机器学习实战

Stephen CUI[1]

February 17, 2023

[1]cuixuanStephen@gmail.com

Contents

Part I

分类

前两部分主要探讨监督学习（supervised learning）。在监督学习的过程中，我们只需要给定输入样本集，机器就可以从中推演出指定目标变量的可能结果。

监督学习一般使用两种类型的目标变量：标称型和数值型。标称型目标变量的结果只在有限目标集中取值，如真与假、动物分类集合 { 爬行类、鱼类、哺乳类、两栖类 } ；数值型目标变量则可以从无限的数值集合中取值，如 0.100、42.001、1000.743 等。数值型目标变量主要用于回归分析。

Chapter 1

k-近邻算法

众所周知，电影可以按照题材分类，然而题材本身是如何定义的?由谁来判定某部电影属于哪个题材?也就是说同一题材的电影具有哪些公共特征?这些都是在进行电影分类时必须要考虑的问题。那么动作片具有哪些共有特征，使得动作片之间非常类似，而与爱情片存在着明显的差别呢？动作片中也会存在接吻镜头，爱情片中也会存在打斗场景，我们不能单纯依靠是否存在打斗或者亲吻来判断影片的类型。但是爱情片中的亲吻镜头更多，动作片中的打斗场景也更频繁，基于此类场景在某部电影中出现的次数可以用来进行电影分类。

1.1 k-近邻算法概述

简单地说，k-近邻算法采用测量不同特征值之间的距离方法进行分类。

k-近邻算法

优点：精度高、对异常值不敏感、无数据输入假定。
缺点：计算复杂度高、空间复杂度高。
适用数据范围：数值型和标称型。

k-近邻算法（kNN），它的工作原理是：存在一个样本数据集合，也称作训练样本集，并且样本集中每个数据都存在标签，即我们知道样本集中每一数据与所属分类的对应关系。输入没有标签的新数据后，将新数据的每个特征与样本集中数据对应的特征进行比较，然后算法提取样本集中特征最相似数据（最近邻）的分类标签。一般来说，我们只选择样本数据集中前k个最相似的数据，这就是k-近邻算法中k的出处，通常k是不大于20的整数。最后，选择k个最相似数据中出现次数最多的分类，作为新数据的分类。

k-近邻算法的一般流程

1. 收集数据：可以使用任何方法。
2. 准备数据：距离计算所需要的数值，最好是结构化的数据格式。
3. 分析数据：可以使用任何方法。
4. 训练算法：此步骤不适用于k-近邻算法。

5. 测试算法：计算错误率。
6. 使用算法：首先需要输入样本数据和结构化的输出结果，然后运行k-近邻算法判定输入数据分别属于哪个分类，最后应用对计算出的分类执行后续的处理。

1.1.1 实施kNN算法

实施kNN算法给出k-近邻算法的伪代码：

Algorithm 1: k-近邻算法

对未知类别属性的数据集中的每个点依次执行以下操作：

1. 计算已知类别数据集中的点与当前点之间的距离；
2. 按照距离递增次序排序；
3. 选取与当前点距离最小的k个点；
4. 确定前k个点所在类别的出现频率；
5. 返回前k个点出现频率最高的类别作为当前点的预测分类。

代码清单

kNN.py
kNN.ipynb

classify0()函数有4个输入参数：用于分类的输入向量是inX，输入的训练样本集为dataSet，标签向量为labels，最后的参数k表示用于选择最近邻居的数目，其中标签向量的元素数目和矩阵dataSet的行数相同。

计算完所有点之间的距离后，可以对数据按照从小到大的次序排序。然后，确定前k个距离最小元素所在的主要分类，输入k总是正整数；最后，将classCount字典分解为元组列表，然后使用程序第二行导入运算符模块的itemgetter方法，按照第二个元素的次序对元组进行排序。此处的排序为逆序，即按照从最大到最小次序排序，最后返回发生频率最高的元素标签。

1.1.2 如何测试分类器

分类器并不会得到百分百正确的结果，我们可以使用多种方法检测分类器的正确率。此外分类器的性能也会受到多种因素的影响，如分类器设置和数据集等。不同的算法在不同数据集上的表现可能完全不同。

