计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.200500077

基于层次标签的机器学习流程组装

陈　艳　陈佳晴　陈　星
福州大学数学与计算机科学学院　福州 350116
福建省网络计算与智能信息处理重点实验室　福州 350116
(chenyan_fzu@163.com)

摘　要　随着机器学习的兴起,算子数目飞速增长,组装算子需要搜索的解空间增大,流程组装时间指数倍增长,如何降低搜索解空间,从而降低组装时间,实现支持适应用户功能性需求的机器学习流程组装成为当前研究的热点。文中提出了一种基于层次标签、支持机器学习领域的流程组装方法。首先,从算子语义中提取标签,根据标签包含语义范围确定层次标签模型;其次,根据机器学习领域发现标签关系,确立领域组装模型,按照用户确定的功能性需求,确定最终领域标签模型;最后领域内算子与标签语义绑定,确定领域内算子关系模型,根据组装规则组装算子,形成满足用户功能性需求的全部算子流程。最后给出了支持该方法的实例,用以说明该方法的可行性;提出结果验证标准,用以说明结果的正确性与完整性。

关键词:服务组装;机器学习流程;语义网;层次标签;领域特性
中图法分类号　TP311

Machine Learning Process Composition Based on Hierarchical Label

CHEN Yan,CHEN Jia-qing and CHEN Xing
College of Mathematics and Computer Science,Fuzhou University,Fuzhou 350116,China
Fujian Provincial Key Laboratory of Networking Computing and Intelligent Information Processing,Fuzhou 350116,China

Abstract　With the rise of machine learning,the number of operators increases rapidly,the solution space of composition operators to search increases,and the process composition time exponentially increases. How to reduce the search solution space,thus reducing the assembly time,and realizing the machine learning process composition to meet the functional needs of users has become the current research hotspot. This paper proposes a process composition method based on hierarchical tagging to support machine learning. Firstly,the label is extracted from the operator semantics,and the hierarchical label model is determined according to the semantic scope of the label. Secondly,according to the machine learning domain discovery label relationship,the domain composition model is established,and the final domain label model is determined according to the functional requirements determined by users. Finally,the domain operators are bound with tag semantics,the domain operator relationship model is determined,and the operators are composed according to the assembly rules to form all operator processes that meet the functional requirements of users. At the end of this paper,an example is given to show the feasibility of the method,and the result verification standard is proposed to show the correctness and integrity of the result.

Keywords　Services composition,Machine learning process,Semantic Web,Hierarchical label,Domain feature

1　引言

随着机器学习的兴起,算子数目与日俱增,扩大了流程组装搜索的解空间。从复用的角度来说,当我们将算子视为可复用的算子构件实体时,流程组装可以看作基于构件的机器学习流程组装。

机器学习流程组装过程繁杂,流程风格多样,需要开发者投入大量的时间和精力进行流程的编排。针对机器学习流程风格多样的问题,需要建立统一语义模型,系统性描述各类算法组件的功能,设计模型驱动的智能组装方法,自动将面向场景的用户需求转换为优化的机器学习流程,从而极大地降低流程设计的难度和复杂度。

针对上述问题,前期工作主要包括两方面:

(1)机器学习领域的研究,以确定需要哪些标签及其间的连接关系[1-2];

(2)在实现某个特定功能的多个标签中如何组合该标签标记的算子。

研究实践表明,软件复用在特定领域内更容易获得成功[2-3]。机器学习算子可以看作一种可复用的软件构件,因此在机器学习领域进行组装实现复用会更有成效。我们将领域内的特定功能需求及其构件称为领域特性;在进行流程组装开发时,充分考虑机器学习领域特性,提高流程组装的开发效率。

基金项目:福建省引导性项目(2018H0017);中央引导地方科技发展专项(2019L3002)
This work was supported by Guiding Project of Fujian Province(2018H0017) and Special Project on Science and Technology Development of Central Government Guiding Local Government(2019L3002).
通信作者:陈佳晴(330297950@qq.com)

本文研究了一种基于层次标签的机器学习流程组装技术。本文的具体贡献如下。

(1)建立面向机器学习流程组装的层次标签模型。根据算子功能提取算子的功能标签，根据组装流程提取流程标签，发现两者间的关系，层次组合形成体系结构。

(2)形成领域组装模型。根据领域特性组装标签，形成领域组装模型。

(3)基于层次标签的机器学习流程组装方法。算子绑定标签，结合领域组装模型和用户功能性需求面生成所有可能的算子组装流程。

本文第 2 节介绍了相关的前期工作；第 3 节给出了基于层次标签的组装方法；第 4 节给出了初步的实验数据，以证明本方法的正确性和完整性；第 5 节对相关工作进行总结；最后总结全文并展望未来。

