遗传顺序IB算法

袁华强[1],叶阳东[2],刘 东[2]
(1.东莞理工学院工程技术学院,广东东莞 523808; 2.郑州大学信息工程学院,河南郑州 450052)

摘 要: 本文提出一种遗传顺序IB算法,该算法以基本顺序IB算法的多次运行结果作为初始种群,并基于集成操作算子将初始种群组合为一个解;然后算法分别计算解中每个元素的不确定性统计量,对解中元素进行选择和变异,最后经过若干代变异后得到优化的解.在数据集上的实验结果表明,相对于顺序IB算法,遗传顺序IB算法具有运行效率高、解更优化的特点.

关键词: IB理论; sIB算法; 遗传变异; 互信息
中图分类号: TP391 **文献标识码:** A **文章编号:** 0372-2112 (2009) 08-1804-06

Genetic Sequential IB Algorithm

YUAN Hua-qiang[1], YE Yang-dong[2], LIU Dong[2]
(1. *Institute of Engineering Technology, Dongguan University of Technology, Dongguan, Guangdong* 523808, *China*;
2. *School of Information Engineering, Zhengzhou University. Zhengzhou, Henan* 450052, *China*)

Abstract: This paper proposes a genetic sequential IB algorithm. It takes several seeding solutions of the basic sequential IB algorithm as initial population, and then integrates this population into a solution using the integration operator. Sequentially, some certain positions of the obtained solution are selected and mutated iteratively based on the defined instability statistic. After mutation of several generations, the iterative process terminates and a more optimal solution is obtained. Experimental results on the benchmark data sets indicate that the proposed algorithm outperforms the sequential IB algorithm in both the accuracy and the efficiency.

Key words: IB theory; sIB algorithm; genetic variation; mutual information

1 引言

Tishby等人于1999年提出了一种基于信息论的数据分析方法—IB方法(Information Bottleneck Method)[1].在数据分析过程中,通过将数据对象压缩到一个事先定义好的"瓶颈"变量中并极大地保持其与另一数据对象的关联性,IB方法有效地发现了数据对象间的相关模式,并在文本聚类[2,3]、图像聚类[4]、星系光谱分析[5]、基因表达式分析[6]、神经系统编码分析[7]、自然语言处理[8]等领域得到应用.2002年,Slonim对IB方法进行归纳总结并将其命名为IB理论[9].近年来IB理论不断发展,相继出现了aIB[10]、dIB[9]和sIB[2]等多种IB算法.

相对于其他IB算法,sIB算法解决了精确度和复杂度之间的平衡问题,但该算法对于随机的初始解选取可能导致不同结果,且容易陷入局部解.Slonim等人从多次随机初始化基本sIB算法的解中选取最优解作为输出,一定程度上解决了解的局部优化问题[2],但该方法仍存在随机性强、优化不充分等问题.2004年,Peltonen等人将IB算法的目标函数改造为贝叶斯因子形式,提出一种基于sIB的fsIB算法[11],该算法特别适宜小规模稀疏数据集,但它仍然不能有效解决sIB算法所面临的问题.本文针对sIB算法存在的不足,提出了遗传顺序IB算法(GsIB).在得到一个合理的变异率基础上,算法以基本sIB算法多次运行结果作为初始种群,并使用集成算子将初始种群组合为一个解.通过计算解中每个元素的不确定性统计量,根据变异率的取值对解中的元素分别进行选择和变异;最后经过若干代迭代优化过程后,得到比sIB算法更有效更精确的解.实验结果表明,与sIB算法相比,GsIB算法不仅得到了更优解,而且效率也得到了明显提高.

2 IB理论

IB理论通过定义源变量 X 的相关变量 Y,推导出一个合理的失真度量函数:

$$d(x,t)=D_{KL}(p(y|x)\|p(y|t)) \tag{1}$$

其中 $D_{KL}(p\|q)$ 是概率分布 p 和 q 之间的Kullback Leibler(KL)距离.

