

Part
02

交叉验证

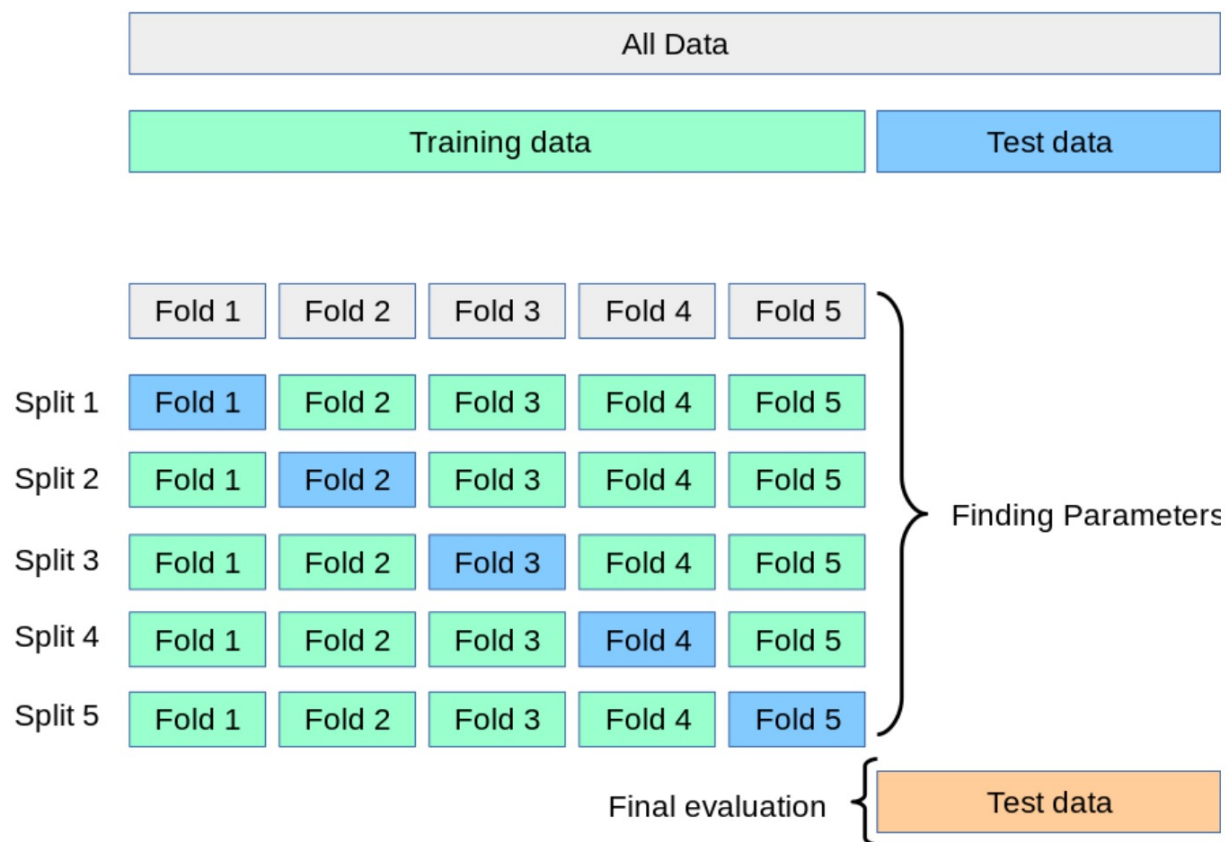
- 为什么要交叉验证
- 交叉验证的几种形式
- 交叉验证在特殊分布上



为什么要交叉验证?

