基于样本密度和分类误差率的增量学习矢量量化算法研究

李 娟 [1, 2] 王宇平 [1]

摘 要 作为一种简单而成熟的分类方法, K 最近邻 (K nearest neighbor, KNN) 算法在数据挖掘、模式识别等领域获得了广泛的应用, 但仍存在计算量大、高空间消耗、运行时间长等问题. 针对这些问题, 本文在增量学习型矢量量化 (Incremental learning vector quantization, ILVQ) 的单层竞争学习基础上, 融合样本密度和分类误差率的邻域思想, 提出了一种新的增量学习型矢量量化方法, 通过竞争学习策略对代表点邻域实现自适应增删、合并、分裂等操作, 快速获取原始数据集的原型集, 进而在保障分类精度基础上, 达到对大规模数据的高压缩效应. 此外, 对传统近邻分类算法进行了改进, 将原型近邻集的样本密度和分类误差率纳入到近邻判决准则中. 所提出算法通过单遍扫描学习训练集可快速生成有效的代表原型集, 具有较好的通用性. 实验结果表明, 该方法同其他算法相比较, 不仅可以保持甚至提高分类的准确性和压缩比, 且具有快速分类的优势.

关键词 学习矢量量化, 增量学习, 分类误差率, 样本密度, 合并, 分裂

引用格式 李娟, 王宇平. 基于样本密度和分类误差率的增量学习矢量量化算法研究. 自动化学报, 2015, **41**(6): 1187−1200

DOI 10.16383/j.aas.2015.c140311

An Incremental Learning Vector Quantization Algorithm Based on Pattern Density and Classification Error Ratio

LI Juan[1, 2] WANG Yu-Ping[1]

Abstract As a simple and mature classification method, the K nearest neighbor algorithm (KNN) has been widely applied to many fields such as data mining, pattern recognition, etc. However, it faces serious challenges such as huge computation load, high memory consumption and intolerable runtime burden when the processed dataset is large. To deal with the above problems, based on the single-layer competitive learning of the incremental learning vector quantization (ILVQ) network, we propose a new incremental learning vector quantization method that merges together pattern density and classification error rate. By adopting a series of new competitive learning strategies, the proposed method can obtain an incremental prototype set from the original training set quickly by learning, inserting, merging, splitting and deleting these representative points adaptively. The proposed method can achieve a higher reduction efficiency while guaranteeing a higher classification accuracy synchronously for large-scale dataset. In addition, we improve the classical nearest neighbor classification algorithm by absorbing pattern density and classification error ratio of the final prototype neighborhood set into the classification decision criteria. The proposed method can generate an effective representative prototype set after learning the training dataset by a single pass scan, and hence has a strong generality. Experimental results show that the method not only can maintain and even improve the classification accuracy and reduction ratio, but also has the advantage of rapid prototype acquisition and classification over its counterpart algorithms.

Key words Learning vector quantification, incremental learning, classification error ratio, pattern density, merge, split

Citation Li Juan, Wang Yu-Ping. An incremental learning vector quantization algorithm based on pattern density and classification error ratio. *Acta Automatica Sinica*, 2015, **41**(6): 1187−1200

K 最近邻 (K nearest neighbor, KNN) 算法是一种成熟而简单的分类算法, 由 Cover 和 Hart[1] 最初提出, 并在数据挖掘、模式识别等诸多领域获得广泛而有效的应用. KNN 算法具有理论简单、易于实现、无需预先训练分类器等优势. 然而因无需预先训练分类器的缘故, 导致产生难以接受的时空消耗, 尤其在处理大规模数据集时. 为此在分类算法中, 如何对大规模数据集去除冗余节点, 保留高效分类贡献的代表点, 进而降低数据规模、提高分类速度, 成为了研究热点. 而基于原型的算法则是解决上述问题的有效途径之一, 其主要通过获取能够反映原始数据集分布及分类特性的原型代表集, 从而实现降低数据量、保持较好分类精度、提高算法分类速度的目标.

原型算法[2] 目标为在不降低分类性能的基础上,

收稿日期 2014-05-08 录用日期 2014-09-27
Manuscript received May 8, 2014; accepted September 27, 2014
国家自然科学基金 (61203372, 61472297) 资助
Supported by National Natural Science Foundation of China (61203372, 61472297)
本文责任编委 封举富
Recommended by Associate Editor FENG Ju-Fu
1. 西安电子科技大学计算机学院 西安 710071 2. 陕西师范大学远程教育学院 西安 710062
1. School of Computer Science and Technology, Xidian University, Xi′an 710071 2. School of Distance Education, Shaanxi Normal University, Xi′an 710062

降低训练集规模和去除噪音, 进而提高算法执行效率. 其模型常见为: 设 TR 为训练集 (包含一些无用信息, 如噪音、冗余信息等), 寻求选择子集 $TP \subset TR$ 或生成较原始数据集规模小的集合 TP, 使得 TP 不包含多余原型, 且 $Acc(TP) \cong Acc(TR)$ ($Acc(X)$ 表示 X 作为训练集所获取到的分类精度).

目前, 原型算法已经获得了飞速的发展, 产生了诸多的研究成果. 其中就原型获取策略而言, 分为三类: 1) 选择法, 即从原始训练集中直接选择出一组具有代表性的点构成一个新子集, 经典算法有 CNN (Condensed nearest neighbor)[3]、FCNN (Fast condensed nearest neighbor)[4]、ENN (Edited nearest neighbor)[5] 及 DROPn (Decremental reduction optimization procedure n)[6] 等. 其中 CNN 及其改进算法 FCNN 算法等, 虽然简单、易实现, 可用于大规模数据约简问题, 但存在着对噪音节点、样本读取序列敏感的弊端; 而 ENN 及 DROPn, 虽能较好克服前者的敏感性弊端, 但算法的时空复杂度较大, 不适应大规模数据约简处理. 2) 生成法, 即通过融合原始数据集中近邻点而重新生成一组具有代表性点形成的新代表点集, 常见算法有采取不断合成最近邻同类样本而逐步缩减样本规模的 PNN (Prototypes for nearest neighbor)[7]、通过合成核心原型而构造若干分离超平面从而克服样本读取序列敏感性的 POC-NN (Pairwise opposite class-nearest neighbor)[8] 等. 3) 集成法, 即融合聚类算法、遗传算法等生成的新原型算法, 如 LVQ3 (Learning vector quantification 3)[9]、TRKNN (Template reduction approach for the k-nearest neighbor)[10] 等. 就原型集压缩策略而言, 也主要分为三类: 1) 降量法, 即以原始数据集为处理对象, 采用不同策略不断缩小数据集规模, 如基于聚类进行数据集划分从而选取边界样本以及生成类内部节点的快速聚类原型生成算法 PSC (Prototype selection by clustering)[11] 和基于竞争策略而不断调整权重参数的矢量量化算法家族[12] 等, 但存在前述的 ENN 及 DROPn 等算法的不足. 2) 增量法[13], 即用部分原始数据构建基本分类器模型, 训练后, 只保存训练好的模型, 而将原始样本抛弃, 并不断学习其他原始数据, 进而得到原型集, 目前已有算法如基于增量支持向量机 (Support vector machine, SVM) 的原型算法[14]、增量聚类原型算法[15−16]、增量学习矢量量化 (Incremental learning vector quantification, ILVQ)[17] 等, 是当前处理大规模数据约简的主要思路. 3) 混合法, 是增量与非增量方式相融合, 并纳入 GA (Genetic algorithm)、PSO (Particle swarm optimization) 以及其他策略而生成的集成算法, 如 WLVQ (Weighted learning vector quantification)[18] 等, 是目前数据约简的研究热点.

本文结构如下: 第 1 节简要介绍相关工作; 第 2 节提出基于样本密度和分类误差率的增量学习矢量量化算法 (Improved learning vector quantification, IMLVQ), 并给出相应的伪代码; 第 3 节通过理论分析和若干具体实验与其他原型算法性能进行比较, 并重点对比分析 IMLVQ 算法的优劣性能以及增量学习特性; 最后对本文进行了总结.

1 相关工作

在这一节中, 主要对增量学习算法、学习矢量量化算法等相关技术进行简要介绍.

1.1 相关技术

增量学习算法是分类算法研究的一个重要分支, 可有效处理数据流挖掘等大规模甚至海量数据分类情况. 本文使用了文献 [13] 中所定义的增量学习定义界定了所研究的增量算法, 其精髓在于无需存储所有的样本, 先通过部分原始数据构建基本分类器模型; 训练后, 只保存训练好的模型, 而抛弃原始样本; 当新训练样本达到时, 利用新样本对模型进行更新, 然后保存更新后的模型, 而将新样本抛弃; 循环此学习过程, 直至学习完所有训练样本, 进而获得原始数据集的最终分类器. 其学习规则主要体现在: 位于类别中心区域的数据样本, 往往对分类没有作用; 位于类间区域的样本, 对分类贡献度较高, 尤其高密度分布类别会影响其他类别, 易造成分类错误; 同时适度控制训练样本数量, 避免引发计算量大的问题. 因此增量学习算法必须扩大样本类间距离、稀疏化类中心区域样本, 为此产生了一系列的增量分类算法, 有增量 SVM 算法、增量聚类算法等. 在近邻原型算法中, 增量学习更能满足大规模数据集分类的缩简需求.

