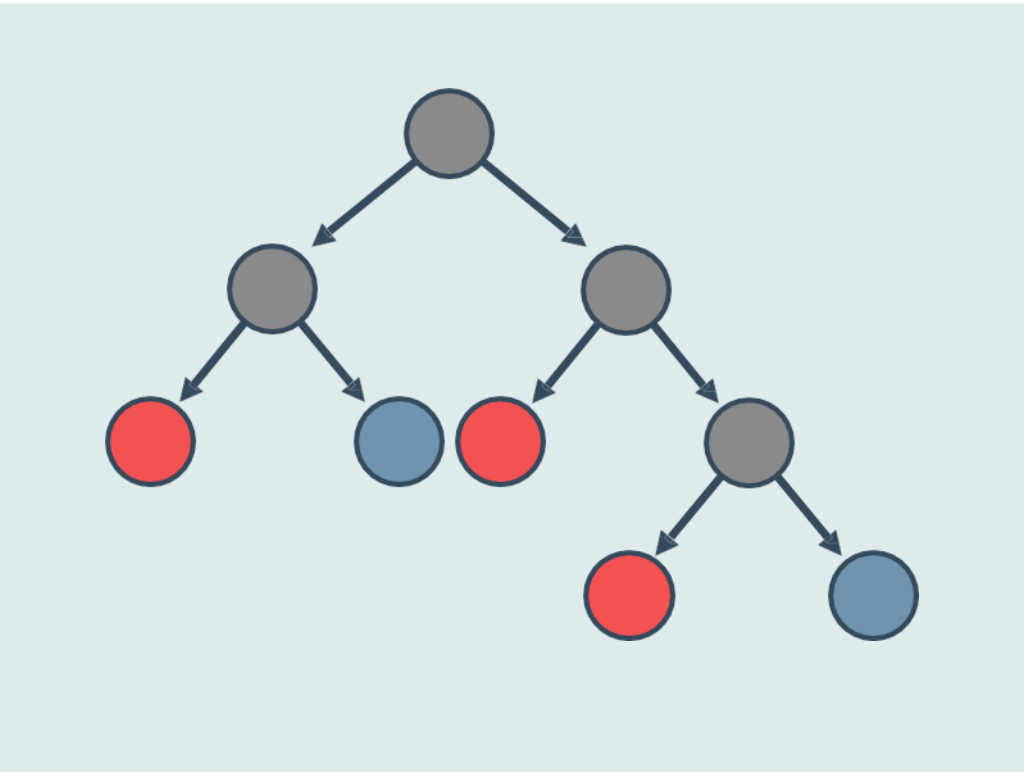
第12章 集成学习

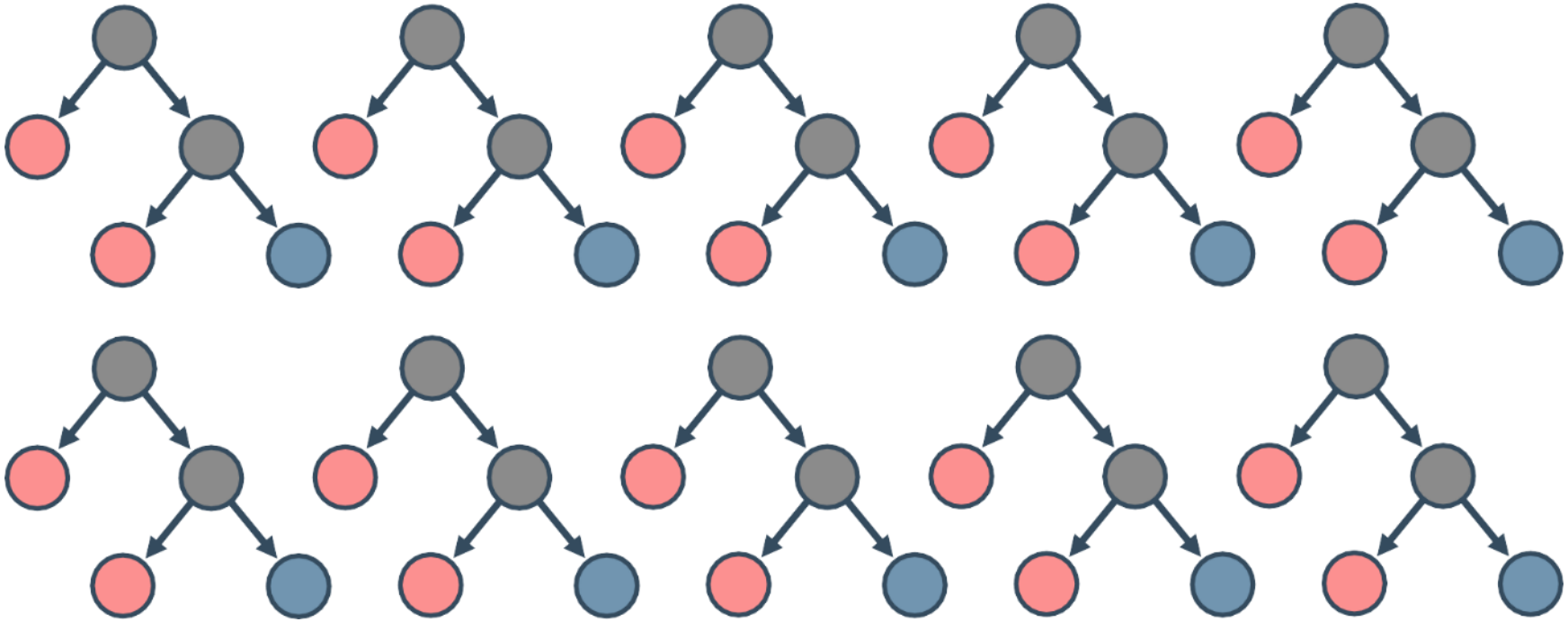
决策树：高方差



- 问题: 决策树容易过拟合
- 剪枝有助于将方差（variance）减少到一定程度
- 通常提高模型泛化的效果并不显著

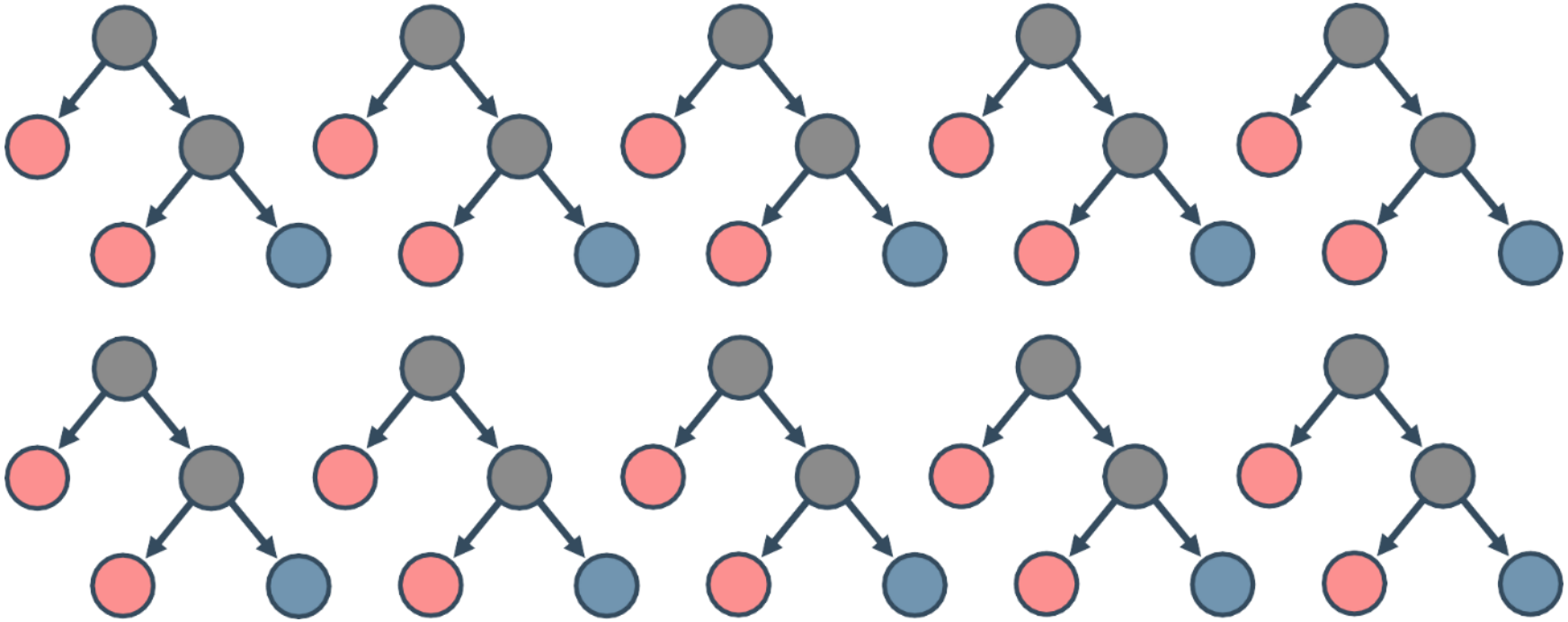
改进：使用多棵树

创建许多棵不同的树

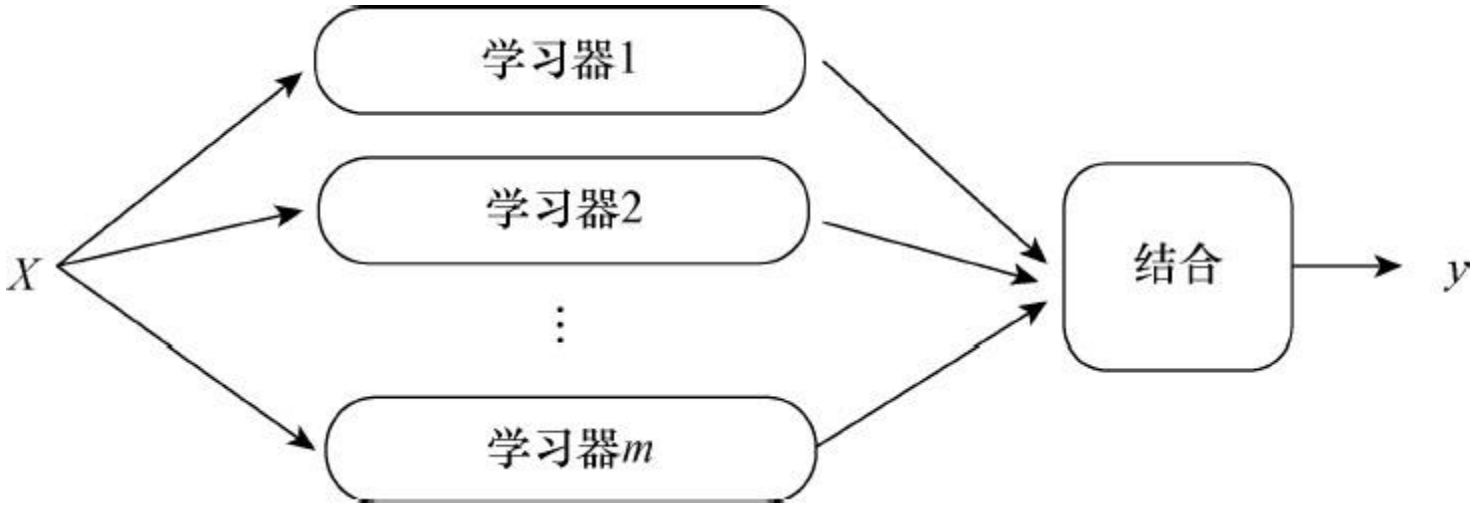


改进：使用多棵树

结合所有树的预测来降低方差



什么是集成学习



训练多个学习器来完成预测任务（分类或回归），然后通过结合多个学习器的预测结果而得到最终的结果。

- 简单平均
- 投票：硬投票、软投票

“三个臭皮匠，顶个诸葛亮”？

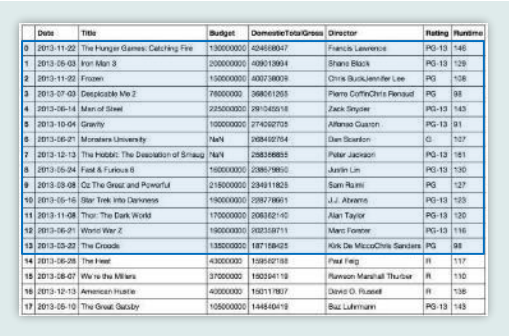
如何构建多个有差异的学习器

- 不同的机器学习模型
- 相同的机器学习模型
  - 不同的训练数据集
  - 不同的特征选择
  - 不同的超参数

BAGGING
袋装

如何创建多棵树

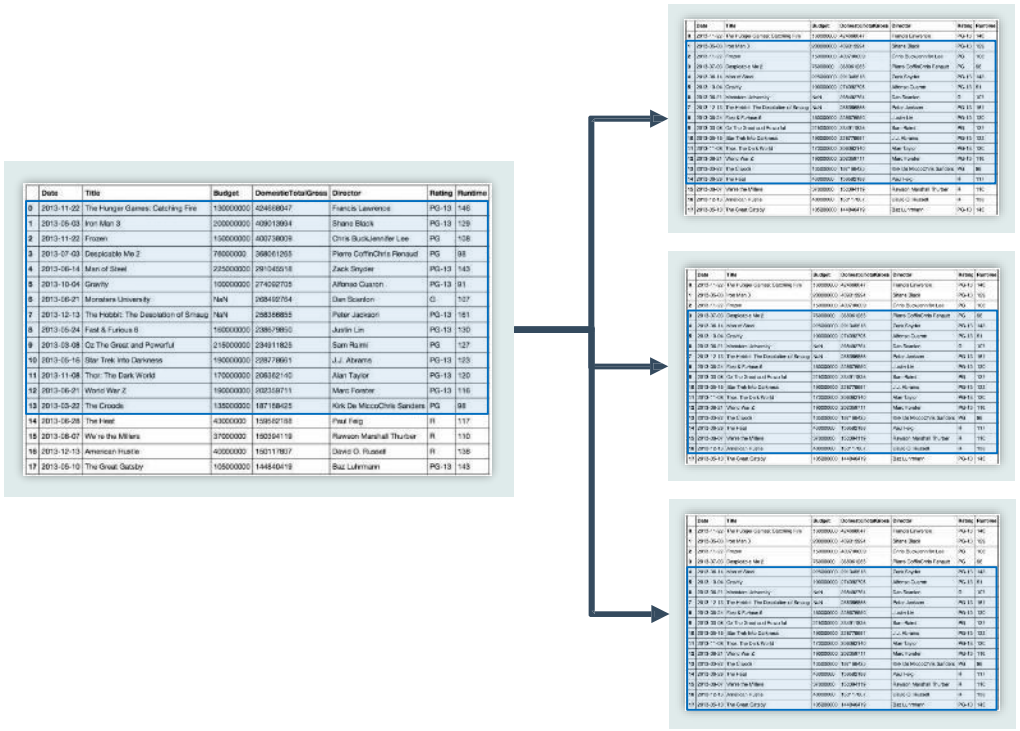
使用bootstrapping，即有放回的随机抽样



	Date	Title	Budget	DomesticTotalGross	Director	Rating	Runtime
0	2013-11-22	The Hunger Games: Catching Fire	130000000	424668047	Francis Lawrence	PG-13	146
1	2013-05-03	Iron Man 3	200000000	409013994	Shane Black	PG-13	129
2	2013-11-22	Frozen	150000000	400738009	Chris BuckJennifer Lee	PG	108
3	2013-07-03	Despicable Me 2	76000000	368061265	Pierre CoffinChris Renaud	PG	98
4	2013-06-14	Man of Steel	225000000	291045518	Zack Snyder	PG-13	143
5	2013-10-04	Gravity	100000000	274092705	Alfonso Cuaron	PG-13	91
6	2013-06-21	Monsters University	NaN	268492764	Dan Scanlon	G	107
7	2013-12-13	The Hobbit: The Desolation of Smaug	NaN	258366855	Peter Jackson	PG-13	161
8	2013-05-24	Fast & Furious 6	160000000	238679850	Justin Lin	PG-13	130
9	2013-03-08	Oz The Great and Powerful	215000000	234911825	Sam Raimi	PG	127
10	2013-05-16	Star Trek Into Darkness	190000000	228778661	J.J. Abrams	PG-13	123
11	2013-11-08	Thor: The Dark World	170000000	206362140	Alan Taylor	PG-13	120
12	2013-06-21	World War Z	190000000	202359711	Marc Forster	PG-13	116
13	2013-03-22	The Croods	135000000	187168425	Kirk De MicoChris Sanders	PG	98
14	2013-06-28	The Heat	43000000	159582168	Paul Feig	R	117
15	2013-08-07	We re the Millers	37000000	150394119	Rawson Marshall Thurber	R	110
16	2013-12-13	American Hustle	40000000	150117807	David O. Russell	R	138
17	2013-05-10	The Great Gatsby	105000000	144840419	Baz Luhrmann	PG-13	143

