基于混合高斯模型的电子邮件多过滤器融合方法

李文斌[1,2], 刘椿年[1], 陈嶷瑛[2]
(1. 北京工业大学多媒体与智能软件北京重点实验室, 北京 100022; 2. 石家庄经济学院信息工程学院, 河北石家庄 050031)

摘 要: 本文提出了一种基于混合高斯模型(GMM)的多贝叶斯过滤器融合方法,并成功应用于电子邮件过滤.该方法使用多元统计分析方法对多个过滤器在训练例上的过滤表现矩阵进行降维和除噪,得到训练数据及各过滤器的分布;然后,从这一分布中学习出对邮件进行类别判定的 GMM. GMM 根据期望代价最小准则进行过滤,避免将正常邮件判定为垃圾.实验结果表明,本文方法具有较好的过滤性能,且对于特征提取率的敏感度低.

关键词: 代价敏感; 邮件过滤; 混合高斯模型; 过滤器融合
中图分类号: TP181 文献标识码: A 文章编号: 0372-2112 (2006) 02-0247-05

Combining Multiple Email Filters of Naïve Bayes Based on GMM

LIW en-bin[1,2], LIU Chun-nian[1], CHEN Y i-ying[2]
(1. Multimedia and Intelligent Software Beijing Key Lab., Beijing University of Technology, Beijing 100022, China;
2. School of Information Engineer, Shijiazhuang University of Economics, Shijiazhuang, Hebei 050031, China)

Abstract An algorithm combining multiple Naïve Bayesian (NB) filters based on GMM is presented, which has been successfully applied to e-mail filtering. The method uses the multiple variates statistics analysis to model the relationship between the training data set and their classification by a collection of NB filters. Then a GMM can be learned from the resulting representation. The GMM filters previously unseen e-mails according to the principle of minimizing expected-error-cost, in order to avoid deleting useful e-mails. Experimental results confirm the validity of our method, and show that our approach is insensitive to ratio of feature subset selection.

Key words cost-sensitive; e-mail filter; GMM; combing multiple filters

1 引言

由于 SMTP协议的缺限,邮件系统不对发件人身份进行认证便可以发送或转发邮件,导致因特网上大量垃圾电子邮件(以下简称为垃圾)的泛滥.因此,研究如何有效地过滤垃圾已成为热点.

目前,许多邮件客户端采用基于规则或关键字的方法过滤垃圾,这类方法的最大缺点是要求用户手动构造规则.尽管研究人员提出了自动学习过滤规则的方法,但由于垃圾制造者经常变换关键词,使这一方法的过滤效果捉磨不定.

文献[1, 2]等使用机器学习的办法过滤垃圾.该类方法的基本思路是从本质上将邮件过滤视为文本分类问题[3],使用文本分类器将邮件分为“垃圾”和“非垃圾”两类.其做法是:首先进行最优的特征提取,然后设计最优的分类器用以分类.但实际中要达到这两个“最优”是非常困难的.近年来,更流行的做法是融合多个分类器,这样既能降低对单分类器性能最优的苛刻要求,又能较容易地得到高性能的识别系统[4].

Bagging、Boosting、神经网络集成、基于证据理论的集成、选举法(Vote)等方法可以用于对多个分类器进行融合,文献[4, 5]对这些方法作了介绍.遗憾的是,这些方法不能直接用于组合过滤邮件的分类器,原因在于:邮件过滤是代价敏感的分类问题,即将垃圾误分为非垃圾(以下称为误收)和将非垃圾误分为垃圾(以下称为误拒)的代价是不同的,而已有的融合方法通常用在非代价敏感的分类场合下.这里所说的代价,通常指的是时间花费、可能的财产损失、或对工作带来的不便程度等方面.