为了测试分类器的效果，我们可以使用已知答案的数据，当然答案不能告诉分类器，检验分类器给出的结果是否符合预期结果。通过大量的测试数据，我们可以得到分类器的错误率——分类器给出错误结果的次数除以测试执行的总数。错误率是常用的评估方法，主要用于评估分类器在某个数据集上的执行效果。

1.2 示例：使用 k-近邻算法改进约会网站的配对效果

我的朋友海伦一直使用在线约会网站寻找适合自己的约会对象。海伦希望我们的分类软件可以更好地帮助她将匹配对象划分到确切的分类中。此外海伦还收集了一些约会网站未曾记录的数据信息，她认为这些数据更有助于匹配对象的归类。

1.2.1 准备数据：从文本文件中解析数据

海伦收集约会数据已经有了一段时间，她把这些数据存放在文本文件datingTestSet.txt中，每个样本数据占据一行，总共有1000行。海伦的样本主要包含以下3种特征：

- 每年获得的飞行常客里程数
- 玩视频游戏所耗时间百分比
- 每周消费的冰淇淋公升数

在将上述特征数据输入到分类器之前，必须将待处理数据的格式改变为分类器可以接受的格式。在kNN.py中创建名为file2matrix的函数，以此来处理输入格式问题。该函数的输入为文件名字符串，输出为训练样本矩阵和类标签向量。

修改一下kNN.py文件，向其添加file2matrix()函数。

首先我们需要知道文本文件包含多少行。打开文件，得到文件的行数。然后创建以零填充的矩阵NumPy。将该矩阵的另一维度设置为固定值3，你可以按照自己的实际需求增加相应的代码以适应变化的输入值。循环处理文件中的每行数据，首先使用函数line.strip()截取掉所有的回车字符，然后使用tab字符\t将上一步得到的整行数据分割成一个元素列表。接着，选取前3个元素，将它们存储到特征矩阵中。需要注意的是，我们必须明确地通知解释器，告诉它列表中存储的元素值为整型，否则Python语言会将这些元素当作字符串处理。

成功导入datingTestSet.txt文件中的数据之后，可以简单检查一下数据内容。

接着我们需要了解数据的真实含义。当然我们可以直接浏览文本文件，但是这种方法非常不友好，一般来说，我们会采用图形化的方式直观地展示数据。

1.2.2 分析数据：使用 Matplotlib 创建散点图

由于没有使用样本分类的特征值，我们很难从Figure 1.1中看到任何有用的数据模式信息。一般来说，我们会采用色彩或其他的记号来标记不同样本分类，以便更好地理解数据信息。Matplotlib库提供的scatter函数支持个性化标记散点图上的点。

1.2.3 准备数据：归一化数值

Table 1.1给出了提取的四组数据，如果想要计算样本3和样本4之间的距离，可以使用下面的方法：

$$\sqrt{(0-67)^2+(20000-32000)^2+(1.1-0.1)^2}$$

Table 1.1: 约会网站原始数据改进之后的样本数据

	玩视频游戏所耗时间百分比	每年获得的飞行常客里程数	每周消费的冰淇淋公升数	样本分类
1	0.8	400	0.5	1
2	12	134, 000	0.9	3
3	0	20, 000	1.1	2
4	67	32, 000	0.1	2