如何创建多棵树

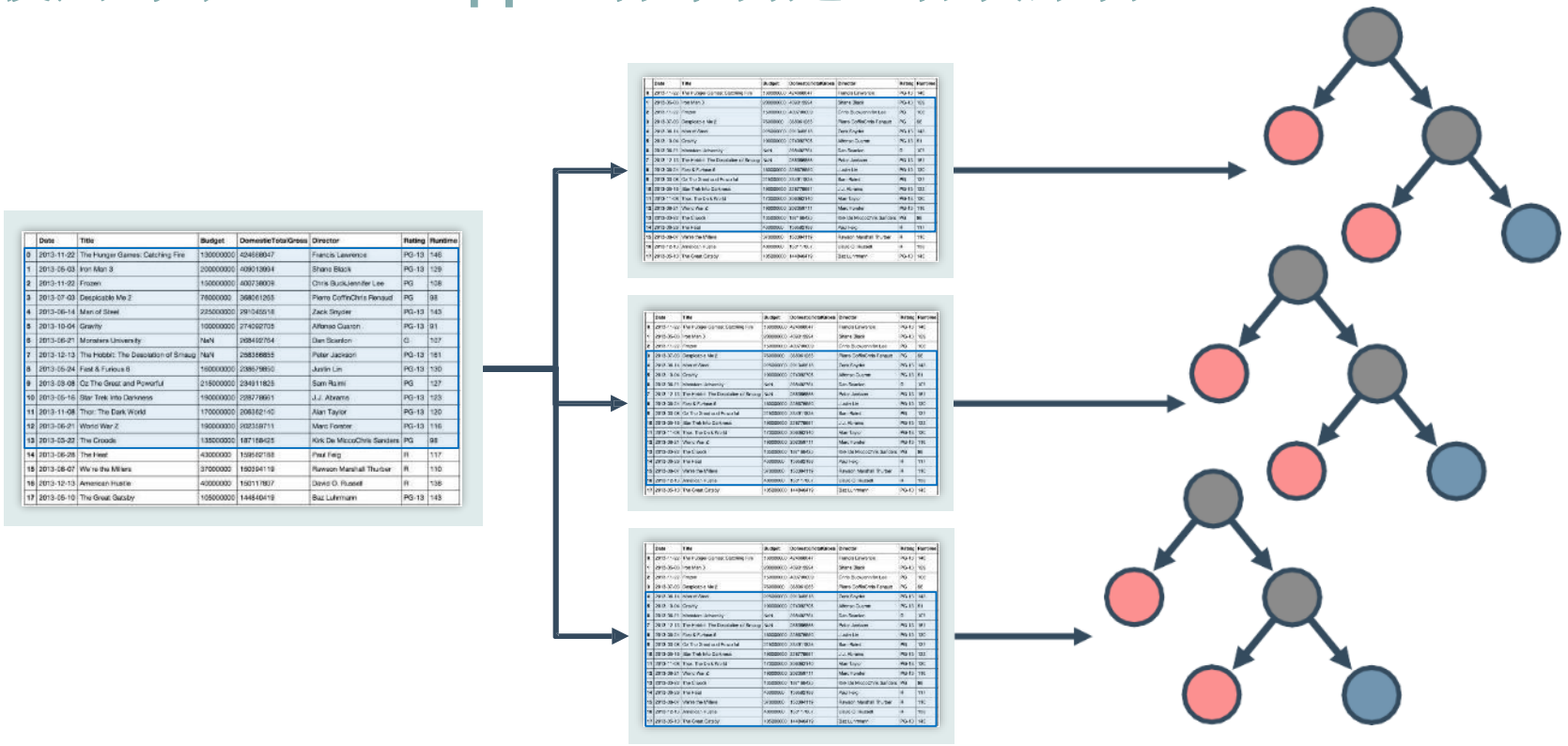
创建多个bootstrapped样本



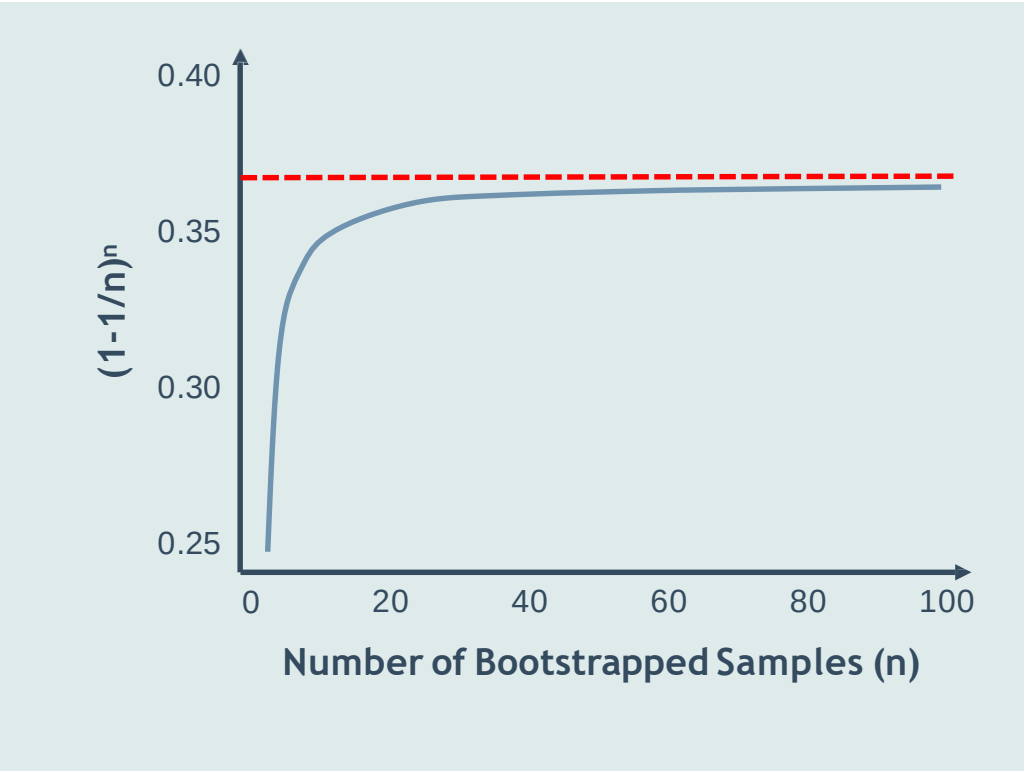
	Date	Title	Budget	DomesticTotalGross	Director	Rating	Runtime
0	2013-11-22	The Hunger Games: Catching Fire	130000000	424668047	Francis Lawrence	PG-13	146
1	2013-05-03	Iron Man 3	200000000	409013994	Shane Black	PG-13	129
2	2013-11-22	Frozen	150000000	400738009	Chris BuckJennifer Lee	PG	108
3	2013-07-03	Despicable Me 2	76000000	368061265	Pierre CoffinChris Renaud	PG	98
4	2013-06-14	Man of Steel	225000000	291045518	Zack Snyder	PG-13	143
5	2013-10-04	Gravity	100000000	274092705	Alfonso Cuaron	PG-13	91
6	2013-06-21	Monsters University	NaN	268492764	Dan Scanlon	G	107
7	2013-12-13	The Hobbit: The Desolation of Smaug	NaN	258366855	Peter Jackson	PG-13	161
8	2013-05-24	Fast & Furious 6	160000000	238679850	Justin Lin	PG-13	130
9	2013-03-08	Oz The Great and Powerful	215000000	234911825	Sam Raimi	PG	127
10	2013-05-16	Star Trek Into Darkness	190000000	228778661	J.J. Abrams	PG-13	123
11	2013-11-08	Thor: The Dark World	170000000	206362140	Alan Taylor	PG-13	120
12	2013-06-21	World War Z	190000000	202359711	Marc Forster	PG-13	116
13	2013-03-22	The Croods	135000000	187168425	Kirk De MicoChris Sanders	PG	98
14	2013-06-28	The Heat	43000000	159582168	Paul Feig	R	117
15	2013-08-07	We're the Millers	37000000	150394119	Rawson Marshall Thurber	R	110
16	2013-12-13	American Hustle	40000000	150117807	David O. Russell	R	138
17	2013-05-10	The Great Gatsby	105000000	144840419	Baz Luhrmann	PG-13	143

如何创建多棵树

使用每个bootstrapped样本构建一棵决策树



样本中的数据分布



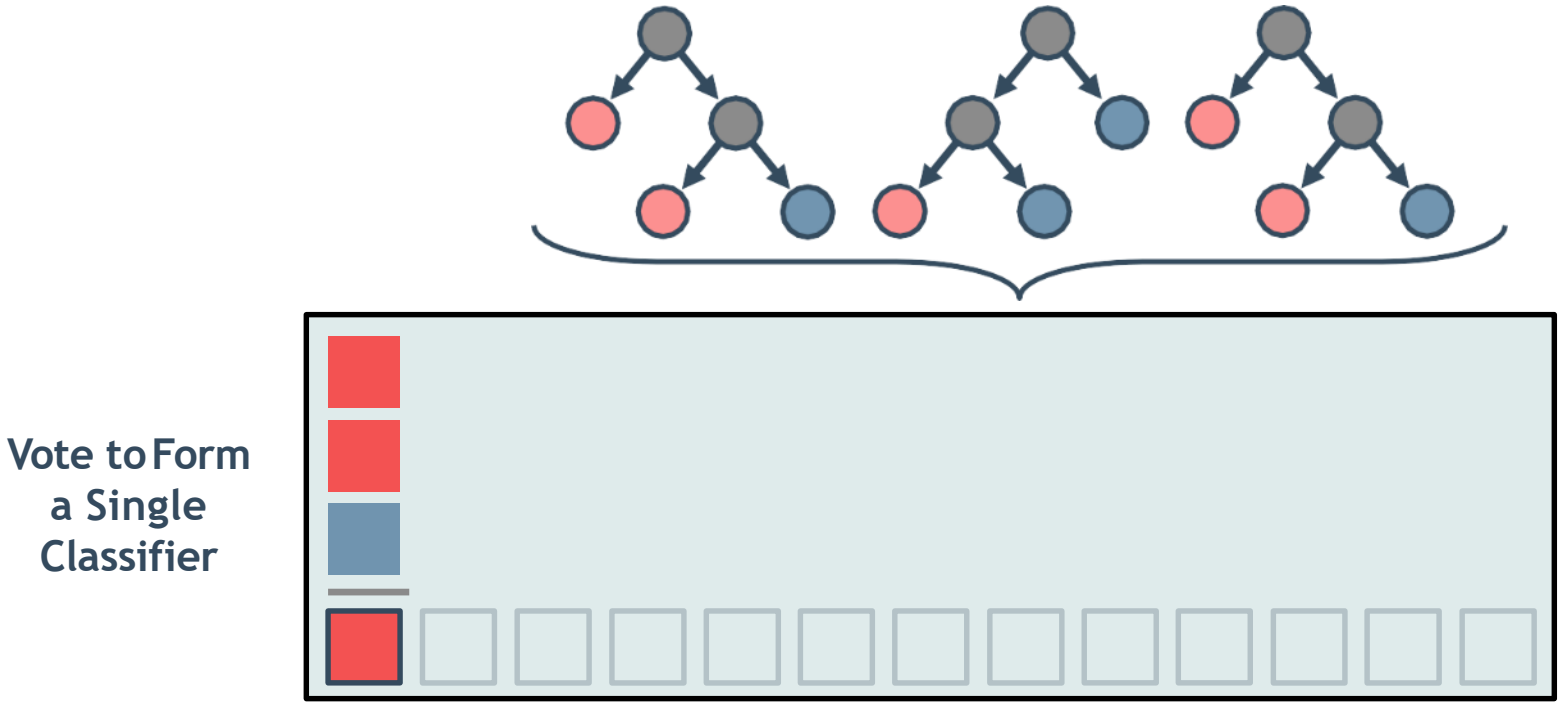
- 给定一个包含n条记录的数据集，创建n个bootstrapped样本
- 对于给定的一条记录 x,

