《机器学习》周志华

在计算机系统中，“经验”通常以数据形式存在，因此，机器学习所研究的主要内容是关于在计算机上从数据中产生“模型”的算法，即学习算法

模型：泛指从数据中学得的结果

模式：局部性结果（如一条规则）

学得模型对应了关于数据的某处潜在的规律，因此亦称为“假设”

机器学习的目标：使学得的模型能很好地适用于”新样本”

尽管训练集通常只是样本空间的一个很小的采样，我们仍希望它能很好地反映出样本空间的特性，否则就很难期望在训练集上学得的模型能在整个样本空间上都工作得很好。通常假设样本空间中总体样本服从一个未知“分布”D，我们获得的每个样本都是独立地从这个分布上采样获得的，即独立同分布i.i.d

学习过程看作一个在所有假设组成的空间中进行搜索的过程，搜索目标是找到与训练集“匹配”的假设，假设的表示一旦确定，假设空间及其规模大小就确定了。如我们的假设空间由形如“(色泽=?) and (根蒂=?) and (敲声=?)的可能取值所形成的假设组成。可以有许多策略对这个假设空间进行搜索，搜索过程中可以不断删除与正例不一致的假设、和与反倒一致的假设，最终将会获得与训练集一致（即对所有训练样本能够进行正确判断）的假设。但学习过程是基于有限样本训练集进行的，因此，可能有多个假设与训练集一致，应该采用哪一些模型（假设）呢？由机器学习算法在学习过程中对某种类型假设的偏好决定，即归纳偏好(inductive bias)。

任何一个有效的机器学习算法必有其归纳偏好。比如若认为相似的样本应有相似的输出（例如，在各种属性上都很相像的西瓜，成熟程度应该比较接近），则对应的学习算法可能偏好比较”平滑“的曲线，而不是比较”崎岖“的曲线。再比如Occam’s razor原则，若有多个假设与观察一致，则选最简单的那个。

归纳偏好对应了学习算法本身所做出的关于”什么样的模型更好“的假设，在具体的现实问题中，这个假设是否成立，即算法的归纳偏好是否与问题本身匹配，大多数时候直接决定了算法能否取得好的性能。

若所有”问题“出现的机会相同，或所有问题同等重要，那么无论学习算法a多聪明，学习算法b多笨拙，它们的期望性能是相同的，这就是”没有免费的午餐“定理。然后实现情形并不是这样，很多时候，我们只关注自己正在试图解决的问题（例如某个具体应用任务）。所以要谈论算法的相对优劣，必须要针对具体的学习问题，在某些问题上表现好的学习算法，在另一些问题上却可能不尽如人意，学习算法自身的归纳偏好与问题是否相配，往往会起到决定性的作用。

1950 – 1970 推理期

基于符号知识表示、通过演绎推理技术

1970 - 知识期

基于符号知识表示，通过获取和利用领域知识来建立专家系统

机械学习：没有进行真正的学习，仅是在进行信息存储与检索

归纳学习：从样例中学习，即从训练样例中归纳出学习结果

= 符号主义学习 （产生明确的概念，如决策树，基于逻辑学习）

连接主义学习 （产生的是黑箱模型，如神经网络，深度网络）

统计学习 （如SVM 以及更一般的核方法）

数据挖掘：从海量数据中发掘知识 = 数据库+机器学习+统计学

模型评估与选择 （数据集=训练集+测试集）

经验误差（训练误差）：学习器在训练集上的误差

Overfitting: 应该从训练样本中尽可能学出适用于所有潜在样本的“普遍规律“，如果学习器把训练样本学得“太好”了的话，很可能已经把样本自身的一些特点当作了所有潜在样本都会具有的一般性质，这样就会导致泛化性能下降

很多因素导致过拟合，最常见的情况是由于学习能力过于强大，以至于把训练样本所包含的不太一般的特性都学到了，过拟合是无法彻底避免的，我们所能做的只是“缓解”，或者说减小其风险。

测试误差：测试集上的“测试误差”作为泛化误差的近似，通常我们假设测试样本也是从样本真实分布中独立分布采样而得。

训练集与测试集如何划分？

训练/测试集的划分要尽可能保持数据分布的一致性，避免因数据划分过程引入额外的偏差而对最终结果产生影响

分层采样(stratified sampling): 保留类别比例的采样方式

留出法hold-out: 单次使用hold-out得到的估计结果往往不够稳定可靠，在使用hold-out时，一般要采用若干次随机划分、重复进行实验评估后取平均值作为hold-out的评估结果。常见做法是将大约2/3-4/5的样本用于训练，剩余样本用于测试

