Python机器学习基础教程

Stephen CUI

February 17, 2023

Chapter 1

数据表示与特征工程

到目前为止，我们一直假设数据是由浮点数组成的二维数组，其中每一列是描述数据点的连续特征（continuous feature）。对于许多应用而言，数据的收集方式并不是这样。一种特别常见的特征类型就是分类特征（categorical feature），也叫离散特征（discrete feature）。这种特征通常并不是数值。分类特征与连续特征之间的区别类似于分类和回归之间的区别，只是前者在输入端而不是输出端。

无论你的数据包含哪种类型的特征，数据表示方式都会对机器学习模型的性能产生巨大影响。额外的特征扩充（augment）数据也很有帮助，比如添加特征的交互项（乘积）或更一般的多项式。

对于某个特定应用来说，如何找到最佳数据表示，这个问题被称为特征工程（feature engineering），它是数据科学家和机器学习从业者在尝试解决现实世界问题时的主要任务之一。用正确的方式表示数据，对监督模型性能的影响比所选择的精确参数还要大。

1.1 分类变量

作为例子，我们将使用美国成年人收入的数据集，该数据集是从 1994 年的普查数据库中导出的。

1.1.1 One-Hot编码（虚拟变量）

到目前为止，表示分类变量最常用的方法就是使用 one-hot 编码（one-hot-encoding）或 N 取一编码（one-out-of-N encoding），也叫虚拟变量（dummy variable）。虚拟变量背后的思想是将一个分类变量替换为一个或多个新特征，新特征取值为 0 和 1。对于线性二分类（以及 scikit-learn 中其他所有模型）的公式而言，0 和 1 这两个值是有意义的，我们可以像这样对每个类别引入一个新特征，从而表示任意数量的类别。

> 我们使用的 one-hot 编码与统计学中使用的虚拟编码（dummy encoding）非常相似，但并不完全相同。为简单起见，我们将每个类别编码为不同的二元特征。在统计学中，通常将具有 k 个可能取值的分类特征编码为 k-1 个特征（都等于零表示最后一个可能取值）。这么做是为了简化分析（更专业的说法是，这可以避免使数据矩阵秩亏）。

将数据转换为分类变量的 one-hot 编码有两种方法：一种是使用 pandas，一种是使用 scikit-learn。

检查字符串编码的分类数据

读取完这样的数据集之后，最好先检查每一列是否包含有意义的分类数据。在处理人工（比如网站用户）输入的数据时，可能没有固定的类别，拼写和大小写也存在差异，因此可能需要预处理。举个例

Table 1.1: The first few entries in the adult dataset

	age	workclass	education	gender	hours-per-week	occupation	income
0	39	State-gov	Bachelors	Male	40	Adm-clerical	<=50K
1	50	Self-emp-not-inc	Bachelors	Male	13	Exec-managerial	<=50K
2	38	Private	HS-grad	Male	40	Handlers-cleaners	<=50K
3	53	Private	11th	Male	40	Handlers-cleaners	<=50K
4	28	Private	Bachelors	Female	40	Prof-specialty	<=50K
5	37	Private	Masters	Female	40	Exec-managerial	<=50K
6	49	Private	9th	Female	16	Other-service	<=50K
7	52	Self-emp-not-inc	HS-grad	Male	45	Exec-managerial	>50K
8	31	Private	Masters	Female	50	Prof-specialty	>50K
9	42	Private	Bachelors	Male	40	Exec-managerial	>50K
10	37	Private	Some-college	Male	80	Exec-managerial	>50K

子，有人可能将性别填为“male”（男性），有人可能填为“man” （男人），而我们希望能用同一个类别来表示这两种输入。检查列的内容有一个好方法，就是使用 pandas Series（Series 是 DataFrame 中单列对应的数据类型）的 `value_counts` 函数，以显示唯一值及其出现次数。

在实际的应用中，你应该查看并检查所有列的值。

用 pandas 编码数据有一种非常简单的方法，就是使用 `get_dummies` 函数。`get_dummies` 函数自动变换所有具有对象类型（比如字符串）的列或所有分类的列（这是 pandas 中的一个特殊概念）。

将输出变量或输出变量的一些导出属性包含在特征表示中，这是构建监督机器学习模型时一个非常常见的错误。

注意：pandas 中的列索引包括范围的结尾，因此 `'age':'occupation_ Transport-moving'` 中包括 `occupation_ Transport-moving`。这与 NumPy 数组的切片不同，后者不包括范围的结尾，例如 `np.arange(11)[0:10]` 不包括索引编号为 10 的元素。

警告

在这个例子中，我们对同时包含训练数据和测试数据的数据框调用 `get_dummies`。这一点很重要，可以确保训练集和测试集中分类变量的表示方式相同。
如果训练集和测试集的数据字段不同，或者字段的未知排列不同，将导致严重的错误!!!

