Chapter 6
Learning Analysis

It is event that most big data represented as non-structured or semi-structured forms, such as images, text and others. It is important to study how to use an abstract form to show data, either structured, non-structured or semi-structured or use label proportions to categorize the nature of data so that a data mining or data analytic algorithm can be performed smoothly. Leaning methods are very useful tools for understanding the data. Learning algorithms can be considered from different aspects, such as cognitive computing, mathematics, and machine learning.

This chapter deals with different learning techniques in the contexts of data science. Section 6.1 discusses the view of learning through the concept (the abstract of big data), which includes four subsections. Section 6.1.1 is about concept-cognitive learning model for Incremental concept learning [58]. Section 6.1.2 is a concurrent concept-cognitive learning model for classification [60]. Section 6.1.3 is a semi-supervised concept learning by concept-cognitive learning and conceptual clustering [42]. Section 6.1.4 is a fuzzybased concept learning method-exploiting data with fuzzy conceptual clustering [43]. Section 6.2 presents how to use the label proportion for learning that consists of another four subsections. Section 6.2.1 is a fast algorithm for learning from label proportions [84]. Section 6.2.2 is a learning from label proportions with generative adversarial networks [39]. Section 6.2.3 is a learning from label proportions on high-dimensional data [57]. Section 6.2.4 is a learning from label proportions with pinball loss [59]. Section 6.3 explores other enlarged learning models with two subsections. Section 6.3.1 is about classifying with adaptive hyper-spheres: an incremental classifier based on competitive learning [38]. Section 6.3.2 is a construction of robust representations for small data sets using broad learning system [66].

6.1 Concept of the View of Learning

6.1.1 Concept-Cognitive Learning Model for Incremental Concept Learning

Cognitive computing is viewed as an emerging computing paradigm of intelligent science that implements computational intelligence by trying to solve the problems of imprecision, uncertainty and partial truth in biological system [44, 68, 72]. As far as we know, it has been investigated by simulating human cognitive processes such as memory [33, 62] learning [14, 30, 36], thinking [69] and problem solving [70].

In this subsection, a novel CCLM is proposed based on a formal decision context. Moreover, to reduce its computational complexity, granular computing is included in our model. The main contributions are as follows:

(1) We describe a new model for incremental learning from the perspective of cognitive learning by the fusion of concept learning, granular computing, and formal decision context theory. More precisely, it is an attempt to construct a novel incremental algorithm by imitating human cognitive processes, and a new theory has been proposed for concept classification under a formal decision context.
(2) Beyond traditional CCL systems such as approximate CCL system [35, 36], three-way CCL system [37] and theoretical CCL system [68–70], CCLM has obtained incremental concept learning and generalization ability.
(3) Different from other classifiers, similar to the human learning processes, the previously acquired knowledge can be directly stored into concept lattice space in CCLM and it performs a good interpretation by concept hierarchies (e.g., Hasse diagram [16]).

6.1.1.1 Preliminaries

Now, we briefly review some basic notions related to (1) formal context, (2) formal decision context and (3) concept-cognitive learning.

A. Formal Context and Formal Decision Context

Definition 6.1 ([75]) A formal context is a triplet (G, M, I), where G is a set of objects, M is a set of attributes, and $I \subseteq G \times M$ is a binary relation between G and M. Here, gIm means that the object g has the attribute m.

Furthermore, the derivation operator $(\cdot)'$ is defined for $A \subseteq G$ and $B \subseteq M$ as follows:

$$\begin{aligned} A' &= \{m \in M | gIm \text{ for all } g \in A\}, \\ B' &= \{g \in G | gIm \text{ for all } m \in B\}. \end{aligned} \tag{6.1}$$

A' is the maximal set of the attributes that all the objects in A have in common and B' is the maximal set of the objects shared by all the attributes in B. A *concept* in the context (G, M, I) is defined to be an ordered pair (A, B) if $A' = B$ and $B' = A$, where the elements A and B of the concept (A, B) are called the extent and intent, respectively. The set of all concepts forms a complete lattice, called the concept lattice and denoted by $L(G, M, I)$.

Definition 6.2 ([74, 82]) A formal decision context is a quintuple (G, M, I, D, J), where (G, M, I) and (G, D, J) are two formal contexts. M and D are respectively called the conditional attribute set and the decision attribute set with $M \cap D = \emptyset$.

Definition 6.3 ([34]) Let (G, M, I, D, J) be a formal decision context and $E \subseteq M$. For any $(A, B) \in L(G, E, I_E)$ and $(Y, Z) \in L(G, D, J)$, if $A \subseteq Y$, and A, B, Y and Z are nonempty, then we say that (Y, Z) can be implied by (A, B), which is denoted by $(A, B) \to (Y, Z)$.

By Definitions 6.2 and 6.3, we obtain the relationship between the conditional attribute set and the decision attribute set.

B. Concept-Cognitive Learning

Let G be an object set and M be an attribute set. We denote the power sets of G and M by 2^G and 2^M, respectively. In addition, $\mathcal{F} : 2^G \to 2^M$ and $\mathcal{H} : 2^M \to 2^G$ are supposed to be two set-valued mappings, and they are rewritten as $\mathcal{F}$ and $\mathcal{H}$ for short.

Definition 6.4 ([36]) Set-valued mappings $\mathcal{F}$ and $\mathcal{H}$ are called cognitive operators if for any $A_1, A_2 \subseteq G$ and $B \subseteq M$, the following properties hold:

$$\begin{aligned} &\text{(i)} && A_1 \subseteq A_2 \Rightarrow \mathcal{F}(A_2) \subseteq \mathcal{F}(A_1), \\ &\text{(ii)} && \mathcal{F}(A_1 \cup A_2) \supseteq \mathcal{F}(A_1) \cap \mathcal{F}(A_2), \\ &\text{(iii)} && \mathcal{H}(B) = \{g \in G | B \subseteq \mathcal{F}(\{g\})\}. \end{aligned}$$

For convenience, hereinafter $\mathcal{F}(\{g\})$ is rewritten as $\mathcal{F}(g)$ for short when there is no confusion.

Definition 6.5 ([36]) Let $\mathcal{F}$ and $\mathcal{H}$ be cognitive operators. For $g \in G$ and $m \in M$, we say that $(\mathcal{H}\mathcal{F}(g), \mathcal{F}(g))$ and $(\mathcal{H}(m), \mathcal{F}\mathcal{H}(m))$ are granular concepts.

Definition 6.6 ([36]) Let G_{i-1}, G_i be object sets of $\{G_t\} \uparrow$ and M_{i-1}, M_i be attribute sets of $\{M_t\} \uparrow$, where $\{G_t\} \uparrow$ is a non-decreasing sequence of object sets $G_1, G_2, \ldots, G_n$ and $\{M_t\} \uparrow$ is a non-decreasing sequence of attribute sets $M_1, M_2, \ldots, M_m$. Denote $\Delta G_{i-1} = G_i - G_{i-1}$ and $\Delta M_{i-1} = M_i - M_{i-1}$. Suppose

1) $\mathcal{F}_{i-1} : 2^{G_{i-1}} \rightarrow 2^{M_{i-1}}$, $\mathcal{H}_{i-1} : 2^{M_{i-1}} \rightarrow 2^{G_{i-1}}$,

2) $\mathcal{F}_{\Delta G_{i-1}} : 2^{\Delta G_{i-1}} \rightarrow 2^{M_{i-1}}$, $\mathcal{H}_{\Delta G_{i-1}} : 2^{M_{i-1}} \rightarrow 2^{\Delta G_{i-1}}$,

3) $\mathcal{F}_{\Delta M_{i-1}} : 2^{G_i} \rightarrow 2^{\Delta M_{i-1}}$, $\mathcal{H}_{\Delta M_{i-1}} : 2^{\Delta M_{i-1}} \rightarrow 2^{G_i}$,

4) $\mathcal{F}_i : 2^{G_i} \rightarrow 2^{M_i}$, $\mathcal{H}_i : 2^{M_i} \rightarrow 2^{G_i}$

are four pairs of cognitive operators satisfying the following properties:

$$\mathcal{F}_i(g) = \begin{cases} \mathcal{F}_{i-1}(g) \cup \mathcal{F}_{\Delta M_{i-1}}(g), & \text{if } g \in G_{i-1}, \\ \mathcal{F}_{\Delta G_{i-1}}(g) \cup \mathcal{F}_{\Delta M_{i-1}}(g), & \text{otherwise}, \end{cases} \tag{6.2}$$

$$\mathcal{H}_i(m) = \begin{cases} \mathcal{H}_{i-1}(m) \cup \mathcal{H}_{\Delta G_{i-1}}(m), & \text{if } m \in M_{i-1}, \\ \mathcal{H}_{\Delta M_{i-1}}(m), & \text{otherwise}, \end{cases} \tag{6.3}$$

where $\mathcal{F}_{\Delta G_{i-1}}(g)$ and $\mathcal{H}_{\Delta G_{i-1}}(m)$ are set to be empty when $\Delta G_{i-1} = \emptyset$, and $\mathcal{F}_{\Delta M_{i-1}}(g)$ and $\mathcal{H}_{\Delta M_{i-1}}(m)$ are set to be empty when $\Delta M_{i-1} = \emptyset$. Then we say that $\mathcal{F}_i$ and $\mathcal{H}_i$ are extended cognitive operators of $\mathcal{F}_{i-1}$ and $\mathcal{H}_{i-1}$ with the newly input information ΔG_{i-1} and ΔM_{i-1}.

In other words, based on Definitions 6.4 and 6.5, the basic mechanism of concept-cognitive process is shown in Definition 6.6.

6.1.1.2 Theoretical Foundation

In this section, for adapting to dynamic learning and classification task, we show some new notions and properties for the proposed CCLM.

A. Initial Concept Generation

Definition 6.7 A regular formal decision context is a quintuple (G, M, I, D, J), where for any $z_1, z_2 \in D$, $\mathcal{H}(z_1) \cap \mathcal{H}(z_2) = \emptyset$. (G, M, I) and (G, D, J) are called the conditional formal context and the decision formal context, respectively.

Note that it means that each real-world object is associated with a single label.

Definition 6.8 Let (G, M, I, D, J) be a regular formal decision context, and $\mathcal{F}$ and $\mathcal{H}$ be cognitive operators. For $g \in G$ and $m \in M$, we say that $(\mathcal{H}\mathcal{F}(g), \mathcal{F}(g))$

and $(\mathcal{H}(m), \mathcal{F}\mathcal{H}(m))$ are conditional granular concepts. Similarly, for $y \in G$ and $z \in D$, $(\mathcal{H}\mathcal{F}(y), \mathcal{F}(y))$ and $(\mathcal{H}(z), \mathcal{F}\mathcal{H}(z))$ are decision granular concepts. For simplicity, we denote

$$\mathcal{G}^C = \{(\mathcal{H}\mathcal{F}(g), \mathcal{F}(g))|g \in G\} \cup \{(\mathcal{H}(m), \mathcal{F}\mathcal{H}(m))|m \in M\},$$

$$\mathcal{G}^D = \{(\mathcal{H}\mathcal{F}(y), \mathcal{F}(y))|y \in G\} \cup \{(\mathcal{H}(z), \mathcal{F}\mathcal{H}(z))|z \in D\},$$

where $\mathcal{G}^C$ and $\mathcal{G}^D$ are respectively called as condition-concept space and decision-concept space (or class-concept space) under cognitive operators $\mathcal{F}$ and $\mathcal{H}$.

Property 6.1 Let (G, M, I, D, J) be a regular formal decision context, and $\mathcal{F}$ and $\mathcal{H}$ be cognitive operators. Then for any $A, Y \subseteq G$, $B \subseteq M$ and $Z \subseteq D$, we have

$$\mathcal{F}(A) = \bigcap_{g \in A} \mathcal{F}(g), \mathcal{F}(Y) = \bigcap_{y \in Y} \mathcal{F}(y),$$
$$\mathcal{H}(B) = \bigcap_{m \in B} \mathcal{H}(m), \mathcal{H}(Z) = \bigcap_{z \in Z} \mathcal{H}(z). \tag{6.4}$$

Proof It is immediate from Definitions 6.4 and 6.7. □

Property 6.2 Let (G, M, I, D, J) be a regular formal decision context. For any $(A_G, B_G) \in \mathcal{G}^C$ and $(Y_G, Z_G) \in \mathcal{G}^D$, if $A_G \subseteq Y_G$, and A_G, B_G, Y_G and Z_G are nonempty, then we say that A_G is associated with the class of Z_G under the attribute set B_G. It means that the object g can be represented by a single label z when $A_G = \mathcal{H}\mathcal{F}(g)$ and $Z_G = \mathcal{F}\mathcal{H}(z)$.

Proof It is immediate from Definitions 6.3 and 6.8. □

Definition 6.9 Let (G, M, I, D, J) be a regular formal decision context and $D_1, D_2, \ldots, D_l$ be nonempty and finite class sets of D, where $D = D_1 \cup D_2 \cup \ldots \cup D_l$ and $D_r \cap D_j = \emptyset (1 \leq r, j \leq l, r \neq j)$. We call $G_i^D = G_i^{D_1} \cup G_i^{D_2} \cup \ldots \cup G_i^{D_l}$ is class-object set under the i-th cognitive state.

For brevity, we write G_i^D as G_i and the corresponding set of G_i is denoted by $\{G_i\} = \bigcup_{j=1}^{l} \{G_i^{D_j}\}$. Considering that the information will be updated by different classes, we initiate and learn concepts by different labels. For convenience, for any $D_j \subseteq D$, the subclass-object sets $G_1^{D_j}, G_2^{D_j}, \ldots, G_n^{D_j}$ with $G_1^{D_j} \subseteq G_2^{D_j} \subseteq \ldots \subseteq G_n^{D_j}$ are denoted by $\{G_t^{D_j}\}\uparrow$.

Property 6.3 Let (G, M, I, D, J) be a regular formal decision context, we have

$$\left\{G_t^D\right\}\uparrow = \{G_t^{D_1}\}\uparrow \cup \{G_t^{D_2}\}\uparrow \cup \ldots \cup \{G_t^{D_l}\}\uparrow . \tag{6.5}$$

Proof It is immediate from Definitions 6.6 and 6.9. □

From Definitions 6.7 and 6.8, and Property 6.1, the initial concepts can be constructed by condition-concept space and class-concept space in a regular formal decision context. Then, an object can be associated with a single label by the interaction between condition-concept space and class-concept space from Property 6.2. Property 6.3 means that a cognitive state can be decomposed into some cognitive sub-states by different categories in a regular formal decision context. Therefore, hereinafter we only discuss the situation under a cognitive sub-state D_j.

B. Concept-Cognitive Process

Considering the information on the object set G and the attribute set M will be updated as time goes by in the real world, we discuss that how the concept spaces are timely updated in a regular formal decision context.

Definition 6.10 Let (G, M, I, D, J) be a regular formal decision context, $G_{i-1}^{D_j}, G_i^{D_j}$ be two subclass-objects of $\{G_t^{D_j}\}\uparrow$ and M_{i-1}, M_i be attribute sets of $\{M_t\}\uparrow$. Denote $\Delta G_{i-1}^{D_j} = G_i^{D_j} - G_{i-1}^{D_j}$, $\Delta M_{i-1} = M_i - M_{i-1}$. Suppose

1) $\mathcal{F}_{D_j,i-1}^{M}: 2^{G_{i-1}^{D_j}} \to 2^{M_{i-1}}$, $\quad \mathcal{H}_{D_j,i-1}^{M}: 2^{M_{i-1}} \to 2^{G_{i-1}^{D_j}}$,

2) $\mathcal{F}_{D_j,i-1}^{D}: 2^{G_{i-1}^{D_j}} \to 2^{D}$, $\quad \mathcal{H}_{D_j,i-1}^{D}: 2^{D} \to 2^{G_{i-1}^{D_j}}$,

3) $\mathcal{F}_{D_j,\Delta G_{i-1}^{D_j}}^{M}: 2^{\Delta G_{i-1}^{D_j}} \to 2^{M_{i-1}}$, $\quad \mathcal{H}_{D_j,\Delta G_{i-1}^{D_j}}^{M}: 2^{M_{i-1}} \to 2^{\Delta G_{i-1}^{D_j}}$,

4) $\mathcal{F}_{D_j,\Delta G_{i-1}^{D_j}}^{D}: 2^{\Delta G_{i-1}^{D_j}} \to 2^{D}$, $\quad \mathcal{H}_{D_j,\Delta G_{i-1}^{D_j}}^{D}: 2^{D} \to 2^{\Delta G_{i-1}^{D_j}}$,

5) $\mathcal{F}_{D_j,\Delta M_{i-1}}^{M}: 2^{G_i^{D_j}} \to 2^{\Delta M_{i-1}}$, $\quad \mathcal{H}_{D_j,\Delta M_{i-1}}^{M}: 2^{\Delta M_{i-1}} \to 2^{G_i^{D_j}}$,

6) $\mathcal{F}_{D_j,i}^{M}: 2^{G_i^{D_j}} \to 2^{M_i}$, $\quad \mathcal{H}_{D_j,i}^{M}: 2^{M_i} \to 2^{G_i^{D_j}}$,

7) $\mathcal{F}_{D_j,i}^{D}: 2^{G_i^{D_j}} \to 2^{D}$, $\quad \mathcal{H}_{D_j,i}^{D}: 2^{D} \to 2^{G_i^{D_j}}$

are seven pairs of cognitive operators in a regular formal decision context satisfying the following properties:

$$\mathcal{F}_{D_j,i}^{M}(g) = \begin{cases} \mathcal{F}_{D_j,i-1}^{M}(g) \cup \mathcal{F}_{D_j,\Delta M_{i-1}}^{M}(g), & \text{if } g \in G_{i-1}^{D_j}, \\ \mathcal{F}_{D_j,\Delta G_{i-1}^{D_j}}^{M}(g) \cup \mathcal{F}_{D_j,\Delta M_{i-1}}^{M}(g), & \text{otherwise,} \end{cases} \tag{6.6}$$

$$\mathcal{H}_{D_j,i}^{M}(m) = \begin{cases} \mathcal{H}_{D_j,i-1}^{M}(m) \cup \mathcal{H}_{D_j,\Delta G_{i-1}^{D_j}}^{M}(m), & \text{if } m \in M_{i-1}, \\ \mathcal{H}_{D_j,\Delta M_{i-1}}^{M}(m), & \text{otherwise,} \end{cases} \tag{6.7}$$

$$\mathcal{F}^D_{D_j,i}(y) = \begin{cases} \mathcal{F}^D_{D_j,i-1}(y), & \text{if } y \in G^{D_j}_{i-1}, \\ \mathcal{F}^D_{D_j,\Delta G^{D_j}_{i-1}}(y), & \text{otherwise}, \end{cases} \tag{6.8}$$

$$\mathcal{H}^D_{D_j,i}(z) = \mathcal{H}^D_{D_j,i-1}(z) \cup \mathcal{H}^D_{D_j,\Delta G^{D_j}_{i-1}}(z), \text{ if } z \in D, \tag{6.9}$$

where $\mathcal{F}^M_{D_j,\Delta G^{D_j}_{i-1}}(g)$, $\mathcal{H}^M_{D_j,\Delta G^{D_j}_{i-1}}(m)$ and $\mathcal{H}^D_{D_j,\Delta G^{D_j}_{i-1}}(z)$ are set to be empty when $\Delta G^{D_j}_{i-1} = \emptyset$, and $\mathcal{F}^M_{D_j,\Delta M_{i-1}}(g)$ and $\mathcal{H}^M_{D_j,\Delta M_{i-1}}(m)$ are set to be empty when $\Delta M_{i-1} = \emptyset$.

Then we say that $\mathcal{F}^M_{D_j,i}, \mathcal{F}^D_{D_j,i}$ and $\mathcal{H}^M_{D_j,i}, \mathcal{H}^D_{D_j,i}$ are respectively extended cognitive operators of $\mathcal{F}^M_{D_j,i-1}, \mathcal{F}^D_{D_j,i-1}$ and $\mathcal{H}^M_{D_j,i-1}, \mathcal{H}^D_{D_j,i-1}$ with the newly input data $\Delta G^{D_j}_{i-1}$ and ΔM_{i-1}. For convenience, cognitive operators $\mathcal{F}^M_{D,i}$ and $\mathcal{H}^M_{D,i}$ denote the combination of $\mathcal{F}^M_{D_1,i}, \mathcal{F}^M_{D_2,i}, \ldots, \mathcal{F}^M_{D_l,i}$ and $\mathcal{H}^M_{D_1,i}, \mathcal{H}^M_{D_2,i}, \ldots, \mathcal{H}^M_{D_l,i}$, respectively. Similarly, we can define $\mathcal{F}^D_{D,i}$ and $\mathcal{H}^D_{D,i}$.

