数据降维和模型建立方法

孔德瑞

2021 年 10 月 14 日

所谓的数据降维，也就是指采用某种特定的映射方法, 将原高维空间中的数据点映射到低维度的空间中。通过数据降维，我们可以使得数据便于计算和可视化，同时降维可以提取数据内部的本质结构，减少冗余信息和噪声信息造成的误差，提高应用中的精度。因此，数据降维的关键在于寻找合适的映射方法，使得数据维度降低的同时，能最大程度的保有原数据的特征，同时减少冗余信息和噪声信息。

1 主成分分析 PCA

主成分分析是一种常用的线性数据降维方法，最早由Pearson (1901) 提出，并由Hotelling (1933) 拓展。主成分分析中以方差衡量数据的信息，因此主成分分析的目标是使得降维后样本的方差尽可能大以尽可能保留原数据的特征。

定义数据集 $\mathbf{X} = \{x_1, x_2, \ldots, x_K\}_{N\times K}$，其中 K 表示数据维度。若我们要将数据转换为 L 维数据，则记降维后的数据集 $\mathbf{Y} = \mathbf{XW}$，其中 $\mathbf{W}_{K\times L}$ 代表线性转换矩阵。此时，降维后的数据方差最大的最优化问题为：

$$\begin{aligned}
\max_{\mathbf{W}} \quad var(\mathbf{Y}) &= \frac{1}{n-1} tr(\mathbf{Y}^T\mathbf{Y}) && (1)\\
&= \frac{1}{n-1} tr(\mathbf{W}^T\mathbf{X}^T\mathbf{XW}) && (2)\\
&= tr\left(\mathbf{W}^T \frac{1}{n-1}\mathbf{X}^T\mathbf{XW}\right) && (3)\\
&= tr\left(\mathbf{W}^T\Sigma\mathbf{W}\right) && (4)\\
s.t. \quad \mathbf{w}_i^T\mathbf{w}_i &= 1 \forall i \in \{1, 2, \ldots, L\} && (5)
\end{aligned}$$

其中 $\Sigma = \frac{1}{n-1}\mathbf{X}^T\mathbf{X}$ 代表原始数据集的协方差矩阵。解最优化问题，构造拉格朗日函数：

$$\mathcal{L}(\mathbf{W}, \lambda) = tr\left(\mathbf{W}^T\Sigma\mathbf{W}\right) - \sum_{L}^{i=1} \lambda_i(w_i^T w_i - 1) \tag{6}$$

其一阶条件为：

$$\Sigma \mathbf{w}_i = \lambda_i \mathbf{w}_i \tag{7}$$

则可知，$\mathbf{w}_i$ 和 λ_i 为 Σ 的特征向量及其对应的特征值。同时，因为 $\Sigma \mathbf{w}_i = \lambda_i \mathbf{w}_i$，可知 $tr\left(\mathbf{W}^T \Sigma \mathbf{W}\right) = \sum_{i=1}^{L} \lambda_i$，因此我们只需求 Σ 的特征值和特征向量，取其中最大的 L 个特征值及其对应的特征向量组成 $\mathbf{W}$，即可得到最优化的解。

2 随机森林 Random Forest

随机森林是通过集成学习的思想将多棵树集成的一种算法，由Breiman (2001) 提出。它的基本单元是决策树，而它的本质属于机器学习的一大分支——集成学习（Ensemble Learning）方法。

决策树是一个树结构（可以是二叉树或非二叉树），其每个非叶节点表示一个特征属性上的测试，每个分支代表这个特征属性在某个值域上的输出，而每个叶节点存放一个类别。使用决策树进行决策的过程就是从根节点开始，测试待分类项中相应的特征属性，并按照其值选择输出分支，直到到达叶子节点，将叶子节点存放的类别作为决策结果。

随机森林，是用随机的方式建立一个森林，森林里面有很多的决策树组成，随机森林的每一棵决策树之间是没有关联的。在得到森林之后，当有一个新的输入样本进入的时候，就让森林中的每一棵决策树分别进行判断，将决策树中得出的结果中最多的结果作为最终的输出结果。