由于IB理论中的变量 X, T 和 Y 形成Markov链[1,9],

收稿日期:2008-07-23;修回日期:2008-12-01
基金项目:国家自然科学基金(No.60573029, No.60773048, No.60773050)

因此 $p(x,t,y)=p(x,t)p(y|x,t)=p(x,t)p(y|x)$. 可利用这一关系对式(1)的期望失真 $E(d(x,t))$进行推导得到:

$$E(d(x,t))=I(X;Y)-I(T;Y) \tag{2}$$

由于式(2)中 $I(X;Y)$是一个常数,因此得到 IB 理论的形式化表示:

$$R(D)=\min_{\{p(t|x):I(T;Y)\geqslant D'\}} I(X;T) \tag{3}$$

该式的求解可采用拉格朗日乘数法,即最小化 IB 目标函数,

$$F_{\min}(p(t|x))=I(X;T)-\beta I(T;Y) \tag{4}$$

该目标函数的形式解[1]:

$$p(t|x)=\frac{p(t)}{Z(x,\beta)}\exp(-\beta D_{KL}(p(y|x)\|p(y|t))) \tag{5}$$

$$p(y|t)=\frac{1}{p(t)}\sum_{x\in X}p(y|x)p(t|x)p(x) \tag{6}$$

$$p(t)=\sum_{x\in X}p(t|x)p(x) \tag{7}$$

其中 $Z(x,\beta)$是一个概率归一化函数. 式(5),(6),(7)描述了使用 IB 理论对源数据 X 进行划分的结果.

3　遗传顺序 IB 算法——GsIB

3.1　GsIB 算法描述

GsIB 算法包含三种操作算子:集成算子、选择算子和变异算子.

首先,集成算子对数据集的多个不同划分结果进行合并,这样算法以较低代价获得了一个组合优化解. 由于多次调用 sIB 算法可以获得相似的数据划分结果,因此可以针对划分结果中的相异元素,对相应的类标记进行统一. 在统一类标记之后,GsIB 算法通过投票策略产生一个解. 在该过程中,算法分别记录每个元素的不确定性统计量. 例如,针对以下三个等长度的相似解,

$s1=[1,1,2,2,2,2,3,3,3]$

$s2=[2,2,1,3,3,3,1,1,1]$

$s3=[3,3,3,1,1,1,1,2,2]$

参照 $s1$,可以将这三个相似解的类标记统一如下:

$s1'=[1,1,2,2,2,2,3,3,3]$

$s2'=[1,1,3,2,2,2,3,3,3]$

$s3'=[1,1,1,2,2,2,2,3,3]$

统一类标记之后,在第七个元素位置上类标记 3 出现两次,而类标记 2 仅出现一次,根据投票策略将该位置类标记记为 3. 而在第三个元素位置上类标记 1、2、3 等概率出现,这种情况下投票策略不再有效. 集成算子将以上三个解进行合并生成以下组合解:

$s=[1,1,?,2,2,2,3,3,3]$

其中,第三个元素"?"表示该位置元素的类标记不确定,从 $s1'$,$s2'$,$s3'$可以看出第三个和第七个位置元素的类标记都是不确定的.

我们引入不确定性统计量 Instability. 假设 GsIB 算法针对数据集 X 生成 Psize 个不同的聚类结果,对于数据集中的任意元素 x_i,用 Count(x_i,c_j)记录在 Psize 个结果中元素 x_i 被指派到簇 c_j 的次数,用 Max(Count(x_i,:))表示元素 x_i 映射到所有簇的最大次数,相应的类标记为 c. 则可按下式计算 x_i 的最终划分簇和它对应的不确定性统计量值.

Solution(x_i) = c

Instability(x_i) = Psize-Max(Count(x_i,:))

根据定义,则上述解的不确定性统计量 Instability 为[0,0,2,0,0,0,1,0,0]. 从解的 Instability 值可以看出,一个元素对应簇标记的不确定性越强,其对应的 Instability 值越大.

确定了解的 Instability 之后,选择算子根据变异率 Rate 的取值和解中各元素 Instability 值的大小顺序,选择 Rate * |X|个不确定元素进行优化处理. 元素的不确定性越强,其被选择优化的可能性就越大. 选择的不确定元素数目由变异率 Rate 的取值确定,本文通过实验获得了一个合理的变异率取值.