Meanwhile, for any $D_j \subseteq D$, $\mathcal{G}^C_{\mathcal{F}^M_{D_j,i-1},\mathcal{H}^M_{D_j,i-1}}$ means subcondition-concept space under cognitive operators $\mathcal{F}^M_{D_j,i-1}$ and $\mathcal{H}^M_{D_j,i-1}$, and $\mathcal{G}^C_{\mathcal{F}^M_{D,i-1},\mathcal{H}^M_{D,i-1}}$ is called as condition-concept space under cognitive operators $\mathcal{F}^M_{D,i-1}$ and $\mathcal{H}^M_{D,i-1}$. In a similar manner, we can define $\mathcal{G}^D_{\mathcal{F}^D_{D,i-1},\mathcal{H}^D_{D,i-1}}$ and $\mathcal{G}^D_{\mathcal{F}^D_{D_j,i-1},\mathcal{H}^D_{D_j,i-1}}$. In $\mathcal{G}^C_{\mathcal{F}^M_{D,i-1},\mathcal{H}^M_{D,i-1}}$, we can obtain the k-th granular concept $\left(A^{D_j}_{G,k}, B^{D_j}_{G,k}\right)$ from $\mathcal{G}^C_{\mathcal{F}^M_{D,i-1},\mathcal{H}^M_{D,i-1}}$ with a class set D_j. Moreover, for dynamic information $\Delta G^{D_j}_{i-1}$, we write $\mathcal{G}^C_{\mathcal{F}^M_{D_j,\Delta G^{D_j}_{i-1}},\mathcal{H}^M_{D_j,\Delta G^{D_j}_{i-1}}}$ and $\mathcal{G}^D_{\mathcal{F}^D_{D_j,\Delta G^{D_j}_{i-1}},\mathcal{H}^D_{D_j,\Delta G^{D_j}_{i-1}}}$ as $\mathcal{G}^C_{\Delta G^{D_j}_{i-1}}$ and $\mathcal{G}^D_{\Delta G^{D_j}_{i-1}}$ $\left(\text{Similarly, } \mathcal{G}^C_{\Delta M_{i-1}} \text{ and } \mathcal{G}^D_{\Delta M_{i-1}} \text{ for } \Delta M_{i-1}\right)$ under operators $\mathcal{F}^M_{D_j,\Delta G^{D_j}_{i-1}}, \mathcal{H}^M_{D_j,\Delta G^{D_j}_{i-1}}$ and $\mathcal{F}^D_{D_j,\Delta G^{D_j}_{i-1}}, \mathcal{H}^D_{D_j,\Delta G^{D_j}_{i-1}}$ $\left(\mathcal{F}^M_{D_j,\Delta M_{i-1}}, \mathcal{H}^M_{D_j,\Delta M_{i-1}} \text{ and } \mathcal{F}^D_{D_j,\Delta M_{i-1}}, \mathcal{H}^D_{D_j,\Delta M_{i-1}}\right)$.

In theory, although we can update concepts by objects and attributes simultaneously, we are extremely interested in the new object information because the attributes can be regarded as relatively stable under certain conditions.

Theorem 6.1 *Let $G^{D_j}_i$ be a subclass-object set under a set D_j and $\left(\mathcal{G}_{\mathcal{F}_{D_j,i-1},\mathcal{H}_{D_j,i-1}}, \mathcal{F}^M_{D_j,\Delta G^{D_j}_{i-1}}, \mathcal{F}^D_{D_j,\Delta G^{D_j}_{i-1}}, \mathcal{H}^M_{D_j,\Delta G^{D_j}_{i-1}}, \mathcal{H}^D_{D_j,\Delta G^{D_j}_{i-1}}\right)$ be an object-oriented cognitive computing state, where $\mathcal{G}_{\mathcal{F}_{D_j,i-1},\mathcal{H}_{D_j,i-1}}$ is the concept space under cognitive operators $\mathcal{F}_{D_j,i-1}$ and $\mathcal{H}_{D_j,i-1}$. Then the following statements hold:*

1) *For any* $g \in G_i^{D_j}$, *if* $g \in G_{i-1}^{D_j}$, *then*

$$\big(\mathcal{H}^M_{D_j,i}\mathcal{F}^M_{D_j,i}(g), \mathcal{F}^M_{D_j,i}(g)\big) = \big(\mathcal{H}^M_{D_j,i-1}\mathcal{F}^M_{D_j,i-1}(g) \cup \mathcal{H}^M_{D_j,\Delta G_{i-1}^{D_j}}\mathcal{F}^M_{D_j,i-1}(g), \mathcal{F}^M_{D_j,i-1}(g)\big);$$

otherwise,

$$\big(\mathcal{H}^M_{D_j,i}\mathcal{F}^M_{D_j,i}(g), \mathcal{F}^M_{D_j,i}(g)\big) = \big(\mathcal{H}^M_{D_j,i-1}\mathcal{F}^M_{D_j,\Delta G_{i-1}^{D_j}}(g) \cup \mathcal{H}^M_{D_j,\Delta G_{i-1}^{D_j}}\mathcal{F}^M_{D_j,\Delta G_{i-1}^{D_j}}(g), \mathcal{F}^M_{D_j,\Delta G_{i-1}^{D_j}}(g)\big).$$

2) *For any* $m \in M_{i-1}$, *we have*

$$\big(\mathcal{H}^M_{D_j,i}(m), \mathcal{F}^M_{D_j,i}\mathcal{H}^M_{D_j,i}(m)\big) = \big(\mathcal{H}^M_{D_j,i-1}(m) \cup \mathcal{H}^M_{D_j,\Delta G_{i-1}^{D_j}}(m), \mathcal{F}^M_{D_j,i-1}\mathcal{H}^M_{D_j,i-1}(m) \cap \mathcal{F}^M_{D_j,\Delta G_{i-1}^{D_j}}\mathcal{H}^M_{D_j,\Delta G_{i-1}^{D_j}}(m)\big).$$

3) *For any* $y \in G_i^{D_j}$, *if* $y \in G_{i-1}^{D_j}$, *then*

$$\big(\mathcal{H}^D_{D_j,i}\mathcal{F}^D_{D_j,i}(y), \mathcal{F}^D_{D_j,i}(y)\big) = \big(\mathcal{H}^D_{D_j,i-1}\mathcal{F}^D_{D_j,i-1}(y) \cup \mathcal{H}^D_{D_j,\Delta G_{i-1}^{D_j}}\mathcal{F}^D_{D_j,i-1}(y), \mathcal{F}^D_{D_j,i-1}(y)\big);$$

otherwise,

$$\big(\mathcal{H}^D_{D_j,i}\mathcal{F}^D_{D_j,i}(y), \mathcal{F}^D_{D_j,i}(y)\big) = \big(\mathcal{H}^D_{D_j,i-1}\mathcal{F}^D_{D_j,\Delta G_{i-1}^{D_j}}(y) \cup \mathcal{H}^D_{D_j,\Delta G_{i-1}^{D_j}}\mathcal{F}^D_{D_j,\Delta G_{i-1}^{D_j}}(y), \mathcal{F}^D_{D_j,\Delta G_{i-1}^{D_j}}(y)\big).$$

4) *For any* $z \in D$, *we obtain*

$$\big(\mathcal{H}^D_{D_j,i}(z), \mathcal{F}^D_{D_j,i}\mathcal{H}^D_{D_j,i}(z)\big) = \big(\mathcal{H}^D_{D_j,i-1}(z) \cup \mathcal{H}^D_{D_j,\Delta G_{i-1}^{D_j}}(z), \mathcal{F}^D_{D_j,i-1}\mathcal{H}^D_{D_j,i-1}(z) \cap \mathcal{F}^D_{D_j,\Delta G_{i-1}^{D_j}}\mathcal{H}^D_{D_j,\Delta G_{i-1}^{D_j}}(z)\big).$$

Proof The proof of Theorem 6.1 can be found in the original paper [58]. □

From Theorem 6.1, we observe that the i-th concept space can be constructed under cognitive operators $\mathcal{F}_{D_j,i-1}$ and $\mathcal{H}_{D_j,i-1}$, and the concept space $\mathcal{G}_{\mathcal{F}_{D_j,i},\mathcal{H}_{D_j,i}}$

can be obtained by $\mathcal{G}_{\mathcal{F}_{D_j,i-1},\mathcal{H}_{D_j,i-1}}$ with the newly input data $\Delta G_{i-1}^{D_j}$. This means that we can obtain concepts based on the past concepts rather than reconstructing them from the beginning.

However, in the previous discussions, we still do not know which class-object set $G_{i-1}^{D_*}$ should be theoretically updated with $\Delta G_{i-1}^{D_j}$. In other words, although we obtain class-object set $G_{i-1}^{D_j}$ which will be actually updated by $\Delta G_{i-1}^{D_j}$, we are not sure if the updated class-object set in the model is in accordance with $G_{i-1}^{D_j}$. Thus, we will further discuss the relationship between $G_{i-1}^{D_j}$ and $G_{i-1}^{D_*}$.

Definition 6.11 Let $(\mathcal{HF}(g), \mathcal{F}(g))$ be a granular concept, for any $(A_{G,e}, B_{G,e}) \in \mathcal{G}^C$, where $e \in \{1, 2, \ldots, |\mathcal{G}^C|\}$. Then we can define concept-similarity degree (CS) as follows:

$$\theta_{CS} = CS(\mathcal{F}(g), B_{G,e}) = \frac{W_p \cdot M^T}{|\mathcal{F}(g) \cup B_{G,e}|}, \tag{6.10}$$

where M^T is the transpose of the vector M, and $W_p = (w_1, w_2, \ldots, w_m)$ is a cognitive weight vector that is associated with an attribute vector $M = (m_1, m_2, \ldots, m_m)$ consisting of (1) the elements from $\mathcal{F}(g) \cap B_{G,e}$ which are all set to be 1, and (2) the elements from $M - (\mathcal{F}(g) \cap B_{G,e})$ which are all set to be 0.

Let E be training times. For any $t \in E$, the cognitive weight vector of the t-th training is denoted by $W_{i,p}^t = (w_{i,1}^t, w_{i,2}^t, \ldots, w_{i,m}^t)$. Then we denote

$$\begin{bmatrix} W_{1,p}^t \\ \vdots \\ W_{n,p}^t \end{bmatrix} = \begin{bmatrix} w_{1,1}^t & \cdots & w_{1,m}^t \\ \vdots & \ldots & \vdots \\ w_{n,1}^t & \cdots & w_{n,m}^t \end{bmatrix}, \tag{6.11}$$

where $n = \left|\bigcup_{i=1}^{n}\{G_i\}\right|$. Our purpose is to obtain an optimal cognitive weight vector $W_{n,p}^t$ by computing concept-similarity degree vectors.

Definition 6.12 Let $\{G_{i-1}\}$ be a class-object set under G_{i-1}^D, $\Delta G_{i-1}^{D_*}$ be a new object set under D_*, and $G_{i-1}^{D_j}$ and $G_{i-1}^{D_r}$ be class-object sets under class sets D_j and D_r ($D_j \cap D_r = \emptyset$), respectively. For any granular concept $\left(A_{G,e}^{D_j}, B_{G,e}^{D_j}\right) \in \mathcal{G}^C_{\mathcal{F}^M_{D_j,i-1},\mathcal{H}^M_{D_j,i-1}}$ and a new granular concept $\left(\mathcal{H}^M_{D_*,\Delta G_{i-1}^{D_*}}\mathcal{F}^M_{D_*,\Delta G_{i-1}^{D_*}}(g), \mathcal{F}^M_{D_*,\Delta G_{i-1}^{D_*}}(g)\right)$, the degree of similarity between the concepts is defined as $CS_{i-1}^{D_j} = CS\left(B_{G,e}^{D_j}, \mathcal{F}^M_{D_*,\Delta G_{i-1}^{D_*}}(g)\right)$. Then, we denote

$$MCS^{D_j} = \max_{e=1}^{n}\left(CS_{i-1}^{D_j}\right) = \max_{e=1}^{n}\left(CS\left(B_{G,e}^{D_j}, \mathcal{F}^M_{D_*,\Delta G_{i-1}^{D_*}}(g)\right)\right),$$

where $n = \left|\mathcal{G}^{C}_{\mathcal{F}^{M}_{D_j,i-1},\mathcal{H}^{M}_{D_j,i-1}}\right|$. Then, we further denote

$$MMCS^{D_j} = \max_{j=1}^{l}\left(MCS^{D_j}\right). \tag{6.12}$$

From (6.12), we know that the subclass-object set $G^{D_j}_{i-1}$ should be updated in the class-object set G^{D}_{i-1}. Therefore, if $D_* = D_j$, it means that the theoretically updated subclass-object set $G^{D_*}_{i-1}$ is in accordance with the actually updated subclass-object set $G^{D_j}_{i-1}$. Otherwise, we should adjust cognitive weight vectors as follows.

$$\begin{aligned} &w_i^t \leftarrow w_i^t \pm \Delta w_i^t, \\ &\Delta w_i^t = activationFunction(\eta w_i^t), \end{aligned} \tag{6.13}$$

where the operator $+$ is adopted when the attributes are from $B^{D_j}_{G,e} \bigcap \mathcal{F}^{M}_{D_* \Delta G^{D_*}_{i-1}}(g)$ and the another operator $-$ is used for the elements from $B^{D_r}_{G,e} \bigcap \mathcal{F}^{M}_{D_* \Delta G^{D_*}_{i-1}}(g)$, and $activationFunction(\eta w_i^t) = \frac{exp(\eta w_i^t) - exp(-\eta w_i^t)}{exp(\eta w_i^t) + exp(-\eta w_i^t)}$ with the learning rate $\eta \in (0, 1)$.

6.1.1.3 Proposed Model

In this section, based on the above discussion, we put forward a CCLM with dynamic learning, which can perform a good performance in incremental learning and classification task.

A. Initial Concept Learning

We split raw data into training data $\overline{G}$ and testing data $\overline{\overline{G}}$. For the training data, let $\{\overline{G}\}$ be the set of the objects sets $\overline{G}_1, \overline{G}_2, \ldots, \overline{G}_n$ with $\overline{G}_i \cap \overline{G}_j = \emptyset (i \neq j)$, we denote

$$\{\overline{G}\} = \bigcup_{i=1}^{n}\{\overline{G}_i\}. \tag{6.14}$$

Here, $\overline{G}_1$ is an initial training data, and the rest of training data $\bigcup\limits_{i=2}^{n}\{\overline{G}_i\}$ is used for concept cognition.

From Definitions 6.7 and 6.8, the initial concept learning consists of two parts: constructing condition-concept space and decision-concept space. The details are shown in Algorithm 6.1, and its time complexity is $O\big(|\{\overline{G}_1\}|(|M|+|D|+|\overline{G}_1^{D_j}|)\big)$.

Algorithm 6.1 Initial concept learning

```
1: **Input:** the initial training data set $\overline{G}_1$.
2: **Output:** the initial concept space.
3: **for** each $\overline{G}_1^{D_j} \in \{\overline{G}_1\}$ **do**
4:     **for** each $m \in M$ **do**
5:         $\mathcal{G}^C_{\mathcal{F}^M_{D,1},\mathcal{H}^M_{D,1}} \leftarrow \big(\mathcal{H}^M_{D_j,1}(m), \mathcal{F}^M_{D_j,1}\mathcal{H}^M_{D_j,1}(m)\big)$
6:     **end for**
7:     **for** each $g \in \overline{G}_1^{D_j}$ **do**
8:         $\mathcal{G}^C_{\mathcal{F}^M_{D,1},\mathcal{H}^M_{D,1}} \leftarrow \big(\mathcal{H}^M_{D_j,1}\mathcal{F}^M_{D_j,1}(g), \mathcal{F}^M_{D_j,1}(g)\big)$
9:     **end for**
10:    **for** each $z \in D$ **do**
11:        $\mathcal{G}^D_{\mathcal{F}^D_{D,1},\mathcal{H}^D_{D,1}} \leftarrow \big(\mathcal{H}^D_{D_j,1}(z), \mathcal{F}^D_{D_j,1}\mathcal{H}^D_{D_j,1}(z)\big)$
12:    **end for**
13:    **for** each $y \in \overline{G}_1^{D_j}$ **do**
14:        $\mathcal{G}^D_{\mathcal{F}^D_{D,1},\mathcal{H}^D_{D,1}} \leftarrow \big(\mathcal{H}^D_{D_j,1}\mathcal{F}^D_{D_j,1}(y), \mathcal{F}^D_{D_j,1}(y)\big)$
15:    **end for**
16: **end for**
17: **Return** $\mathcal{G}^C_{\mathcal{F}^M_{D,1},\mathcal{H}^M_{D,1}}$ and $\mathcal{G}^D_{\mathcal{F}^D_{D,1},\mathcal{H}^D_{D,1}}$
```

B. Concept-Cognitive Process

Let E, err_0, W, AW, IW be the training epochs, learning error rate, cognitive weight vector, active weight vector and inhibited weight vector, respectively. It should be pointed out that AW and IW are to enhance and weaken the corresponding attributes, respectively. Based on the theory in Sect. 6.1.1 Theoretical Foundation, the concept-cognitive process can be briefly represented as follows:

Firstly, construct a conditional granular concept $\big(A^{D_*}_{G,k}, B^{D_*}_{G,k}\big)$ and a decision granular concept $\big(Y^{D_*}_{G,k}, Z^{D_*}_{G,k}\big)$.

Secondly, for a new concept $\big(A^{D_*}_{G,k}, B^{D_*}_{G,k}\big)$, we compute its concept-similarity degree with each granular concept $\big(A^{D_j}_{G,e}, B^{D_j}_{G,e}\big)$ from $\mathcal{G}^C_{\mathcal{F}^M_{D,i-1},\mathcal{H}^M_{D,i-1}}$.

Thirdly, if the predicted label is not in accordance with the actual label, the weight vectors W, AW and IW will be updated.

Finally, for the first training, we will update the condition-concept space and decision-concept space by dynamic concepts $\big(A^{D_*}_{G,k}, B^{D_*}_{G,k}\big)$ and $\big(Y^{D_*}_{G,k}, Z^{D_*}_{G,k}\big)$, respectively. Using recursive approach, we can obtain a final cognitive weight vector

$W^{E}_{n,p}$ and a final concept space (i.e., the condition-concept space $\mathcal{G}^{C}_{\mathcal{F}^{M}_{D,n},\mathcal{H}^{M}_{D,n}}$ and decision-concept space $\mathcal{G}^{D}_{\mathcal{F}^{D}_{D,n},\mathcal{H}^{D}_{D,n}}$).

The details of concept-cognitive process are shown in Algorithm 6.2. Note that params[1], params[2] and params[3] are $\Big((A^{D*}_{G,k}, B^{D*}_{G,k}), \mathcal{G}^{C}_{\mathcal{F}^{M}_{D,i-1},\mathcal{H}^{M}_{D,i-1}}, \mathcal{G}^{D}_{\mathcal{F}^{D}_{D,i-1},\mathcal{H}^{D}_{D,i-1}}, W^{t-1}_{i-1,p}\Big)$, $\Big(\eta, j, type, B^{D_{type}}_{G,k}, \theta L_{max}[|D|], W^{t-1}_{i-1,p}, AW^{t-1}_{i-1,p}, IW^{t-1}_{i-1,p}\Big)$ and $\Big(\mathcal{G}^{C}_{\mathcal{F}^{M}_{D,i-1},\mathcal{H}^{M}_{D,i-1}}, \mathcal{G}^{D}_{\mathcal{F}^{D}_{D,i-1},\mathcal{H}^{D}_{D,i-1}}, (A^{D_{type}}_{G,k}, B^{D_{type}}_{G,k}), (Y^{D_{type}}_{G,k}, Z^{D_{type}}_{G,k})\Big)$, respectively.

Now, we analyze the time complexity of Algorithm 6.2. Running Step 18 takes $O(1)$ because of updating objects one by one in CCLM. In Step 20, it will revoke Algorithm 6.3, and the running time is decided by two for loops. Thus, running Steps 18–26 takes $O\Big(|\mathcal{G}^{C}_{\mathcal{F}^{M}_{D,i-1},\mathcal{H}^{M}_{D,i-1}}||\mathcal{G}^{D}_{\mathcal{F}^{D}_{D,i-1},\mathcal{H}^{D}_{D,i-1}}|\Big)$, where $|\mathcal{G}^{D}_{\mathcal{F}^{D}_{D,i-1},\mathcal{H}^{D}_{D,i-1}}|$ is the number of $|D|$ and often very small. For Steps 27–32, it will call Algorithms 6.4 and 6.5. Therefore, the time complexity of Steps 27–32 is $O\Big(|D|\big((|activeSet| + |inhibitSet|) + (|M| + |D|)\big)\Big)$. To sum up, the time complexity of Algorithm 6.2 is $O\Big(P|\bigcup_{i=2}^{n}\{\overline{G}_i\}|\,(|\overline{G}^{D*}_{i}| + |\mathcal{G}^{C}_{\mathcal{F}^{M}_{D,i-1},\mathcal{H}^{M}_{D,i-1}}||D| + Q)\Big)$ $\big(P = max\{E, E_{err_0}\}, Q = (|M| + |D|)(|D| + 1) + |D|(|activeSet| + |inhibitSet|)\big)$, where E is the number of training epochs and E_{err_0} is the running times about err_0.

C. Overall Procedure and Concept Prediction

Figure 6.1 shows the overall procedure of CCLM which includes three stages: initial concept generation, concept-cognitive process and concept prediction. Suppose there are still three classes to predict. The stage of initial concept generation is to generate concept space by mapping objects into concepts, and then the second stage will update the concept space by the concept-similarity degree with labeled data.