2　相关工作

标签系统构建层次体系[4-5]包括 3 个主要的步骤：建模标签语义、发现标签间关系、层次组合形成体系结构。语义建模是标签层次体系构建的基础。Trattner 等[6]提出 HyperTag 模型来考查对象的更多类型以建模标签的语义信息。Sun 等[7]和 Liu 等[8]提出一种自描述方法来来建模标签的语义信息。发现标签间关系是利用标签的语义模型来发现标签间的偏序关系。一种基本的标签关系发现方法是关联规则挖掘[9]，可以使用集合关系判断语义距离的方法[10]，还可以使用预定义的规则[11]发现标签间关系。依据标签语义建模及关系发现方法的不同，各个研究也使用不同的层次关系组合方法，如基于加权聚类的方法[12]等。

3　前期工作

3.1　层次标签模型

机器学习的流程是按照预处理、训练、测试的顺序执行。通过归纳算子功能，形成标签，最终得到标签集合{预处理，训练，测试，特征提取，特征选择，降维，分类，回归，聚类，分类测试，回归测试，聚类测试}。

通过人工分析发现，标签间的关系所蕴含的层次语义关系包括以下方面。

(1)严格上下位关系(＞)：上层标签蕴含的语义范围大于下层标签，且下层标签仅含有它的上位的语义，不包含其他上位概念。例如预处理的语义范围大于特征提取，特征提取仅含有特征提取的语义，不包含训练和测试的语义，因此预处理＞特征提取。

(2)同层关系(‖)：标签所蕴含的语义范围等大。

标签间关系所蕴含的层次语义关系使原有无序的标签具有了层次性和结构性。

如图 1 所示：虚线为分层的标志，因此一级标签测试‖训练‖预处理；二级标签特征提取‖特征选择‖降维‖分类‖回归‖聚类‖分类测试‖回归测试‖聚类测试。连线表示严格上下位关系：预处理＞(特征提取‖特征选择‖降维)；训练＞(分类‖回归‖聚类)；测试＞(分类测试‖回归测试‖聚类测试)。

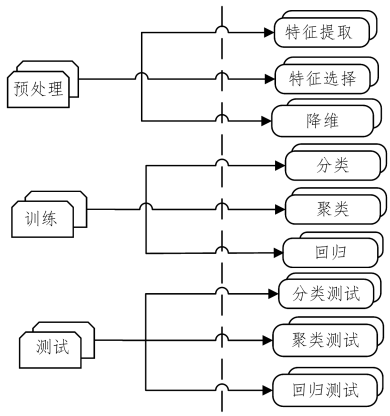


图 1　层次标签结构

Fig. 1　Hierarchical label structure

考虑到二级标签这一层具体明确，只提供二级标签供用户选择，并且下文考虑的都是二级标签。

3.2　领域特性

根据机器学习领域的相关研究，我们可以将标签的领域特性分为以下 4 类。

(1)必须的(mandatory)：所有流程都必须有这类标签。

(2)可选的(optional)：部分流程具有这类标签，但并非全部流程都具有。

(3)可丢弃的(Abandoned)：根据用户需求确定流程中不能具有这类标签。

(4)多选一的(alternative)：即流程组装在多个标签中，必须选且只能选其中一种标签。这是一组互相之间存在着互斥关系的特性。例如〈分类，聚类，回归〉这样一组标签，当单独地考查每项功能时，它们都是可选的特性。但是，一个特定的流程必须具有其中的一项功能，又只能具有其中的一项。即〈分类，聚类，回归〉形成互斥集。这类特性体现了系统间的变化性，另外还包括依赖、互斥关系。

(1)依赖关系是指只有在特性 p 存在的情况下，才能存在特性 q，这时称特性 q 依赖于特性 p。

(2)互斥关系是指特性 p 和 q 不能同时存在于一个流程中。上面讨论的多选一的特性是具有互斥关系的特性的一种特殊情况。

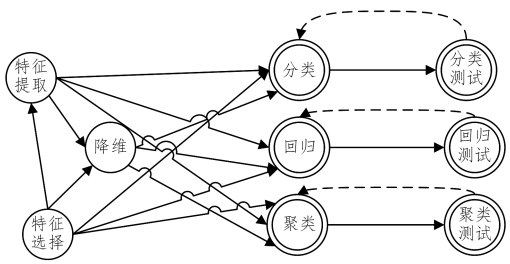
我们采用三元组描述领域组装模型，设为$\langle T, Alt, Dep\rangle$，其中 $T=\{T_1, T_2, \cdots, T_n\}$。对于其中的标签 T_i 设为四元组$\langle T_{pro}, T_{suc}, Opt, Aba\rangle$，其中 T_{pro} 和 T_{suc} 分别代表 T_i 的直接前驱标签集合和直接后继标签集合，若 T_i 为第一个起始标签，则 T_{pro} 为空，或最后一个结束标签，则 T_{suc} 为空；Opt 取值为 0 或 1，取 1 标签选择该标签，否则未选中；对于用户选中的标签，该项必为 1。Aba 取值为 0 或 1，取 1 则抛弃该标签，流程中不可具有该标签，否则可以具有。三元组$\langle T, Alt, Dep\rangle$中的 Alt 为描述领域中标签之间多选一关系的集合，即 $Alt=\{Alt_1, Alt_2, \cdots, Alt_k\}$，其中 Alt_i 为一个集合，该集合为 $Alt_i=\{T_i^j \mid T_i^j \in T\}$，表示该集合中的标签的属性为多选一，因此标签之间为互斥关系；而 Dep 是 T 集合中具有依赖关系标签的集合，即 $Dep=\{Dep_1, \cdots, Dep_m\}$，其中每个 Dep_i 为一个二元关系，表示为 $Dep_i=\{\langle T_a, T_b\rangle, T_a, T_b \in T\}$，代表 T_b 依赖于 T_a。