此外, 由 Kohonen[12] 发展起来的学习型矢量量化 LVQ, 是一种可用于模式识别的有指导的学习方法, 是对无监督自组织神经网络算法 (Self-organizing feature maps, SOFM) 的扩展, 其基本思想很好体现了近邻学习思想, 通过利用少量的权向量来表示数据的拓扑结构, 实质上就是一种基于竞争的学习策略, 其本质可以看成是最近邻分类. 所谓竞争性神经网络, 即把欧氏距离最小的神经元作为胜出神经元, 针对正确分类和不正确分类情况进行调整权值. 后来又发展了 LVQ 2、LVQ 2.1、LVQ 3 等算法, 在本质上未发生改变, 除最优胜神经元外, 考虑了次获胜神经元. 相对于 LVQ 而言, LVQ 2 及以后算法提出同时更新两个权向量的方法, 首先寻求输入样本最近近邻的两个代表向量, 继

而判断这两个代表向量与新到样本的类别信息, 若其中一个向量 (V_k^t) 与新到样本同类, 而另外一个向量 (V_j^t) 与输入样本异类, 同时更新两个权向量. 式 (1) 显示了 LVQ 算法的权重更新方式, 其中 $\eta(t)$ 为竞争学习速率参数.

$$\begin{cases} V_k^{(t+1)} = V_k^t + \eta(t)(X^t - V_k^t) \\ V_j^{(t+1)} = V_j^t - \eta(t)(X^t - V_k^t) \end{cases} \tag{1}$$

1.2　增量学习矢量量化算法

分析 LVQ 算法, 虽然经过有限次迭代可获得稳定状态, 但仍然存在一定缺点: 1) 在训练过程中与输入样本异类的权向量可能不收敛, 由于算法仅仅是对异类权向量采取一直远离输入样本的调整, 却没有考虑其最终的位置, 这样会导致该权向量的不收敛; 2) 需要有限次迭代才能实现对原始数据集的约简, 当遇到大规模数据, 其运行消耗量不可接受; 3) 虽可通过预先设定每类的原型数目精确控制原型集规模, 但引入预设参数有诸多不合理之处; 4) 通过融合随机梯度下降算法, LVQ 虽可实现对原型的高效竞争学习, 但未能较好解决原型集更新问题, 无法生成较切合数据集分布的原型集, 且无法适应新类别等数据变化.

结合 LVQ 不足和增量学习算法特征, Xu 等[17] 在 2012 年对传统 LVQ 改进, 首先, 针对 LVQ 传统两层竞争学习机制的多次学习策略进行了分析, 在结合两层学习机制相似性的基础上, 将原有的两层竞争学习机制简化为单层竞争学习机制, 减少了竞争学习的次数, 提高了竞争学习的效率. 其次, 针对 LVQ 多次迭代学习、原型个数无法动态调整、不能较好适应大规模数据约简的问题, 采取了三种处理策略: 1) 在单层竞争学习机制下, 扩充记录了代表向量被选作最优近邻原型 $winner$ 次数及近邻链接关系, 采用 LVQ 2 竞争学习策略调整 $winner$ 代表的权向量; 2) 在一定的周期内删除长期未作为 $winner$ 或 $winner$ 次数不符合一定阀值的代表向量; 3) 单遍学习样本集, 直到样本集全部扫描完毕, 从而实现了原型的增量获取. ILVQ 算法实现了神经网络在增量原型算法的应用新思路. 然而 ILVQ 仍存在以下不足: 1) 虽然实现了增量学习, 但仍存在着传统 LVQ 算法异类权向量偏离而导致的不收敛弊端; 2) 虽然实现了原型集周期性更新, 即存在人为预设带来的不足, 也带来了原型更新速率过缓等问题; 3) ILVQ 仅关注了最优和次优原型的被选择次数及近邻链接关系, 而忽略了最优及次优原型所代表区域的样本信息, 如区域的大小、$winner$ 所学习样本的个数、样本识别的误差率等重要信息, 而这些信息包含数据集的样本分布、样本密度等情况, 除了在原型获取阶段发挥识别作用, 也可在测试样本等未知样本分类过程中起到作用, 见图 1 和图 2 所示的原型学习与原型分类示意图. 其中图 1 显示包含三类别样本的数据集分布情况, 圆形区域表示已学习的原型邻域信息, 所示邻域中包含样本的密度和学习累积误差信息, 带圈样本 x 为当前待学习样本, s_w 和 s_r 为 x 的最优和次优近邻原型邻域, 带边框样本为邻域的代表原型 (图 2 相同); 图 2 显示原型学习完成后的原型集分布情况, 黑色方形样本为待分类样本 x, p_w 和 p_r 为其最优和次优近邻原型, 圆形区域为其代表原型对应的邻域, 邻域内的样本为该邻域已学习过的原始样本, 故邻域保留了原始样本的分布及密度等信息.

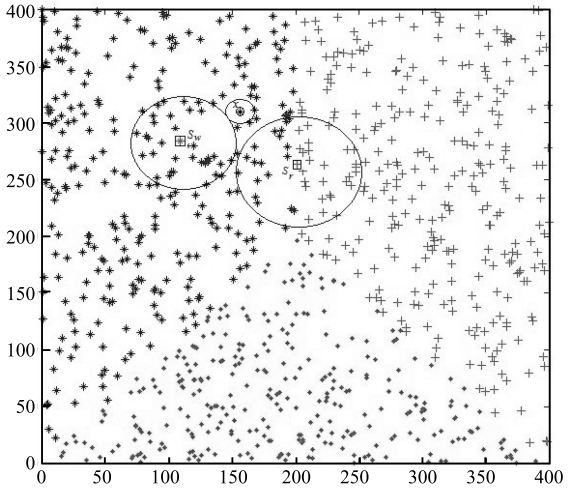


图 1　原型邻域信息在原型学习过程应用示意图

Fig. 1　The application diagram of the prototype neighborhood information in the prototype learning process

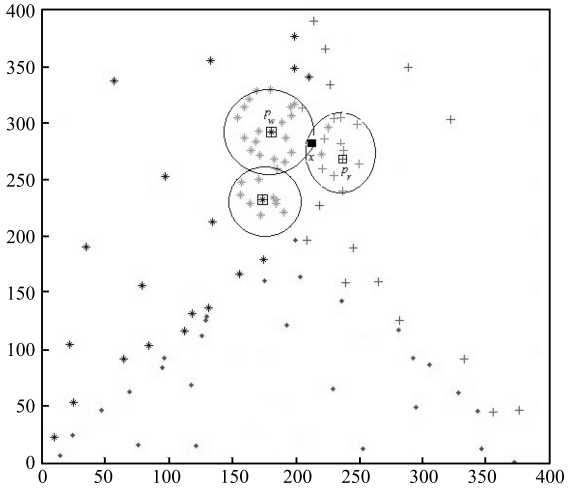


图 2　原型邻域信息在原型分类过程应用示意图

Fig. 2　The application diagram of the prototype neighborhood information in the prototype classification process

通过图 1 可获取到原型邻域信息在学习过程的三种应用情形: 1) 待学习样本 x 未处于 s_w 和 s_r 范围内, 则将其作为新的原型纳入到原型集中; 2) x 处于 s_w 范围内, 仅对 s_w 学习; 3) x 同时位于 s_w 和 s_r 范围内, 则同时对这两个邻域进行竞争学习.

通过图 2 可获取原型邻域信息在分类过程中的两种应用情形: 仅考虑距离等简单分类准则, 则退化为最近邻分类, 样本 x 的类别与 p_w 相同; 若考虑到原型邻域的密度以及累计误差等情况, 则 x 的类别与 p_r 相同, 能较好体现原始数据集的真实状态.

针对 ILVQ 算法不足, 本文首先重新考虑了节点邻域的累积误差、节点邻域密度在原型识别与更新中的作用; 其次扩充了 ILVQ 仅记录被选中次数与近邻链接关系为原型邻域关系; 然后采取自适应的动态调整策略, 加速了原型集的动态更新, 且一定程度上避免了 ILVQ 算法异类权向量过度偏离的弊端; 最后在 ILVQ 算法的基础上, 通过定义的原型邻域合并、分裂以及删除策略实现对原型的增量生成, 并将保留的节点邻域信息带入到原型分类过程.

