1.2.2 k-折交叉验证法 k-Fold Cross-Validation ——概述

- 交叉验证主要思想：数据集合中的每个样本都有机会被测试
- k–折交叉验证步骤：

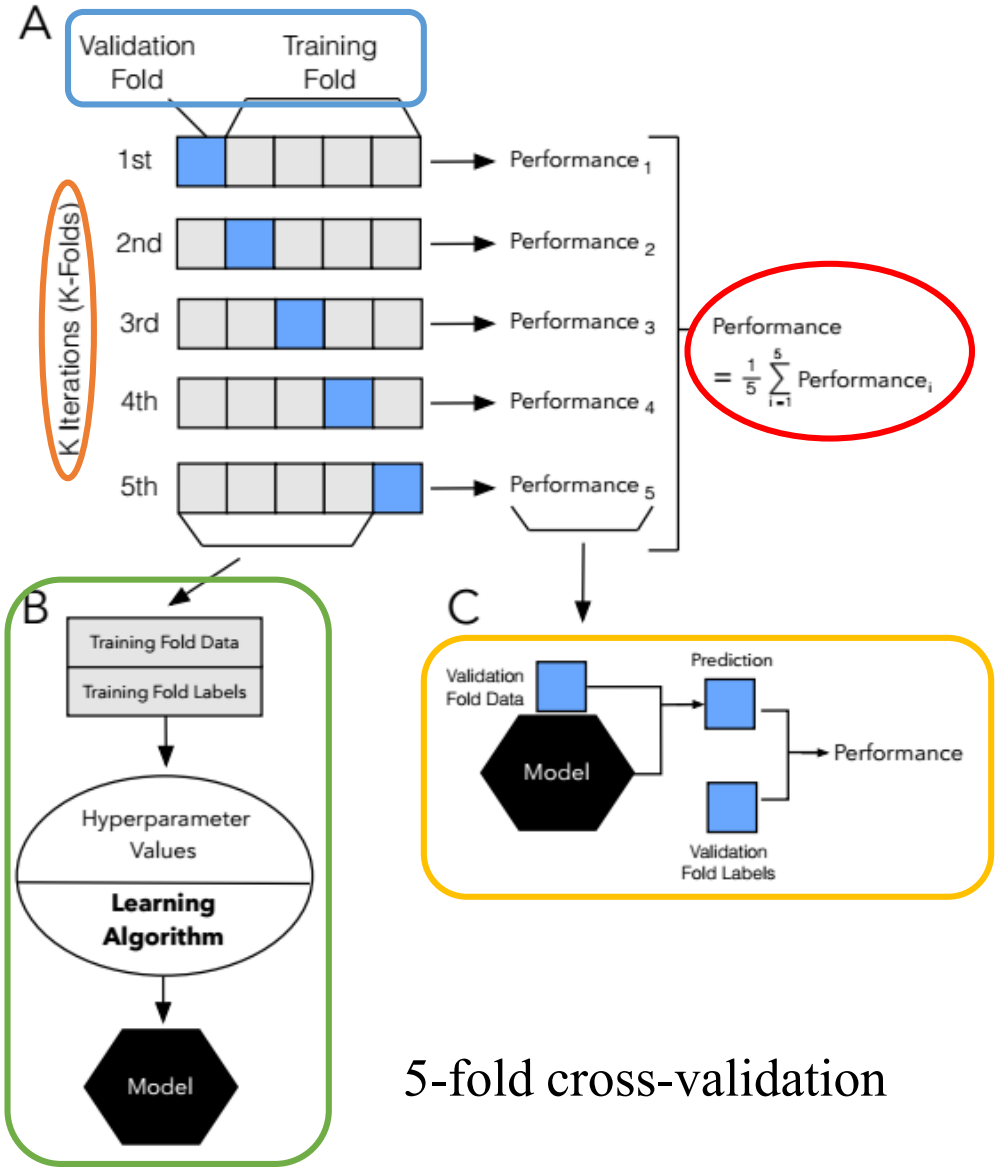
(1)在数据集上迭代k次

(2)在每次迭代中,将数据集分为k个部分

a. k–1部分进行训练

b.第k部分用于验证

(3)计算性能的平均值



5-fold cross-validation

1.2.2 k-折交叉验证法——k的选择

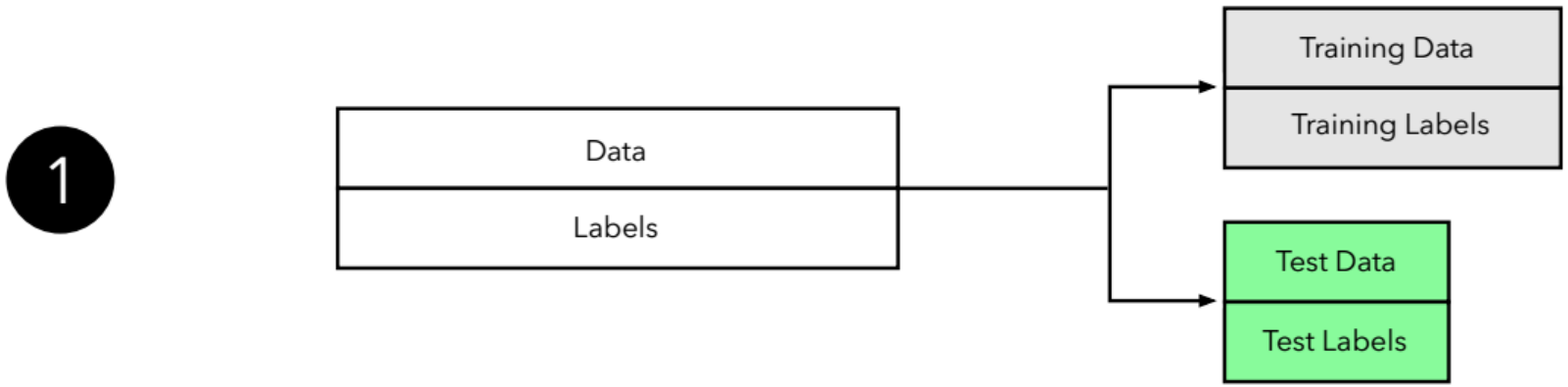
偏差和方差
会怎么变化？

- 增加k值后的k折交叉验证：
  - 性能估计的偏差**减小(更准确)**
  - 性能估计的方差**增加(更多的可变性)**
  - 计算成本**增加(更多的迭代，在拟合过程中更大的训练集)**
- 在实践中，一般选择k = 5，k=10

1.2.2 k-折交叉验证法——比较

- k-折交叉验证与重复留出法的区别:
  - k-折交叉验证：**使用所有数据**进行训练和测试，通过使用更多的训练数据来减少验证集的偏差
  - 重复留出法：**重复将相对较大的数据集**作为测试数据，随机抽取导致有些样本可能永远都不会是测试集的一部分

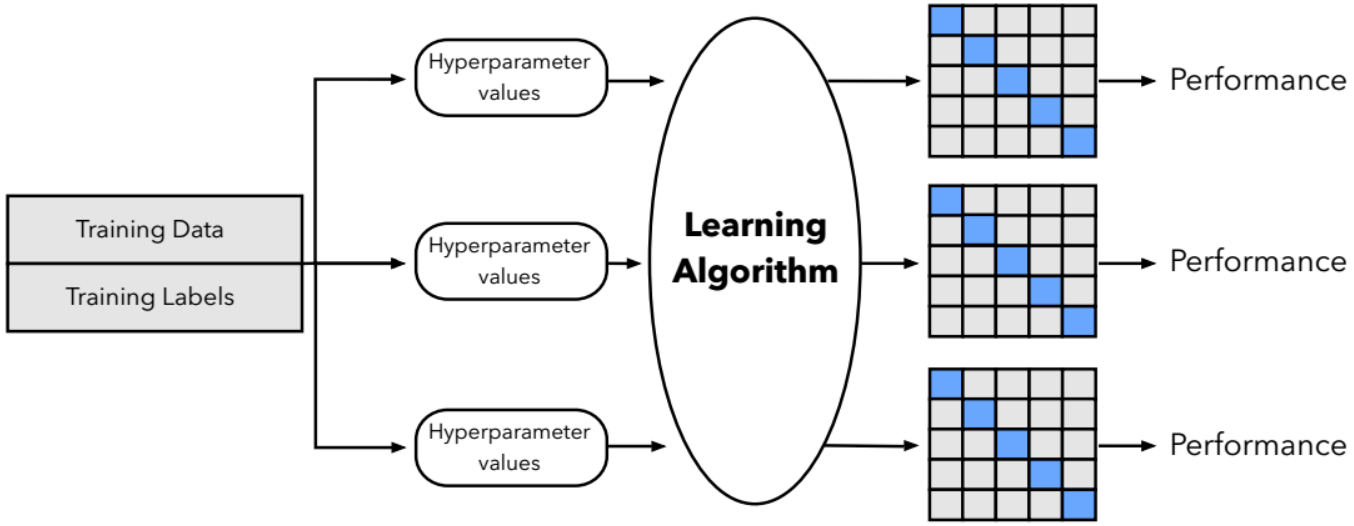
1.2.2 k-折交叉验证法评估
k-Fold Cross-Validation Evaluation



