

机器学习cheatsheet!

目录

机器学习cheatsheet!

目录

决策树

决策树是怎么构建的？举个例子。比如，我们现在有五个特征，给定一个metric，怎么把这些数据分成两半？

也就是说，在分割时，选一个特征，再选一个阈值？

但是，如果连续特征粒度非常细怎么办？

熵的公式？

三种树的构造方法分别用了什么公式？

这个具体是怎么用的？比如我有一团数据，现在分成了两半。我怎么计算基尼不纯度？

ID3和C4.5也是一样吗？不用特别说明

多值属性偏好？

那ID3和C4.5与CART的区别是什么？

CART: 使用成本复杂度剪枝策略进行后剪枝。谈一下这个？

bagging & boosting

bagging的训练数据集可能重合吗？

解释一下adaboost和gradient boost？

那gradient boosting每个分类器是不是有weights呢

PCA

PCA的流程？

我记得有两种解释来着？

SVM

解释一下SVM？

1. 基本思想:
2. 支持向量:
3. 线性SVM:
4. 核技巧:
5. 软间隔:
6. 回归问题:

写出简便的推导公式

1. 函数间隔 (Functional Margin) :
2. 几何间隔 (Geometric Margin) :
3. 主问题 (Primal Problem) :
4. 软间隔 (Soft Margin) :

决策树

决策树是怎么构建的？举个例子。比如，我们现在有五个特征，给定一个metric，怎么把这些数据分成两半？

决策树是一种常用的分类和回归算法，它通过对特征进行递归分割来构建。构建决策树的基本思想是选择一个特征来分割数据，使得分割后的子集尽可能地"纯"。

我会简化地描述一下决策树的构建过程，并以一个简单的例子为你解释。

基本步骤：

- 1. **选择分割的特征：**这需要遍历所有的特征并计算分割后的指标值（例如：信息增益、基尼不纯度等）。选择指标值最优的特征来进行分割。
- 2. **分割数据：**根据选择的特征和其分割点（对于连续特征）将数据分为两个或多个子集。
- 3. **递归构建子树：**对于每一个子集，重复上述步骤，直到满足某个停止条件，例如：数据子集的大小、子集的纯度达到阈值、达到树的最大深度等。

简单的例子：

假设我们有以下数据：

+-----+-----+-----+		
Feature1	Feature2	Label
+-----+-----+-----+		
1	5	A
2	3	A
3	4	B
4	2	B
5	1	B
+-----+-----+-----+		

我们使用基尼不纯度作为评价指标。

- 1. **对于Feature1：**尝试所有可能的分割点（例如在1和2之间、2和3之间等），计算基尼不纯度。找到基尼不纯度最小的分割点。
- 2. **对于Feature2：**同样尝试所有的分割点，计算基尼不纯度。
- 3. 比较两个特征的最佳分割点，选择基尼不纯度更小的那个特征来分割数据。

例如，如果Feature2在3和4之间的分割点基尼不纯度最小，则选择Feature2和这个分割点来分割数据。

构建决策树的算法有很多，例如ID3、C4.5和CART。这些算法在选择分割特征和分割点的方法上有所不同，但基本思想是类似的。

这只是一个非常简化的描述和例子。实际上，构建决策树需要考虑很多细节和优化，特别是当数据和特征变得复杂时。

也就是说，在分割时，选一个特征，再选一个阈值？

是的，正是如此。简单来说，决策树的构建过程就是选择一个特征以及一个相应的阈值来进行分割。

对于离散特征：

- 你通常会为每个可能的特征值创建一个分支。例如，如果一个特征是颜色，可能的值为红、绿、蓝，那么你可能会基于这些值分为三个分支。

对于连续特征：

- 你会选择一个阈值来进行分割。例如，如果有一个特征表示身高，阈值可能是170cm，那么小于170cm的数据点会走一个分支，而大于或等于170cm的会走另一个分支。

选择哪个特征以及对于连续特征来说选择哪个阈值，是基于某种度量来确定的，这种度量可以是信息增益、基尼不纯度、均方误差等，取决于你使用的算法和问题的性质（分类还是回归）。