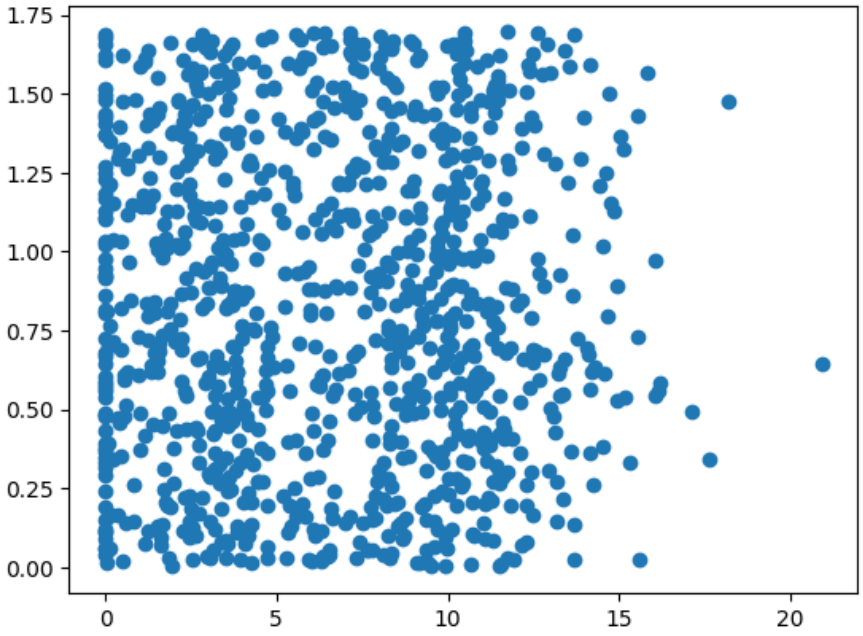

Figure 1.1: Dating data without class labels

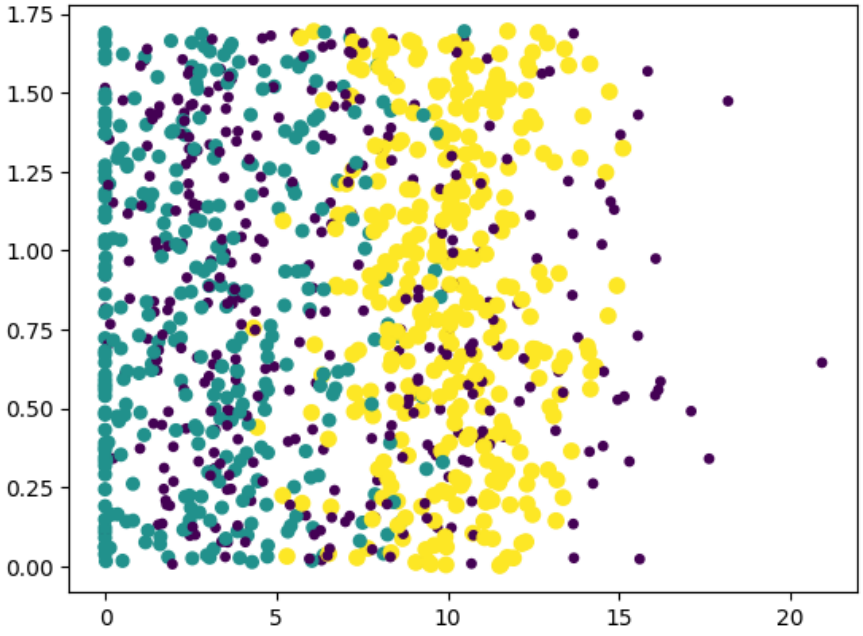

Figure 1.2: Dating data with markers changed by class label

我们很容易发现，上面方程中数字差值最大的属性对计算结果的影响最大，也就是说，每年获取的飞行常客里程数对于计算结果的影响将远远大于表2-3中其他两个特征——玩视频游戏的和每周消费冰淇淋公升数——的影响。而产生这种现象的唯一原因，仅仅是因为飞行常客里程数远大于其他特征值。但海伦认为这三种特征是同等重要的，因此作为三个等权重的特征之一，飞行常客里程数并不应该如此严重地影响到计算结果。

在处理这种不同取值范围的特征值时，我们通常采用的方法是将数值归一化，如将取值范围处理为0到1或者　1到1之间。下面的公式可以将任意取值范围的特征值转化为0到1区间内的值：

$$newValue = \frac{oldValue - min}{max - min}$$

其中min和max分别是数据集中的最小特征值和最大特征值。虽然改变数值取值范围增加了分类器的复杂度，但为了得到准确结果，我们必须这样做。我们需要在文件kNN.py中增加一个新函数autoNorm()，该函数可以自动将数字特征值转化为0到1的区间。

在函数autoNorm()中，我们将每列的最小值放在变量minVals中，将最大值放在变量maxVals中，其中dataSet.min(0)中的参数0使得函数可以从列中选取最小值，而不是选取当前行的最小值。然后，函数计算可能的取值范围，并创建新的返回矩阵。为了归一化特征值，我们必须使用当前值减去最小值，然后除以取值范围。需要注意的是，特征值矩阵有1000×3个值，而minVals和range的值都为1×3。为了解决这个问题，我们使用NumPy库中tile()函数将变量内容复制成输入矩阵同样大小的矩阵，注意这是具体特征值相除