- 类似于留出法，将可用数据随机划分为两个子集:
- 一个**训练集**和一个**测试集**

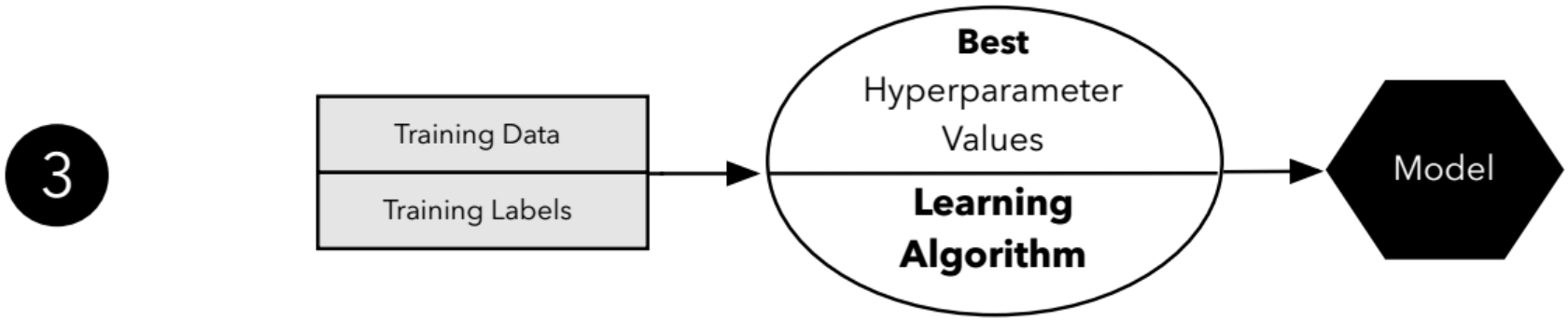
1.2.2 k-折交叉验证法评估
k-Fold Cross-Validation Evaluation

2



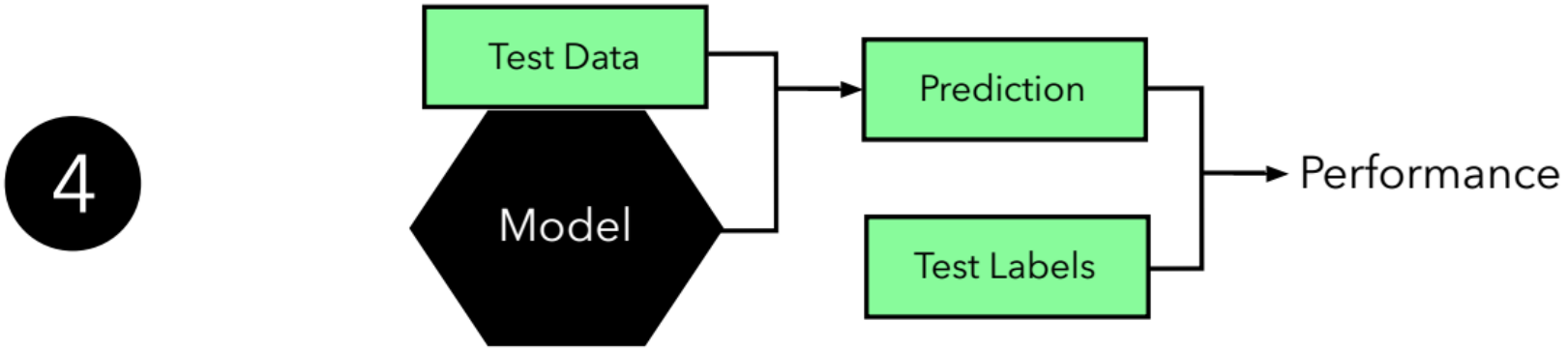
- 选择**不同的超参数**，将k-折交叉验证方法应用于训练集，得到了**多个模型和性能估计**

1.2.2 k-折交叉验证法评估
k-Fold Cross-Validation Evaluation



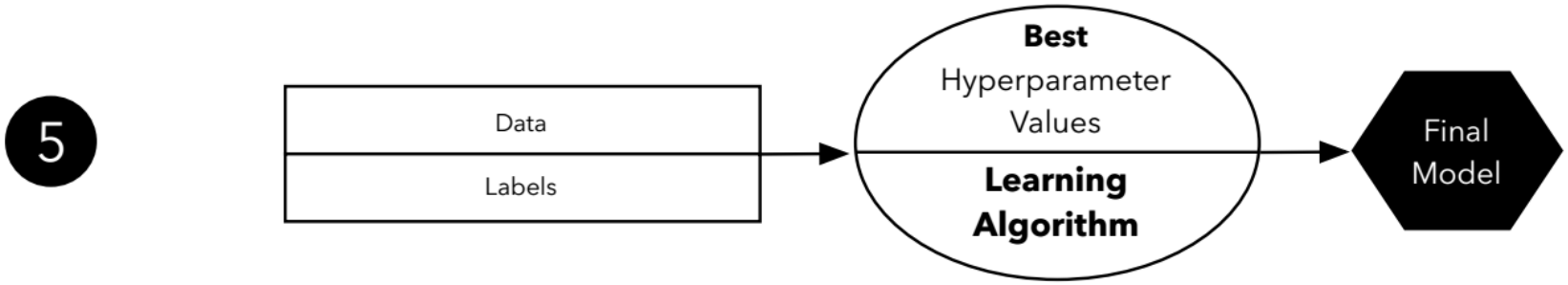
- 使用由k-折交叉验证产生**最佳性能的超参数设置**
- 使用**完整的训练集**与这些设置进行模型拟合