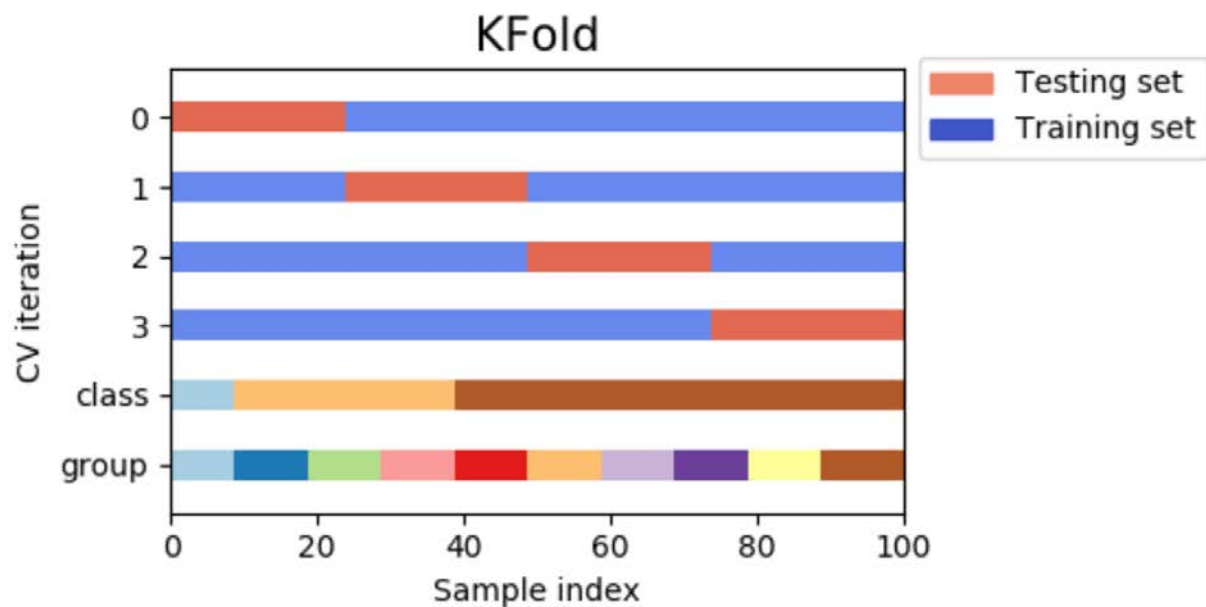
复杂模型与有限数据的矛盾

有些类似于bootstrap, 扩大数据的可用数量



```
>>> import numpy as np
>>> from sklearn.model_selection import KFold

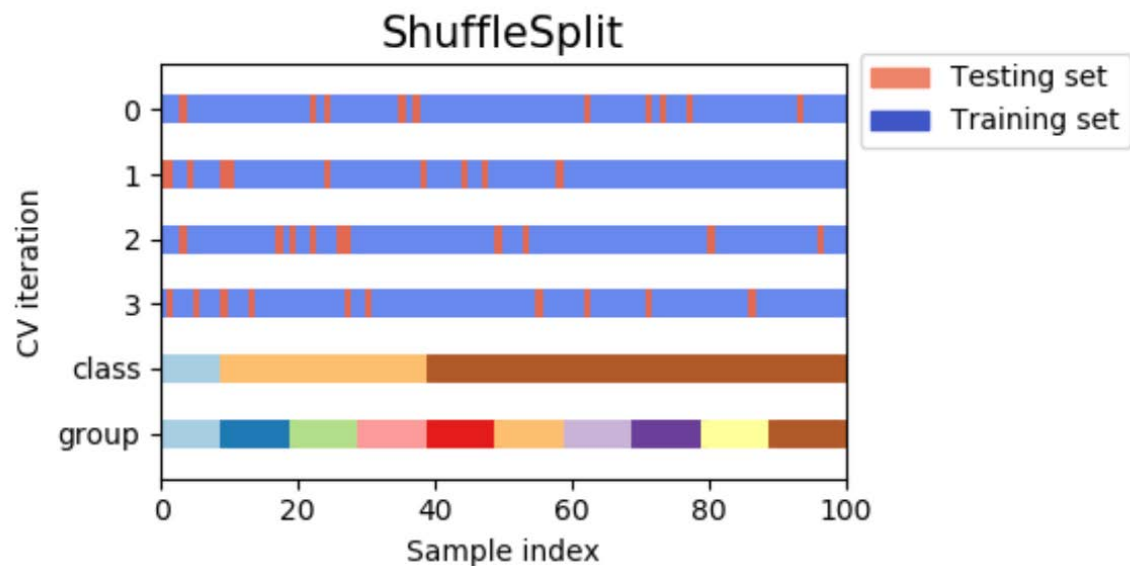
>>> X = ["a", "b", "c", "d"]
>>> kf = KFold(n_splits=2)
>>> for train, test in kf.split(X):
...     print("%s %s" % (train, test))
[2 3] [0 1]
[0 1] [2 3]
```