本文重点研究增量原型生成算法, 主要通过对 ILVQ 的有效改进, 在其基础上提出了基于样本密度和分类误差率的增量学习矢量量化算法 IMLVQ. 本文的主要创新点有: 1) 扩充 ILVQ 算法神经元简单权值信息为超元组, 不仅记录着权向量信息, 还记录了该神经元所代表邻域的节点数目、误差率等信息; 2) 改进了 ILVQ 算法学习策略, 通过设定不同的学习策略, 可对最优胜原型和次优原型进行同步学习, 加速了算法的学习率, 同时减少了异类原型的远离样本集的程度, 进而强化了算法的收敛性; 3) 本文相对于文献 [17] 而言, 采用自适应的原型动态调整方式, 去除了 $AgeOld$ 参数, 简化了 IMLVQ 算法参数设置, 降低了算法性能对外部输入的依赖, 提高了算法更新效率; 4) 改变传统原型生成算法仅关注最终原型集, 而丢弃了原型所代表区域的其他重要信息, 本文充分利用了原型邻域的特征信息, 在分类算法中将原型邻域的错误率和所代表样本数目纳入到分类判别准则中.

2 改进的增量学习矢量量化算法

本文扩充 ILVQ 中仅记录获胜节点被选中次数及其近邻关系, 将原型扩展为超元组的代表原型邻域, 而超元组 s 通过五元组 $\langle cls, radius, num, rep, err\rangle$ 表示, 其中, cls 表示 s 邻域的类别信息; $radius$ 表示 s 的覆盖半径; num 表示 s 已经学习过的训练样本个数; err 表示 s 邻域的分类误差; rep 表示 s 的代表点. $LifeTime$ 表示原型生命周期, 记录了两个原型同时被选中作为最优原型及次优原型的次数. 在超元组 s 定义的基础上, 进一步定义了本文中的样本密度及分类误差率.

定义 1. 邻域样本密度. 设 s 为任一原型所对应的邻域, 该邻域样本密度为

$$d(s) = \frac{s.num}{\Omega(s.radius)}$$

其中, $\Omega(\cdot)$ 为 s 的邻域测度.

定义 2. 邻域分类误差. 当新样本 x^t 到达, 从所有包含 x^t 的邻域 $D(x^t, s_k^t.rep) < s_k^t.radius$ 中获取最近的 s_w^t $(w = \arg\min_k(D(x^t, s_k^t.rep) - s_k^t.radius))$ 及 $v_w^t = s_k^t.rep$, 若 $label(x^t) = s_w^t.cls$, 则 x^t 不增加 s_w^t 的邻域分类误差, 否则为

$$s_w^{t+1} = s_w^t.err + \frac{s_w^t.radius}{D(x^t, v_w^t)} \tag{2}$$

本文所提出的 IMLVQ 算法核心思想就是以 ILVQ 已有单层改进工作为基础, 依赖原型所对应超元组的信息, 在每次动态学习中通过设定的合并、分裂策略, 完成对相关原型及其邻域的更新; 其次在一定的更新周期内, 通过自适应的更新策略, 实现对已有原型集的更新及原型邻域动态调控; 最后直到训练样本集被全部扫描结束, 则算法完毕, 获取最终增量原型集合.

2.1 IMLVQ 算法介绍

围绕上述 IMLVQ 算法核心思想, 可得出算法关键处理策略: 首先扩充了 ILVQ 算法竞争学习涉及的简单原型近邻等信息为涵盖邻域类别、覆盖半径、已学习样本数、分类误差及代表点的超元组; 同时使用了 $LifeTime$ 表示原型生命周期, 记录了两个原型同时被选中作为最优原型及次优原型的次数, 而取消了 ILVQ 算法 $AgeOld$ 结构, 加速了算法的学习速率. 其次不同于 ILVQ 的增量学习方式, 本文中增量学习过程是针对于原型及其邻域超元组同步进行, 除新样本为新类别或者处于最近与次近邻原型邻域范围而作为新原型执行插入算法外, 执行待学习样本的最近与次近原型及其邻域的学习过程; 此外当最近与次优原型邻域交叠且类型相异, 则执行分裂过程, 降低原型邻域的分类错误率; 若两邻域交叠且类型相同时, 则执行合并过程, 可较好地稀疏化原型密集区域; 最后在预设的更新周期内, 采取自适应的更新策略对无近邻、长期未进行学习的低邻域样本密度等不满足条件的原型邻域进行删除操作, 进而对原型集进行动态更新. 基于样本密度和分类误差率的 IMLVQ 算法, 其算法运行示意图如图 3 所示.

图 3 示意了 IMLVQ 原型学习的主要处理过程, 展现了 IMLVQ 与 ILVQ 算法的主要区别, 即本文的改进创新处. IMLVQ 算法除兼顾 ILVQ 最优与次

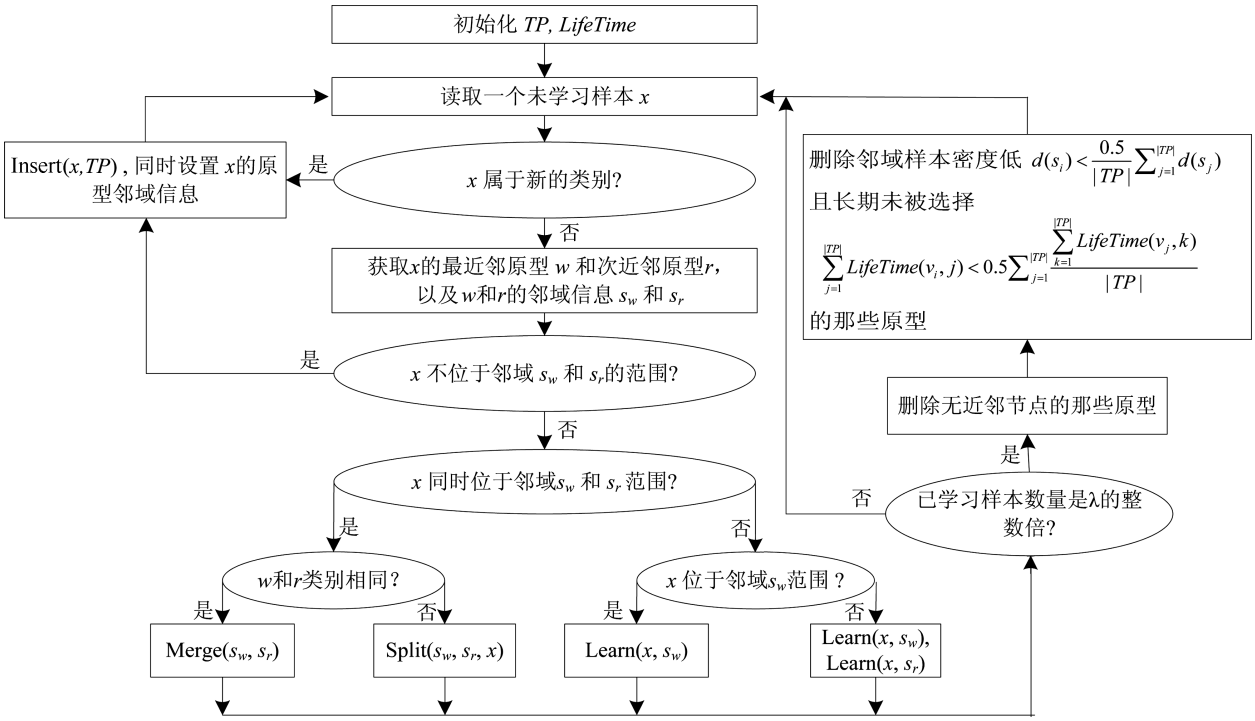


图 3　IMLVQ 算法运行示意图

Fig. 3　The running diagram of IMLVQ algorithm

优原型权向量竞争学习外, 还重点考虑了原型已学习样本的分布、学习误差以及样本密度等原始信息, 并根据待学习样本与最优和次优原型邻域的位置以及类别关系, 进行了分类处理. 其中自适应动态更新策略加速了原型学习, 扩充的邻域超元组保留了原型学习的以往记录, 合并操作稀疏化了类内部原型个数进而提高了算法的压缩率, 分裂操作降低原型邻域的误差, 使其更较好地贴合原始数据集分布状况, 提高了原型集的分类识别精度.

2.2　IMLVQ 算法主要处理过程

由图 3 可得出, IMLVQ 算法的关键步骤在于原型插入、学习、合并以及分裂策略. 相对于 ILVQ 算法, IMLVQ 算法不再保存记录了节点的 $winner$ 节点近邻连接关系; 同时不再通过 $AgeOld$ 结果仅记录原型被选作为 $winner$ 的次数, 而是使用 $LifeTime$ 矩阵记录了原型被选作最优近邻原型 $winner$ 和次优近邻原型 $runner$ 的次数, 即其生命周期, 采取自适应动态原型调整策略加速了原型集的更新. 故在原型学习过程中, 当任一待学习的新样本 x^t 到来, IMLVQ 会获取 x^t 的最优原型 v_w^t 和次优原型 v_r^t 以及对应的超元组邻域 s_w^t 和 s_r^t, 根据新样本与相关邻域的位置、邻域类别信息等因素进行不同的操作处理, 即形成了 IMLVQ 算法的关键步骤.

1) 当 x^t 不在最优与次优原型的邻域范围或为新类别样本, 则新增 x^t 为新原型, 同步初始化其原型邻域.

