朴素贝叶斯分类器实验报告

计 45 王晨阳

2014011407

1 文件说明

源代码在 src 文件夹下，主要包含 $main.cpp$、$binning.cpp$、、$missing.cpp$ 和 $tools.h$。$main.cpp$ 是主程序，包含整个训练、分类、评价的过程；$binning.cpp$ 是用将连续值离散化的方法进行检测的程序；$missing.cpp$ 是将缺失值当做一个单独特征取值的程序；$tools.h$ 是使用到的工具函数。程序运行会在当前目录下生成一个 $output$ 文件，里面记录了训练集的信息和测试结果及评价。程序中使用的是相对路径，所以源代码位置不要改变。

2 算法实现

主程序包含四个主要步骤：初始化、训练数据、测试数据和性能评价。

- 初始化由函数 $init()$ 完成，初始了一些全局变量的值，为训练数据做准备。
- 训练数据由函数 $loadData()$ 完成，函数首先读取 $adult.train$ 文件中的所有信息，并把每个条目保存到变量 $samples$ 中。之后再根据设定的比率，取其中一部分条目作为训练集，进行一次或多次遍历（取决于对连续取值特征的处理办法，至多两次遍历），从中获取训练信息，主要获得的信息有训练集规模、每个类别的总数、每个特征的每种取值在不同类别中的数目、连续取值特征在不同类别的均值和方差、各个特征在不同类别下最可能的取值（出现次数最多）、各个特征分别有多少种取值。根据对一些问题的不同解决方法这里对信息的处理也会有所不同。最终将这些信息输出到文件中。
- 测试主要由函数 $classify()$ 完成，参数为一个人的特征向量，输出为一个布尔值，表示预测和实际是否相符。其中会调用函数 $P()$ 计算 $P(x_i \mid y)$，计算方法有所不同，后面会详细讨论。
- 最后主函数中会根据测试的结果对性能进行评价，主要通过三个常见指标来衡量，分别为准确度 $Accuracy$、精度 $Precision$、召回率 $Recall$ 和 $F1$ 值，具体将在下一小节介绍。

3 性能评价

若对于某一个类别 y，记 $correct_y$ 为正确预测为类别 y 的个数，$pred_y$ 为不论对错预测为 y 的个数，all_y 为测试集中所有属于 y 的个数，则三个评价指标可以如下计算：

准确度 $Accuracy$：

$$Accuracy = \frac{\sum_y correct_y}{\sum_y all_y}$$

某一类别 y 下的精度 $Precision_y$：

$$Precision_y = \frac{correct_y}{pred_y}$$

某一类别 y 下的召回率 $Recall_y$：

$$Recall_y = \frac{correct_y}{all_y}$$

某一类别 y 下的 $F1_y$ 值：

$$\frac{1}{F_y} = \frac{1}{2}(\frac{1}{Precision_y} + \frac{1}{Recall_y})$$

由此可以看出准确度体现了总体上的预测正确率，精度体现了某一类别下的预测正确率，召回率体现了对某类别中的所有条目能正确预测的概率。三者都从不同方面一定程度上反映了算法表现和分类性能。$F1$ 值则是精度和召回率的调和平均，在两者中间达到某种平衡。

举具体例子来说，在 $main.cpp$ 中，对连续取值的特征用高斯分布拟合，并对缺失的特征值用训练集中出现频次最高的取值代替，通过该方法进行分类最后结果如下：

Training Scale	Testcase	Accuracy	Precision		Recall		F1-Measure	
			$\leq 50K$	$> 50K$	$\leq 50K$	$> 50K$	$\leq 50K$	$> 50K$
32561(100%)	16281	83.12%	86.15%	69.04%	92.82%	51.77%	0.8936	0.5917

4 结果分析

对结果的分析主要从三个方面进行，分别是训练集的规模、零概率问题、特征值连续和缺失处理。将讨论在三个问题上采用不同方法对算法性能的影响并加以分析。

4.1 Issue1：训练集规模

程序中先读入全部样本，再通过全局 TRAIN_RATE 变量控制参与训练的样本规模。这里先令 TRAIN_RATE 为 1、0.5、0.05，即每次从全部样本随机取样 100%、50%、5% 观察结果。