机器学习领域组装模型如图 2(a)所示。

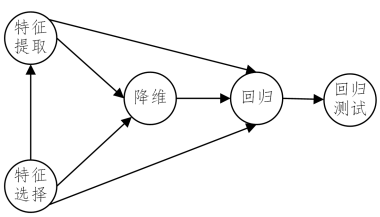
T={特征提取,特征选择,降维,分类,回归,聚类,分类测试,回归测试,聚类测试},其中假设用户选择了回归,则回归的可选属性为 1,为必选标签,其他为可选标签。

根据机器学习领域知识我们得知分类、回归、聚类为互斥关系,分类测试、回归测试、聚类测试也是互斥关系,我们用带下标的双圆圈表示哪几个标签具有互斥关系,下标相同的表示属于同一个互斥集,比如分类、回归、聚类属于下标为 1 的互斥集。根据机器学习领域知识我们还可以得知分类测试依赖于分类,回归测试依赖于回归,聚类测试依赖于聚类,在图 2(a)上我们用虚线指明标签的依赖关系,箭头所指代表被依赖方。对于 T 中的元素我们以标签回归为例,回归可表示为〈{特征提取,特征选择,降维},{回归测试},1,0〉,而对于该领域组装模型的 Alt 和 Dep,Alt={Alt_1,Alt_2},其中 Alt_1={分类,回归,聚类},Alt_2={分类测试,回归测试,聚类测试};Dep={Dep_1,Dep_2,Dep_3},其中 Dep_1={〈分类,分类测试〉},Dep_2={〈回归,回归测试〉},Dep_3={〈聚类,聚类测试〉}。

我们可以根据用户的需求和领域模型对标签进行选择,确定标签是否在最终组装中出现,以根据标签之间的关系确定最终的组装流程。我们可以获得如图 2(b)所示的最终流程。



(a)领域组装模型



(b)最终组装流程

图 2 领域组装模型和最终组装流程

Fig. 2 Domain assembly model and final assembly process

3.3 算子绑定标签

考虑到二级标签具体明确,我们将算子归入到二级标签中,并且只提供二级标签供用户选择。

采用“用户-标签-算子”三元组的社会标注模型。如图 3 所示,“用户-标签-算子”三元组模型中,U 表示用户的集合;T 表示标签的集合,包括特征提取、特征选择、降维、分类、回归、聚类、分类测试、回归测试、聚类测试;Op 表示算子的集合,共涉及 21 个算子,包括 Dicvectorize、Chi Square、Variance Threshold、SVM-RFE、Kernel approximation、Isomap、Embedding、K-Mean 聚类、模糊 K-Mean 聚类、Canopy 聚类、谱聚类、Logistic、SVM、KNN、Decision Tree、Lasso、SVR_Linear、Ridge Regression、Clustering_Test、Regression_Test、Classification_Test。将算子根据功能与对应的标签绑定。对于 $u_i \in U$,$T_i \in T$,$Op_i \in Op$,三元组(u_i,T_i,Op_i)表示用户 u_i 使用标签 T_i 标注了算子 Op_i。

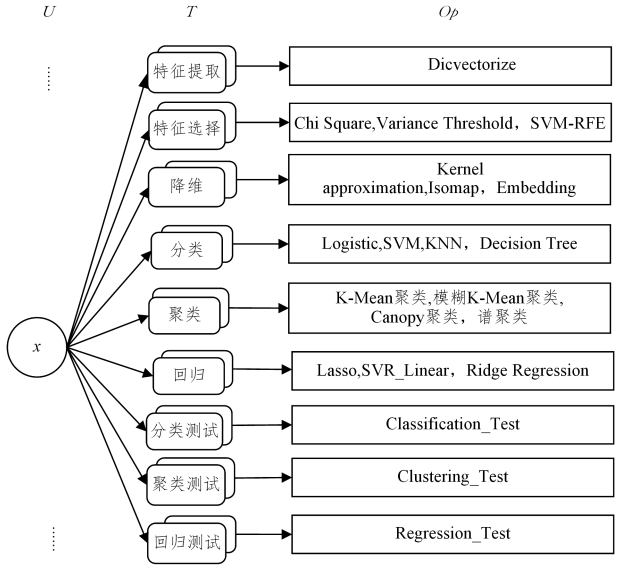


图 3 用户-标签-算子三元组模型图

Fig. 3 User-Label-Operator triple model

4 基于层次标签的机器学习流程组装方法

本文针对特定于机器学习领域的算子组装进行研究,这样能利用包含领域特性的领域组装模型指导组装,这一类型的组装关键是根据用户的功能性需求和领域模型固定领域变化性来组装算子。