交叉验证法cross validation: 数据集D划分k个大小相似的互斥子集, 每个子集都尽可能保持数据分布的一致性，即从D中通过分层采样得到。通常把交叉验证法称为“k折交叉验证”k-fold cross validation, k最常用的取值是10. 与hold-out相似，将数据集D划分为k个子集同样存在多种划分方式，为减少因样本划分不同而引入的差别，k折交叉验证通常要随机使用不同的划分重复p次，最终结果取均值. 若令k = 数据集数目，则为留一法leave-one-out. 留一法中被实际评估的模型与期望评估的用D训练出的模型很相似。

自助法bootstrapping:

希望评估的是用D训练出的模型，而保留了一部分样本用于测试，因此实际评估的模型所使用的训练集比D小，这必然会引入一些因训练样本规模不同而导致的估计偏差。有没什么办法可以减少样本规模不同造成的影响？

自助法是一个比较好的解决方案，有放回抽样，则样本在m次采样中始终不被采到的概率是(1-1/m)m，取极限得到1/e = 0.368，即通过自助采样，初始数据集D中约有36.8%的样本未出现在采样数据集中。M个样本（含重复样本）作为训练集，未被采样的样本用于测试，亦称包外估计out-of-bag estimate

自助法在数据集较小，难以有效划分训练/测试集时很有用； 此外，自助法能从初始训练集中产生多个不同的训练集，这对集成学习等方法有很大的好处。然后自助法产生的数据集改变了初始数据集的分布，这会引入估计偏差。因此在初始数据量足够时，留出法和交叉验证法更常用一些。

调参与最终模型 （训练集 = 训练集 + 验证集）

在研究对比不同算法的泛化性能时，我们用测试集上的判别效果来估计模型在实际使用时的泛化能力，而把训练数据另外划分为训练集和验证集，基于验证集上的性能来进行模型选择和调参。模型选择完成后，学习算法和参数配置已选定，此时应该用数据集D重新训练模型。

性能度量

Error

Accuracy

precision(挑出的西瓜中有多少比例是好瓜？)

recall(所有好瓜中有多少比例被挑了出来？)

P-R曲线

预测结果是概率，将测试集的样例预测结果排序，顺序逐个把样本作为正例进行预测，每次计算出当前的precision and recall, 以precision为x轴，以recall为y轴作图

F1：precision and recall的调和平均

1/F1 = 1/2 \* (1/P + 1/R)

在一些应用中，对precision and recall的重视程度有所不同，例如在商品推荐中， 为了尽可能少打扰用户，更希望推荐内容确是用户感兴趣的，此时precision重重要；而在逃犯信息的信息检索系统中，更希望尽可能少漏掉逃犯，此时recall更重要，需要用加权F1

加权F1: 加权调和平均

1/Fb = 1/(1+b2)(1/P + b2/R)

(Note: 与算术平均(P+R)/2 和几何平均(P+R)1/2相比，调和平均更重视较小值

ROC:

预测结果是概率，将测试集的样例预测结果排序，顺序逐个把样本作为正例进行预测，每次计算出当前的真正率True Positive Rate = TP/(TP+FN), 假正率False Positive Rate = FP/(TN+FP), 以假正率为x轴，以真正率为y轴作图

基于实验评估方法和性能度量，如何比较学习器？ 见page 37 -

比想像中的困难，有两个因素：

因为比较的是泛化性能，然后实验评估方法获得的是测试集上的性能

测试集上的性能与测试集本身的选择有很大关系

需要基于统计假设检验hypothesis test，即若在测试集上观察到学习器A比B好，则A的泛化性能是否在统计意义上优于B，以及这个结论的把握有多大？

* 对单个学习器泛化性能

1. 假设检验

假设检验中的”假设“是对学习器泛化错误率分布的某种判断或猜想，例如泛化错误率是否不大于0.3, 即error <= 0.3

已知测试错误率，在一定置信度下，接受还是拒绝假设

估推出泛化错误率的分布

假定泛化错误率为error, 测试错误率为error\_t (意味着在m个测试样本中恰有error\_t \* m个被误分类)

误分类样本数的概率分布满足二项分布

P(error\_t; error) = Cmn \* errorerror\_t\*m \* (1-error)m-(error\_t\*m)

假设error <= e0, 求得在置信度1 – alpha的概率内所能观测到的最大错误率max\_error

若测试错误率error\_t<最大错误率max\_error，则假设error<= e0不能被拒绝，即以1-alpha的置信度认为，学习器的泛化错误率不大于e0

1. t检验(t-test)

若通过多次重复hold-out or cross-validation进行多次训练/测试，会得到多个测试错误率，此时用t检验(t-test)

* 对多个学习器泛化性能

1. 交叉验证t检验
2. McNemar检验

偏差与方差

对学习算法除了通过实验估计其泛化性能，还可以借助“偏差-方差分解”