In the stage of concept prediction, for any test instance, concept-similarity degree is further used to compute similarity degree, and then the final prediction will be completed by the sum of the maximum class vector as shown in the right of Fig. 6.1. Note that, compared with the second stage, the concept space will not be updated in the third stage.

Based on the final concept space $\mathcal{G}^{C}_{\mathcal{F}^{M}_{D,n},\mathcal{H}^{M}_{D,n}}$, $\mathcal{G}^{D}_{\mathcal{F}^{D}_{D,n},\mathcal{H}^{D}_{D,n}}$, and the final weight vector $W^{E}_{n,p}$, we can make predictions in $\overline{\overline{G}}$. The details are described in Algorithm 6.6. Considering that running Step 6 will revoke the function of Algorithm 6.3, it is easy to verify that the time complexity of Algorithm 6.6 is $O\big(|\overline{\overline{G}}||\mathcal{G}^{C}_{\mathcal{F}^{M}_{D,i-1},\mathcal{H}^{M}_{D,i-1}}||D|\big)$.

Algorithm 6.2 Concept-cognitive process

```
1: Input: initial concept spaces $\mathcal{G}^{C}_{\mathcal{F}^M_{D,1},\mathcal{H}^M_{D,1}}$ and $\mathcal{G}^{D}_{\mathcal{F}^D_{D,1},\mathcal{H}^D_{D,1}}$.
2: Output: a final concept space and a final weight vector.
3: Initialize $W^1_{1,p}$, $AW^1_{1,p}$, $IW^1_{1,p}$, $\eta$, $err_0$, and $E$.
4: while $t \le E || err_{min} \le err_0$ do                                ▷ Initialize t=2.
5:     for each $\overline{G}_i^{D*} \in \bigcup_{i=2}^{n}\{\overline{G}_i\}$ do
6:         for each $m \in M$ do
7:             $\mathcal{G}^{C}_{\overline{G}_i^{D*}} \leftarrow (\mathcal{H}^M_{D*,i}(m), \mathcal{F}^M_{D*,i}\mathcal{H}^M_{D*,i}(m))$
8:         end for
9:         for each $g \in \overline{G}_i^{D*}$ do
10:            $\mathcal{G}^{C}_{\overline{G}_i^{D*}} \leftarrow (\mathcal{H}^M_{D*,i}\mathcal{F}^M_{D*,i}(g), \mathcal{F}^M_{D*,i}(g))$
11:        end for
12:        for each $z \in D$ do
13:            $\mathcal{G}^{D}_{\overline{G}_i^{D*}} \leftarrow (\mathcal{H}^D_{D*,i}(z), \mathcal{F}^D_{D*,i}\mathcal{H}^D_{D*,i}(z))$
14:        end for
15:        for each $y \in \overline{G}_i^{D*}$ do
16:            $\mathcal{G}^{D}_{\overline{G}_i^{D*}} \leftarrow (\mathcal{H}^D_{D*,i}\mathcal{F}^D_{D*,i}(y), \mathcal{F}^D_{D*,i}(y))$
17:        end for
18:        for each$(A^{D*}_{G,k}, B^{D*}_{G,k}) \in \mathcal{G}^{C}_{\overline{G}_i^{D*}}$ do
19:            Get a concept $(Y^{D*}_{G,k}, Z^{D*}_{G,k}) \in \mathcal{G}^{D}_{\overline{G}_i^{D*}}$.
20:            Get concept-similarity degrees by Algorithm 6.3.
21:            $\theta_{max}[index] \leftarrow max(\theta_{max}[|D|])$
22:            type ← indexType$(Y^{D*}_{G,k})$                              ▷ Get a label.
23:            if index ≠ type then
24:                errFunction$_i(A^{D*}_{G,k})$=1                            ▷ Misclassification.
25:            end if
26:        end for
27:        for j=0 to $|D|$ do
28:            if $\theta_{max}[j] \ge \theta_{max}[type]$ then
29:                Update weight vector by Algorithm 6.4.
30:            end if
31:            Update concept space by Algorithm 6.5.
32:        end for
33:    end for
34:    $err_{min} \leftarrow$ minimize$\left(\frac{\sum_{i=2}^{n} \text{errFunction}_i (A^{D*}_{G,k})}{|\bigcup_{i=2}^{n}\overline{G}_i|}\right)$
35:    $++t$
36: end while
37: Return $\mathcal{G}^{C}_{\mathcal{F}^M_{D,n},\mathcal{H}^M_{D,n}}$, $\mathcal{G}^{D}_{\mathcal{F}^D_{D,n},\mathcal{H}^D_{D,n}}$ and $W^E_{n,p}$.
```

Algorithm 6.3 Concept-similarity degree

```
1: function GETCONCEPTSIMILARITY(params[1])
2:     Initialize $\theta_{max}[|D|], \theta L_{max}[|D|]$
3:     for $\left(A_{G,e}^{D_j}, B_{G,e}^{D_j}\right) \in \mathcal{G}_{\mathcal{F}_{D,i-1}^M, \mathcal{H}_{D,i-1}^M}^C$ do
4:         for $\left(Y_{G,q}^{D_r}, Z_{G,q}^{D_r}\right) \in \mathcal{G}_{\mathcal{F}_{D,i-1}^D, \mathcal{H}_{D,i-1}^D}^D$ do
5:             if $A_{G,e}^{D_j} \subseteq Y_{G,q}^{D_r}$ then
6:                 $\theta_{CS_{i-1}}^{t-1} = CS_{i-1}\left(B_{G,k}^{D_*}, B_{G,e}^{D_j}\right)$
7:                 type $\leftarrow$ indexType$(Y_{G,q}^{D_r})$
8:                 if $\theta_{CS_{i-1}}^{t-1} \geq \theta_{max}[type]$ then
9:                     $\theta_{max}[type] = \theta_{CS_{i-1}}^{t-1}$
10:                    $\theta L_{max}[type] = B_{G,e}^{D_j}$
11:                end if
12:            end if
13:        end for
14:    end for
15:    return $\theta_{max}[|D|], \theta L_{max}[|D|]$
16: end function
```

Algorithm 6.4 Adjust weight

```
1: function ADJUSTWEIGHT(params[2])
2:     typeSet=$\theta L_*[type] \bigcap B_{G,k}^{type}$
3:     jSet=$\theta L_*[j] \bigcap B_{G,k}^{type}$
4:     activeSet=typeSet -jSet
5:     inhibitSet=jSet - typeSet
6:     while $m_1 \in$ activeSet do
7:         indexA=indexAttribtue$(m_1)$
8:         Update $AW_{i-1,p}^{t-1}$ by $aw_{i-1,indexA}^{t-1}++$.
9:         Update $W_{i-1,p}^{t-1}$ by (6.13).                    ▷ Input $\eta aw_{i-1,indexA}^{t-1}$.
10:    end while
11:    while $m_2 \in$ inhibitSet do
12:        indexB=indexAttribtue$(m_2)$
13:        Update $IW_{i-1,p}^{t-1}$ by $iw_{i-1,indexB}^{t-1}++$.
14:        Update $W_{i-1,p}^{t-1}$ by (6.13).                    ▷ Input $\eta iw_{i-1,indexB}^{t-1}$.
15:    end while
16:    return $W_{i-1,p}^{t-1}, AW_{i-1,p}^{t-1}, IW_{i-1,p}^{t-1}$
17: end function
```

Algorithm 6.5 Update concept space

```
1: function UPDATECONCEPTS(params[3])
2:     for each $m \in M$ do
3:         Update $\mathcal{G}^{C}_{\mathcal{F}^{M}_{D,i-1},\mathcal{H}^{M}_{D,i-1}}$ by Theorem 6.1.
4:     end for
5:     for each $z \in D$ do
6:         Update $\mathcal{G}^{D}_{\mathcal{F}^{D}_{D,i-1},\mathcal{H}^{D}_{D,i-1}}$ by Theorem 6.1.
7:     end for
8:     return $\mathcal{G}^{C}_{\mathcal{F}^{M}_{D,i-1},\mathcal{H}^{M}_{D,i-1}}$, $\mathcal{G}^{D}_{\mathcal{F}^{D}_{D,i-1},\mathcal{H}^{D}_{D,i-1}}$
9: end function
```

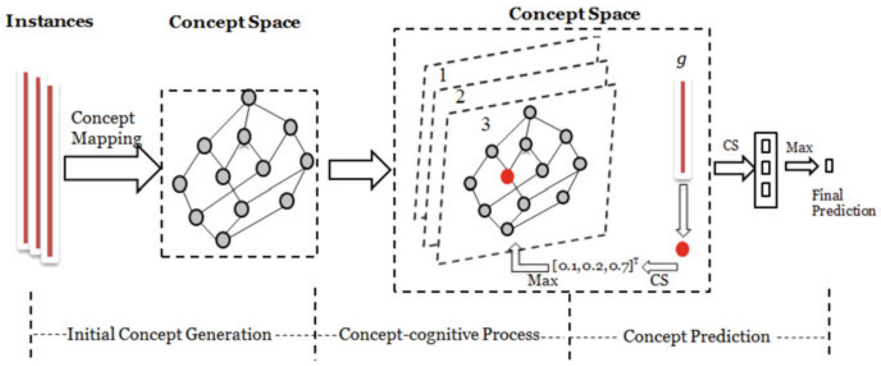


Fig. 6.1 Illustration of overall procedure for CCLM. Suppose there are three classes to predict, and the maximum class vector is obtained by concept-similarity degree

Algorithm 6.6 Concept prediction

```
1: Input: the testing data $\overline{\overline{G}}$, $W^{E}_{n,p}$, $\mathcal{G}^{C}_{\mathcal{F}^{M}_{D,n},\mathcal{H}^{M}_{D,n}}$, $\mathcal{G}^{D}_{\mathcal{F}^{D}_{D,n},\mathcal{H}^{D}_{D,n}}$.
2: Output: the class labels of test data.
3: for each $g_i \in \overline{\overline{G}}$ do
4:     $\left(A^{D_*}_{G,k}, B^{D_*}_{G,k}\right) \leftarrow \left(\mathcal{H}^{M}_{D_*,i}\mathcal{F}^{M}_{D_*,i}(g_i), \mathcal{F}^{M}_{D_*,i}(g_i)\right)$
5:     $\left(Y^{D_*}_{G,k}, Z^{D_*}_{G,k}\right) \leftarrow \left(\mathcal{H}^{D}_{D_*,i}\mathcal{F}^{D}_{D_*,i}(y_i), \mathcal{F}^{D}_{D_*,i}(y_i)\right)$
6:     $\theta_{max}[|D|] \leftarrow$ getConceptSimilarity(params[1])
7:     $\theta_{max}[index] \leftarrow \max(\theta_{max}[|D|])$
8:     type $\leftarrow$ indexType$\left(Y^{D_*}_{G,k}\right)$
9:     if index=type then
10:        $correctNum+ = 1$
11:    else
12:        $incorrectNum+ = 1$
13:    end if
14: end for
```

6.1.2 *Concurrent Concept-Cognitive Learning Model for Classification*

In this subsection, we discuss the design of a new theoretical framework for concurrent computing, which comprises three aspects: initial concurrent concept learning, the concurrent concept-cognitive process, and the concept generalization process.

6.1.2.1 Initial Concurrent Concept Learning in C3LM

In the real world, not all methods can be concurrent, as this often depends on their separability. In order to guarantee concurrency for the C3LM in theory, we need to consider the following definitions and propositions.

Definition 6.13 Let (G, M, I, D, J) be a regular formal decision context. Suppose that $D_1, D_2, \ldots, D_K$ is a partition of D by class labels, and let $G = G^{D_1} \cup G^{D_2} \cup \ldots \cup G^{D_K}$. Then, we say that G^{D_k} $(k \in \{1, 2, \ldots, K\})$ is a subclass-object set. For the sake of brevity, hereinafter we write G^{D_k} as G^k.

Definition 6.13 indicates that an object set G can be decomposed into several subclass-object sets in a regular formal decision context. Moreover, we only consider objects that are updated by newly input objects, as attributes can be taken as relatively stable in real life. Therefore, in the following, we discuss the scenario of a subclass-object G^k.

Let G^k be a subclass-object set, and M and D be attribute sets. The set-valued mappings $\mathcal{F}^k : 2^{G^k} \to 2^M$, $\mathcal{H}^k : 2^M \to 2^{G^k}$ and $\widetilde{\mathcal{F}}^k : 2^{G^k} \to 2^D$, $\widetilde{\mathcal{H}}^k : 2^D \to 2^{G^k}$ are respectively referred to as the conditional and decision cognitive operators with a subclass-object set G^k when no confusion exists.

Definition 6.14 Let $G_1^k, G_2^k, \ldots, G_n^k$ be a partition of an object set G^k. If the following cognitive operators:

$$\mathcal{F}_j^k : 2^{G_j^k} \to 2^M, \qquad \mathcal{H}_j^k : 2^M \to 2^{G_j^k}, \; j = 1, 2, \ldots, n,$$

$$\mathcal{F}^k : 2^{G^k} \to 2^M, \qquad \mathcal{H}^k : 2^M \to 2^{G^k}$$

satisfy $\mathcal{F}^k(g) = \mathcal{F}_j^k(g)$, where $g \in G_j^k$, we say that $\mathcal{HS}_{\mathcal{F}^k\mathcal{H}^k} = (\mathcal{F}_1^k, \ldots, \mathcal{F}_n^k; \mathcal{H}_1^k, \ldots, \mathcal{H}_n^k)$ is a conditional horizontal partition state.

Proposition 6.1 *Let* $\mathcal{HS}_{\mathcal{F}^k\mathcal{H}^k} = (\mathcal{F}_1^k, \ldots, \mathcal{F}_n^k; \mathcal{H}_1^k, \ldots, \mathcal{H}_n^k)$ *be a conditional horizontal partition state. For any* $g \in G_{j_1}^k$ $(j_1 \in \{1, 2, \ldots, n\})$*, if there exist objects* $g_1, g_2, \ldots, g_n \in G_{j_2}^k$ $(j_2 \in \{1, 2, \ldots, n\})$ *such that* $\mathcal{F}_{j_1}^k(g) \subseteq \mathcal{F}_{j_2}^k(g_i)$ $(i =$

$1, 2, \ldots, n$), *we have*

$$(\mathcal{H}^k\mathcal{F}^k(g), \mathcal{F}^k(g)) = \Big(\big\{g \cup (\bigcup_{i=1}^{n} g_i)\big\}, \mathcal{F}^k_{j_1}(g)\Big); \tag{6.15}$$

otherwise,

$$(\mathcal{H}^k\mathcal{F}^k(g), \mathcal{F}^k(g)) = (\{g\}, \mathcal{F}^k_{j_1}(g)). \tag{6.16}$$

Proof The proof of Proposition 6.1 can be found in the original paper [60]. □

In fact, from the perspective of objects, Definition 6.14 and Proposition 6.1 demonstrate that the separability holds for C3LM in the conditional formal context (G, M, I). Analogously, we can determine that the separability also holds for C3LM in the decision formal context (G, D, J) under the decision cognitive operators $\widetilde{\mathcal{F}}^k$ and $\widetilde{\mathcal{H}}^k$.

Definition 6.15 Let $M_1, M_2, \ldots, M_d$ be a partition of M. For any $G^k \subseteq G$, if the following cognitive operators:

$$\mathcal{F}^k_j : 2^{G^k} \to 2^{M_j}, \qquad \mathcal{H}^k_j : 2^{M_j} \to 2^{G^k}, j = 1, 2, \ldots, d,$$
$$\mathcal{F}^k : 2^{G^k} \to 2^{M}, \qquad \mathcal{H}^k : 2^{M} \to 2^{G^k}$$

satisfy $\mathcal{F}^k\mathcal{H}^k(m) = \bigcup_{j=1}^{d} \mathcal{F}^k_j\mathcal{H}^k(m)$ where $m \in M$, we say that $\mathcal{VS}_{\mathcal{F}^k\mathcal{H}^k} = (\mathcal{H}^k_1, \ldots, \mathcal{H}^k_d; \mathcal{F}^k_1, \ldots, \mathcal{F}^k_d)$ is a conditional vertical partition state.

Proposition 6.2 *Let* $\mathcal{VS}_{\mathcal{F}^k\mathcal{H}^k} = (\mathcal{H}^k_1, \ldots, \mathcal{H}^k_d; \mathcal{F}^k_1, \ldots, \mathcal{F}^k_d)$ *be a conditional vertical partition state. For any* $m \in M_{j_1}$ ($j_1 \in \{1, 2, \ldots, d\}$), *if there exist attributes* $m_1, m_2, \ldots, m_r \in M_{j_2}$ ($j_2 \in \{1, 2, \ldots, d\}$) *such that* $\mathcal{H}^k_{j_1}(m) \subseteq \mathcal{H}^k_{j_2}(m_i)$ ($i = 1, 2, \ldots, r$), *we have*

$$(\mathcal{H}^k(m), \mathcal{F}^k\mathcal{H}^k(m)) = \Big(\mathcal{H}^k(m), \big\{m \cup (\bigcup_{i=1}^{r} m_i)\big\}\Big); \tag{6.17}$$

otherwise,

$$(\mathcal{H}^k(m), \mathcal{F}^k\mathcal{H}^k(m)) = (\mathcal{H}^k(m), \{m\}). \tag{6.18}$$

Proof The proof of Proposition 6.2 can also be found in the original paper [60]. □

From Definition 6.15 and Proposition 6.2, we know that the separability holds for C3LM in the conditional formal context (G, M, I) from the attribute perspective. Similarly, under decision cognitive operators $\widetilde{\mathcal{F}}^k$ and $\widetilde{\mathcal{H}}^k$, there exists the same property for C3LM in the decision formal context (G, D, J).

Based on the above theory, we present an initial concurrent computing framework (see Fig. 6.2 for details) and its corresponding algorithm (see Algorithm 6.7)

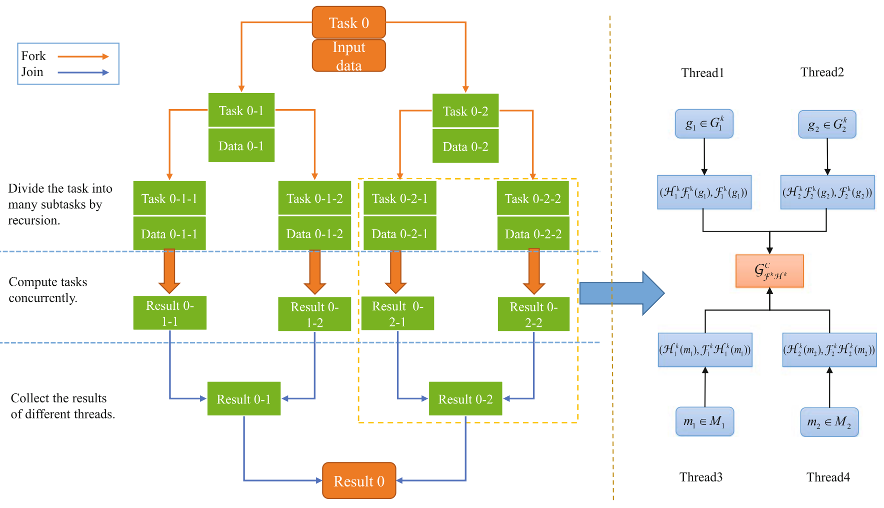


Fig. 6.2 Framework of constructing initial concepts in C3LM

Algorithm 6.7 Concurrent computation of initial concept space

```
1:  Input: Initial training dataset $G^k$ and chunk size.
2:  Output: The initial concept spaces $\mathcal{G}^C_{\mathcal{F}^k,\mathcal{H}^k}$ and $\mathcal{G}^D_{\widetilde{\mathcal{F}}^k,\widetilde{\mathcal{H}}^k}$.
3:  $n = \lceil |G^k|/\text{chunk-size} \rceil$, $d = \lceil |M|/\text{chunk-size} \rceil$, and $l = \lceil |D|/\text{chunk-size} \rceil$ are the numbers
    of threads for the objects, conditional attributes, and decision attributes, respectively.
4:  for $G^k_j = G^k_1$ to $G^k_n$ do in parallel
5:      for each $g \in G^k_j (j \in \{1, 2, \ldots, n\})$ do
6:          $\mathcal{G}^C_{\mathcal{F}^k,\mathcal{H}^k} \leftarrow \big(\mathcal{H}^k_j\mathcal{F}^k_j(g), \mathcal{F}^k_j(g)\big)$
7:      end for
8:  end for
9:  for $M_j = M_1$ to $M_d$ do in parallel
10:     for each $m \in M_j (j \in \{1, 2, \ldots, d\})$ do
11:         $\mathcal{G}^C_{\mathcal{F}^k,\mathcal{H}^k} \leftarrow \big(\mathcal{H}^k_j(m), \mathcal{F}^k_j\mathcal{H}^k_j(m)\big)$
12:     end for
13: end for
14: for $G^k_j = G^k_1$ to $G^k_n$ do in parallel
15:     for each $y \in G^k_j (j \in \{1, 2, \ldots, n\})$ do
16:         $\mathcal{G}^D_{\widetilde{\mathcal{F}}^k,\widetilde{\mathcal{H}}^k} \leftarrow \big(\widetilde{\mathcal{H}}^k_j\widetilde{\mathcal{F}}^k_j(y), \widetilde{\mathcal{F}}^k_j(y)\big)$
17:     end for
18: end for
19: for $D_j = D_1$ to $D_l$ do in parallel
20:     for each $z \in D_j (j \in \{1, 2, \ldots, l\})$ do
21:         $\mathcal{G}^D_{\widetilde{\mathcal{F}}^k,\widetilde{\mathcal{H}}^k} \leftarrow \big(\widetilde{\mathcal{H}}^k_j(z), \widetilde{\mathcal{F}}^k_j\widetilde{\mathcal{H}}^k_j(z)\big)$
22:     end for
23: end for
24: Return $\mathcal{G}^C_{\mathcal{F}^k,\mathcal{H}^k}$ and $\mathcal{G}^D_{\widetilde{\mathcal{F}}^k,\widetilde{\mathcal{H}}^k}$
```

for constructing the initial concepts. The overall process in Fig. 6.2 can be described as follows: first, a task can be divided into many subtasks by the recursion method, based on Definitions 6.14 and 6.15, and Propositions 6.1 and 6.2. Second, according to Propositions 6.1 and 6.2, threads can concurrently calculate the concepts of each task. Finally, the results of different threads will be collected by Propositions 6.1 and 6.2. Moreover, the right of Fig. 6.2 illustrates that four threads calculate granular concepts based on the object and attribute sets. It should be pointed out that the proposed C3LM is based on the fork/join framework.[1]

Furthermore, it is easy to determine that the time complexity of Algorithm 6.7 is $O(\frac{1}{n}|G^k| + \frac{1}{d}|M| + \frac{1}{l}|D|)$. For an object set G, by means of Algorithm 6.7, we can obtain the conditional concept space $\mathcal{G}^C_{\mathcal{F}\mathcal{H}}$ and decision concept space $\mathcal{G}^D_{\tilde{\mathcal{F}}\tilde{\mathcal{H}}}$.