4.1 形式化标签领域约束

对于 3.2 节所述的领域标签特性,我们采用下列方式进行形式化。

(1)必须标签

假设在领域组装模型中有一组必须的标签,记为 T_{man}。对于每一个在 T_{man} 中的元素,式(1)将作为约束条件加入。这一等式用以保证在目标系统中,每一在领域组装模型出现的标签必会出现。

$$x_i = 1(T_i \in T_{man}) \tag{1}$$

(2)可选标签

由功能需求确定该标签的选择与否,这是在算法之外确定的,其值在进入算法前已确定。

(3)可丢弃标签

由功能需求确定该标签的丢弃与否,这是在算法之外确定的,其值在进入算法前已确定。

(4)多选一标签

对于一组多选一的 T_{alt},等式(2)能保证有且仅有一个在 T_{alt} 中的标签能出现在目标系统的组装模型中。

$$\sum_{T_i \in T_{alt}} x_i = 1 \tag{2}$$

(5)依赖关系

对于两个标签 T_p 和 T_q,假设 T_p 依赖于 T_q,且对应于 T_p 的变量是 x_p,对应于 x_q 的标签是 T_q,那么不等式(3)能保证如果 T_p 被选中,T_q 也必会被选中。在式(3)中,如果 x_p 的值是 1,那么 x_q 的值也必会是 1。

$$x_p - x_q \leqslant 0 \tag{3}$$

(6)互斥关系

对于两个互斥的标签 T_p 和 T_q，假设 T_p 的变量是 x_p，T_q 的变量是 x_q，不等式(4)能保证这两个服务不能同时被选中。

$$x_p+x_q\leqslant 1 \tag{4}$$

4.2 形式化算子及其约束

对于3.3节所述的绑定标签的算子，对于任一 Op 的元素 Op_i 采用二元组〈$name$，T〉来表示；算子约束类比标签约束，我们采用下列方式进行形式化。

(1)必须算子

假设在流程组装中有一组必须的算子，记为 Op_{man}。对于每一个在 Op_{man} 中的元素，等式(5)将作为约束条件加入。这一等式用以保证在目标系统中，每一在流程中出现的算子必会出现。

$$x_i=1(Op_i\in Op_{man}) \tag{5}$$

(2)可选算子

由功能需求确定该算子的选择与否，这是在算法之外确定的，其值在进入算法前已确定。

(3)可丢弃算子

由功能需求确定该算子的丢弃与否，这是在算法之外确定的，其值在进入算法前已确定。

(4)多选一算子

对于指定的可选算子，对于与可选算子绑定同一标签的算子集合 Op_s，等式(6)能保证有且仅有一个在 Op_s 中的算子能出现在目标系统的组装模型中。

$$\sum_{Ops_i\in Ops} x_i=1 \tag{6}$$

4.3 流程定义

领域模型建立，采用四元组〈T^1_{pro}，T^1_{suc}，Opt，Aba〉，〈T^2_{pro}，T^2_{suc}，Opt，Aba〉，…，〈T^n_{pro}，T^n_{suc}，Opt，Aba〉为算子集对应的标签，对于两个的标签 T^p 和 T^q，如果满足不等式(7)，则被该标签标记的算子是可组装的。

$$T^p_{suc}\cap T^q_{pro}\neq\varnothing \tag{7}$$

算子组装流程用简单有向图 G 表示，由二元组组成，则 $G=\langle V,E\rangle$，其中 V 由算子组成，即 $V=\{Op_1,Op_2,\cdots,Op_m\mid m$ 为算子的个数$\}$，E 是由算子组成的边的集合，即 $e_{ij}=\langle Op_i,Op_j\rangle$，方向是从 Op_i 到 Op_j。