偏差-方差分解试图对学习算法的期望泛化错误率进行拆解。

x: 测试样本

yD: 数据集中的标记

y: x的真实标记

f(x; D)：训练集D上学得模型f在x上的预测输出

f\_t = ED[f(x; D)]: 学习算法的期望预测

var(x) = ED[(f – f\_t)2]: 使用样本数相同的不同训练集

e\_n2 = ED[(yD – y )2]: 噪声产生的误差

bias = f\_t – y: 偏差 （期望输出与真实标记的差别）

算法的期望泛化误差：

ED[( f(x; D) – yD)2] = bias2 + var(x) + e\_n2

泛化误差可分解为偏差、方差与噪声之和

偏差度量了学习算法的期望预测与真实结果的偏离程度，即刻画了学习算法本身的拟合能力

方差度量了同样大小的训练集的变动所导致的学习性能的变化，即刻画了数据扰动所造成的影响

噪声则表达了在当前任务上任何学习算法所能达到的期望泛化误差的下界，即刻画了学习问题本身的难度。

偏差-方差分解说明：泛化性能是由学习算法的能力、数据的充分性以及学习任务本身的难度所共同决定的。

给定学习任务，为了取得好的泛化性能，则需使偏差较小，即能够充分拟合数据，并且使方差较小，即使得数据扰动产生的影响小

一般来说，偏差与方差是有冲突的。给定学习任务，若在训练不足时，学习器的拟合能力不强，训练数据的扰动不足以使学习器产生显著变化，此时偏差主导了泛化错误率； 随着训练程度的加深，学习器的拟合能力逐渐增强，训练数据发生的扰动渐渐能被学习器学到，方差逐渐主导了泛化错误率； 在训练程度充足后，学习器的拟合能力非常强，训练数据发生的轻微扰动都会导致学习器发生显著变化，若训练数据自身的、非全局的特性被学习器学到了，则将发生过拟合。

对离散属性，若属性值间存在序order关系，可通过连续化将其转化为连续值，例如值属性“身高”的取值{高，矮}可转化为{1.0, 0.0}，三值属性”高度“的取值{高，中，低}可转化为{1.0, 0.5, 0.0}; 若属性值间不存在序关系，假定有k个属性值，则通常转化为k维向量。例如属性”瓜类“的取值{西瓜，南瓜，黄瓜}可转化为(0, 0, 1), (0, 1, 0), (1, 0, 0)

多分类学习与多标记学习(multi-label learning)区别?

多分类学习中，每个样本仅属性一个类别，多标记学习中一个样本同时预测出多个类别标记，如一幅图像可同时标注为“蓝天”，“白云”，“羊群”，“自然场景”

多分类学习 见p63

基本策略：拆解法，将多分类任务拆为若干个二分类任务求解。关键是如何对多分类任务进行拆分，以及如何对多个分类器进行集成。

一对一：将N个类别两两配对 ，从而产生N(N-1)/2个二分类任务。在测试阶段，新样本同时提交给所有分类器，得到N(N-1)/2分类结果，最终结果通过投票产生。

一对多：每次将一个类的样例作为正例，所有其他类作为反例，从而训练N个分类器，在测试时若仅有一个分类器预测为正类，则对应的类别标记作为最终分类结果，若有多个分类器预测为正类，选择置信度最大的类别标记作为分类结果

多对多：每次将若干个类作为正类，若干个其他类作为反类，需要特殊的设计，不能随意选取，常见的有”纠错输出码”方法

类别不平衡问题 则p66

一个基本策略：再缩放rescaling

线性分类器，预测出的y值与一个阈值进行比较，通常y>0.5时判别为正例，否则为反例。阈值为0.5表明分类器认为真实正反例可能性相同，即分类器决策规则为若y / (1-y) > 1， 则预测为正例

若类别不平均，要强调类少的重要性，用观测几率代表真实几度，即只要分类器的预测几率高于观测几率就应判定为正例，即若y / ( 1-y) > m+ / m-, 则预测为正例

相当于对预测值进行调整

y’ / (1 – y’) = 1 / (1 – y) \* m+ / m-

rescaling不常用，因为“训练集是真实样本总体的无偏采样”这个假设往往并不成立，也就是说，我们未必能有效地基于训练集观测几率来推断出真实几率。

常采用做法：

欠采样：丢弃很多反倒，当然不能随机丢弃反倒，因为可能丢失一些重要信息；欠采样法的代表性算法EasyEnsemble[Liu et al., 2009]则是利用集成学习机制，将反倒划分为若干个集合供不同学习器使用，这样对每个学习器来看都进行了欠采样，但在分局来看却不会丢失重要信息。