6.1.2.2 Concurrent Concept-Cognitive Process in C3LM

In the real world, objects will be updated as time passes, which means that the obtained concept spaces need to be updated accordingly. For a person, learning is not simply a matter of acquiring a description, but involves taking something new and integrating it sufficiently with the existing thought processes [41]. The learning ability in humans is known as a gradual cognitive process. Therefore, in this subsection, we explore the concept-cognitive process under a concurrent environment.

As with the classical cognitive process [36], combining Definitions 6.6 and 6.13, we obtain the cognitive operators for C3LM with the newly input objects $\Delta G^k_{i-1} = G^k_i - G^k_{i-1}$, as follows:

$$\begin{array}{lll}
\text{(i)} & \mathcal{F}^k_{i-1} : 2^{G^k_{i-1}} \to 2^M, & \mathcal{H}^k_{i-1} : 2^M \to 2^{G^k_{i-1}}, \\
\text{(ii)} & \mathcal{F}^k_{\Delta G^k_{i-1}} : 2^{\Delta G^k_{i-1}} \to 2^M, & \mathcal{H}^k_{\Delta G^k_{i-1}} : 2^M \to 2^{\Delta G^k_{i-1}}, \\
\text{(iii)} & \mathcal{F}^k_i : 2^{G^k_i} \to 2^M, & \mathcal{H}^k_i : 2^M \to 2^{G^k_i},
\end{array} \tag{6.19}$$

and

$$\begin{array}{lll}
\text{(iv)} & \tilde{\mathcal{F}}^k_{i-1} : 2^{G^k_{i-1}} \to 2^D, & \tilde{\mathcal{H}}^k_{i-1} : 2^D \to 2^{G^k_{i-1}}, \\
\text{(v)} & \tilde{\mathcal{F}}^k_{\Delta G^k_{i-1}} : 2^{\Delta G^k_{i-1}} \to 2^D, & \tilde{\mathcal{H}}^k_{\Delta G^k_{i-1}} : 2^D \to 2^{\Delta G^k_{i-1}}, \\
\text{(vi)} & \tilde{\mathcal{F}}^k_i : 2^{G^k_i} \to 2^D, & \tilde{\mathcal{H}}^k_i : 2^D \to 2^{G^k_i}.
\end{array} \tag{6.20}$$

[1]https://docs.oracle.com/javase/tutorial/essential/concurrency/forkjoin.html.

Definition 6.16 Let $\Delta G_{i-1}^k = G_i^k - G_{i-1}^k$ be a singleton set with a new object, and $\mathcal{F}_{\Delta G_{i-1}^k}^k, \mathcal{H}_{\Delta G_{i-1}^k}^k$ and $\widetilde{\mathcal{F}}_{\Delta G_{i-1}^k}^k, \widetilde{\mathcal{H}}_{\Delta G_{i-1}^k}^k$ be cognitive operators. For any $g \in \Delta G_{i-1}^k$, if $\left(\mathcal{H}_{\Delta G_{i-1}^k}^k \mathcal{F}_{\Delta G_{i-1}^k}^k(g), \mathcal{F}_{\Delta G_{i-1}^k}^k(g)\right) = \left(\{g\}, \mathcal{F}_{\Delta G_{i-1}^k}^k(g)\right)$ and $\left(\widetilde{\mathcal{H}}_{\Delta G_{i-1}^k}^k \widetilde{\mathcal{F}}_{\Delta G_{i-1}^k}^k(g), \widetilde{\mathcal{F}}_{\Delta G_{i-1}^k}^k(g)\right) = \left(\{g\}, \widetilde{\mathcal{F}}_{\Delta G_{i-1}^k}^k(g)\right)$, $\left(\mathcal{H}_{\Delta G_{i-1}^k}^k \mathcal{F}_{\Delta G_{i-1}^k}^k(g), \mathcal{F}_{\Delta G_{i-1}^k}^k(g)\right)$ and $\left(\widetilde{\mathcal{H}}_{\Delta G_{i-1}^k}^k \widetilde{\mathcal{F}}_{\Delta G_{i-1}^k}^k(g), \widetilde{\mathcal{F}}_{\Delta G_{i-1}^k}^k(g)\right)$ are referred to as the newly formed conditional atomic concept and decision atomic concept, respectively, with a single object g.

In fact, we consider that the obtained concept spaces $\mathcal{G}_{\mathcal{F}^k\mathcal{H}^k}^C$ and $\mathcal{G}_{\widetilde{\mathcal{F}}^k\widetilde{\mathcal{H}}^k}^D$ are updated by a newly input object, rather than adding multiple objects simultaneously. For the sake of convenience, we denote the initial concept spaces obtained by Algorithm 6.7, namely $\mathcal{G}_{\mathcal{F}^k\mathcal{H}^k}^C, \mathcal{G}_{\widetilde{\mathcal{F}}^k\widetilde{\mathcal{H}}^k}^D$ and $\mathcal{G}_{\mathcal{F}\mathcal{H}}^C, \mathcal{G}_{\widetilde{\mathcal{F}}\widetilde{\mathcal{H}}}^D$, as $\mathcal{G}_{\mathcal{F}_0^k\mathcal{H}_0^k}^C, \mathcal{G}_{\widetilde{\mathcal{F}}_0^k\widetilde{\mathcal{H}}_0^k}^D$ and $\mathcal{G}_{\mathcal{F}_0\mathcal{H}_0}^C, \mathcal{G}_{\widetilde{\mathcal{F}}_0\widetilde{\mathcal{H}}_0}^D$, respectively. According to Eqs. (6.19) and (6.20), the cognitive operators $\mathcal{F}_i^k, \mathcal{H}_i^k$ and $\widetilde{\mathcal{F}}_i^k, \widetilde{\mathcal{H}}_i^k$ in the i-th period can be obtained by the cognitive operators $\mathcal{F}_{i-1}^k, \mathcal{H}_{i-1}^k$ and $\widetilde{\mathcal{F}}_{i-1}^k, \widetilde{\mathcal{H}}_{i-1}^k$ in the $(i-1)$-th period with incremental objects, respectively. Moreover, we denote their corresponding concept spaces by $\mathcal{G}_{\mathcal{F}_i^k\mathcal{H}_i^k}^C$ and $\mathcal{G}_{\widetilde{\mathcal{F}}_i^k\widetilde{\mathcal{H}}_i^k}^D$. Furthermore, the entire concept spaces in the i-th period are further denoted by $\mathcal{G}_{\mathcal{F}_i\mathcal{H}_i}^C$ and $\mathcal{G}_{\widetilde{\mathcal{F}}_i\widetilde{\mathcal{H}}_i}^D$.

Proposition 6.3 *Let* $\left(\mathcal{H}_{\Delta G_{i-1}^k}^k \mathcal{F}_{\Delta G_{i-1}^k}^k(g), \mathcal{F}_{\Delta G_{i-1}^k}^k(g)\right)$ *and* $\left(\widetilde{\mathcal{H}}_{\Delta G_{i-1}^k}^k \widetilde{\mathcal{F}}_{\Delta G_{i-1}^k}^k(g), \widetilde{\mathcal{F}}_{\Delta G_{i-1}^k}^k(g)\right)$ *be the newly formed conditional and decision atomic concepts, respectively. Then, the following statements hold:*

(i) For any granular concept $(A_{k,j}, B_{k,j}) \in \mathcal{G}_{\mathcal{F}_{i-1}^k\mathcal{H}_{i-1}^k}^C$ $(j \in \{1, 2, \ldots, |\mathcal{G}_{\mathcal{F}_{i-1}^k\mathcal{H}_{i-1}^k}^C|\})$, *if*

$$B_{k,j} \cap \mathcal{F}_{\Delta G_{i-1}^k}^k(g) \neq \emptyset, (A_{k,j}, B_{k,j}) = \left(A_{k,j} \cup \mathcal{H}_{\Delta G_{i-1}^k}^k \mathcal{F}_{\Delta G_{i-1}^k}^k(g), B_{k,j} \cap \mathcal{F}_{\Delta G_{i-1}^k}^k(g)\right);$$

otherwise,

$$\mathcal{G}_{\mathcal{F}_i^k\mathcal{H}_i^k}^C = \mathcal{G}_{\mathcal{F}_{i-1}^k\mathcal{H}_{i-1}^k}^C \cup \left(\mathcal{H}_{\Delta G_{i-1}^k}^k \mathcal{F}_{\Delta G_{i-1}^k}^k(g), \mathcal{F}_{\Delta G_{i-1}^k}^k(g)\right).$$

(ii) For any granular concept $(Y_{k,j}, Z_{k,j}) \in \mathcal{G}_{\widetilde{\mathcal{F}}_{i-1}^k\widetilde{\mathcal{H}}_{i-1}^k}^D$ $(j \in \{1, 2, \ldots, |\mathcal{G}_{\widetilde{\mathcal{F}}_{i-1}^k\widetilde{\mathcal{H}}_{i-1}^k}^D|\})$, *if*

$$Z_{k,j} \cap \widetilde{\mathcal{F}}_{\Delta G_{i-1}^k}^k(g) \neq \emptyset, (Y_{k,j}, Z_{k,j}) = \left(Y_{k,j} \cup \widetilde{\mathcal{H}}_{\Delta G_{i-1}^k}^k \widetilde{\mathcal{F}}_{\Delta G_{i-1}^k}^k(g), Z_{k,j} \cap \widetilde{\mathcal{F}}_{\Delta G_{i-1}^k}^k(g)\right);$$

otherwise,

$$\mathcal{G}_{\widetilde{\mathcal{F}}_i^k\widetilde{\mathcal{H}}_i^k}^D = \mathcal{G}_{\widetilde{\mathcal{F}}_{i-1}^k\widetilde{\mathcal{H}}_{i-1}^k}^D \cup \left(\widetilde{\mathcal{H}}_{\Delta G_{i-1}^k}^k \widetilde{\mathcal{F}}_{\Delta G_{i-1}^k}^k(g), \widetilde{\mathcal{F}}_{\Delta G_{i-1}^k}^k(g)\right).$$

Proof *The proof of Proposition 6.3 can be found in the original paper [60].* □

For any object g, according to Definitions 6.5 and 6.6, we can obtain a concept $\big(\{g\}, \mathcal{F}_{\Delta G_{i-1}}(g)\big)$, as the concept spaces are updated by adding objects sequentially. The concept similarity (CS) degree [58] is used in this study to explore the interaction of attributes in the concept-cognitive process.

Definition 6.17 ([58]) Suppose that $\big(\{g\}, \mathcal{F}_{\Delta G_{i-1}}(g)\big)$ is a new concept. For any $(A_{k,j}, B_{k,j}) \in \mathcal{G}^C_{\mathcal{F}^k_{i-1},\mathcal{H}^k_{i-1}}$ $(j \in \{1, 2, \ldots, |\mathcal{G}^C_{\mathcal{F}^k_{i-1},\mathcal{H}^k_{i-1}}|\})$, the CS degree can be defined as follows:

$$\theta_{k,j} = \frac{W \cdot M^T}{\left|\mathcal{F}_{\Delta G_{i-1}}(g) \bigcup B_{k,j}\right|}, \tag{6.21}$$

where $W = (w_1, w_2, \ldots, w_m)$ is a cognitive weight vector regarding a conditional attribute set M, and $M = (m_1, m_2, \ldots, m_m)$ is an attribute vector that contains (1) the value of attributes from $\mathcal{F}_{\Delta G_{i-1}}(g) \cap B_{k,j}$, which are set to 1, and (2) the elements from $M - (\mathcal{F}_{\Delta G_{i-1}}(g) \cap B_{k,j})$, which are all set to 0.

For any object, there always exists a unique class that is most similar to it by the sample separation axiom [79]. Thus, based on Definition 6.17, we can determine the maximum CS degree $\theta^*_{k,j^*} = \max\limits_{j \in \{1,2,\ldots,|\mathcal{G}^C_{\mathcal{F}^k_{i-1},\mathcal{H}^k_{i-1}}|\}} \{\theta_{k,j}\}$ and its corresponding concept (A_{k,j^*}, B_{k,j^*}) in the concept space $\mathcal{G}^C_{\mathcal{F}^k_{i-1},\mathcal{H}^k_{i-1}}$. Moreover, for the entire concept space $\mathcal{G}^C_{\mathcal{F}_{i-1},\mathcal{H}_{i-1}}$, we can further determine the global maximum CS degree $\theta^*_{k^*,j^*} = \max\limits_{k \in \{1,2,\ldots,K\}} \{\theta^*_{k,j^*}\}$ and its corresponding concept $(A_{k^*,j^*}, B_{k^*,j^*})$.

Definition 6.18 If $\theta^*_{k^*,j^*}$ is the global maximum CS degree in the entire concept space $\mathcal{G}^C_{\mathcal{F}_{i-1},\mathcal{H}_{i-1}}$, we say that a new concept $\big(\{g\}, \mathcal{F}_{\Delta G_{i-1}}(g)\big)$ can be classified into the concept space $\mathcal{G}^C_{\mathcal{F}^{k^*}_{i-1},\mathcal{H}^{k^*}_{i-1}}$ by the optimal concept $(A_{k^*,j^*}, B_{k^*,j^*})$. Moreover, for any $(Y_k, Z_k) \in \mathcal{G}^D_{\widetilde{\mathcal{F}}_{i-1},\widetilde{\mathcal{H}}_{i-1}}$ $(k \in \{1, 2, \ldots, |\mathcal{G}^D_{\widetilde{\mathcal{F}}_{i-1},\widetilde{\mathcal{H}}_{i-1}}|\})$, if $A_{k^*,j^*} \subseteq Y_k$, we say that the object g is associated with a single label z, where $Z_k = \{z\}$ in a regular formal decision context.

From Definition 6.18, we can determine that an object g is associated with a class label z if and only if the real class label $\widetilde{\mathcal{F}}_{\Delta G_{i-1}}(g)$ is consistent with the predicted class label z. However, when the ground truth label is not the same as the predicted value, we adjust the cognitive weight as follows:

$$\begin{aligned} w_i &\leftarrow w_i \pm \Delta w_i, \\ \Delta w_i &= activationFunction(\eta w_i), \end{aligned} \tag{6.22}$$

where the operator $+$ is adopted when the attributes are from $\mathcal{F}_{\Delta G_{i-1}}(g) \cap B_{k^*,j^*}$, and the other operator $-$ is used for the elements from $\mathcal{F}_{\Delta G_{i-1}}(g) \cap B_{k,j^*}$. Moreover,

$activationFunction(\eta w_i) = \frac{e^{\eta w_i} - e^{-\eta w_i}}{e^{\eta w_i} + e^{-\eta w_i}}$, where $\eta \in (0, 1)$ is known as the learning rate.

In the following, a computational procedure for a concurrent concept-cognitive process (see Algorithm 6.8) is proposed based on the above discussion. The inputs of Algorithm 6.8 are the concept spaces obtained from the output results of Algorithm 6.7. In Algorithm 6.8, running steps 9 and 12 requires $O\big(|\mathcal{G}^C_{\mathcal{F}^k_{i-1}\mathcal{H}^k_{i-1}}|\big)$ and $O\big(|\mathcal{G}^D_{\widetilde{\mathcal{F}}^k_{i-1}\widetilde{\mathcal{H}}^k_{i-1}}|\big)$, respectively. In line 15, the runtime is $O\big(|\mathcal{G}^C_{\mathcal{F}^k_{i-1}\mathcal{H}^k_{i-1}}|\big)$. Hence, it is easy to determine that the time complexity of Algorithm 6.8 is $O\big(n(\frac{1}{m}|\mathcal{G}^C_{\mathcal{F}^k_{i-1}\mathcal{H}^k_{i-1}}| + \frac{1}{p}|\mathcal{G}^D_{\widetilde{\mathcal{F}}^k_{i-1}\widetilde{\mathcal{H}}^k_{i-1}}| + |\mathcal{G}^D_{\widetilde{\mathcal{F}}_{i-1},\widetilde{\mathcal{H}}_{i-1}}|)\big)$. Then, we can obtain the collections of all conditional and decision concepts in the final period, which are denoted by $\mathcal{G}^C_{\mathcal{F}_n,\mathcal{H}_n}$ and $\mathcal{G}^D_{\widetilde{\mathcal{F}}_n,\widetilde{\mathcal{H}}_n}$, respectively.

6.1.2.3 Concept Generalization Process in C3LM

Based on the final concept spaces obtained, we can achieve classification ability. This can be understood in terms of two aspects: (1) it can complete the static classification task when the final concept spaces are directly obtained from the initial concept learning, and (2) by combining the initial concept construction process with the CCL process, it is suitable for the dynamic classification task. However, both methods predict label information by means of the CS degree.

For a test instance g, let $\Delta G_{i-1} = \{g\}$, and we obtain a new concept $\big(\mathcal{H}_{\Delta G_{i-1}}\mathcal{F}_{\Delta G_{i-1}}(g), \mathcal{F}_{\Delta G_{i-1}}(g)\big) = \big(\{g\}, \mathcal{F}_{\Delta G_{i-1}}(g)\big)$ by Definitions 6.5 and 6.16. Furthermore, according to Definitions 6.17 and 6.18, a procedure is proposed for the concept generalization task (see Algorithm 6.9). It is easy to determine that the time complexity of Algorithm 6.9 is $O\big(|\overline{G}|(|\mathcal{G}^C_{\mathcal{F}_n,\mathcal{H}_n}| + |\mathcal{G}^D_{\widetilde{\mathcal{F}}_n,\widetilde{\mathcal{H}}_n}|)\big)$.

6.1.3 Semi-Supervised Concept Learning by Concept-Cognitive Learning and Conceptual Clustering

In this subsection, we will first introduce the initial concept spaces with labeled data, and then the concept-cognitive process with unlabeled data, followed by the concept recognition and theoretical analysis of S2CL. Finally, we present the whole procedure and computational cost of our methods.