本文的研究重点主要集中在两个方面:研究对朴素贝叶斯过滤器进行融合的方法;研究实现代价敏感的过滤器及评价方法.本文方法的主要思想是:(1)将训练例集表示成一个二维表.表中每一行代表一个训练例;每一列代表

收稿日期: 2005-07-18; 修回日期: 2005-10-28
基金项目: 国家自然科学基金 (No. 60496322)

一个过滤器, 表中的每一个单元表示某过滤器判定相应的训练例属于垃圾或非垃圾的后验概率. 对该表作对应分析可以得到垃圾或非垃圾训练数据在 K 维空间的分布情况及分布关系; (2)因为混合高斯模型 (GMM)可以用多个简单高斯分布的线性组合来逼近任意的概率分布[6,7], 因此采用 GMM 可以有效地表示分布在 K 维空间的训练数据的类条件概率密度; (3)对于新邮件, 可以转换成 K 维空间的点, 利用 GMM 可以计算出新邮件属于垃圾或非垃圾的概率, 结合代价敏感的判定规则, 即可以为新邮件指定最终的类别.

2 GMM 及 EM 算法

垃圾邮件过滤的本质是两类分类问题, 即将邮件分为“垃圾”和“非垃圾”两类. 设 c_0 = “非垃圾”, c_1 = “垃圾”, c_i 的训练样本集为 X_i (i = 0, 1), 并假定每个类别出现的先验概率相等. 样本 $\boldsymbol{x}$ 在 c_i 中出现的条件概率密度为:

$$p(c_i|\boldsymbol{x}, \lambda) = \sum_{j=1}^{M} w_{ij} p_{ij}(\boldsymbol{x})$$

上式中, $\boldsymbol{x}$ 代表一个训练例, 它是一个 p 维向量; λ代表 GMM 模型参数; $p_{ij}(\boldsymbol{x}) \sim N_p(\mu_{ij}, \upsilon_{ij})$, 是混合密度中第 j 个分量的密度函数; w_{ij} 是第 j 个分量的权重, 且 $\sum_{j=1}^{M} w_{ij} = 1$; μ_{ij}, υ_{ij} 分别是均值向量和协方差矩阵; M 是高斯混合模型的阶数. $\boldsymbol{w}_{ij}$, μ_{ij}, υ_{ij} 都是待估计的参数, 将它们统一记为 λ

GMM 的训练就是在 X_i 的基础上, 对 GMM 的参数进行估计, 以使其和 X_i 中各向量的分布最相近. 对 GMM 的参数学习的方法通常采用 EM (Expectation Maximization) 方法. 它从给定的初始的 λ开始, 计算 X_i 中每个训练例向量在每一个高斯分布的统计概率, 再用这些统计概率来计算每一个高斯分布的期望值, 然后以这些期望值反过来最大化 GMM 的参数值, 得到相应的参数 λ 重复上面的步骤, 直到收敛为止. EM 算法分为 E 步和 M 步两个主要步骤, 下面以从数据集 X_i 中学习参数为例对该算法进行说明.

E步, 计算:

$$h_j^{(k)}(\boldsymbol{x}_n) = \frac{w_{ij}^{(k)} p_{ij}^{(k)}(\boldsymbol{x}_n)}{\sum_{j=1}^{M} w_{ij}^{(k)} p_{ij}^{(k)}(\boldsymbol{x}_n)}$$

M 步, 更新参数, 分别如下面三个公式:

$$w_{ij}^{(k+1)} = \frac{1}{N} \sum_{n=1}^{N} h_j^{(k)}(\boldsymbol{x}_n)$$

$$\mu_{ij}^{(k+1)} = \frac{\sum_{n=1}^{N} h_j^{(k)}(\boldsymbol{x}_n) \boldsymbol{x}_n}{\sum_{n=1}^{N} h_j^{(k)}(\boldsymbol{x}_n)}$$

$$v_{ij}^{(k+1)} = \frac{\sum_{n=1}^{N} h_j^{(k)}(\boldsymbol{x}_n) [\boldsymbol{x}_n - \mu_{ij}^{(k+1)}] [\boldsymbol{x}_n - \mu_{ij}^{(k+1)}]^T}{\sum_{n=1}^{N} h_j^{(k)}(\boldsymbol{x}_n) \boldsymbol{x}_n}$$

其中, $N = |X_i|$, 表示 X_i 中样本的个数. λ^{k+1} 表示第 K + 1次迭代时模型的参数.

3 基于 GMM 的多过滤器融合

3.1 方法流程

图 1显示了本文方法的主要流程.