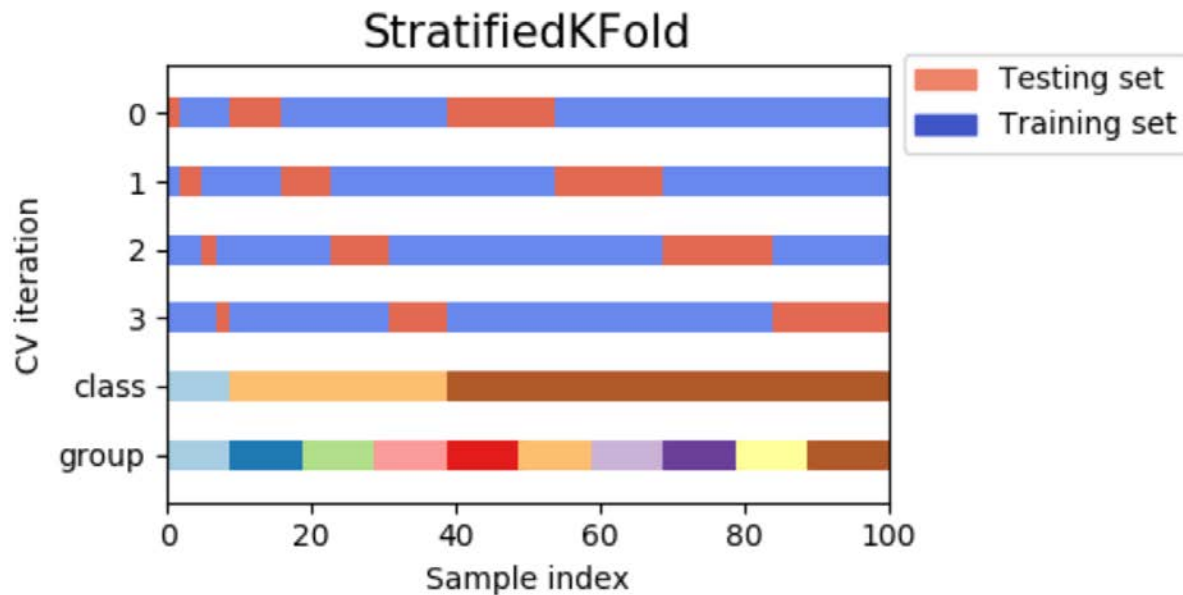
2.2

加入随机: shuffle

```
>>> from sklearn.model_selection import ShuffleSplit
>>> X = np.arange(10)
>>> ss = ShuffleSplit(n_splits=5, test_size=0.25, random_state=0)
>>> for train_index, test_index in ss.split(X):
...     print("%5s %5s" % (train_index, test_index))
[9 1 6 7 3 0 5] [2 8 4]
[2 9 8 0 6 7 4] [3 5 1]
[4 5 1 0 6 9 7] [2 3 8]
[2 7 5 8 0 3 4] [6 1 9]
[4 1 0 6 8 9 3] [5 2 7]
```

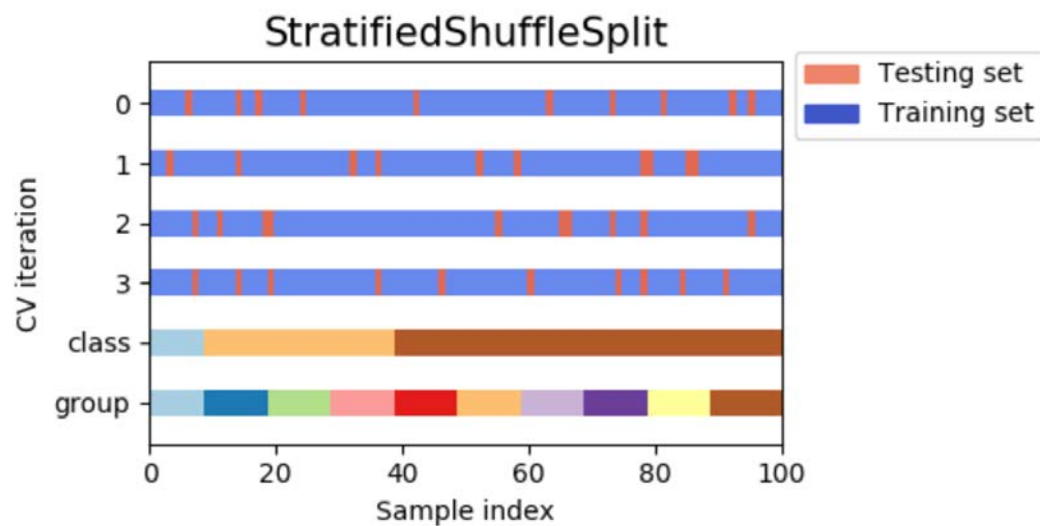


```
>>> from sklearn.model_selection import StratifiedKFold, KFold
>>> import numpy as np
>>> X, y = np.ones((50, 1)), np.hstack(([0] * 45, [1] * 5))
>>> skf = StratifiedKFold(n_splits=3)
>>> for train, test in skf.split(X, y):
...     print('train - {} | test - {}'.format(
...         np.bincount(y[train]), np.bincount(y[test])))
train - [30  3] | test - [15  2]
train - [30  3] | test - [15  2]
train - [30  4] | test - [15  1]
>>> kf = KFold(n_splits=3)
>>> for train, test in kf.split(X, y):
...     print('train - {} | test - {}'.format(
...         np.bincount(y[train]), np.bincount(y[test])))
train - [28  5] | test - [17]
train - [28  5] | test - [17]
train - [34] | test - [11  5]
```