2) 当 x^t 位于最优原型邻域且不在次优原型邻域内, 则只更新最优原型邻域信息, 分两种情况进行更新:

a) 若 x^t 与最优原型邻域类别相同, 则仅更新最优原型邻域的已学习样本数量和代表原型权重;

b) 若 x^t 与最优原型邻域类别不同, 则还需多更新最优原型邻域的累积误差信息.

3) 当 x^t 同时位于两个原型邻域内, 且这两个邻域类别相同, 则进行合并算法.

4) 当 x^t 同时位于两个原型邻域内, 但这两个原型邻域类别不同, 则除上述学习算法外, 还进行分裂算法, 分裂的目标是减少原型邻域重叠, 并提升对应区域的分类识别能力.

下面分别介绍 IMLVQ 主要关键处理过程.

算法 1. 插入算法 Insert (x, TP)

输入. 训练样本 x, 原型集 TP, 生命周期矩阵 $LifeTime$.

输出. 学习后的 TP, $LifeTime$.

1) Initialize $v=\emptyset$
2) $v.cls = label(x)$
3) $v.mum = 1$
4) $v.err = 0$
5) $v.rep = x$
6) **if** $TP=\emptyset$ **then**
7)　$v.radius = 0$

8) **else**
9) **if** $|TP| = 1$ **then**
10) $v.radius = D(x, s_1.rep)/2$
11) $s_1.radius = D(x, s_1.rep)/2$
12) **else**
13) $v.radius = \frac{\sum_{i=1}^{|TP|} s_i.radius}{|TP|}$
14) **end if**
15) **end if**
16) add v into the set $LifeTime$ by
$LifeTime \leftarrow LifeTime \cup \{(v,1)\} \cup \{(1,v)\}$
17) set $LifeTime(v,:)$ and $LifeTime(:,v)$ to zero
18) $TP \leftarrow TP \cup \{v\}$
19) return TP and $LifeTime$

在插入策略下, 算法 1 中一个样本作为原型加入到原型集, 并为其构建了邻域超元组, 该邻域的半径采取原型邻域半径的平均值. 由于原型集周期性动态更新, 故新增原型邻域半径同样具有动态适应性, 更好体现了生成原型集的训练集分布状态. 此外, 插入算法有效解决了 LVQ 算法原型动态增加的问题, 可较好适应数据集的变化.

算法 2. 学习算法 Learn (x, s_i)

输入. 超元组 s_i, 训练样本 x.
输出. 学习后的新 s_i.
1) $s_i.num = s_i.num + 1$
2) **if** $s_i.cls = label(x)$ **then**
3) $s_i.rep = s_i.rep + \eta_1(x - s_i.rep)$
4) **else**
5) $s_i.rep = s_i.rep - \eta_2(x - s_i.rep)$
6) $s_i.err = s_i.err + s_i.radius/D(x, s_i.rep)$
7) **end if**
8) return s_i

在算法 2 中, 针对最优邻域的代表原型和次优邻域的代表原型通过动态竞争学习速率参数 η_1 和 η_2 进行学习. 本文选择了文献 [17] 中竞争学习策略, 其中 $\eta_1 = 1/s.num$, $\eta_2 = 1/(100 \times s.num)$. 与 ILVQ 算法一致, IMLVQ 动态变化的竞争调整策略具有文献 [17] 所具备的算法性能. 区别于 ILVQ 仅对优胜节点执行竞争学习, IMLVQ 加速了竞争学习的样本数目及学习速率, 可有利于算法快速获取原型, 且能削弱权向量远离样本集的程度, 原型取值更加贴近训练集样本的分布.

算法 3. 合并算法 Merge (s_1, s_2)

输入. 原型集 TP, 生命周期矩阵 $LifeTime$, 超元组 s_1, s_2.
输出. 合并后的新 TP, $LifeTime$.
1) $radius\_tmp = \frac{s_1.radius + s_2.radius + D(s_1.rep, s_2.rep)}{2}$
2) $s_1.num = s_1.num + s_2.num$
3) $s_1.rep = \frac{s_1.num \times s_1.rep + s_2.num \times s_2.rep}{s_1.num + s_2.num}$
4) $s_1.err = \frac{s_1.err \times radius\_tmp}{s_1.radius} + \frac{s_2.err \times radius\_tmp}{s_2.radius}$
5) $s_1.radius = radius\_tmp$
6) $LifeTime(s_1,:) =$
$LifeTime(s_1,:) + LifeTime(s_2,:)$
7) $delete(s_2, TP)$
8) delete the information of s_2 from $LifeTime$
9) return TP and $LifeTime$

当新样本 x^t 同时位于最优原型与次优原型邻域内, 且两个邻域类别相同, 表明这两个邻域类型一致且存在一定区域重叠. 合并核心思想为合并类别一致且存在重叠区域的原型邻域, 可实现稀疏化类中心区域原型, 进而降低原型集规模.

算法 3 实现合并操作, 将存在区域交叠且类型相同的原型邻域合并更新为一个新的大邻域, 并引发其代表原型及 $LifeTime$ 更新调整, 完成了有一定区域交叠的相邻原型及其原型更新, 较好实现了对原型密集区域的稀疏化处理, 从而加速更新原型集.

算法 4. 分裂算法 Split (s_1, s_2, x)

输入. 原型集 TP, 生命周期矩阵 $LifeTime$, 超元组 s_1, s_2, 训练样本 x.
输出. 分裂后的新 TP, $LifeTime$.
1) **if** $s_1.cls = label(x)$ or $s_1.err < s_2.err$ **then**
2) $s_2.err = \frac{s_2.err}{\max\left(\frac{2 \times s_2.radius}{D(s_1.rep, s_2.rep)}, \frac{D(s_1.rep, s_2.rep)}{2 \times s_2.radius}\right)}$
3) $s_2.num =$
$\frac{s_2.num \times (D(s_1.rep, s_2.rep) - s_1.radius + s_2.radius)}{2 \times s_2.radius}$
4) $s_2.rep = s_2.rep + 2 \times (s_2.rep - s_1.rep) \times$
$(D(s_1.rep, s_2.rep) - s_1.radius - s_2.radius)$
5) $s_2.radius =$
$\frac{(D(s_1.rep, s_2.rep) - s_1.radius + s_2.radius)}{2}$
6) **else**
7) $s_1.err = \frac{s_1.err}{\max\left(\frac{2 \times s_1.radius}{D(s_1.rep, s_2.rep)}, \frac{D(s_1.rep, s_2.rep)}{2 \times s_1.radius}\right)}$
8) $s_1.num =$
$\frac{s_1.num \times (s_1.radius + D(s_1.rep, s_2.rep) - s_2.radius)}{2 \times s_1.radius}$
9) $s_1.rep = s_1.rep + 2 \times (s_1.rep - s_2.rep) \times$
$(D(s_1.rep, s_2.rep) - s_1.radius - s_2.radius)$
10) $s_1.radius = \frac{s_1.radius - s_2.radius + D(s_1.rep, s_2.rep)}{2}$
11) **end if**
12) $LifeTime(s_1, s_2) = LifeTime(s_1, s_2) + 1$ and
$LifeTime(s_2, s_1) = LifeTime(s_2, s_1) + 1$
13) return TP and $LifeTime$

当新样本 x^t 同时位于最优原型邻域和次优原型邻域内, 但这两个原型邻域类别相异, 说明这两个邻域类别相异且存在一定区域重叠, 而重叠区域易造成较大分类误差且影响原型集的识别性能. 分裂思想通过降低原型邻域的重叠, 降低邻域误差, 进而提高原型集的正确样本识别率.

算法 4 给出了最近原型邻域 s_w^t 和次优原型邻域 s_r^t 分裂的处理过程, 分裂策略保证了邻域误差率降低. IMLVQ 算法通过分裂操作将重叠相异邻域分离为不重叠的两个新邻域, 并触发相应代表原型及其邻域信息的更新. 分裂操作降低了原型邻域间的重叠冲突, 提高了原型及其邻域的代表能力, 降低了原型邻域的误差率.

2.2.1　改进的分类算法

执行 IMLVQ 完毕, 得到涵盖代表原型集的邻域超元组集 S. 传统的原型分类算法, 仅使用了原型样本, 而忽略了原型所代表的区域信息, 如区域样本数、区域误差等, 故本文在后续分类中充分利用 S 信息, 对传统近邻算法进行改进, 将原型邻域样本数和邻域误差引入到分类判决准则中. 故在传统的近邻分类判断准则中, 纳入待分类样本的最近和次近邻两个原型 n_w 和 n_r 的邻域超元组信息, 建立相应的分类判决函数, 如式 (3) 所示.