构建决策树的目标是使得每次分割后，得到的子集相对于之前更"纯"，也就是说子集中的数据点应当尽可能地属于同一类（对于分类问题）或者具有相似的值（对于回归问题）。

但是，如果连续特征粒度非常细怎么办？

对于粒度非常细的连续特征，直接对每一个可能的值都计算分割指标可能会非常低效，甚至有过拟合的风险。以下是一些常用的处理方法：

1. 离散化：

- 将连续特征分桶或分区。例如，将年龄从0-100分为10个区间：0-10, 10-20, ..., 90-100。
- 使用无监督的方法如分位数分桶或使用有监督的方法基于目标变量进行分桶。

2. 采样分割点：

- 而不是考虑所有可能的值，随机选择或均匀采样一些分割点来测试。

3. 使用决策树的变种：

- 一些决策树算法，如C4.5, 对连续特征的处理方法是选择两个连续值的中间值作为候选分割点。

4. 限制树的深度：

- 这可以避免对于非常精细粒度的数据产生过深的树，从而减轻过拟合。

5. 正则化：

- 例如，为决策树增加一个叶子节点的数量的惩罚，以避免构建过于复杂的树。

6. 后剪枝：

- 先完全构建决策树，然后删除或合并那些提供很小信息增益的部分。

以上方法可以独立或组合使用，具体取决于数据的特性和问题的性质。最重要的是，始终验证模型的泛化性能，确保不是在训练数据上过拟合。

熵的公式？

熵（Entropy）是一个衡量随机变量不确定性的指标。在信息论中，熵用来表示信息的量。

对于一个离散随机变量(X)，其可能的取值为(x_1, x_2, \dots, x_n)，对应的概率为($p(x_1), p(x_2), \dots, p(x_n)$)，其熵定义为：

$$H(X) = - \sum_{i=1}^n p(x_i) \log p(x_i)$$

这里的对数底通常取为2，这样得到的熵的单位是比特。但在一些情境下，也可以使用自然对数（底为（e）），这时熵的单位是纳特。

在决策树中，特别是ID3算法，熵用于评估数据集的不纯度。如果一个数据集中的所有元素都是同一类别，那么熵为0，表示该数据集完全纯净。而当数据集中的元素均匀分布于多个类别时，熵达到最大值，表示最大的不确定性。

三种树的构造方法分别用了什么公式？

1. ID3 (Iterative Dichotomiser 3):

- 使用的指标：信息增益 (Information Gain, IG)

- 定义：

$$IG(D, A) = H(D) - \sum \left(\frac{|D_v|}{|D|} \times H(D_v) \right)$$

2. C4.5 (successor of ID3):

- 使用的指标：信息增益率 (Gain Ratio)

- 定义：

$$Gain\ Ratio(D, A) = \frac{IG(D, A)}{H_A(D)}$$

其中，属性（A）的熵是：

$$H_A(D) = - \sum_{v \in Values(A)} \left(\frac{|D_v|}{|D|} \times \log \frac{|D_v|}{|D|} \right)$$

3. CART (Classification and Regression Trees):

- 对于分类问题，使用的指标是基尼不纯度 (Gini Impurity):

$$Gini(D) = 1 - \sum_{i=1}^m p_i^2$$

- 对于回归问题，使用的指标是均方误差 (Mean Squared Error, MSE):