$$P(rec\ x\ not\ selected) = (1 - {}^{1}/_{n})^{n}$$

- 每个bootstrapped样本包含大约2/3条记录

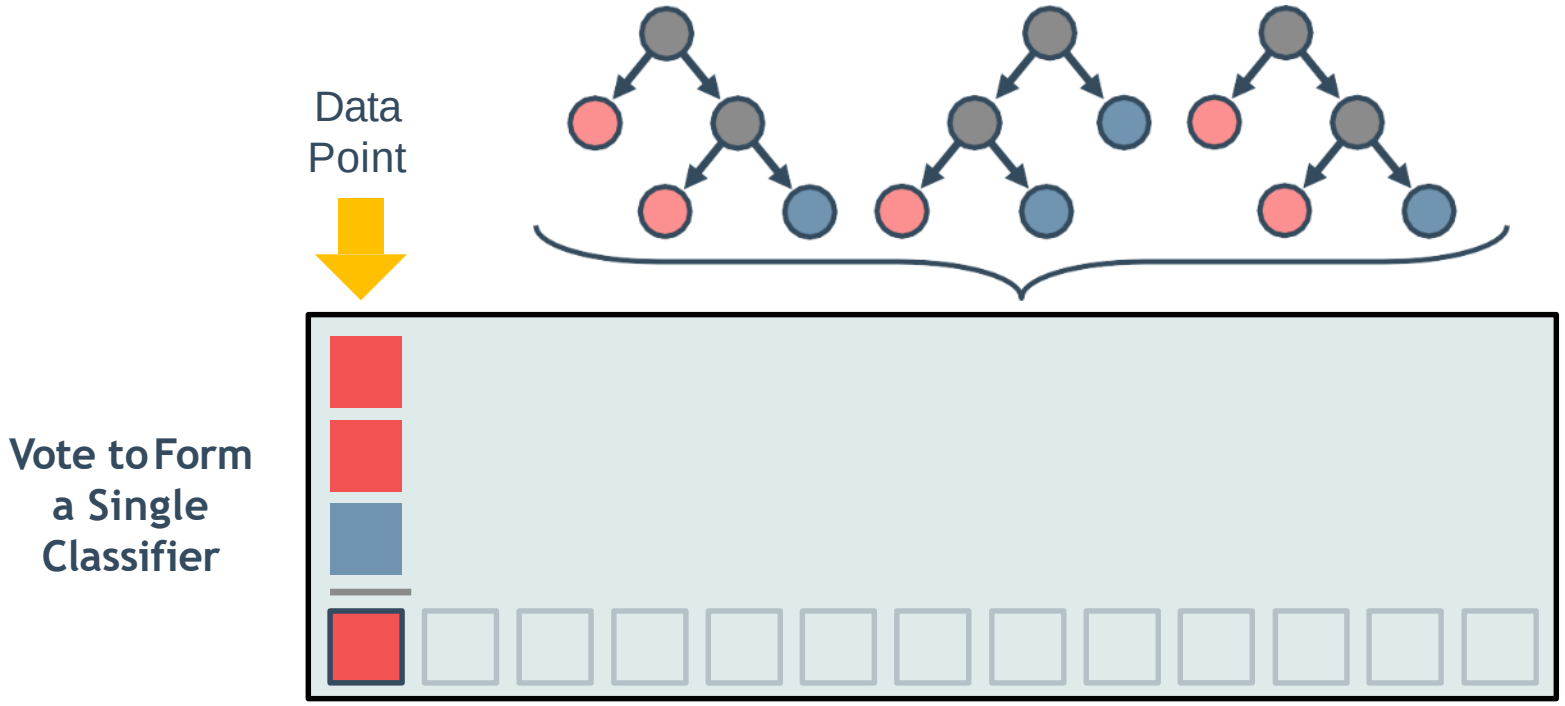
聚集结果

多棵树的结果通过**投票**或者取**平均**得到每个样例点的最终结果



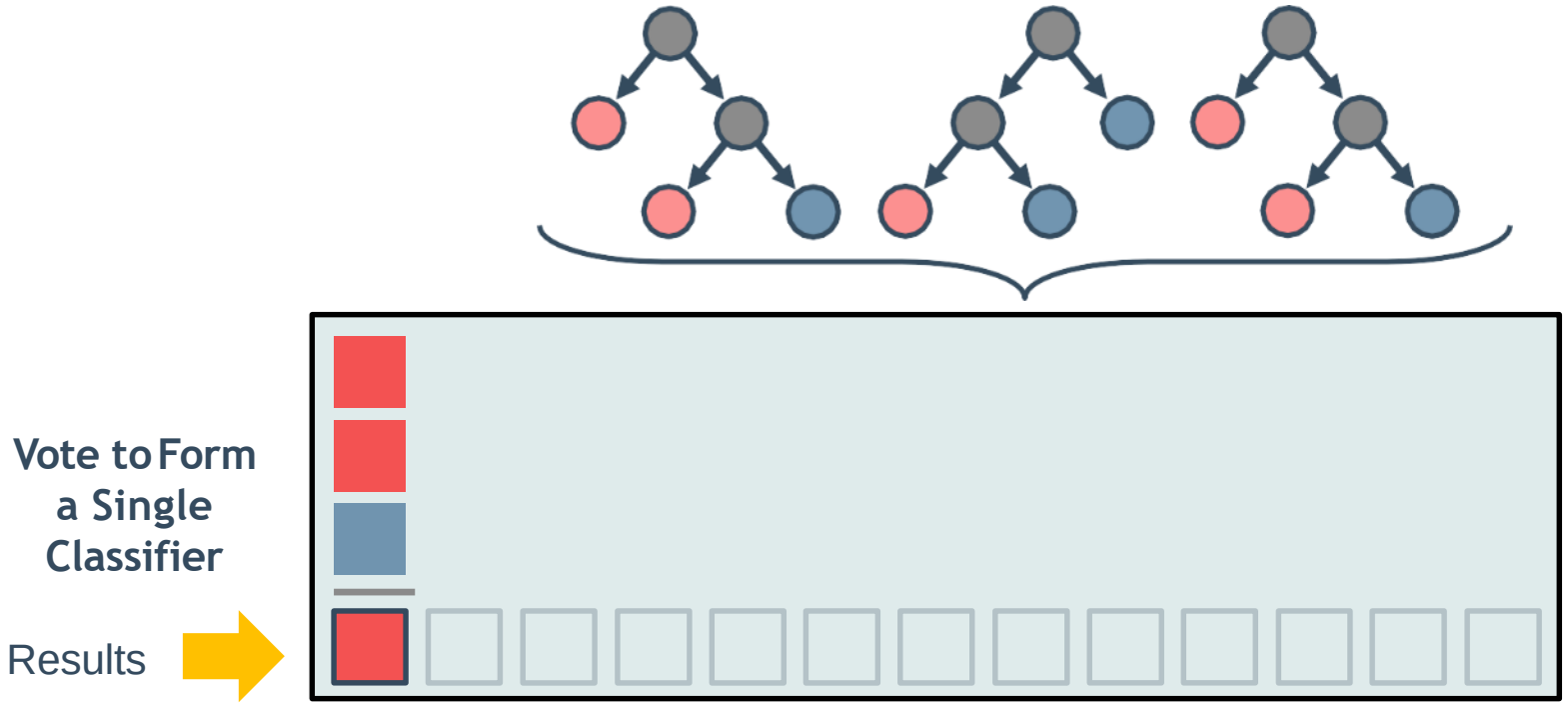
聚集结果

多棵树的结果通过**投票**或者取**平均**得到每个样例点的最终结果



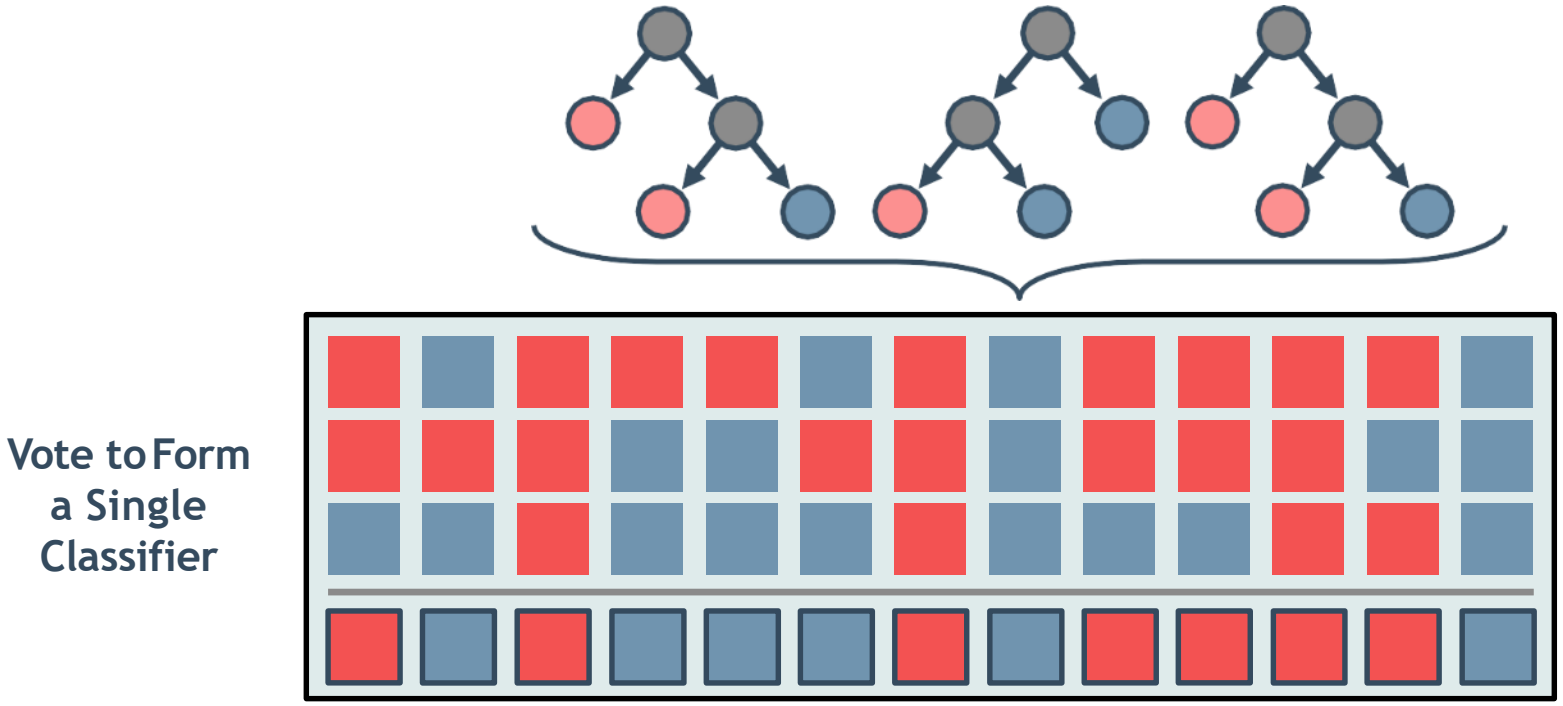
聚集结果

多棵树的结果通过**投票**或者取**平均**得到每个样例点的最终结果



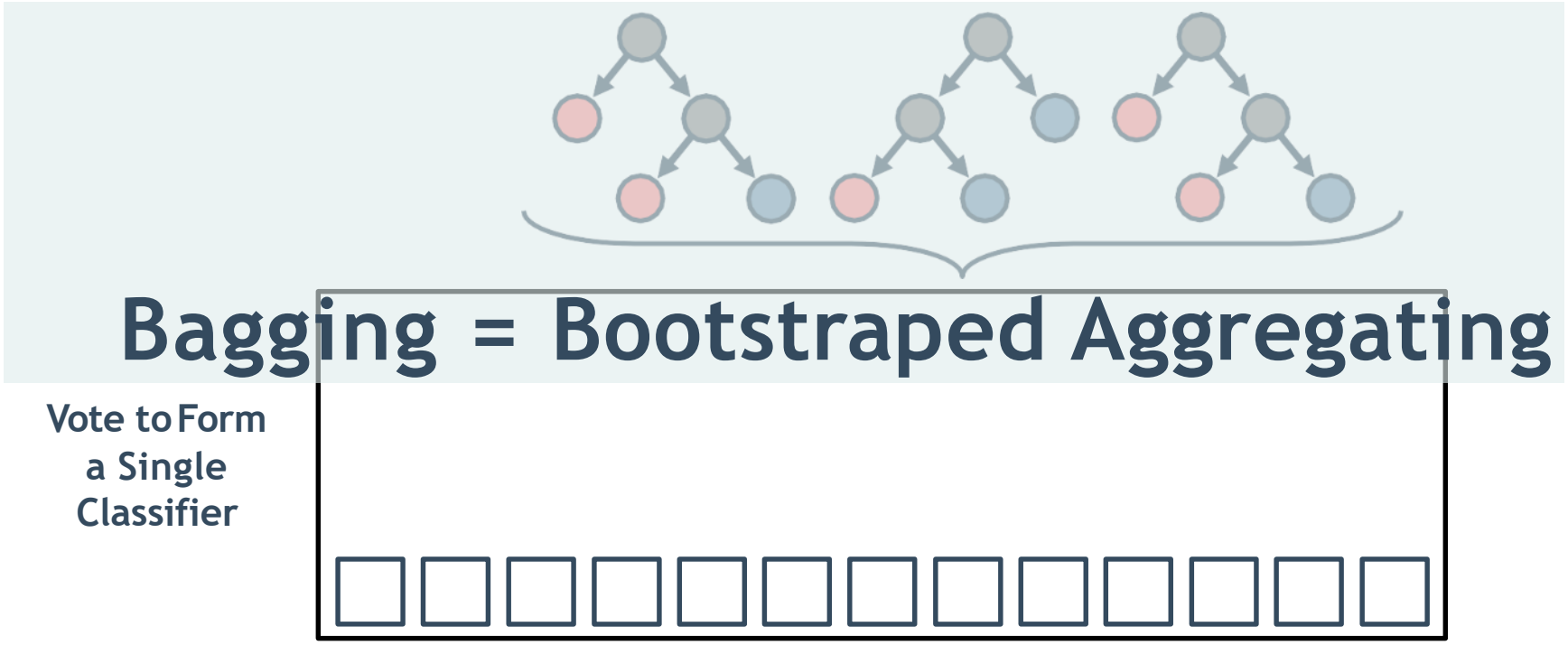
聚集结果

多棵树的结果通过投票或者取平均得到每个样例点的最终结果

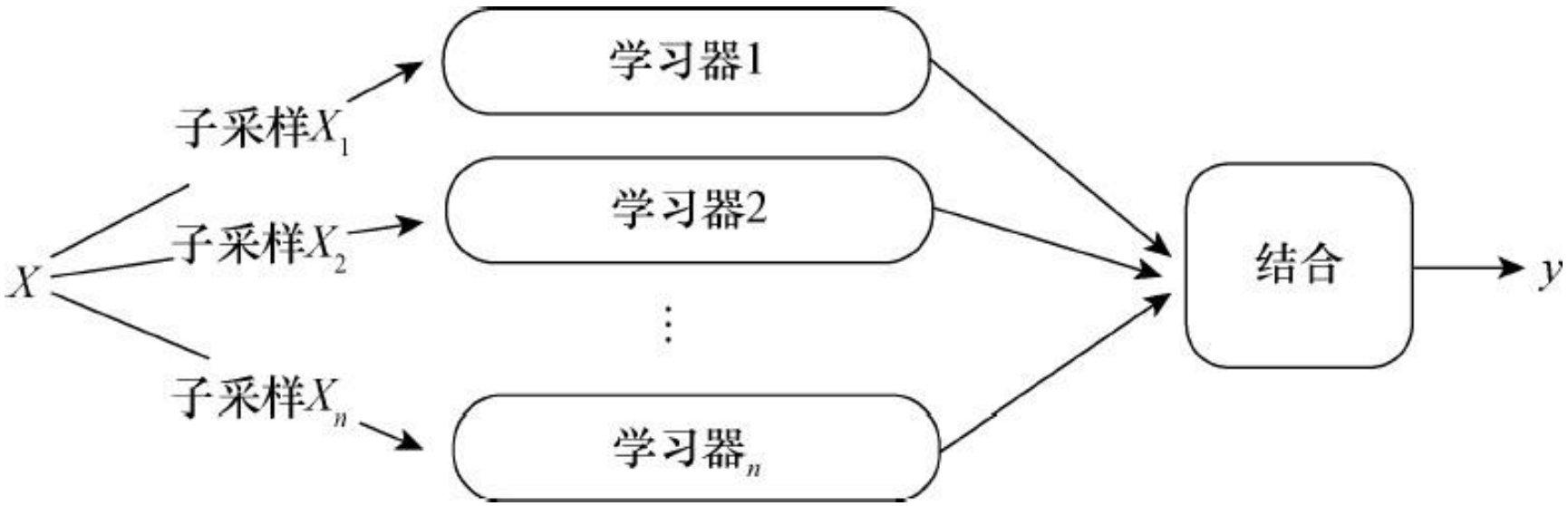


聚集结果

多棵树的结果通过投票或者取平均得到每个样例点的最终结果

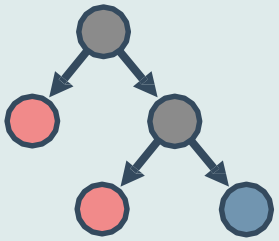


Bagging集成学习

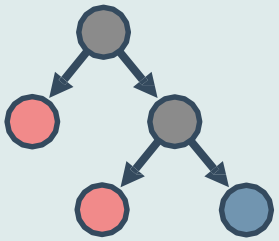


Bagging错误率的计算

	Date	Title	Budget	DomesticTotalGross	Director	Rating	Runtime
0	2013-11-22	The Hunger Games: Catching Fire	130000000	424668047	Francis Lawrence	PG-13	146
1	2013-05-03	Iron Man 3	200000000	409013994	Shane Black	PG-13	129
2	2013-11-22	Frozen	150000000	400738009	Chris BuckJennifer Lee	PG	108
3	2013-07-03	Despicable Me 2	76000000	368061265	Pierre CoffinChris Renaud	PG	98
4	2013-06-14	Man of Steel	225000000	291045518	Zack Snyder	PG-13	143
5	2013-10-04	Gravity	100000000	274092705	Alfonso Cuaron	PG-13	91
6	2013-06-21	Monsters University	NaN	268492764	Dan Scanlon	G	107
7	2013-12-13	The Hobbit: The Desolation of Smaug	NaN	258366855	Peter Jackson	PG-13	161
8	2013-05-24	Fast & Furious 6	160000000	238679850	Justin Lin	PG-13	130
9	2013-03-08	Oz The Great and Powerful	215000000	234911825	Sam Raimi	PG	127
10	2013-05-16	Star Trek Into Darkness	190000000	228778661	J.J. Abrams	PG-13	123
11	2013-11-08	Thor: The Dark World	170000000	206362140	Alan Taylor	PG-13	120
12	2013-06-21	World War Z	190000000	202359711	Marc Forster	PG-13	116
13	2013-03-22	The Croods	135000000	187168425	Kirk De MicoChris Sanders	PG	98
14	2013-06-28	The Heat	43000000	159582188	Paul Feig	R	117
15	2013-08-07	We're the Millers	37000000	150394119	Rawson Marshall Thurber	R	110
16	2013-12-13	American Hustle	40000000	150117807	David O. Russell	R	138
17	2013-05-10	The Great Gatsby	105000000	144840419	Baz Luhrmann	PG-13	143