1.2.2 k-折交叉验证法评估
k-Fold Cross-Validation Evaluation



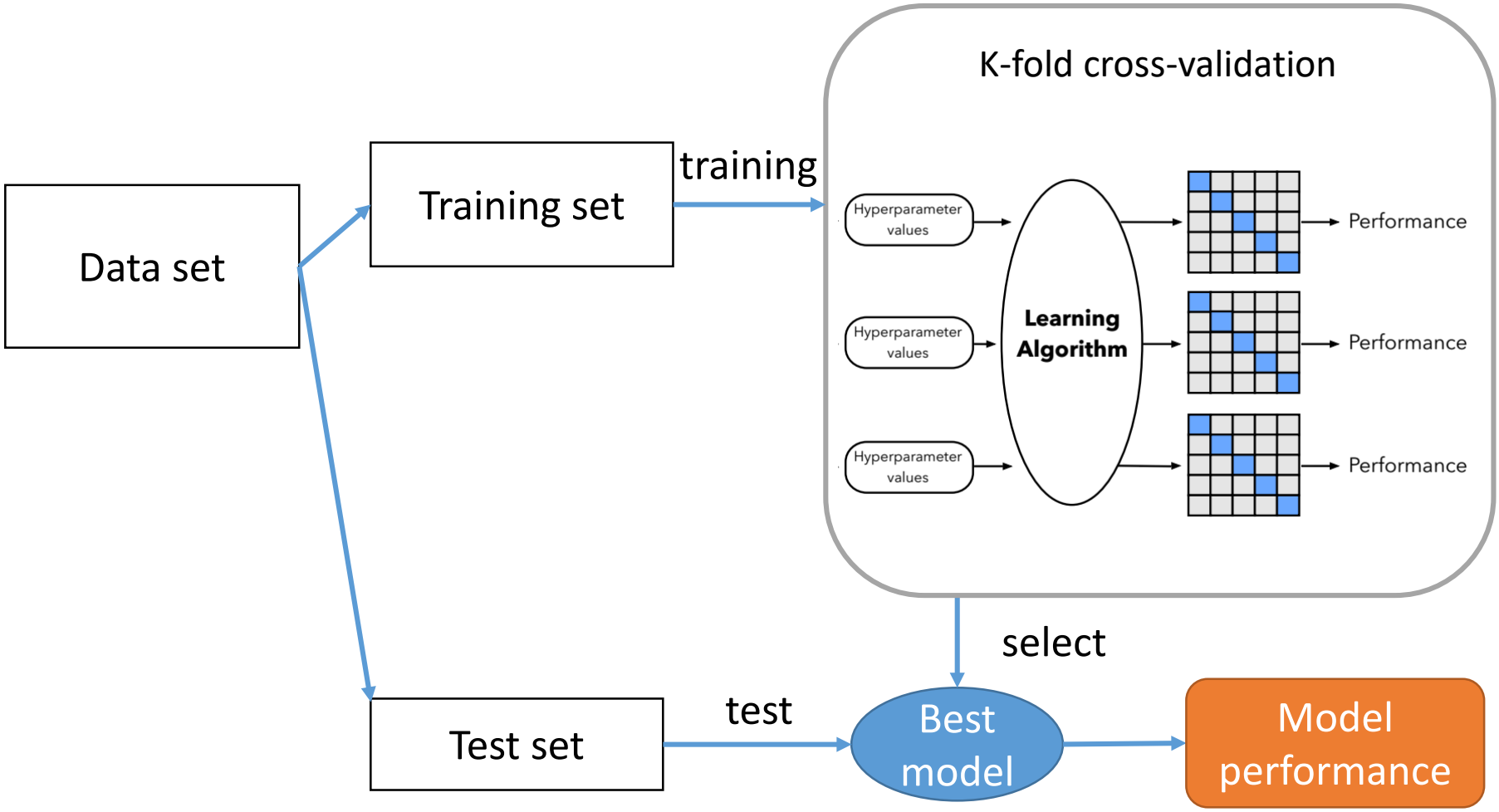
- 使用前面保留的**独立测试集**来评估获得的模型

1.2.2 k-折交叉验证法评估
k-Fold Cross-Validation Evaluation

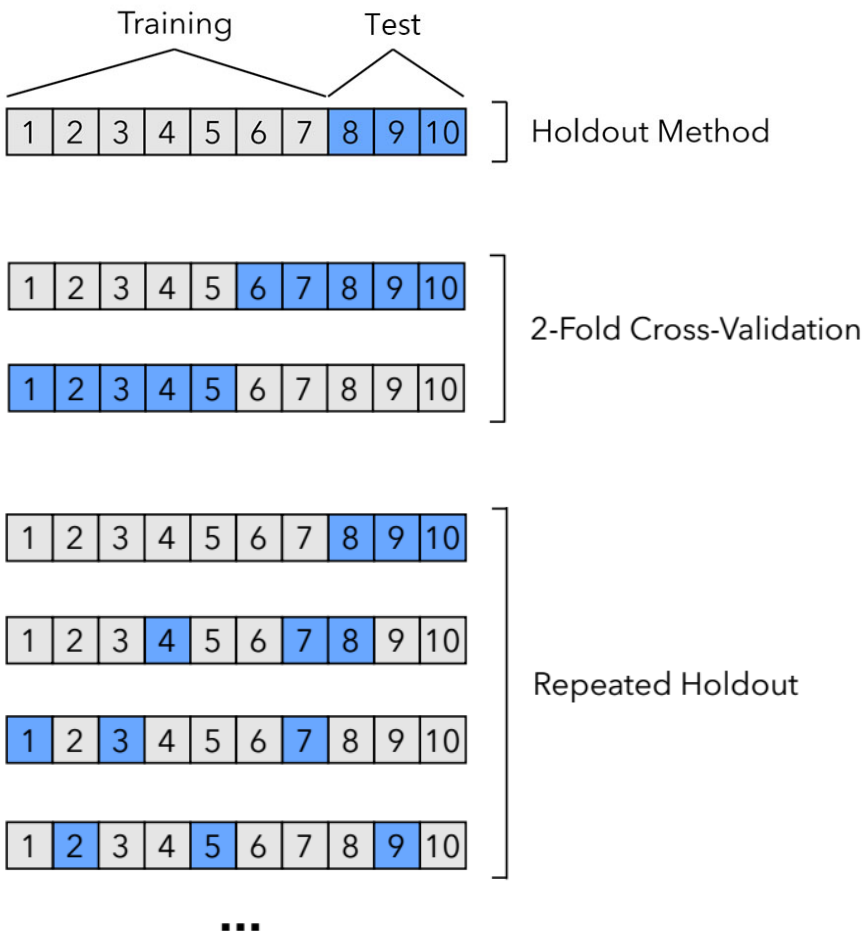


- 完成评估后，**拟合所有数据**(训练和测试数据集的组合)，得到部署模型（可选步骤）

1.2.2 k-折交叉验证法评估——小结
k-Fold Cross-Validation Evaluation



1.2.3 特殊交叉验证 k=2

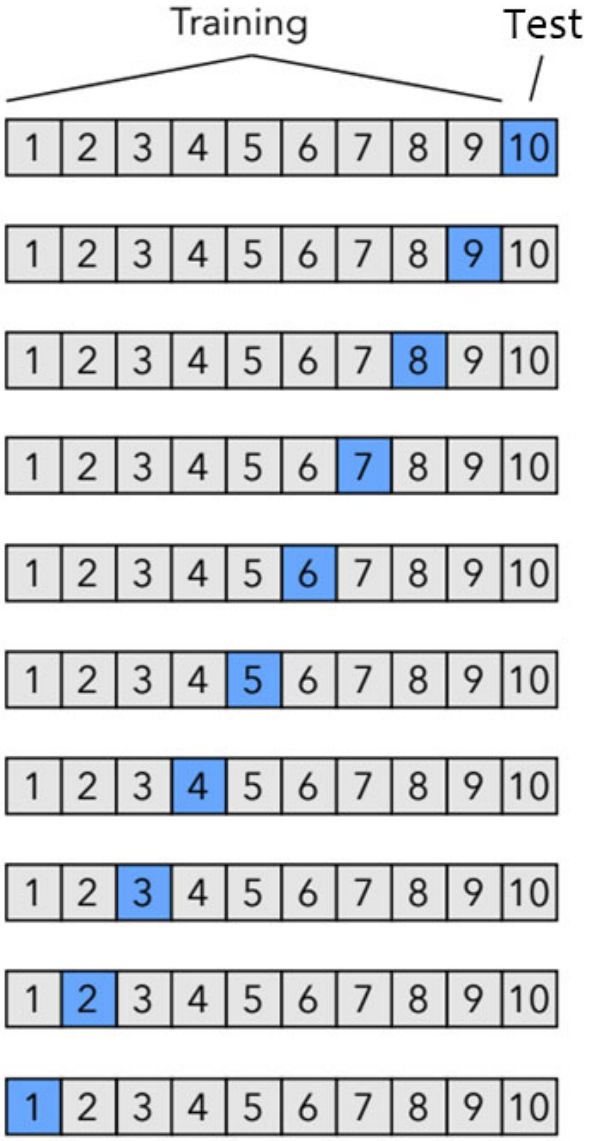


- 2-折交叉验证：使用两次留出法，互换训练集和验证集

例如：

- 使用50%的数据进行训练和50%的数据进行验证
- 交换这些数据,重复训练和验证的步骤
- 最终计算平均性能

