

Ensemble Methods 集成学习

Parallel Ensemble Methods

Bootstrapping: 从dataset 中又放回的随机抽样

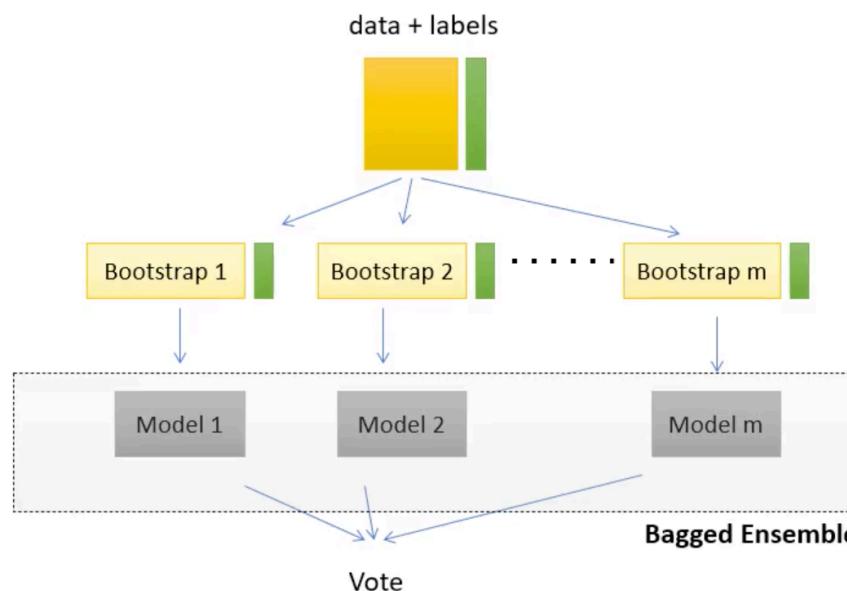
Bagging (Bootstrap AGGREGATING) 一种并行式的集成学习方法
各个学习器之间没有依赖关系，可以并行生成

进行m次有放回的随机采样操作，
从而得到m个样本的采样集

这样训练集中有接近36.8%的样本
没有被采到 包含63%原始数据

$$\lim_{m \rightarrow \infty} \left(1 - \frac{1}{m}\right)^m \mapsto \frac{1}{e} \approx 0.368$$

Bagging关注于降低方差



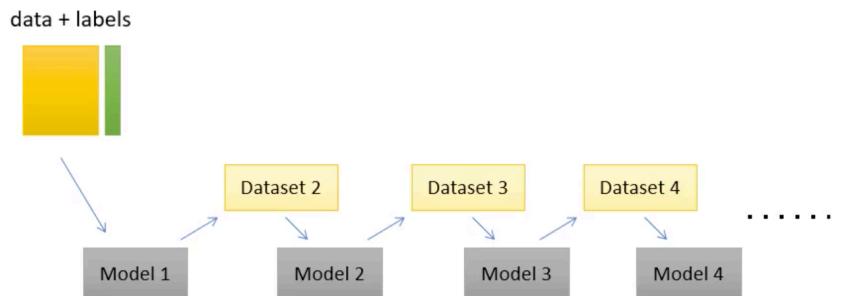
Sequential Ensemble Methods

Boosting 一种串行的工作机制

增加前一个基学习器在训练过程中预测错误样本的权重，使得后续基学习器更加关注这些打标错误的训练样本，尽可能纠正这些错误，一直向下串行直至产生需要的T个基学习器

Include Adaboost, Xgboost
gradient boosted tree

Boosting主要关注于降低偏差



Define a distribution over the training set, $D_1(i) = \frac{1}{N}, \forall i$.
for $t = 1$ to T do

Build a model h_t from the training set, using distribution D_t .

Update D_{t+1} from D_t :

 Increase the weight on examples that h_t incorrectly classifies.