从下表中可以看出，当训练集规模减少时，得到的结果并没有如预想般准确率降低，反而各项性能都有提高。这一方面说明朴素贝叶斯分类方法具有很好的鲁棒性，即使训练集的规模较小也能得到比较好的分类效果；另一方面说明该数据集中可能有一些极端数据，使得训练的数据集增大时受到影响，特别是 $> 50K$ 的数据，因为可以看出该类别的评价指标在训练集减小时提升较大。

但是当训练集规模过小时预测准确率也会明显降低。比如取 0.1% 的样本作为训练集，最终的准确率就只有 78.92%，显著下降，这也符合一般预期。

TRAIN_RATE	Testcase	Accuracy	Precision		Recall		F1-Measure	
			$\leq 50K$	$> 50K$	$\leq 50K$	$> 50K$	$\leq 50K$	$> 50K$
1(100%)	16281	83.12%	86.15%	69.04%	92.82%	51.77%	0.8936	0.5917
0.5(50%)	16281	83.23%	86.19%	69.39%	92.92%	51.87%	0.8943	0.5937
0.05(5%)	16281	83.67%	86.89%	69.59%	92.59%	54.81%	0.8965	0.6132

之后采用随机选取训练集规模的方法，令 TRAIN_RATE 为随机 0 到 1 中间的浮点数，测试五次，结果如下：

TRAIN_RATE	Testcase	Accuracy	Precision		Recall		F1-Measure	
			$\leq 50K$	$> 50K$	$\leq 50K$	$> 50K$	$\leq 50K$	$> 50K$
0.061(6.1%)	16281	83.62%	86.65%	69.95%	92.86%	53.74%	0.8965	0.6079
0.457(45.7%)	16281	83.23%	86.30%	69.13%	92.76%	52.39%	0.8942	0.5961
0.873(87.3%)	16281	83.20%	83.18%	69.30%	92.90%	51.82%	0.8941	0.5930
0.042(4.2%)	16281	**83.84**%	87.49%	68.95%	92.00%	57.46%	0.8969	0.6269
0.854(85.4%)	16281	83.20%	86.23%	69.20%	92.83%	52.05%	0.8941	0.5942
Average	16281	83.42%	85.97%	69.31%	92.67%	53.49%	0.8952	0.6036

表中列出了五次随机取样的各项指标平均值，总体 $Accuracy$ 基本在 83.4% 上下浮动。同时也标出了最大值和最小值，分别用加粗和下划线表示，最大值出现在使用 4.2% 训练集时，最小值有两个，分别是使用 87.3% 和 85.4% 训练集时出现。也符合上面总结的变化规律，在一定范围内，当训练集较小时准确率反而较高。

4.2 Issue2：零概率问题

当测试集中某一个特征取值在训练集中某类别下没有出现过时，计算所得的概率 $P(x_i \mid y)$ 就会是 0，与其他概率相乘后最终得到概率也为 0，进行比较的话一定不会判断属于该类别，但不应因为一个在该类别中未出现过的特征取值而否定属于该类，因此领概率会对类别的判定带来问题。

因此在计算 $P(x_i \mid y)$ 时加入平滑的概念，令 N_{yi} 表示第 i 维特征的某个取值在类别 y 下的数目，N_y 表示属于类别 y 的总数目，α 为系数，M 为第 i 维特征最大可能的取值个数，修正

$P(x_i \mid y)$ 的计算如下：

$$P(x_i \mid y) = \frac{N_{yi} + \alpha}{N_y + \alpha M}$$

当系数 α 的取值为 1 时，即为 *Laplace* 平滑，或称加 1 平滑。针对该数据集，在保证其他条件不变的情况下，改变 α 的值比较结果，发现取 0.2 时得到的准确率最高，因此在本实验中取 $\alpha = 0.2$。而 M 在程序对训练数据进行遍历时，可以记录每一维特征出现过的不重复的取值个数，作为该维特征的 M。