1.2.3 特殊交叉验证k=n——LOOCV



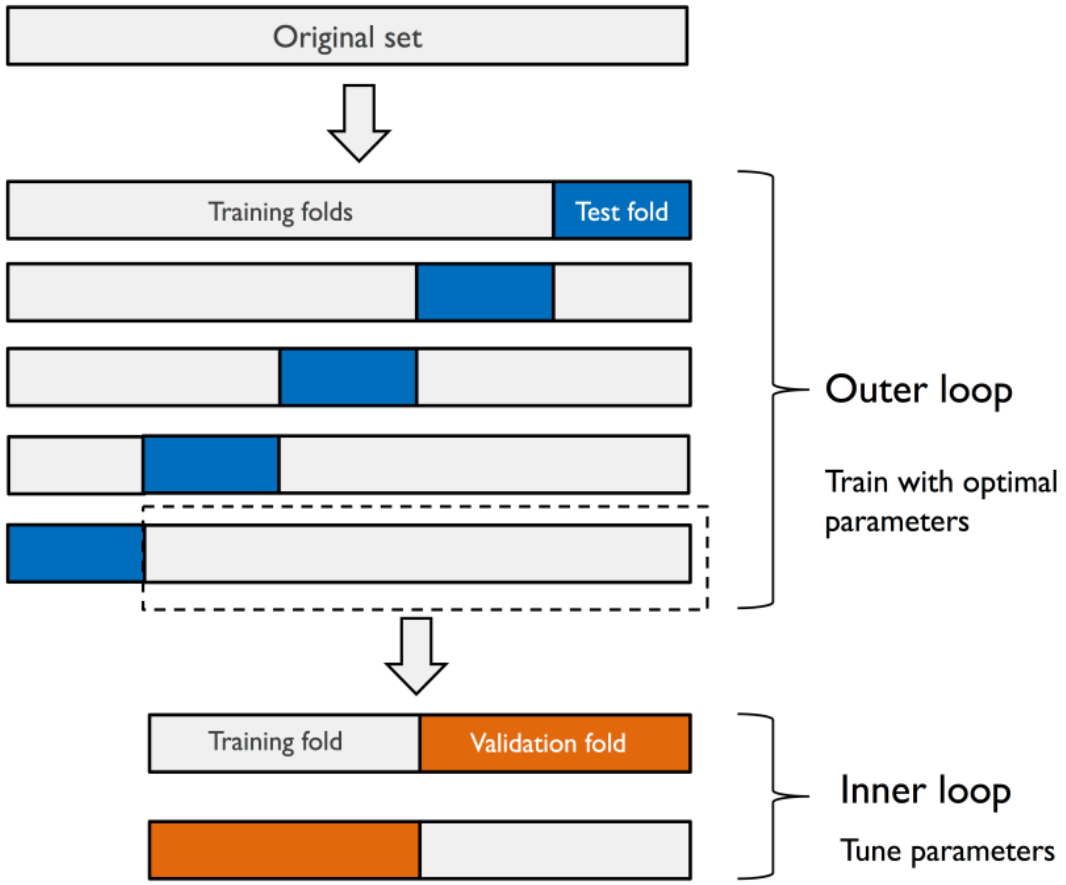
- k=n，将折叠数等于训练样本的数量
- 称为Leave-One-Out Cross-Validation（LOOCV）
- 训练集：n-1个样本
- 验证集：1个样本
- n次迭代，计算成本很高
- 对小型数据集很有用

1.2.3 特殊交叉验证k=n——LOOCV

- LOOCV训练集中有n-1个样本，被认为对真实误差的预测是近似无偏的
- 但测试集只包含一个样本，具有很高的方差
- 重复k折交叉验证得到“更可靠”的估计

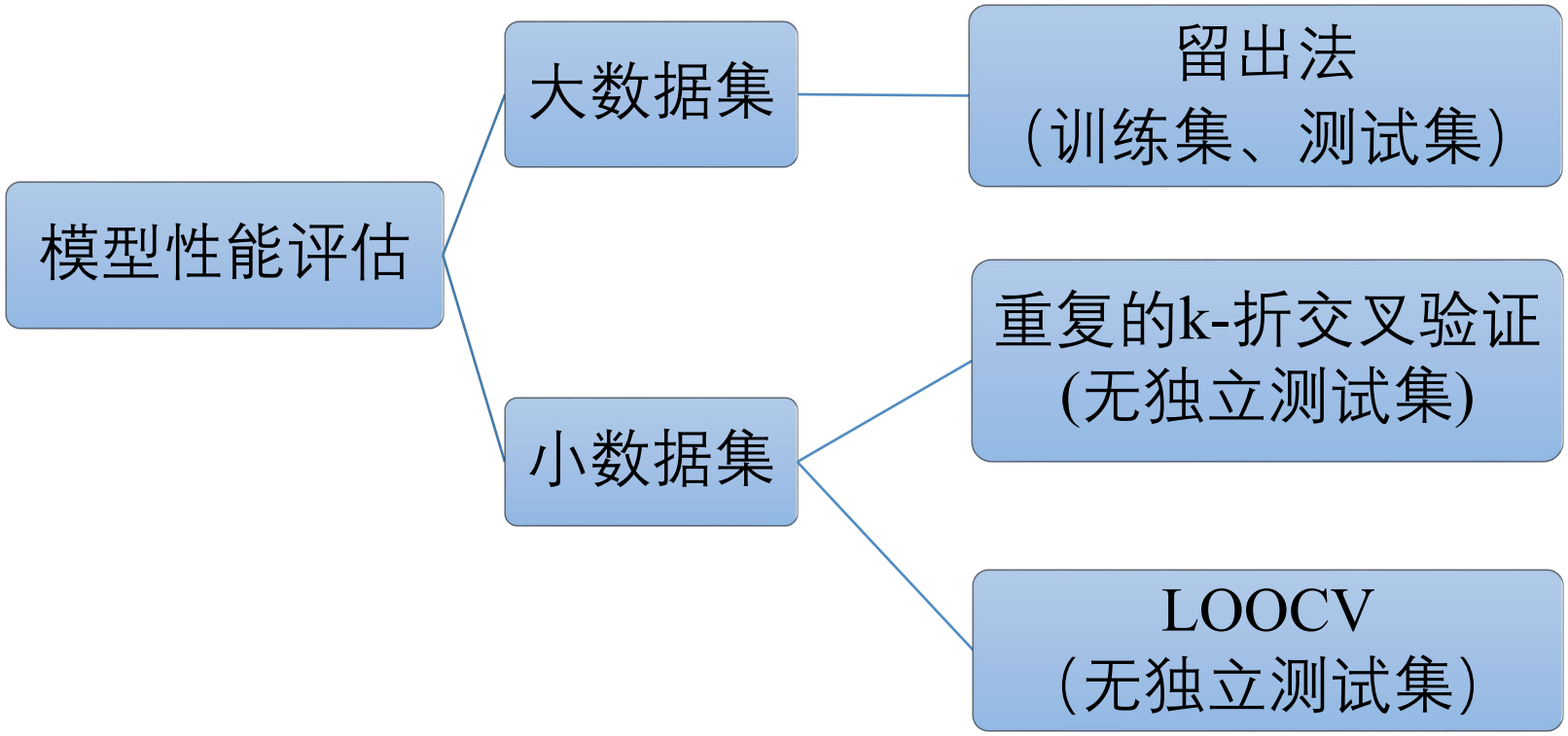
1.2.3 特殊交叉验证——嵌套交叉验证
Nested Cross-Validation

- 将调参和模型选择结合
- **内层交叉验证**(innner loop): 用于模型选择，超参数调优
- **外层交叉验证**(outer loop): 用于模型评估，外层每一折都使用内层得到的最优参数组合进行训练