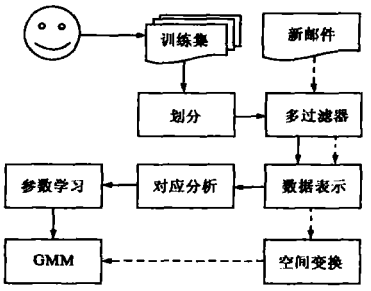

图 1 本文方法的主要流程

图 1中的“划分”指在原训练集中进行 Q (Q > 1)次无放回的抽样, 生成 Q 份训练子集的过程; “多过滤器”指的是在 Q 个不同的训练集上分别训练并得到 Q 个过滤器; “数据表示”指将垃圾邮件表示成向量 $\langle p_1, 1-p_1, p_2, 1-p_2, \cdots, p_Q, 1-p_Q, 1, 0\rangle$, 将非垃圾邮件表示成 $\langle p_1, 1-p_1, p_2, 1-p_2, \cdots, p_Q, 1-p_Q, 0, 1\rangle$, 其中 p_i 是第 i 个过滤器判定当前邮件属于垃圾的后验概率; “对应分析”指的是将所有训练例向量组成训练矩阵, 并对训练矩阵进行对应分析的过程; “参数学习”指的是在对应分析后的训练数据的基础上进行 GMM 参数学习的过程; “空间变换”则指将新邮件向量转换成对应分析后的训练空间中的向量的过程.

可见, 本文方法的基本思路是: 将各过滤器在训练数据上的过滤结果做为新的训练数据. 用这样的训练数据在进行数据降维、噪声清理 (即对应分析)后对 GMM 进行训练. 当给出各过滤器在新邮件上的判定结果后, GMM 即可以判定这样的组合判定结果有多大概率指示新邮件是“垃圾”或“非垃圾”, 结合代价敏感的判定规则, GMM 输出对新邮件的判定类别.

3.2 对应分析及空间变换方法

对每一个训练例进行“数据表示”后, 可以将所有的训练例向量构成一个训练矩阵, 它是一个二维列联表, 以 X 代表该列联表. 显然, X 的行数为训练例的个数, 设为 I; 设 X 的列数为 J (J = 2Q + 2), 通常 I > J. x_{ij} 表示 X 中的第 i 行和第 j列的元素.

首先, 对 X 中每个元素除以所有元素的和, 得到比例矩阵 $\boldsymbol{P}$.

$$\boldsymbol{P}_{(I\times J)} = \frac{1}{\sum_{i=1}^{I}\sum_{j=1}^{J} X_{ij}} X_{(I\times J)}$$

下一步, 对 $\boldsymbol{P}$ 作中心化 (即对每个元素减去行和与列和的乘积), 得到 $\boldsymbol{S}$.

$$\boldsymbol{S}_{(I\times J)} = \boldsymbol{P}_{(I\times J)} - \boldsymbol{b}_{(I\times 1)} \boldsymbol{d}^T_{(J\times 1)}$$

上式中, $\boldsymbol{b}=\boldsymbol{P}\boldsymbol{A}$, $\boldsymbol{d}=\boldsymbol{P}^{\mathrm{T}}\boldsymbol{A}$, $\boldsymbol{A}=[1, 1, \cdots, 1]^{\mathrm{T}}$.

接下来, 构造标准化矩阵 $\boldsymbol{S}^{*}$.

$$\boldsymbol{S}^{*}_{I\times J}=\boldsymbol{B}^{-1/2}_{I\times I}\boldsymbol{S}_{(I\times J)}\boldsymbol{D}^{-1/2}_{J\times J}$$

其中, $B=diag(b_1, b_2, \cdots, b_I)$, b_i $(i=1, 2, \cdots, I)$是向量 $\boldsymbol{b}$ 中的第 i列的元素; $D=diag(d_1, d_2, \cdots, d_J)$, d_j $(j=1, 2, \cdots, J)$是向量 $\boldsymbol{d}$的第 j行的元素.