1.1.2 数字可以编码分类变量

在 adult 数据集的例子中，分类变量被编码为字符串。一方面，可能会有拼写错误；但另一方面，它明确地将一个变量标记为分类变量。无论是为了便于存储还是因为数据的收集方式，分类变量通常被编码为整数。例如，假设 adult 数据集中的人口普查数据是利用问卷收集的，workclass 的回答被记录为 0（在第一个框打勾）、1（在第二个框打勾）、2（在第三个框打勾），等等。现在该列包含数字 0 到 8，而不是像 ”Private” 这样的字符串。如果有人观察表示数据集的表格，很难一眼看出这个变量应该被视为连续变量还是分类变量。但是，如果知道这些数字表示的是就业状况，那么很明显它们是不同的状态，不

应该用单个连续变量来建模。

> 分类特征通常用整数进行编码。它们是数字并不意味着它们必须被视为连续特征。一个整数特征应该被视为连续的还是离散的（one-hot 编码的），有时并不明确。如果在被编码的语义之间没有顺序关系（比如 workclass 的例子），那么特征必须被视为离散特征。对于其他情况（比如五星评分），哪种编码更好取决于具体的任务和数据，以及使用哪种机器学习算法。

1.2 分箱、离散化、线性模型与树

数据表示的最佳方法不仅取决于数据的语义，还取决于所使用的模型种类。线性模型与基于树的模型（比如决策树、梯度提升树和随机森林）是两种成员很多同时又非常常用的模型，它们在处理不同的特征表示时就具有非常不同的性质。

正如你所知，线性模型只能对线性关系建模，对于单个特征的情况就是直线。决策树可以构建更为复杂的数据模型，但这强烈依赖于数据表示。有一种方法可以让线性模型在连续数据上变得更加强大，就是使用特征**分箱**（binning，也叫离散化，即 discretization）将其划分为多个特征，如下所述。

我们假设将特征的输入范围划分成固定个数的箱子（bin），比如 10 个，那么数据点就可以用它所在的箱子来表示。

为了确定这一点，我们首先需要定义箱子，可以用`np.linspace`函数来定义箱子。接下来，我们记录每个数据点所属的箱子。这可以用 np.digitize 函数轻松计算出来。要想在这个数据上使用 scikit-learn 模型，我们利用 preprocessing 模块的 OneHotEncoder 将这个离散特征变换为 one-hot 编码。OneHotEncoder 实现的编码与 `pandas.get_dummies` 相同，但目前它只适用于值为整数的分类变量。

虚线和实线完全重合，说明线性回归模型和决策树做出了完全相同的预测。对于每个箱子，二者都预测一个常数值。因为每个箱子内的特征是不变的，所以对于一个箱子内的所有点，任何模型都会预测相同的值。比较对特征进行分箱前后模型学到的内容，我们发现，线性模型变得更加灵活了，因为现在它对每个箱子具有不同的取值，而决策树模型的灵活性降低了。分箱特征对基于树的模型通常不会产生更好的效果，因为这种模型可以学习在任何位置划分数据。从某种意义上来看，决策树可以学习如何分箱对预测这些数据最为有用。此外，决策树可以同时查看多个特征，而分箱通常针对的是单个特征。不过，线性模型的表现力在数据变换后得到了极大的提高。

对于特定的数据集，如果有充分的理由使用线性模型——比如数据集很大、维度很高，但有些特征与输出的关系是非线性的——那么分箱是提高建模能力的好方法。

1.3 交互特征与多项式特征

想要丰富特征表示，特别是对于线性模型而言，另一种方法是添加原始数据的**交互特征**（interaction feature）和**多项式特征**（polynomial feature）。这种特征工程通常用于统计建模，但也常用于许多实际的机器学习应用中。们知道，线性模型不仅可以学习偏移，还可以学习斜率。想要向分箱数据上的线性模型添加斜率，一种方法是重新加入原始特征。这样的话模型在每个箱子中都学到一个偏移，还学到一个斜率。但是学到的斜率在所有箱子中都相同——只有一个 x 轴特征，也就只有一个斜率。因为斜率在所有箱子中是相同的，所以它似乎不是很有用。我们更希望每个箱子都有一个不同的斜率！为了实现这一点，我们可以添加交互特征或乘积特征，用来表示数据点所在的箱子以及数据点在 x 轴上的位置。这个特征是箱子指示符与原始特征的乘积。