2.2

类别间平衡 & shuffle

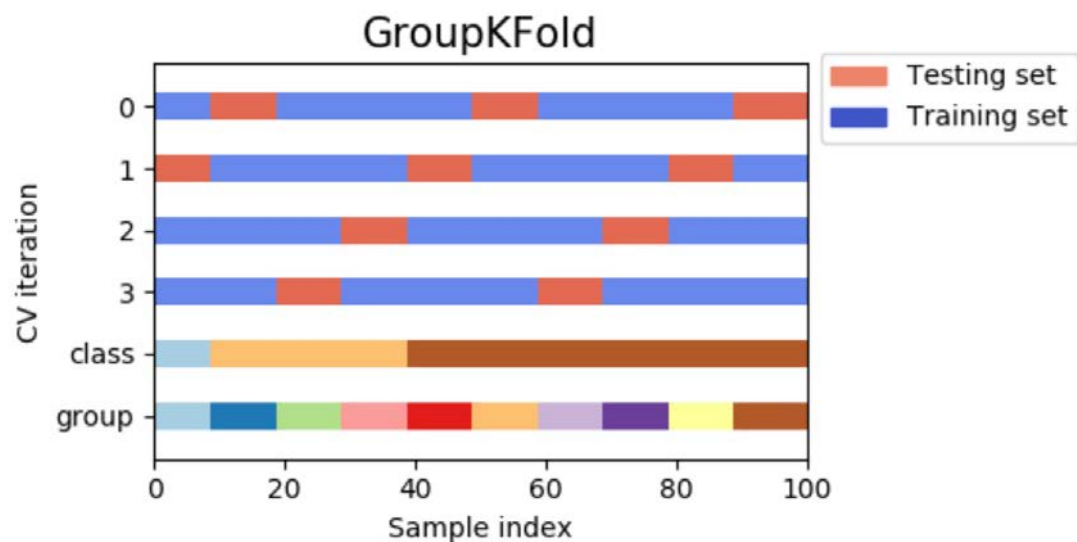


非iid情况：有小组出现

```
>>> from sklearn.model_selection import GroupKFold

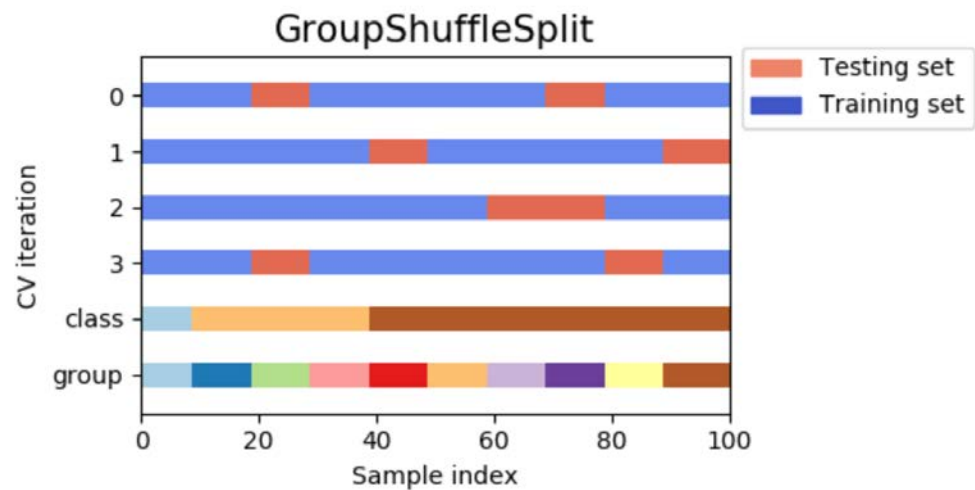
>>> X = [0.1, 0.2, 2.2, 2.4, 2.3, 4.55, 5.8, 8.8, 9, 10]
>>> y = ["a", "b", "b", "b", "c", "c", "c", "d", "d", "d"]
>>> groups = [1, 1, 1, 2, 2, 2, 3, 3, 3, 3]

>>> gkf = GroupKFold(n_splits=3)
>>> for train, test in gkf.split(X, y, groups=groups):
...     print("%5s %5s" % (train, test))
[0 1 2 3 4 5] [6 7 8 9]
[0 1 2 6 7 8 9] [3 4 5]
[3 4 5 6 7 8 9] [0 1 2]
```



