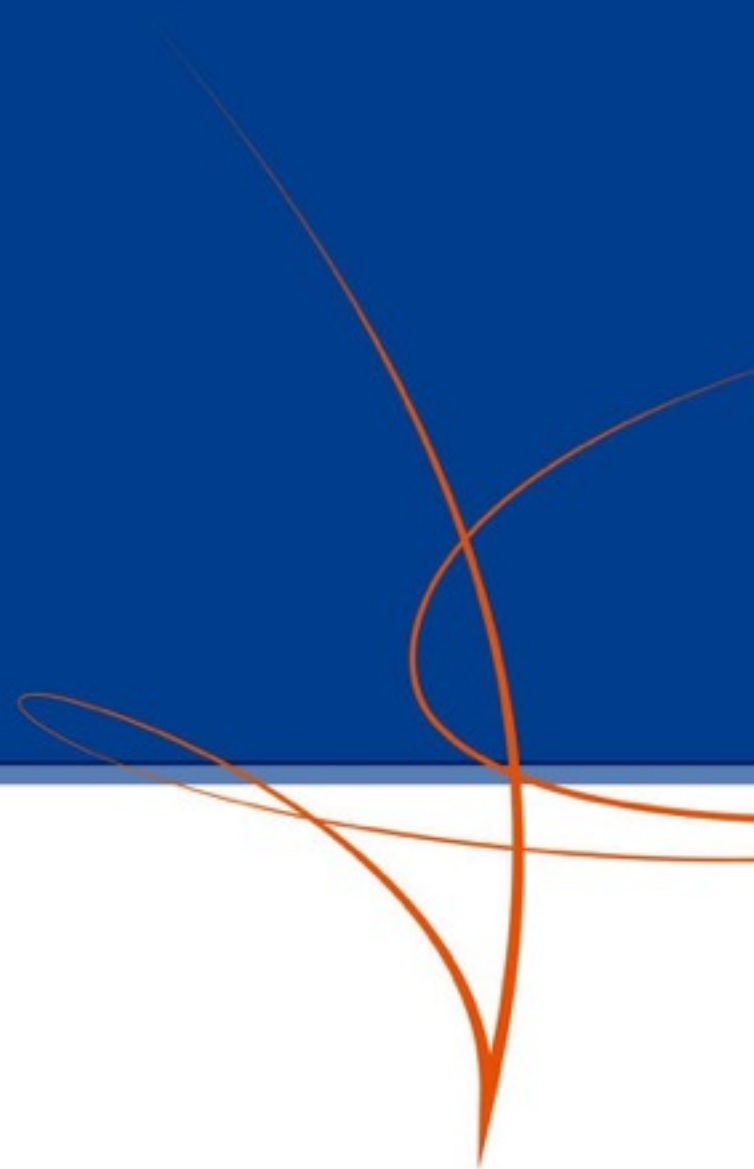


# 模型调优与融合



## ■ 前序工作流程

1. 数据处理
2. 特征工程
3. 模型选择
4. 交叉验证
5. 寻找最佳超参数

## ■ 模型优化

1. 模型状态
2. 权重分析
3. bad-case分析
4. 模型融合

### □ 数据清洗

- 不可信的样本丢掉
- 缺省值极多的字段考虑不用

### □ 数据采样

- 下/上采样
- 保证样本均衡

### □ 特征处理

- ① 数值型
- ② 类别型
- ③ 时间类
- ④ 文本型
- ⑤ 统计型
- ⑥ 组合特征

## □ 特征选择

### ① 过滤型

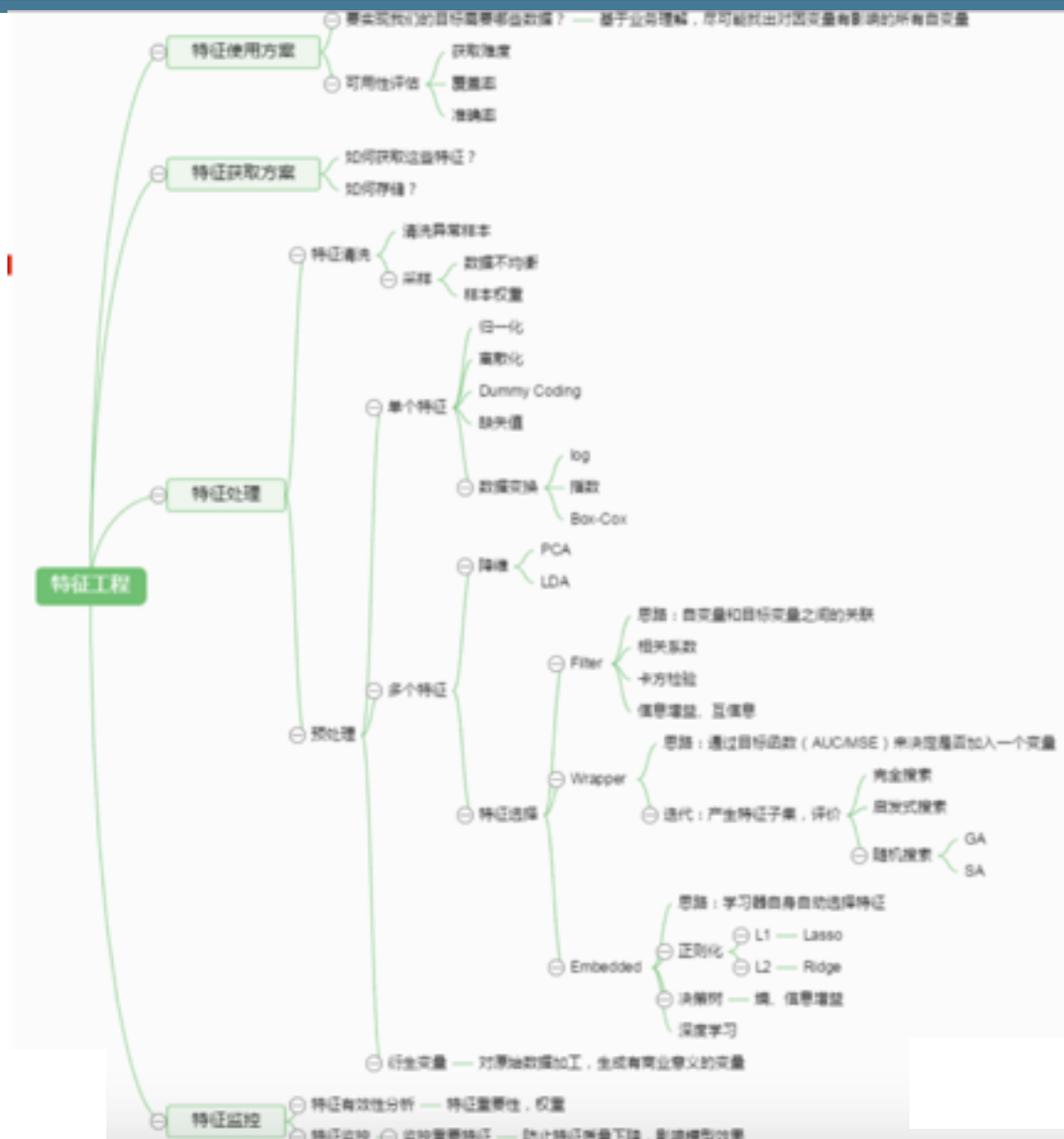
- `sklearn.feature_selection.SelectKBest`