1.2.4 测试算法：作为完整程序验证分类器

机器学习算法一个很重要的工作就是评估算法的正确率，通常我们只提供已有数据的90%作为训练样本来训练分类器，而使用其余的10%数据去测试分类器，检测分类器的正确率。需要注意的是，10%的测试数据应该是随机选择的，由于海伦提供的数据并没有按照特定目的来排序，所以我们可以随意选择10%数据而不影响其随机性。

对于分类器来说，错误率就是分类器给出错误结果的次数除以测试数据的总数。

函数datingClassTest首先使用了file2matrix和autoNorm()函数从文件中读取数据并将其转换为归一化特征值。接着计算测试向量的数量，此步决定了normMat向量中哪些数据用于测试，哪些数据用于分类器的训练样本；然后将这两部分数据输入到原始kNN分类器函数classify0。最后，函数计算错误率并输出结果。

运行`kNN.datingClassTest()`，得到分类器处理约会数据集的错误率是2.4%，这是一个相当不错的结果。我们可以改变函数datingClassTest内变量hoRatio和变量k的值，检测错误率是否随着变量值的变化而增加。依赖于分类算法、数据集和程序设置，分类器的输出结果可能有很大的不同。

1.2.5 使用算法：构建完整可用系统

现在终于可以使用这个分类器为海伦来对人们分类。我们会给海伦一小段程序，通过该程序海伦会在约会网站上找到某个人并输入他的信息。程序会给出她对对方喜欢程度的预测值。

在kNN.py中添加classifyPerson()函数。

目前为止，我们已经看到如何在数据上构建分类器。这里所有的数据让人看起来都很容易，但是如何在人不太容易看懂的数据上使用分类器呢？下一个例子将在在二进制存储的图像数据上使用kNN。

1.3 示例：手写识别系统

本节我们一步步地构造使用k-近邻分类器的手写识别系统。为了简单起见，这里构造的系统只能识别数字0到9。需要识别的数字已经使用图形处理软件，处理成具有相同的色彩和大小[1]：宽高是32像素×32像素的黑白图像。尽管采用文本格式存储图像不能有效地利用内存空间，但是为了方便理解，我们还是将图像转换为文本格式。

1.3.1 准备数据：将图像转换为测试向量

实际图像存储在两个目录内：目录trainingDigits中包含了大约2000个例子，每个例子的内容如图2-6所示，每个数字大约有200个样本；目录testDigits中包含了大约900个测试数据。我们使用目录trainingDigits中的数据训练分类器，使用目录testDigits中的数据测试分类器的效果。两组数据没有重叠，你可以检查一下这些文件夹的文件是否符合要求。

为了使用前面两个例子的分类器，我们必须将图像格式化处理为一个向量。我们将把一个32×32的二进制图像矩阵转换为1×1024的向量，这样前两节使用的分类器就可以处理数字图像信息了。

首先编写一段函数img2vector，将图像转换为向量：该函数创建1×1024的NumPy数组，然后打开给定的文件，循环读出文件的前32行，并将每行的头32个字符值存储在NumPy数组中，最后返回数组。

1.3.2 测试算法：使用 k-近邻算法识别手写数字

handwritingClassTest()是测试分类器的代码，将其写入kNN.py文件中。在写入这些代码之前，我们必须确保将`from os import listdir`写入文件的起始部分，这段代码的主要功能是从os模块中导入函数listdir，它可以列出给定目录的文件名。

handwritingClassTest()将trainingDigits目录中的文件内容存储在列表中，然后可以得到目录中有多少文件，并将其存储在变量m中。接着，代码创建一个m行1024列的训练矩阵，该矩阵的每行数据存储一个图像。我们可以从文件名中解析出分类数字。该目录下的文件按照规则命名，如文件9_45.txt的分类是9，它是数字9的第45个实例。然后我们可以将类代码存储在hwLabels向量中，使用前面讨论的img2vector函数载入图像。在下一步中，我们对testDigits目录中的文件执行相似的操作，不同之处是我们并不将这个目录下的文件载入矩阵中，而是使用classify0()函数测试该目录下的每个文件。