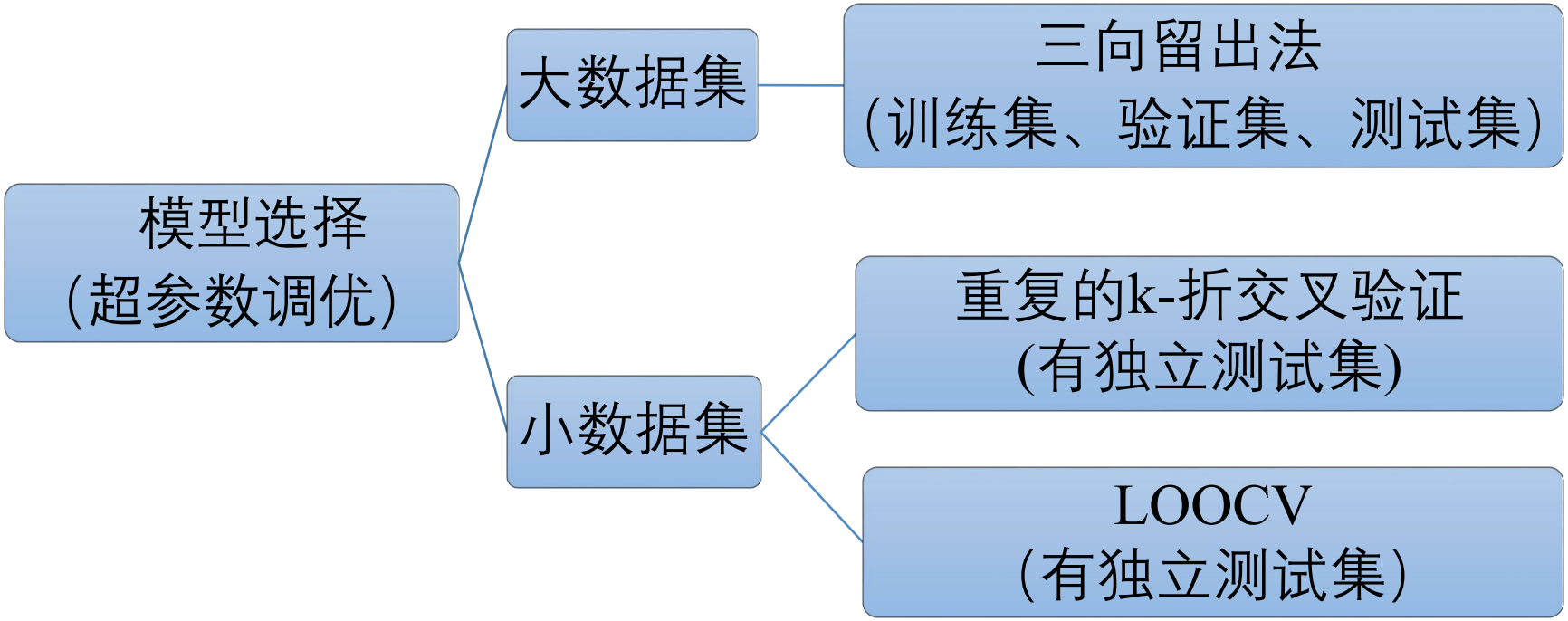


5*2 cross-validation

1.2.4 模型选择——小结1

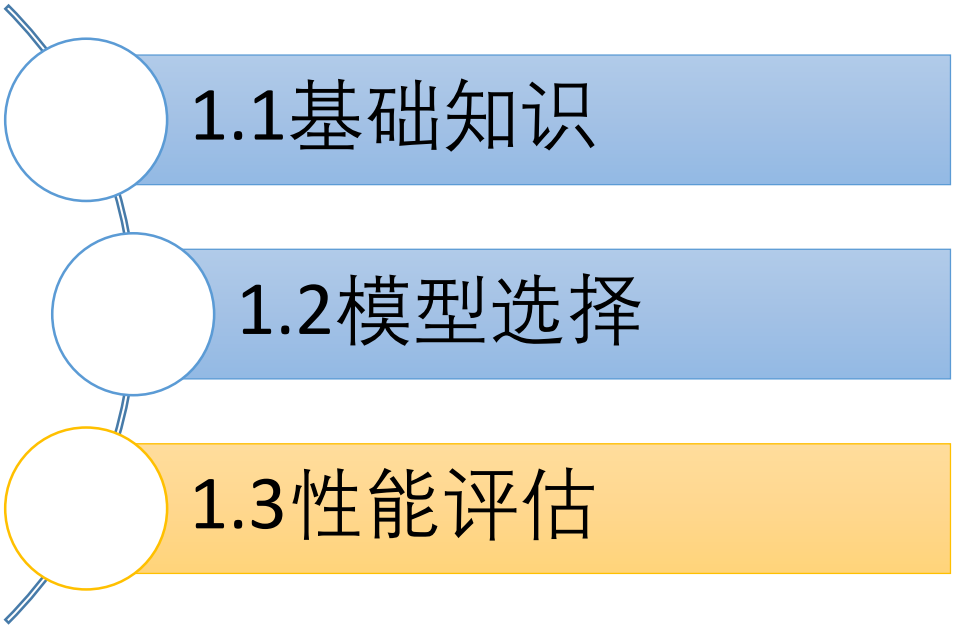


1.2.4 模型选择——小结2


模型选择
(超参数调优)
大数据集
三向留出法
(训练集、验证集、测试集)
小数据集
重复的k-折交叉验证
(有独立测试集)
LOOCV
(有独立测试集)

1.2.4 模型选择——小结3

测试方法	数学表达	注意事项	优缺点
留出法 Hold-out	$D = S \cup T$ $S \cap T = \emptyset$	可采用分层抽样，重复多次取平均值评估结果	测试集小，评估结果方差大； 训练集小，评估结果偏差大
交叉验证法 Cross-validation	$D = D_1 \cup \cdots \cup D_k$ $D_i \cap D_j = \emptyset (i \neq j)$	可取p次k-折交叉验证求平均值评估结果	稳定性很大程度上取决于k
留一交叉验证法 Leave-one-out cross-validation LOOCV	$D = D_1 \cup \cdots \cup D_k$ $D_i \cap D_j = \emptyset (i \neq j)$ $k = \lvert D \rvert$	每次使用一个样本进行验证	不受随机样本划分影响，但数据量大时计算量大


1.1基础知识
1.2模型选择
1.3性能评估

1.3模型性能评估

- 1.3.1混淆矩阵
  - 2*2混淆矩阵
  - 多元混淆矩阵
- 1.3.2查准率、查全率、F1
- 1.3.3灵敏度与特异度
- 1.3.4 FPR与TPR
- 1.3.5ROC曲线
  - ROC曲线
  - ROC曲线绘制
  - ROC-AUC

1.3.1混淆矩阵Confusion Matrix ——2×2混淆矩阵

		Predicted class: P	Predicted class: N
Actual Class	P	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)

2×2混淆矩阵

查准率 Precision $$P = \frac{TP}{TP + FP}$$

查全率 Recall $$R = \frac{TP}{TP + FN}$$

F1 $$F_1 = 2 \cdot \frac{PRE \cdot REC}{PRE + REC}$$

1.3.1混淆矩阵Confusion Matrix ——多元混淆矩阵

Predicted Labels

True Labels	Class 0	Class 1	Class 2
Class 0	T(0,0)		
Class 1		T(1,1)	
Class 2			T(2,2)

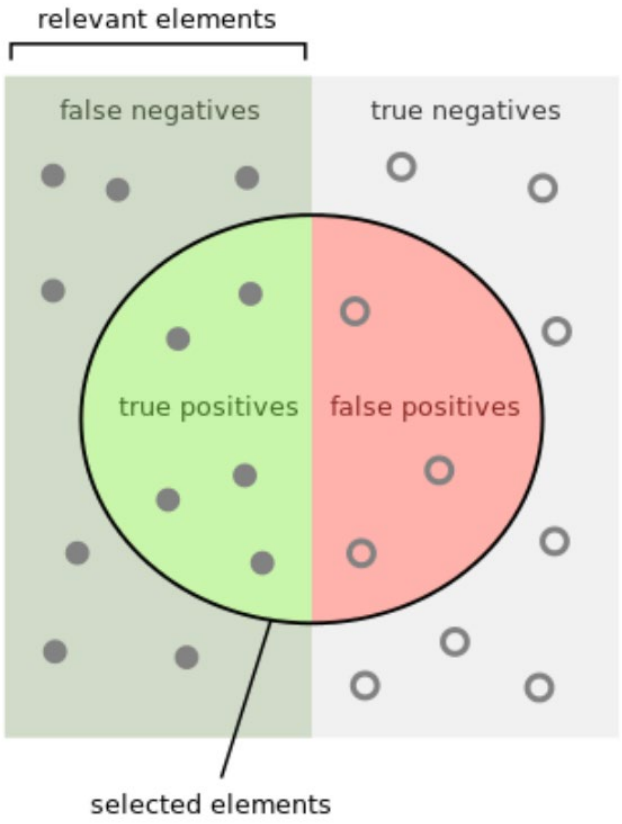
Predicted Labels

True Labels	Class 0	Class 1	Class 2
Class 0	3	0	0
Class 1	7	50	12
Class 2	0	0	18

$$ACC = \frac{T}{n}$$

$$ACC = \frac{3 + 50 + 18}{90} \approx 0.79$$

1.3.2查准率与查全率



How many selected items are relevant?