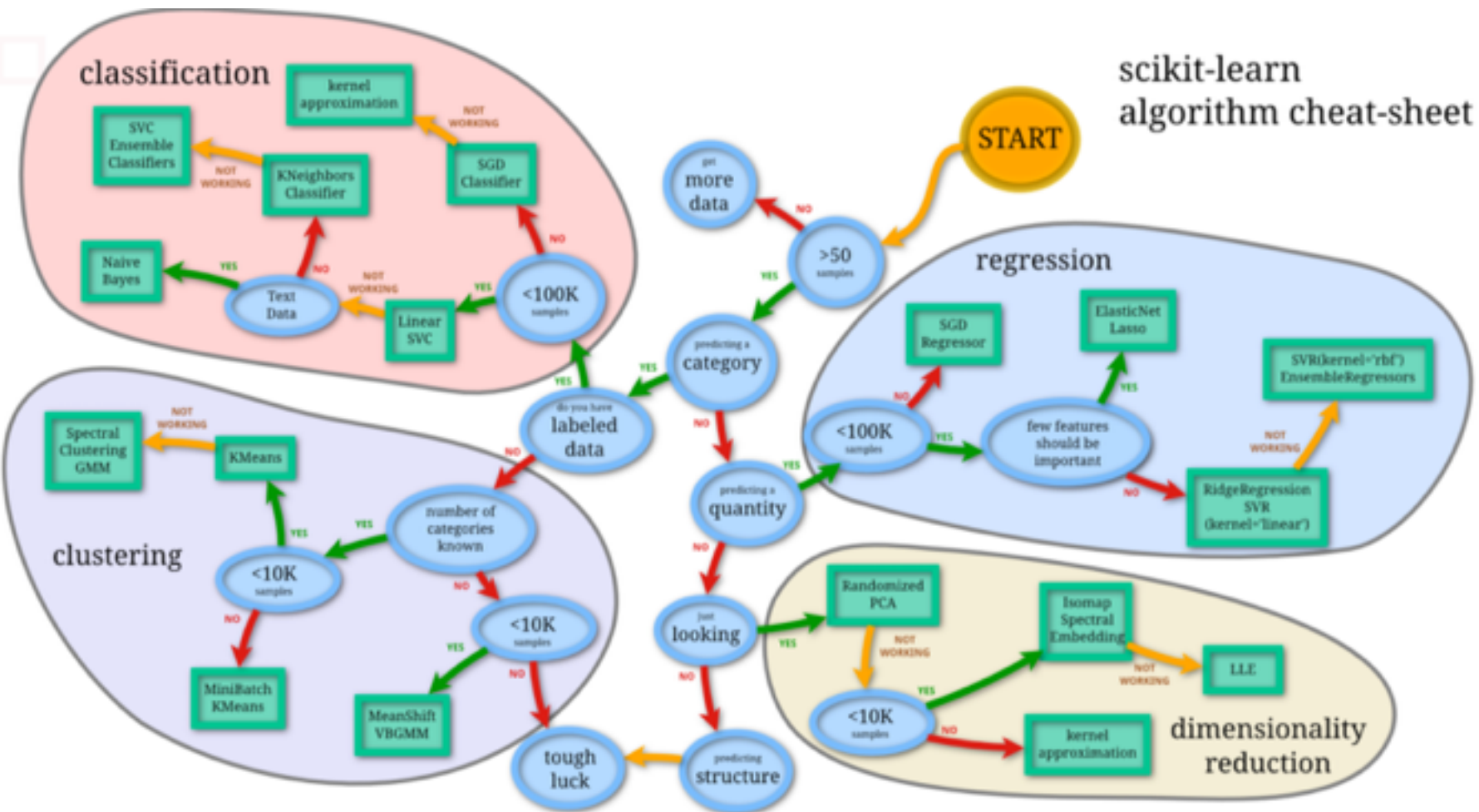
### ② 包裹型

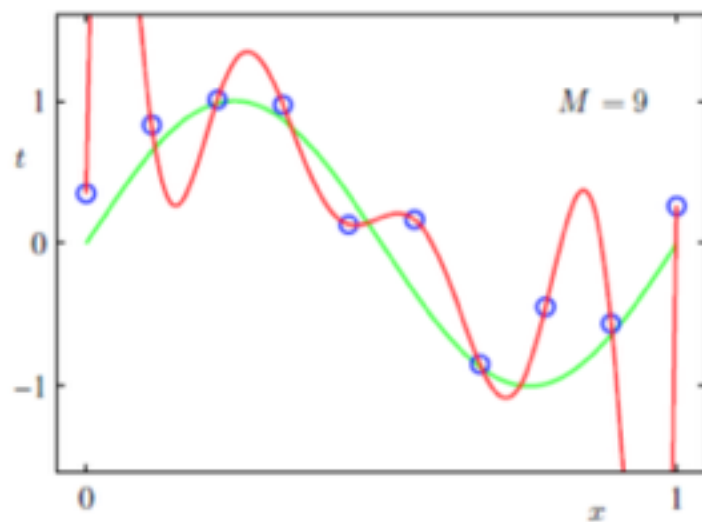
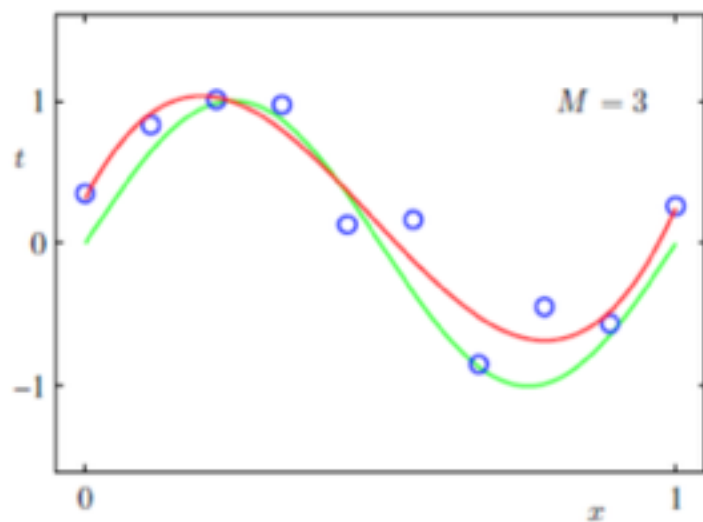