$$MSE(D) = \frac{1}{|D|} \sum_{i=1}^{|D|} (y_i - \bar{y})^2$$

这个具体是怎么用的？比如我有一团数据，现在分成了两半。我怎么计算基尼不纯度？

对不起，我明白了。您希望完整的步骤和相关的公式都按照之前的格式转化。我为您重新组织：

1. 计算单个数据集的基尼不纯度：

设数据集（D）有（m）个类别，每个类别（i）在数据集中的占比是（p_i）。

$$Gini(D) = 1 - \sum_{i=1}^m p_i^2$$

2. 计算基于某个特征分割后的加权基尼不纯度：

假设你根据某个特征分割数据集（D）成了两部分：（D₁）和（D₂）。

（D₁）的权重是：

$$w_1 = \frac{|D_1|}{|D|}$$

（D₂）的权重是：

$$w_2 = \frac{|D_2|}{|D|}$$

加权基尼不纯度为：

$$Gini_{split}(D) = w_1 \times Gini(D_1) + w_2 \times Gini(D_2)$$

3. 选择基尼不纯度最小的特征进行分割：

对于每一个特征，计算该特征分割数据后的加权基尼不纯度。

选择加权基尼不纯度最小的特征进行分割。

ID3和C4.5也是一样吗？不用特别说明

不完全一样。ID3 和 C4.5 使用的是信息论的概念来进行决策，但它们的关注点和优化方法有所不同。

- **ID3** 使用**信息增益**来选择最佳的特征进行分割。信息增益基于整个数据集的熵和特征分割后子集的熵的差值来计算。

公式是：

$$IG(D, A) = H(D) - \sum \left(\frac{|D_v|}{|D|} \times H(D_v) \right)$$

其中， $H(D)$ 是数据集的熵，而 $H(D_v)$ 是属性 (A) 的某一取值 (v) 下的子集的熵。

- **C4.5** 则使用**信息增益率**，它是信息增益除以属性的固有信息值（即属性的熵）来进行优化的。这是为了克服 ID3 对多值属性有偏好的问题。

公式是：

$$Gain\ Ratio(D, A) = \frac{IG(D, A)}{H_A(D)}$$

其中， $H_A(D)$ 表示属性 (A) 的熵。

总的来说，ID3 和 C4.5 都是使用信息论的概念，但它们的计算方式和优化策略有所不同。CART 通常使用基尼不纯度或均方误差（对于回归任务）作为分割标准。

多值属性偏好？

多值属性偏好是指ID3算法在选择分割属性时对那些取值较多的属性有所偏好。这种偏好主要来源于ID3使用信息增益作为属性选择的准则。

考虑一个极端情况：如果一个属性对于每一个样本都有一个不同的取值，那么按照这个属性分割数据集，会使每一个子集只包含一个样本，从而子集的熵为0（因为它是纯的）。这会导致信息增益很大。然而，这样的分割并没有提供任何分类上的帮助，因为每一个子集只包含一个样本。

因此，对于那些取值较多的属性，即使它们并不一定在分类上很有区分力，ID3也会倾向于选择它们。

为了克服这一问题，C4.5算法引入了信息增益率。它除了考虑信息增益外，还考虑了属性的取值的数量（即属性的固有信息值或属性的熵）。这样，属性的取值数量也被考虑在内，减少了对多值属性的偏好。

简而言之，多值属性偏好是指算法过分地倾向于选择那些取值较多的属性进行分割，而C4.5通过引入信息增益率来解决这一问题。

那ID3和C4.5与CART的区别是什么？

ID3、C4.5和CART都是决策树构建算法，但它们之间存在若干主要区别：

1. 分割准则:

- **ID3**: 使用**信息增益**作为属性选择的准则。
- **C4.5**: 为了克服ID3对多值属性的偏好, C4.5引入了**信息增益率**作为分割准则。
- **CART**: 对于分类问题, 使用**基尼不纯度**作为分割准则; 对于回归问题, 使用**均方误差**。

2. 二叉树 vs. 多叉树:

- **ID3 和 C4.5**: 可以创建多叉树, 即一个节点可以有多个子节点。
- **CART**: 始终生成二叉树。

3. 处理连续属性:

- **ID3**: 原始ID3不直接处理连续属性。
- **C4.5**: 可以处理连续属性, 通过为连续属性设定阈值来将其转化为二元属性。
- **CART**: 同样可以处理连续属性, 也是通过设定阈值将其转化为二元属性。

4. 剪枝策略:

- **ID3**: 原始ID3没有剪枝策略。
- **C4.5**: 有后剪枝策略, 它首先构建完整的决策树, 然后剪去那些对分类贡献不大的部分。
- **CART**: 使用成本复杂度剪枝策略进行后剪枝。