Precision = 

$$PRE = \frac{TP}{TP + FP}$$

How many relevant items are selected?

Recall = 

$$REC = \frac{TP}{TP + FN}$$

1.3.3灵敏度与特异度

- 灵敏度sensitivity=查全率recall

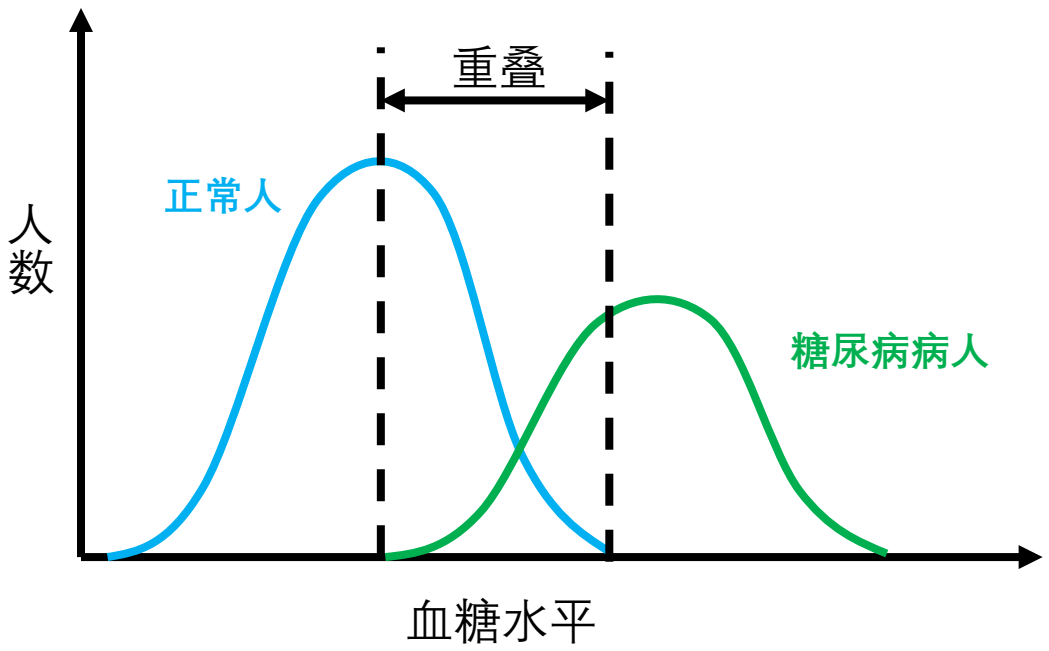
$$SEN = TPR = REC = \frac{TP}{P} = \frac{TP}{FN + TP}$$

- 特异度specificity

$$SPC = TNR = \frac{TN}{N} = \frac{TN}{FP + TN}$$

1.3.3灵敏度与特异度举例

糖尿病病人筛查



灵敏度高=漏诊率低

特异度高=误诊率低

1.3.4 FPR与TPR

- **False Positive Rate假正例率：**

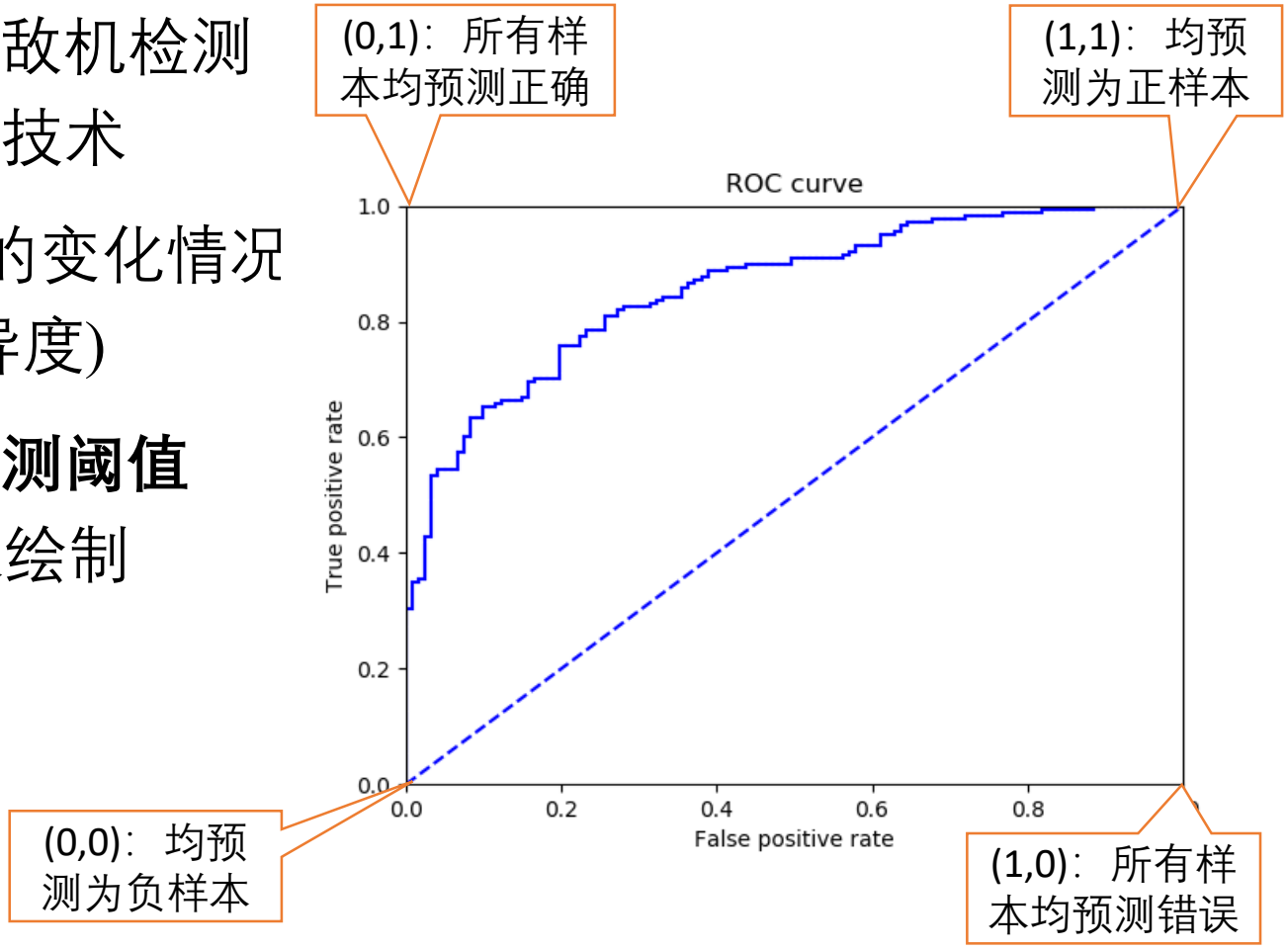
$$FPR = \frac{FP}{N} = \frac{FP}{FP + TN} \quad = 1 - \text{specificity}$$

- **True Positive Rate真正例率：**

$$TPR = \frac{TP}{P} = \frac{TP}{FN + TP} \quad = \text{sensitivity}$$

1.3.5 ROC曲线

- ROC(Receiver Operating Characteristic) “受试者工作特征”
- 源于二战中用于敌机检测的雷达信号分析技术
- 表示FPR与TPR的变化情况(灵敏度与非特异度)
- 可以通过改变**预测阈值**（Threshold）来绘制



