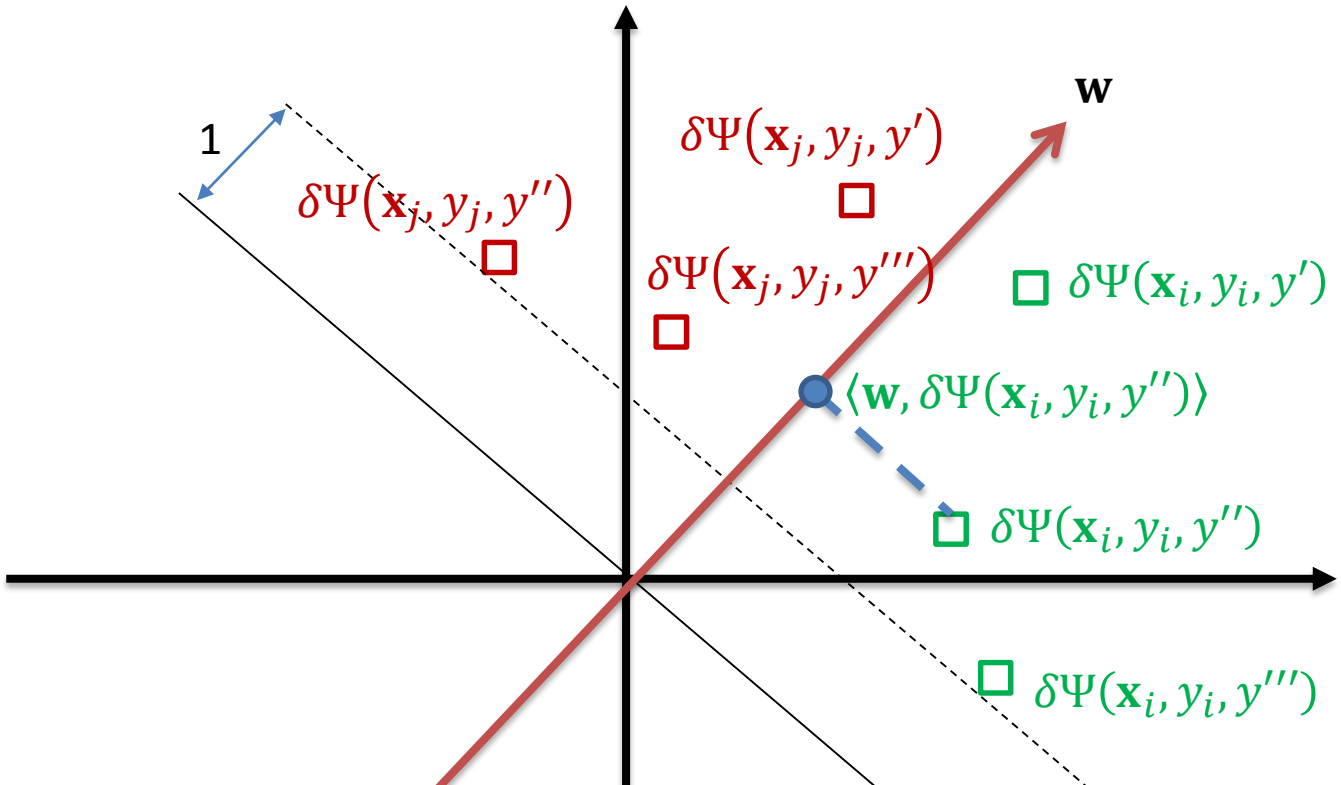
SVM-struct

$$\min_{\mathbf{w}} \frac{1}{2}|\mathbf{w}|^2$$

$$s.t. \quad \forall i, \forall y \in \mathcal{Y}_{\mathbf{x}_i} \backslash \{y_i\}:$$

$$\langle \mathbf{w}, \Psi(\mathbf{x}_i, y_i) \rangle - \langle \mathbf{w}, \Psi(\mathbf{x}_i, y) \rangle = \langle \mathbf{w}, \delta\Psi(\mathbf{x}_i, y_i, y) \rangle \geq 1$$

$$\delta\Psi(\mathbf{x}_i, y_i, y) = \Psi(\mathbf{x}_i, y_i) - \Psi(\mathbf{x}_i, y)$$