随机森林的构造过程如下：

1. 假如有 N 个样本，则有放回的随机选择 N 个样本 (每次随机选择一个样本，然后返回继续选择)。这选择好了的 N 个样本用来训练一个决策树，作为决策树根节点处的样本。
2. 当每个样本有 M 个属性时，在决策树的每个节点需要分裂时，随机从这 M 个属性中选取出 m 个属性，满足条件 $m < M$。然后从这 m 个属性中采用某种策略来选择一个属性作为该节点的分裂属性。
3. 决策树形成过程中每个节点都要按照步骤 2 来分裂，直至无法分裂为止。则易知，如果下一次该节点选出来的那一个属性是刚刚其父节点分裂时用过的属性，则该节点已经达到了叶子节点，无须继续分裂了。（与一般决策树算法相比，随机森林算法生成决策树并没有剪枝过程。）
4. 按照以上三个步骤建立大量的决策树，这样就能构成随机森林。

由于我们要解决的是回归问题，因此生成的决策树选取为 CART 回归树。CART 回归树预测回归连续型数据。CART 假设决策树是二叉树，内部结点特征的取值只有“是”和“否”，左分支

是取值为“是”的分支，有分支则相反。这样的决策树等价于递归地二分每个特征。。假设 X 与 Y 分别是输入和输出变量，并且 Y 是连续变量。一般 CART 回归树的生成过程如下：

1. 选择最优切分变量 j 与切分点 s，即求解：

$$\min_{j,s}[\min_{c_1}\sum_{x_i\in R_1(j,s)}(y_i-c_1)^2]+\min_{c_1}\sum_{x_i\in R_2(j,s)}(y_i-c_2)^2] \tag{8}$$

   其中 R_m 是被划分的输入空间，其对应的输出值为 c_m。遍历变量 j，对固定的切分变量 j 扫描切分点 s，选择使上式取得最小值（平方误差最小）的 (j,s)。

2. 用选定的对 (j,s) 划分区域并决定相应的输出值：

$$R_1(j,s)=\{x|x^{(j)}\le s\},\quad R_2(j,s)=\{x|x^{(j)}>s\} \tag{9}$$

$$\hat{c_m}=\frac{1}{N_m}\sum_{x_i\in R_m(j,s)}y_i,\quad x\in R_m,m=1,2 \tag{10}$$

3. 继续对两个子区域调用步骤 1，直至满足停止条件

4. 将输入空间划分为 M 个区域 $R_1,R_2,\ldots,R_m$ 生成决策树：

$$f(x)=\sum_{m=1}^{M}\hat{c_m}I(x\in R) \tag{11}$$

5. 当输入空间划分确定时，可以用平方误差 $\sum_{x_i\in R_m}(y_i-f(x_i))^2$ 来表示回归树对于训练数据的预测方法，用平方误差最小的准则求解每个单元上的最优输出值。

应用随机森林进行数据降维，或者说选取数据中更“重要”的特征，主要参考的是袋外错误率（out-of-bag error）和 RF 特征重要性。所谓袋外错误率即对于随机森林中某一棵树而言，没有参与其形成的样本（袋外样本 oob）中预测错误的频率。袋外错误率是随机森林泛化误差的一个无偏估计。

在计算袋外错误率后便可以计算 RF 特征重要性，其原理为：给某个特征随机的加入噪声后，如果袋外错误率增大，说明这个特征对样本分类的结果影响比较大，说明重要程度比较高。具体计算方法如下：对于某特征 j，对于每一个决策树可计算其袋外错误率 $ooberror_0$，随机对特征 j 加入噪声干扰后可计算袋外错误率 $ooberror_1$，则对该特征：

$$\text{RF 重要性}=\frac{\sum(ooberror_1-ooberror_0)}{\text{随机森林中决策树的个数}} \tag{12}$$