你可以将乘积特征看作每个箱子 x 轴特征的单独副本。它在箱子内等于原始特征，在其他位置等于零。

使用分箱是扩展连续特征的一种方法。另一种方法是使用原始特征的多项式（polynomial）。对于给定特征 x，我们可以考虑 x^2、x^3、x^4，等等。这在 preprocessing 模块的 PolynomialFeatures 中实现。

将多项式特征与线性回归模型一起使用，可以得到经典的多项式回归（polynomial regression）模型。

如你所见，多项式特征在这个一维数据上得到了非常平滑的拟合。但高次多项式在边界上或数据很少的区域可能有极端的表现。

显然，在使用 Ridge 时，交互特征和多项式特征对性能有很大的提升。但如果使用更加复杂的模型（比如随机森林），情况会稍有不同。

1.4 单变量非线性变换

我们刚刚看到，添加特征的平方或立方可以改进线性回归模型。其他变换通常也对变换某些特征有用，特别是应用数学函数，比如 log、exp 或 sin。虽然基于树的模型只关注特征的顺序，但线性模型和神经网络依赖于每个特征的尺度和分布。如果在特征和目标之间存在非线性关系，那么建模就变得非常困难，特别是对于回归问题。log 和 exp 函数可以帮助调节数据的相对比例，从而改进线性模型或神经网络的学习效果。在处理具有周期性模式的数据时，sin 和 cos 函数非常有用。

大部分模型都在每个特征（在回归问题中还包括目标值）大致遵循高斯分布时表现最好。使用诸如 log和 exp 之类的变换并不稀奇，但却是实现这一点的简单又有效的方法。在一种特别常见的情况下，这样的变换非常有用，就是处理整数计数数据时。计数数据是指类似“用户 A 多长时间登录一次？”这样的特征。计数不可能取负值，并且通常遵循特定的统计模式。

这种类型的数值分布（许多较小的值和一些非常大的值）在实践中非常常见[1]。但大多数线性模型无法很好地处理这种数据。

为数据集和模型的所有组合寻找最佳变换，这在某种程度上是一门艺术。在这个例子中，所有特征都具有相同的性质，这在实践中是非常少见的情况。通常来说，只有一部分特征应该进行变换，有时每个特征的变换方式也各不相同。前面提到过，对基于树的模型而言，这种变换并不重要，但对线性模型来说可能至关重要。对回归的目标变量 y 进行变换有时也是一个好主意。尝试预测计数（比如订单数量）是一项相当常见的任务，而且使用 $log(y+1)$ 变换也往往有用。

从前面的例子中可以看出，分箱、多项式和交互项都对模型在给定数据集上的性能有很大影响，对于复杂度较低的模型更是这样，比如线性模型和朴素贝叶斯模型。与之相反，基于树的模型通常能够自己发现重要的交互项，大多数情况下不需要显式地变换数据。其他模型，比如 SVM、最近邻和神经网络，有时可能会从使用分箱、交互项或多项式中受益，但其效果通常不如线性模型那么明显。

1.5 自动化特征选择

有了这么多种创建新特征的方法，你可能会想要增大数据的维度，使其远大于原始特征的数量。但是，添加更多特征会使所有模型变得更加复杂，从而增大过拟合的可能性。在添加新特征或处理一般的高维数据集时，最好将特征的数量减少到只包含最有用的那些特征，并删除其余特征。这样会得到泛化能力更好、更简单的模型。但你如何判断每个特征的作用有多大呢？有三种基本的策略：**单变量统计**（univariate statistics）、**基于模型的选择**（model-based selection）和**迭代选择**（iterative selection）。

1.5.1 单变量统计

在单变量统计中，我们计算每个特征和目标值之间的关系是否存在统计显著性，然后选择具有最高置信度的特征。对于分类问题，这也被称为**方差分析**（analysis of variance， ANOVA）。这些测试的一个

[1] 这是泊松分布，对计数数据相当重要。

关键性质就是它们是单变量的（univariate），即它们只单独考虑每个特征。因此，如果一个特征只有在与另一个特征合并时才具有信息量，那么这个特征将被舍弃。单变量测试的计算速度通常很快，并且不需要构建模型。另一方面，它们完全独立于你可能想要在特征选择之后应用的模型。