针对选择算子选出的 Rate * |X|个元素,使用变异算子对选择的元素进行类标记的变异,进一步对解进行优化. 首先生成一个从数据元素集合$\{x_1,x_2,\cdots,x_{|X|}\}$到簇集合$\{t_1,t_2,\cdots,t_{|T|}\}$的多对一的映射:

$$\{x_i\to t_j\,|\,x_i\in X,t_j\in T,i\in\{1,2,\cdots,|X|\},j\in\{1,2,\cdots,|T|\}\}$$

该映射反应了将数据 X 进行"硬"划分的结果,所有被映射为同一个 t 的元素被认为划分到簇 t 中,最终结果包含了|T|个簇. 在该映射的基础上,我们应用变异算子,针对特定的变异率 Rate,从上述映射中随机选取 Rate * |X|个映射,将其中每个映射 $x_i\to t_j$ 随机地修改为 $x_i\to t_k$,$k\in\{1,2,\cdots,|T|\}$,以此作为当前一代 sIB 算法的解;然后用当前迭代结果初始化基本 sIB 算法,再调用 sIB 算法对该解进行优化.

根据上述步骤,选择算子与变异算子相互作用. 通过多次迭代,更多的元素被映射到相同的簇,不确定元素趋于统一,最终得到了相应的优化解.

3.2　GsIB 算法

GsIB 算法首先对基本 sIB 算法进行 Psize 次重复调用,并将得到的 Psize 个聚类结果作为初始种群;然后使用集成算子生成一个解,并分别计算相应元素的不确定性统计量;基于不确定统计量和变异率,使用选择算子随机选择解中若干元素,并使用变异算子改变相应元素的类标记,最后调用 sIB 算法对解进行优化以产生

新一代种群.算法最终输出进化过程中的最优解.

GsIB 算法

输入:联合分布,$p(x,y)$;平衡参数,β;聚类个数,C;种群规模,Psize;进化代数,n;变异率,$Rate$.

输出:X 到 T 的一个映射;评估值,$I(T;Y)$.

算法步骤:

(1)调用基本 sIB 算法 $Psize$ 次,将得到的初始种群记做 $SolutionMatrix$,且将对应的多个 $I(T;Y)$值组成一个向量 Ity_Vec;

(2)for $i=1:n$

(3)统一 $SolutionMatrix$ 的类标记,计算其中各类标记的不确定性统计量 $Instability$;

(4)使用集成算子从种群 $SolutionMatrix$ 中产生 1 个集成解;

(5)for $j=1:Psize$

(6)基于不确定性统计量 $Instability$ 和变异率 $Rate$,由选择算子随机选择 pos 个元素;

(7)由变异算子对 pos 个元素进行变异得到新解,并用它初始化 sIB 算法,得到新的解 $New\_Solution$ 和相应的估值序列 New_Ity;

(8)if $New\_Ity \geqslant Ity\_Vec(j)$

(9)$Ity\_Vec(j)=New\_Ity$;

(10)$SolutionMatrix(j,:)=New\_Solution$;

(11)返回 $NewPt\_x$ 和 Ity_Vec;

该算法对收敛的 sIB 算法的初始解空间进行局部的优化处理,整个过程相当于在深度优先的基础上进行局部的广度优先搜索,这有效拓展了搜索空间,最终收敛到更优的解.

4 实验

4.1 评估标准

本文和文献[2,9]一样,在固定聚类个数的条件下,通过文本聚类实验对 GsIB 算法和 sIB 算法进行对比.对于目标函数(8),只要最大化相关信息就可以得到满意解.Slonim 等人把 sIB 算法中的参数 β 设置为 ∞,则目标函数 $F_{max}=I(T;Y)-\beta^{-1}I(X;T)=I(T;Y)$.与此类似,实验中我们把 GsIB 算法的相应参数也设置为 ∞.Slonim 等人验证了目标函数 $I(T;Y)$的变化趋势与文本分类中精确率(precision)和召回率(recall)组成的 F-度量变化趋势一致,所以本文仍采用它作为评估标准,即 $I(T;Y)$值越大,聚类精度越高.由于该评估标准不需要类标号的先验知识,因此是一个较好的无监督学习度量标准.为了测试算法的稳定性,本文分别对比了 GsIB 算法和 sIB 算法在每个数据集上 10 次实验所得到的 $I(T;Y)$值和运行时间的平均值.