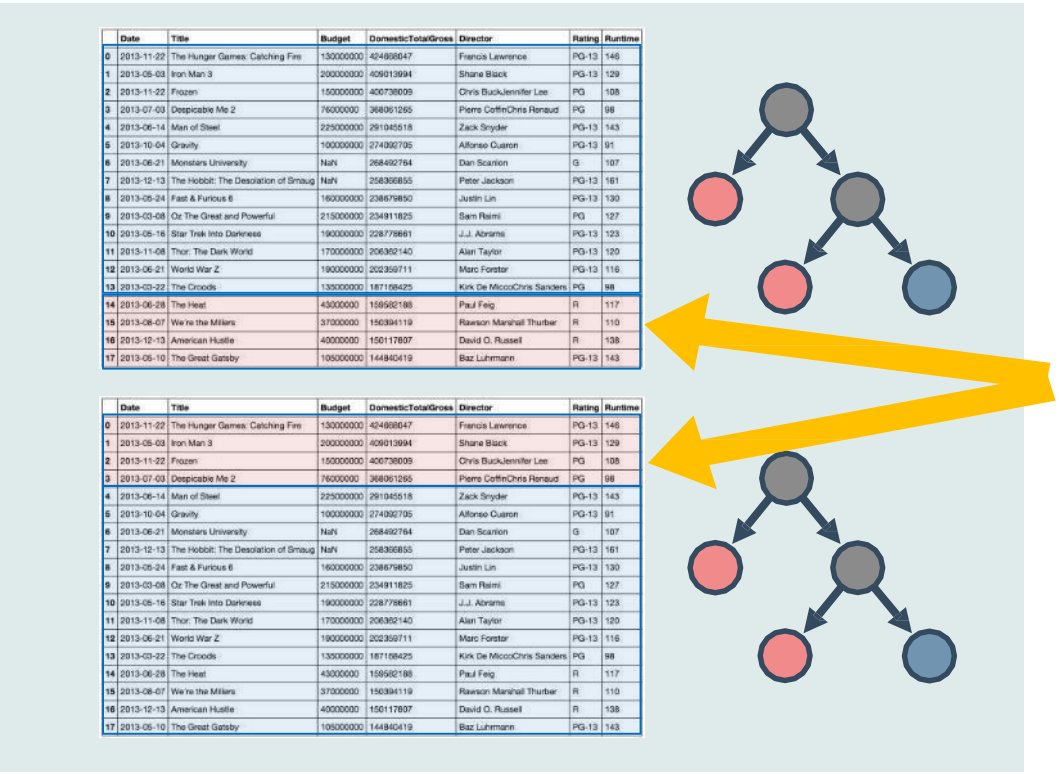


	Date	Title	Budget	DomesticTotalGross	Director	Rating	Runtime
0	2013-11-22	The Hunger Games: Catching Fire	130000000	424668047	Francis Lawrence	PG-13	146
1	2013-05-03	Iron Man 3	200000000	409013994	Shane Black	PG-13	129
2	2013-11-22	Frozen	150000000	400738009	Chris BuckJennifer Lee	PG	108
3	2013-07-03	Despicable Me 2	76000000	368061265	Pierre CoffinChris Renaud	PG	98
4	2013-06-14	Man of Steel	225000000	291045518	Zack Snyder	PG-13	143
5	2013-10-04	Gravity	100000000	274092705	Alfonso Cuaron	PG-13	91
6	2013-06-21	Monsters University	NaN	268492764	Dan Scanlon	G	107
7	2013-12-13	The Hobbit: The Desolation of Smaug	NaN	258366855	Peter Jackson	PG-13	161
8	2013-05-24	Fast & Furious 6	160000000	238679850	Justin Lin	PG-13	130
9	2013-03-08	Oz The Great and Powerful	215000000	234911825	Sam Raimi	PG	127
10	2013-05-16	Star Trek Into Darkness	190000000	228778661	J.J. Abrams	PG-13	123
11	2013-11-08	Thor: The Dark World	170000000	206362140	Alan Taylor	PG-13	120
12	2013-06-21	World War Z	190000000	202359711	Marc Forster	PG-13	116
13	2013-03-22	The Croods	135000000	187168425	Kirk De MicoChris Sanders	PG	98
14	2013-06-28	The Heat	43000000	159582188	Paul Feig	R	117
15	2013-08-07	We're the Millers	37000000	150394119	Rawson Marshall Thurber	R	110
16	2013-12-13	American Hustle	40000000	150117807	David O. Russell	R	138
17	2013-05-10	The Great Gatsby	105000000	144840419	Baz Luhrmann	PG-13	143



- bootstrapped样本为每棵决策树提供了内置的错误率估算
- 在数据子集上创建决策树
- 用未使用的样例来计算那棵树的错误率

Bagging错误率的计算



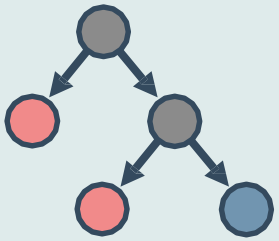
	Date	Title	Budget	DomesticTotalGross	Director	Rating	Runtime
0	2013-11-22	The Hunger Games: Catching Fire	130000000	424668047	Francis Lawrence	PG-13	146
1	2013-05-03	Iron Man 3	200000000	409013994	Shane Black	PG-13	129
2	2013-11-22	Frozen	150000000	400738009	Chris BuckJennifer Lee	PG	108
3	2013-07-03	Despicable Me 2	76000000	368061265	Pierre CoffinChris Renaud	PG	98
4	2013-06-14	Man of Steel	225000000	291045518	Zack Snyder	PG-13	143
5	2013-10-04	Gravity	100000000	274092705	Alfonso Cuaron	PG-13	91
6	2013-06-21	Monsters University	NaN	268492764	Dan Scanlon	G	107
7	2013-12-13	The Hobbit: The Desolation of Smaug	NaN	258366855	Peter Jackson	PG-13	161
8	2013-05-24	Fast & Furious 6	160000000	238679850	Justin Lin	PG-13	130
9	2013-03-08	Oz The Great and Powerful	215000000	234911825	Sam Raimi	PG	127
10	2013-05-16	Star Trek Into Darkness	190000000	228778661	J.J. Abrams	PG-13	123
11	2013-11-08	Thor: The Dark World	170000000	206362140	Alan Taylor	PG-13	120
12	2013-06-21	World War Z	190000000	202359711	Marc Forster	PG-13	116
13	2013-03-22	The Croods	135000000	187168425	Kirk De MiccoChris Sanders	PG	98
14	2013-06-28	The Heat	43000000	159582188	Paul Feig	R	117
15	2013-08-07	We're the Millers	37000000	150394119	Rawson Marshall Thurber	R	110
16	2013-12-13	American Hustle	40000000	150117807	David O. Russell	R	138
17	2013-05-10	The Great Gatsby	105000000	144840419	Baz Luhrmann	PG-13	143

	Date	Title	Budget	DomesticTotalGross	Director	Rating	Runtime
0	2013-11-22	The Hunger Games: Catching Fire	130000000	424668047	Francis Lawrence	PG-13	146
1	2013-05-03	Iron Man 3	200000000	409013994	Shane Black	PG-13	129
2	2013-11-22	Frozen	150000000	400738009	Chris BuckJennifer Lee	PG	108
3	2013-07-03	Despicable Me 2	76000000	368061265	Pierre CoffinChris Renaud	PG	98
4	2013-06-14	Man of Steel	225000000	291045518	Zack Snyder	PG-13	143
5	2013-10-04	Gravity	100000000	274092705	Alfonso Cuaron	PG-13	91
6	2013-06-21	Monsters University	NaN	268492764	Dan Scanlon	G	107
7	2013-12-13	The Hobbit: The Desolation of Smaug	NaN	258366855	Peter Jackson	PG-13	161
8	2013-05-24	Fast & Furious 6	160000000	238679850	Justin Lin	PG-13	130
9	2013-03-08	Oz The Great and Powerful	215000000	234911825	Sam Raimi	PG	127
10	2013-05-16	Star Trek Into Darkness	190000000	228778661	J.J. Abrams	PG-13	123
11	2013-11-08	Thor: The Dark World	170000000	206362140	Alan Taylor	PG-13	120
12	2013-06-21	World War Z	190000000	202359711	Marc Forster	PG-13	116
13	2013-03-22	The Croods	135000000	187168425	Kirk De MiccoChris Sanders	PG	98
14	2013-06-28	The Heat	43000000	159582188	Paul Feig	R	117
15	2013-08-07	We're the Millers	37000000	150394119	Rawson Marshall Thurber	R	110
16	2013-12-13	American Hustle	40000000	150117807	David O. Russell	R	138
17	2013-05-10	The Great Gatsby	105000000	144840419	Baz Luhrmann	PG-13	143

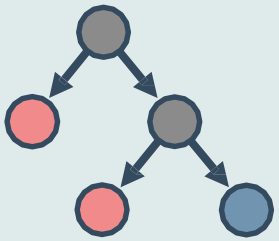
- bootstrapped样本为每棵决策树提供了内置的错误率估算
- 在数据子集上创建决策树
- 用未使用的样例来计算那棵树的错误率

Bagging错误率的计算

	Date	Title	Budget	DomesticTotalGross	Director	Rating	Runtime
0	2013-11-22	The Hunger Games: Catching Fire	130000000	424668047	Francis Lawrence	PG-13	146
1	2013-05-03	Iron Man 3	200000000	409013994	Shane Black	PG-13	129
2	2013-11-22	Frozen	150000000	400738009	Chris BuckJennifer Lee	PG	108
3	2013-07-03	Despicable Me 2	76000000	368061265	Pierre CoffinChris Renaud	PG	98
4	2013-06-14	Man of Steel	225000000	291045518	Zack Snyder	PG-13	143
5	2013-10-04	Gravity	100000000	274092705	Alfonso Cuaron	PG-13	91
6	2013-06-21	Monsters University	NaN	268492764	Dan Scanlon	G	107
7	2013-12-13	The Hobbit: The Desolation of Smaug	NaN	258366855	Peter Jackson	PG-13	161
8	2013-05-24	Fast & Furious 6	160000000	238679850	Justin Lin	PG-13	130
9	2013-03-08	Oz The Great and Powerful	215000000	234911825	Sam Raimi	PG	127
10	2013-05-16	Star Trek Into Darkness	190000000	228778661	J.J. Abrams	PG-13	123
11	2013-11-08	Thor: The Dark World	170000000	206362140	Alan Taylor	PG-13	120
12	2013-06-21	World War Z	190000000	202359711	Marc Forster	PG-13	116
13	2013-03-22	The Croods	135000000	187168425	Kirk De MiccoChris Sanders	PG	98
14	2013-06-28	The Heat	43000000	159582188	Paul Feig	R	117
15	2013-08-07	We're the Millers	37000000	150394119	Rawson Marshall Thurber	R	110
16	2013-12-13	American Hustle	40000000	150117807	David O. Russell	R	138
17	2013-05-10	The Great Gatsby	105000000	144840419	Baz Luhrmann	PG-13	143



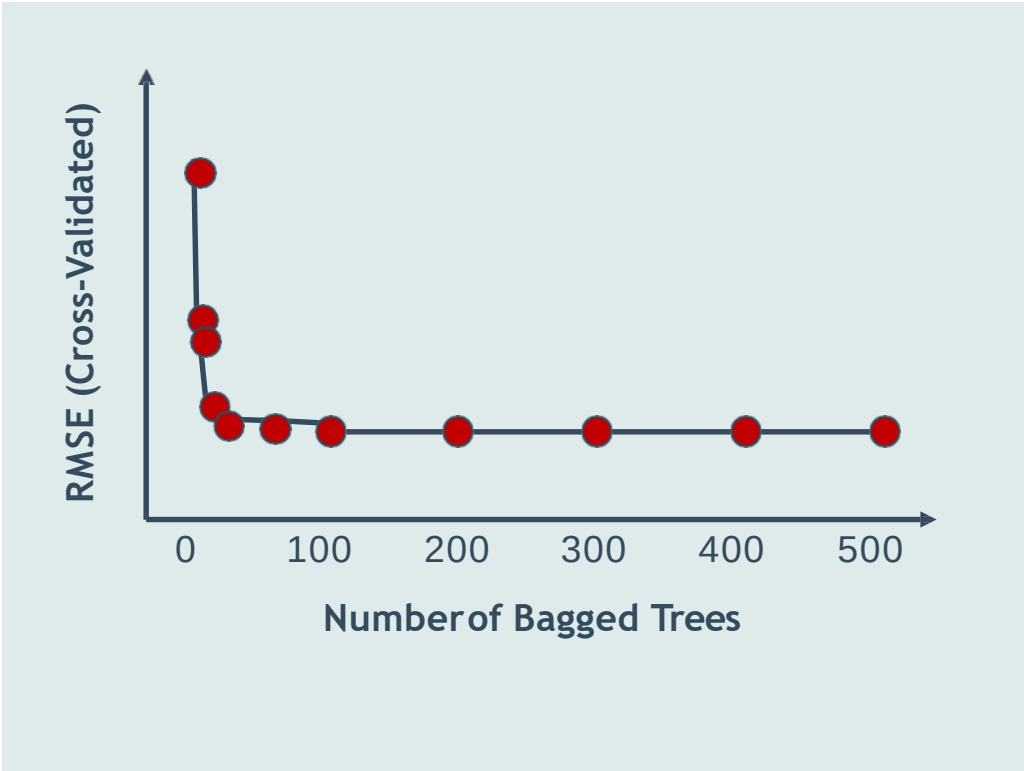
	Date	Title	Budget	DomesticTotalGross	Director	Rating	Runtime
0	2013-11-22	The Hunger Games: Catching Fire	130000000	424668047	Francis Lawrence	PG-13	146
1	2013-05-03	Iron Man 3	200000000	409013994	Shane Black	PG-13	129
2	2013-11-22	Frozen	150000000	400738009	Chris BuckJennifer Lee	PG	108
3	2013-07-03	Despicable Me 2	76000000	368061265	Pierre CoffinChris Renaud	PG	98
4	2013-06-14	Man of Steel	225000000	291045518	Zack Snyder	PG-13	143
5	2013-10-04	Gravity	100000000	274092705	Alfonso Cuaron	PG-13	91
6	2013-06-21	Monsters University	NaN	268492764	Dan Scanlon	G	107
7	2013-12-13	The Hobbit: The Desolation of Smaug	NaN	258366855	Peter Jackson	PG-13	161
8	2013-05-24	Fast & Furious 6	160000000	238679850	Justin Lin	PG-13	130
9	2013-03-08	Oz The Great and Powerful	215000000	234911825	Sam Raimi	PG	127
10	2013-05-16	Star Trek Into Darkness	190000000	228778661	J.J. Abrams	PG-13	123
11	2013-11-08	Thor: The Dark World	170000000	206362140	Alan Taylor	PG-13	120
12	2013-06-21	World War Z	190000000	202359711	Marc Forster	PG-13	116
13	2013-03-22	The Croods	135000000	187168425	Kirk De MiccoChris Sanders	PG	98
14	2013-06-28	The Heat	43000000	159582188	Paul Feig	R	117
15	2013-08-07	We're the Millers	37000000	150394119	Rawson Marshall Thurber	R	110
16	2013-12-13	American Hustle	40000000	150117807	David O. Russell	R	138
17	2013-05-10	The Great Gatsby	105000000	144840419	Baz Luhrmann	PG-13	143