过采样：不能简单地对初始正例样本进行重复采样，否则会招致严重的过拟合

阈值移动threshold-moving: 直接基于原始训练集进行学习，但在用训练好的分类器进行预测时，将rescaling嵌入到其决策过程中。

增量学习 incremental learning

即在接收到新样本后可对已学得的模型进行调整，而不用完全重新学习

增量学习可有效地降低每次接收到新样本后的训练时间开销，但多步增量学习后的模型会与基于全部数据训练而得的模型有较大差别

线性模型

线性模型有很好的可解释性

线性回归：y = WTx + b

Wopt = (XTX)-1XTy

现实任务中XTX往往不是满秩，例如在许多任务中我们会遇到大量的变量，其数目甚至超过样例数，此时可解出多个w, 它们都能使均方误差最小化，选择哪一个解作为输出，将由学习算法的归纳偏好决定，常见的做法是引入正则化regularization项。

（稀疏表示sparse representation：本质上对应L0范数的优化，这在通常条件下是NP难题，LASSO通过L1范数来近似L0范数，是求取稀疏解的重要技术）

对数线性回归：lny = WTx + b

广义线性模型：y = g-1(WTx + b) g(.)单调可微函数

Sigmoid函数：y = 1/(1+e-z), z = WTx + b

等价于 ln(y/(1-y)) = WTx + b 其中ln(y/(1-y))称为对数几率

优点：直接对分类可能性进行建模，无需事先假设数据分布，这样就避免了假设分布不准确所带来的问题，它不是仅预测出“类别”，而是可得到近似概率预测，这对许多需利用概率辅助决策的任务很有用。

线性判别分析LDA = Linear Discriminant Analysis

思想：给定训练集，设法将样例投影到一条直线上，使得同类样例的投影点尽可能接近，异类样例的投影点尽可能远。

y = WTx

Sw: 类内散度矩阵 within-class scatter matrix

Sb: 类间散度矩阵between-class scatter matrix

Wopt = Sw-1 (u0 – u1)

考虑到数值解的稳定性，在实践中通常是对Sw进行奇异值分解

决策树

决策过程中提出的每个判定问题都是对某个属性的“测试”，例如“色泽=？”； 每个测试的结果或是导出最终结论，或是导出进一步的判定问题，其考虑范围是在上次决策结果的限定范围之内

一棵决策树包含一个根结点、若干个内部结点和若干个叶结点；叶结点对应于决策结果，其他每个结点则对应于一个属性测试。

一般而言，随着划分过程不断进行，我们希望决策树的分支结点所包含的样本尽可能属于同一类别，即结点的“纯度”越来越高

度量样本集合纯度purity的指标：

信息熵information entropy: - 类k比例\*log(类k比例)的求和 k个类别

基尼指数Gini index: 反映了从数据集D中随机抽取两个样本，其类别标记不一致的概率

剪枝

决策树分支过多，会把训练样本学得过好，以致于把训练集自身的一些特点当作所有数据都具有的一般性质而导致过拟合，因此可通过主动去掉一些分支来降低过拟合的风险

预剪枝pre-pruning 和后剪枝post-pruning

预剪枝：在决策树生成过程中，对每个结点在划分前先进行估计，若当前结点的划分不能带来决策树泛化性能提升，则停止划分并将当前结点标记为叶结点

有些分支的当前划分虽不能提升泛化性能、甚至可能导致泛化性能暂时下降，但在其基础上进行的后续划分却有可能导致性能显著提高，而预剪枝基于贪心本质禁止这些分支展开，给预剪枝决策树带来了欠拟合的风险

后剪枝：先从训练集生成一棵完事的决策树，然后自底向上地对非叶结点进行考察，若将该结点对应的子树替换为叶结点能带来决策树泛化性能提升，则将该子树替换为叶结点

后剪枝决策树通过比预剪枝保留了更多的分支，一般情形下，后剪枝决策树的欠拟合风险很小，泛化性能往往优于预剪枝。

如何判断决策树泛化性能是否提升呢？从训练集中预留一部分用作验证集以进行性能评估

多变量决策树

决策树所形成的分类边界有一个明显的特点：轴平行，即它的分类边界由若干个与坐标轴平行的分段组成。

显然，分类边界的每一段都是与坐标轴平行的，这样的分类边界使得学习结果有较好的可解释性，因为每一段划分都直接对应了某个属性取值。但在学习任务的其实分类边界比较复杂时，必须使用很多段划分才能获得较好的近似。

如何使用斜的划分边界甚至更复杂划分的决策树？

采用多变量决策树，非叶结点不再是仅对某个属性，而是对属性的线性组合（对应斜划分边界）进行测试。

信息增益，增益率，基尼指数等准则对决策树的尺寸有较大影响，但对泛化性能影响有限，大约2%左右

剪枝方法和程度对决策树泛化性能影响相当显著，在数据带有噪声时通过剪枝甚至可将决策树的泛化性能提高25%