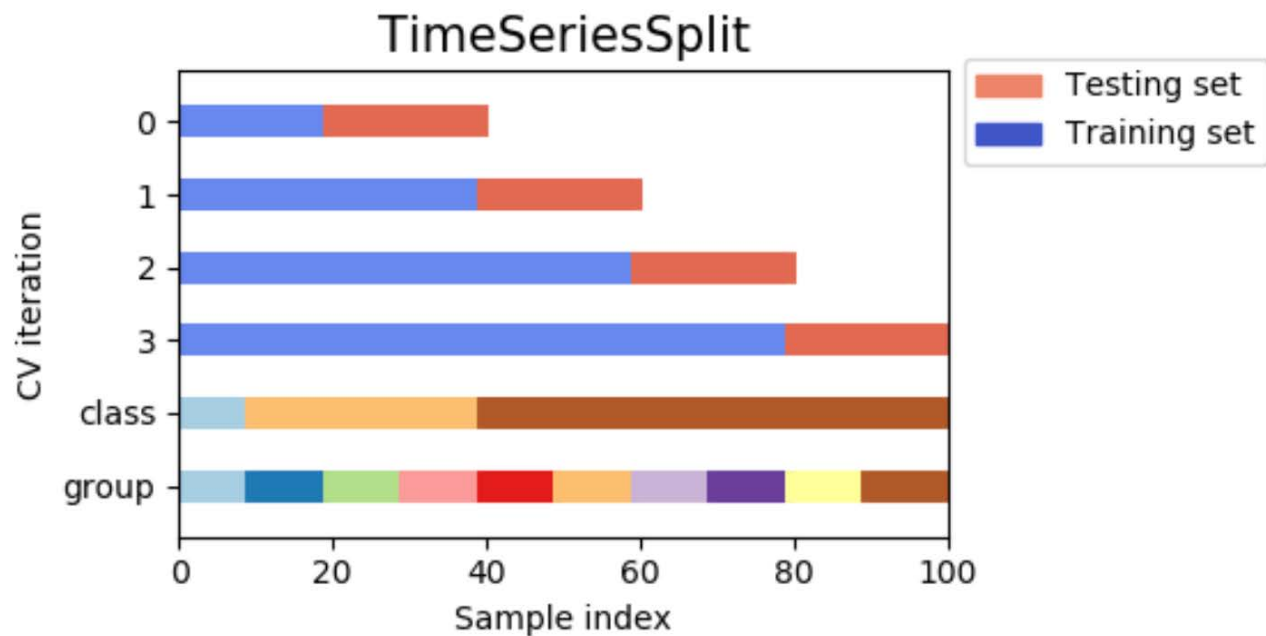

```
>>> from sklearn.model_selection import GroupShuffleSplit

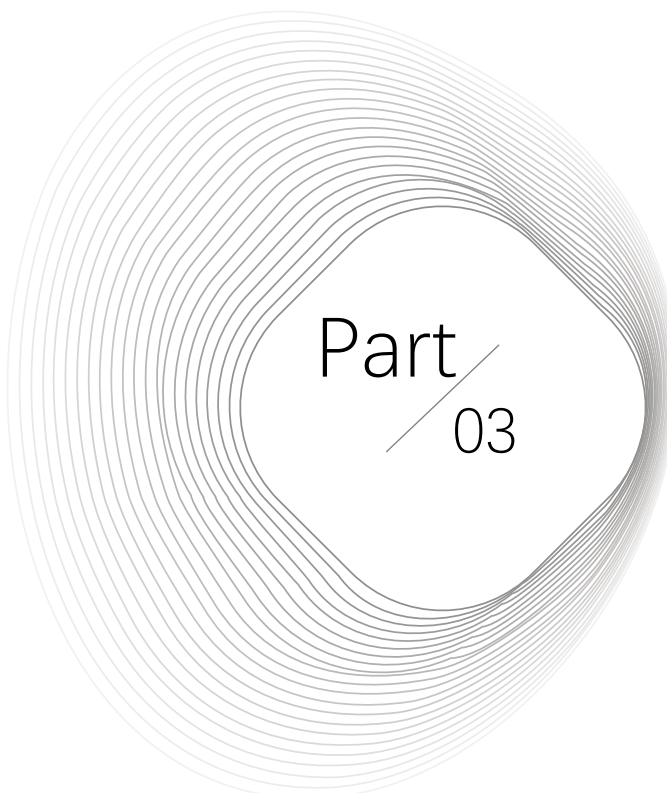
>>> X = [0.1, 0.2, 2.2, 2.4, 2.3, 4.55, 5.8, 0.001]
>>> y = ["a", "b", "b", "b", "c", "c", "c", "a"]
>>> groups = [1, 1, 2, 2, 3, 3, 4, 4]
>>> gss = GroupShuffleSplit(n_splits=4, test_size=0.5, random_state=0)
>>> for train, test in gss.split(X, y, groups=groups):
...     print("%s %s" % (train, test))
...
[0 1 2 3] [4 5 6 7]
[2 3 6 7] [0 1 4 5]
[2 3 4 5] [0 1 6 7]
[4 5 6 7] [0 1 2 3]
```




```
>>> from sklearn.model_selection import TimeSeriesSplit

>>> X = np.array([[1, 2], [3, 4], [1, 2], [3, 4], [1, 2], [3, 4]])
>>> y = np.array([1, 2, 3, 4, 5, 6])
>>> tscv = TimeSeriesSplit(n_splits=3)
>>> print(tscv)
TimeSeriesSplit(max_train_size=None, n_splits=3)
>>> for train, test in tscv.split(X):
...     print("%5 %s" % (train, test))
[0 1 2] [3]
[0 1 2 3] [4]
[0 1 2 3 4] [5]
```