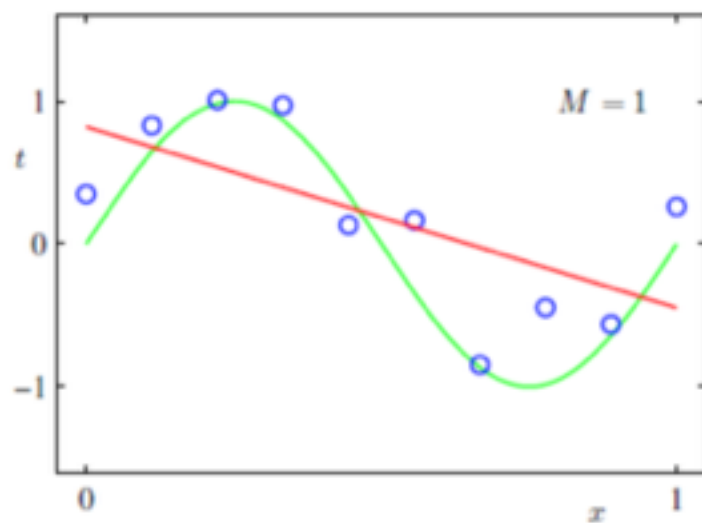
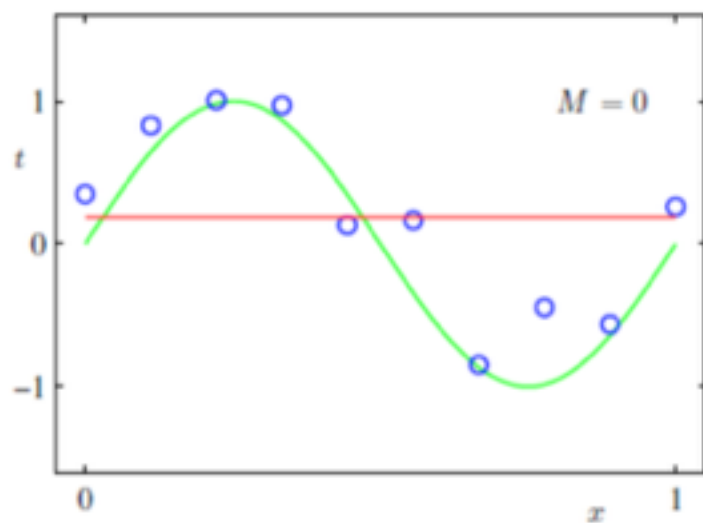
- `sklearn.feature_selection.RFE`