4.2 数据集

本文使用表 1 中的数据集进行实验,实验之前首先根据文献[2]中的方法对数据集进行预处理.每个数据集由 500 篇从一些话题中随机选出的文档组成.对于每个数据集 $M(x,y)$,任意 $x\in X$ 代表一篇文档,任意 $y\in Y$ 表示对应文档中的一个单词,$M(x,y)$表示单词 y 在文档 x 中出现的次数.这些数据集都是稀疏的文档-单词计数矩阵,$|X|=500$ 行,$|Y|=2000$ 列.

表 1 数据集

数据集	聚类个数 C	相关话题	每个话题的文档数	文档总数
Binary_1,2,3	2	talk.politics.mideast, talk.politics.misc	250	500
Multi5_1,2,3	5	comp.graphics, rec.motorcycles, rec.sport.baseball, sci.space, talk.politics.mideast	100	500
Multi10_1,2,3	10	alt.atheism, comp.sys.mac.hardware, misc.forsale, rec.autos, rec.sport.hockey, sci.crypt, sci.electronics, sci.med, sci.space, talk.politics.guns	50	500

上述数据集是根据话题个数进行命名的;聚类个数 C 的确定也是如此.对于 IB 算法的输入,通过对 $M(x,y)$进行标准化处理可得到联合分布 $p(x,y)$.

4.3 变异率

变异率对 GsIB 算法性能有重要影响,它如同一个平衡参数,其值越大越有利于发现更优解,但算法运行时间会随之增加;相反,如果变异率取值过小,算法的运行效率将会提高,但可能陷入局部解.合理的变异率取值可以有效平衡算法运行时间和聚类精度之间的矛盾.

图 1 中两坐标的 x 轴都表示变异率 Rate 的不同取值,y 轴分别表示 GsIB 算法在每个数据集上的运行时间和 $I(T;Y)$值.由上图可以看出,随着变异率的增加,GsIB 算法的运行时间逐渐增加,而互信息 $I(T;Y)$的值先增加然后逐渐减小.综合分析可得变异率的最佳取值为 0.2.我们在本文实验中将设置 GsIB 算法的变异率为 0.2.

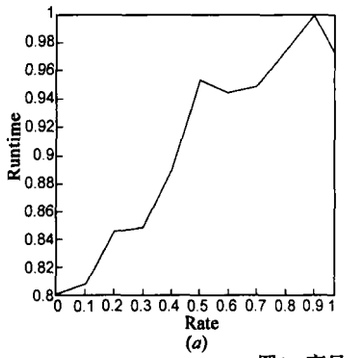
4.4 时间对比

图 2 中坐标 x 轴依次表示 Binary_1,2,3,Multi5_1,2,3 和 Multi10_1,2,3 等 9 个数据集,y 轴表示算法的运行时间.每条曲线上的 9 个数据点分别对应 sIB 算法或 GsIB 算法在相应数据集上的运行时间.由图 2 可以看出,在所有数据集上 GsIB 算法的运行时间均低于 sIB 算法,并且随着聚类个数的增加,两算法的运行时间都呈上升趋势.进一步分析可知,GsIB 算法在 9 个数据集上的运行时间总和为 sIB 算法的 81.40%,这说明 GsIB 算法的总体运行效率高于 sIB 算法.

4.5 精度对比

实验中评估函数 $I(T;Y)$在 0 至 1 之间取值,因此通过 $I(T;Y)$值对两算法精度进行对比难以显示性能差异.我们通过 $\Delta I(T;Y)$来对两算法进行比较(图 3),$\Delta I(T;Y)$由两算法对应的 $I(T;Y)$分别减去 sIB 算法的

$I(T;Y)$得到.在对比互信息之后,我们又通过配对 t 测试进一步比较两个算法的性能差异(图 4).最后从聚类的角度比较了两算法的精确度和召回率.



