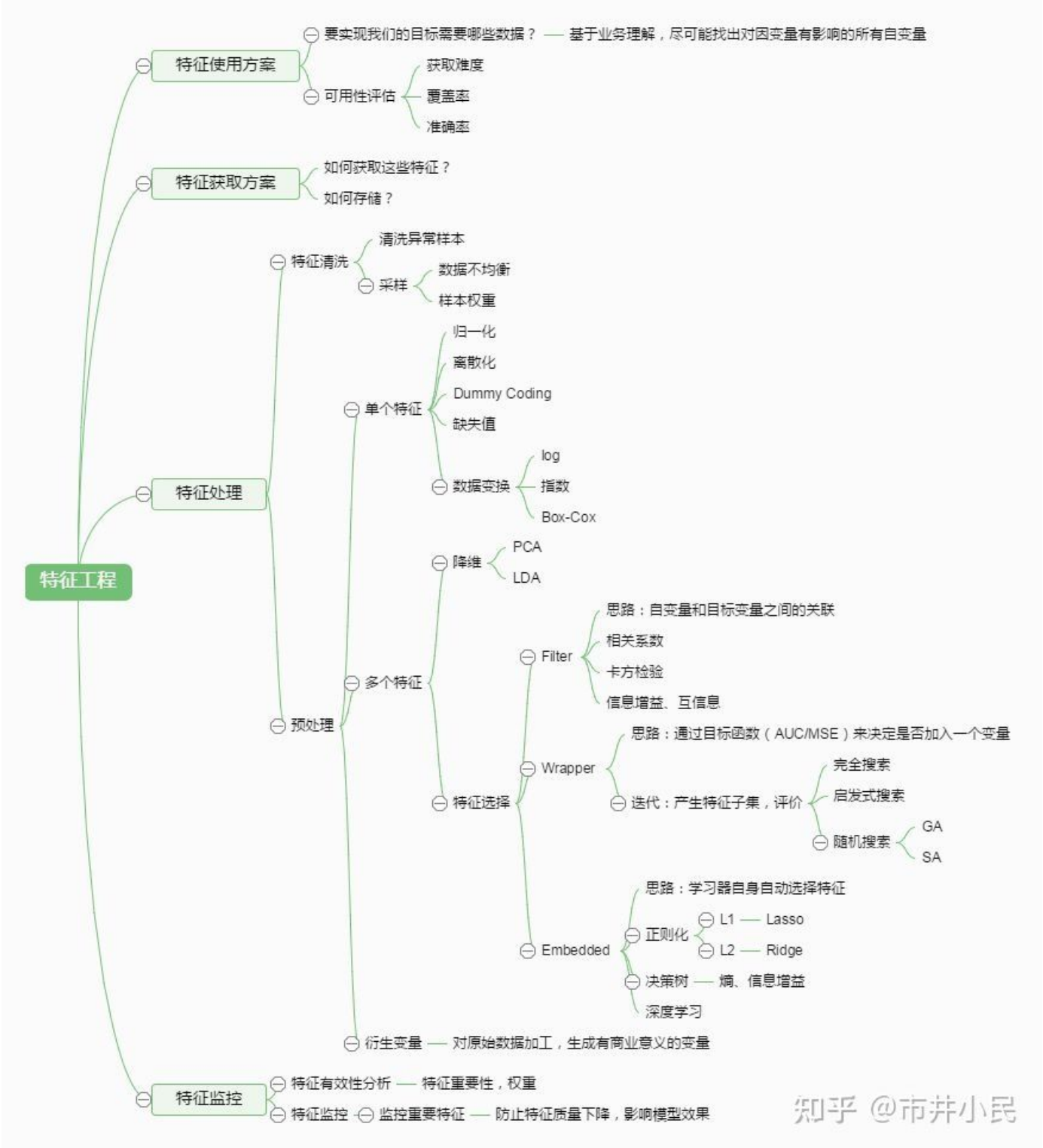
机器学习与深度学习面试系列一（特征工程）



特征工程整体示意图

数据清洗的基本思路?

结合业务情况进行数据的过滤，例如去除crawler抓取，spam，作弊等数据。

异常点检测，采用异常点检测算法对样本进行分析，常用的异常点检测算法包括：

1. 偏差检测，例如聚类，最近邻等。
2. 基于统计的异常点检测算法 例如极差，四分位数间距，均差，标准差等。
3. 基于距离的异常点检测算法，主要通过距离方法来检测异常点，将数据集中与大多数点之间距离大于某个阈值的点视为异常点，主要使用的距离度量方法有绝对距离（曼哈顿距离）、欧氏距离和马氏距离等方法。
4. 基于密度的异常点检测算法，考察当前点周围密度，可以发现局部异常点，例如LOF算法

数据特征有哪些类型?