### ③ 嵌入型

- `feature_selection.SelectFromModel`
- Linear model, L1正则化





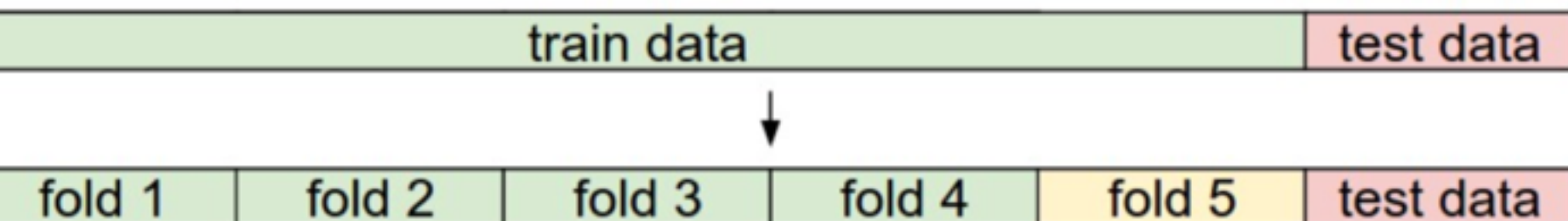




## □ 交叉验证 (cross validation)

- 交叉验证集做参数/模型选择
- 测试集只做模型效果评估

## □ K折交叉验证 (K-fold cross validation)

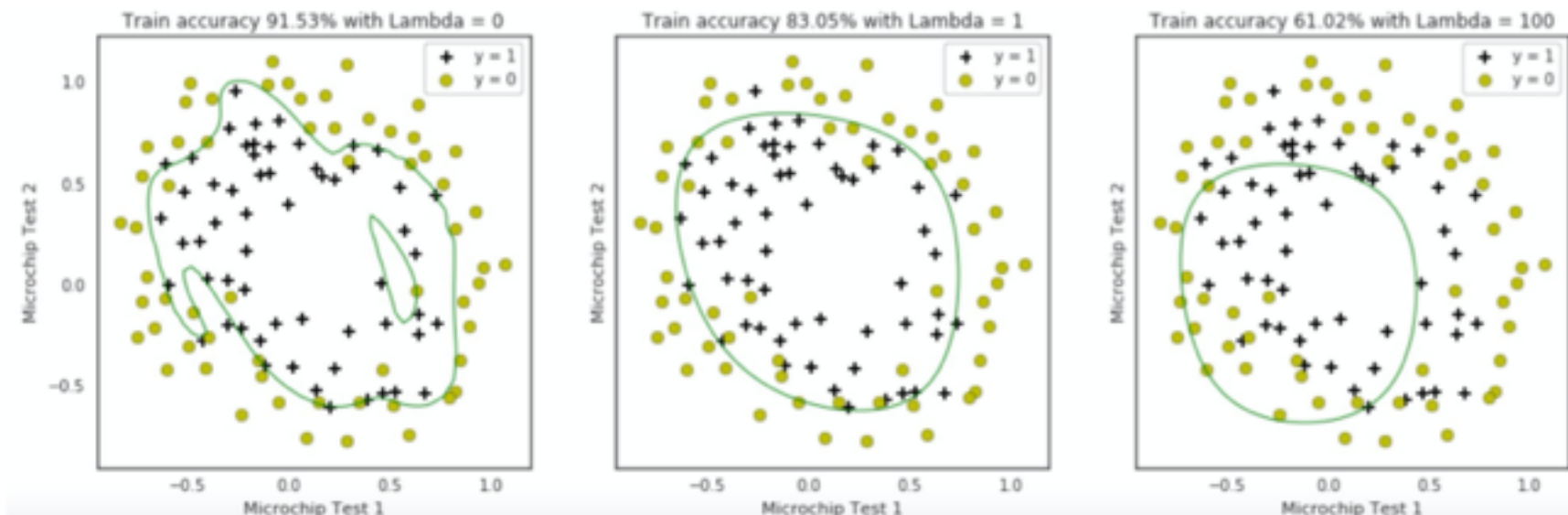


## □ 对模型有何影响

### `sklearn.linear_model.LogisticRegression`

```
class sklearn.linear_model. LogisticRegression (penalty='l2', dual=False, tol=0.0001, C=1.0,  
fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver='liblinear', max_iter=100,  
multi_class='ovr', verbose=0, warm_start=False, n_jobs=1)
```

[\[source\]](#)



### □ 交叉验证选取

#### `sklearn.grid_search.GridSearchCV`

```
class sklearn.grid_search. GridSearchCV (estimator, param_grid, scoring=None, fit_params=None, n_jobs=1,  
iid=True, refit=True, cv=None, verbose=0, pre_dispatch='2*n_jobs', error_score='raise') \[source\]
```