SVM-struct对偶形式

$$\min_{\mathbf{w}} \frac{1}{2}|\mathbf{w}|^2$$

$$s.t. \quad \forall i, \forall y \in \mathcal{Y}_{\mathbf{x}_i} \backslash \{y_i\}: \langle \mathbf{w}, \Psi(\mathbf{x}_i, y_i) \rangle - \langle \mathbf{w}, \Psi(\mathbf{x}_i, y) \rangle \geq 1$$

$$\delta\Psi_i(\mathrm{y}) = \Psi(\mathbf{x}_i, y_i) - \Psi(\mathbf{x}_i, y)$$

$$\max_{\alpha} \sum_{i, \mathbf{y} \neq \mathbf{y}_i} \alpha_{i\mathbf{y}} - \frac{1}{2} \sum_{\substack{i, \mathbf{y} \neq \mathbf{y}_i \\ j, \bar{\mathbf{y}} \neq \mathbf{y}_j}} \alpha_{i\mathbf{y}} \alpha_{j\bar{\mathbf{y}}} \langle \delta\Psi_i(\mathbf{y}), \delta\Psi_j(\bar{\mathbf{y}}) \rangle$$

$$\text{s.t. } \forall i, \forall \mathbf{y} \neq \mathcal{Y} \setminus \mathbf{y}_i : \quad \alpha_{i\mathbf{y}} \geq 0 .$$

软边距SVM-struct

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

$$s.t. \quad \forall i, \forall y \in \mathcal{Y}_{\mathbf{x}_i} \backslash \{y_i\}: \langle \mathbf{w}, \delta\Psi(\mathbf{x}_i, y_i, y)\rangle \geq 1 - \xi_i$$

- $\sum_{i=1}^{N}\xi_i$：对于一个输入$\mathbf{x}_i$，虽然为其生成了多个正例和负例，形成了多个比较，但是只定义一个边距值ξ_i：

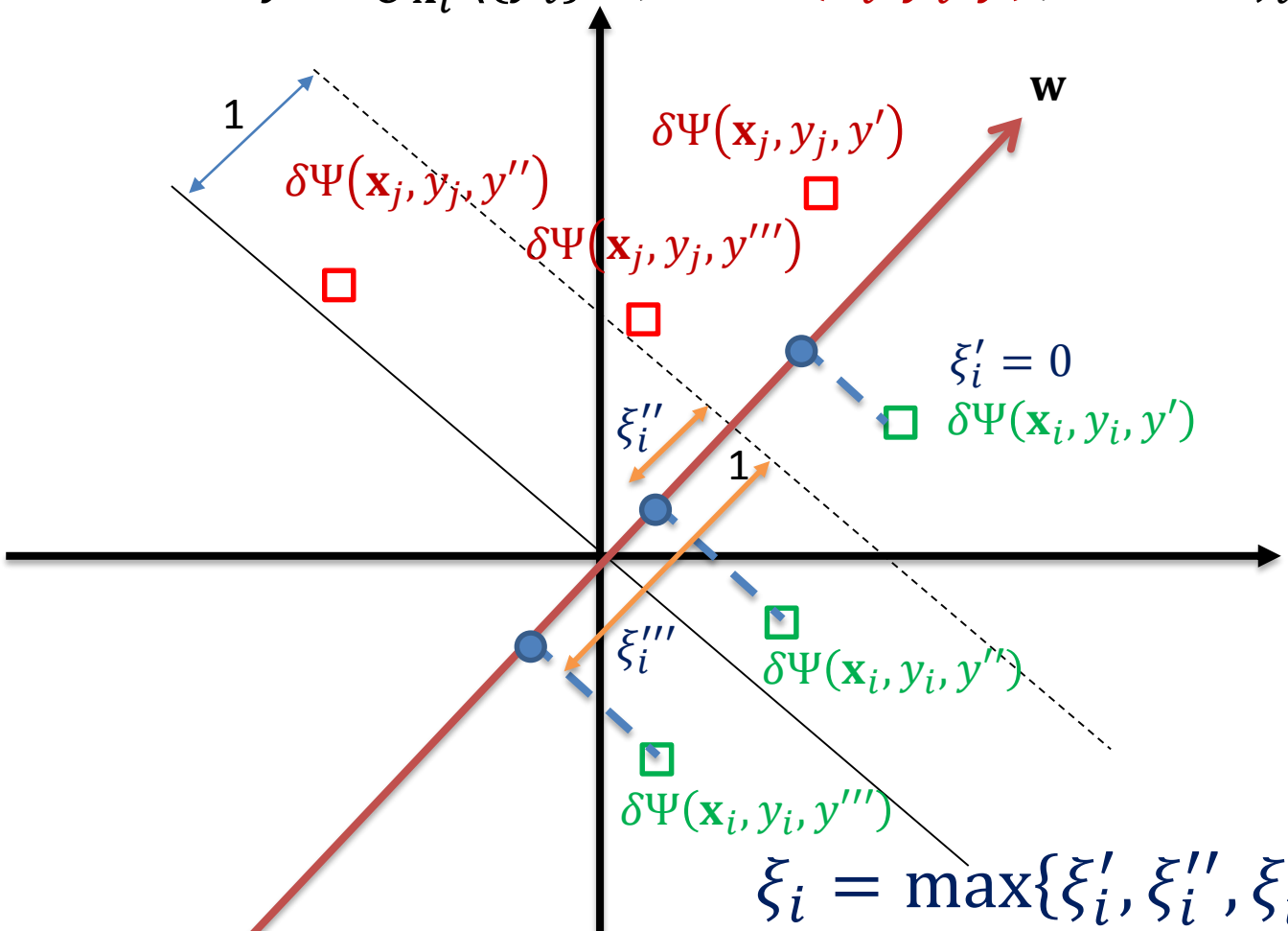
$$\forall y, \xi_i \geq 1 - \langle \mathbf{w}, \delta\Psi(\mathbf{x}_i, y_i, y)\rangle$$