大致可以分为结构化特征和非结构化特征。

结构化特征，通俗的说就是每一行代表一个数据。分为数值型和类别型两种基本类型。

非结构化特征包括文本、图片、视频、音频等，无法用一行来表示。往往每个数据的长度也不同。

什么是特征的无量纲化(归一化)

归一化是针对数值型特征来说的，主要有两种方法:

标准化（零均值归一化）： $x_{norm} = \frac{x - \mu}{\sigma}$ 。假设原始特征的均值为 μ 、标准差为 σ ，它会将原始数据映射到均值为 0、标准差为1的分布上。（sklearn.preprocessing.StandardScaler）

归一化（线性函数归一化）： $x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}}$ 。它对原始数据进行线性变换，使 结果映射到[0, 1]的范围，实现对原始数据的等比缩放。（sklearn.preprocessing.MinMaxScaler）

为什么要进行特征的无量纲化(归一化)

1. 提高模型的精度，确保数据的所有特征都处于同一量纲，提高特征的可比性
2. 加快收敛速度

哪些场景需要使用特征的无量纲化(归一化)

1. 基于距离计算的模型，如：KNN
2. 通过梯度下降求解的模型：线性回归、逻辑回归、支持向量机、神经网络等。

决策树模型一般不需要进行归一化，因为树模型寻找最优点是通过寻找最优分裂点完成的，数值缩放，不影响分裂点位置，从而不需要数据归一化。

缺失值有哪些处理手段

1. 不处理。 确实比例较小，且选择的模型能够自动处理缺失数据（比如Xgboost）
2. 删除。缺失特征对预测影响较小或特征值存在大面积缺失
3. 填充（补齐）。数学统计值填充（中位数、平均值、众数等），算法预测填充（插值法、线性回归、最小K近邻等）

类别型特征（Categorical Feature）应当怎么处理?

1. 序号编码(Ordinal Encoding)。序号编码通常用于处理类别间具有大小关系的数据。例如成绩，可以分为 低、中、高三档，并且存在“高>中>低”的排序关系。序号编码会按照大小关系对 类别型特征赋予一个数值ID，例如高表示为3、中表示为2、低表示为1，转换后依 然保留了大小关系。
2. 独热编码(One-hot Encoding)。独热编码通常用于处理类别间不具有大小关系的特征。例如血型，一共有4个 取值(A型血、B型血、AB型血、O型血)，独热编码会把血型变成一个4维稀疏 向量，A型血表示为(1, 0, 0, 0)，B型血表示为(0, 1, 0, 0)，AB型表示为(0, 0, 1, 0)，O型血表示为(0, 0, 0, 1)。 （sklearn.preprocessing.OneHotEncoder）

如何将数值型特征转为类别型（离散化）？

最简单的二值化：设定一个阈值，大于阈值的赋值为1，小于等于阈值的赋值为0。（sklearn.preprocessing.Binarizer）

常用的离散化方法包括等值划分和等量划分。等值划分是将特征按照值域进行均分，每一段内的取值等同处理。例如某个特征的取值范围为[0，10]，我们可以将其划分为10段，[0，1)，[1，2)，...，[9，10)。等量划分是根据样本总数进行均分，每段等量个样本划分为1段。例如距离特征，取值范围［0，3000000］，现在需要切分成10段，如果按照等比例划分的话，会发现绝大部分样本都在第1段中。使用等量划分就会避免这种问题，最终可能的切分是[0，100)，[100，300)，[300，500)，..，[10000，3000000]，前面的区间划分比较密，后面的比较稀疏。[1]

什么是组合特征？如何处理高维组合特征?

为了提高复杂关系的拟合能力，在特征工程中经常会把一阶离散特征两两组合，构成高阶组合特征。

例：若用户的数量为 `m` 、物品的数量为 `n` ，那么需要学习的参数的规模为 `m*n` 。在互联网环境下，用户数量和物品数量都可以达到千万量级，几乎无法学习 `m*n` 规模的参数。在这种情况下，一种行之有效的方法是将用户和物品分别用k维的低维向量表示 （`k<<m，k<<n`） 。这样就可以是的模型的参数量变为（`m*k + n*k`）。 类似于**矩阵分解**。[2]

怎样有效地找到组合特征?