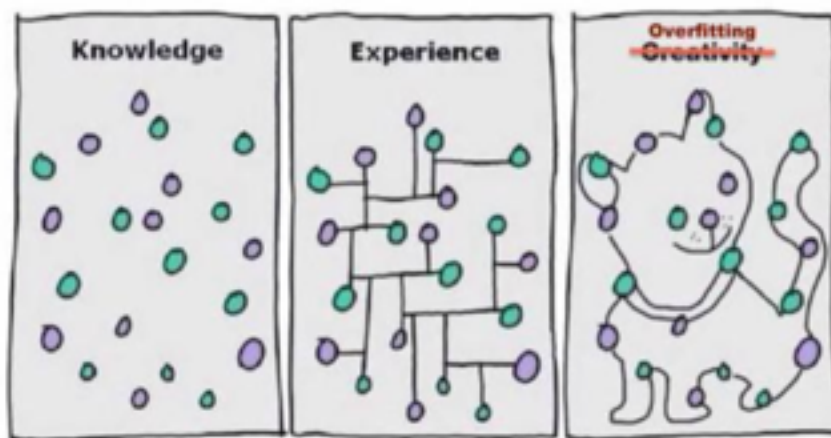
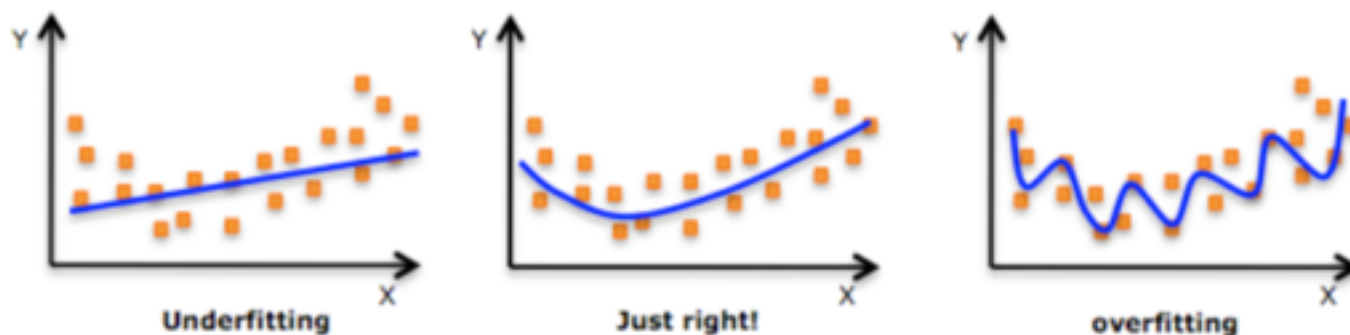
The grid search provided by `GridSearchCV` exhaustively generates candidates from a grid of parameter values specified with the `param_grid` parameter. For instance, the following `param_grid`:

```
param_grid = [  
    {'C': [1, 10, 100, 1000], 'kernel': ['linear']},  
    {'C': [1, 10, 100, 1000], 'gamma': [0.001, 0.0001], 'kernel': ['rbf']},  
]
```

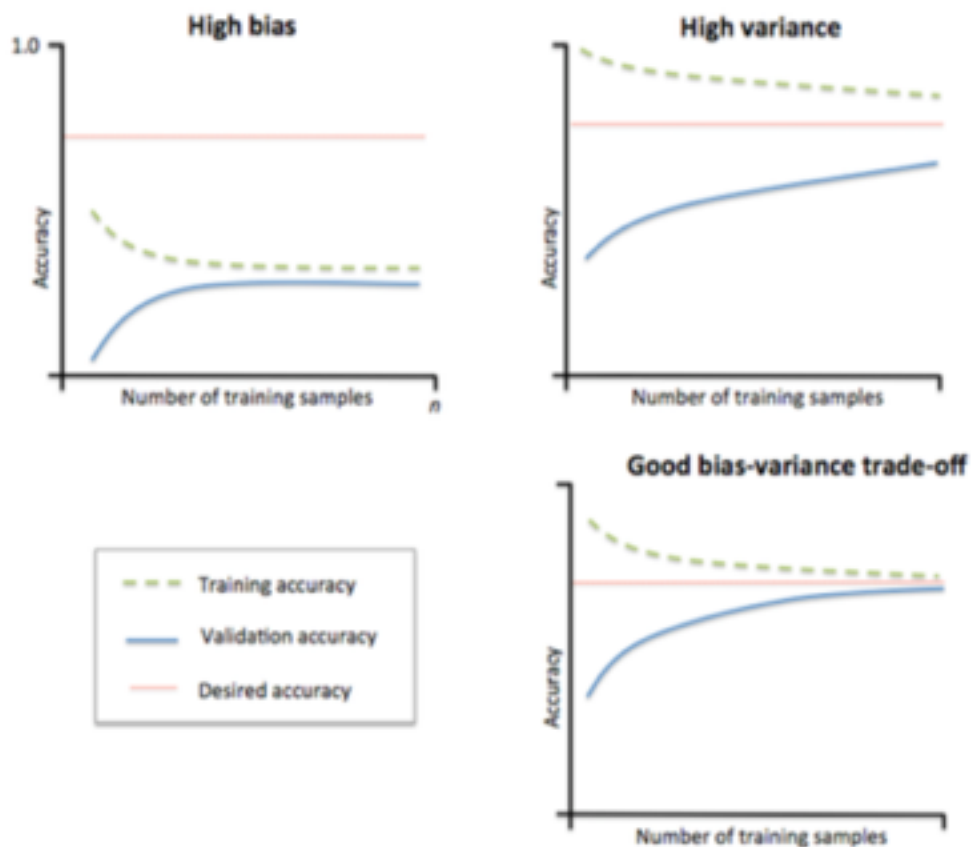
## □ 模型状态

过拟合 (overfitting/high variance)

欠拟合 (underfitting/high bias)



## □ 模型状态验证工具：学习曲线



## □ 不同模型状态处理

### ① 过拟合

- 找更多的数据来学习
- 增大正则化系数
- 减少特征个数(不是太推荐)

注意：不要以为降维可以解决过拟合问题

### ② 欠拟合

- 找更多的特征
- 减小正则化系数

## □ 线性模型的权重分析

### ① 过线性或者线性kernel的model

- Linear Regression
- Logistic Regression
- LinearSVM
- ...