式 (3) 中, $\alpha+\beta=1$ ($\alpha\geq 0$, $\beta\geq 0$). 当 $\alpha+\beta=0$ 时, 式 (3) 也可退化为传统近邻策略. 通过调整 α 和 β, 可变换分类误差率和样本密度在分类判决中的不同侧重. 受制于篇幅, 本文中仅仅简单采用三种参数: $\alpha=\beta=0$, $\alpha=\beta=0.5$ 和 $\alpha=0.3$, $\beta=0.7$. 为克服距离量纲与 $s_w.err/\sum_{i=1}^{|TP|}s_i.err$ 和 $d(s_w)/\sum_{i=1}^{|TP|}d(s_i)$ 等量纲不同对分类的影响, 分类判决函数中的距离度量可通过核函数 $Sim(X_t,X_i)=\exp(-D(X_t,X_i))$ 转化距离为相似度, 使得相似度与 $s_w.err/\sum_{i=1}^{|TP|}s_i.err$ 和 $d(s_w)/\sum_{i=1}^{|TP|}d(s_i)$ 等量纲相同, 均保持在 [0,1] 范围内. 故借助于核函数, 式 (3) 可转换为式 (4).

3　算法分析

为了更好地评估 IMLVQ 的性能指标, 本文选择了 KNN、CNN、ENN、PSC 以及 ILVQ 算法, 从理论分析、算法评估两个方面进行算法的比较.

3.1　理论分析

分析所比较算法, 其中 KNN 算法的时间复杂度为 $\mathrm{O}(kn_1n^2)$; CNN 算法时间复杂度为 $\mathrm{O}(nN^2+n_1N^2)$; ENN 算法时间复杂度为 $\mathrm{O}(kn^3+n_1N^2)$; PSC 算法时间复杂度为 $\mathrm{O}(\tau rn+n_1N^2)$; ILVQ 算法时间复杂度分为 $\mathrm{O}(\sum_{i=1}^{\lceil n/r\rceil}\lambda N_i+n_1N^2)=\mathrm{O}(n\sum_{i=1}^{\lceil n/\lambda\rceil}N_i+n_1N^2)$, 加号前面部分为原型生成时间复杂度, 加号后面部分为原型分类时间复杂度; IMLVQ 算法是增量学习算法, 主要分为两部分: 一部分为增量原型生成时间 $\mathrm{O}(n\sum_{i=1}^{\lceil n/\lambda\rceil}N_i)$, 一部

$$label(x)=\begin{cases} s_w.cls, & \text{若 } D(x,s_w.rep)>s_w.radius \text{ 或 } D(x,s_r.rep)>s_r.radius \\ \arg\min\left(D(x,s_w.rep)+\alpha\dfrac{s_w.err}{\sum\limits_{i=1}^{|TP|}s_i.err}+\beta\dfrac{d(s_w)}{\sum\limits_{i=1}^{|TP|}d(s_i)},\right. & \\ \qquad \left. D(x,s_r.rep)+\alpha\dfrac{s_r.err}{\sum\limits_{i=1}^{|TP|}s_i.err}+\beta\dfrac{d(s_r)}{\sum\limits_{i=1}^{|TP|}d(s_i)}\right).cls, & \text{否则} \end{cases} \tag{3}$$

$$label(x)=\begin{cases} s_w.cls, & \text{若 } D(x,s_w.rep)>s_w.radius \text{ 或 } D(x,s_r.rep)>s_r.radius \\ \arg\max\left(\exp(-D(x,s_w.rep))+\alpha\dfrac{s_w.err}{\sum\limits_{i=1}^{|TP|}s_i.err}+\beta\dfrac{d(s_w)}{\sum\limits_{i=1}^{|TP|}d(s_i)},\right. & \\ \qquad \left. \exp(-D(x,s_r.rep))+\alpha\dfrac{s_r.err}{\sum\limits_{i=1}^{|TP|}s_i.err}+\beta\dfrac{d(s_r)}{\sum\limits_{i=1}^{|TP|}d(s_i)}\right).cls, & \text{否则} \end{cases} \tag{4}$$

分为原型分类时间 $O(n_1N^2)$, 故整体时间复杂度为 $O(n\sum_{i=1}^{\lceil n/\lambda \rceil} N_i + n_1N^2)$. 上述公式中, n 为训练样本数, N 为最终原型数, N_i 为更新周期 i 的原型数, r 为聚类数, τ 为聚类迭代次数, n_1 为测试样本数, $\lceil n/\lambda \rceil$ 为两个增量算法的更新周期数. IMLVQ 算法对于所有的训练样本而言是近似线性的, 但其原型分类所需时间复杂度为传统的近邻分类算法时间需求. IMLVQ 算法是增量算法, 仅在原型生成过程中执行单遍训练样本训练学习, 并不需对训练集进行存储, 因此具备处理大规模数据集的能力.

3.2 实验评估

3.2.1 人工数据实验

为了更好地与 Xu 等[17] 提出的 ILVQ 算法进行比较, 本文选择了文献 [17] 实验所使用的人工数据集进行增量的原型分类比较. 在本节中, 我们引入 2 维数据集进行实验, 详见图 4 和图 5. 图 4 中包含五类数据, 类别 1 和类别 2 的数据满足 2 维高斯分布, 类别 3 和类别 4 数据分布为两个同心圆, 类别 5 数据满足正弦分布. 图 5 在图 4 有效数据分布的基础上, 加入了 20 % 的噪音 (服从二维均匀分布且随机分布, 划分为 5 个类别).

区别于文献 [17] 实验中多种样本读取序列和不同迭代次数, 本文中仅选取简单的单遍随机样本获取的方式. 图 6 和图 7 为两种算法在图 4 数据集上原型生成情况, 可以得出, IMLVQ 算法在保持原始样本集分布的情况下, 对其进行必要缩减, 其结果与 ILVQ 算法结果有可相较性. 图 8 和图 9 为两种算法在图 5 数据集上原型生成情况, IMLVQ 除原型个数明显少于 ILVQ 算法结果外, 还降低了噪音的敏感性, 其噪音数据数量有了明显减少. 此外, 图 7 相对图 6 避免了传统 LVQ 和 ILVQ 中对异类权向量采取一直远离输入样本的调整, 而未考虑其最终的位置的弊端, 原型代表集更能展现原始样本集分布状态, 并降低了原型集与原始数据集的偏离度, 两种数据集保持相一致数据的分布区间. 图 8 与图 9 的分布情况也进一步验证了 IMLVQ 的优势, 针对噪音数据, 虽然 IMLVQ 一定程度上扩大了数据的分布区间, 但 y 轴扩展区域 [350, 400] 的原型个数明显少于 ILVQ 算法运行结果.

3.2.2 手写体数字光学数据集及乳腺癌症图像实验对比

基于 IMLVQ 算法是对 ILVQ 算法必要改进的缘故, 文中亦采取了文献 [10] 中手写体数字光学识别和乳腺癌数据进行两种算法的比较. 其中手写体数字光学数据集含 0～9 阿拉伯手写体数字的 3 823 个训练图像信息和 1 797 个测试图像信息; 乳腺癌症图像数据含 212 个异常图像和 357 个正常图像, 共计 569 个乳腺图像, 该数据将每幅乳腺癌症图像提取 30 个维度详细描述. 在相同的 5-交叉验证下, 选取不同的 λ 和 $AgeOld$ 参数进行实验, 表 1～3 及图 10 和图 11 的数据均为随机样本学习序列下五次