1.3.5 ROC曲线绘制

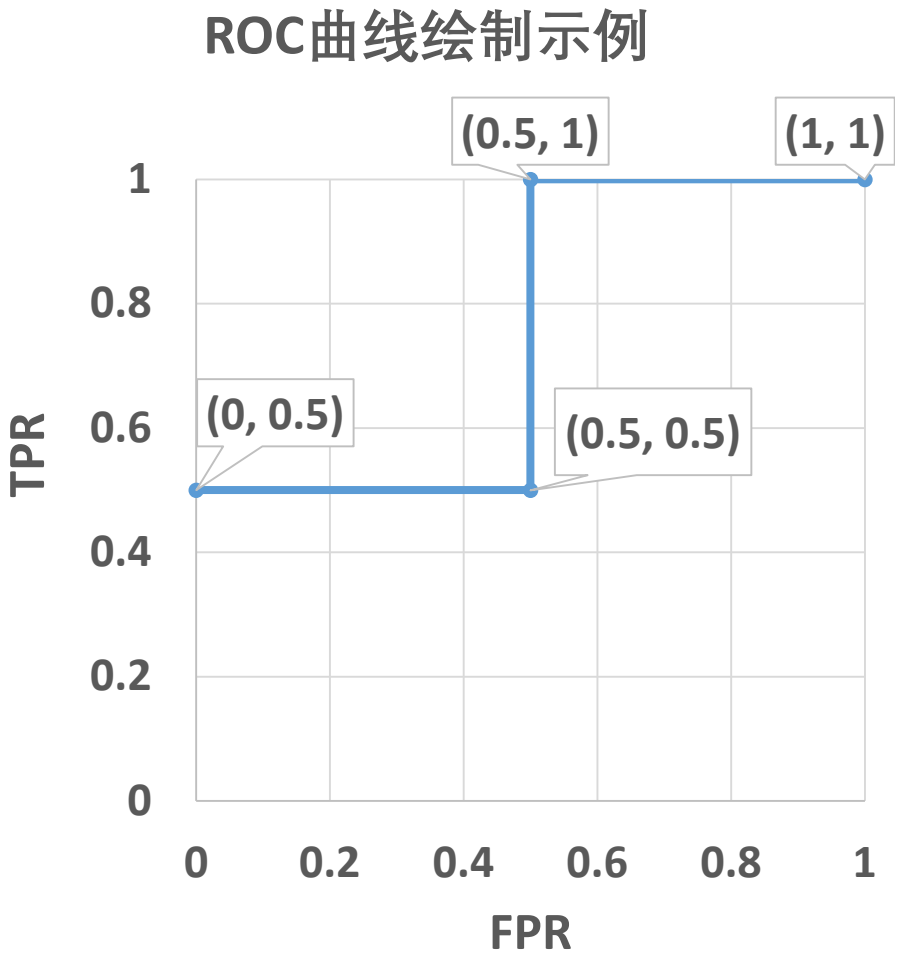
- 二分类问题
  - y_true: 真实值
  - y_score: 预测概率值
  - y_pre: 预测值
- 阈值: 当预测概率值大于阈值时判定为真值

名称	阈值	值1	值2	值3	值4	FPR	TPR
y_true	-	**0**	**1**	**0**	**1**	-	-
y_score	-	0.1	0.35	0.4	0.8	-	-
y_pre	0.1	1	1	1	1	2/2	2/2
y_pre	0.35	0	1	1	1	1/2	2/2
y_pre	0.4	0	0	1	1	1/2	1/2
y_pre	0.8	0	0	0	1	0/2	1/2

1.3.5 ROC曲线绘制

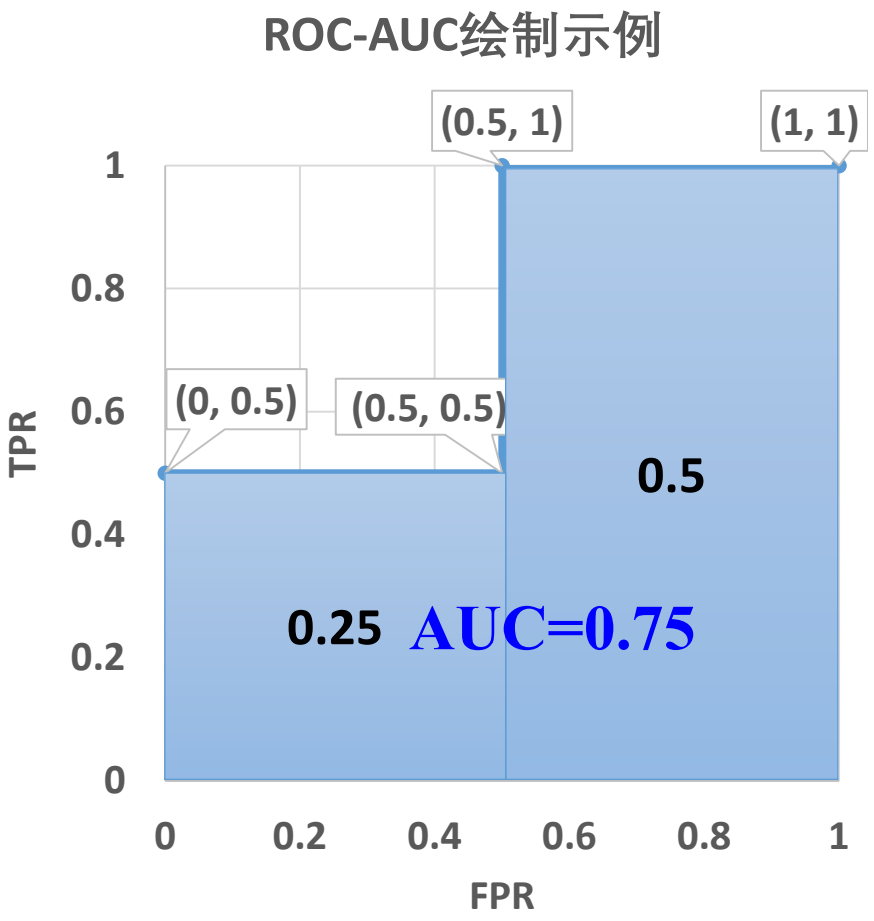
名称	阈值	值1	值2	值3	值4	FPR	TPR
y_true	-	**0**	**1**	**0**	**1**	-	-
y_score	-	0.1	0.35	0.4	0.8	-	-
y_pre	0.1	1	1	1	1	2/2	2/2
y_pre	0.35	0	1	1	1	1/2	2/2
y_pre	0.4	0	0	1	1	1/2	1/2
y_pre	0.8	0	0	0	1	0/2	1/2

FPR=[1, 0.5, 0.5, 0]
TPR=[1, 1, 0.5, 0.5]



1.3.5 ROC-AUC

- AUC(Area Under the Curve)
- ROC曲线下的面积
- AUC值：随机挑选一个正样本和负样本，根据分类器计算得到的Score值将这个正样本排在负样本前面的概率
- 反应了模型的好坏
  - 越接近1，性能越好
  - 越接近0.5，性能越差



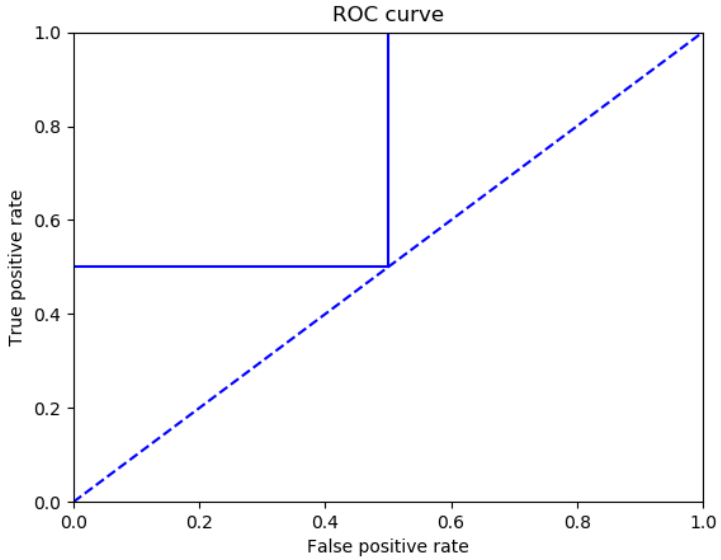
1.3.5 ROC-AUC绘制代码

```
def plotROC(predScores, classLabels):
    cur = (1.0, 1.0)  # 光标绘制初始位置
    ySum = 0.0  # 用于计算AUC
    numPosClas = sum(array(classLabels) == 1.0)  # 统
计正例个数
    yStep = 1 / float(numPosClas)  # y轴步长
    xStep = 1 / float(len(classLabels) - numPosClas)
# x轴步长
    sortedIndicies = predScores.argsort()  # 对预测值进
行从小到大的排序，并提取对应索引
    # 构建画笔
    fig = plt.figure()
    fig.clf()
    ax = plt.subplot(111)
```