Part
03

超参搜索

- 数据分几份?
- 搜索的方式
- 暴力之外

数据分几份？

训练数据

验证数据

测验数据

时间上的泄漏

时间相关的数据，决不能忽略时间

数据分组上的泄漏

数据间并非无关

统计特征上的泄漏

特征工程中，先分割，再变形

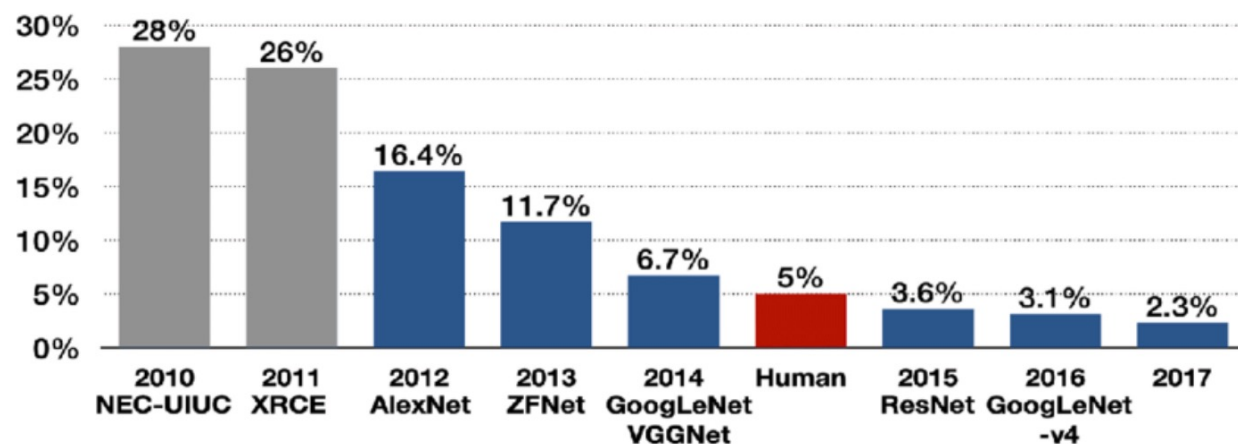
不要押题

百度之前干过这样一件事情

百度：已开除“人工智能测试作弊”事件负责人

2015年06月11日 17:30

Top-5 error



今年5月中旬，百度称公司在ImageNet图像识别测试中将错误率降低至创纪录的4.58%，当时，微软、谷歌软件在最新测试中的错误率分别为4.94%、4.8%。以上图像识别的错误率均低于通过练习人类能够达到的错误率（约为5%）。



超参数搜索方法：

超参数：单个训练无法给出 -> 多个尝试

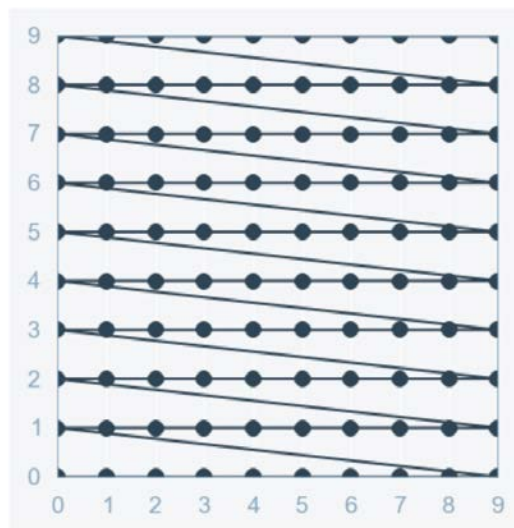
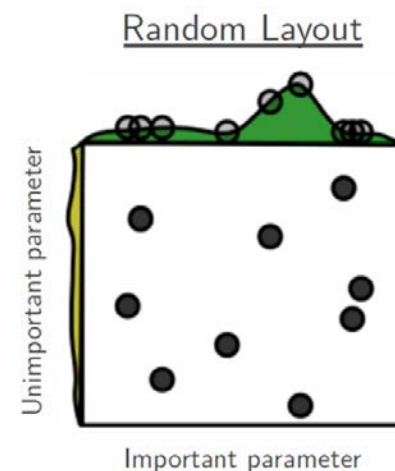
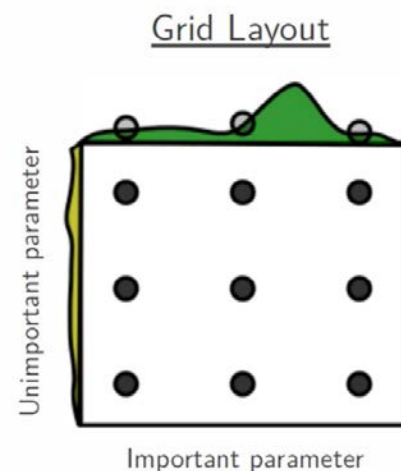
训练数据训练某组超参数下的模型，验证数据比较不同组合
超参数不只是数：模型选择、核、损失函数、权重……