- ξ_i按照第i个训练样本生成的所有正负例对中，惩罚最大的那个选取
- 思考：
  - Structured Perceptron如何处理一个训练样本所生成的多个正负例对？与SVM-struct相比有何优势和劣势？
  - 如何求解？

软边距SVM-struct

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

$$s.t. \quad \forall i, \forall y \in \mathcal{Y}_{\mathbf{x}_i} \backslash \{y_i\}: \langle \mathbf{w}, \delta\Psi(\mathbf{x}_i, y_i, y)\rangle \geq 1 - \xi_i, \xi_i \geq 0$$

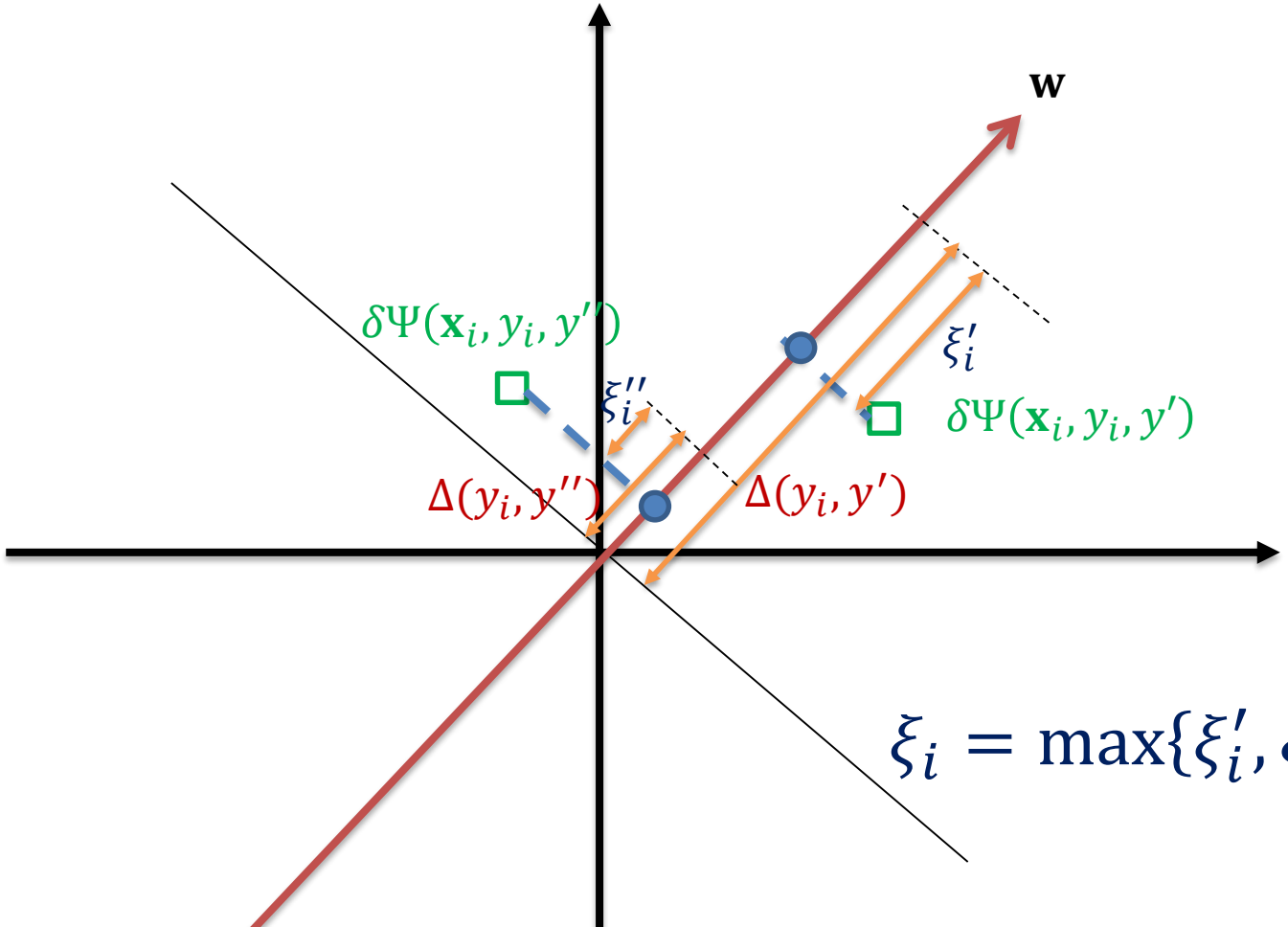


$$\xi_i = \max\{\xi_i', \xi_i'', \xi_i'''\} = \xi_i'''$$

变边距SVM-struct

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

$$s.t. \quad \forall i, \forall y \in \mathcal{Y}_{\mathbf{x}_i} \backslash \{y_i\}: \langle \mathbf{w}, \delta\Psi(\mathbf{x}_i, y_i, y)\rangle \geq \Delta(y_i, y) - \xi_i, \xi_i \geq 0$$



$$\xi_i = \max\{\xi_i', \xi_i''\} = \xi_i'$$

变边距SVM-struct

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

$$s.t. \quad \forall i, \forall y \in \mathcal{Y}_{\mathbf{x}_i} \backslash \{y_i\}: \langle \mathbf{w}, \delta\Psi(\mathbf{x}_i, y_i, y)\rangle \geq \Delta(y_i, y) - \xi_i, \quad \xi_i \geq 0$$

- 如何定义$\Delta(y_i, y)$？
  - 定义为y_i与y在某一个评价指标E意义下的差距，则算法可以直接看成直接优化评价准则E

$$\min_{\mathbf{w}} \sum_i E(y_i) - E(y) = \min \sum_i 1 - E(y)$$

其中y为依照模型参数w所生成的结果：$\text{argmax}_{y \in \mathcal{Y}_{\mathbf{x}}} \langle \mathbf{w}, \Psi(\mathbf{x}, y)\rangle$。