 Decrease the weight on examples that h_t correctly classifies.

end for

For a new testing point (x', y') , we take a weighted majority vote from $\{h_1, \dots, h_T\}$.

1. 初始化训练数据的权值分布

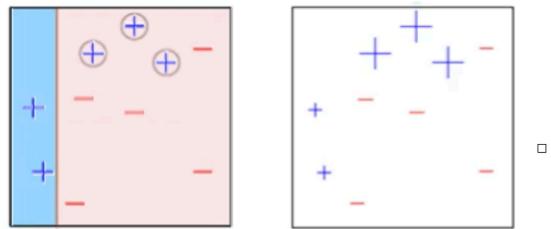
每一个训练样本最开始时都被赋予相同的权值: $1/N$

2. 准确分类，错误率小 \Rightarrow weight -

 没有准确分类 \Rightarrow weight +

Boosting: with re-weighting (example)

x1	x2	weight	Label	prediction
0.1	0.1	0.2 ↓↓	0	0
0.2	0.1	0.2 ↓	0	0.4
0.3	0.4	0.2 ↑	1	0.4
0.2	0.5	0.2 ↓	1	0.6
0.1	0.2	0.2 ↓↓	1	0.9



Update the weight of each training example:

- Downweight if prediction matches label
- Upweight prediction does not match label
- Amount of weight adjustment depends on prediction uncertainty

AdaBoost 核心步骤就是计算基学习器权重和样本权重分布

Input: Data set $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$;

Base learning algorithm L ;

Number of learning rounds T .

Process:

1. $\mathcal{D}_1(i) = 1/m$. % Initialize the weight distribution
2. **for** $t = 1, \dots, T$:
3. $h_t = L(D, \mathcal{D}_t)$; % Train a learner h_t from D using distribution \mathcal{D}_t
4. $\epsilon_t = \Pr_{x \sim \mathcal{D}_t, y} I[h_t(x) \neq y]$; % Measure the error of h_t
5. **if** $\epsilon_t > 0.5$ **then break**
6. $\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)$; % Determine the weight of h_t (1) 计算基学习器权重
7. $\mathcal{D}_{t+1}(i) = \frac{\mathcal{D}_t(i)}{Z_t} \times \begin{cases} \exp(-\alpha_t) & \text{if } h_t(x_i) = y_i \\ \exp(\alpha_t) & \text{if } h_t(x_i) \neq y_i \end{cases}$ (2) 样本权重分布更新

$$\frac{\mathcal{D}_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$
 % Update the distribution, where
% Z_t is a normalization factor which
% enables \mathcal{D}_{t+1} to be distribution

8. **end**

Output: $H(x) = \text{sign} (\sum_{t=1}^T \alpha_t h_t(x))$

1 样本选择

Bagging: 训练集是在原始集中有放回选取的，从原始集中选出的各轮训练集之间是独立的。

Boosting: 每一轮的训练集不变，只是训练集中每个样例在分类器中的权重发生变化。而权值是根据上一轮的分类结果进行调整。

2 样例权重

Bagging: 使用均匀取样，每个样例的权重相等

Boosting: 根据错误率不断调整样例的权值，错误率越大则权重越大。

3 预测函数

Bagging: 所有预测函数的权重相等。

Boosting: 每个弱分类器都有相应的权重，对于分类误差小的分类器会有更大的权重。

4 并行计算

Bagging: 各个预测函数可以并行生成。

Boosting: 各个预测函数只能顺序生成，因为后一个模型参数需要前一轮模型的结果。

Random Forest 随机森林

基学习器固定为决策树 & bootstrapping

Random Forests (input training data+ labels T , number of trees M)

```
for  $j = 1$  to  $M$  do
    Take a bootstrap sample  $T^j$  from  $T$ 
    Build a decision tree using  $T^j$ , but, at every split point:
        - Choose a random fraction  $K$  of the remaining features
        - Pick the best feature (minimising cost) from that subset
    Add the tree to the set, without pruning
end for
return set of trees
```