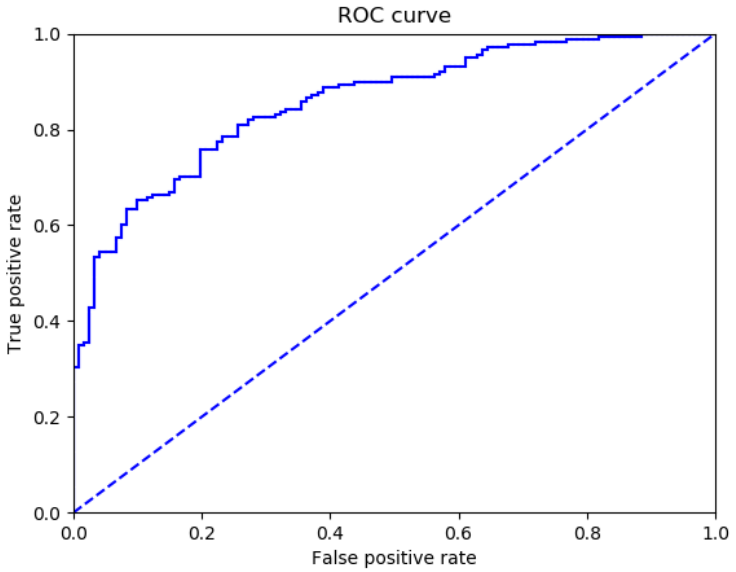
1.3.5 ROC-AUC绘制代码

```
# 遍历所有预测值索引，并根据真实值绘制线段
for index in sortedIndicies.tolist()[0]:
    if classLabels[index] == 1.0:
        # 延y轴下降一个步长。减小TPR
        delX = 0
        delY = yStep
    else:
        # 延x轴下降一个步长，减小FPR
        delX = xStep
        delY = 0
        ySum += cur[1]
    # draw line from cur to (cur[0]-delX,cur[1]-delY)
    ax.plot([cur[0], cur[0] - delX], [cur[1], cur[1] - delY], c='b')
    cur = (cur[0] - delX, cur[1] - delY)
ax.plot([0, 1], [0, 1], 'b--')
ax.axis([0, 1, 0, 1])
plt.show()
print("the Area Under the Curve is: ", ySum * xStep)
```

1.3.5 ROC-AUC举例

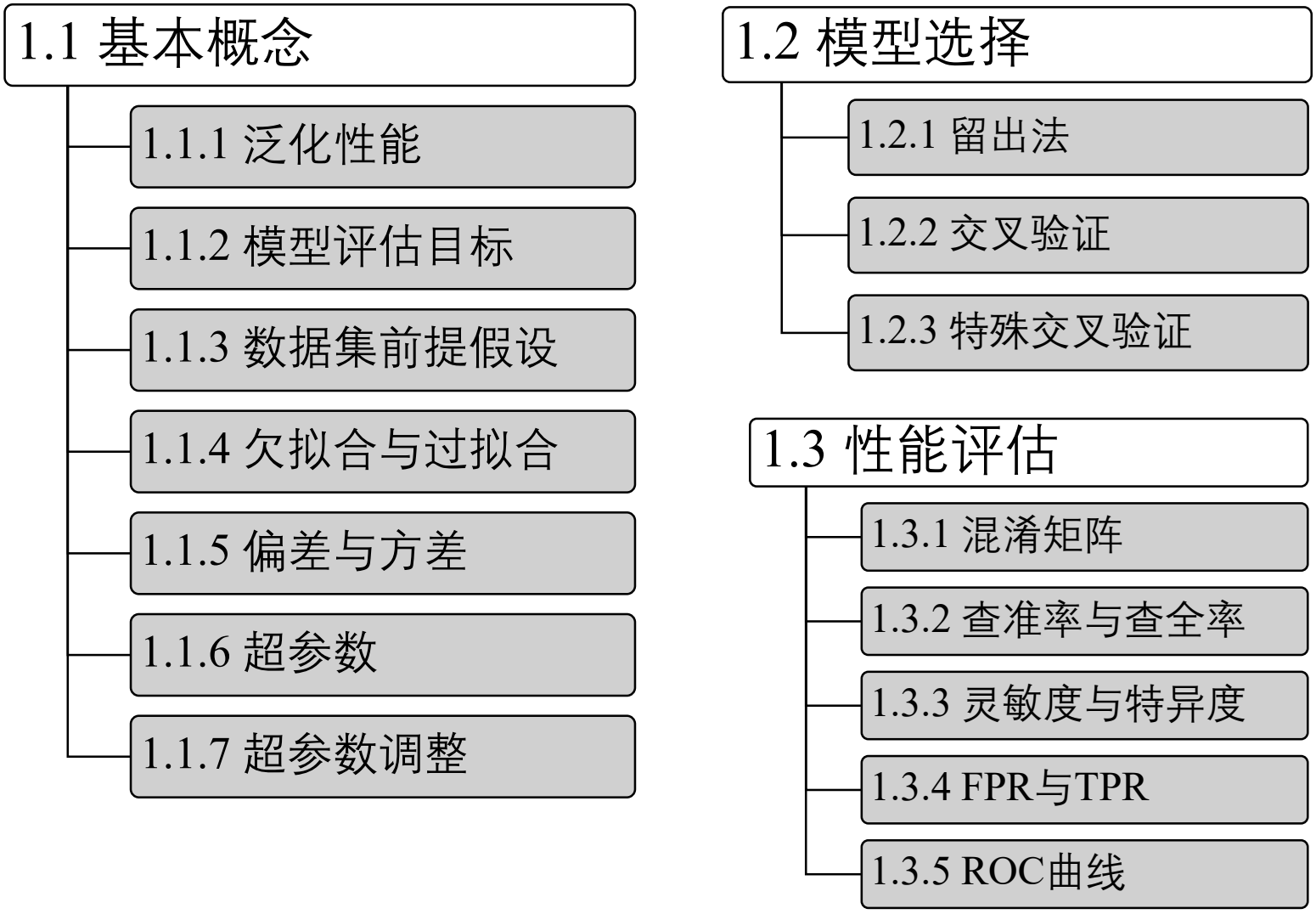


the Area Under the Curve is: 0.75



the Area Under the Curve is: 0.8583

小结


1.1 基本概念
1.1.1 泛化性能
1.1.2 模型评估目标
1.1.3 数据集前提假设
1.1.4 欠拟合与过拟合
1.1.5 偏差与方差
1.1.6 超参数
1.1.7 超参数调整
1.2 模型选择
1.2.1 留出法
1.2.2 交叉验证
1.2.3 特殊交叉验证
1.3 性能评估
1.3.1 混淆矩阵
1.3.2 查准率与查全率
1.3.3 灵敏度与特异度
1.3.4 FPR与TPR
1.3.5 ROC曲线

迁移学习

迁移学习

- 迁移学习能有效降低数据量、计算量和计算时间。

- 迁移学习不是一种算法而是一种机器学习思想，应用到深度学习时称为微调（Fine-tune)。

迁移学习

1. 在大数据库上学习，例如ImageNet

2. 训练自己任务中的**小规模**数据集，把预训练的CNN模型当做特征提取器

固定这些层的参数，在学习过程中不更新

只训练此部分

3. 训练自己任务中的中等规模数据集

固定这些层的参数，在学习过程中不更新

只训练此部分

或者下载预训练模型，Caffe Model Zoo

经验：迁移学习(Fine tuning微调)时，学习率设为原始的0.1倍

迁移学习

	自己任务的数据集与预训练模型的数据集**相似**	自己任务的数据集与预训练模型的数据集**差异较大**
自己任务数据集较小	只学习CNN的分类器部分	是从网络的某层开始取出特征，然后训练SVM分类器
自己任务数据集较大	微调少部分CNN层	微调大部分CNN层

参考资料

- 斯坦福大学机器学习课程 吴恩达 第十章
- 机器学习/周志华 清华大学出版社 2016 P23-47
- 深度学习/Ian Goodfellow,Yoshua Bengio,Aaron Courville 人民邮电出版社 2017 P70-82
- 机器学习实战/Harrington,P, 人民邮电出版社 2013 P128-133
- 模型选择与评估部分课件与代码 Sebastian Raschka 第8-12章
  - https://github.com/rasbt/stat479-machine-learning-fs18