6.1.3.1 Concept Space with Structural Information

Definition 6.19 Suppose G^k is a sub-object set which is associated with a label k, and a quintuple (G^k, M, I, D, J) is known as a regular sub-object formal decision

Algorithm 6.8 Concurrent concept-cognitive process

```
1: Input: Initial concept spaces $\mathcal{G}^C_{\mathcal{F}^k_0,\mathcal{H}^k_0}$, $\mathcal{G}^D_{\widetilde{\mathcal{F}}^k_0,\widetilde{\mathcal{H}}^k_0}$ and $\mathcal{G}^C_{\mathcal{F}_0,\mathcal{H}_0}$, $\mathcal{G}^D_{\widetilde{\mathcal{F}}_0,\widetilde{\mathcal{H}}_0}$, chunk size, and adding new object set $\Delta G^k$.
2: Output: The final concept spaces $\mathcal{G}^C_{\mathcal{F}^k_n,\mathcal{H}^k_n}$ and $\mathcal{G}^D_{\widetilde{\mathcal{F}}^k_n,\widetilde{\mathcal{H}}^k_n}$.
3: Initialize $\Delta G^k = \{\Delta G^k_0, \Delta G^k_1, \ldots, \Delta G^k_{n-1}\} = \{\{g_0\}, \{g_1\}, \ldots, \{g_{n-1}\}\}$ and $W = (w_1, w_2, \ldots, w_m)$
4: for i=1 to $n$ do
5:     $m = \lceil |\mathcal{G}^C_{\mathcal{F}^k_{i-1}\mathcal{H}^k_{i-1}}|/\text{chunk-size}\rceil$ and $p = \lceil |\mathcal{G}^D_{\widetilde{\mathcal{F}}^k_{i-1}\widetilde{\mathcal{H}}^k_{i-1}}|/\text{chunk-size}\rceil$ are the numbers of threads
       for $\mathcal{G}^C_{\mathcal{F}^k_{i-1}\mathcal{H}^k_{i-1}}$ and $\mathcal{G}^D_{\widetilde{\mathcal{F}}^k_{i-1}\widetilde{\mathcal{H}}^k_{i-1}}$
6:     get $g_{i-1}$ from $\Delta G^k_{i-1}$
7:     construct new concepts $\big(\mathcal{H}^k_{\Delta G^k_{i-1}}\mathcal{F}^k_{\Delta G^k_{i-1}}(g_{i-1}), \mathcal{F}^k_{\Delta G^k_{i-1}}(g_{i-1})\big)$ and
       $\big(\widetilde{\mathcal{H}}^k_{\Delta G^k_{i-1}}\widetilde{\mathcal{F}}^k_{\Delta G^k_{i-1}}(g_{i-1}), \widetilde{\mathcal{F}}^k_{\Delta G^k_{i-1}}(g_{i-1})\big)$ by Definition 6.16
8:     for $\mathcal{G}^C_{\mathcal{F}^k_{i-1}\mathcal{H}^k_{i-1},j} = \mathcal{G}^C_{\mathcal{F}^k_{i-1}\mathcal{H}^k_{i-1},1}$ to $\mathcal{G}^C_{\mathcal{F}^k_{i-1}\mathcal{H}^k_{i-1},m}$ do in parallel
9:         get $\mathcal{G}^C_{\mathcal{F}^k_i\mathcal{H}^k_i}$ by updating $\mathcal{G}^C_{\mathcal{F}^k_{i-1}\mathcal{H}^k_{i-1}}$ based on Proposition 6.3
10:    end for
11:    for $\mathcal{G}^D_{\widetilde{\mathcal{F}}^k_{i-1}\widetilde{\mathcal{H}}^k_{i-1},j} = \mathcal{G}^D_{\widetilde{\mathcal{F}}^k_{i-1}\widetilde{\mathcal{H}}^k_{i-1},1}$ to $\mathcal{G}^D_{\widetilde{\mathcal{F}}^k_{i-1}\widetilde{\mathcal{H}}^k_{i-1},p}$ do in parallel
12:        get $\mathcal{G}^D_{\widetilde{\mathcal{F}}^k_i\widetilde{\mathcal{H}}^k_i}$ by updating $\mathcal{G}^D_{\widetilde{\mathcal{F}}^k_{i-1}\widetilde{\mathcal{H}}^k_{i-1}}$ based on Proposition 6.3
13:    end for
14:    for $\mathcal{G}^C_{\mathcal{F}^k_{i-1}\mathcal{H}^k_{i-1},j} = \mathcal{G}^C_{\mathcal{F}^k_{i-1}\mathcal{H}^k_{i-1},1}$ to $\mathcal{G}^C_{\mathcal{F}^k_{i-1}\mathcal{H}^k_{i-1},m}$ do in parallel
15:        compute the maximum CS degree $\theta^*_{k,j^*}$ and the corresponding concept $(A_{k,j^*}, B_{k,j^*})$ by Eq. (6.21)
16:    end for
17:    compute the global maximum CS degree $\theta^*_{k^*,j^*}$ and corresponding concept $(A_{k^*,j^*}, B_{k^*,j^*})$
       in $\mathcal{G}^C_{\mathcal{F}_{i-1},\mathcal{H}_{i-1}}$
18:    for each $(Y_k, Z_k) \in \mathcal{G}^D_{\widetilde{\mathcal{F}}_{i-1},\widetilde{\mathcal{H}}_{i-1}}(k \in \{1, 2, \ldots, |\mathcal{G}^D_{\widetilde{\mathcal{F}}_{i-1},\widetilde{\mathcal{H}}_{i-1}}|\})$ do
19:        if $A_{k^*,j^*} \subseteq Y_k$ then
20:            get the predicted label $z$ by Definition 6.18
21:        end if
22:        if $z \neq \widetilde{\mathcal{F}}^k_{\Delta G^k_{i-1}}(g_{i-1})$ then
23:            update the cognitive weight vector $W$ by Eq. (6.22)
24:        end if
25:    end for
26: end for
27: Return $\mathcal{G}^C_{\mathcal{F}^k_n,\mathcal{H}^k_n}$ and $\mathcal{G}^D_{\widetilde{\mathcal{F}}^k_n,\widetilde{\mathcal{H}}^k_n}$
```

Algorithm 6.9 Generalization process

```
1: Input: The final concept spaces $\mathcal{G}^C_{\mathcal{F}_n,\mathcal{H}_n}$ and $\mathcal{G}^D_{\widetilde{\mathcal{F}}_n,\widetilde{\mathcal{H}}_n}$, and test data $\overline{G}$.
2: Output: The class labels of test data.
3: for each $g \in \overline{G}$ do
4:     construct a new concept $\big(\mathcal{H}_{\Delta G_{i-1}}\mathcal{F}_{\Delta G_{i-1}}(g), \mathcal{F}_{\Delta G_{i-1}}(g)\big)$
5:     for each $(A_{k,j}, B_{k,j}) \in \mathcal{G}^C_{\mathcal{F}_n,\mathcal{H}_n}$ do
6:         get $\theta^*_{k^*,j^*}$ by Eq. (6.21)
7:     end for
8:     for each $(Y_k, Z_k) \in \mathcal{G}^D_{\widetilde{\mathcal{F}}_n,\widetilde{\mathcal{H}}_n}$ $(k \in \{1, 2, \ldots, |\mathcal{G}^D_{\widetilde{\mathcal{F}}_n,\widetilde{\mathcal{H}}_n}|\})$ do
9:         get the class label $z$ by Definition 6.18
10:    end for
11: end for
12: Return class labels
```

context. Then (G^k, M, I) and (G^k, D, J) are respectively called the conditional sub-object formal context and decision sub-object formal context.

Moreover, the set-valued mappings $\mathcal{F}^k : 2^{G^k} \to 2^M, \mathcal{H}^k : 2^M \to 2^{G^k}$, and $\widetilde{\mathcal{F}}^k : 2^{G^k} \to 2^D, \widetilde{\mathcal{H}}^k : 2^D \to 2^{G^k}$ are respectively called the conditional sub-object cognitive operators and decision sub-object cognitive operators with a sub-object set G^k.

Definition 6.20 Let (G^k, M, I) be a conditional sub-object formal context, and $\mathcal{F}^k$, $\mathcal{H}^k$ be the conditional sub-object cognitive operators. For any $x', x'' \in G^k$, if $\mathcal{H}^k\mathcal{F}^k(x') = \{x'\}$ and $\mathcal{H}^k\mathcal{F}^k(x'') \supset \{x''\}$, then the pairs $(\mathcal{H}^k\mathcal{F}^k(x'), \mathcal{F}^k(x'))$ and $(\mathcal{H}^k\mathcal{F}^k(x''), \mathcal{F}^k(x''))$ are referred to as object-oriented conditional granular concepts (or simply object-oriented conditional concepts). For convenience, we denote

$$\begin{aligned} \mathcal{OG}_{\mathcal{F}^k\mathcal{H}^k} =& \{(\mathcal{H}^k\mathcal{F}^k(x'), \mathcal{F}^k(x'))|x' \in G^k\} \cup \\ & \{(\mathcal{H}^k\mathcal{F}^k(x''), \mathcal{F}^k(x''))|x'' \in G^k\}. \end{aligned}$$

Simultaneously, for any $a', a'' \in M$, if $\mathcal{F}^k\mathcal{H}^k(a') = \{a'\}$ and $\mathcal{F}^k\mathcal{H}^k(a'') \supset \{a''\}$, then the pairs $(\mathcal{H}^k(a'), \mathcal{F}^k\mathcal{H}^k(a'))$ and $(\mathcal{H}^k(a''), \mathcal{F}^k\mathcal{H}^k(a''))$ are called attribute-oriented conditional granular concepts (or simply attribute-oriented conditional concepts). For brevity, we further denote

$$\begin{aligned} \mathcal{AG}_{\mathcal{F}^k\mathcal{H}^k} =& \{(\mathcal{H}^k(a'), \mathcal{F}^k\mathcal{H}^k(a'))|a' \in M\} \cup \\ & \{(\mathcal{H}^k(a''), \mathcal{F}^k\mathcal{H}^k(a''))|a'' \in M\}. \end{aligned}$$

Definition 6.21 Let (G^k, D, J) be a decision sub-object formal context and $\widetilde{\mathcal{F}}^k$, $\widetilde{\mathcal{H}}^k$ be the decision sub-object cognitive operators. For any $x', x'' \in G^k$, if $\widetilde{\mathcal{H}}^k\widetilde{\mathcal{F}}^k(x') = \{x'\}$ and $\widetilde{\mathcal{H}}^k\widetilde{\mathcal{F}}^k(x'') \supset \{x''\}$, then the pairs $(\widetilde{\mathcal{H}}^k\widetilde{\mathcal{F}}^k(x'), \widetilde{\mathcal{F}}^k(x'))$ and $(\widetilde{\mathcal{H}}^k\widetilde{\mathcal{F}}^k(x''), \widetilde{\mathcal{F}}^k(x''))$ are known as object-oriented decision granular concepts (or simply object-oriented decision concepts). For convenience, we denote

$$\begin{aligned}\mathcal{OG}_{\widetilde{\mathcal{F}}^k\widetilde{\mathcal{H}}^k} =&\{(\widetilde{\mathcal{H}}^k\widetilde{\mathcal{F}}^k(x'),\widetilde{\mathcal{F}}^k(x'))|x' \in G^k\}\cup\\ &\{(\widetilde{\mathcal{H}}^k\widetilde{\mathcal{F}}^k(x''),\widetilde{\mathcal{F}}^k(x''))|x'' \in G^k\}.\end{aligned}$$

Meanwhile, for any $k', k'' \in D$, if $\widetilde{\mathcal{F}}^k\widetilde{\mathcal{H}}^k(k') = \{k'\}$ and $\widetilde{\mathcal{F}}^k\widetilde{\mathcal{H}}^k(k'') \supset \{k''\}$, then the pairs $(\widetilde{\mathcal{H}}^k(k'),\widetilde{\mathcal{F}}^k\widetilde{\mathcal{H}}^k(k'))$ and $(\widetilde{\mathcal{H}}^k(k''),\widetilde{\mathcal{F}}^k\widetilde{\mathcal{H}}^k(k''))$ are called attribute-oriented decision granular concepts (or simply attribute-oriented decision concepts). For brevity, we further denote

$$\begin{aligned}\mathcal{AG}_{\widetilde{\mathcal{F}}^k\widetilde{\mathcal{H}}^k} =&\{(\widetilde{\mathcal{H}}^k(k'),\widetilde{\mathcal{F}}^k\widetilde{\mathcal{H}}^k(k'))|k' \in D\}\cup\\ &\{(\widetilde{\mathcal{H}}^k(k''),\widetilde{\mathcal{F}}^k\widetilde{\mathcal{H}}^k(k''))|k'' \in D\}.\end{aligned}$$

To facilitate the subsequent discussion, in a regular sub-object formal decision context, the conditional concept space and decision concept space are respectively denoted by

$$\begin{aligned}\mathcal{G}_{\mathcal{F}^k\mathcal{H}^k} &= \mathcal{OG}_{\mathcal{F}^k\mathcal{H}^k} \cup \mathcal{AG}_{\mathcal{F}^k\mathcal{H}^k}\\ &= \{(\mathcal{H}^k\mathcal{F}^k(x),\mathcal{F}^k(x))|x \in G^k\}\cup\\ &\quad\{(\mathcal{H}^k(a),\mathcal{F}^k\mathcal{H}^k(a))|a \in M\}, \text{and}\\ \mathcal{G}_{\widetilde{\mathcal{F}}^k\widetilde{\mathcal{H}}^k} &= \mathcal{OG}_{\widetilde{\mathcal{F}}^k\widetilde{\mathcal{H}}^k} \cup \mathcal{AG}_{\widetilde{\mathcal{F}}^k\widetilde{\mathcal{H}}^k}\\ &= \{(\widetilde{\mathcal{H}}^k\widetilde{\mathcal{F}}^k(x),\widetilde{\mathcal{F}}^k(x))|x \in G^k\}\cup\\ &\quad\{(\widetilde{\mathcal{H}}^k(k'),\widetilde{\mathcal{F}}^k\widetilde{\mathcal{H}}^k(k'))|k' \in D\}.\end{aligned}$$

It means that the concept spaces of sub-object set G^k can be constructed by means of the object-oriented concepts and attribute-oriented concepts.

Theorem 6.2 *Let $\mathcal{G}_{\mathcal{F}^k\mathcal{H}^k}$ and $\mathcal{G}_{\widetilde{\mathcal{F}}^k\widetilde{\mathcal{H}}^k}$ be the conditional concept space and decision concept space, respectively. Then the following statements hold:*

(1) For any conditional concepts $(\mathcal{H}^k\mathcal{F}^k(x),\mathcal{F}^k(x))$ and $(\mathcal{H}^k(a),\mathcal{F}^k\mathcal{H}^k(a))$, if there exists a conditional concept $(\mathcal{H}^k\mathcal{F}^k(x_i),\mathcal{F}^k(x_i)) \in \mathcal{G}_{\mathcal{F}^k\mathcal{H}^k}$ such that $\mathcal{F}^k(x) \subseteq \mathcal{F}^k(x_i)$ $(i \in \{1,2,\ldots,|G^k|\})$ and a conditional concept $(\mathcal{H}^k(a_j),\mathcal{F}^k\mathcal{H}^k(a_j))$ $\in \mathcal{G}_{\mathcal{F}^k\mathcal{H}^k}$ such that $\mathcal{H}^k(a) \subseteq \mathcal{H}^k(a_j)$ $(j \in \{1,2,\ldots,|M|\})$, then we have

$$\begin{aligned}(\mathcal{H}^k\mathcal{F}^k(x),\mathcal{F}^k(x)) &= (\{x \cup \bigcup_{i\in\{1,2,\ldots,|G^k|\}} x_i\},\mathcal{F}^k(x)),\\ (\mathcal{H}^k(a),\mathcal{F}^k\mathcal{H}^k(a)) &= (\mathcal{H}^k(a),\{a \cup \bigcup_{j\in\{1,2,\ldots,|M|\}} a_j\});\end{aligned} \tag{6.23}$$

otherwise,

$$\begin{aligned}(\mathcal{H}^k\mathcal{F}^k(x), \mathcal{F}^k(x)) &= (\{x\}, \mathcal{F}^k(x)),\\ (\mathcal{H}^k(a), \mathcal{F}^k\mathcal{H}^k(a)) &= (\mathcal{H}^k(a), \{a\}).\end{aligned} \tag{6.24}$$

(2) For any decision concepts $(\widetilde{\mathcal{H}}^k\widetilde{\mathcal{F}}^k(x), \widetilde{\mathcal{F}}^k(x))$ *and* $(\widetilde{\mathcal{H}}^k(k'), \widetilde{\mathcal{F}}^k\widetilde{\mathcal{H}}^k(k'))$*, if there exists a decision concept* $(\widetilde{\mathcal{H}}^k\widetilde{\mathcal{F}}^k(x_i), \widetilde{\mathcal{F}}^k(x_i)) \in \mathcal{G}_{\widetilde{\mathcal{F}}^k\widetilde{\mathcal{H}}^k}$ *such that* $\widetilde{\mathcal{F}}^k(x) \subseteq \widetilde{\mathcal{F}}^k(x_i)$ $(i \in \{1, 2, \ldots, |G^k|\})$ *and a decision concept* $(\widetilde{\mathcal{H}}^k(k_j), \widetilde{\mathcal{F}}^k\widetilde{\mathcal{H}}^k(k_j)) \in \mathcal{G}_{\widetilde{\mathcal{F}}^k\widetilde{\mathcal{H}}^k}$ *such that* $\widetilde{\mathcal{H}}^k(k') \subseteq \widetilde{\mathcal{H}}^k(k_j)$ $(j \in \{1, 2, \ldots, |D|\})$*, then following statements hold:*

$$\begin{aligned}(\widetilde{\mathcal{H}}^k\widetilde{\mathcal{F}}^k(x), \widetilde{\mathcal{F}}^k(x)) &= (\{x \cup \bigcup_{i\in\{1,2,\ldots,|G^k|\}} x_i\}, \widetilde{\mathcal{F}}^k(x)),\\ (\widetilde{\mathcal{H}}^k(k'), \widetilde{\mathcal{F}}^k\widetilde{\mathcal{H}}^k(k')) &= (\widetilde{\mathcal{H}}^k(k'), \{k' \cup \\ &\qquad \bigcup_{j\in\{1,2,\ldots,|D|\}} k_j\});\end{aligned} \tag{6.25}$$

otherwise,

$$\begin{aligned}(\widetilde{\mathcal{H}}^k\widetilde{\mathcal{F}}^k(x), \widetilde{\mathcal{F}}^k(x)) &= (\{x\}, \widetilde{\mathcal{F}}^k(x)),\\ (\widetilde{\mathcal{H}}^k(k'), \widetilde{\mathcal{F}}^k\widetilde{\mathcal{H}}^k(k')) &= (\widetilde{\mathcal{H}}^k(k'), \{k'\}).\end{aligned} \tag{6.26}$$

Proof The proof of Theorem 6.2 can be found in the original paper [43]. □

Property 6.4 Let $\mathcal{G}^{\diamond}_{\mathcal{F}^k\mathcal{H}^k}$ and $\mathcal{G}^{\diamond}_{\widetilde{\mathcal{F}}^k\widetilde{\mathcal{H}}^k}$ be two concept spaces, $\mathcal{AG}_{\mathcal{F}^k\mathcal{H}^k}$ and $\mathcal{AG}_{\widetilde{\mathcal{F}}^k\widetilde{\mathcal{H}}^k}$ be the attribute-oriented conditional concept space and attribute-oriented decision concept space, respectively; meanwhile, initialize $\mathcal{G}^{\diamond}_{\mathcal{F}^k\mathcal{H}^k} = \mathcal{AG}_{\mathcal{F}^k\mathcal{H}^k}$ and $\mathcal{G}^{\diamond}_{\widetilde{\mathcal{F}}^k\widetilde{\mathcal{H}}^k} = \mathcal{AG}_{\widetilde{\mathcal{F}}^k\widetilde{\mathcal{H}}^k}$. Then we have

(1) For each $x \in G^k$, if there exists $(\mathcal{H}^k(a), \mathcal{F}^k\mathcal{H}^k(a)) \in \mathcal{AG}_{\mathcal{F}^k\mathcal{H}^k}$ such that $\mathcal{F}^k(x) = \mathcal{F}^k\mathcal{H}^k(a)$, then $(\mathcal{H}^k\mathcal{F}^k(x), \mathcal{F}^k(x)) = (\mathcal{H}^k(a), \mathcal{F}^k\mathcal{H}^k(a))$;
otherwise,
$\mathcal{G}^{\diamond}_{\mathcal{F}^k\mathcal{H}^k} = \mathcal{G}^{\diamond}_{\mathcal{F}^k\mathcal{H}^k} \cup (\mathcal{H}^k\mathcal{F}^k(x), \mathcal{F}^k(x))$.

(2) For each $x \in G^k$, if there exists $(\widetilde{\mathcal{H}}^k(k'), \widetilde{\mathcal{F}}^k\widetilde{\mathcal{H}}^k(k')) \in \mathcal{AG}_{\widetilde{\mathcal{F}}^k\widetilde{\mathcal{H}}^k}$ such that $\widetilde{\mathcal{F}}^k(x) = \widetilde{\mathcal{F}}^k\widetilde{\mathcal{H}}^k(k')$, then $(\widetilde{\mathcal{H}}^k\widetilde{\mathcal{F}}^k(x), \widetilde{\mathcal{F}}^k(x)) = (\widetilde{\mathcal{H}}^k(k'), \widetilde{\mathcal{F}}^k\widetilde{\mathcal{H}}^k(k'))$;
otherwise,
$\mathcal{G}^{\diamond}_{\widetilde{\mathcal{F}}^k\widetilde{\mathcal{H}}^k} = \mathcal{G}^{\diamond}_{\widetilde{\mathcal{F}}^k\widetilde{\mathcal{H}}^k} \cup (\widetilde{\mathcal{H}}^k\widetilde{\mathcal{F}}^k(x), \widetilde{\mathcal{F}}^k(x))$.