- bootstrapped样本为每棵决策树提供了内置的错误率估算
- 在数据子集上创建决策树
- 用未使用的样例来计算那棵树的错误率
- 称作“袋外（out-of-bag）”错误率

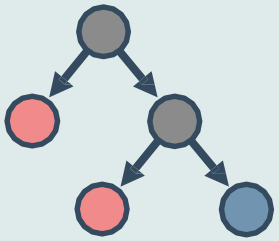
拟合多少棵树？

- Bagging模型的性能随着树的数目增大而改进
- 一般在大约50棵树时获得最大的改进

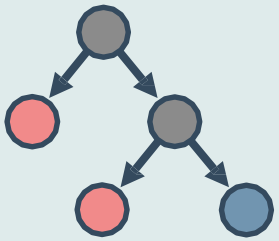


Bagging的优势

	Date	Title	Budget	DomesticTotalGross	Director	Rating	Runtime
0	2013-11-22	The Hunger Games: Catching Fire	130000000	424668047	Francis Lawrence	PG-13	146
1	2013-05-03	Iron Man 3	200000000	409013994	Shane Black	PG-13	129
2	2013-11-22	Frozen	150000000	400738009	Chris BuckJennifer Lee	PG	108
3	2013-07-03	Despicable Me 2	76000000	368061265	Pierre CoffinChris Renaud	PG	98
4	2013-06-14	Man of Steel	225000000	291045518	Zack Snyder	PG-13	143
5	2013-10-04	Gravity	100000000	274092705	Alfonso Cuaron	PG-13	91
6	2013-06-21	Monsters University	NaN	268492764	Dan Scanlon	G	107
7	2013-12-13	The Hobbit: The Desolation of Smaug	NaN	258366855	Peter Jackson	PG-13	161
8	2013-05-24	Fast & Furious 6	160000000	238679850	Justin Lin	PG-13	130
9	2013-03-08	Oz The Great and Powerful	215000000	234911825	Sam Raimi	PG	127
10	2013-05-16	Star Trek Into Darkness	190000000	228778661	J.J. Abrams	PG-13	123
11	2013-11-08	Thor: The Dark World	170000000	206362140	Alan Taylor	PG-13	120
12	2013-06-21	World War Z	190000000	202359711	Marc Forster	PG-13	116
13	2013-03-22	The Croods	135000000	187168425	Kirk De MiccoChris Sanders	PG	98
14	2013-06-28	The Heat	43000000	159582188	Paul Feig	R	117
15	2013-08-07	We're the Millers	37000000	150394119	Rawson Marshall Thurber	R	110
16	2013-12-13	American Hustle	40000000	150117807	David O. Russell	R	138
17	2013-05-10	The Great Gatsby	105000000	144840419	Baz Luhrmann	PG-13	143



	Date	Title	Budget	DomesticTotalGross	Director	Rating	Runtime
0	2013-11-22	The Hunger Games: Catching Fire	130000000	424668047	Francis Lawrence	PG-13	146
1	2013-05-03	Iron Man 3	200000000	409013994	Shane Black	PG-13	129
2	2013-11-22	Frozen	150000000	400738009	Chris BuckJennifer Lee	PG	108
3	2013-07-03	Despicable Me 2	76000000	368061265	Pierre CoffinChris Renaud	PG	98
4	2013-06-14	Man of Steel	225000000	291045518	Zack Snyder	PG-13	143
5	2013-10-04	Gravity	100000000	274092705	Alfonso Cuaron	PG-13	91
6	2013-06-21	Monsters University	NaN	268492764	Dan Scanlon	G	107
7	2013-12-13	The Hobbit: The Desolation of Smaug	NaN	258366855	Peter Jackson	PG-13	161
8	2013-05-24	Fast & Furious 6	160000000	238679850	Justin Lin	PG-13	130
9	2013-03-08	Oz The Great and Powerful	215000000	234911825	Sam Raimi	PG	127
10	2013-05-16	Star Trek Into Darkness	190000000	228778661	J.J. Abrams	PG-13	123
11	2013-11-08	Thor: The Dark World	170000000	206362140	Alan Taylor	PG-13	120
12	2013-06-21	World War Z	190000000	202359711	Marc Forster	PG-13	116
13	2013-03-22	The Croods	135000000	187168425	Kirk De MiccoChris Sanders	PG	98
14	2013-06-28	The Heat	43000000	159582188	Paul Feig	R	117
15	2013-08-07	We're the Millers	37000000	150394119	Rawson Marshall Thurber	R	110
16	2013-12-13	American Hustle	40000000	150117807	David O. Russell	R	138
17	2013-05-10	The Great Gatsby	105000000	144840419	Baz Luhrmann	PG-13	143



同决策树：

- 易于解释和实现
- 输入数据可以是异构的，不要求预处理

特有的：

- 比决策树的方差低
- 可以并行地构建多棵树

Bagging分类器的语法

导入包含分类方法的类：

```
from sklearn.ensemble import BaggingClassifier
```

创建该类的一个对象：

```
BC = BaggingClassifier(n_estimators=50)
```

拟合训练数据，并预测：

```
BC = BC.fit(X_train, y_train)
y_predict = BC.predict(X_test)
```

使用交叉验证调参。回归用**BaggingRegressor**。

http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingClassifier.html

BaggingClassifier参数设置

base_estimator	基学习器 在采样得到的数据子集上训练的基学习器。 缺省值是None。等于None时，则使用决策树做基学习器。
n_estimators	基学习器个数 用于集成的基学习器的个数。 缺省值是10。
max_samples	最大样例个数 从训练集中随机抽取的，用于训练每个基学习器的样例数目 缺省值是1.0。整数值表示抽取样例的个数；小数值表示抽取样例数占训练集的总样例数的比。
max_features	最大特征数 从训练集中随机抽取的，用于训练每个基学习器的特征数目 缺省值是1.0。整数值表示抽取特征的个数；小数值表示抽取特征数占训练集的总特征数的比。

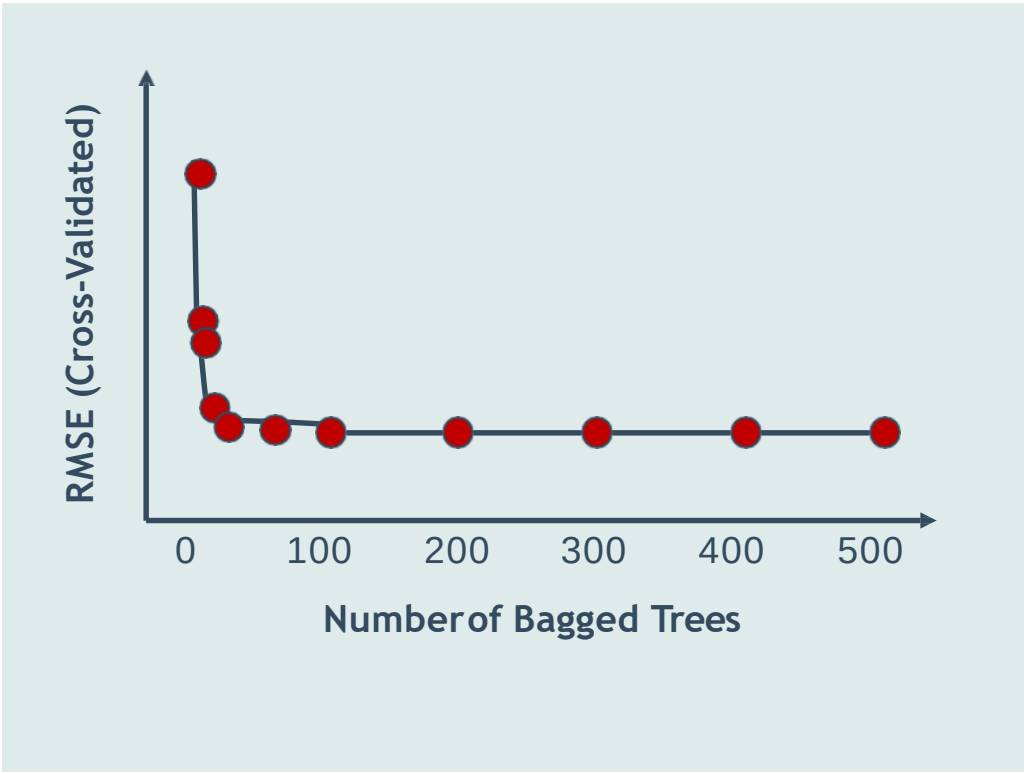
Bagging减少的方差

- 有 n 棵独立的决策树，每棵树的方差是 σ^2，则bagged方差是：

$$\frac{\sigma^2}{n}$$

- 然而，bootstrap样本是相关的（ρ）：

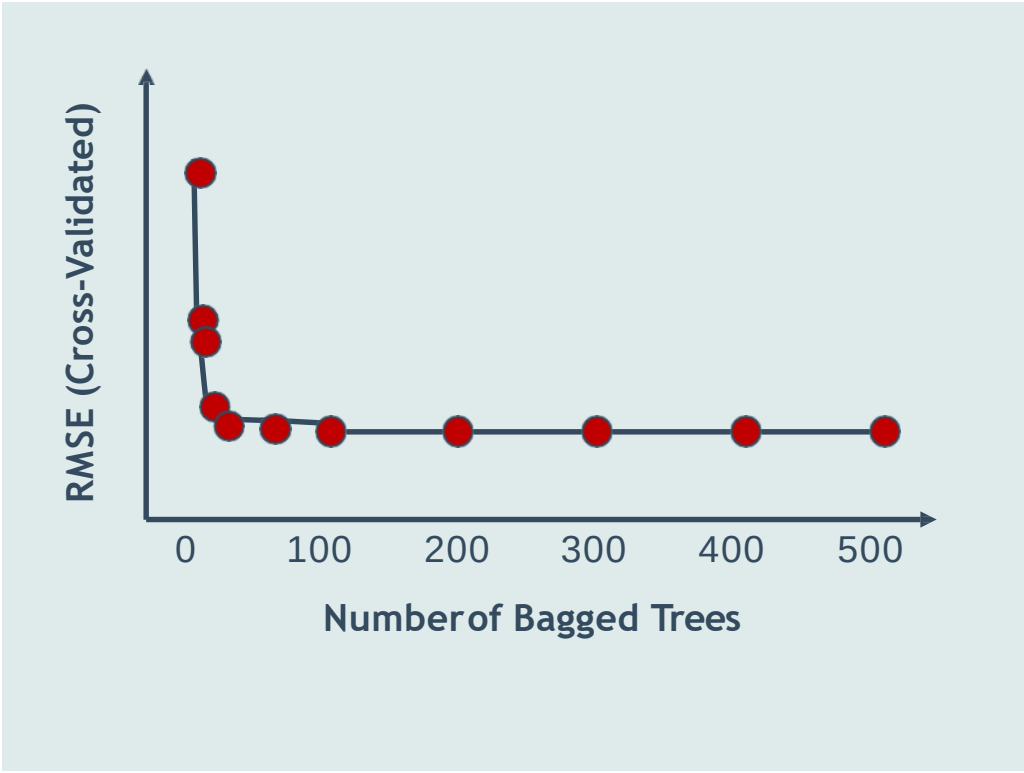
$$\rho\sigma^2 + \frac{1-\rho}{n}\sigma^2$$



引入更多的随机性

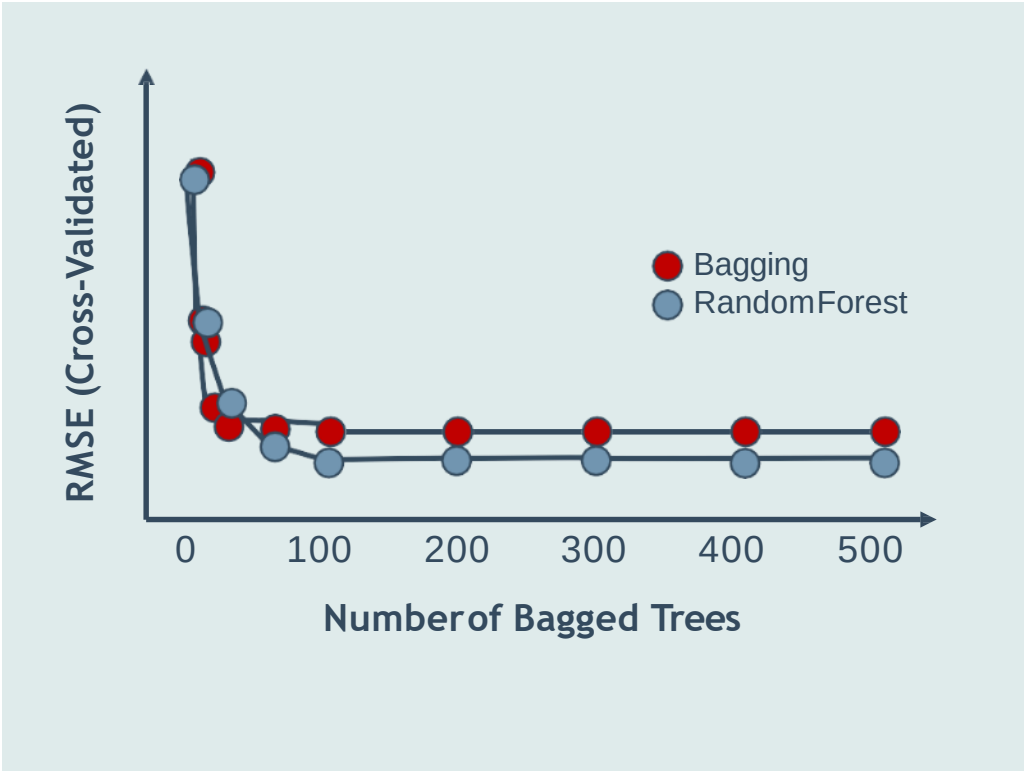
- 解决方案：进一步消除树之间的相关性
- 每棵树使用一组随机抽取的特征：
  - 分类：$\sqrt{m}$
  - 回归：$m/3$

Why ?