  - 我们将在排序学习章节继续讨论这个问题。

提纲

- 背景介绍
- 常用的监督学习算法
  - 决策树
  - 支持向量机/Perceptron
  - AdaBoost
- 结构化输出预测
- 总结

监督学习

- 监督学习是数据挖掘/机器学习中最重要的一类学习任务和算法，本次课程只覆盖了很少的一部分
  - 监督学习的背景和问题定义
  - 具有代表性的分类算法
  - 结构化输出预测
- 本节课未涉及到的重要内容
  - 排序学习(将在互联网搜索部分讲述)、神经网络(深度学习)
  - 模型泛化能力(机器学习课程中会涉及到)
  - 最优化技术(SVM涉及到SMO)
  - 半监督学习
  - 强化学习(reinforcement learning)

监督学习

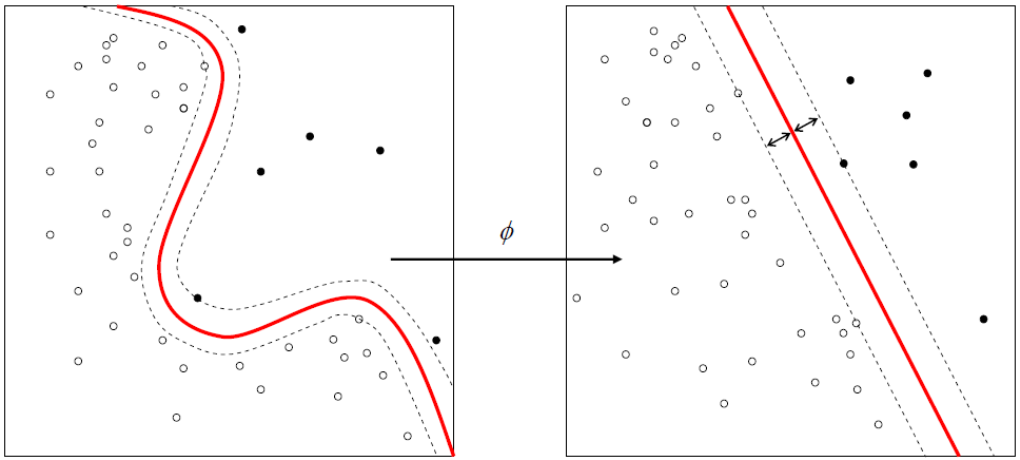
- 监督学习算法应用广泛
  - 从人工标注数据中学习模型，有的放矢
  - 不仅可以预测简单标签/数值，也可以预测相对复杂的结构化标签(Structured output prediction)
  - 任务相关的特征，融合人的知识
  - 在未知数据上的预测能力
  - 有效的评价方法

Thanks!

核函数

如何学习非线性SVM模型？

- SVM是线性分类器，要求数据线性可分
  - 实际任务可能并非如此
- 解决方案之一
  - 把数据映射到高维空间中
  - 在高维空间中应用线性SVM
  - 原理：在低维空间线性不可分的问题在高维空间可能线性可分



空间转换

- 通过一个转换函数ϕ将输入空间中的点$\mathbf{x}$转换到(高维)特征空间中的点$\phi(\mathbf{x})$，记为$\mathbf{x} \mapsto \phi(\mathbf{x})$
- 训练数据$\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_N, y_N)\}$转换到新的空间中，成为
  $\{(\phi(\mathbf{x}_1), y_1), (\phi(\mathbf{x}_2), y_2), \cdots, (\phi(\mathbf{x}_N), y_N)\}$
- 在高维空间的分类器表达为
  $$f(\mathbf{x}) = \langle \mathbf{w}, \phi(\mathbf{x}) \rangle$$
- 空间转换的一个例子$\phi\big((x_1, x_2)\big) = \big(x_1^2, x_2^2, \sqrt{2}x_1x_2\big)$
  数据点$((2, 3), +1)$转换为$((4, 9, 6\sqrt{2}), +1)$

显式空间转换的问题

- $\phi(\mathbf{x})$的维度变高，影响算法执行效率(维度诅咒, the curse of dimensionality)
- 当$\phi(\mathbf{x})$维度过高甚至无穷维时，显式的空间转换无法进行
- 如何避免？利用核函数。
  - 观察SVM的对偶问题，其与$\mathbf{x}$有关的项只是两两的内积

$$\max_{\alpha_1,\cdots,\alpha_N} \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{k=1}^{N}\sum_{l=1}^{N} \alpha_k \alpha_l y_k y_l \langle \mathbf{x}_k, \mathbf{x}_l \rangle$$

$$\text{s.t.} \quad \forall i: 0 \le \alpha_i \le C, \sum_{i=1}^{N} y_i \alpha_i = 0$$