4.4 面向需求流程组装

用四元组〈$ChooseT$，$ChooseOp$，$AbandonT$，$AbandonOp$〉表示用户的功能性需求。

(1)$ChooseT$ 是用户指定的标签，则与其互斥的标签不能加入到流程中；

(2)$ChooseOp$ 是用户指定要加入到流程中的算子；

(3)$AbandonT$ 是用户指定的不要的标签，比如不要特征提取，则该类算子不能加入到流程中；

(4)$AbandonOp$ 是用户指定的不要的算子，则该算子不能加入到流程中。

我们需要根据功能性需求执行算法1得到面向需求的算子组装流程。

算法1　算子流程组装算法

```
输入：〈T，Op，ChooseT，ChooseOp，AbandonT，AbandonOp〉
输出：HashMap〈k，G〉
1. 初始化：k←0，G←null
2. /* 根据功能性需求处理算子集合 Op */
3. /* 从算子集合中删除不可加入的算子 */
4. for Op_i in Op
5.     for Op_j in AbandonOp
6.         if Op_i == Op_j
7.             remove Op_i from Op
8.         end if
9.     end for
10. end for
11. /* 从算子集合中删除被不可加入标签标记的算子 */
12. for Op_i in Op
13.     for T_i in AbabdonT
14.         if Op_i. T == T_i
15.             remove Op_i from Op
16.         end if
17.     end for
18. end for
19. /* 将于选中标签之间存在依赖关系的标签添加到选中标签集合 */
20. for T_i in T
21.     for T_j in ChooseT
22.         if x_i − x_j ≤ 0
23.             add T_i to ChooseT
24.         end if
25.     end for
26. end for
27. /* 从算子集合中删除被选中标签互斥的标签标记的算子 */
28. for Op_i in Op
29.     for T_j in ChooseT
30.         if 〈Op_i. T，T_j〉∈ T_alt
31.             remove Op_i from Op
32.         end if
33.     end for
34. end for
35. /* 从算子集合中删除与选中算子同属于一个标签的其他算子 */
36. for Op_i in Op
37.     for Op_j in ChooseOp
38.         if Op_i. T == Op_j. T
39.             remove Op_i from Op
40.         end if
41.     end for
42. end for
43. /* 根据组装规则组装算子流程 */
44. for Op_i in Op
45.     Op_start = Op_i
46.     for Op_j in Op
47.         Op_end = Opj
48.         if Op_i. T. T_suc ∩ Op_j. T. T_pro ≠ Ø
49.             add 〈Op_i，Op_j〉 to G. E，add {Op_i，Op_j} to G. V
50.             Op_start = Op_end，j←0
51.         end if
52.     end for
53.     if G. V ∩ ChooseOp ≠ Ø
54.         add G to HashMap〈k++，G〉
55.         G = null
56.     end if
57. end for
```

5 实验研究

面向用户功能性需求，构建组装模型，对组装结果的正确性和完整性进行验证。首先，验证方法是否能够实现面向用户需求的算子组装模型；其次，对得到的组装流程进行正确性和完整性验证。

5.1 实验设置

如图 3 所示，模型包括 9 个标签，分别是特征提取、特征选择、降维、聚类、分类、回归、聚类测试、分类测试、回归测试。在领域组装模型中，不同的标签存在直接前驱、直接后继、互斥、依赖等关系，分类、回归和聚类三者两两存在互斥关系；{特征提取}是特征选择的直接前驱；{特征选择，降维，聚类，分类，回归}是特征提取的直接后继；分类测试依赖于分类。共包括绑定 9 个不同标签的算子 21 个，分别是：

(1)绑定特征提取的算子 Dicvectorize；

(2)绑定特征选择的算子 Chi Square，Variance Threshold，SVM-RFE；

(3)绑定降维的算子 Kernel approximation，Isomap，Embedding；

(4)绑定聚类的算子 K-Mean 聚类、模糊 K-Mean 聚类、Canopy 聚类、谱聚类；

(5)绑定分类的算子 Logistic，SVM，KNN，Decision Tree；

(6)绑定回归的算子 Lasso，SVR_Linear，Ridge Regression；

(7)绑定聚类测试的算子 Clustering_Test；

(8)绑定分类测试的算子 Classification_Test；

(9)绑定回归测试的算子 Regression_Test。

5.2 算子组装模型建模

5.2.1 算子直接前驱和直接后继推导

$Op_i=\langle name, T\rangle$，其中 $T=\langle T_{pro}, T_{suc}, Opt, Aba\rangle$，标签关系只考虑 T_{pro} 和 T_{suc}，所以 T 在这里视为二元组 $\langle T_{pro}, T_{suc}\rangle$，因此 $Op_i=\langle name, \langle \{Op_x.\ name \mid Op_x.\ T\in T_{pro}, Op_x\in Op\}, \{Op_y.\ name \mid Op_y.\ T\in T_{suc}, Op_y\in Op\}\rangle\rangle$。

算子之间的直接后继关系如图 4 所示：圆圈表示算子，有向边表示直接后继，箭头指向的方向是算子的直接后继集合，为了使表示清晰，根据绑定的不同标签进行分类，比如与绑定的算子 Dicvectorize 的后继是绑定的标签属于 $T_{suc}=${特征提取，特征选择，降维，聚类，分类，回归}的算子集合，因此 Dicvectorize 的后继算子有：