下面列出使用平滑和不使用平滑的性能对比（100% 训练集）：

Smooth	Testcase	Accuracy	Precision		Recall		F1-Measure	
			$\leq 50K$	$> 50K$	$\leq 50K$	$> 50K$	$\leq 50K$	$> 50K$
No	16281	83.1153%	86.15%	69.03%	92.82%	51.74%	0.8936	0.5915
Yes	16281	83.1214%	86.15%	69.04%	92.82%	51.77%	0.8936	0.5917

从表中可以看出加入平滑之后在准确率以及 $> 50K$ 类别的各项指标上都有一定的提升，说明平滑方法确实可以避免一些分类错误，特别是对于 $> 50K$ 的类别，从数据可以看出也许测试集中该类别出现了一些训练集没有的特征取值，因此加入平滑后能正确分到该类别，而不使用平滑错误率相对更高一些。

4.3 Issue3：特征值连续和缺失处理

首先考虑特征值连续的问题。对于连续的特征取值，对于测试集中的某一个特征值计算某类别下的 $P(x_i \mid y)$ 时，没法去找训练集中出现该特征值的数目，因为该特征整个取值是连续的。为此就要想办法使得 $P(x_i \mid y)$ 可以方便地求出。

简单来想的话可以将连续的取值分成互不重叠的小段，每一段代表一个特征取值，这样就把连续的取值离散化了。*src* 文件夹中的 *binning.cpp* 是使用该方法解决连续取值问题的程序，基本思想是，在第一次遍历训练集时找到所有连续取值特征的最大值和最小值；之后定义一个组数 *gapGroup* 即要把这些连续值离散为多少段，程序中为 20，用各个最大值减去相应最小值再除以这个组数得到间隔距离 *gap*，即对于第 i 维特征

$$gap_i = \frac{maxValue_i - minValue_i}{gapGroup}$$

这样对于每个连续值 num_i，都可以通过公式

$$\frac{num_i - minValue_i}{gap_i}$$

得到对应的段标号。只需再遍历一遍训练集将所有的连续取值特征划分到对应的段并计数保存。在进行检测时，找到待检测数落入的段内训练集数据的个数就可以计算 $P(x_i \mid y)$ 了。

再考虑一下其他方法，可以假定连续数据符合高斯分布，如此就可以用高斯分布概率公式求出任意给定特征值的概率

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_{yi}^2}} exp\,(-\frac{(x_i - \mu_{yi})^2}{2\sigma_{yi}^2})$$

其中 μ_{yi} 和 σ_{yi}^2 分别表示类别 y 下第 i 维特征的均值和方差。这里均值和方差均通过最大似然法得到，由概率论知识可知最大似然法得到的均值即为样本均值，方差为总体方差，即

$$\mu_{yi} = \frac{1}{N_y}\sum x_i \qquad \sigma_{yi}^2 = \sum \frac{(x_i - \mu_{yi})^2}{N_y}$$

为此只需在第一遍遍历训练集时计算出均值，第二遍遍历时计算出方差，之后计算 $P(x_i \mid y)$ 就可以直接使用高斯分布的公式来计算了。

这里将两种方法得到的结果对比如下（100% 训练集）：

Method	Testcase	Accuracy	Precision		Recall		F1-Measure	
			$\leq 50K$	$> 50K$	$\leq 50K$	$> 50K$	$\leq 50K$	$> 50K$
Binning	16281	82.49%	92.75%	59.82%	83.62%	78.86%	0.8795	0.6804
Gaussian	16281	83.12%	86.15%	69.04%	92.82%	51.77%	0.8936	0.5917

从表中可以看出，使用高斯分布拟合的方法得到的准确率明显高于离散化方法，毕竟高斯分布对于已知信息概率分布的刻画更为细腻，能更好地表示某一取值的概率。而离散化方法强行分为若干段还是有些粗糙了。从离散化方法 $\leq 50K$ 的 Precision 高 Recall 低，$> 50K$ 的 Precision 高 Recall 低，可以看出离散化方法的最终预测错误地把更多的人分类到 $> 50K$ 的类别中，更倾向于将人分到 $> 50K$ 的类别中，而对分到 $\leq 50K$ 的类别比较谨慎，所以导致这样的数据差异。