然后, 对 $\boldsymbol{S}^{*}$ 进行奇异值分解, 可得:

$$\boldsymbol{S}^{*}_{I\times J}=\boldsymbol{U}_{I\times(J-1)}\sum_{(J-1)\times(J-1)}\boldsymbol{V}^{\mathrm{T}}_{(J-1)\times J}$$

其中 $r(\boldsymbol{S}^{*})=r(\boldsymbol{S})<=J-1$(因为 $\boldsymbol{S}=\boldsymbol{P}-\boldsymbol{b}\boldsymbol{d}^{\mathrm{T}}$, 故 $\boldsymbol{S}\boldsymbol{A}=\boldsymbol{P}\boldsymbol{A}-\boldsymbol{b}\boldsymbol{d}^{\mathrm{T}}\boldsymbol{A}=\boldsymbol{b}-\boldsymbol{b}(\boldsymbol{P}^{\mathrm{T}}\boldsymbol{A})^{\mathrm{T}}\boldsymbol{A}=0$, 所以 $r(\boldsymbol{S})<=J-1$); $\boldsymbol{U}^{\mathrm{T}}\boldsymbol{U}=\boldsymbol{V}^{\mathrm{T}}\boldsymbol{V}=\boldsymbol{A}$; $\sum=diag(\lambda_1, \cdots, \lambda_{J-1})$, $\lambda_1\geqslant\lambda_2\geqslant\cdots\geqslant\lambda_{J-1}>0$.

最后, 按下面两个公式分别计算原矩阵 $\boldsymbol{X}$ 的行和列的坐标:

$$\boldsymbol{Y}_{I\times(J-1)}=\boldsymbol{b}^{-1/2}_{I\times I}U_{I\times(J-1)}\sum_{(J-1)\times(J-1)}$$

$$\boldsymbol{Y}_{J\times(J-1)}=\boldsymbol{d}^{1/2}_{J\times J}V_{J\times(J-1)}\sum_{(J-1)\times(J-1)}$$

Y 的第 i行对应 X 中的第 i行, Z 的第 j行对应 X 的第 j列. 令 $K<=r(S)<=J-1$, 取 $\boldsymbol{Y}$ 的前 K 列构成新的矩阵 $\boldsymbol{Y}_K$, 取 $\boldsymbol{Z}$ 的前 K 列构成新矩阵 $\boldsymbol{Z}_K$. $\boldsymbol{Y}_K$ 的第 i行可以看作是对 $\boldsymbol{X}$ 中第 i行的近似; 同理, $\boldsymbol{Z}_K$ 的第 i行可以看作是对 $\boldsymbol{X}$ 第 i列的近似. 做这种近似, 势必会造成信息损失, 信息损失量 (设为 (l)) 可以根据惯性值来定义, 如下式:

$$(l)=1-\frac{\sum_{i=1}^{K}\lambda_i^2}{\sum_{i=1}^{J-1}\lambda_i^2}$$

其中, $\sum_{i=1}^{J-1}\lambda_i^2$ 是总惯性, 而 $\sum_{i=1}^{K}\lambda_i^2$ 是降维后的总惯性.

这样近似的目的是在允许的信息损失量范围内对原数据进行降维, 使降维后的数据能在较大程度上反映原数据的分布. 同时, 降维操作也使噪声数据得以清理. K 值的大小取决于用户定义的信息损失量 ⓗ的大小. 通常 ⓗ越小, K 值越大; ⓗ越大, K 值越小, 本文建议 ⓗ的大小应该 $\leqslant 0.1$.

对于新邮件 E, Q 个过滤器将产生判定 E 为垃圾的 Q 个概率值, 设为 $p_1, p_2, \cdots, p_Q$. 若分别假定 E 为垃圾和非垃圾, 则对 E 可以构造出两个向量, 分别为 $e_0=\langle p_1, 1-p_1, p_2, 1-p_2, \cdots, p_Q, 1-p_Q, 1, 0\rangle$和 $e_1=\langle p_1, 1-p_1, p_2, 1-p_2, \cdots, p_Q, 1-p_Q, 0, 1\rangle$. 为了将 e_i $(i=0, 1)$转化为 K 维空间的点, 首先对 e_i 进行归一化处理, 即: $e_i^{*}=e_i/(Q+1)$, 接下来, 按下式对 e_i^{*} 进行变换.

$$y_i=e_i^{*}\,Z\sum{}^{-1}$$

取 y_i 的前 K 列作为 e_i 的近似, 设为 y_i^{*}. 至此, 即将新邮件映射到了由训练数据形成的 K 维空间中.