(1)与特征选择绑定的算子 Chi Square，Variance Threshold，SVM-RFE；

(2)与降维绑定的算子 Kernel approximation，Isomap，Embedding；

(3)与聚类绑定的算子 K-Mean 聚类、模糊 K-Mean 聚类、Canopy 聚类、谱聚类；

(4)与分类绑定的算子 Logistic，SVM，KNN，Decision Tree；

(5)与回归绑定的算子 Lasso，SVR_Linear，Ridge Regression。

因此后继集合是有向边指向的集合的并集。由于直接前驱和直接后继是对应关系，在这边就不另外说明。

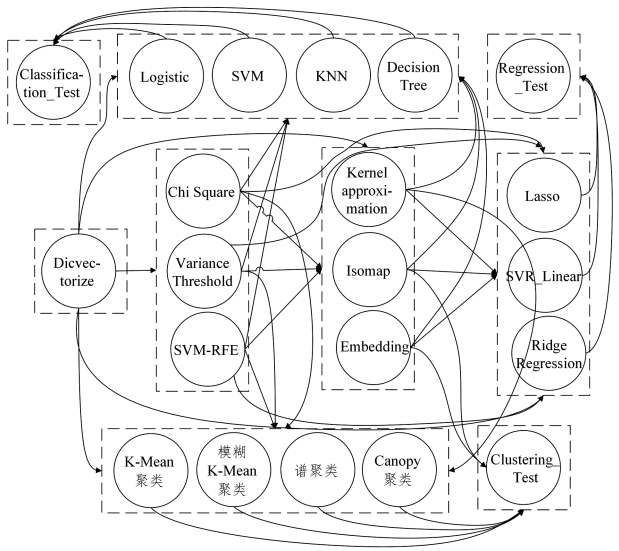


图 4 算子直接后继模型

Fig. 4 Operator direct successor model

5.2.2 算子互斥关系推导

$Alt=\{Alt_1, \cdots, Alt_k\}$，$Alt_i=\{T_i^j \mid T_i^j\in T\}$，$Op=\langle$name，$T\rangle$，因此 $Alt_i=\{\{Op_x.\ \text{name} \mid Op_x.\ T=T_i^j\} \mid T_i^j\in T\}$，即 $Alt_i=\{Op_x.\ \text{name}, Op_y.\ \text{name}, Op_z.\ \text{name} \mid Op_x.\ T, Op_y.\ T, Op_z.\ T\in T_i^j\}$。

算子互斥关系如图 5 所示，其中圆圈表示算子，无向边表示互斥关系。

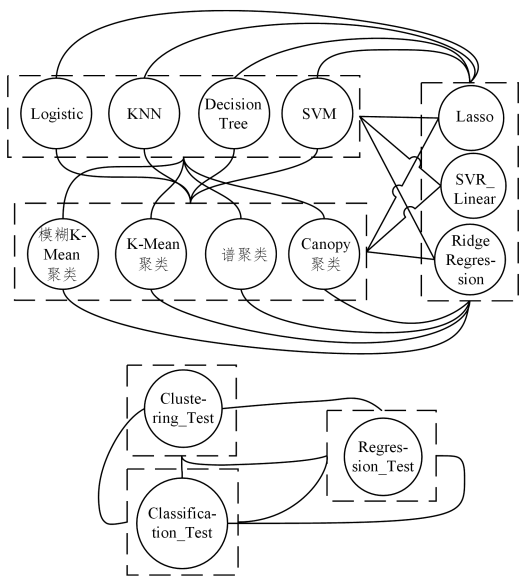


图 5 算子互斥模型

Fig. 5 Operator mutual exclusion model

通过推导关系可知，绑定标签属于互斥集合的算子集合之间存在互斥关系，可以认为，满足条件的算子集合中的任一算子满足互斥关系。比如，绑定聚类的算子 K-mean 聚类、绑定回归的算子 Lasso、绑定分类的算子 KNN 这 3 个算子存在互斥关系，即 $Alt_i=${K-means 聚类，Lasso，KNN}。为了更为形象，以算子为一端点，绑定标签的集合算子为另一端点，则算子与绑定标签的集合算子中任一算子有互斥关系。比如 Logistic 与{Lasso，SVR_Linear，Ridge Regression}有连线，即

Logistic 与属于{Lasso，SVR_Linear，Ridge Regression}的任一算子存在互斥关系。

5.2.3　算子依赖关系推导

$Dep=\{Dep_1,\cdots,Dep_m\}$，$Dep_i=\{\langle T_a,T_b\rangle,T_a,T_b\in T\}$；$Op_i=\langle \text{name},T\rangle$，因此 $Dep_i=\{\langle\{Op_x.\text{name}|Op_x.T=T_a\},\{Op_y.\text{name}|Op_y.T=T_b\}\rangle,T_a,T_b\in T\}$。

算子互斥关系如图 6 所示：我们用圆圈表示算子，箭头指向被依赖对象。通过推导关系我们可知，如果两个算子绑定的标签存在依赖关系，则存在依赖关系，比如与分类绑定的算子 Logistic 和与分类测试绑定的算子 Classification_Test，则〈Logistic，Classification_Test〉。