3 梯度提升决策树 Gradient Boosting Decision Tree

GBDT(Gradient Boosting Decision Tree) , 又叫 MART（Multiple Additive Regression Tree)，是一种迭代的决策树算法，该算法由多棵决策树组成，所有树的结论累加起来做最终答案。

GBDT 算法可以看成是由 K 棵树组成的加法模型：

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), f_k \in F \tag{13}$$

从上面公式可以看出，GBDT 预测的结果就是多个串联的树的预测结果的和。首先定义目标函数为：$Obj = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$ Ω 表示决策树的复杂度（树的复杂度可以考虑树的节点数量、树的深度或者叶子节点所对应的分数的 L2 范数等），作为参数正则化项；前一部分为损失函数。

学习加法模型，可以用前向分布算法（Forward Stagewise Algorithm）。因为学习的是加法模型，如果能够从前往后，每一步只学习一个基函数及其系数（结构），逐步逼近优化目标函数，那么就可以简化复杂度。具体来说，我们从一个常量预测开始，每次学习一个新的函数，过程如下：

$$\hat{y}_i^0 = 0 \tag{14}$$

$$\hat{y}_i^1 = f_1(x_i) = \hat{y}_i^0 + f_1(x_i) \tag{15}$$

$$\hat{y}_i^1 = f_2(x_i) = \hat{y}_i^1 + f_2(x_i) \tag{16}$$

$$\dots \tag{17}$$

$$\hat{y}_i^t = \sum_{k=1}^{t} = \hat{y}_i^{t-1} + f_t(x_i) \tag{18}$$

在第 t 步，模型对于 x_i。这个时候目标函数可以写为：

$$Obj^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^t) + \sum_{i=1}^{t} \Omega(f_i) \tag{19}$$

$$= \sum_{i=1}^{n} l(y_i, \hat{y}_i^{t-1} + f_t(x_i)) + \Omega(f_t) + constant \tag{20}$$

$$= \sum_{i=1}^{n} [l(y_i, \hat{y}_i^{t-1}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t) + constant \tag{21}$$

其中式21应用了泰勒公式的二阶展开式；g_i 为损失函数 $l(x)$ 的一阶导数：$g_i = \frac{\partial l(y_i, \hat{y}^{t-1})}{\partial \hat{y}_i^{t-1}}$；$h_i$ 为损失函数的二阶导数：$h_i = \frac{\partial^2 l(y_i, \hat{y}^{t-1})}{\partial (\hat{y}_i^{t-1})^2}$。由于函数中的常量在函数最小化的过程中不起作用，因此我们可以移除掉常量项，得：

$$Obj^{(t)} \approx \sum_{i=1}^{n} [g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t) \tag{22}$$

由于要学习的函数仅仅依赖于目标函数，从式22可以看出只需为学习任务定义好损失函数，并为每个训练样本计算出损失函数的一阶导数和二阶导数，通过在训练样本集上最小化式22, 即可求得每步要学习的函数 $f(x)$，从而根据加法模型可得最终要学习的模型。

在 GBDT 中运用加法模型。对于一颗生成好的决策树，假设其叶子节点个数为 T，该决策树是由所有叶子节点对应的值组成的向量 $w \in R^T$，以及一个把特征向量映射到叶子节点索引的函数

$q : Rd \to 1, 2, \ldots, T$ 组成的。因此，决策树可以定义为：$f_t(x) = wq(x)$。决策树的复杂度可以由正则项 $\Omega(f_t) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$ 来定义，即决策树模型的复杂度由生成的树的叶子节点数量和叶子节点对应的值向量的 L2 范数决定。定义集合 $I_j = i|q(x_i) = j$ 为所有被划分到叶子节点 j 的训练样本的集合。式22可以根据树的叶子节点重新组织为 T 个独立的二次函数的和：

$$Obj^{(t)} \approx \sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t) \tag{23}$$

$$=\approx \sum_{i=1}^{n} \left[g_i w_{q(x_i)} + \frac{1}{2} h_i w_{q(x_i)}^2 \right] + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \tag{24}$$

$$= \sum_{j=1}^{T} \left[(\sum_{i \in I_j} g_i) w_j + \frac{1}{2} (\sum_{i \in I_j} h_i + \lambda) w_j^2 \right] + \gamma T \tag{25}$$