3.3 判决方法

算法 **1** 过滤算法

输入: Q 个过滤器, GMM, 新邮件
输出: 新邮件的类别
处理过程:
Q 个过滤器对新邮件进行处理;
IF (Q 个过滤器的判定结果一致) THEN
　返回结果;
对新邮件进行向量表示和空间变换;
用 GMM 结合定理 1过滤新邮件;
输出 GMM 的判定结果.

Y_K 中的数据反映的是各过滤器在训练例上的表现, 采用第 2 节描述的方法, 显然可以在 Y_K 的基础上训练 GMM. 从而, 可以计算出 $p(c_0|\boldsymbol{y}_0^{*}, \lambda_0)$, $p(c_1|\boldsymbol{y}_1, \lambda_1)$.

邮件过滤是个代价敏感的分类问题, 用户希望两种错误对他造成的损失尽可能小, 而不是希望两种错误发生的总次数尽可能少. 也就是说, 过滤判定法则不能依据“期望预测误差最小化”的原则, 即不能简单地通过比较 $p(c_0|\boldsymbol{y}_0^{*}, \lambda_0)$、$p(c_1|\boldsymbol{y}_1^{*}, \lambda_1)$的大小作出判断; 而应该使用“期望错分代价最小化 (ECM)”的准则. ECM 定义如下:

$$ECM=c_{10}p(c_1|c_0)p_0+c_{01}p(c_0|c_1)p_1$$

上式中, c_{10}表示误拒代价; c_{01}表示误收代价; $p(c_0|c_1)$表示将垃圾判定成非垃圾的概率; $p(c_1|c_0)$表示将非垃圾判定成垃圾的概率; p_0, p_1 是先验概率, 分别表示非垃圾训练例和垃圾训练例的比率.

由 ECM 法则, 有以下定理.

定理 **1** 已知邮件向量 $\boldsymbol{x}$, 判定它为非垃圾的条件是:

$$\frac{p(c_0|\boldsymbol{x}, \lambda_0)}{p(c_1|\boldsymbol{x}, \lambda)}\geqslant\frac{c_{01}p_1}{c_{10}p_0}$$

证明:

$$\because p(c_0|c_1)=\int_{X_0}p(c_1|\boldsymbol{x}, \lambda_1)d\boldsymbol{x}$$

$$p(c_1|c_0)=\int_{X_0}p(c_0|\boldsymbol{x}, \lambda_0)d\boldsymbol{x}$$

上两式中, 对条件概率密度函数的积分表示密度函数在训练例区域中形成的体积, 下同. 由于:

$$\int_{X_1}p(c_0|\boldsymbol{x}, \lambda_0)d\boldsymbol{x}+\int_{X_0}p(c_0|\boldsymbol{x}, \lambda_0)d\boldsymbol{x}=1$$

因此, ECM 式可以转化为:

$$ECM=c_{10}p_0[1-\int_{X_0}p(c_0|\boldsymbol{x}, \lambda_0)d\boldsymbol{x}]+c_{01}p_1\int_{X_0}p(c_1|\boldsymbol{x}, \lambda_1)d\boldsymbol{x}$$

$$=c_{10}p_0+\int_{X_0}(c_{01}p_1p(c_1|\boldsymbol{x}, \lambda_1)-c_{10}p_0p(c_0|\boldsymbol{x}, \lambda_0))d\boldsymbol{x}$$

为了使 ECM 最小, 应使 X_0 区域内的所有 $\boldsymbol{x}$ 都满足条件:

$$c_{01}p_1p(c_1|\boldsymbol{x}, \lambda_0)=-c_{10}p_0(c_0|x, \lambda_0)\leqslant 0$$

得证. 结合上面描述的内容, 设计对新邮件过滤的算法, 如算法 1.