### ② 对权重绝对值高/低的特征

- 做更细化的工作
- 特征组合

## □ Bad-case分析

### ① 分类问题

- 哪些训练样本分错了？
- 我们哪部分特征使得它做了这个判定？
- 这些bad cases有没有共性
- 是否有还没挖掘的特性
- ...

### ② 回归问题

- 哪些样本预测结果差距大，为什么？
- ...



## □ Bad-case分析

### ① 分类问题

- 哪些训练样本分错了？
- 我们哪部分特征使得它做了这个判定？
- 这些bad cases有没有共性
- 是否有还没挖掘的特性
- ...

### ② 回归问题




- 哪些样本预测结果差距大，为什么？
- ...

## □ 模型融合 (model ensemble)

### ① 是什么

- Ensemble Learning 是一组 individual learner 的组合
  - 如果 individual learner 同质, 称为 base learner
  - 如果 individual learner 异质, 称为 component learner

### ② 为什么

统计上	计算上	表现上
<p>Statistical</p> 	<p>Computational</p> 	<p>Representational</p> 
假设空间 $h$ 的平均更接近真实假设 $f$	迭代求解很可能找到局部最优解 多个局部最优解的平均更接近全局最优解	真实假设 $f$ 可能不在已知的假设空间 $H$ 内 平均可能更接近 $H$ 外的真实假设 $f$

## □ 模型融合 (model ensemble)

✓ 简单说来，我们信奉几条信条

① 群众的力量是伟大的，集体智慧是惊人的

- Bagging
- 随机森林/Random forest

② 站在巨人的肩膀上，能看得更远

- 模型stacking

③ 一万小时定律

- Adaboost
  - 逐步增强树/Gradient Boosting Tree
-

## □ Bagging

① 模型很多时候效果不好的原因是什么？

➤ 过拟合啦！！！！

② 如何缓解？

➤ 少给点题，别让它死记硬背这么多东西

➤ 多找几个同学来做题，综合一下他们的答案

## □ Bagging

### ① 用一个算法

- 不用全部的数据集，每次取一个子集训练一个模型
- 分类：用这些模型的结果做vote
- 回归：对这些模型的结果取平均

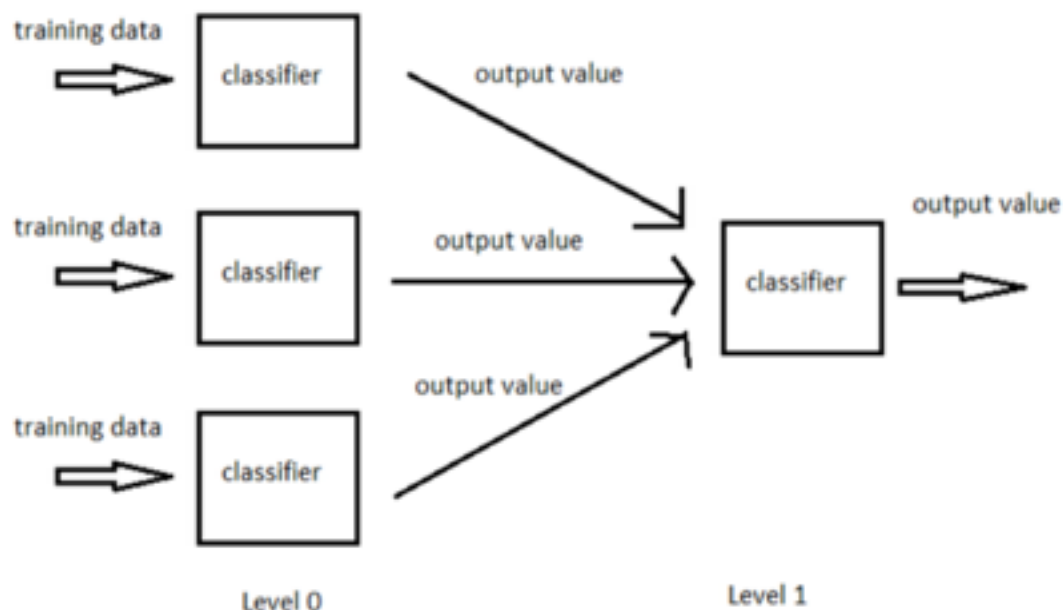
### ② 用不同的算法

- 用这些模型的结果做vote 或 求平均

## □ Stacking

➤ 用多种predictor结果作为特征训练

Concept Diagram of Stacking



## □ Stacking

➤ 用多种predictor结果作为特征训练

$$\hat{y}_1 = f_1(x_1, x_2, \dots)$$

$$\hat{y}_2 = f_2(x_1, x_2, \dots)$$

...



$$\hat{y}_e = \text{sign}(\sum \alpha_i \hat{y}_i)$$

$f_e() = \text{majority}$  — 等价于vote(majority vote)

$f_e() = \text{linear}$  — 等价于加权平均

## □ Stacking

➤ 用多种predictor结果作为特征训练

$$\hat{y}_1 = f_1(x_1, x_2, \dots)$$

$$\hat{y}_2 = f_2(x_1, x_2, \dots)$$

...



$$\hat{y}_e = \text{sign}(\sum \alpha_i \hat{y}_i)$$

$f_e() = \text{majority}$  — 等价于vote(majority vote)