可以基于决策树的特征组合寻找方法。根据原始输入和标签构造出决策树（如梯度提升决策树），然后每一条从根节点到叶节点的路径都可以看成一种特征组合的方式。

为什么要做特征选择

特征选择的目标是寻找最优特征子集。

1. 特征冗余。特征选择能剔除不相关(irrelevant)或冗余(redundant)的特征，从而达到减少特征个数，提高模型精确度，减少运行时间的目的。
2. 噪声。部分特征无用或者有副作用
3. 可以将特征选择理解为一种降维方法，它可以降低模型复杂度，从而减轻过拟合。

特征选择有哪些方法?

特征选择的一般过程：

1. 生成子集：搜索特征子集，为评价函数提供特征子集
2. 评价函数：评价特征子集的好坏
3. 停止准则：与评价函数相关，一般是阈值，评价函数达到一定标准后就可停止搜索
4. 验证过程：在验证数据集上验证选出来的特征子集的有效性

通常来说，从两个方面考虑来选择特征：
特征是否发散：如果一个特征不发散，例如方差接近于0，也就是说样本在这个特征上基本上没有差异，这个特征对于样本的区分并没有什么用。
特征与目标的相关性：这点比较显见，与目标相关性高的特征，应当优先选择。[3]

根据特征选择的形式，可分为三大类：

- Filter(过滤法)：按照发散性（方差法）或 相关性（其他方法）对各个特征进行评分，设定阈值或者待选择特征的个数进行筛选
- Wrapper(包装法)：根据目标函数（往往是预测效果评分），每次选择若干特征，或者排除若干特征
- Embedded(嵌入法)：先使用某些机器学习的模型进行训练，得到各个特征的权值系数，根据系数从大到小选择特征（类似于Filter，只不过系数是通过训练得来的）[4]

特征选择-Filter法?

1. 方差选择法。使用方差选择法，先要计算各个特征的方差，然后根据阈值，选择方差大于阈值的特征。（sklearn.feature_selection.VarianceThreshold）
2. Pearson相关系数法。皮尔森相关系数是一种最简单的，能帮助理解特征和响应变量之间关系的方法，衡量的是变量之间的线性相关性，结果的取值区间为[-1,1]， -1 表示完全的负相关(这个变量下降，那个就会上升)， +1 表示完全的正相关， 0 表示没有线性相关性。（sklearn.feature_selection.SelectKBest）。
3. 卡方检验法。
4. 互信息法。
5. 距离相关系数法。

特征选择-Wrapper法?

基于hold-out方法，对于每一个待选的特征子集，都在训练集上训练一遍模型，然后在测试集上根据误差大小选择出特征子集。需要先选定特定算法，通常选用普遍效果较好的算法， 例如Random Forest， SVM， kNN等等。西瓜书上说包装法应该欲训练什么算法，就选择该算法进行评估。随着学习器（评估器）的改变，最佳特征组合可能会改变。由于在特征选择过程中需多次训练学习器，因此Wrapper特征选择的计算开销通常比Filter特征边择大得多.

1. 前向搜索。每次增量地从剩余未选中的特征选出一个加入特征集中，待达到阈值或者 n 时，从所有的待选特征集中选出错误率最小的。
2. 后向搜索。既然有增量加，那么也会有增量减，后者称为后向搜索。先将待选特征集设置为全量的 {1,2,...,n} ，然后每次删除一个特征，并评价，直到达到阈值或者为空，然后选择最佳的特征集。
3. 递归特征消除法。递归消除特征法使用一个基模型来进行多轮训练，每轮训练后通过学习器返回的 coef_ 或者feature_importances_ 消除若干权重较低的特征，再基于新的特征集进行下一轮训练。

特征选择-Embedded法?

1. 基于惩罚项的特征选择法 (L1正则化天然具有特征选择作用)
2. 基于树模型的特征选择法（树模型中GBDT也可用来作为基模型进行特征选择）
3. 深度学习

数据增强有哪些方法?

以图像类数据为例：

1. 一定程度上进行随机的旋转（通常为180度、90度，否则有尺寸匹配问题）、平移、缩放、裁剪、填充、翻转等

2. 噪声扰动，如椒盐噪声、高斯噪声等
3. 颜色、清晰度、锐度等变换
4. 使用生成模型合成一些新样本，如GAN
5. 迁移学习。借用一个在大规模数据集上预训练好的模型，并在对目标任务的小数据集上进行微调。

参考

1. ^ https://zhuanlan.zhihu.com/p/56316302
2. ^ https://zhuanlan.zhihu.com/p/111195026
3. ^ https://zhuanlan.zhihu.com/p/337469074
4. ^ https://zhuanlan.zhihu.com/p/74198735