k-近邻算法识别手写数字数据集，错误率为1%。改变变量k的值、修改函数handwritingClassTest随机选取训练样本、改变训练样本的数目，都会对k-近邻算法的错误率产生影响，感兴趣的话可以改变这些变量值，观察错误率的变化。实际使用这个算法时，算法的执行效率并不高。因为算法需要为每个测试向量做2000次距离计算，每个距离计算包括了1024个维度浮点运算，总计要执行900次。是否存在一种算法减少存储空间和计算时间的开销呢？k决策树就是k-近邻算法的优化版，可以节省大量的计算开销。

k-近邻算法是分类数据最简单最有效的算法，是基于实例的学习，使用算法时我们必须有接近实际数据的训练样本数据。k-近邻算法必须保存全部数据集，如果训练数据集的很大，必须使用大量的存储空间。此外，由于必须对数据集中的每个数据计算距离值，实际使用时可能非常耗时。

k-近邻算法的另一个缺陷是它无法给出任何数据的基础结构信息，因此我们也无法知晓平均实例样本和典型实例样本具有什么特征。

[1] The dataset is a modified version of the "Optical Recognition of Handwritten Digits Data Set" by E. Alpaydin, C. Kaynak, Department of Computer Engineering at Bogazici University, 80815 Istanbul Turkey, retrieved from the UCI Machine Learning Repository (http://archive.ics.uci.edu/ml) on October 3, 2010.

Chapter 2

决策树

你是否玩过二十个问题的游戏，游戏的规则很简单：参与游戏的一方在脑海里想某个事物，其他参与者向他提问题，只允许提20个问题，问题的答案也只能用对或错回答。问问题的人通过推断分解，逐步缩小待猜测事物的范围。决策树的工作原理与20个问题类似，用户输入一系列数据，然后给出游戏的答案。我们经常使用决策树处理分类问题，近来的调查表明决策树也是最经常使用的数据挖掘算法[1]。

它之所以如此流行，一个很重要的原因就是不需要了解机器学习的知识，就能搞明白决策树是如何工作的。

k-近邻算法可以完成很多分类任务，但是它最大的缺点就是无法给出数据的内在含义，决策树的主要优势就在于数据形式非常容易理解。

决策树的一个重要任务是为了数据中所蕴含的知识信息，因此决策树可以使用不熟悉的数据集合，并从中提取出一系列规则，在这些机器根据数据集创建规则时，就是机器学习的过程。

2.1 决策树的构造

决策树

优点：计算复杂度不高，输出结果易于理解，对中间值的缺失不敏感，可以处理不相关特征数据。
缺点：可能会产生过度匹配问题。
适用数据类型：数值型和标称型。

首先我们讨论数学上如何使用**信息论**划分数据集，然后编写代码将理论应用到具体的数据集上，最后编写代码构建决策树。

在构造决策树时，我们需要解决的第一个问题就是，当前数据集上哪个特征在划分数据分类时起决定性作用。为了找到决定性的特征，划分出最好的结果，我们必须评估每个特征。完成测试之后，原始数据集就被划分为几个数据子集。这些数据子集会分布在第一个决策点的所有分支上。如果某个分支下的数据属于同一类型，则无需进一步对数据集进行分割。如果数据子集内的数据不属于同一类型，则需要重复划分数据子集的过程。如何划分数据子集的算法和划分原始数据集的方法相同，直到所有具有相同类型的数据均在一个数据子集内。

[1] Giovanni Seni and John Elder, Ensemble Methods in Data Mining: Improving Accuracy Through Combining Predictions, Synthesis Lectures on Data Mining and Knowledge Discovery (Morgan and Claypool, 2010), 28.

创建分支的伪代码函数createBranch()如下所示：

Algorithm 2: 决策树分支判断

```
if 数据集中的每个子项属于同一分类 then
    return 类标签
else
    寻找划分数据集的最好特征;
    划分数据集;
    创建分支节点;
    for 每个划分的子集 do
        调用函数createBranch并增加返回结果到分支节点中
    end
    return 分支节点(返回子树或者说决策树更好一点)
end
```