$f_e() = \text{linear}$  — 等价于加权平均

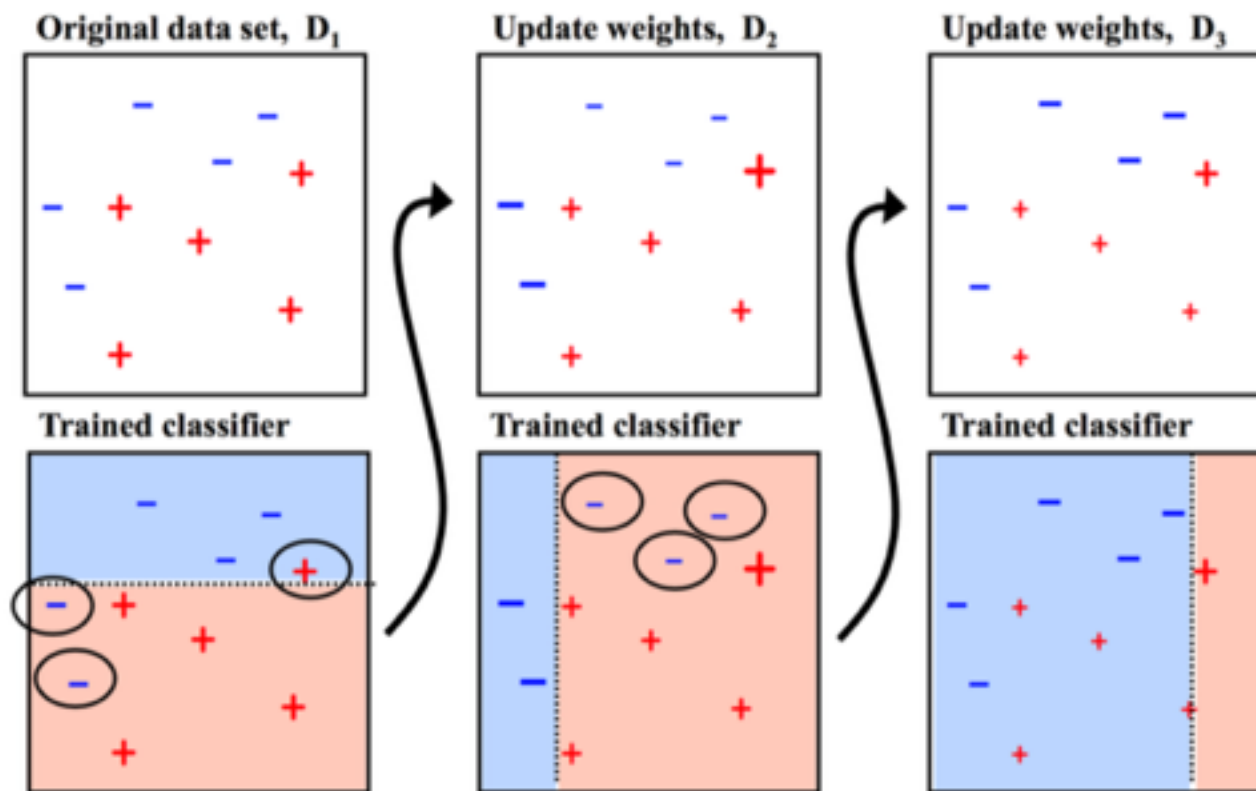


## □ Adaboost

### ① 考得不好的原因是什么？

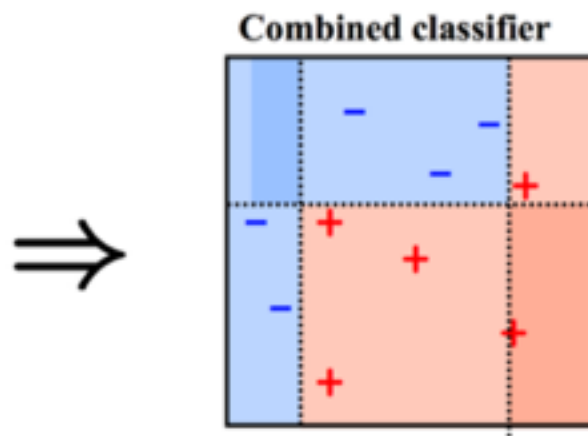
- 还不够努力，练习题要多次学习
  - 重复迭代和训练
- 时间分配要合理，要多练习之前做错的题
  - 每次分配给分错的样本更高的权重
- 我不聪明，但是脚踏实地，用最简单的知识不断积累，成为专家
  - 最简单的分类器的叠加

## □ Adaboost



## □ Adaboost

$$.33 * \begin{array}{|c|} \hline \text{blue} \\ \hline \text{orange} \\ \hline \end{array} + .57 * \begin{array}{|c|} \hline \text{blue} \\ \hline \text{orange} \\ \hline \end{array} + .42 * \begin{array}{|c|} \hline \text{blue} \\ \hline \text{orange} \\ \hline \end{array} \geq 0$$