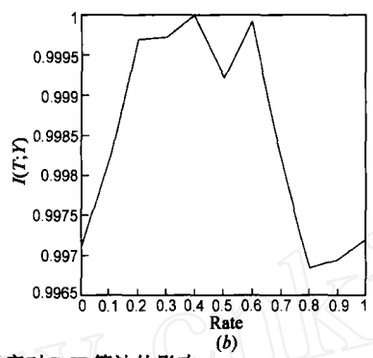


图1　变异率对GsIB算法的影响

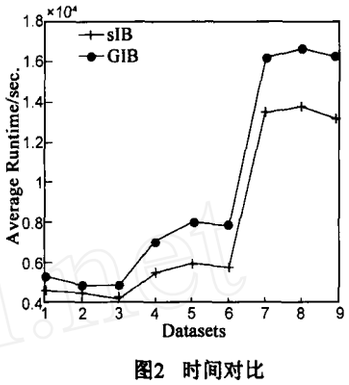


图2　时间对比

4.5.1　直接对比

图 3 中坐标 x 轴依次表示 9 个数据集,顺序和图 2 一致;y 轴表示 $\Delta I(T;Y)$.由于 $\Delta I(T;Y)$由两算法对应的 $I(T;Y)$分别减去 sIB 算法的 $I(T;Y)$得到,因此 sIB 算法对应的 $\Delta I(T;Y)$值为 0,在图 3 中 sIB 算法对应的结果为一条水平直线.从图 3 可知,在 9 个数据集中的其中 8 个上面,GsIB 算法的聚类精度优于 sIB 算法;且随着聚类个数的增加,GsIB 算法的精度优势更为显著.

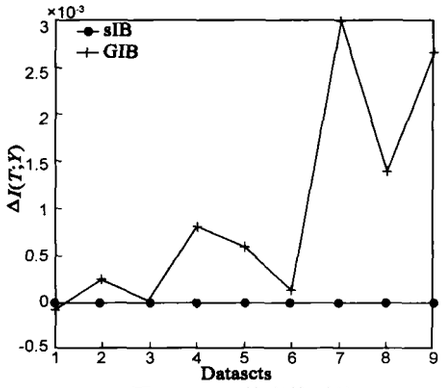


图3　$I(T;Y)$的直接对比

4.5.2　配对 t 测试上的对比

配对 t 测试用以说明两组数据是否相似.如两组数据 X 和 Y 是相似的,则它们间的随机差 $d=X-Y$ 服从正态分布,且均值为 0.检验假设为 $H_0: u_d=0, H_1: u_d\neq 0$.分别把 d 的样本均值和样本方差记作 $\bar{d}$ 和 s^2,则拒绝域为

$$|t|=\left|\frac{\bar{d}-0}{s/\sqrt{n}}\right|\geqslant t_{\alpha/2}(n-1)$$

其中,样本容量 $n=10$,对应算法在每个数据集上的 10 次实验.令显著性水平 $\alpha=0.05$,则 $t_{\alpha/2}(9)=t_{0.025}(9)=2.2622$,拒绝域为 $|t|=\left|\frac{\bar{d}-0}{s/\sqrt{n}}\right|\geqslant 2.2622$.若配对 t 测试结果落在拒绝域中,则说明两组数据相异.

本文用配对 t 测试衡量两个算法在每个数据集上 10 次实验得到的 $I(T;Y)$值形成的向量 X 和 Y 的差异.将测试结果 t 与拒绝域边界 2.2622 进行对比:(1)如果 $t\geqslant 2.2622$,则说明 X 明显优于 Y;如果 $0<t<2.2622$,则说明 X 比 Y 较好;(2)如果 $-2.2622<t<0$,则说明 Y 比 X 较好;如果 $t\leqslant -2.2622$,则说明 Y 明显优于 X.

表 2 和图 4 列出了配对 t 测试结果.从表 2 中可以看出,t 测试值被 0 和 2.2622 分割在 3 个区域中:$S_1\geqslant 2.2622$,表示 GsIB 算法对应的 $I(T;Y)$值有显著优势;$0\leqslant S_2<2.2622$,表示 GsIB 算法对应的 $I(T;Y)$值比 sIB 算法优越,但结果相似;$-1<S_3<0$,表示 sIB 算法对应的 $I(T;Y)$值比 GsIB 算法优越,但结果也相似.