引入更多的随机性

- 解决方案：进一步消除树之间的相关性
- 每棵树使用一组随机抽取的特征：
  - 分类：$\sqrt{m}$
  - 回归：$m/3$
- 称作“随机森林”



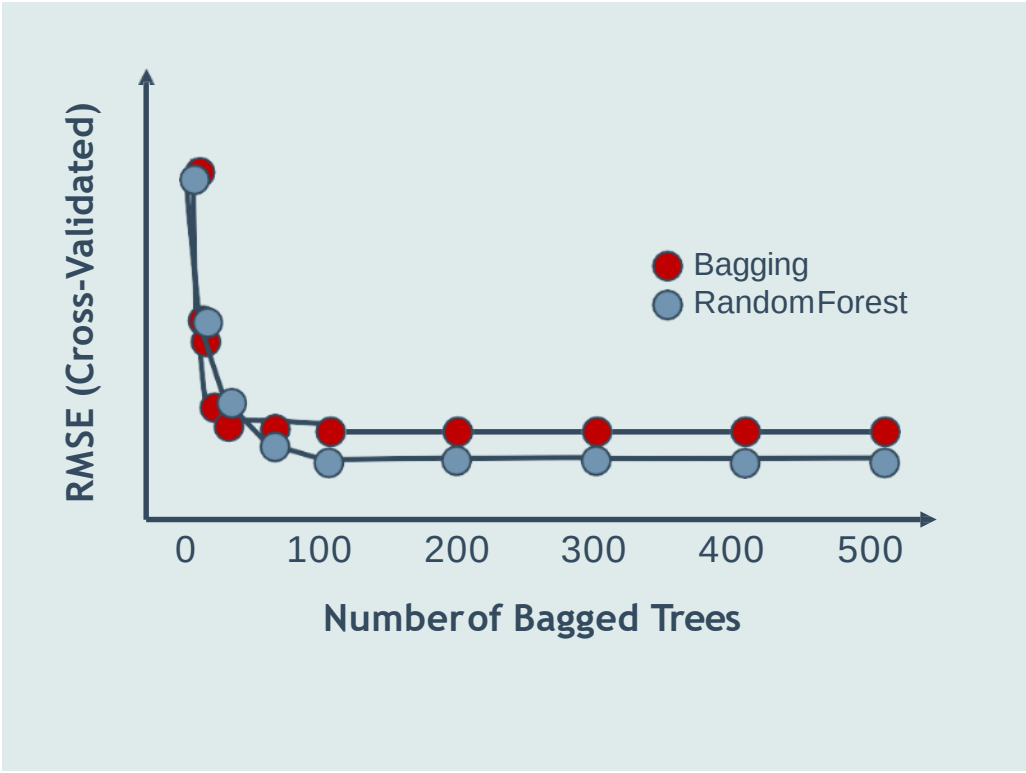
随机森林

李奥·布瑞曼
Leo Breiman
1928–2005

- 二十世纪伟大的统计学家
- 创建CART决策树
- 对集成学习三大贡献：
  - Bagging、
  - 随机森林
  - 关于Boosting的理论探讨

随机森林多少棵树？

- 相对于Bagging，随机森林的错误率会进一步降低。
- 增加足够的树直到错误率不再变化为止
- 新增加树不会改善结果了。



随机森林的语法

导入包含分类方法的类：

```
from sklearn.ensemble import RandomForestClassifier
```

创建该类的一个对象：

```
RC = RandomForestClassifier(n_estimators=100, max_features=10)
```

拟合训练数据，并预测：

```
RC = RC.fit(X_train, y_train)
y_predict = RC.predict(X_test)
```

使用交叉验证调参数。回归用RandomForestRegressor。

http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

随机森林参数设置

n_estimators	基学习器个数，森林中树的个数。 缺省值是10。
criterion	分裂条件选择标准 RandomForestClassifier的缺省值是“gini”，即用基尼指数作为衡量指标，也可以是“entropy”，即使用熵作为衡量指标； RandomForestRegressor的缺省值是“mse”，即使用均方差作为衡量指标，也可以是“mae”，即使用平均绝对值误差作为衡量指标
max_features	选择分裂条件时考虑的最大特征数 有以下多种可能的取值： ★ 整数，即可考虑的特征的最多个数； ★ 小数，即可考虑的特征的最多个数占总的特征数的比例； ★ “auto”，即最多考虑（√总的特征数）个特征； ★ “log2”，即最多考虑（$\log_2$ 总的特征数）个特征； ★ “sqrt”，即最多考虑（√总的特征数）个特征； 缺省值是“auto”。 直到找到一个有效的分裂为止，即使需要考虑多于max_features个特征。
max_depth	决策树的最大深度 缺省值是None。如果为None，则节点会一直分裂下去直到每个叶子节点都纯了或者叶子节点包含的样例个数少于min_samples_split

引入更多的随机性

- 有时需要比随机森林更多的随机性
- 解决方案：随机选择特征，并创建随机划分---即不使用贪婪划分
- 称作“超随机森林”

超随机森林分类器的语法

导入包含分类方法的类：

```
from sklearn.ensemble import ExtraTreesClassifier
```

创建该类的一个对象：

```
EC = ExtraTreesClassifier(n_estimators=100, max_features=10)
```

拟合训练数据，并预测：

```
EC = EC.fit(X_train, y_train)
y_predict = EC.predict(X_test)
```

使用交叉验证调参。回归用ExtraTreesRegressor。

http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html

BOOSTING
提升

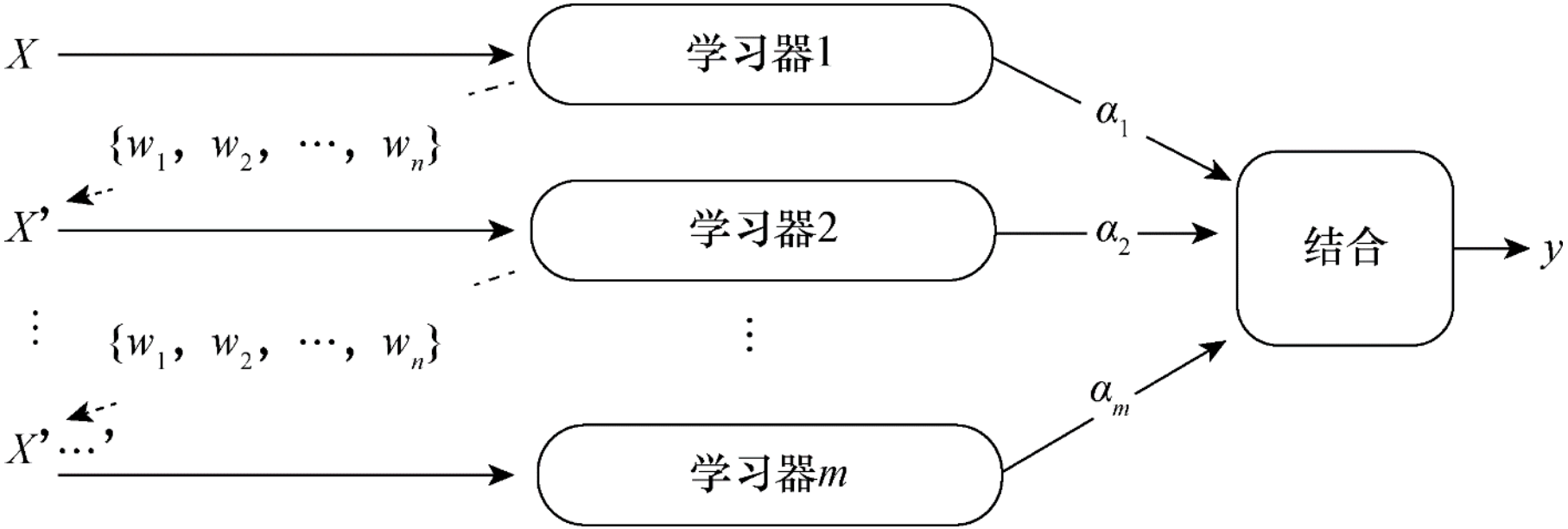
起源

- PAC（Probably Approximately Correct）学习框架
- Valiant & Kearns (1984)
  - 强可学习:一个概念（或类）如果存在一个多项式时间内的学习算法能够学习它，并且正确率很高
  - 弱可学习
- 瓦利安特Valiant & 卡恩斯Kearns (1989)
  - 证明了弱可学习和强可学习的等价性
- 夏皮罗Schapire (1990)
  - 第一次提出一个多项式时间的Boosting算法
- 弗罗因德Freund (1991)
  - 提出一种效率更高的Boosting算法
- Freund & Schapire (1995)
  - 提出**AdaBoost**（Adaptive Boosting）算法
- 弗里德曼Friedman (1999)
  - 提出**梯度提升**（Gradient Boosting）算法

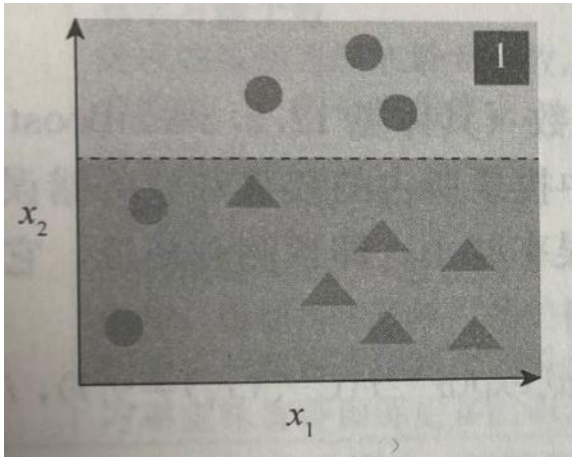
Boosting模型的基本思路

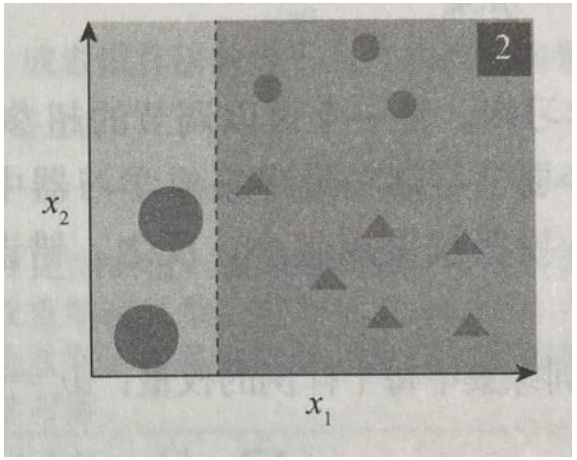
- 相继地训练多个弱学习器，每个弱学习器力图纠正前面学习器的错误，最后将多个弱学习器加权结合起来。

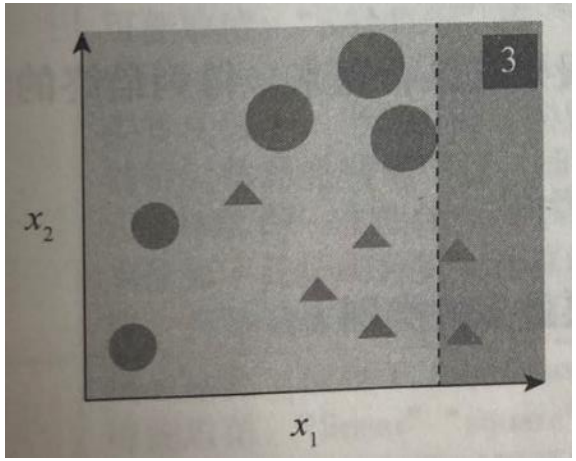
AdaBoost

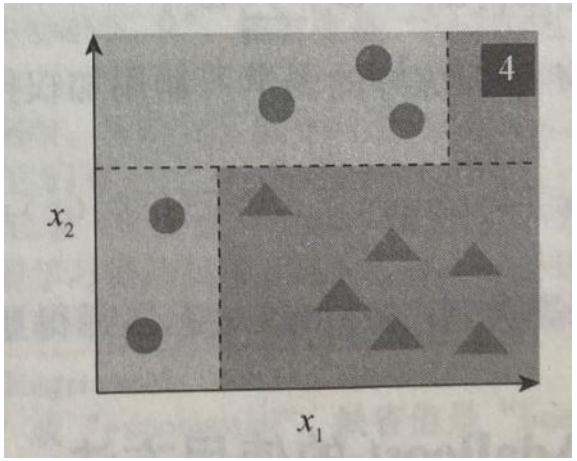

X
学习器1
α_1
$\{w_1, w_2, \cdots, w_n\}$
X'
学习器2
α_2
结合
y
$\{w_1, w_2, \cdots, w_n\}$
α_m
X'
学习器m

Boosting概述









AdaBoost伪码

1. 给定包含n个样本的训练集 $\{x_1, x_2, \ldots, x_n\}$，初始化每个样本的权重$w_i = 1/n$。
2. 分别设$m = 1, 2, \ldots, M$，重复下面的步骤：
   a）在当前训练集上训练一个弱学习器：$C^{(m)}$
   b）用训练出的弱学习器预测训练集中每个样例的类别：$C^{(m)}(x_i)$
   c）计算该弱学习器的加权错误率：

$$err^{(m)} = \frac{\sum_{i=1}^{n} \omega_i I(C^{(m)}(x_i) \neq y_i)}{\sum_{i=1}^{n} \omega_i}$$

   其中$I(C^{(m)}(x_i) \neq y_i)$是指示函数，$C^{(m)}(x_i) \neq y_i$成立时，值为1，否则为0。

   d) 计算该弱学习器的权重值：

$$\alpha^{(m)} = \eta \log \frac{(1 - err^{(m)})}{err^{(m)}}$$

   其中η是学习率。

   e) 修改训练集中每个样例的权重：$\omega_i = \omega_i \exp(\alpha^{(m)} I(C^{(m)}(x_i) \neq y_i)), i = 1, 2, \ldots, n$

   f) 归一化ω_i: $\omega_i = \omega_i / \sum_{i=1}^{n} \omega_i$
3. 将M个训练好的弱学习器用加权投票法结合起来，得到最终的集成学习器C：

$$C(x) = \underset{k}{\operatorname{argmax}} \sum_{m=1}^{M} \alpha^{(m)} I(C^{(m)}(x) = k)$$

AdaBoost参数设置

base_estimator	基学习器 缺省是使用决策树做基学习器。要求基学习器支持样本加权。
n_estimators	基学习器个数 缺省值是50。集成的基学习器的最大个数。或者说算法的最大迭代次数，如果已 经完美拟合数据了，则会提前停止。
learning_rate	学习率 缺省值是1.0。一般设为0到1之间的值，收缩每个基学习器对最终结果的贡献量，也即每个基学习器的权重缩减系数。它和n_estimators之间存在一个折衷，一起来决定算法的拟合效果。如果要达到一定的拟合效果，更小的学习率意味着要训练更多的弱学习器。
algorithm	AdaBoost分类算法（仅用于AdaBoostClassifier） 可能取值：“SAMME”或“SAMME.R”，缺省值是“SAMME.R”。 Scikit-Learn中实现的AdaBoost分类算法实际是一个被称作SAMME的多分类算法版本，当只有两个类别时，SAMME就等同于AdaBoost。如果基学习器可以估算类别概率值（即它们有一个predict_prob()方法），则可以使用SAMME的一个变体SAMME.R（R表示“real”，即实数），它基于类别概率值而不是类别标签来计算弱学习器的权重。通常它比SAMME算法收敛更快，且测试错误率更低，但要求基学习器必须支持类别概率值的计算。
loss	损失函数（仅用于AdaBoostRegressor） 可能取值：“linear”，“square”或“exponential”，缺省值是“linear”。 每一轮调整样本权重时需要用到的损失函数，用于计算每一轮训练出的弱学习器 在训练集中每个样本上的预测值与真实值之间的误差，可以是线性误差，平方误 差或指数误差。