想要在 scikit-learn 中使用单变量特征选择，你需要选择一项测试——对分类问题通常是 `f_classif`（默认值），对回归问题通常是 `f_regression`——然后基于测试中确定的 p 值来选择一种舍弃特征的方法。所有舍弃参数的方法都使用阈值来舍弃所有 p 值过大的特征（意味着它们不可能与目标值相关，不能拒绝原假设）[2]。计算阈值的方法各有不同，最简单的是 SelectKBest 和 SelectPercentile，前者选择固定数量的 k 个特征，后者选择固定百分比的特征。

我们可以用 `get_support` 方法来查看哪些特征被选中，它会返回所选特征的布尔遮罩（mask）。

如果特征量太大以至于无法构建模型，或者你怀疑许多特征完全没有信息量，那么单变量特征选择还是非常有用的。

1.5.2 基于模型的特征选择

基于模型的特征选择使用一个监督机器学习模型来判断每个特征的重要性，并且仅保留最重要的特征。用于特征选择的监督模型不需要与用于最终监督建模的模型相同。特征选择模型需要为每个特征提供某种重要性度量，以便用这个度量对特征进行排序。决策树和基于决策树的模型提供了 `feature_importances_` 属性，可以直接编码每个特征的重要性。线性模型系数的绝对值也可以用于表示特征重要性。L1 惩罚的线性模型学到的是稀疏系数，它只用到了特征的一个很小的子集。这可以被视为模型本身的一种特征选择形式，但也可以用作另一个模型选择特征的预处理步骤。与单变量选择不同，基于模型的选择同时考虑所有特征，因此可以获取交互项（如果模型能够获取它们的话）。要想使用基于模型的特征选择，我们需要使用 SelectFromModel 变换器。SelectFromModel 类选出重要性度量（由监督模型提供）大于给定阈值的所有特征。

1.5.3 迭代特征选择

在单变量测试中，我们没有使用模型，而在基于模型的选择中，我们使用了单个模型来选择特征。在迭代特征选择中，将会构建一系列模型，每个模型都使用不同数量的特征。**有两种基本方法：开始时没有特征，然后逐个添加特征，直到满足某个终止条件；或者从所有特征开始，然后逐个删除特征，直到满足某个终止条件**。由于构建了一系列模型，所以这些方法的计算成本要比前面讨论过的方法更高。其中一种特殊方法是**递归特征消除**（recursive feature elimination，RFE），它从所有特征开始构建模型，并根据模型舍弃最不重要的特征，然后使用除被舍弃特征之外的所有特征来构建一个新模型，如此继续，直到仅剩下预设数量的特征。为了让这种方法能够运行，用于选择的模型需要提供某种确定特征重要性的方法，正如基于模型的选择所做的那样。

如果你不确定何时选择使用哪些特征作为机器学习算法的输入，那么自动化特征选择可能特别有用。它还有助于减少所需要的特征数量，加快预测速度，或允许可解释性更强的模型。在大多数现实情况下，使用特征选择不太可能大幅提升性能，但它仍是特征工程工具箱中一个非常有价值的工具。

1.6 利用专家知识

对于特定应用来说，在特征工程中通常可以利用专家知识（expert knowledge）。虽然在许多情况下，机器学习的目的是避免创建一组专家设计的规则，但这并不意味着应该舍弃该应用或该领域的先验知识。通常来说，领域专家可以帮助找出有用的特征，其信息量比数据原始表示要大得多。

[2] 原假设——无差异

在对这种时间序列上的预测任务进行评估时，我们通常希望从过去学习并预测未来。也就是说，在划分训练集和测试集的时候，我们希望使用某个特定日期之前的所有数据作为训练集，该日期之后的所有数据作为测试集。

LinearRegression 的效果差得多，而且周期性模式看起来很奇怪。其原因在于我们用整数编码一周的星期几和一天内的时间，它们被解释为连续变量。因此，线性模型只能学到关于每天时间的线性函数——它学到的是，时间越晚，租车数量越多。但实际模式比这要复杂得多。我们可以通过将整数解释为分类变量（用 OneHotEncoder 进行变换）来获取这种模式。

1.7 小结与展望

本章讨论了如何处理不同的数据类型（特别是分类变量）。我们强调了使用适合机器学习算法的数据表示方式的重要性，例如 one-hot 编码过的分类变量。还讨论了通过特征工程生成新特征的重要性，以及利用专家知识从数据中创建导出特征的可能性。特别是线性模型，可能会从分箱、添加多项式和交互项而生成的新特征中大大受益。对于更加复杂的非线性模型（比如随机森林和 SVM），在无需显式扩展特征空间的前提下就可以学习更加复杂的任务。在实践中，所使用的特征（以及特征与方法之间的匹配）通常是使机器学习方法表现良好的最重要的因素。