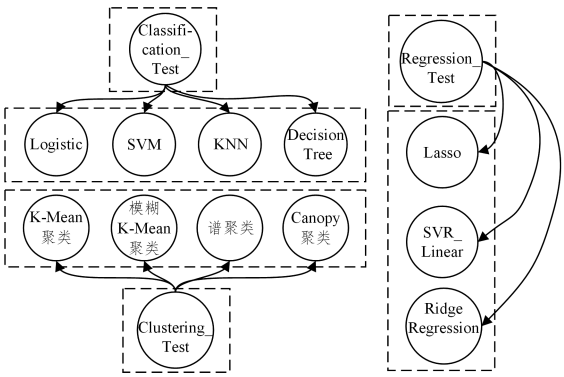


图 6　算子依赖模型

Fig. 6　Operator dependent model

5.3　算子组装模型建模

实例化功能性需求：ChooseT＝{特征提取，分类}；ChooseOp＝{Isomap}；AbandonT＝∅；AbandonOp＝{SVM-RFE，Logistic}。

按照算法 1 面向需求组装算子，最终得到的所有面向需求的算子组装流程如表 1 所列。其中序号代表流程编号，流程中的箭头表示执行顺序。

表 1　面向需求流程组装结果

Table 1　Requirements oriented process composition results

序号	组装结果
1	Dicvectorize－＞Chi Square－＞Isomap－＞KNN－＞Classification_Test
2	Dicvectorize－＞Chi Square－＞Isomap－＞SVM－＞Classification_Test
3	Dicvectorize－＞Chi Square－＞Isomap－＞Decision Tree－＞Classification_Test
4	Dicvectorize－＞Variance Threshold－＞Isomap－＞KNN－＞Classification_Test
5	Dicvectorize－＞Variance Threshold－＞Isomap－＞SVM－＞Classification_Test
6	Dicvectorize－＞Variance Threshold－＞Isomap－＞Decision Tree－＞Classification_Test
7	Dicvectorize－＞Isomap－＞KNN－＞Classification_Test
8	Dicvectorize－＞Isomap－＞SVM－＞Classification_Test
9	Dicvectorize－＞Isomap－＞Decision Tree－＞Classification_Test

5.3.1　有效性验证

根据算法组装流程生成 9 个满足需求的流程，如表 1 所列，根据序号对生成流程进行编号。选取 KDD_UP 数据集，分别对 9 个流程进行训练，生成 9 个模型，最后模型准确度如图 7 所示。

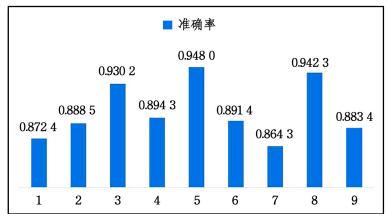


图 7　模型准确率

Fig. 7　Model accuracy

如图 5 所示，通过 KDD_UP 数据集训练，不同的 9 条流程训练出 9 个模型，且模型平均准确率达到近 90%，可以认为通过本方法组装流程所训练出的模型具有较高的准确率。

5.3.2　方法评估

为了验证方法的有效性，如表 2 所列，设置了 3 个不同的实例化之后的功能性需求，对这 3 个实例化后的功能性需求进行编号。

表 2　实验需求设置

Table 2　Experiment requirement setting

需求序号	ChooseT	ChooseOp	AbandonT	AbandonOp
1	回归	Chi Square	特征提取	Ridge Regression
		Embedding		Kernel approximation
2	聚类	Chi Square	特征提取	Ridge Regression
		Embedding		模糊 K-Mean 聚类
3	回归	Isomap	特征选择	Ridge Regression

表 3 对不同的 3 种实验设置，通过两种方法分别生成对应的算子组装流程。

表 3　两种方法组装结果比对

Table 3　Comparison of composition results of two methods

需求序号	人工流程组装	本方法
1	Chi Square－＞Embedding－＞Lasso－＞Regression_Test	Chi Square－＞Embedding－＞Lasso－＞Regression_Test
	Chi Square－＞Embedding－＞SVR_Linear－＞Regression_Test	Chi Square－＞Embedding－＞SVR_Linear－＞Regression_Test
2	Chi Square－＞Embedding－＞K-Mean 聚类－＞Clustering_Test	Chi Square－＞Embedding－＞K-Mean 聚类－＞Clustering_Test
	Chi Square－＞Embedding－＞Canopy 聚类－＞Clustering_Test	Chi Square－＞Embedding－＞Canopy 聚类－＞Clustering_Test
	Chi Square－＞Embedding－＞谱聚类－＞Clustering_Test	Chi Square－＞Embedding－＞谱聚类－＞Clustering_Test
3	Dicvectorize－＞Isomap－＞Lasso－＞Regression_Test	Dicvectorize－＞Isomap－＞Lasso－＞Regression_Test
	Dicvectorize－＞Isomap－＞SVR_Linear－＞Regression_Test	Dicvectorize－＞Isomap－＞SVR_Linear－＞Regression_Test
	Isomap－＞Lasso－＞Regression_Test	Isomap－＞Lasso－＞Regression_Test
	Isomap－＞SVR_Linear－＞Regression_Test	Isomap－＞SVR_Linear－＞Regression_Test