定义 $G_j = \sum_{i \in Ij} g_i$，$H_j = \sum_{i \in I_j} h_i$，则：

$$Obj^{(t)} = \sum_{j=1}^{T} \left[G_j w_j + \frac{1}{2} (H_j + \lambda) w_j^2 \right] + \gamma T \tag{26}$$

假设树的结构是固定的，即函数 $q(x)$ 确定，令目标函数 $Obj^{(t)}$ 的一阶导数为 0，即可求得叶子节点 j 对应的值为：

$$w_j^* = -\frac{G_j}{H_j + \lambda} \tag{27}$$

此时，目标函数的值为:

$$Obj^{(t)} = -\frac{1}{2} \sum_{j=1}^{T} \frac{G_j^2}{H_j + \lambda} + \gamma T \tag{28}$$

综上，为了便于理解，单颗决策树的学习过程可以大致描述为：

1. 枚举所有可能的树结构 q
2. 用式28为每个 q 计算其对应的分数 Obj，分数越小说明对应的树结构越好
3. 根据上一步的结果，找到最佳的树结构，用式27为树的每个叶子节点计算预测值

然而，可能的树结构数量是无穷的，所以实际上我们不可能枚举所有可能的树结构。通常情况下，我们采用贪心策略来生成决策树的每个节点：

1. 从深度为 0 的树开始，对每个叶节点枚举所有的可用特征
2. 针对每个特征，把属于该节点的训练样本根据该特征值升序排列，通过线性扫描的方式来决定该特征的最佳分裂点，并记录该特征的最大收益（采用最佳分裂点时的收益）
3. 选择收益最大的特征作为分裂特征，用该特征的最佳分裂点作为分裂位置，把该节点生长出左右两个新的叶节点，并为每个新节点关联对应的样本集
4. 回到第 1 步，递归执行到满足特定条件为止

在上述算法的第二步，样本排序的时间复杂度为 $O(n\log n)$，假设公用 K 个特征，那么生成一颗深度为 K 的树的时间复杂度为 $O(dKn\log n)$。具体实现可以进一步优化计算复杂度，比如可以缓存每个特征的排序结果等。

如何计算每次分裂的收益呢？假设当前节点记为 C，分裂之后左子节点记为 L，右子节点记为 R，则该分裂获得的收益定义为当前节点的目标函数值减去左右两个孩子节点的目标函数值之和：$Gain = Obj_C - Obj_L - Obj_R$，即：

$$Gain = \frac{1}{2}\left[\frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{(G_L+G_R)^2}{H_L+H_R+\lambda}\right] - \gamma \tag{29}$$

其中，$-\gamma$ 项表示因为增加了树的复杂性（该分裂增加了一个叶子节点）带来的惩罚。

因此，GBDT 的学习算法可总结为：

1. 算法每次迭代生成一颗新的决策树
2. 在每次迭代开始之前，计算损失函数在每个训练样本点的一阶导数 g_i 和二阶导数 h_i
3. 通过贪心策略生成新的决策树，通过27计算每个叶节点对应的预测值
4. 把新生成的决策树 $f_t(x)$ 添加到模型中：$\hat{y}_i^t = \hat{y}_i^{t-1} + f_t(xi)$

在第 4 步中，我们可以将模型更新换为：$\hat{y}_i^t = \hat{y}_i^{t-1} + \epsilon f_t(xi)$，其中 ϵ 称之为步长或者学习率。增加 ϵ 因子的目的是为了避免模型过拟合。

参考文献

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32. doi:10.1023/A:1010933404324

Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, *24*(6), 417–441. doi:10.1037/h0071325

Pearson, K. (1901). Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, *2*(11), 559–572. doi:10.1080/14786440109462720