网格搜索 vs 随机搜索

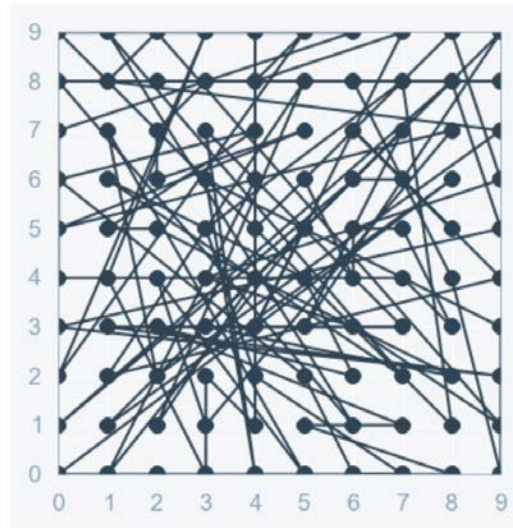
网格均匀取点、随机搜索随机取点

随机的好处：限定总开销、避免错误阈值

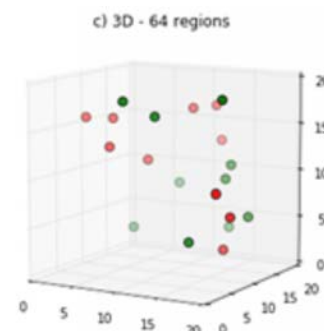
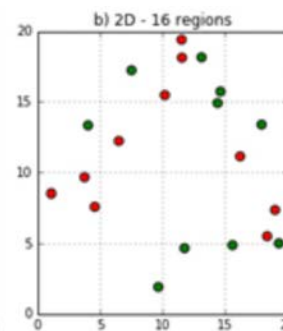
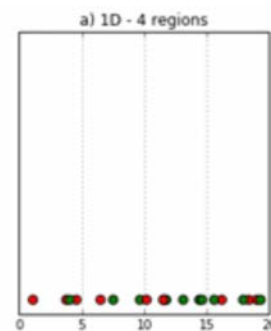
随机的威胁：维度诅咒、局部最优、阈值选择



Visual Representation of grid search



Visual Representation of Random search



超参数搜索建议

评估分数是不是正确选择

如果 $g=f$ 的衡量都错了，那么怎样都错了

多项评估分数同时进行

训练成本>>测试，多算几个分数边际收益>>边际成本

嵌套参数选择

在随机和网格之外还有一种“树状搜索”

回忆一下限制树高为4和限制树的节点数量为16

不要让CV触碰测试集：保留最后的高考卷

由于CV的存在，超参数其实也称为一种可训练的参数

并行化、更强的算法、更强的计算机

`n_jobs`参数，尝试使用高级算法&GPU、其他的计算框架PyTorch，更强的计算机（GPU、内存、CPU）

认识并尊重大脑

一定要去多看数据，一定要明白自己在干什么，一定不要忽略“小错误”“老掉牙的手段”