4 性能分析与实验结果

4.1 时间复杂度分析

过滤算法的时间复杂度主要要考虑两个方面: 过滤器训练的时间复杂度及对新邮件进行过滤的时间复杂度.

训练的时间主要由以下几个部分组成: (1)Q 个过滤器的训练时间; (2)对应分析中奇异分解的时间; (3)GMM 的训练时间. 对 Q 个过滤器的训练显然可以并发实现, 所以这一部分的时间复杂度可以表示为 $\max_{1\leqslant q\leqslant Q}(O(F_q))$, $O(F_q)$指的是第 q 个过滤器的训练时间复杂度. 对 S^* 进行奇异值分解的时间复杂度为 $O(|V^*|K^2)$. GMM 的训练时间指的是 EM 算法的执行时间, EM 算法的迭代次数无法直接得出, 但不难知道 EM 算法的时间花费与训练例的大小($|X|$)、迭代次数(设为 T)及 GMM 的阶数 M 有关, 故可以记对 GMM 进行训练的时间复杂度为 $O(f(|X|,T,M))$, $f(|X|,T,M)$代表自变量为 $|X|$、T 和 M 的某函数. 因此对本文算法进行训练的时间复杂度为 $O(\max_{1\leqslant q\leqslant Q}+O(|V^*|K^2)+O(f(|X|,T,M))$. 在实验中观察, 得知这一时间通常在几分钟左右. 在实践中, 可以采用 Agent技术, 实现无用户监督下的透明训练过程, 从而, 即使这一时间较长, 用户也感觉不到它的存在.

由算法 1可知, 对于新邮件的过滤, 在最坏情况下, 时间复杂度将由两部分组成: (1)Q 个过滤器的过滤时间; (2)GMM 的过滤时间. 由于 Q 个过滤器互不相干, 相互独立, 故它们对新邮件过滤可以并发执行, 因此, 这一部分的时间复杂度可以表示为 $\max_{1\leqslant q\leqslant Q}(O(F_q^*))$, $O(F_q^*)$指第 q 个过滤器过滤的时间复杂度. GMM 的过滤时间仅仅是公式(1)的计算时间, 不妨计为 O(GMM). 因此过滤的时间复杂度为 $O(\max_{1\leqslant q\leqslant Q}(O(F_q^*))+O(GMM))$, 这似乎比单个过滤器的时间要长, 但由于 O(GMM)代表的时间非常短, 所以本文算法的时间复杂度约等于复杂度最高的那个单个过滤器的时间复杂度. 在实验中观察得知, 本文算法对每封邮件过滤的平均时间仅在 0.1秒左右, 这对于实时过滤需要而言, 是比较合理的. 同样, 采用 Agent技术, 也可以实现无用户干预下的自动地透明过滤.

实际上, 对于垃圾邮件过滤的问题, 用户更关心的是算法的正确性, 即用户希望过滤算法不发生或几乎不发生将误拒错误; 而误收错误应尽量少发生, 否则, 过滤就失去了意义. 至于过滤时间的长短, 若采用基于 Agent技术的过滤, 训练和过滤都可以在用户计算机的 CPU 空闲时进行, 故时间复杂度并不是用户评价算法好坏的主要考虑因素. 因此, 本文的实验主要针对于算法的正确性进行比较.

4.2 评价指标

邮件过滤是个代价敏感的分类问题, 直接使用对分类算法进行评价的指标(查准率(precision)、查全率(recall)、F_1)无法准确地反映出算法的好坏, 因为这些指标仅考虑了错分的个数, 而没有考虑两种不同错误所造成的不同代价. 为了更好地比较代价敏感的过滤算法的好坏, 定义了以下指标.

定义 1 (误拒率, Error Rejecting Ratio(EJR))

$$EJR=\frac{F_{10}}{F_{10}+F_{00}}$$

定义 2 (误收率, Error Accepting Ratio(EAR))

$$EAR=\frac{F_{01}}{F_{01}+F_{11}}$$

定义 3 (错误代价, Error Cost(EC))

$$EC=C_{10}P_0EJR+C_{01}P_1EAR$$

上述定义中, F_{10}表示误拒错误个数; F_{01}则表示误收错误个数; p_0 是非垃圾测试例占总测试例的比例值; p_1 是垃圾测试例占总测试例的比例值.