5. 处理缺失值:

- **ID3**: 原始ID3不直接处理缺失值。
- **C4.5**: 提供了对缺失值处理的策略。
- **CART**: 也有处理缺失值的机制。

6. 算法输出:

- **ID3 和 C4.5**: 仅生成分类树。
- **CART**: 可以生成分类树和回归树。

以上是ID3、C4.5和CART之间的主要区别。需要注意的是, 决策树算法都有其各自的优缺点, 并根据具体应用场景选择合适的算法。

CART: 使用成本复杂度剪枝策略进行后剪枝。谈一下这个?

成本复杂度剪枝 (Cost Complexity Pruning) 是CART (Classification and Regression Trees) 中用于防止过拟合的一种策略。它试图平衡模型的复杂度和模型的拟合程度。

在构建决策树的过程中, 我们可能会得到一个过于复杂、对训练数据拟合得非常好的树, 但这样的树可能会在新的、未见过的数据上表现得不太好。这就是所谓的过拟合问题。

成本复杂度剪枝是一种后剪枝策略, 意思是先构建完整的决策树, 然后再进行剪枝, 与此相对的策略是预剪枝, 即在构建树的过程中就决定是否继续分裂节点。

以下是成本复杂度剪枝的基本思想:

1. **定义复杂度**：对于每个子树（从根到叶的任何子集），我们都可以计算其成本复杂度，它是树的误差率和树的大小的加权和。公式是：
$$C(T) = Err(T) + \alpha|T|$$
其中， $Err(T)$ 是树 (T) 在训练数据上的误差， $(|T|)$ 是树 (T) 的叶节点数量，而 (α) 是一个控制模型复杂度的参数。
2. **确定最优子树**：为了找到最优的子树，我们会尝试移除每一个非叶节点及其子节点，并计算新子树的复杂度。如果移除该节点后，子树的复杂度降低了，我们就真的移除它。
3. **调整复杂度参数**：通过调整 (α) 的值，我们可以得到一系列的最优子树。通常，我们会使用交叉验证的方法，通过在验证集上测试不同的 (α) 值来选择最佳的 (α) 。

最终，成本复杂度剪枝的目标是找到一个既不太复杂（不容易过拟合）又具有合理预测性能的模型。

bagging & boosting

bagging的训练数据集可能重合吗？

是的，Bagging (Bootstrap Aggregating) 中的训练数据集可能会重合。在Bagging的过程中，每一个基模型（例如决策树）都是使用一个自助采样 (bootstrap sampling) 得到的训练集进行训练的。自助采样是一个有放回的随机采样方法，这意味着在一个新的采样集合中某些原始数据可能会被多次选中，而另一些原始数据可能一次都不会被选中。

例如，假设我们有一个包含5个数据点的原始训练集：[A, B, C, D, E]。一个可能的自助采样结果是：[A, B, B, C, E]，在这个采样中，数据点B被选中了两次，而数据点D没有被选中。

这种采样方式确保了每一个基模型都在稍微不同的训练数据上进行训练，从而增加了模型的多样性。这有助于降低模型的方差，并且在整体上改善了集成模型的泛化能力。

解释一下adaboost和gradient boost?

当我们讨论集成学习方法时，AdaBoost 和 Gradient Boosting 是两种非常流行的技术。它们都旨在通过组合多个弱学习器来构建一个强学习器，但它们的方法和动机有所不同。

1. AdaBoost (Adaptive Boosting):

- **基本思想**: AdaBoost 的工作原理是，对于每一次迭代，它都会给那些之前被错误分类的样本分配更大的权重，这样下一个弱学习器就会重点关注这些错误分类的样本。
- **权重调整**: 如果某个样本被正确分类，则其权重在下次迭代中会被降低；如果被错误分类，则其权重会增加。
- **输出组合**: 最后，所有的弱学习器都会被线性组合起来，每个学习器的权重是基于其在训练数据上的表现确定的。
- **常用的弱学习器**: 决策树（特别是深度为1的决策树，也称为决策树桩）。