但是也要明白大脑可能使得模型陷入局部最优，模型要加上人的智能



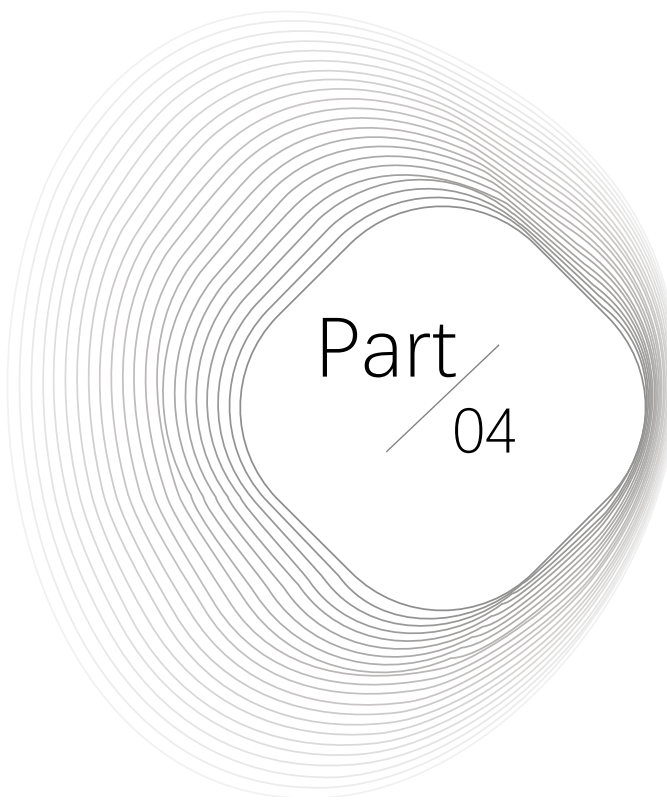
暴力方法之外： 内置CV的模型

<code>linear_model.ElasticNetCV</code> (*[, l1_ratio, ...])	Elastic Net model with iterative fitting along a regularization path.
<code>linear_model.LarsCV</code> (*[, fit_intercept, ...])	Cross-validated Least Angle Regression model.
<code>linear_model.LassoCV</code> (*[, eps, n_alphas, ...])	Lasso linear model with iterative fitting along a regularization path.
<code>linear_model.LassoLarsCV</code> (*[, fit_intercept, ...])	Cross-validated Lasso, using the LARS algorithm.
<code>linear_model.LogisticRegressionCV</code> (*[, Cs, ...])	Logistic Regression CV (aka logit, MaxEnt) classifier.
<code>linear_model.MultiTaskElasticNetCV</code> (*[, ...])	Multi-task L1/L2 ElasticNet with built-in cross-validation.
<code>linear_model.MultiTaskLassoCV</code> (*[, eps, ...])	Multi-task Lasso model trained with L1/L2 mixed-norm as regularizer.
<code>linear_model.OrthogonalMatchingPursuitCV</code> (*)	Cross-validated Orthogonal Matching Pursuit model (OMP).
<code>linear_model.RidgeCV</code> ([alphas, ...])	Ridge regression with built-in cross-validation.
<code>linear_model.RidgeClassifierCV</code> ([alphas, ...])	Ridge classifier with built-in cross-validation.



ensemble.RandomForestClassifier ([...])	A random forest classifier.
ensemble.RandomForestRegressor ([...])	A random forest regressor.
ensemble.ExtraTreesClassifier ([...])	An extra-trees classifier.
ensemble.ExtraTreesRegressor ([n_estimators, ...])	An extra-trees regressor.
ensemble.GradientBoostingClassifier (*[, ...])	Gradient Boosting for classification.
ensemble.GradientBoostingRegressor (*[, ...])	Gradient Boosting for regression.



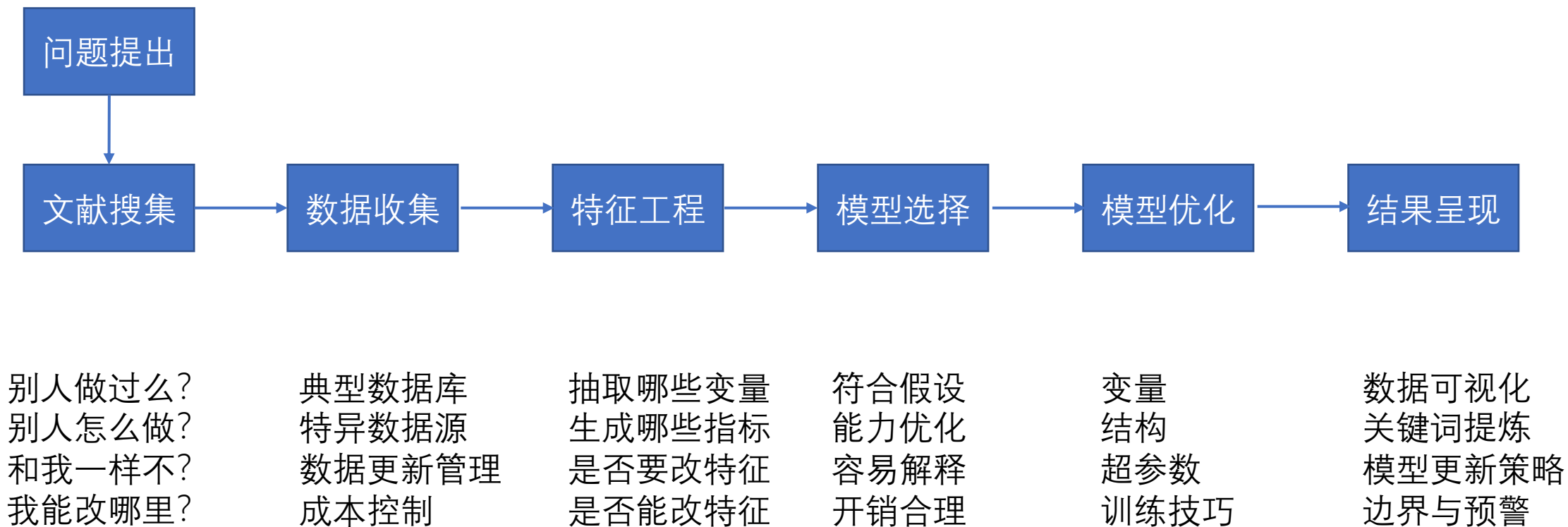


Part
04

整体流程

- 哪些步骤
- 哪些选择
- 哪些方法







2⁰₂3

THANKS