1. Bootstrapping 选取n个样本；
2. 对n个样本选取a个特征中的随机k个，用建立决策树的方法获得最佳分割点；
3. 重复m次，获得m个决策树；
4. 对输入样例进行预测时，每个子树都产生一个结果，采用多数投票机制输出。

数据集的随机选取：从原始的数据集中采取有放回的抽样 (bagging)

待选特征的随机选取：随机森林中的子树的每一个分裂过程并未用到所有的待选特征，而是从所有的待选特征中随机选取一定的特征 (random feature selection)

注意点

1. 在构建决策树的过程中是不需要剪枝的。
2. 整个森林的树的数量和每棵树的特征需要人为进行设定。
3. 构建决策树的时候分裂节点的选择是依据最小基尼系数的。

	装袋法 Bagging	提升法 Boosting
评估器	相互独立，同时运行	相互关联，按顺序依次构建，后建的模型会在先建模型预测失败的样本上有更多的权重
抽样数集	有放回抽样	有放回抽样，但会确认数据的权重，每次抽样都会给容易预测失败的样本更多的权重
决定集成的结果	平均或少数服从多数原则	加权平均，在训练集上表现更好的模型会有更大的权重
目标	降低方差，提高模型整体的稳定性	降低偏差，提高模型整体的精确度
单个评估器存在过拟合问题的时候	能够一定程度上解决过拟合问题	可能会加剧过拟合问题
单个评估器的效力比较弱的时候	不是非常有帮助	很可能会提升模型表现
代表算法	随机森林	梯度提升树， Adaboost

Feature selection

Wrapper Methods

直接把最后要使用的分类器作为特征选择的评价函数，对于特定的分类器选择最优的特征子集

Bit set to 1 means we use that feature, otherwise 0 ...

0 1 1 1 0 1 1 0 0 0 0 1 1 1 0 0 0 0 0 0

...so use 8 features.

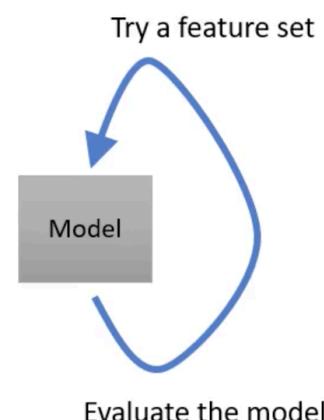
With M total features...

2^M possible sets!