2. Gradient Boosting:

- **基本思想**: Gradient Boosting 的核心思想是，每一个新的模型都是在前面所有模型的残差（或错误）上进行训练的。换句话说，每一个新的模型都试图纠正前面模型的错误。

对于每一个迭代步，我们都会计算基模型的预测值与真实值之间的差异（或残差），这就是所

- **残差:** 在每一次迭代中，我们都会计算前面模型的预测值与真实值之间的差异（或者说，梯度）。这就是所谓的“残差”，新的模型会尝试预测这个残差。
- **输出组合:** 就像 AdaBoost 一样，所有的模型都会被组合起来。但在 Gradient Boosting 中，这是通过加法的方式进行的，也就是说，每个模型的输出都会被加到前面模型的输出上，从而得到最终的预测值。
- **常用的弱学习器:** 也是决策树，但可以是更深的树。

总之，虽然 AdaBoost 和 Gradient Boosting 都是通过组合多个弱学习器来提高性能的，但它们关注的错误和调整策略是不同的。AdaBoost 主要关注之前被错误分类的样本，而 Gradient Boosting 则是关注前面模型的残差或错误。

那gradient boosting每个分类器是不是有weights呢

是的，Gradient Boosting 中的每个分类器确实有权重，但这里的权重与 AdaBoost 中的权重有所不同。

在 Gradient Boosting 中，每个新模型都是在之前模型的残差上进行训练的。然后，这个新模型的预测将乘以一个权重（通常称为学习率或步长）并加到整体模型的预测中。学习率是一个预定义的参数，它决定了我们在每一步中应该调整多少以纠正之前模型的错误。通过这种方式，每个模型的预测都会被逐步加入到最终结果中。

较小的学习率意味着每个模型对最终预测的贡献较小，可能需要更多的模型来获得良好的性能；而较大的学习率则可能导致模型过于强烈地纠正错误，从而导致过拟合。

所以，尽管 AdaBoost 为每个模型分配一个基于其性能的权重，Gradient Boosting 为每个模型分配一个固定的权重（学习率）。然而，这两者都利用权重机制来合并模型的预测。

PCA

PCA的流程？

主成分分析（Principal Component Analysis, PCA）是一种降维技术，用于将多维数据集转化为低维数据，同时保留尽可能多的原始数据的方差。下面是PCA的基本步骤：

1. **中心化数据:**
 - 计算每个特征的平均值。
 - 将原始数据的每个特征减去其平均值，这样新的平均值为0。
2. **计算协方差矩阵:**
 - 用中心化后的数据计算协方差矩阵。
3. **计算协方差矩阵的特征值和特征向量:**
 - 使用线性代数的方法，如特征分解或奇异值分解(SVD)。
4. **排序特征值，并选择最大的k个特征值:**
 - 这些特征值对应的特征向量将是我们要保留的主成分。
 - k 是一个用户定义的参数，代表我们希望降低到的维数。
5. **构造投影矩阵:**
 - 从上一步中选出的k个特征向量构成这个矩阵。

◦ 将原始数据转换到新的空间

b. 将原始数据转换到新的空间：

- 通过将中心化后的原始数据与投影矩阵相乘，将数据投影到选定的主成分上。

完成这些步骤后，你将得到一个降维后的数据集，其维度为k。这个新的数据集在k个主成分上尽可能地保留了原始数据的方差。

我记得有两种解释来着？

哦，我明白了。你提到的是PCA的两种直观解释或直观背景：

1. 最大方差解释 (Variance Maximization)：

- 在这种解释中，PCA试图找到一个方向（在数学上，这被称为一个“向量”），当数据投影到这个方向上时，它的方差是最大的。第一个主成分就是在这个方向上。为什么要这样做？因为方差捕获了数据的“扩散”程度，当方差最大时，意味着在这个方向上数据分布得最广，从而保留了大部分信息。
- 第二个主成分选择的方向是与第一个方向正交（垂直）的，并且在这个新方向上，方差是最大的。以此类推，为后续的每一个主成分选择一个方向。