对表 2 中的数据进行统计可知,在 9 个测试结果中,分别有 2 个和 6 个在 S_1 和 S_2 区,只有 1 个在 S_3 区.这说明,GsIB 算法的多数结果(89%)优于 sIB 算法;另外从图 4 可以看出,随着聚类个数的增加,配对 t 测试结果不断增大,GsIB 算法的优势也更加显著.

表 2　数据集上的 t 测试结果

数据集	Binary_1	Binary_2	Binary_3	Multi5_1	Multi5_2	Multi5_3	Multi10_1	Multi10_2	Multi10_3
t 测试值	-1.0569	0.6185	0.1374	2.0712	2.1590	1.6077	2.8215	1.1091	2.7537

表 3　两种算法的精确度对比

数据	Binary_1	Binary_2	Binary_3	Multi5_1	Multi5_2	Multi5_3	Multi10_1	Multi10_2	Multi10_3
GsIB	91.9	91.3	93.3	90.2	89.1	94.2	71.4	64.6	68.5
sIB	91.4	89.2	93.0	89.4	89.4	94.2	70.2	63.8	67.0

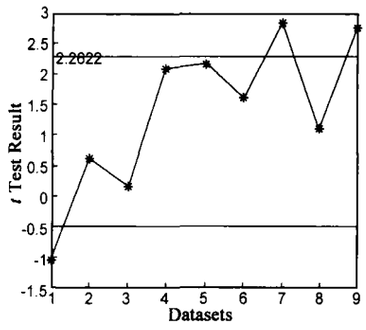
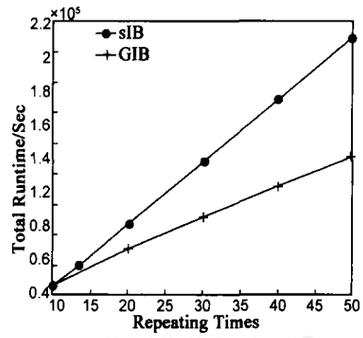
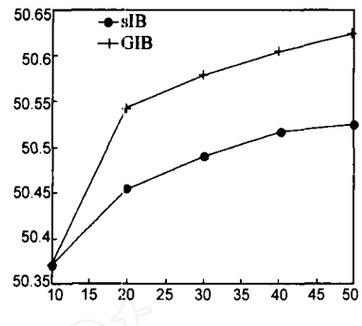


图4 配对t测试结果



(a) 迭代次数对运行时间的影响



(b) 迭代次数的算法精度的影响

图5 迭代次数的影响

4.5.3 精确度和召回率对比

精确度和召回率常用来作为聚类精度的评价准则.在已知聚类结果 T 的条件下,假设 $\alpha(c,T)$表示被正确归类到类 c 的元素数目,$\beta(c,T)$表示被错误归类到类 c 的元素数目,$\gamma(c,T)$表示本来应归类为 c 而未正确归类的元素数目.对于单类标记的数据集,精确度 $P(T)$和召回率 $R(T)$的值相等,因此本文只给出了精确度的对比.表 3 分别列出了两种算法在 9 个数据集上的精确度.由表 3 可知,GsIB 算法在文本聚类的精确度和召回率上总体优于 sIB 算法.

4.6 迭代次数分析

增大基本 sIB 算法的调用次数,在此基础上进一步比较两种算法的运行时间和聚类精度.图 5(a)显示了基本 sIB 算法调用次数分别为 10、20、30、40 时两种算法的运行时间.由图中可以看出,在调用次数相同的条件下,GsIB 算法的运行时间明显小于 sIB 算法.

图 5(b)显示了两种算法随着基本 sIB 算法迭代次数的增加,$I(T;Y)$的变化趋势.由图中可以看出,当迭代次数为 20 时,$I(T;Y)$的变化最为显著;且在迭代次数相同的条件下,GsIB 算法的 $I(T;Y)$明显大于 sIB 算法.这说明 GsIB 算法在精度上优于 sIB 算法,并在迭代次数为 20 时两算法获得满意解,而继续增大迭代次数对算法精度提升不大,这验证了两种算法的收敛性.