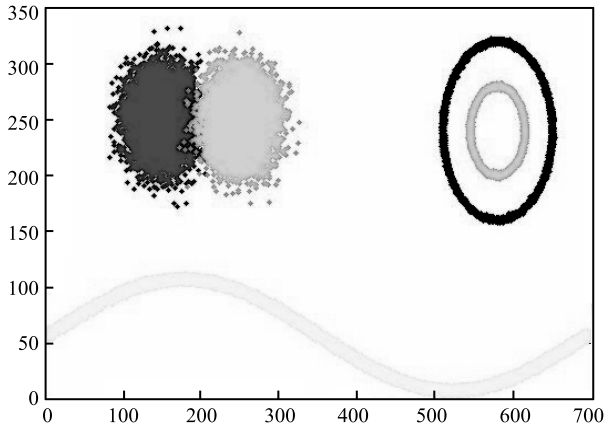


图 4 无噪音的人工数据集

Fig. 4 Nonoise artificial dataset

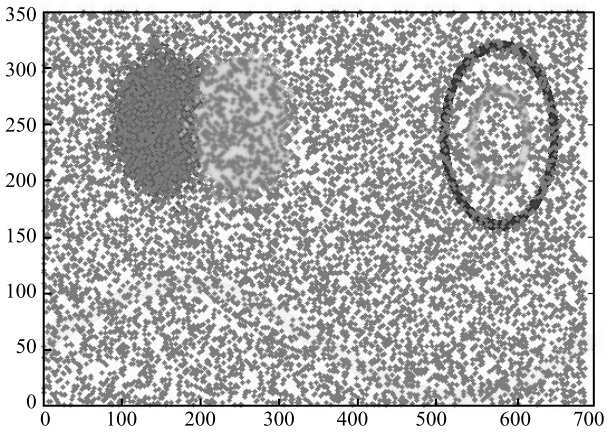


图 5 含噪音的人工数据集

Fig. 5 Noise artificial dataset

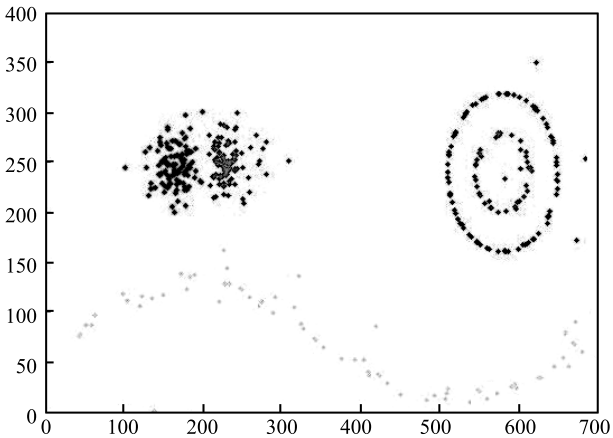


图 6 ILVQ 在无噪音数据集的原型集

Fig. 6 The prototype set obtained by ILVQ on nonoise dataset

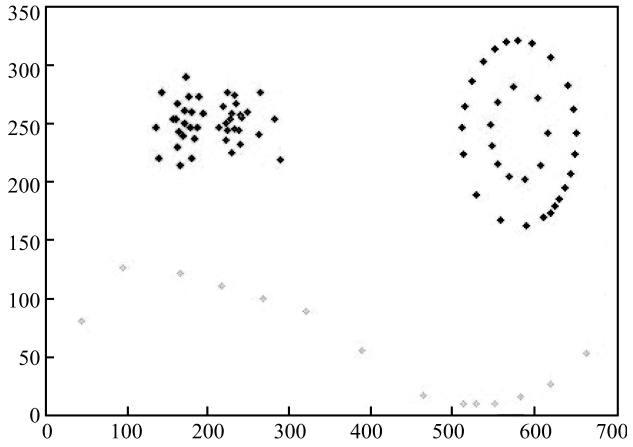


图 7　IMLVQ 在无噪音数据集的原型集

Fig. 7　The prototype set obtained by IMLVQ on nonoise dataset

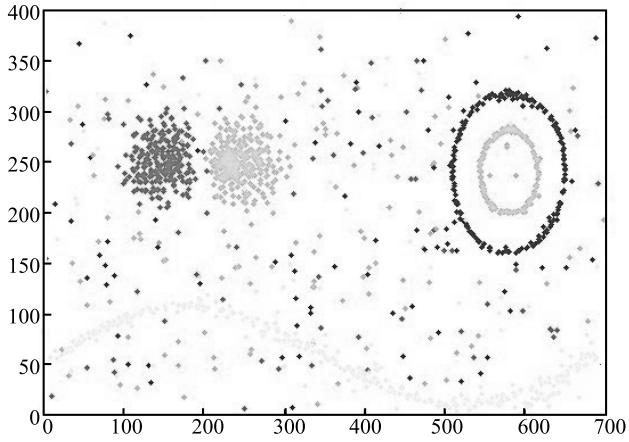


图 8　ILVQ 在噪音数据集的原型集

Fig. 8　The prototype set obtained by ILVQ on noise dataset

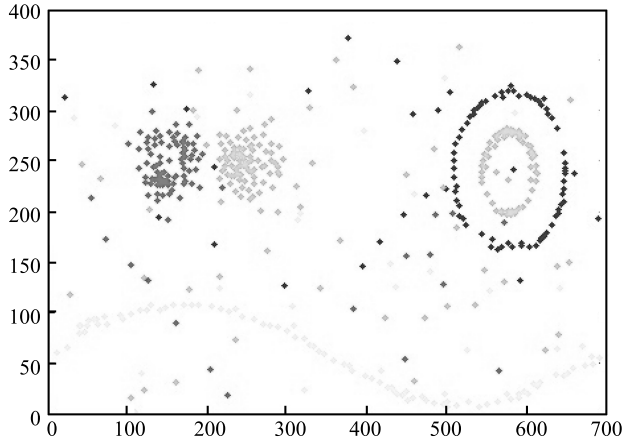


图 9　IMLVQ 在噪音数据集的原型集

Fig. 9　The prototype set obtained by IMLVQ on noise dataset

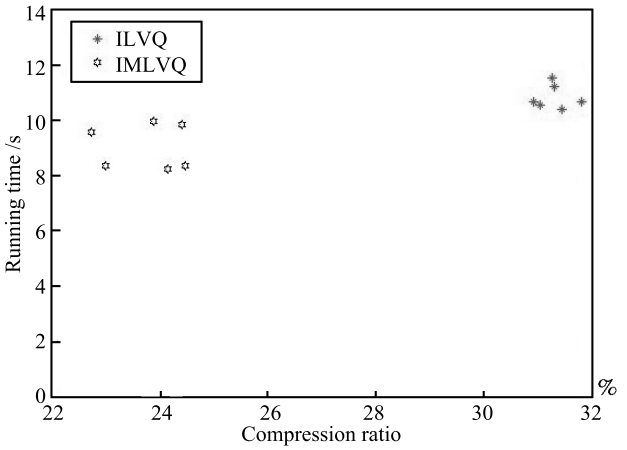


图 10　IMLVQ 和 ILVQ 在手写体数字光学识别数据集的压缩比率和运行时间散点图

Fig. 10　The scatter diagram of reduction ratio and running time obtained by IMLVQ and ILVQ on handwritten digits dataset

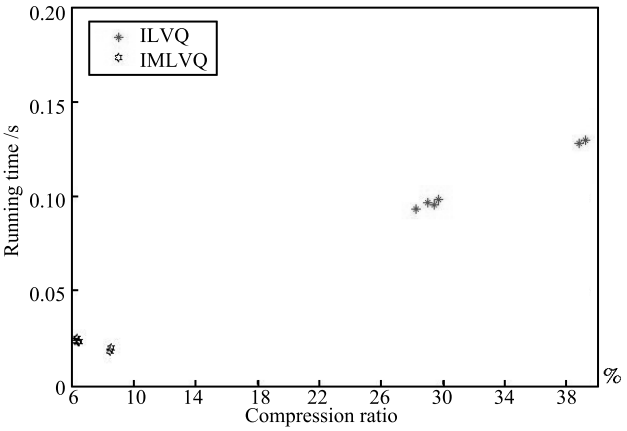


图 11　IMLVQ 和 ILVQ 在乳腺癌图像集的压缩比率和运行时间散点图

Fig. 11　The scatter diagram of reduction ratio and running time obtained by IMLVQ and ILVQ on breast cancer dataset

原型生成算法运行的平均值, 表 1 和表 2 中 IMLVQ 算法是 α、β 均为 0 时获取的近邻分类精度, 而表 3 则是在两种 α、β 参数取值下原型分类精度.

注 1. 压缩率 = 100 × 生成的代表样本个数/原始训练样本个数, 即 $str = 100 \times |TP|/|TR|$, $|\cdot|$ 表示集合元素的个数[7]; 近邻分类精度 = 100 × 被正确分类的测试样本个数/原始训练样本个数, 即 $Acc = 100 \times |TS_{\text{correct}}|/|TS|$, $|TS_{\text{correct}}|$ 表示被正确分类的测试样本个数; $Time$ 为比较算法的运行时间, 包含原型选择及原型分类等阶段, 以秒作为度量单位.

由表 1 可得出, 在手写体数字光学识别数据集上, 在 6 个不同的 λ 和 $AgeOld$ 情况下, IMLVQ 单

表 1 手写体数字光学识别数据集下 IMLVQ 和 ILVQ 比较

Table 1 The comparison between IMLVQ and ILVQ on handwritten digits dataset

算法	λ	*AgeOld*	$\lvert TP\rvert$	*Acc*	*Str*
ILVQ	200	200	1 406	97.78	31.27
	300	300	1 431	98.08	31.83
	400	400	1 408	97.83	31.32
	500	500	1 396	97.86	31.05
	600	600	1 415	97.90	31.47
	700	700	1 391	97.97	30.94
IMLVQ	200	–	1 033	98.10	22.98
	300	–	1 099	98.26	24.44
	400	–	1 086	97.79	24.15
	500	–	1 022	97.97	22.73
	600	–	1 073	98.17	23.87
	700	–	1 097	98.34	24.40

表 2 乳腺癌数据集下 IMLVQ 和 ILVQ 比较

Table 2 The comparison between IMLVQ and ILVQ on breast cancer dataset

算法	λ	*AgeOld*	$\lvert TP\rvert$	*Acc*	*Str*
ILVQ	7	7	177	87.96	38.88
	14	14	178	89.46	39.26
	21	21	129	89.19	28.32
	28	28	136	89.89	29.77
	35	35	134	87.44	29.46
	42	42	132	89.63	29.09
IMLVQ	7	–	22	89.90	4.88
	14	–	23	92.44	5.12
	21	–	29	91.69	6.44
	28	–	29	92.18	6.46
	35	–	29	91.30	6.33
	42	–	33	92.71	7.29

遍扫描获取的原型集, 其分类精度和压缩比均高于对应的 λ 和 *AgeOld* 参数下 ILVQ 实验结果, 但压缩比率的稳定性稍弱于 ILVQ 算法, 在图 10 中有所体现; 由图 10 可得出, IMLVQ 算法整体运行时间均小于 ILVQ 算法; 由表 2 及散点图 11 可得出, 针对不同的 λ 和 *AgeOld* 参数, IMLVQ 可取得较好的分类精度和压缩比, 并能保持较快的运行速度. 图像实验对比表明, IMLVQ 算法相比 ILVQ 算法而言, 有着分类精度、分类速度以及压缩比率等综合优势.