核函数SVM

$$\max_{\alpha_1,\cdots,\alpha_N} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{k=1}^{N} \sum_{l=1}^{N} \alpha_k \alpha_l y_k y_l \langle \phi(\mathbf{x}_k), \phi(\mathbf{x}_l) \rangle$$

$$\text{s.t.} \quad \forall i: 0 \le \alpha_i \le C, \sum_{i=1}^{N} y_i \alpha_i = 0$$

$$f(\mathbf{x}) = \left\langle \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i, \mathbf{x} \right\rangle = \sum_{i=1}^{N} \alpha_i y_i \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}) \rangle$$

- SVM的训练和测试只需要知道数据点间的内积(相似度)
- 不需要显式计算ϕ， 空间转换函数ϕ的作用只是在转换后空间内计算内积
- 绕过ϕ函数直接定义内积（核函数）
  - $K(\mathbf{x}_i, \mathbf{x}_j) := \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$

核函数SVM

- 选择核函数$K: X \times X \to R$

可预先基于训练数据计算N*N矩阵

模型训练

$$\max_{\alpha_1,\cdots,\alpha_N} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{k=1}^{N} \sum_{l=1}^{N} \alpha_k \alpha_l y_k y_l K(\mathbf{x}_i, \mathbf{x}_j)$$

$$\text{s.t.} \quad \forall i: 0 \le \alpha_i \le C, \sum_{i=1}^{N} y_i \alpha_i = 0$$

预测函数

$$f(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x})$$

核函数举例

- 多项式核函数$K(\mathbf{x},\mathbf{z})=\langle\mathbf{x},\mathbf{z}\rangle^{\mathrm{d}}$
- $\mathrm{d}=2,\mathbf{x}=(x_1,x_2),\mathbf{z}=(z_1,z_2)$

$$
\begin{aligned}
&K(\mathbf{x},\mathbf{z})=\langle\mathbf{x},\mathbf{z}\rangle^2=(x_1z_1+x_2z_2)^2\\
&=x_1^2z_1^2+x_2^2z_2^2+2x_1z_1x_2z_2\\
&=\left\langle\left(x_1^2,x_2^2,\sqrt{2}x_1x_2\right),\left(z_1^2,z_2^2,\sqrt{2}z_1z_2\right)\right\rangle\\
&=\langle\phi(\mathbf{x}),\phi(\mathbf{z})\rangle
\end{aligned}
$$

- $\langle\mathbf{x},\mathbf{z}\rangle^2$为在转换空间内的内积

常用的核函数

$$K(\mathbf{x}, \mathbf{z}) = \langle \mathbf{x} \cdot \mathbf{z} \rangle. \tag{85}$$

Commonly used kernels include

$$\text{Polynomial:} \quad K(\mathbf{x}, \mathbf{z}) = (\langle \mathbf{x} \cdot \mathbf{z} \rangle + \theta)^d \tag{86}$$

$$\text{Gaussian RBF:} \quad K(\mathbf{x}, \mathbf{z}) = e^{-\|\mathbf{x}-\mathbf{z}\|^2 / 2\sigma} \tag{87}$$

$$\text{Sigmoidal:} \quad K(\mathbf{x}, \mathbf{z}) = \tanh(k\langle \mathbf{x} \cdot \mathbf{z} \rangle - \delta) \tag{88}$$

where $\theta \in R$, $d \in N$, $\sigma > 0$, and $k, \delta \in R$.

- 核函数可以通过组合计算得到更多核函数
- 假设$K(\mathbf{x}, \mathbf{z})$和$K'(\mathbf{x}, \mathbf{z})$是核函数，则以下函数也是核函数
  - $K(\mathbf{x}, \mathbf{z}) + K'(\mathbf{x}, \mathbf{z})$
  - $\alpha K(\mathbf{x}, \mathbf{z})$ for $\alpha > 0$