EJR 反映了过滤器发生误拒错误的频繁程度, 其值越大表示越容易犯这一错误. EAR 则反映了过滤器发生误收错误的频度, 其值越大表示越容易犯误收错误. EC 反映的是犯两类错误造成的总代价, 由于 c_{10}通常比 c_{01}要大, 所以 EC 指标强调了 EJR 对其值的影响, 这样做是为了强调将非垃圾判定为垃圾的错误代价. 同时 EC 也兼顾考虑了 EAR, 也就是说, EC 指标同时考虑了两种错误及它们的代价. 另外, 由于过滤器在测试例数目较多的类别上发生的错误的机会越多, 因此, 在 EC 中考虑了 p_0 和 p_1 这两个值.

实践中, 要求 EJR 和 EAR 的值尽可能接近 0, 因此, EC 的值越小, 算法的性能越好. 根据定理 1, 可知, c_{10}与 c_{01}的比值而不是它们各自的值影响算法的性能, 具体而言, EJR 与 c_{10}/c_{01}的值成正比关系, 而 EAR 与 c_{10}/c_{01}成反比关系. 因此, 为了保证 EJR 的值尽可能小, 应使 c_{10}/c_{01}足够大, 本文建议 c_{10}/c_{01}应该大于 5. 同时, 建议 c_{01}小于 1.

4.3 实验结果

在 PU1和 Ling-spam 两个数据集上, 将本文的方法与 NB, KNN, Rocchio, Vote进行了比较实验.

表 1 PU1数据集上不同特征子集提取率对算法 EC值的影响

过滤器 / 提取率	NB	Rocchio	Vote	KNN	本文算法
0.20%	0.157643	0.187898	0.109873	2.292994	0.030255
0.50%	0.124204	0.133758	0.078025	2.292994	0.035032
1%	0.105096	0.105096	0.082803	0.11465	0.035032
2%	0.068471	0.068471	0.085987	0.143312	0.046178
10%	0.074841	0.367834	0.08121	0.205414	0.041401

表 2 Ling-spam 数据集上不同特征子集提取率对 EC值的影响

过滤器 / 提取率	NB	KNN	Rocchio	Vote	本文算法
0.20%	0.046178	1.515924	0.038217	0.186306	0.033439
0.50%	0.023885	1.515924	0.05414	0.184713	0.030255
1%	0.047325	0.12963	0.055556	0.12963	0.045267
2%	0.035032	1.515924	0.022293	0.046178	0.041401
10%	0.044586	0.207006	0.036624	0.08758	0.051656

PU1包含垃圾 480封, 非垃圾 610封; 垃圾测试例个数为 134封, 用于测试的非垃圾邮件为 180封, 余下的做为训练例. Ling-spam 包含垃圾 481封, 非垃圾邮件 2412封; 分别取 124封垃圾和 119封非垃圾邮件作为测试例, 其余的做为测试例.

特征子集提取算法为 IG 方法, 特征子集提取率为 1%. KNN 算法中, K 的取值为 120 本文算法采用的一些参数为: GMM 的阶数为 6, 信息损失量的阈值为 0.1; Q 的取值为 8, 融合的过滤器均为朴素贝叶斯算法, $c_{10} = 4$, $c_{01} = 0.5$. Vote算法指的是采用 Q 个 NB 进行投票, 每个 NB 的训练方法与本文算法中训练 NB 的方法一样. 图 2和图 3中给出了五种算法在两个数据集上的实验结果. 从图 2可以看出, 在 PU1数据集上, 本文算法的 EJR、EAR 及 EC 值较其它算法都低. 在 Ling-spam 数据集上, 本文算法的 EJR 值接近于 0, 尽管 EAR 比其它算法的要高, 但 EC 值与表现最好的 NB 算法相当, 这正是本文算法所"追求"的目标.