再来看对数据缺失的处理。在训练和测试中都可能遇到某些特征的取值缺失的问题，对这些缺失一种简单的方法是直接忽略，不考虑这些缺失。朴素贝叶斯方法本身由于对各个特征的独立性假设，对特征缺失不是很敏感，直接忽略是一种可行的方法，只是缺失的特征对判断该条目没有任何贡献。

另一种方法是将每个特征里的缺失（本数据集中为"?"）作为一个特殊取值，这样在测试中遇到缺失就看训练集中同样该特征缺失的数量有多少，以此来计算相应概率 $P(x_i \mid y)$。但这样处理的问题是，两个条目的某一特征均缺失并不能说明两者有相似性，以此来计算概率可以算为一种处理方法，但实际对分类结果的贡献并不一定为正。

最后一种方法是通过训练集预测测试集中缺失的特征取值。具体思想是在训练时记录每个类别下每一维特征出现次数最多的取值，作为该特征最可能的取值（连续取值的特征取均值），记为 $mostPossible$。这样这个最可能取值代表了最普遍的一种特征，遇到测试集中某特征缺失的话就可以用该特征在某类别下的 $mostPossible$ 来替代计算。

下面对三种方法得到的结果列表如下，处理连续特征值的方法均为离散化，训练集均为 100%（src 文件夹下 $missing.cpp$ 即为将缺失当做一种特殊取值处理的程序）：

Method	Accuracy	Precision		Recall		F1-Measure	
		$\leq 50K$	$> 50K$	$\leq 50K$	$> 50K$	$\leq 50K$	$> 50K$
Ignore	82.58%	92.73%	60.02%	83.77%	78.76%	0.8802	0.6812
Special Category	82.81%	92.65%	60.51%	84.17%	78.42%	0.8821	0.6831
Most Possible Pred	82.49%	92.75%	59.82%	83.62%	78.86%	0.8795	0.6804

分析表中数据，直接忽略缺失值得到的结果可以作为一个 $Baseline$。将缺失值当做特殊取值在性能上有所提升，并且是三种方法中表现最好的一个。说明在本数据集中，两个条目同一特征缺失对预测为相同类别确实有正向的贡献。属于同一类别的数据在缺失的特征值上有相似性。

而用训练集出现最多的取值来代替缺失值的方法效果反而比 $Baseline$ 还要差，这个结果出乎原本的预期。*main.cpp* 中实现的方法就是用最可能的值代替，然而这个方法却是表现最差的。从 $Precision$ 和 $Recall$ 出发分析原因，$\leq 50K$ 的 Precision 比 $Baseline$ 高而 Recall 较小；$> 50K$ 的 Precision 比 $Baseline$ 低而 Recall 较高。这个差异说明该方法相比 $Baseline$ 更倾向于分类到 $> 50K$ 的类别中，分到 $\leq 50K$ 相对比较谨慎。输出两个类别下各个特征的 $mostPossible$ 可以看出不同类别的最可能取值大多是不同的，因此猜想可能的原因如下：在 $> 50K$ 的类别下某些 $mostPossible$ 对于将一个人判定为 $> 50K$ 贡献很大。这样当对一个本来是 $\leq 50K$ 的人进行判定时，把其中的缺失值定为 $> 50K$ 类别中最可能的取值时，由于可能该特征对判定贡献很大，就使加上该特征后属于 $> 50K$ 的概率更高，从而导致错误地将这个人分到 $> 50K$ 的类别中。

5 总结与讨论

总之，本次实验实现了一个简单的朴素贝叶斯分类器，并在一个特定的数据集上进行了测试。同时对算法细节上的一些问题，如训练集规模、零概率问题、特征值连续和确实等。用不同的方法进行实现并对比分析，并通过多种评价指标对方法的性能进行评价。最终得到的结果如下：

- 在该数据集下，训练集在一定范围内较小时性能较好（4% 左右）；
- 加入平滑避免零概率问题对性能表现有一定提升，参数 $\alpha = 0.2$ 时效果最好；
- 对于连续特征值用高斯分布拟合能得到更好的效果，而对缺失值当做一种特殊的特征取值性能最高。

该实验中也尚有一些问题，比如用最可能的值代替缺失值只给出了一个简单的解释；对 $P(x_i \mid y)$ 的计算也有其他更优的拟合方法，有待进一步研究。