Proof The proof of Property 6.4 can be found in the original paper [43]. □

Property 6.4 means that we do not need to construct concepts $(\mathcal{H}^k\mathcal{F}^k(x), \mathcal{F}^k(x))$ and $(\widetilde{\mathcal{H}}^k\widetilde{\mathcal{F}}^k(x), \widetilde{\mathcal{F}}^k(x))$ like [58] when $\mathcal{F}^k(x) = \mathcal{F}^k\mathcal{H}^k(a)$ and $\widetilde{\mathcal{F}}^k(x) =$

$\widetilde{\mathcal{F}}^k\widetilde{\mathcal{H}}^k(k')$. Then, using this approach, we can finally obtain $\mathcal{G}_{\mathcal{F}^k\mathcal{H}^k} = \mathcal{G}^{\diamond}_{\mathcal{F}^k\mathcal{H}^k}$ and $\mathcal{G}_{\widetilde{\mathcal{F}}^k\widetilde{\mathcal{H}}^k} = \mathcal{G}^{\diamond}_{\widetilde{\mathcal{F}}^k\widetilde{\mathcal{H}}^k}$.

For convenience, we denote the labeled dataset S_L by G_0, and the initial concept spaces by $\mathcal{G}_{\mathcal{F}_0\mathcal{H}_0}$ and $\mathcal{G}_{\widetilde{\mathcal{F}}_0\widetilde{\mathcal{H}}_0}$. Note that, in the initial concept space period, if the object set G^k is replaced with G_0^k, then the corresponding cognitive operators $\mathcal{F}^k$, $\mathcal{H}^k$ and $\widetilde{\mathcal{F}}^k$, $\widetilde{\mathcal{H}}^k$ can be expressed as $\mathcal{F}_0^k$, $\mathcal{H}_0^k$ and $\widetilde{\mathcal{F}}_0^k$, $\widetilde{\mathcal{H}}_0^k$, respectively.

6.1.3.2 Cognitive Process with Unlabeled Data in Concept Learning

In the concept-cognitive process, suppose the obtained concept spaces will be updated by a newly added object instead of inputting multi-objects simultaneously. Then, for the unlabeled set S_U, we can denote S_U as $\Delta G = \{\Delta G_0, \Delta G_1, \ldots, \Delta G_{n-1}\}$ in which each learning step only consists of one object x (i.e., $\Delta G_i = \{x_i\}$). For brevity, in what follows, we write $\{x_i\}$ as x_i and then we have $\Delta G = \{x_0, x_1, \ldots, x_{n-1}\}$.

Different from [58], we assume that an object x is connected with a virtual label k^* due to no label information. Then we have the conditional sub-object cognitive operators and decision sub-object cognitive operators with the newly input data $\Delta G_{i-1}^{k^*} = G_i^{k^*} - G_{i-1}^{k^*}$ as follows:

$$\begin{aligned}
&\text{(i) } \mathcal{F}_{i-1}^{k^*} : 2^{G_{i-1}^{k^*}} \to 2^M, && \mathcal{H}_{i-1}^{k^*} : 2^M \to 2^{G_{i-1}^{k^*}},\\
&\text{(ii) } \mathcal{F}^{k^*}_{\Delta G_{i-1}^{k^*}} : 2^{\Delta G_{i-1}^{k^*}} \to 2^M, && \mathcal{H}^{k^*}_{\Delta G_{i-1}^{k^*}} : 2^M \to 2^{\Delta G_{i-1}^{k^*}},\\
&\text{(iii) } \mathcal{F}_i^{k^*} : 2^{G_i^{k^*}} \to 2^M, && \mathcal{H}_i^{k^*} : 2^M \to 2^{G_i^{k^*}},
\end{aligned} \tag{6.27}$$

and

$$\begin{aligned}
&\text{(i) } \widetilde{\mathcal{F}}_{i-1}^{k^*} : 2^{G_{i-1}^{k^*}} \to 2^D, && \widetilde{\mathcal{H}}_{i-1}^{k^*} : 2^D \to 2^{G_{i-1}^{k^*}},\\
&\text{(ii) } \widetilde{\mathcal{F}}^{k^*}_{\Delta G_{i-1}^{k^*}} : 2^{\Delta G_{i-1}^{k^*}} \to 2^D, && \widetilde{\mathcal{H}}^{k^*}_{\Delta G_{i-1}^{k^*}} : 2^D \to 2^{\Delta G_{i-1}^{k^*}},\\
&\text{(iii) } \widetilde{\mathcal{F}}_i^{k^*} : 2^{G_i^{k^*}} \to 2^D, && \widetilde{\mathcal{H}}_i^{k^*} : 2^D \to 2^{G_i^{k^*}}.
\end{aligned} \tag{6.28}$$

Theorem 6.3 *Let $\mathcal{AG}_{\mathcal{F}_{i-1}^{k^*}\mathcal{H}_{i-1}^{k^*}}$, $\mathcal{AG}_{\widetilde{\mathcal{F}}_{i-1}^{k^*}\widetilde{\mathcal{H}}_{i-1}^{k^*}}$ and $\mathcal{OG}_{\mathcal{F}_{i-1}^{k^*}\mathcal{H}_{i-1}^{k^*}}$, $\mathcal{OG}_{\widetilde{\mathcal{F}}_{i-1}^{k^*}\widetilde{\mathcal{H}}_{i-1}^{k^*}}$ be the attribute-oriented concept spaces and object-oriented concept spaces, respectively. Then we have*

*(1) For any $a' \in M$ and $(\mathcal{H}_{i-1}^{k^*}(a''), \mathcal{F}_{i-1}^{k^*}\mathcal{H}_{i-1}^{k^*}(a'')) \in \mathcal{AG}_{\mathcal{F}_{i-1}^{k^*}\mathcal{H}_{i-1}^{k^*}}$, if $\mathcal{F}_{i-1}^{k^*}\mathcal{H}_{i-1}^{k^*}(a'') \cap$
$\mathcal{F}^{k^*}_{\Delta G_{i-1}^{k^*}}\mathcal{H}^{k^*}_{\Delta G_{i-1}^{k^*}}(a') \neq \emptyset$, then*

$(\mathcal{H}_i^{k^*}(a''), \mathcal{F}_i^{k^*}\mathcal{H}_i^{k^*}(a'')) = (\mathcal{H}_{i-1}^{k^*}(a'') \cup \mathcal{H}_{\Delta G_{i-1}^{k^*}}^{k^*}(a'), \mathcal{F}_{i-1}^{k^*}\mathcal{H}_{i-1}^{k^*}(a'') \cap \mathcal{F}_{\Delta G_{i-1}^{k^*}}^{k^*}\mathcal{H}_{\Delta G_{i-1}^{k^*}}^{k^*}(a'))$;

otherwise,
$(\mathcal{H}_i^{k^*}(a'), \mathcal{F}_i^{k^*}\mathcal{H}_i^{k^*}(a')) = (\mathcal{H}_{\Delta G_{i-1}^{k^*}}^{k^*}(a'), \mathcal{F}_{\Delta G_{i-1}^{k^*}}^{k^*}\mathcal{H}_{\Delta G_{i-1}^{k^*}}^{k^*}(a'))$.

(2) For any $x' \in \Delta G^{k^*}$ *and* $(\mathcal{H}_{i-1}^{k^*}\mathcal{F}_{i-1}^{k^*}(x''), \mathcal{F}_{i-1}^{k^*}(x'')) \in \mathcal{OG}_{\mathcal{F}_{i-1}^{k^*}\mathcal{H}_{i-1}^{k^*}}$, *if* $\mathcal{F}_{i-1}^{k^*}(x'') \subseteq \mathcal{F}_{\Delta G_{i-1}^{k^*}}^{k^*}(x')$, *then* $(\mathcal{H}_i^{k^*}\mathcal{F}_i^{k^*}(x''), \mathcal{F}_i^{k^*}(x'')) = (\mathcal{H}_{i-1}^{k^*}\mathcal{F}_{i-1}^{k^*}(x'') \cup \mathcal{H}_{\Delta G_{i-1}^{k^*}}^{k^*}\mathcal{F}_{\Delta G_{i-1}^{k^*}}^{k^*}(x'), \mathcal{F}_{i-1}^{k^*}(x''))$;

if $\mathcal{F}_{\Delta G_{i-1}^{k^*}}^{k^*}(x') \subseteq \mathcal{F}_{i-1}^{k^*}(x'')$, *then*
$(\mathcal{H}_i^{k^*}\mathcal{F}_i^{k^*}(x''), \mathcal{F}_i^{k^*}(x'')) = (\mathcal{H}_{i-1}^{k^*}\mathcal{F}_{i-1}^{k^*}(x'') \cup \mathcal{H}_{\Delta G_{i-1}^{k^*}}^{k^*}\mathcal{F}_{\Delta G_{i-1}^{k^*}}^{k^*}(x'), \mathcal{F}_{\Delta G_{i-1}^{k^*}}^{k^*}(x'))$;

otherwise,
$(\mathcal{H}_i^{k^*}\mathcal{F}_i^{k^*}(x'), \mathcal{F}_i^{k^*}(x')) = (\mathcal{H}_{\Delta G_{i-1}^{k^*}}^{k^*}\mathcal{F}_{\Delta G_{i-1}^{k^*}}^{k^*}(x'), \mathcal{F}_{\Delta G_{i-1}^{k^*}}^{k^*}(x'))$.

(3) For any $k' \in D$ *and* $(\widetilde{\mathcal{H}}_{i-1}^{k^*}(k''), \widetilde{\mathcal{F}}_{i-1}^{k^*}\widetilde{\mathcal{H}}_{i-1}^{k^*}(k'')) \in \mathcal{AG}_{\widetilde{\mathcal{F}}_{i-1}^{k^*}\widetilde{\mathcal{H}}_{i-1}^{k^*}}$, *if* $\widetilde{\mathcal{F}}_{i-1}^{k^*}\widetilde{\mathcal{H}}_{i-1}^{k^*}(k'') \cap \widetilde{\mathcal{F}}_{\Delta G_{i-1}^{k^*}}^{k^*}\widetilde{\mathcal{H}}_{\Delta G_{i-1}^{k^*}}^{k^*}(k') \neq \emptyset$, *then*
$(\widetilde{\mathcal{H}}_i^{k^*}(k''), \widetilde{\mathcal{F}}_i^{k^*}\widetilde{\mathcal{H}}_i^{k^*}(k'')) = (\widetilde{\mathcal{H}}_{i-1}^{k^*}(k'') \cup \widetilde{\mathcal{H}}_{\Delta G_{i-1}^{k^*}}^{k^*}(k'), \widetilde{\mathcal{F}}_{i-1}^{k^*}\widetilde{\mathcal{H}}_{i-1}^{k^*}(k'') \cap \widetilde{\mathcal{F}}_{\Delta G_{i-1}^{k^*}}^{k^*}\widetilde{\mathcal{H}}_{\Delta G_{i-1}^{k^*}}^{k^*}(k'))$;

otherwise,
$(\widetilde{\mathcal{H}}_i^{k^*}(k'), \widetilde{\mathcal{F}}_i^{k^*}\widetilde{\mathcal{H}}_i^{k^*}(k')) = (\widetilde{\mathcal{H}}_{\Delta G_{i-1}^{k^*}}^{k^*}(k'), \widetilde{\mathcal{F}}_{\Delta G_{i-1}^{k^*}}^{k^*}\widetilde{\mathcal{H}}_{\Delta G_{i-1}^{k^*}}^{k^*}(k'))$.

(4) For any $x' \in \Delta G^{k^*}$ *and* $(\widetilde{\mathcal{H}}_{i-1}^{k^*}\widetilde{\mathcal{F}}_{i-1}^{k^*}(x''), \widetilde{\mathcal{F}}_{i-1}^{k^*}(x'')) \in \mathcal{OG}_{\widetilde{\mathcal{F}}_{i-1}^{k^*}\widetilde{\mathcal{H}}_{i-1}^{k^*}}$, *if* $\widetilde{\mathcal{F}}_{i-1}^{k^*}(x'') \subseteq \widetilde{\mathcal{F}}_{\Delta G_{i-1}^{k^*}}^{k^*}(x')$, *then* $(\widetilde{\mathcal{H}}_i^{k^*}\widetilde{\mathcal{F}}_i^{k^*}(x''), \widetilde{\mathcal{F}}_i^{k^*}(x'')) = (\widetilde{\mathcal{H}}_{i-1}^{k^*}\widetilde{\mathcal{F}}_{i-1}^{k^*}(x'') \cup \widetilde{\mathcal{H}}_{\Delta G_{i-1}^{k^*}}^{k^*}\widetilde{\mathcal{F}}_{\Delta G_{i-1}^{k^*}}^{k^*}(x'), \widetilde{\mathcal{F}}_{i-1}^{k^*}(x''))$;

if $\widetilde{\mathcal{F}}_{\Delta G_{i-1}^{k^*}}^{k^*}(x') \subseteq \widetilde{\mathcal{F}}_{i-1}^{k^*}(x'')$, *then*
$(\widetilde{\mathcal{H}}_i^{k^*}\widetilde{\mathcal{F}}_i^{k^*}(x''), \widetilde{\mathcal{F}}_i^{k^*}(x'')) = (\widetilde{\mathcal{H}}_{i-1}^{k^*}\widetilde{\mathcal{F}}_{i-1}^{k^*}(x'') \cup \widetilde{\mathcal{H}}_{\Delta G_{i-1}^{k^*}}^{k^*}\widetilde{\mathcal{F}}_{\Delta G_{i-1}^{k^*}}^{k^*}(x'), \widetilde{\mathcal{F}}_{\Delta G_{i-1}^{k^*}}^{k^*}(x'))$;

otherwise,
$(\widetilde{\mathcal{H}}_i^{k^*}\widetilde{\mathcal{F}}_i^{k^*}(x'), \widetilde{\mathcal{F}}_i^{k^*}(x')) = (\widetilde{\mathcal{H}}_{\Delta G_{i-1}^{k^*}}^{k^*}\widetilde{\mathcal{F}}_{\Delta G_{i-1}^{k^*}}^{k^*}(x'), \widetilde{\mathcal{F}}_{\Delta G_{i-1}^{k^*}}^{k^*}(x'))$.

Proof The proof of Theorem 6.3 can be found in the original paper [42]. □

Although Theorem 6.3 shows how to update the concept spaces when adding an instance, concept recognition is exceedingly difficult since we cannot directly recognize the real class label of each instance $\boldsymbol{x}$. Namely, unlike the initial concept spaces generation, there is still a mystery that which sub-concept space will be updated when inputting a new object without label information.

6.1.3.3 Concept Recognition

For any newly input object x, the concept $(\mathcal{H}^{k^*}_{\Delta G_i^{k^*}}\mathcal{F}^{k^*}_{\Delta G_i^{k^*}}(x), \mathcal{F}^{k^*}_{\Delta G_i^{k^*}}(x))$ can be rewritten as $(\{x\}, \mathcal{F}^{k^*}_{\Delta G_i^{k^*}}(x))$ due to $|\Delta G_i^{k^*}| = 1$. Meanwhile, to meet the demand of lots of unlabeled data, a new similarity metric for concept learning is proposed in this subsection. As a matter of fact, a good assessing similarity for concepts is a key success of S2CL.

Definition 6.22 Let $\mathcal{G}_{\mathcal{F}_{i-1},\mathcal{H}_{i-1}}$ be the concept space and $\mathcal{G}_{\mathcal{F}^{k^*}_{i-1},\mathcal{H}^{k^*}_{i-1}}$ be a sub-concept space with a virtual label k^* in the $(i-1)$-th state. For any concept $(X_j, B_j) \in \mathcal{G}_{\mathcal{F}^{k^*}_{i-1},\mathcal{H}^{k^*}_{i-1}}$, where $j \in \{1, 2, \dots, |\mathcal{G}_{\mathcal{F}^{k^*}_{i-1},\mathcal{H}^{k^*}_{i-1}}|\}$, the global information w_{i-1,k^*} and the local information $z^{k^*}_{i-1,j}$ in the $(i-1)$-th state are, respectively, defined as

$$w_{i-1,k^*} = \frac{|\mathcal{G}_{\mathcal{F}^{k^*}_{i-1},\mathcal{H}^{k^*}_{i-1}}|}{|\mathcal{G}_{\mathcal{F}_{i-1},\mathcal{H}_{i-1}}|}, \tag{6.29}$$

$$z^{k^*}_{i-1,j} = \frac{|X_j|}{|\mathcal{G}_{\mathcal{F}^{k^*}_{i-1},\mathcal{H}^{k^*}_{i-1}}|}. \tag{6.30}$$

More generally, considering the entire concept space $\mathcal{G}_{\mathcal{F}_{i-1},\mathcal{H}_{i-1}}$ in the $(i-1)$-th state, we denote

$$\boldsymbol{w}_{i-1} = (w_{i-1,1}, w_{i-1,2}, \dots, w_{i-1,l}), \tag{6.31}$$

$$\boldsymbol{z}_{i-1} = \begin{bmatrix} z^1_{i-1} \\ \vdots \\ z^l_{i-1} \end{bmatrix} = \begin{bmatrix} z^1_{i-1,1} & \cdots & z^1_{i-1,m_1} \\ \vdots & \ddots & \vdots \\ z^l_{i-1,1} & \cdots & z^l_{i-1,m_l} \end{bmatrix}, \tag{6.32}$$

where $m_{k^*} = \left|\mathcal{G}_{\mathcal{F}^{k^*}_{i-1},\mathcal{H}^{k^*}_{i-1}}\right|$ and $k^* \in \{1, 2, \dots, K\}$.

Definition 6.23 Let $C = (\{x\}, \mathcal{F}_{\Delta G_i^{k^*}}(x))$ be a newly input concept. For any concept $(X_j, B_j) \in \mathcal{G}_{\mathcal{F}^{k^*}_{i-1},\mathcal{H}^{k^*}_{i-1}}$, where $j \in \{1, 2, \dots, |\mathcal{G}_{\mathcal{F}^{k^*}_{i-1},\mathcal{H}^{k^*}_{i-1}}|\}$, the concept similarity (CS) can be defined as:

$$\theta^I_j = I\frac{|A^* \cap B_j|}{|A^* \cap B_j| + 2(\alpha|A^* - B_j| + (1-\alpha)|B_j - A^*|)}, \tag{6.33}$$

where $I = 1/(1 + w_{i-1,k^*} \times e^{-z^{k^*}_{i-1,j}})$, $A^* = \mathcal{F}_{\Delta G_i^{k^*}}(x)$ and $\alpha \in [0, 1]$.

For Eq. (6.33), I is set to be 1 when without considering the global and local information. In this case, Eq. (6.33) can further be formulated as

$$\theta_j = \frac{|A^* \cap B_j|}{|A^* \cap B_j| + 2(\alpha|A^* - B_j| + (1-\alpha)|B_j - A^*|)}. \tag{6.34}$$

In Eq. (6.34), $A^* - B_j$ represents the characteristics appearing in A^* but not in B_j, and it has the same meaning for $B_j - A^*$. Moreover, the parameters α and $(1-\alpha)$ can be, respectively, considered as the weight information added to $|A^* - B_j|$ and $|B_j - A^*|$, which express the importance of different features of $A^* - B_j$ and $B_j - A^*$ relative to the overall similarity degree. In fact, when $\alpha = 0.5$, Eq. (6.34) is degenerated into Jaccard similarity [27, 65].

According to sample separation axiom [79], for any instance, there always exists a unique class that is most similar to it. Hence, given an instance $\boldsymbol{x}$, the class vector can be generated as follows: each sub-concept space will first produce a set of CS degrees by computing the CS degree between the given concept and any concept from a sub-concept space. Then, the maximum CS degree $(\widehat{\theta}_j^I)$ of each sub-concept space will be obtained, namely, $\widehat{\theta}_j^I = \max\limits_{j \in J}\{\theta_j^I\}$, where $J = \{1, 2, \ldots, |\mathcal{G}_{\mathcal{F}_{i-1}^{k*}, \mathcal{H}_{i-1}^{k*}}|\}$. Finally, the estimated class distribution will form a maximum class vector $(\widehat{\theta}_1^I, \widehat{\theta}_2^I, \ldots, \widehat{\theta}_l^I)^{\mathrm{T}}$. In the same manner, we can obtain an average class vector $(\overline{\theta}_1^I, \overline{\theta}_2^I, \ldots, \overline{\theta}_l^I)^{\mathrm{T}}$.

Note that a SSL method, which is designed by combining the concept-cognitive process with the structural concept similarity θ_j, is referred to as a semi-supervised concept learning method, and it is abbreviated as S2CL for convenience. In the meanwhile, an extended version of S2CL is further proposed by taking full advantage of the global and local conceptual information (i.e., the structural concept similarity θ_j^I) within a concept space. For conciseness, we also write it as S2CL$^\alpha$ when no confusion exists.

6.1.3.4 Theoretical Analysis

Essentially, α mainly reflects the influences of different characteristics in sets $A^* - B_j$ and $B_j - A^*$ for the overall concept similarity measure. Hence, it is very important to discuss how to select an appropriate α on each dataset.