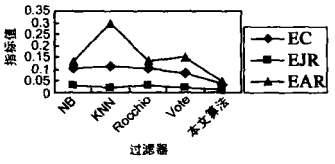

图 2　五种算法在 PU1数据集上的表现

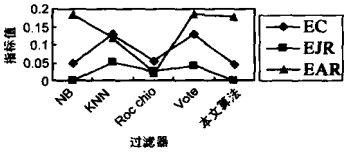

图 3　五种算法在 Ling-spam 数据集上的表现

另外, 还在两个数据集上, 测试了五个算法随特征提取率变化的性能变化情况. 表 1是在 PU1上的的实验结果, 表 2是在 Ling-spam 上的测试结果. 从这两个表可以看出, 一方面本文算法在各个提取率下的 EC 值都较低; 另一方面, 在两个数据集上对于特征提取率的敏感程度都不是很高, 这正是第 1节提到的人们采用多过滤器融合方法的原由之一. NB、Vote、Rochhio 在两个数据集上表现出不同的敏感程度, KNN 则对特征提取率非常敏感.

5　结论

垃圾电子邮件给因特网用户带来的损失越来越巨大, 如何有效地过滤垃圾已成为研究热点. 传统的过滤方法是基于规则或关键字的, 但这类方法并不是十分有效, 因此, 人们将眼光转向了开发基于机器学习的过滤方法. 而单个过滤器往往无法满足用户对过滤精度的要求.

本文采用基于 GMM 的方法融合多个过滤器, 该方法使用多元统计分析方法对多个过滤器在训练例上的过滤表现矩阵进行降维和除噪, 同时得到训练数据及各过滤器的分布; 然后, 从这一分布中学习出对邮件进行类别判定的 GMM.

本文推导了适用于 GMM 判定的期望代价最小的原则, 按该原则进行过滤, 可以尽可能避免发生误拒错误; 同时, 还提出了适用于评价代价敏感的过滤算法的评价指标, 这些指标能很好地反映出算法的性能.

在两个数据集上, 将本文的方法与其它四种方法做了对比实验. 实验结果表明: 本文算法的性能要优于其它算法; 且对于特征提取率取值大小的敏感程度也较低, 从而在实际使用中不需要用户"绞尽脑汁"选择最好的提取率.

参考文献:

[1] Karl Michael Schneider. A comparison of event models for naïve bayes anti-spam e-mail filtering[A]. Proc. 10th Conference of the European Chapter of the Association for Computational Linguistics[C]. Budapest, Hungary, 2003. 307-314.

[2] 熊应, 朱斌, 朱海云. 电子邮件智能分类系统的设计[J]. 电子学报, 2001, 29(12): 1653-1655.
Xiong ying et al. Design of email intelligent classifier[J]. Acta Electronica Sinica, 2001, 29(2): 1653-1655.

[3] Christian Siefkes, et al. Combining winnow and orthogonal sparse bigrams for incremental spam filtering[A]. Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2004)[C]. 2004. 410-421.

[4] 孙怀江, 胡钟山, 杨静宇. 基于证据理论的多分类器融合方法研究[J]. 计算机学报, 2001, 24(3): 231-235.

[5] Bauer E, Kohavi R. An empirical comparison of voting classification algorithms: bagging, boosting, and variants[J]. Machine Learning, 1999, 36(1-2): 105-139.

[6] D Reynolds, R Rose. Robust text-independent speaker identification using Gaussian mixture speaker models[J]. IEEE Trans Speech and Audio Proc, 1995, 3(1): 72-83.

[7] A K Jain, R P W Duin, J Mao. Statistical pattern recognition: a review[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2000, 22(1): 4-37.

作者简介:

李文斌　男, 1974年 10月生于江西南昌县, 北京工业大学多媒体与智能软件北京重点实验室博士研究生, 石家庄经济学院信息工程学院讲师, 主要研究方向为: 网上智能、数据挖掘等. E-mail: mr_liwb@emails.bjpu.edu.cn.

刘椿年　男, 1944年 3月生于江苏连云港市, 博士生导师, 其主要研究方向包括: 逻辑程序设计 (约束逻辑程序设计 CLP, 归纳逻辑程序设计 ILP, 约束归纳逻辑程序设计 CILP); 机器学习 (数据挖掘与知识发现 KDD, KDD 进程); 软计算等.