AdaBoost分类器的语法

导入包含分类方法的类：

```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
```

基学习器可以被手工设置

创建该类的一个对象：

```
ABC = AdaBoostClassifier(base_estimator=DecisionTreeClassifier(),
        learning_rate=0.1, n_estimators=200)
```

AdaBoost分类器的语法

导入包含分类方法的类：

```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
```

这里也可以设置最大深度

创建该类的一个对象：

```
ABC = AdaBoostClassifier(base_estimator=DecisionTreeClassifier(),
        learning_rate=0.1, n_estimators=200)
```

AdaBoost分类器的语法

导入包含该分类方法的类：

```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
```

创建该类的一个对象：

```
ABC = AdaBoostClassifier(base_estimator=DecisionTreeClassifier(),
          learning_rate=0.1, n_estimators=200)
```

拟合训练数据，并预测：

```
ABC = ABC.fit (X_train, y_train)

y_predict = ABC.predict(X_test)
```

使用交叉验证调节参数。回归用**AdaBoostRegressor**

http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html

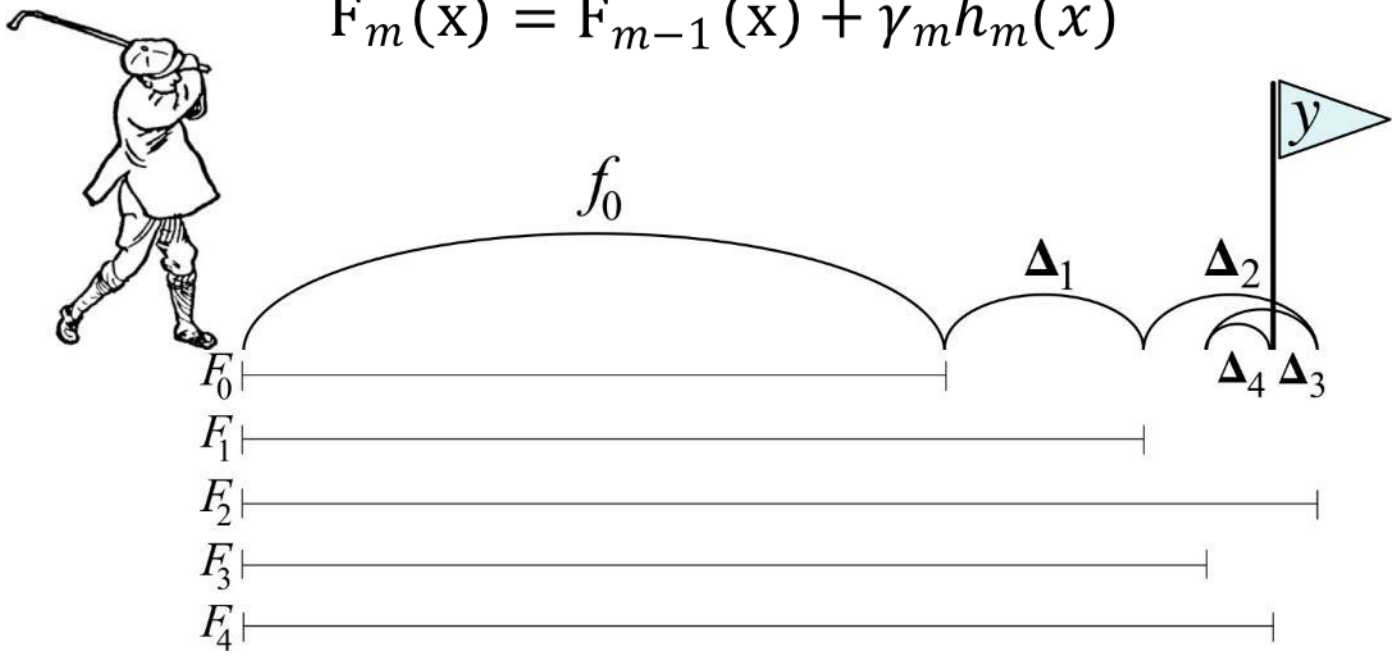
Gradient Boosting

- 加性模型（additive model）：

$$\mathrm{F}(\mathrm{x}) = \sum_{m=1}^{M} \gamma_m \, h_m(x)$$

- 前向分步式构建加性模型：

$$\mathrm{F}_m(\mathrm{x}) = \mathrm{F}_{m-1}(\mathrm{x}) + \gamma_m h_m(x)$$



Gradient Boosting

- 每一步弱学习器$h_m(x)$的训练目标是减少当前模型的预测值$F_{(m-1)}(x)$和目标值y之间的差距，即最小化损失函数$L(y, F_{(m-1)}(x)+h_m(x))$：

$$h_m(\mathrm{x}) = \underset{h_m}{\operatorname{argmin}} \sum_{i=1}^{n} \mathrm{L}(\mathrm{y}_i, \mathrm{F}_{m-1}(x_i) + h_m(x_i))$$

- 如果L是平方误差，则 $h_m(x_i) \approx \mathrm{y}_i - \mathrm{F}_{m-1}(x_i)$ 残差

- 对于任意可微的损失函数L，数值化地近似求解最小化问题

$$\mathrm{F}_m(\mathrm{x}) = \mathrm{F}_{m-1}(\mathrm{x}) - \gamma_m \sum_{i=1}^{n} \nabla_{F_{m-1}} \mathrm{L}(\mathrm{y}_i, \mathrm{F}_{m-1}(x_i))$$

$$h_m(x_i) \approx -\left[\frac{\partial L(y_i, F(\boldsymbol{x_i}))}{\partial F(\boldsymbol{x}_i)}\right]_{F(\boldsymbol{x})=F_{m-1}(\boldsymbol{x})}$$

伪残差

$$\gamma_m = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} \mathrm{L}(\mathrm{y}_i, \mathrm{F}_{m-1}(x_i) - \gamma \sum_{i=1}^{n} \nabla_{F_{m-1}} \mathrm{L}(\mathrm{y}_i, \mathrm{F}_{m-1}(x_i)))$$

Gradient Boosting伪码

1. 初始化弱学习器为一个常数值：

$$F_0(x) = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, \gamma))$$

2. 分别设*m = 1, 2, ..., M*，重复下面的步骤：

a) 计算伪残差（pseudo residuals）：

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \qquad i = 1, 2, \ldots, n$$

b) 训练一个弱学习器$h_m(x)$拟合伪残差，即用训练集$\{(x_i, r_{im})\}_{i=1}^{n}$训练

c) 求解下面的一维优化问题得到γ_m：

$$\gamma_m = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) - \gamma \sum_{i=1}^{n} \nabla_{F_{m-1}} L(y_i, F_{m-1}(x_i)))$$

d) 更新模型：

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x)$$

3. 输出最终模型：$F_M(x)$

Boosting的损失函数

- Boosting基本思想是“知错就改”
- 每一步训练的弱学习器都是去纠正前面学习器所犯的错误
- 最终将多个弱学习器结合起来得到一个很强的学习器

Boosting的损失函数

分类常见的损失函数：

- 指数损失

$$L\big(y, f(x)\big) = \exp(-yf(x))$$

- 二项对数似然损失

$$L\big(y, f(x)\big) = \log(1 + \exp(-yf(x)))$$

Boosting的损失函数

分类常见的损失函数：

- 多项对数似然损失

  k个类别分类器输出值归一化为概率值：

$$p_k(x) = \frac{\exp(f_k(x))}{\sum_{l=1}^{K} \exp(f_l(x))}$$

$$L(y, f(x)) = -\sum_{k=1}^{K} I(y = k) \log p_k(x)$$

$$= -\sum_{k=1}^{K} I(y = k) f_k(x) + \log \sum_{l=1}^{K} \exp(f_l(x))$$

Boosting的损失函数

回归常见的损失函数：

- 平方损失

$$L\big(y, f(x)\big) = (y - f(x))^2$$

- 绝对损失

$$L\big(y, f(x)\big) = |y - f(x)|$$

Boosting的损失函数

回归常见的损失函数：

- Hubber损失

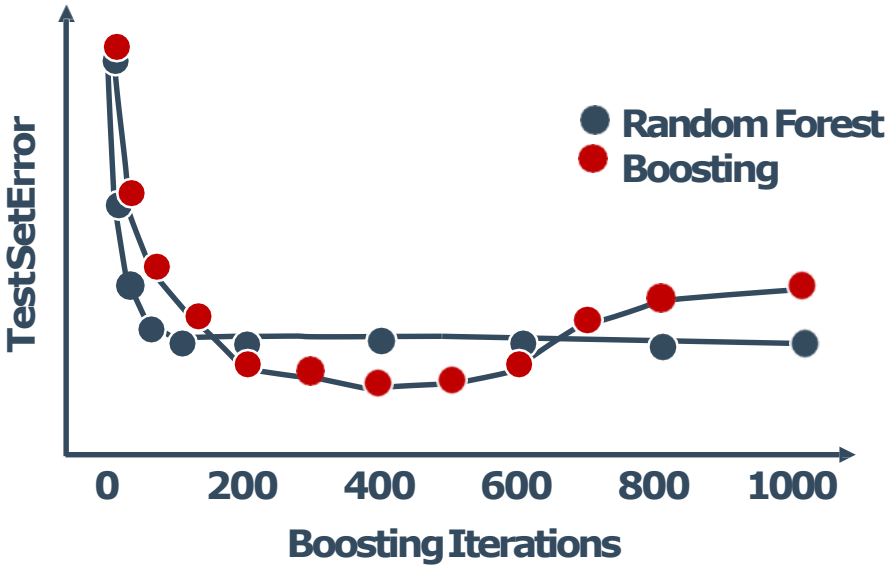
$$L\big(y, f(x)\big) = \begin{cases} (y - f(x))^2, & |y - f(x)| \le \alpha \\ 2\alpha|y - f(x)| - \alpha^2, & |y - f(x)| > \alpha \end{cases}$$

- 分位数损失

$$L\big(y, f(x)\big) = \sum_{y \ge f(x)} \alpha|y - f(x)| + \sum_{y < f(x)} (1 - \alpha)\,|y - f(x)|$$

调节Gradient Boosting模型

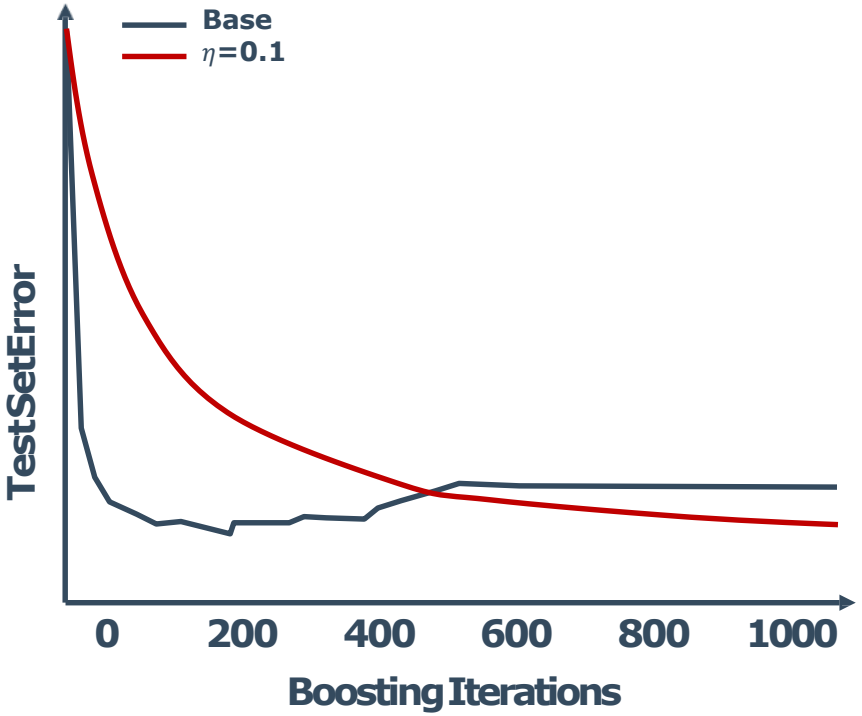
- Boosting是递加的，所以可能过拟合
- 使用交叉验证来设置决策树的个数



调节Gradient Boosting模型

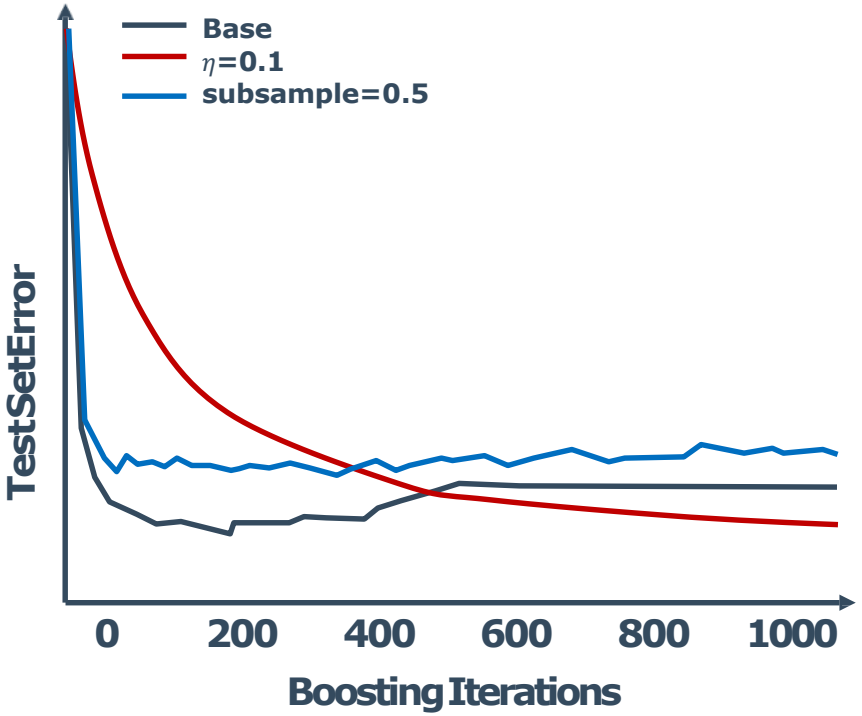
- 学习率（η）：设为<1.0用于正则化。又称作“shrinkage”

$$F_m(x) = F_{m-1}(x) + \eta\gamma_m h_m(x)$$



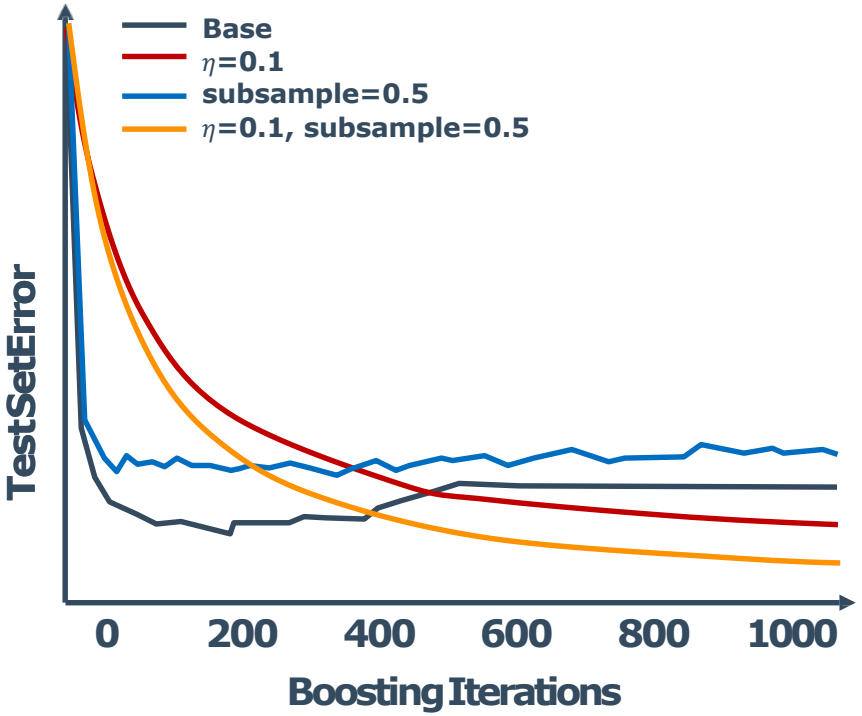
调节Gradient Boosting模型

- 学习率（η）：设为<1.0用于正则化。又称作“shrinkage”
- 子采样：只使用部分数据用于训练基学习器（stochastic gradient boosting,随机梯度提升）