20 features ... 1 million feature sets to check

25 features ... 33.5 million sets

30 features ... 1.1 billion sets



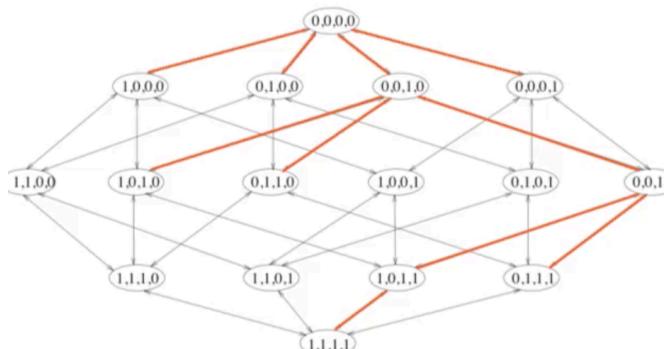
Feasible only for $M=20$

1. Greedy forward selection (greedy search)

贪心找最优解

贪婪搜索方法，评估所有可能的特征组合

Greedy forward search evaluates $\frac{M(M+1)}{2}$ sets

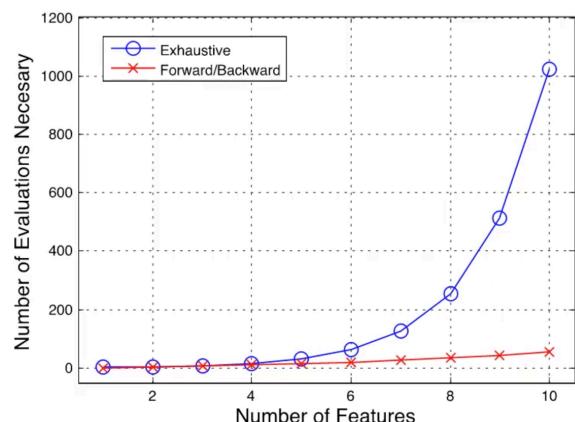


2. Backward feature elimination

从完整模型（包括所有自变量）开始，然后删除具有最高p值（>显著性水平）的无关紧要的特征。这个过程一次又一次地重复.until no improvement is observed on removal of features.

Pros and cons

- No guarantee of best solution
- Need to train and evaluate lots of models.
- Impact of a feature on the ML classifier is explicit
- Better than an exhaustive search



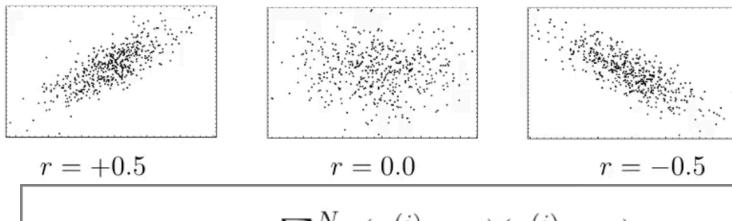
Filter Methods

先进行特征选择，然后去训练学习器，所以特征选择的过程与学习器无关。

相当于先对于特征进行过滤操作，然后用特征子集来训练分类器。

Pearson Correlation Coefficient 皮尔森相关系数

最简单的，特征和响应变量之间关系的方法，衡量的是变量之间的线性相关性
结果取值区间为[-1, 1], -1表示完全的负相关, +1表示完全的正相关, 0表示没有线性相关。



Feature : $\mathbf{x}_k = \{x_k^{(1)}, \dots, x_k^{(N)}\}^T$

Target : $\mathbf{y} = \{y^{(1)}, \dots, y^{(N)}\}^T$

$$r(\mathbf{x}, \mathbf{y}) = \frac{\sum_{i=1}^N (x^{(i)} - \bar{x})(y^{(i)} - \bar{y})}{\sqrt{\sum_{i=1}^N (x^{(i)} - \bar{x})^2} \sqrt{\sum_{i=1}^N (y^{(i)} - \bar{y})^2}}$$

$J(X_k) = |r(\mathbf{x}_k, \mathbf{y})|$ (i.e. absolute correlation with target)

Algorithm

10. Rank features in descending order by J .

20. Evaluate predictor on M nested subsets.

30. Choose subset with lowest validation error.

Fisher score

Limitation: only linear relationships

鉴别性能强的特征的表现

only for one feature

==> 类内部样本点的距离

assuming two real-valued variables

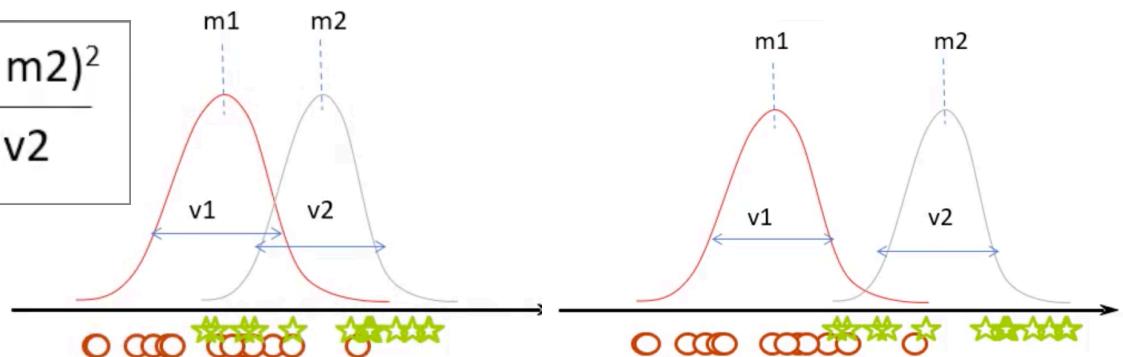
尽可能小，类之间的距离尽量大

fisher score 越大 ==> 区分度越好

类间方差: $(m_1 - m_2)^2$ is called the between-class scatter – BIG for good features.