对比表 1～3 中的数据, 可得出以下结论: 由于随机样本读取序列, 表 1～3 中压缩比率虽有所波动, 但仍保持着高压缩效应; 针对于 α 和 β 在不同取值下的原型分类精度具有可比较性, 其中表 3 改进原型分类精度较好于表 1 和表 2 中传统原型分类精度. 结合表 1～3 中的实验数据, 基准数据集下设置不同的参数 α、β, 均可取得较好的分类精度和压缩比率. 实际应用中, 可根据关注点不同, 设置不同的 α、β 参数以满足不同应用需求.

3.2.3 UCI 基准数据集实验比较

为了更好地验证所提出的算法, 本文既选取了文献 [17] 中的数据集 (表 4), 也选取了其他 10 个 UCI[19] 小、中、大规模的数据集 (表 7) 进行实验对比; 同时为了展示所比较算法的平均运行效果, 采取 5-交叉验证方式的随机样本选取方式进行试验数据的获取, 所涉及表中的数据均来自于五次 5-交叉验证下的平均值. 本文选取了 CNN、ENN、PSC 以及 ILVQ 作为比较算法, 其中 CNN 和 ENN 是最简单经典的两个原型选择算法, PSC 算法体现了快速的边界数据获取的特点, 根据文献 [11], 在 $C = 6r$ 和 $C = 8r$ 情况下, 可以获得最佳的运行效率, 本文亦分别对此两种取值进行了五次 5-交叉验证, 获得了 PSC 算法的平均运行数据; ILVQ 则是本文比较的主要算法, 也是本文所提出改进算法的基础. 针对于 UCI 基准数据集, 从篇幅和简化实验出发, ILVQ 算法选择了相同的 λ 和 *AgeOld* 参数设置; IMLVQ 算法虽无需选择 *AgeOld* 参数, 但为了尽可能公平比较, 选择了与 ILVQ 相同的 λ 参数, 并设定了 $\alpha = \beta = 0$.

我们将 KNN 算法的分类实验数据列为对比数据之一, 主要以 KNN 分类实验数据作为参考基准, 对比所提出算法在保持较高压缩率的情况下, 分类精度的宏观体现. 而以四个对比算法的实验数据, 作为所提出算法性能指标比较的详细对比展示. 针对于 KNN 和 ENN 算法需预先设置 K 值的情况, 本文中选取了五个常见 K 值, 分别为 3, 5, 7, 9, 11. 表 5 显示各算法在表 4 数据集上的压缩比和分类精度实验数据, 表 6 显示各比较算法在表 4 数据集上的分类时间. 表 8 显示各算法在表 7 数据集上的压缩比和分类精度实验数据, 表 9 显示各比较算法在表 7 数据集上的分类时间.

基于表 5 数据, 可以得到如下结论: 对比 CNN 和 ENN, IMLVQ 在保持明显的数据优势情况下, 有着较高比例数据集的分类精度优势; 对比 ILVQ 算

表 3　不同参数下 IMLVQ 运行结果比较

Table 3　The operational results obtained by IMLVQ using the different parametric

算法	λ	$\alpha=0.3,\ \beta=0.7$			$\alpha=\beta=0.5$		
		Acc	*Str*	Time (s)	*Acc*	*Str*	Time (s)
OptialDigist	200	97.90	20.35	283.40	97.69	20.73	285.91
	300	98.13	22.96	303.32	98.02	22.89	306.84
	400	97.99	23.28	295.41	98.31	23.53	319.01
	500	98.33	24.82	362.02	98.10	25.20	359.88
	600	98.25	25.84	388.20	98.33	24.17	371.88
	700	98.11	26.28	394.99	98.63	25.32	385.57
Wdbc	7	91.22	3.56	4.38	91.28	4.22	4.95
	14	93.32	4.61	5.25	91.91	4.66	5.31
	21	89.98	4.83	5.43	91.76	4.39	5.10
	28	91.56	4.66	5.34	92.68	4.57	5.19
	35	92.62	5.62	6.15	91.74	6.46	6.84
	42	92.27	5.89	6.39	91.98	5.76	6.33

表 4　所选取的 UCI 数据集信息

Table 4　The information of the selected UCI benchmark datasets

数据集	特征数	类别数	样本数
Glass	9	6	214
Ionosphere	34	2	351
Iris	4	3	150
Liver disorders	6	2	345
Pima Indians	8	2	768
Wine	13	3	178

法, IMLVQ 在 5 个数据集上分类效率优势明显, 同时保证了 6 个数据集上的高压缩比; 相对 PSC 快速原型算法, IMLVQ 在保持 6 个数据集的全优分类效率之外, 仍能在 4 个数据集上取得较好的压缩比, 达到较好的分类效率和较高的压缩比率. 通过表 6 运行时间数据可得出, 由于数据规模较小, 故所提出的原型算法相对于 CNN 和 ENN 等经典原型算法而言, 无明显的时间优势; 此外 IMLVQ 相对于 ILVQ 算法而言, 无论数据集的规模如何, 均有着显著的运行时间优势; 而相对于目前快速原型算法之一 PSC 算法, IMLVQ 也有着明显的 6 取 5 和平均的两项时间优势.

综合表 5 和表 6 结论, 与 ILVQ 算法相似, 其改进算法 IMLVQ 不失为一种快速的原型生成算法, 能在保持较好压缩比的情况下, 具有较高的分类精度.

分析表 8 数据, 在保持小规模数据集优势处理性能的基础上, 对于大规模数据集 Pendigits 和 Letter, IMLVQ 仍可保持着不高于 10 % 压缩率和与 PSC 算法可相较的快速处理效率. 除 Yeast 外, IMLVQ 相对于其他比较算法, 获取了最佳的压缩率, 且分类精度相对于 ILVQ 算法而言, 获得 7 个数据集上的分类优势; 相对于 PSC、CNN 算法, 获得了绝对分类优势; 相对于 ENN 算法, 除 Breast Cancer 数据集外, IMLVQ 在其他数据集上均获得了较好的分类精度和压缩率. 此外, 从平均实验数据行中, 也得出 IMLVQ 有着较明显的运行优势. 故 IMLVQ 可适应不同规模数据集, 获得高效的分类精度和压缩率.

分析表 9 数据, KNN、CNN 因无需预先原型获取或运行简单等原因, 在小规模数据集中优势突出, 然而却导致了大规模数据集下无明显运行优势; 对于降量原型算法 ENN、PSC 及增量原型算法 ILVQ 和 IMLVQ 算法而言, 由于这些算法需要进行较为复杂的原型获取预处理, 故而小规模数据集下, 整体的运行时间优势不明显. 而针对于大规模数据集, KNN、ENN 算法, 因算法原理缘故, 其运行时间难以接受; PSC、ILVQ 和 IMLVQ 算法则显示快速运行优势, 其中 IMLVQ 是三者中最快的算法.

表 5 比较算法分类精度与压缩比的实验数据

Table 5 The results of classification accuracy and reduction ratio obtained by compared algorithms on UCI benchmark datasets

数据集	KNN		CNN		ENN		PSC		ILVQ		IMLVQ	
	Acc	*Str*	*Acc*	*Str*	*Acc*	*Str*	*Acc*	*Str*	*Acc*	*Str*	*Acc*	*Str*
Glass	65.08	100	62.64	66.59	59.77	48.07	60.69	72.78	64.69	28.74	62.44	19.98
Ionosphere	88.68	100	85.89	48.33	86.93	50.17	86.18	45.19	89.29	19.91	89.01	10.51
Iris	96.67	100	95.50	59.71	90.83	71.63	92.89	64.83	93.07	45.04	93.60	12.17
Liver disorders	67.61	100	55.80	16.67	60.72	54.80	61.88	67.21	60.87	15.89	60.41	28.37
Pima Indians	71.74	100	61.24	18.80	69.89	64.53	64.89	29.30	66.78	10.25	69.69	11.64
Wine	70.80	100	71.23	67.94	68.25	68.19	62.21	73.74	67.64	41.12	69.52	9.83
Average	76.76	100	72.05	46.34	72.73	59.57	71.46	58.84	73.72	26.83	74.11	15.42

表 6 比较算法的运行时间数据 (s)

Table 6 The running time results obtained by compared algorithms on UCI benchmark datasets (s)

数据集	KNN	CNN	ENN	PSC	ILVQ	IMLVQ
Glass	0.94	1.12	3.72	3.10	2.50	1.81
Ionosphere	2.53	2.58	4.39	3.02	3.93	3.84
Iris	0.48	0.57	2.10	0.74	0.67	0.53
Liver disorders	2.35	1.37	8.78	2.43	3.49	3.94
Pima Indians	2.33	2.75	6.58	2.48	2.94	2.30
Wine	0.64	1.30	3.02	1.28	2.11	0.57
Average	1.55	1.61	4.76	2.18	2.61	2.16

表 7 不同数据规模的 UCI 数据集信息

Table 7 The information of different scale UCI datasets

数据集	特征数	类别数	样本数
Haberman	3	2	306
Ecoli	7	8	336
Breast cancer	9	2	699
Transfusion	4	2	748
Ctg	20	3	2 126
Yeast	8	10	1 484
Landsat	36	6	4 435
Page	10	5	5 473
Pendigits	16	10	10 992
Letter	16	26	20 000

综上, 对比表 8 和表 9 更进一步确定了在不同规模数据集下, IMLVQ 相对于所比较算法有着较好的分类精度、压缩率和分类时间, 与第 3.2.1 节的结论相一致. 故 IMLVQ 是一种可适应不同数据规模的快速原型生成算法.