Let $\mathcal{Y} = \{1, 2, \ldots, l\}$ be the label space. The concept spaces with different α_r ($\alpha_r \in [0, 1]$) in the $(i-1)$-th period can be formulated as

$$\begin{bmatrix} \mathcal{G}^{\alpha_1}_{\mathcal{F}_{i-1},\mathcal{H}_{i-1}} \\ \vdots \\ \mathcal{G}^{\alpha_n}_{\mathcal{F}_{i-1},\mathcal{H}_{i-1}} \end{bmatrix} = \begin{bmatrix} \mathcal{G}^{\alpha_1,1}_{\mathcal{F}_{i-1},\mathcal{H}_{i-1}} & \cdots & \mathcal{G}^{\alpha_1,l}_{\mathcal{F}_{i-1},\mathcal{H}_{i-1}} \\ \vdots & \ddots & \vdots \\ \mathcal{G}^{\alpha_n,1}_{\mathcal{F}_{i-1},\mathcal{H}_{i-1}} & \cdots & \mathcal{G}^{\alpha_n,l}_{\mathcal{F}_{i-1},\mathcal{H}_{i-1}} \end{bmatrix}, \tag{6.35}$$

where $\sum\limits_{r=1}^{n} \alpha_r = 1$.

For an object x_i, we can obtain its corresponding concept $C_i = (\{x_i\}, B_i)$. Then, based on Definition 6.23, we denote

$$Sim(C_i, \mathcal{G}^{\alpha_r,k}_{\mathcal{F}_{i-1},\mathcal{H}_{i-1}}) = \{Sim(C_i, C^{\alpha_r}_j)\}^{m_k}_{j=1} = \{\theta^I_j\}^{m_k}_{j=1}, \tag{6.36}$$

where $C^{\alpha_r}_j \in \mathcal{G}^{\alpha_r,k}_{\mathcal{F}_{i-1},\mathcal{H}_{i-1}} (k \in \mathcal{Y})$ and $m_k = |\mathcal{G}^{\alpha_r,k}_{\mathcal{F}_{i-1},\mathcal{H}_{i-1}}|$.

Combining Eqs. (6.35) with (6.36), the corresponding concept similarity in the $(i-1)$-th state can be described as

$$\begin{bmatrix} S(C_i, \mathcal{G}^{\alpha_1}_{i-1}) \\ \vdots \\ S(C_i, \mathcal{G}^{\alpha_n}_{i-1}) \end{bmatrix} = \begin{bmatrix} S(C_i, \mathcal{G}^{\alpha_1,1}_{i-1}) & \cdots & S(C_i, \mathcal{G}^{\alpha_1,l}_{i-1}) \\ \vdots & \ddots & \vdots \\ S(C_i, \mathcal{G}^{\alpha_n,1}_{i-1}) & \cdots & S(C_i, \mathcal{G}^{\alpha_n,l}_{i-1}) \end{bmatrix}, \tag{6.37}$$

where $S(C_i, \mathcal{G}^{\alpha_r}_{i-1}) = Sim(C_i, \mathcal{G}^{\alpha_r}_{\mathcal{F}_{i-1},\mathcal{H}_{i-1}})$ $(r \in \{1, 2, \ldots, n\})$ and $S(C_i, \mathcal{G}^{\alpha_r,k}_{i-1}) = Sim(C_i, \mathcal{G}^{\alpha_r,k}_{\mathcal{F}_{i-1},\mathcal{H}_{i-1}})$.

Furthermore, inspired by [79], the category similarity function between the given concept C_i and a class space $\mathcal{G}^{\alpha_r,k}_{\mathcal{F}_{i-1},\mathcal{H}_{i-1}}$ can be defined as

$$\phi_{Sim}(C_i, \mathcal{G}^{\alpha_r,k}_{\mathcal{F}_{i-1},\mathcal{H}_{i-1}}) = \frac{|N^{\alpha_r}_k(C_i)|}{K}, \tag{6.38}$$

where $N^{\alpha_r}_k(C_i) = \{C_j | C_j \in \mathcal{G}^{\alpha_r,k}_{\mathcal{F}_{i-1},\mathcal{H}_{i-1}} \wedge C_j \in N^{\alpha_r}_K(C_i)\}$, and $N^{\alpha_r}_K(C_i)$ is a set of near neighbor instances related to $\boldsymbol{x}_i$ under the parameter α_r.

According to top-K set similarity [77], if $\widehat{k} = \arg\max_{k\in\mathcal{Y}} \frac{|N^{\alpha_r}_k(C_i)|}{K}$, then the instance $\boldsymbol{x}_i$ is classified into the $\widehat{k}$-th class. Therefore, given the parameter K, the objective function can be formulated as

$$\begin{aligned} \widehat{\alpha}_r &= \underset{\alpha_r\in[0,1]}{\arg\min} \sum_{i=1}^{m} \Big(\frac{|N^{\alpha_r}_k(C_i)|}{K} - y_i\Big)^2 \\ \text{s.t.} &\sum_{r=1}^{n} \alpha_r = 1. \end{aligned} \tag{6.39}$$

In Eq. (6.39), our aim is to capture an optimal concept space with the concept structural information.

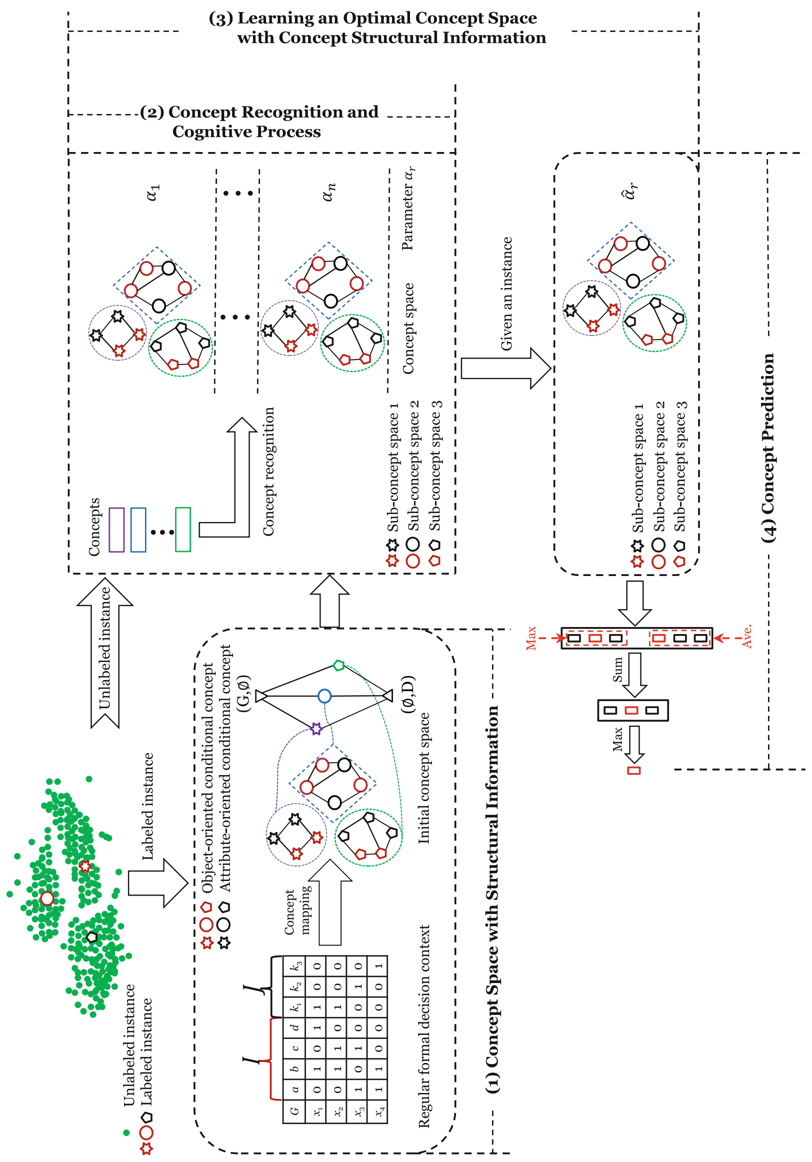
6.1.3.5 Framework and Computational Complexity Analysis

For brevity, we can consider that there are three classes to predict. Figure 6.3 illustrates the whole procedure of S2CL. From a dataset (that contains a small set of labeled data and a large amount of unlabeled data), we first obtain a corresponding regular formal decision context. Then, the initial concept spaces (that include a conditional concept space and its corresponding decision concept space) with concept structural information will be constructed based on the cognitive operators. Specifically speaking, the conditional concept space contains three sub-concept spaces, where each sub-concept space is composed of different concepts. As shown in the stage of initial concept spaces of Fig. 6.3 (see the left of Fig. 6.3 for details), there exist three sub-concept spaces corresponding to three classes in a conditional concept space, and each sub-concept space contains two different types of concepts, namely object-oriented conditional concepts (indicated by red shapes in Fig. 6.3) and attribute-oriented conditional concepts (denoted by black shapes in Fig. 6.3). Meanwhile, each sub-concept space is also associated with a decision concept in the corresponding decision concept space as shown in the first stage of Fig. 6.3. Thirdly, for any newly input unlabeled data, they are first used to form concepts, and then the concept-cognitive process is completed by concept recognition. Finally, given the parameter K, S2CL (or S2CL$^{\alpha}$) trys to learn an optimal concept space based on the concept recognition and concept-cognitive process under different parameters $\alpha_r (r = 1, 2, \ldots, n)$. In other words, the objective of S2CL (or S2CL$^{\alpha}$) is to seek an appropriate concept space to represent the underlying data distributions by the concept-cognitive process.

In the prediction stage, given an instance, the final concept space can produce two estimates of class distribution (including a maximum class vector and an average class vector) by employing the CS degree θ_j (or θ_j^I). Then the final CS degree vector will be obtained by the sum of the two 3-dimensional class vectors, and the class with maximum value will be output as shown in Fig. 6.3.

Based on the above discussion, we are ready to propose the corresponding algorithm of S2CL (see Algorithm 6.10 for details). In Algorithm 6.10, Step 3 is to generate the initial concept spaces; then the concept recognition and concept-cognitive process are conducted by running Steps 4–8; at last, the final prediction will be completed by Steps 9–12. In Steps 9–12, if the prediction value $\widehat{k}$ is consistent with the ground truth label, then it means that the predicted value of S2CL is correct. Formally, the accuracy on a test dataset T can be descried as $acc = \frac{N}{|T|}$, where N denotes the number of correct predicted values. Simultaneously, it will be easy to obtain the algorithm of S2CL$^{\alpha}$ by means of replacing the structural concept similarity θ_j with θ_j^I in Step 6 of Algorithm 6.10.

The time complexity of S2CL is mainly composed of two parts, i.e. constructing the initial concept spaces and the concept-cognitive process with concept structural information. Let the time complexity of constructing a concept, computing the CS degree and updating the concept space be $O(t_1)$, $O(t_2)$ and $O(t_3)$, respectively. Then, it is easy to verify that the time complexity of Step 3 is $O(t_1|S_L|(|M|+|D|))$, and the complexity of accomplishing the concept-cognitive process by concept recognition is $O(|S_U|(t_1 + t_2 + t_3))$. Note that, CCL is an incremental learning process, as the proposed method is updated by inputting objects one by one. Therefore, S2CL can also be regarded as an incremental method for SSL in dynamic environments. For convenience, let E and C (that randomly selects instances from



G	a	b	c	d	k_1	k_2	k_3
x_1	0	1	0	1	1	0	0
x_2	0	1	1	0	1	0	0
x_3	1	0	1	0	0	1	0
x_4	1	1	0	0	0	0	1

Fig. 6.3 Illustration of the framework of the proposed methods. Considering that there are three classes to predict, the concept spaces with concept structural information will be constructed, which include three different sub-concept spaces

S_U) be the incremental learning step and the sample size of each incremental learning step, respectively. Thus, the time complexity of incremental learning (see Algorithm 6.11 for details) is $O(E(|S_U|(t_1 + t_2 + t_3)) + |T|)$.

Algorithm 6.10 S2CL algorithm

```
1: **Input:** Labeled dataset $S_L$, unlabeled dataset $S_U$, test dataset $T$, and hyperparameters $K$ and $\alpha_r$.
2: **Output:** The class labels of the test dataset $T$.
3: Based on labeled dataset $S_L$, S2CL can construct two initial concept spaces $\mathcal{G}_{\mathcal{F}_0\mathcal{H}_0}$ and $\mathcal{G}_{\widetilde{\mathcal{F}}_0\widetilde{\mathcal{H}}_0}$ by Theorem 6.2.
4: for each $x_i \in S_U$ do
5:     Get two concepts $(\{x_i\}, \mathcal{F}_{\Delta G_i^{k*}}(x_i))$ and $(\{x_i\}, \widetilde{\mathcal{F}}_{\Delta G_i^{k*}}(x_i))$.
6:     Compute the CS degree by Eq. (6.33) (or Eq. (6.34)).
7:     Update concept spaces $\mathcal{G}^{\alpha_r}_{\mathcal{F}_{i-1}\mathcal{H}_{i-1}}$ and $\mathcal{G}^{\alpha_r}_{\widetilde{\mathcal{F}}_{i-1}\widetilde{\mathcal{H}}_{i-1}}$ by Theorem 6.3.
8: end for
9: for each $x_j \in T$ do
10:     Construct a concept $C_j = (\{x_j\}, B_j)$.
11:     $\widehat{k} = \arg\max_{k \in \mathcal{Y}} \frac{|N_k^{\alpha_r}(C_j)|}{K}$.
12: end for
13: return class labels.
```

Algorithm 6.11 Incremental learning

```
1: function INCREMENTALLEARNINGMETHOD
2:     for $e$=1 to $E$ do
3:         Conduct the same operation as Steps 4–8 of Algorithm 6.10.
4:         Conduct the same operation as Steps 9–12 of Algorithm 6.10.
5:     end for
6:     return class labels.
7: end function
```

6.1.4 Fuzzy-Based Concept Learning Method: Exploiting Data with Fuzzy Conceptual Clustering

6.1.4.1 Preliminaries

In this subsection, we review some notions related to the fuzzy formal decision context.

In a classical formal decision context, the conditional attributes are discrete. However, in the real world, many tasks (e.g., classification, image segmentation, etc.) are described with numerical (or fuzzy) data, which means that classical formal decision contexts cannot cope with them directly. Therefore, a fuzzy formal decision context is proposed based on fuzzy sets [81].

Let G be a universe of discourse. A fuzzy set $\widetilde{X}$ on G can be defined as follows:

$$\widetilde{X} = \{< x, \mu_{\widetilde{X}}(x) > | x \in G\},$$

where $\mu_{\widetilde{X}} : G \to [0, 1]$, and $\mu_{\widetilde{X}}(x)$ is referred to as the membership degree to $\widetilde{X}$ of the object $x \in G$. And we denote by L^G the set of all fuzzy sets on G.

Definition 6.24 ([83]) A fuzzy formal context $(G, M, \widetilde{I})$ is a triple, where G is a set of objects, M is a set of attributes, and $\widetilde{I}$ is a fuzzy relation between G and M. Each relation $(x, a) \in \widetilde{I}$ has a membership degree $\mu_{\widetilde{I}}(x, a)$ in [0, 1], and we denote by $\widetilde{I}(x, a) = \mu_{\widetilde{I}}(x, a)$ for the sake of convenience.

Definition 6.25 ([4, 78, 83]) Let $(G, M, \widetilde{I})$ be a fuzzy formal context. For $X \subseteq G$ and $\widetilde{B} \in L^M$, the operator $(\cdot)^*$ is defined as follows:

$$\begin{aligned} X^*(a) &= \bigwedge_{x \in X} \widetilde{I}(x, a), a \in M, \\ \widetilde{B}^* &= \{x \in G | \forall a \in M, \widetilde{B}(a) \le \widetilde{I}(x, a)\}. \end{aligned} \tag{6.40}$$

Then, we say that a pair $(X, \widetilde{B})$ is a fuzzy concept of a fuzzy formal context $(G, M, \widetilde{I})$ if $X^* = \widetilde{B}$, $\widetilde{B}^* = X$, and X and $\widetilde{B}$ are respectively known as the extent and intent of the fuzzy concept $(X, \widetilde{B})$. For convenience, the set of all fuzzy concepts is denoted by $L(G, M, \widetilde{I})$. In [83], $L(G, M, \widetilde{I})$ is called a special crisp-fuzzy variable threshold concept lattice under the circumstance of the threshold being set to be 1. For $(X_1, \widetilde{B}_1), (X_2, \widetilde{B}_2) \in L(G, M, \widetilde{I})$, we define the order relation $(X_1, \widetilde{B}_1) \le (X_2, \widetilde{B}_2)$ if and only if $X_1 \subseteq X_2$ (or $\widetilde{B}_2 \subseteq \widetilde{B}_1$). Then we say that $(X_1, \widetilde{B}_1)$ is a sub-concept of $(X_2, \widetilde{B}_2)$ and $(X_2, \widetilde{B}_2)$ is a super-concept of $(X_1, \widetilde{B}_1)$.

Definition 6.26 ([55]) Let $(G, M, \widetilde{I})$ and $(G, D, \widetilde{J})$ be two fuzzy formal contexts, where $\widetilde{I} : G \times M \to [0, 1]$ and $\widetilde{J} : G \times D \to [0, 1]$. Then $(G, M, \widetilde{I}, D, \widetilde{J})$ is referred to as a fuzzy formal decision context, where $M \cap D = \emptyset$, and M and D are the conditional and decision attribute sets, respectively.

Note that a quintuple $(G, M, I, D, \widetilde{J})$ is called a crisp-fuzzy formal decision context in [48], where (G, M, I) and $(G, D, \widetilde{J})$ are respectively a classical formal context and fuzzy formal context.

6.1.4.2 Fuzzy Concept Learning Method

In this subsection, we first show some new notions and properties for the proposed FCLM, which includes a regular fuzzy formal decision context, an object-oriented fuzzy conceptual clustering, and the related theoretical analysis. Based on them, we further present the detailed procedure of FCLM.

	$\tilde{I}$					$\tilde{J}$	
G	a	b	c	d	e	k_1	k_2
x_1	0.5	0.7	0.7	0.5	0.7	0.6	0.5
x_2	0.6	0.7	1.0	0.5	1.0	0.7	0.8
x_3	1.0	0.9	1.0	0.9	1.0	0.9	0.4

(a)

	I					$\tilde{J}$	
G	a	b	c	d	e	k_1	k_2
x_1	0	1	1	0	1	0.6	0.5
x_2	0	1	1	0	1	0.7	0.8
x_3	1	1	1	1	1	0.9	0.4

(b)

	$\tilde{I}$					J	
G	a	b	c	d	e	k_1	k_2
x_1	0.5	0.7	0.7	0.5	0.7	1	1
x_2	0.6	0.7	1.0	0.5	1.0	1	1
x_3	1.0	0.9	1.0	0.1	1.0	1	0

(c)

Fig. 6.4 Illustration of three different forms of fuzzy formal decision contexts

A. Regular Fuzzy Formal Decision Context

According to Definition 6.26, Fig. 6.4a and b represents two different fuzzy formal decision contexts $(G, M, \widetilde{I}, D, \widetilde{J})$ and $(G, M, I, D, \widetilde{J})$, respectively. More precisely, Fig. 6.4a expresses a fuzzy formal decision context in which M and D are both numerical; Fig. 6.4b denotes a fuzzy formal decision context, where M is discrete and D is numerical. However, in the real application, most original data are often presented in the form of Fig. 6.4c. It means that the decision attribute set D is described with discrete label information and the conditional attribute set M is constitutive of fuzzy data.

Definition 6.27 Let $(G, M, \widetilde{I})$ be a fuzzy formal context and (G, D, J) be a classical formal context. Then the quintuple $(G, M, \widetilde{I}, D, J)$ is known as a fuzzy-crisp formal decision context, where $\widetilde{I} : G \times M \to [0, 1]$ and $J : G \times D \to \{0, 1\}$.

Definition 6.28 Let $(G, M, \widetilde{I}, D, J)$ be a fuzzy-crisp formal decision context. For any $k_1, k_2 \in D$, if $\mathcal{H}^d(k_1) \cap \mathcal{H}^d(k_2) = \emptyset$, then we say that $(G, M, \widetilde{I}, D, J)$ is a regular fuzzy-crisp formal decision context.

Generally speaking, constructing a fuzzy concept lattice in a standard fuzzy context is sometimes quite complicated, as it is completed in exponential time complexity in the worst case. Hence, GrC should be introduced into the process of generating fuzzy concept lattices for greatly reducing the amount of calculation.

Let $(G, M, \widetilde{I})$ be a fuzzy formal context. $\widetilde{\mathcal{F}}^c : 2^G \to L^M$ and $\widetilde{\mathcal{H}}^c : L^M \to 2^G$ are supposed to be two mappings. Hence, $X^*(a)$ and $\widetilde{B}^*$ in Definition 4 can be rewritten as $\widetilde{\mathcal{F}}^c(X)(a)$ and $\widetilde{\mathcal{H}}^c(\widetilde{B})$, respectively. Especially, for an object set $\{x\}(x \in G)$, $\widetilde{\mathcal{F}}^c(\{x\})(a)$ is abbreviated as $\widetilde{\mathcal{F}}^c(x)$ for brevity.