类内方差: $v_1 + v_2$ is called the within-class scatter – SMALL for good features.

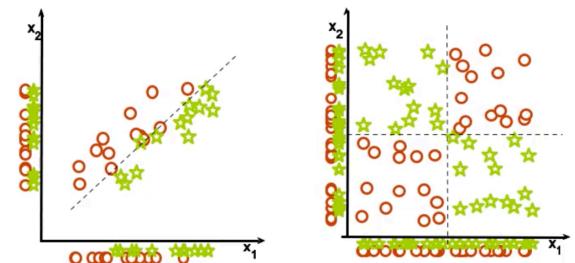
$$F = \frac{(m_1 - m_2)^2}{v_1 + v_2}$$



Single feature may not tell us the whole story
Two irrelevant features may be relevant together

Mutual Information

MI, 表示两个变量X与Y是否有关系，以及关系的强弱

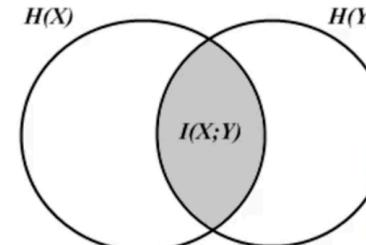


$$J(X_k) = I(X_k; Y) = \sum_{x \in X_k} \sum_{y \in Y} p(xy) \log_2 \frac{p(xy)}{p(x)p(y)}$$

$$H(X) = -\sum_{x \in X} p(x) \log_2 p(x)$$

$$H(X|Y) = -\sum_{x \in X} \sum_{y \in Y} p(x,y) \log_2 p(x|y)$$

$$\begin{aligned} I(X; Y) &= H(X) - H(X|Y) \\ &= -\sum_{x \in X} p(x) \log_2 p(x) + \sum_{x \in X} \sum_{y \in Y} p(x,y) \log_2 p(x|y) \\ &= \sum_{x \in X} \sum_{y \in Y} p(x,y) \left(\log_2 \frac{p(x,y)}{p(x)} \right) \\ I(X; Y) &= \sum_{x \in X} \sum_{y \in Y} p(x,y) \log_2 \frac{p(x,y)}{p(x)p(y)} \end{aligned} \quad (12)$$



X,Y关系越密切， $I(X,Y)$ 就越大

$I(X,Y)$ 取0时， 代表X与Y独立

Measures dependency of X,Y

Zero when independent.
Maximal when identical.

$I(X;Y)$ 的性质：

- 1) $I(X;Y) \geq 0$
- 2) $H(X) - H(X|Y) = I(X;Y) = I(Y;X) = H(Y) - H(Y|X)$
- 3) 当X,Y独立时， $I(X;Y)=0$ ，
- 4) 当X,Y知道一个就能推断另一个时， $I(X;Y)=H(X)=H(Y)$

Filter using mutual information

Calculate MI for any combination of features
从小到大排

Cut-off point decided by user, e.g. $|S| = 5$, so $S = \{35, 42, 10, 654, 22\}$.

i	$J(X_k)$
35	0.846
42	0.811
10	0.810
654	0.611
22	0.443
59	0.388
...	...
212	0.09
20	0.05

Filter VS Wrapper

- Filter methods quantify the relevance of features using statistics about the data
- Much faster (+ scales better to larger data sets)
- No guarantee that redundant features won't be selected