4 结论

本文在 ILVQ 的基础上, 引入了样本密度和分类误差率, 通过代表向量领域的合并、分裂以及删除操作, 进行 IMLVQ 代表向量集的动态增长. IMLVQ 算法的核心思想是在保证样本分布基础上, 基于样本密度和分类误差率来调整并获得动态增加原型代表样本. 实验数据表明, 所提出算法具有较好压缩比和分类精度的双优势, 且可保障原型集的快速获取, 能较好地应用到大规模数据集分类中.

表 8 比较算法分类精度与压缩比的实验数据

Table 8 The results of classification accuracy and reduction ratio obtained by compared algorithms on different scale UCI datasets

Dataset	KNN		CNN		ENN		PSC		ILVQ		IMLVQ	
	Acc	*Str*	*Acc*	*Str*	*Acc*	*Str*	*Acc*	*Str*	*Acc*	*Str*	*Acc*	*Str*
Haberman	71.98	100	53.45	28.66	71.75	71.08	71.57	41.18	72.04	26.94	71.42	13.82
Ecoli	86.45	100	77.44	53.71	76.32	53.82	74.93	42.89	83.87	42.96	84.13	12.46
BreastCancer	96.50	100	88.12	7.15	97.13	80.41	78.05	10.55	96.00	15.56	94.99	8.40
Transfusion	76.07	100	67.72	15.49	69.39	67.27	69.44	46.77	73.72	20.63	73.78	5.76
Ctg	82.09	100	62.68	42.91	74.43	52.08	74.86	41.66	68.06	8.23	76.04	7.45
Yeast	57.19	100	48.11	37.56	44.99	26.76	40.17	18.46	55.32	7.70	48.59	19.84
Landsat	90.27	100	76.71	24.79	65.19	48.82	80.65	25.65	87.73	24.45	88.21	15.78
Page	95.54	100	80.09	24.53	92.17	47.41	88.17	33.28	91.69	6.37	92.73	5.96
Pendigits	99.13	100	98.67	33.16	84.76	10.19	82.67	13.30	91.27	10.14	98.57	8.02
Letter	94.56	100	72.39	6.18	91.92	50.62	78.71	23.92	72.55	9.81	92.32	6.88
Average	84.98	100	72.54	27.41	76.81	50.85	73.923	29.77	79.23	17.28	82.08	10.44

表 9 比较算法的运行时间数据 (s)

Table 9 The running time results obtained by compared algorithms on different scale UCI datasets (s)

Dataset	KNN	CNN	ENN	PSC	ILVQ	IMLVQ
Haberman	1.43	0.78	1.73	0.91	0.79	0.65
Ecoli	4.01	3.61	3.71	1.88	3.42	2.07
Breast Cancer	11.47	3.06	22.66	2.25	27.27	3.53
Transfusion	4.41	3.09	13.02	6.68	4.61	2.77
Ctg	90.92	42.90	133.26	41.82	37.64	38.60
Yeast	166.19	61.91	249.16	31.58	13.05	7.03
Landsat	690.73	170.89	843.52	212.66	209.37	180.42
Page	300.48	122.66	429.02	192.13	187.15	156.85
Pendigits	1 629.88	593.64	1 917.92	402.96	549.78	142.61
Letter	6 488.37	462.77	9 683.57	650.37	428.09	329.45
Average	938.79	146.53	1 329.76	154.62	146.12	86.40

References

1 Wu X D, Kumar V, Quinlan J R, Ghosh J, Yang Q, Motoda H, McLachlan G J, Ng A, Liu B, Yu P S, Zhou Z H, Steinbach M, Hand D J, Steinberg D. Top 10 algorithms in data mining. *Knowledge and Information Systems*, 2008, **14**(1): 1−37

2 Triguero I, Derrac J, Garcia S, Herrera F. A taxonomy and experimental study on prototype generation for nearest neighbor classification. *IEEE Transactions on Systems, Man, and Cybernetics — Part C: Applications and Reviews*, 2012, **42**(1): 86−100

3 Rico-Juan J R, Iñesta J M. New rank methods for reducing the size of the training set using the nearest neighbor rule. *Pattern Recognition Letters*, 2012, **33**(5): 654−660

4 Angiulli F. Fast nearest neighbor condensation for large data sets classification. *IEEE Transactions on Knowledge and Data Engineering*, 2007, **19**(11): 1450−1464

5 Wilson D L. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on System, Man, and Cybernetics*, 1972, **SMC-2**(3): 408−421

6 Wilson D R, Martinez T R. Reduction techniques for instance-based learning algorithms. *Machine Learning*, 2000, **38**(3): 257−286

7 Chang C L. Finding prototypes for nearest neighbor classifiers. *IEEE Transactions on Computers*, 1974, **C-23**(11): 1179−1184

8 Raicharoen T, Lursinsap C. A divide-and-conquer approach to the pairwise opposite class-nearest neighbor (POC-NN) algorithm. *Pattern Recognition Letters*, 2005, **26**(10): 1554−1567

9 Kim S W, Oommen B J. Enhancing prototype reduction schemes with LVQ3-type algorithms. *Pattern Recognition*, 2003, **36**(5): 1083−1093

10 Fayed H A, Atiya A F. A novel template reduction approach for the k-nearest neighbor method. *IEEE Transactions on Neural Networks*, 2009, **20**(5): 890−896

11 Olvera-López J A, Carrasco-Ochoa J A, Martínez-Trinidad J F. A new fast prototype selection method based on clustering. *Pattern Analysis and Applications*, 2010, **13**(2): 131−141

12 Kohonen T. *Self-Organizing Maps (3rd Edition)*. New York: Springer-Verlag, 2001.

13 Polikar R, Byorick J, Krause S, Marino A, Moreton M. Learn++: a classifier independent incremental learning algorithm for supervised neural networks. In: Proceedings of the 2002 International Joint Conference on Neural Networks. Honolulu, HI: IEEE, 2002, **2**: 1742−1747

14 Zheng J, Shen F R, Fan H J, Zhao J X. An online incremental learning support vector machine for large-scale data. *Neural Computing and Applications*, 2013, **22**(5): 1023−1035

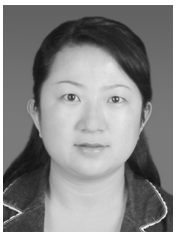
15 Liu J, Lee J P Y, Li L J, Luo Z Q, Wong K M. Online clustering algorithms for radar emitter classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2005, **27**(8): 1185−1196

16 Seo S, Mohr J, Obermayer K. A new incremental pairwise clustering algorithm. In: Proceedings of the 2009 International Conference on Machine Learning and Applications. Miami Beach, FL: IEEE, 2009. 223−228

17 Xu Y, Shen F R, Zhao J X. An incremental learning vector quantization algorithm for pattern classification. *Neural Computing and Applications*, 2012, **21**(6): 1205−1215

18 Calaña Y P, Reyes E G, Alzate M O, Duin R P W. Prototype selection for dissimilarity representation by a genetic algorithm. In: Proceedings of the 20th International Conference on Pattern Recognition. Istanbul: IEEE, 2010. 177−180

19 Newman A D. UCI Machine Learning Repository [Online], available: http://www.ics.uci.edu/~mlearn, February 24, 2014.



李 娟 西安电子科技大学计算机学院博士研究生, 陕西师范大学远程教育学院讲师. 主要研究方向为数据挖掘, 模式识别. E-mail: ally_2004@126.com

(**LI Juan** Ph. D. candidate at the School of Computer Science and Technology, Xidian University, and lecturer at the School of Distance Education, Shaanxi Normal University. Her research interest covers data mining and pattern recognition.)

王宇平 西安电子科技大学计算机学院教授. 主要研究方向为进化计算, 运筹学, 模式识别, 机器学习. 本文通信作者. E-mail: ywang@xidian.edu.cn

(**WANG Yu-Ping** Professor at the School of Computer Science and Technology, Xidian University. His research interest covers evolutionary computation, operational research, pattern recognition, and machine learning. Corresponding author of this paper.)