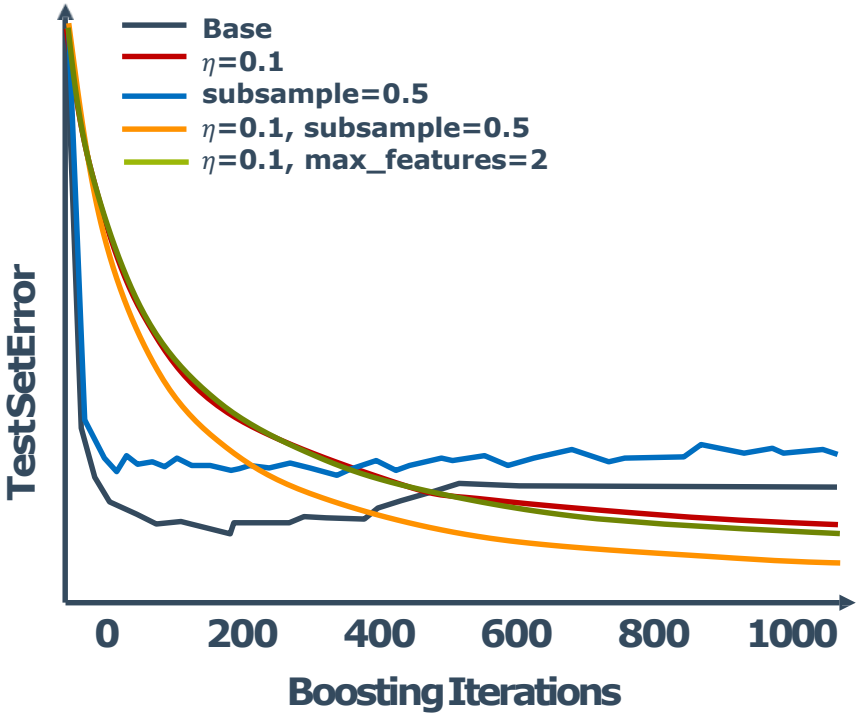
调节Gradient Boosting模型

- 学习率（η）：设为<1.0用于正则化。又称作“shrinkage”
- 子采样：只使用部分数据用于训练基学习器（stochastic gradient boosting,随机梯度提升）



调节Gradient Boosting模型

- 学习率（η）：设为<1.0用于正则化。又称作“shrinkage”
- 子采样：只使用部分数据用于训练基学习器（stochastic gradient boosting,随机梯度提升）
- 最大特征数：基学习器分裂时考虑的特征数目



Gradient Boosting分类器的语法

导入包含分类方法的类：

```
from sklearn.ensemble import GradientBoostingClassifier
```

创建该类的一个对象：

```
GBC = GradientBoostingClassifier(learning_rate=0.1,
        max_features=1, subsample=0.5, n_estimators=200)
```

拟合训练数据，并预测：

```
GBC = GBC.fit (X_train, y_train)
y_predict = GBC.predict(X_test)
```

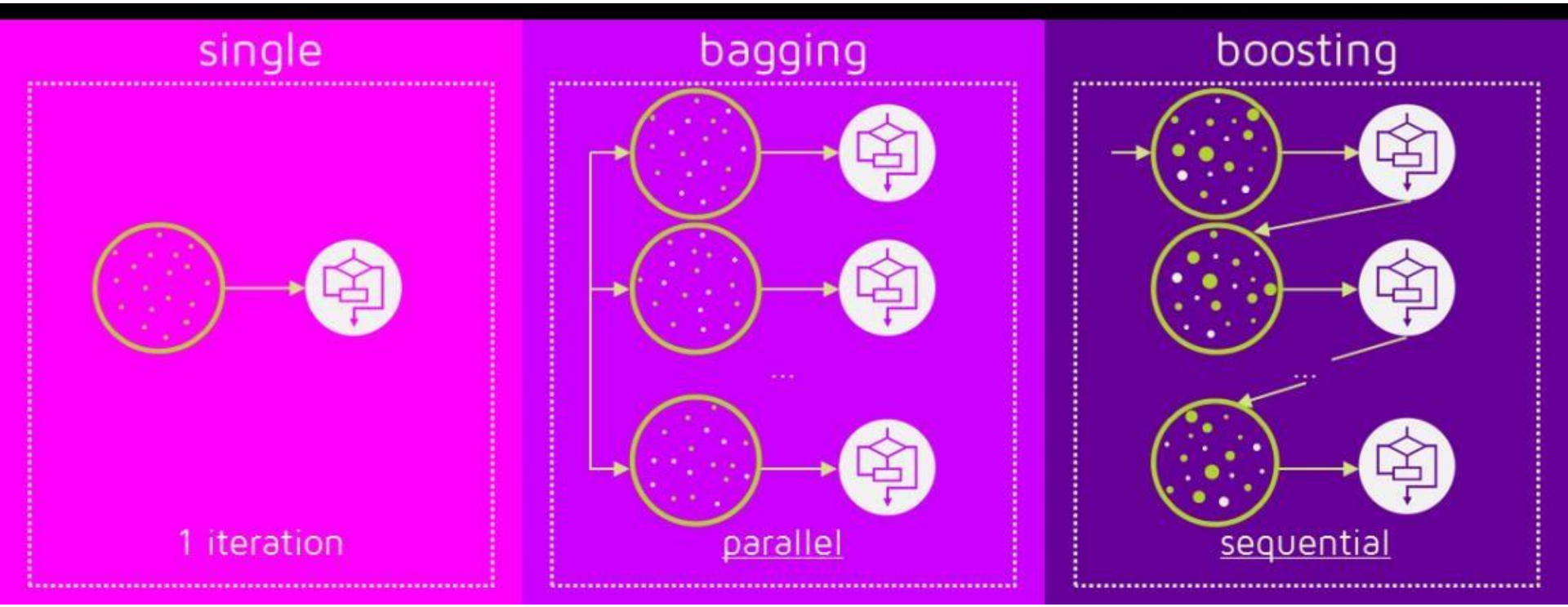
使用交叉验证调节参数，回归用GradientBoostingRegressor

http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html

GradientBoostingClassifier参数设置

loss	损失函数 分类和回归的损失函数不同。 对于分类，可以是对数似然损失“deviance”或者指数损失“exponential”缺省值是“deviance”。使用“exponential”则等同于AdaBoost算法。 对于回归，可以是平方损失“ls”、绝对损失“lad”、Huber损失“huber”或分位数损失“quantile”，缺省值是“ls”。一般来说，如果数据的噪音点不多，用默认的“ls”比较好。如果噪音点较多，则推荐使用抗噪音的“huber”损失函数。如果我们需要对训练集进行分段预测时，则采用“quantile”损失函数。
n_estimators	基学习器个数 集成的基学习器的个数。或者说算法的迭代次数。 缺省值是100。
learning_rate	学习率 缺省值是0.1。一般设为0到1之间的值，缩减每个基学习器对最终结果的贡献量，即每个基学习器的权重缩减系数，也称步长。它和n_estimators之间存在一个折衷，一起来决定算法的拟合效果。如果要达到一定的拟合效果更小的学习率意味着要训练更多的弱学习器。
subsample	子采样 用于训练每个基学习器的样本数量占整个训练集的比例。 缺省值是1.0，即用全部数据训练基学习器。小于1.0则是随机梯度提升。小于1.0的值会降低系统的方差，防止过拟合，但会增大系统的偏差。

Bagging vs. Boosting



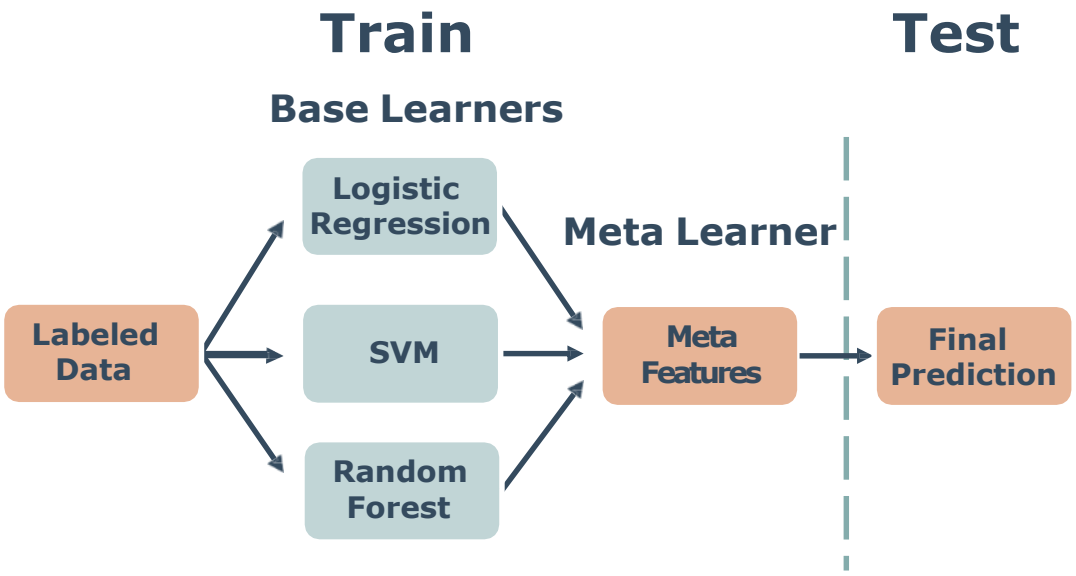
Bagging vs. Boosting

bagging	boosting
Bootstrap产生的样本	拟合全部数据集
独立创建的基学习器	相继创建的基本树
只考虑数据点	利用前面创建的模型的残差
不使用权重	增加错误分类点的权重
不会造成过拟合	当心过拟合
主要关注降低方差	主要关注降低偏差

STACKING
堆叠

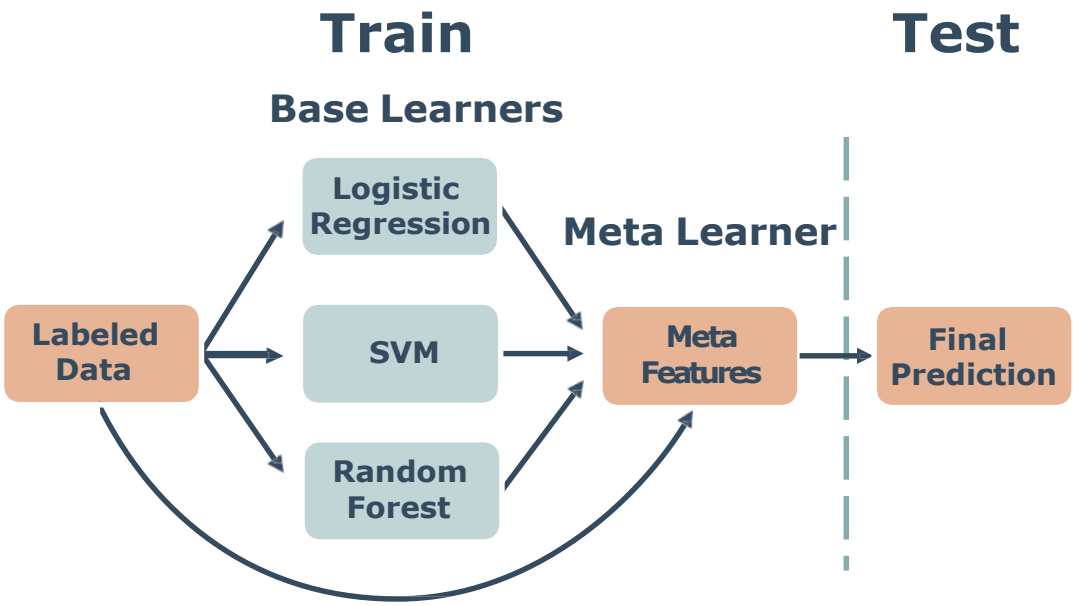
Stacking: 结合多个异构的分类器

- 不同类型的模型可以结合起来构建stacked模型
- 类似bagging，但不局限于决策树



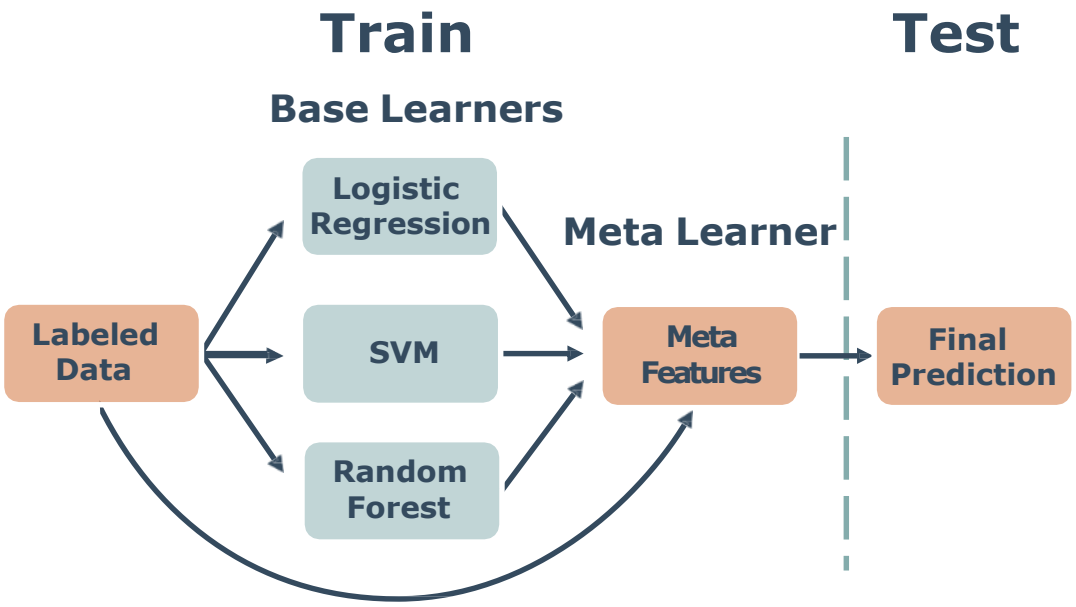
Stacking: 结合多个异构的分类器

- 不同类型的模型可以结合起来构建stacked模型
- 类似bagging，但不局限于决策树
- 基学习器的输出产生的特征，可以和数据特征结合在一起



Stacking: 结合多个异构的分类器

- 基学习器的输出可以用多数投票或加权和等方式结合起来
- 如果元学习器有参数，则需要另外取出的数据做预测
- 当心增加的模型复杂度



多层堆叠