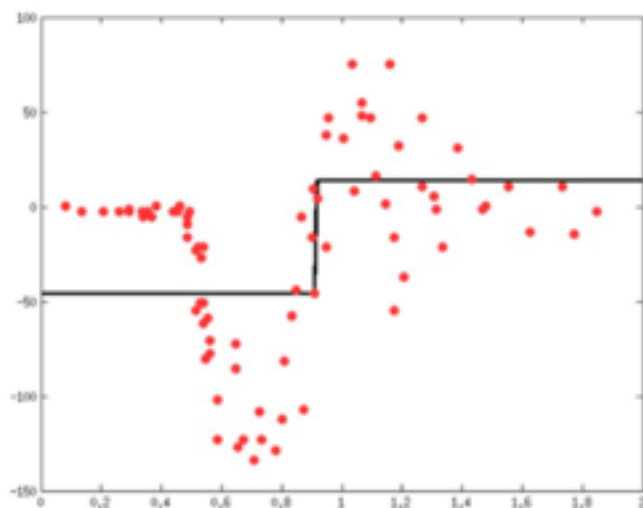


1-node decision trees  
"decision stumps"  
*very simple classifiers*

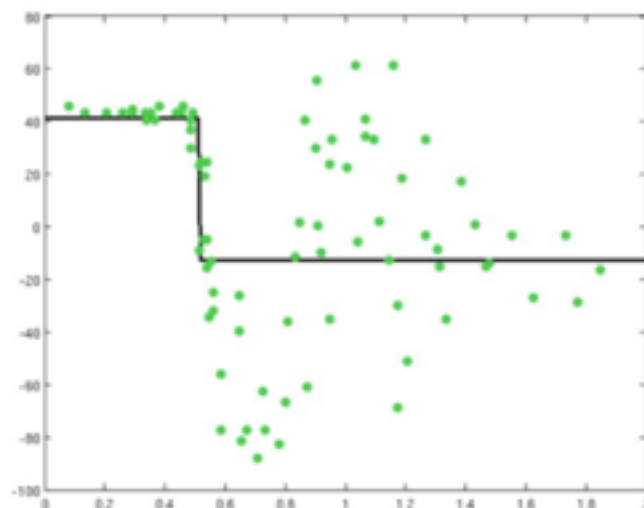
## □ Gradient Boosting Tree

① 和Adaboost思路类似，解决回归问题

Learn a simple predictor...

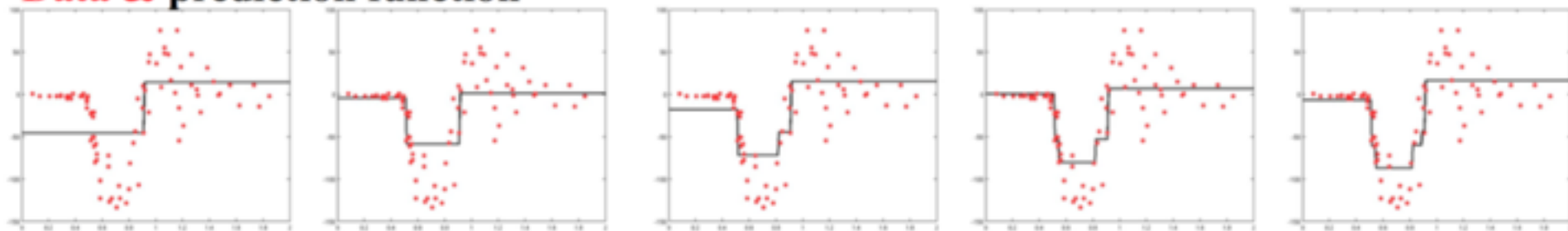


Then try to correct its errors

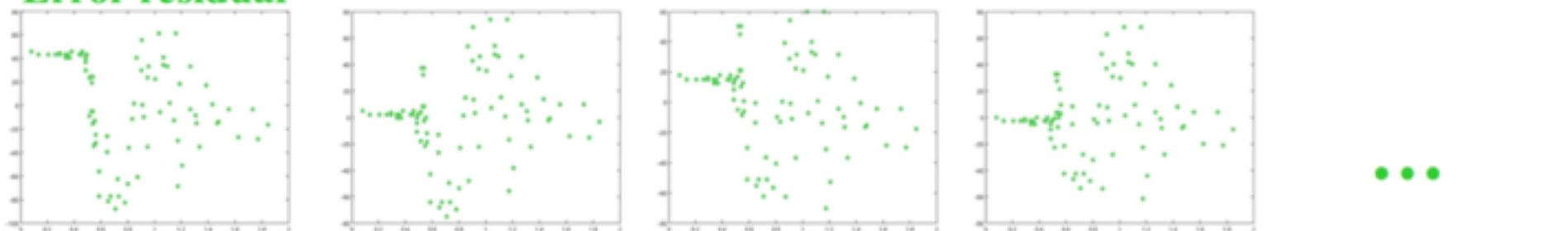


## □ Gradient Boosting Tree

**Data & prediction function**



**Error residual**



# Bagging Methods vs. Boosting Methods

	learner弱依赖Methods eg.Bagging	learner强依赖Methods eg.Boosting
方法	1.部分数据/部分参数/1或N个算法训练model 2.上述多个model的组合	1.训练基础算法，后续算法利用前面算法结果重点处理错误case 2.上述多个stage的组合
流程	<p> <math display="block">Y_M(\mathbf{x}) = \text{sign}\left(\sum_{m=1}^M \alpha_m y_m(\mathbf{x})\right)</math> </p>	<p> <math display="block">Y_M(\mathbf{x}) = \text{sign}\left(\sum_{m=1}^M \alpha_m y_m(\mathbf{x})\right)</math> </p>
偏差-方差分析	Bagging主要关注 <b>降低方差</b>  因此在不剪枝DT、Neural Network等易受样本扰动影响learner效果更明显	Boosting主要关注 <b>降低偏差</b>  因此Boosting基于泛化能力相当弱的learner构建很强的集成
适用范围	<b>高噪声</b>	<b>低噪声</b>
串行并行	并行 Bagging的各个预测函数没有权重,各个预测函数可以并行生成	串行 Boosting是有点权重的,各个预测函数只能顺序生成
样例	Random Forest	AdaBoost GDBT