2. 最小均方误差解释 (Minimum Mean Squared Error)：

- 在这种解释中，PCA可以被看作是一个尽量减少原始数据与其低维表示之间重建误差的方法。换句话说，如果你只能使用较少的主成分来近似描述原始数据，你会选择哪些主成分，以便使这种近似的误差最小化？PCA确保选取的主成分是最佳的，从而最小化了这种重建误差。
- 这也意味着，如果你使用所有的主成分，那么重建的数据将与原始数据完全一致（假设没有进行其他处理）。

这两种解释从不同的角度描述了PCA的目的，但它们都与同一种数学结构有关。这两种直观理解帮助我们理解PCA的目的和作用。

SVM

解释一下SVM？

支持向量机 (Support Vector Machine, SVM) 是一种监督学习算法，主要用于分类和回归。最初被设计为二分类器，但也可以用于多分类问题。以下是关于SVM的基本介绍：

1. 基本思想：

SVM的核心思想是找到一个决策边界（在二维空间中是一条线，在三维中是一个平面，更高维度是一个超平面），使得两个类别的数据点之间的间隔（即“margin”）最大化。这个决策边界被称为“最大间隔超平面”。

2. 支持向量:

决策边界的位置由那些离边界最近的数据点决定，这些数据点被称为“支持向量”。即使我们移除非支持向量的其他数据点，决策边界的位置也不会改变。

3. 线性SVM:

在最简单的情况下，数据是线性可分的，这意味着可以通过一个直线（或超平面）完美地分隔两个类别。这种情况下的SVM被称为“线性SVM”。

4. 核技巧:

在很多实际应用中，数据并不是线性可分的。为了处理这种情况，SVM引入了所谓的“核技巧”（Kernel trick）。核函数可以将原始特征空间映射到一个更高维度的空间，使得数据在新的空间中变得线性可分。常见的核函数有：线性核、多项式核、径向基函数（Radial Basis Function, RBF）核和sigmoid核。

5. 软间隔:

在某些情况下，即使使用核技巧，数据也可能不是完全线性可分的，或者存在一些异常点。为了处理这种情况，SVM引入了“软间隔”概念，允许某些数据点位于决策边界的错误一侧。引入了一个“惩罚参数” C ，它决定了误分类的“成本”。当 C 很大时，误分类的成本也很大，SVM会尽量避免误分类；当 C 较小时，SVM可能会允许一些误分类，以得到更简单的决策边界。

6. 回归问题:

虽然SVM最初是为分类设计的，但它也可以用于回归问题，称为支持向量回归（Support Vector Regression, SVR）。在SVR中，目标是找到一个函数，使得大部分数据点的预测误差都在一个给定的范围内。

SVM在许多实际应用中都表现得很好，特别是在中小型数据集和高维数据上。但与大多数机器学习方法一样，SVM的性能也取决于特定任务、数据特性和参数设置。

写出简便的推导公式

1. 函数间隔 (Functional Margin) :

给定一个训练样本 (x_i, y_i) ，函数间隔为：

$$\hat{\gamma}_i = y_i(w \cdot x_i + b)$$

2. 几何间隔 (Geometric Margin) :

几何间隔是：

$$\gamma_i = y_i \left(\frac{w}{\|w\|} \cdot x_i + \frac{b}{\|w\|} \right)$$

3. 主问题 (Primal Problem) :

SVM的目标是最大化间隔, 对于线性可分的情况, 问题可以定义为:

Maximize γ

subject to: $y_i(w \cdot x_i + b) \geq \gamma \quad \forall i$

$\|w\| = 1$

通过转化, 我们得到:

Minimize $\frac{1}{2} \|w\|^2$

subject to: $y_i(w \cdot x_i + b) \geq 1 \quad \forall i$

4. 软间隔 (Soft Margin) :

引入松弛变量 (ξ_i):

Minimize $\frac{1}{2} \|w\|^2 + C \sum_i \xi_i$

subject to: $y_i(w \cdot x_i + b) \geq 1 - \xi_i \quad \forall i$

其中, (C) 是一个超参数。