5 结论

本文针对 sIB 算法容易陷入局部解以及优化不充分等问题,提出了一种 GsIB 算法.该算法在保留基本 sIB 算法优点的同时,通过遗传变异思想有效拓展了搜索空间.算法在发现一个合理变异率的基础上,从基本 sIB 算法产生的初始解中随机选取部分元素,并对其类标号进行随机变异并优化,最终得到了相应的优化解.实验结果表明,与 sIB 算法相比,GsIB 算法运行效率较高且解更加优化.本文主要贡献可以概括如下:(1)将遗传变异思想引入到 IB 算法的研究中,提高了 sIB 算法的运行效率;(2)提出的 GsIB 算法得到了更优的聚类结果,提高了原有 sIB 算法的聚类精度.

GsIB 算法在实际应用中的有效性是我们下一阶段将要研究的问题之一.

致谢:在 IB 理论的研究过程中,美国 Princeton 大学的 Noam Slonim 博士和澳大利亚 Deakin 大学的 Gang Li 博士为我们提供了很多帮助,在此对他们表示感谢.

参考文献:

[1] N Tishby, F Pereira, W Bialek. The information bottleneck method[A]. Proc of 37th Allerton Conference on Communication, Control and Computing[C]. Monticello, Illinois: Curran Associates, Inc., 1999. 368 - 377.

[2] N Slonim, N Friedman, N Tishby. Unsupervised document classification using sequential information maximization[A]. Proc of the 25th Ann Int ACM SIGIR Conf on Research and Development in Information Retrieval[C]. Tampere, Finland: ACM, 2002. 129 - 136.

[3] N Slonim, N Tishby. The Power of Word Clusters for Text Classification[A]. Proc of 23rd European Colloquium on Information Retrieval Research[C]. Darmstadt: Springer Verlag, 2001. 1 - 12.

[4] S Gordon, H Greenspan, J Goldberger. Applying the Information Bottleneck Principle to Unsupervised Clustering of Discrete and Continuous Image Representations[A]. Proc of the 9th International Conference on Computer Vision[C]. Nice, France: IEEE Computer Society, 2003. 370 - 377.

[5] N Slonim, R Somerville, N Tishby, et al. Objective classification of galaxies spectra using the information bottleneck method [J]. Monthly Notices of the Royal Astronomical Society, 2001 (323): 270 - 284.

[6] N Tishby, N Slonim. Data clustering by Markovian relaxation and the information bottleneck method[A]. Proc of Advances in Neural Information Processing Systems[C]. Vancouver: MIT Press, 2001. 640 - 646.

[7] E Schneidman, W Bialek, M J Berry. An information theoretic approach to the functional classification of neurons[A]. Proc

Advances in Neural Information Processing Systems[C]. Vancouver: MIT Press, 2002. 197 - 204.

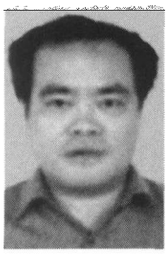
[8] M Gorodetsky. Methods for discovering semantic relations between words based on co-occurrence patterns in corpora[D]. Jerusalem: Hebrew university, Israel, 2002.

[9] N Slonim. The information bottleneck: Theory and Application [D]. Jerusalem: Hebrew University, Israel, 2002.

[10] N Slonim, N Tishby. Agglomerative information bottleneck [A]. Proc of Advances in Neural Information Processing Systems[C]. Denver: MIT Press, 1999. 617 - 623.

[11] Jaakko Peltonen, Janne Sinkkonen, Samuel Kaski. Sequential information bottleneck for finite data[A]. Proc of 21st International Conference on Machine Learning[C]. Alberta, Canada: ACM, 2004. 647 - 654.

作者简介:



袁华强　男,1966 年生于湖南衡东.博士、教授,研究方向为机器学习、生物特征识别.
E-mail: hyuan66@163.com.

叶阳东　男,1962 年生于河南潢川.工学博士、教授、博士生导师,研究方向为知识工程、机器学习、数据库.