我们雇佣了 20 名学生，从 KDD_UP 中取 1 000 条数据，这些学生事先经过培训，已经掌握流程组装方法，对于每个实例化后的功能性需求，通过人工组装出流程，并且互相核对结果，结果如表 3 所列。同时记录这 20 个学生得到所有满足功能性需求组装流程所需时间，去掉最高值和最低值取平均值，最终得到人工组装所需时间（见表 4）。

针对 3 个实验设置，使用我们的方法得到所有的流程如

表 3 所列。将其与人工结果进行比对，得到覆盖率并记录组装运行时间，结果如表 4 所列。

表 4 两种方法性能比对

Table 4 Performance comparison of two methods

需求序号	1	2	3
人工流程组装/h	2.04	3.18	4.30
本方法/s	180.70	198.43	200.37
流程覆盖率/%	100	100	100

如表 4 所列，本文方法可以实现对人工流程组装方法结果的 100%覆盖，这意味着本文方法可以得到与人工流程组装一样的结果。同时在保证 100%覆盖人工组装结果的基础上，本文方法大大降低了组装时间，大大节省了人工成本。

结束语 本文从层次标签的角度进行机器学习领域的算子组装，提出了一种基于层次标签的机器学习流程组装。通过从标签中挖掘各种关系，构建一个多关系的标签网络。算子绑定标签，组装充分考虑领域特性。实验结果表明，本文方法可以面向用户需求，组装出所有的算子流程。

未来工作的重点主要包含以下两个方面：一方面，探索更多的非功能属性[13]（如性能 、可靠性、可用性和安全性等），纳入当前工作模型中；另一方面，利用高效的搜索算法以解决算子数量增速过快的问题。

参 考 文 献

[1] ARANGO G, PRIET O-DIAZ R. Domain analysis concepts and research directions[C]// Domain Analysis and Software S ystem Modeling. Los Alamitos, California: IEEE Computer Society Press, 1991.

[2] PRIET O D R. Domain analysis for reu sability[C]// Proceedings of COMPSAC'87. Tokyo, Japan, 1987: 23-29.

[3] MILI H, MILI F, MILI A. Reusing software: Issues and research directions[J]. IEEE Transactions on Software Engineering, 1995, 21(6): 528-562.

[4] SONG Y, QIU B, FAROOQ U. Hierarchical tag visualization and applicationn for tag recommendations[C]// Proceedings of the 20th ACM Conference on Information and Knowledge Management. New York: ACM, 2011: 1331-1340.

[5] CANDAN K S, CARO L D, SAPINO M L. Creating tag hierarchies for effective navigation in social media[C]// Proceeding of the 2008 ACM Workshop on Search in Social Media. New York, ACM, 2008. 75-82.

[6] TRATTNER C, KORNER C, HELIC D. Enhancing the navigability of social tagging systems with tag taxonomies[C]// Proceedings of the 11th International Conference on Knowledge Management and Knowledge Technologies. New Work: ACM, 2011.

[7] SUN Y, QIU H P, WANG Q X. A method to generate hashtag vector and build hierarchy architecture of network users' self-describing hashtag [J]. Information Technology and Network Security, 2018, 37(11): 44-49.

[8] LIU S Q, BAI G W, SHEN H. A hierarchical classification system based on user self description label [J]. Computer Science, 2016, 43(7): 224-229, 239.

[9] SOLSKINNSBAKK G, GULLA J. A hybrid approach to constructing tag hierachies[C]// Proceedings of the 2010 Conference on the Move to Meaningful Internet Systems. Berlin: Springer, 2010: 975-982.

[10] LIU K P, FANG B X. Ontology learning method based on social annotation [J]. Journal of Computer Science, 2010, 33(10): 1823-1834.

[11] HELIC D, STROHMAIER M. Building directories for socail tagging systems[C]// Proceedings of the 20th ACM Conference on Information and Knowledge Management. New York: ACM, 2011: 525-534.

[12] ZHANG M Q, BAI L, WANG J B. Label propagation algorithm based on weighted clustering ensemble [J]. Journal of Intelligent Systems, 2018, 13(6): 134-138.

[13] ZHAO J F, XIE B, ZHANG L, et al. A web service assembly method supporting domain features [J]. Journal of Computer Science, 2005(4): 731-738.

CHEN Yan, born in 1997, postgraduate. Her main research interests include blockchain and service composition.

CHEN Jia-qing, born in 1996, postgraduate. Her main research interests include software adaptive and so on.