Definition 6.29 Let $(G, M, \widetilde{I}, D, J)$ be a fuzzy-crisp formal decision context, and $\widetilde{\mathcal{F}}^c : 2^G \to L^M$,$\widetilde{\mathcal{H}}^c : L^M \to 2^G$ and $\mathcal{F}^d : 2^G \to 2^D$,$\mathcal{H}^d : 2^D \to 2^G$ be four mappings. For any $x \in G$, $(\widetilde{\mathcal{H}}^c\widetilde{\mathcal{F}}^c(x), \widetilde{\mathcal{F}}^c(x))$ and $(\mathcal{H}^d\mathcal{F}^d(x), \mathcal{F}^d(x))$ are called a fuzzy conditional granular concept and classical decision granular concept, respectively. The sets of all fuzzy conditional granular concepts and classical decision granular concepts are respectively represented as follows:

$$\mathcal{G}_{\widetilde{\mathcal{F}}^c\widetilde{\mathcal{H}}^c} = \{(\widetilde{\mathcal{H}}^c\widetilde{\mathcal{F}}^c(x), \widetilde{\mathcal{F}}^c(x))|x \in G\},$$

$$\mathcal{G}_{\mathcal{F}^d\mathcal{H}^d} = \{(\mathcal{H}^d\mathcal{F}^d(x), \mathcal{F}^d(x))|x \in G\},$$

where $\mathcal{G}_{\widetilde{\mathcal{F}}^c\widetilde{\mathcal{H}}^c}$ and $\mathcal{G}_{\mathcal{F}^d\mathcal{H}^d}$ are respectively referred to as the fuzzy conditional concept space and classical decision concept space.

It should be pointed out that fuzzy concept lattice has a good performance on classification but is very time-consuming. To the best of our knowledge, the reason is that fuzzy concept lattice may consist of many redundant elements. So, similar to classical concept lattice, it is better to replace fuzzy concept lattice with fuzzy concept space (only containing part of elements of fuzzy concept lattice) in achieving classification tasks with the purpose of improving learning efficiency.

Property 6.5 Let $(G, M, \widetilde{I}, D, J)$ be a fuzzy-crisp formal decision context. For any $(X_1, \widetilde{B}) \in \mathcal{G}_{\widetilde{\mathcal{F}}^c\widetilde{\mathcal{H}}^c}$ and $(X_2, K) \in \mathcal{G}_{\mathcal{F}^d\mathcal{H}^d}$, if $X_1 \subseteq X_2$, and X_1, $\widetilde{B}$, X_2 and K are nonempty, then the object set X_1 is connected with the decision attribute set K under the conditional attribute set $\widetilde{B}$.

Proof The proof is immediate from Definition 6.3 and Property 6.2. □

From Definition 6.29 and Property 6.5, we know that an object can also be connected with a label in a fuzzy-crisp formal decision context.

Based on the above discussion, the complete algorithm of constructing two concept spaces (including a fuzzy conditional concept space and classical decision concept space) is presented in Algorithm 6.12.

Algorithm 6.12 Constructing two concept spaces

```
1: Input: A dataset $G$.
2: Output: The fuzzy conditional concept space $\mathcal{G}_{\widetilde{\mathcal{F}}^c\widetilde{\mathcal{H}}^c}$ and classical decision concept space
   $\mathcal{G}_{\mathcal{F}^d\mathcal{H}^d}$.
3: for each $x \in G$ do
4:     % Construct a conditional concept space.
5:     Construct a fuzzy concept $(\widetilde{\mathcal{H}}^c\widetilde{\mathcal{F}}^c(x), \widetilde{\mathcal{F}}^c(x))$.
6:     $\mathcal{G}_{\widetilde{\mathcal{F}}^c\widetilde{\mathcal{H}}^c} \leftarrow (\widetilde{\mathcal{H}}^c\widetilde{\mathcal{F}}^c(x), \widetilde{\mathcal{F}}^c(x))$.
7:     % Construct a decision concept space.
8:     Construct a classical concept $(\mathcal{H}^d\mathcal{F}^d(x), \mathcal{F}^d(x))$.
9:     $\mathcal{G}_{\mathcal{F}^d\mathcal{H}^d} \leftarrow (\mathcal{H}^d\mathcal{F}^d(x), \mathcal{F}^d(x))$.
10: end for
11: Return $\mathcal{G}_{\widetilde{\mathcal{F}}^c\widetilde{\mathcal{H}}^c}$ and $\mathcal{G}_{\mathcal{F}^d\mathcal{H}^d}$.
```

B. Object-Oriented Fuzzy Conceptual Clustering

In order to generate fuzzy ontologies, a fuzzy conceptual clustering [50] was adopted in [67]. In fact, it was based on a crisp-crisp variable threshold concept lattice and implemented conceptual clustering via fuzzy sets intersection and union. However, to adapt to granular concepts based on a crisp-fuzzy variable threshold concept lattice, we need to consider the following notions.

Let $(G, M, \widetilde{I})$ be a fuzzy formal context. For any $(X, \widetilde{B}) \in \mathcal{G}_{\widetilde{\mathcal{F}}^c \widetilde{\mathcal{H}}^c}$, $|X|$ is called the object-oriented cardinality with reference to $(X, \widetilde{B})$.

Definition 6.30 Let $(X_j, \widetilde{B}_j)$ be a fuzzy granular concept and $(X_i, \widetilde{B}_i)$ be its sub-concept, then the object-oriented fuzzy concept similarity (object-oriented FCS) is defined as follows:

$$\theta^o = C^O(X_i, X_j) = \frac{|X_i \bigcap X_j|}{|X_i \bigcup X_j|}. \tag{6.41}$$

Definition 6.31 Let $(X_j, \widetilde{B}_j)$ and $(X_l, \widetilde{B}_l)$ be two fuzzy granular concepts, then the attribute-oriented fuzzy concept similarity (attribute-oriented FCS) is defined as follows:

$$\theta^a = C^A(\widetilde{B}_j, \widetilde{B}_l) = ||\widetilde{B}_j - \widetilde{B}_l||_2^2. \tag{6.42}$$

Definition 6.32 Let $\mathcal{G}^{S_\lambda}_{\widetilde{\mathcal{F}}^c \widetilde{\mathcal{H}}^c}$ be a sub-concept space of $\mathcal{G}_{\widetilde{\mathcal{F}}^c \widetilde{\mathcal{H}}^c}$. For any $(X_i, \widetilde{B}_i) \in \mathcal{G}^{S_\lambda}_{\widetilde{\mathcal{F}}^c \widetilde{\mathcal{H}}^c}$, we say that $\mathcal{G}^{S_\lambda}_{\widetilde{\mathcal{F}}^c \widetilde{\mathcal{H}}^c}$ is an object-oriented conceptual cluster of the concept space with an object-oriented FCS threshold λ if the following properties hold:

1. There exists a supremum concept $(X_p, \widetilde{B}_p) \in \mathcal{G}^{S_\lambda}_{\widetilde{\mathcal{F}}^c \widetilde{\mathcal{H}}^c}$ that is not similar to any of its super-concepts.
2. There exists at least one super-concept $(X_j, \widetilde{B}_j) \in \mathcal{G}^{S_\lambda}_{\widetilde{\mathcal{F}}^c \widetilde{\mathcal{H}}^c}$ such that $C^O(X_i, X_j) > \lambda$ when $X_i \neq X_p$.
3. Any fuzzy concept $(X_i, \widetilde{B}_i)$ only belongs to one object-oriented conceptual cluster $\mathcal{G}^{S_\lambda}_{\widetilde{\mathcal{F}}^c \widetilde{\mathcal{H}}^c}$.

Definition 6.33 Let $\mathcal{G}^{S_\lambda}_{\widetilde{\mathcal{F}}^c \widetilde{\mathcal{H}}^c}$ be an object-oriented conceptual cluster. For $(X_1, \widetilde{B}_1)$, $(X_2, \widetilde{B}_2), \ldots, (X_p, \widetilde{B}_p) \in \mathcal{G}^{S_\lambda}_{\widetilde{\mathcal{F}}^c \widetilde{\mathcal{H}}^c} (p = |\mathcal{G}^{S_\lambda}_{\widetilde{\mathcal{F}}^c \widetilde{\mathcal{H}}^c}|)$, let $X_{S_\lambda} = \bigcup\limits_{i=1}^{p} X_i$ and $\widetilde{B}_{S_\lambda} = (\widetilde{B}_{S_\lambda}(a_1), \widetilde{B}_{S_\lambda}(a_2), \ldots, \widetilde{B}_{S_\lambda}(a_{|M|}))$, where $\widetilde{B}_{S_\lambda}(a_j) = \frac{1}{p} \sum\limits_{i=1}^{p} \widetilde{B}_i(a_j)$ $(j \in \{1, 2, \ldots, |M|\})$. Then we say that the crisp-fuzzy pair $(X_{S_\lambda}, \widetilde{B}_{S_\lambda})$ is a pseudo concept induced by the object-oriented conceptual cluster $\mathcal{G}^{S_\lambda}_{\widetilde{\mathcal{F}}^c \widetilde{\mathcal{H}}^c}$.

In what follows, the pseudo concept $(X_{S_\lambda}, \widetilde{B}_{S_\lambda})$ is called the representation of the object-oriented conceptual cluster $\mathcal{G}^{S_\lambda}_{\widetilde{\mathcal{F}}^c \widetilde{\mathcal{H}}^c}$. Note that the process of generating a new pseudo concept is known as concept generation. Hereinafter, we do not distinguish pseudo concepts from fuzzy concepts since pseudo concepts are only intermediate variables in the subsequent fuzzy conceptual clustering. In other words, sometimes we also call pseudo concepts as fuzzy concepts when no confusion exists.

Statistically speaking, Definition 6.33 can completely characterize a new fuzzy concept. However, in cognitive science, concept cognition was often considered to be incremental due to individual cognitive limitations and incomplete cognitive environments. Inspired by this issue, the process of constructing a new fuzzy concept can be rephrased as follows.

Definition 6.34 Let $(X_p, \widetilde{B}_p)$ be the supremum concept of $\mathcal{G}^{S_\lambda}_{\widetilde{\mathcal{F}}^c\widetilde{\mathcal{H}}^c}$. For $(X_1, \widetilde{B}_1)$, $(X_2, \widetilde{B}_2), \ldots, (X_p, \widetilde{B}_p) \in \mathcal{G}^{S_\lambda}_{\widetilde{\mathcal{F}}^c\widetilde{\mathcal{H}}^c}$, each dimension of the intent of a new fuzzy concept $(X_{S_\lambda}, \widetilde{B}_{S_\lambda})$ can be rewritten as follows:

$$\begin{aligned}\widetilde{B}_{S_\lambda}(a_j) = &\frac{1}{2^{p-1}}(\widetilde{B}_1(a_j) + \widetilde{B}_2(a_j) + 2\widetilde{B}_3(a_j) + \\ &4\widetilde{B}_4(a_j)+, \ldots, +2^{p-2}\widetilde{B}_p(a_j)),\end{aligned} \tag{6.43}$$

where $j \in \{1, 2, \ldots, |M|\}$.

Theorem 6.4 *Let $\widetilde{B}_{S_\lambda}(a_j)$ be any dimension of the intent of a new fuzzy concept $(X_{S_\lambda}, \widetilde{B}_{S_\lambda})$. Then we have*

$$\frac{\widetilde{B}_p(a_j)}{2} \leq \widetilde{B}_{S_\lambda}(a_j) \leq 1. \tag{6.44}$$

Proof It is immediate from Definitions 6.24 and 6.34. □

For any $(X_i, \widetilde{B}_i), (X_j, \widetilde{B}_j) \in \mathcal{G}^{S_\lambda}_{\widetilde{\mathcal{F}}^c\widetilde{\mathcal{H}}^c}$, if $(X_j, \widetilde{B}_j)$ is a super-concept of $(X_i, \widetilde{B}_i)$, we say that $(X_j, \widetilde{B}_j)$ presents more strongly conceptual representation ability than $(X_i, \widetilde{B}_i)$. Equation (6.43) represents that the process of incremental cognition for concept formation by means of the hierarchical relations between sub-concepts and super-concepts, and the coefficient of each dimension will be heighten along with the increase of conceptual representation ability. Equation (6.44) denotes that the upremum concept has a great influence on the process of constructing new fuzzy concepts.

Definition 6.35 Let $\mathcal{G}^{S_{\lambda,1}}_{\widetilde{\mathcal{F}}^c\widetilde{\mathcal{H}}^c}, \mathcal{G}^{S_{\lambda,2}}_{\widetilde{\mathcal{F}}^c\widetilde{\mathcal{H}}^c}, \ldots, \mathcal{G}^{S_{\lambda,m}}_{\widetilde{\mathcal{F}}^c\widetilde{\mathcal{H}}^c}$ be a partition of $\mathcal{G}_{\widetilde{\mathcal{F}}^c\widetilde{\mathcal{H}}^c}$ with an object-oriented FCS threshold λ. Then a new concept space can be defined as follows:

$$\mathcal{G}^{S_{\lambda,*}}_{\widetilde{\mathcal{F}}^c\widetilde{\mathcal{H}}^c} = \bigcup_{i=1}^{m} \mathcal{G}^{S_{\lambda,i}}_{\widetilde{\mathcal{F}}^c\widetilde{\mathcal{H}}^c} = \bigcup_{i=1}^{m} (X_{S_{\lambda,i}}, \widetilde{B}_{S_{\lambda,i}}). \tag{6.45}$$

Theorem 6.5 *Let $\mathcal{G}^{S_{\lambda,*}}_{\widetilde{\mathcal{F}}^c\widetilde{\mathcal{H}}^c}$ be a concept space with an object-oriented FCS threshold λ. We have*

$$1 \le |\mathcal{G}^{S_{\lambda,*}}_{\widetilde{\mathcal{F}}^c\widetilde{\mathcal{H}}^c}| \le |\mathcal{G}_{\widetilde{\mathcal{F}}^c\widetilde{\mathcal{H}}^c}|. \tag{6.46}$$

Proof The proof of Theorem 6.5 can be found in the original paper [43]. □

Based on the above theory, the procedure of object-oriented fuzzy conceptual clustering is summarized in Algorithm 6.13.

Algorithm 6.13 Object-oriented fuzzy conceptual clustering method

```
1:  Input: A fuzzy concept space $\mathcal{G}_{\widetilde{\mathcal{F}}^c\widetilde{\mathcal{H}}^c}$ and an object-oriented FCS threshold $\lambda$.
2:  Output: A new fuzzy conceptual cluster space $\mathcal{G}^{S_{\lambda,*}}_{\widetilde{\mathcal{F}}^c\widetilde{\mathcal{H}}^c}$.
3:  $\mathcal{G}^{S_{\lambda,*}}_{\widetilde{\mathcal{F}}^c\widetilde{\mathcal{H}}^c} = \emptyset$ and $\mathcal{G}^{S_{\lambda,i}}_{\widetilde{\mathcal{F}}^c\widetilde{\mathcal{H}}^c} = \emptyset$.
4:  $\mathcal{G}^{S_{\lambda,i}}_{\widetilde{\mathcal{F}}^c\widetilde{\mathcal{H}}^c} \leftarrow (X_p, \widetilde{B}_p)$.
5:  for each sub-concept $(X_j, \widetilde{B}_j) \in \mathcal{G}_{\widetilde{\mathcal{F}}^c\widetilde{\mathcal{H}}^c}$ of $(X_p, \widetilde{B}_p)$ do
6:      Get $\theta^O$ by Definition 6.30.
7:      if $\theta^O \ge \lambda$ then
8:          $\mathcal{G}^{S_{\lambda,i}}_{\widetilde{\mathcal{F}}^c\widetilde{\mathcal{H}}^c} \leftarrow (X_j, \widetilde{B}_j)$.
9:      end if
10: end for
11: Construct a new fuzzy concept $(X_{S_{\lambda,i}}, \widetilde{B}_{S_{\lambda,i}})$ by Definition 6.34.
12: $\mathcal{G}^{S_{\lambda,*}}_{\widetilde{\mathcal{F}}^c\widetilde{\mathcal{H}}^c} = \mathcal{G}^{S_{\lambda,*}}_{\widetilde{\mathcal{F}}^c\widetilde{\mathcal{H}}^c} \bigcup (X_{S_{\lambda,i}}, \widetilde{B}_{S_{\lambda,i}})$.
13: Return $\mathcal{G}^{S_{\lambda,*}}_{\widetilde{\mathcal{F}}^c\widetilde{\mathcal{H}}^c}$.
```

6.1.4.3 Theoretical Analysis

From Definition 6.35 and Theorem 6.5, we know that the object-oriented FCS threshold has a significant impact on the construction of a new concept space. Hence, it is very necessary to select an optimal (or approximate optimal) λ for each dataset.

Let $\lambda = \lambda(i)$ $(i \in \{1, 2, \dots, n\})$, and $\lambda(i) \propto i$. For all the newly constructed concept spaces with different $\lambda(i)$, we denote

$$\begin{bmatrix} \mathcal{G}^{S_{\lambda(1),*}}_{\widetilde{\mathcal{F}}^c\widetilde{\mathcal{H}}^c} \\ \mathcal{G}^{S_{\lambda(2),*}}_{\widetilde{\mathcal{F}}^c\widetilde{\mathcal{H}}^c} \\ \vdots \\ \mathcal{G}^{S_{\lambda(n),*}}_{\widetilde{\mathcal{F}}^c\widetilde{\mathcal{H}}^c} \end{bmatrix} = \begin{bmatrix} \mathcal{G}^{S_{\lambda(1),1}}_{\widetilde{\mathcal{F}}^c\widetilde{\mathcal{H}}^c} & \mathcal{G}^{S_{\lambda(1),2}}_{\widetilde{\mathcal{F}}^c\widetilde{\mathcal{H}}^c} & \cdots & \mathcal{G}^{S_{\lambda(1),m_1}}_{\widetilde{\mathcal{F}}^c\widetilde{\mathcal{H}}^c} \\ \mathcal{G}^{S_{\lambda(2),1}}_{\widetilde{\mathcal{F}}^c\widetilde{\mathcal{H}}^c} & \mathcal{G}^{S_{\lambda(2),2}}_{\widetilde{\mathcal{F}}^c\widetilde{\mathcal{H}}^c} & \cdots & \mathcal{G}^{S_{\lambda(2),m_2}}_{\widetilde{\mathcal{F}}^c\widetilde{\mathcal{H}}^c} \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{G}^{S_{\lambda(n),1}}_{\widetilde{\mathcal{F}}^c\widetilde{\mathcal{H}}^c} & \mathcal{G}^{S_{\lambda(n),2}}_{\widetilde{\mathcal{F}}^c\widetilde{\mathcal{H}}^c} & \cdots & \mathcal{G}^{S_{\lambda(n),mn}}_{\widetilde{\mathcal{F}}^c\widetilde{\mathcal{H}}^c} \end{bmatrix}, \tag{6.47}$$

where $m_i = |\mathcal{G}^{S_{\lambda(i),*}}_{\widetilde{\mathcal{F}}^c\widetilde{\mathcal{H}}^c}|$, and $\mathcal{G}^{S_{\lambda(i),*}}_{\widetilde{\mathcal{F}}^c\widetilde{\mathcal{H}}^c}$ is computed with $\lambda(i)$.

In Eq. (6.47), we say that $\mathcal{G}_{\widetilde{\mathcal{F}}^c\widetilde{\mathcal{H}}^c}^{S_{\lambda(i),j}}(j \in \{1, 2, \ldots, m_i\})$ is a conceptual subcluster of $\mathcal{G}_{\widetilde{\mathcal{F}}^c\widetilde{\mathcal{H}}^c}^{S_{\lambda(i),*}}$. Meanwhile, according to Definition 6.33, each object-oriented conceptual cluster can be represented as a new fuzzy concept. Hence, Eq. (6.47) can be rewritten as Eq. (6.48).

Note that there is only the fuzzy conditional concept space $\mathcal{G}_{\widetilde{\mathcal{F}}^c\widetilde{\mathcal{H}}^c}$ which will be influenced by the object-oriented FCS threshold $\lambda(i)$. The concept space $\mathcal{G}_{\widetilde{\mathcal{F}}^c\widetilde{\mathcal{H}}^c}^{S_{\lambda(i),*}}$ can be simplified by omitting the suffix $\widetilde{\mathcal{F}}^c\widetilde{\mathcal{H}}^c$ when no confusion exists, namely $\mathcal{G}^{S_{\lambda(i),*}}$.

Property 6.6 Let $\mathcal{G}^{S_{\lambda(i),*}}$ be a set of object-oriented conceptual clusters with the object-oriented FCS threshold $\lambda(i)$. Then we have

$$\begin{bmatrix} \mathcal{G}_{\widetilde{\mathcal{F}}^c\widetilde{\mathcal{H}}^c}^{S_{\lambda(1),*}} \\ \mathcal{G}_{\widetilde{\mathcal{F}}^c\widetilde{\mathcal{H}}^c}^{S_{\lambda(2),*}} \\ \vdots \\ \mathcal{G}_{\widetilde{\mathcal{F}}^c\widetilde{\mathcal{H}}^c}^{S_{\lambda(n),*}} \end{bmatrix} = \begin{bmatrix} \left(X_{S_{\lambda(1),1}}, \widetilde{B}_{S_{\lambda(1),1}}\right) & \left(X_{S_{\lambda(1),2}}, \widetilde{B}_{S_{\lambda(1),2}}\right) & \cdots & \left(X_{S_{\lambda(1),m_1}}, \widetilde{B}_{S_{\lambda(1),m_1}}\right) \\ \left(X_{S_{\lambda(2),1}}, \widetilde{B}_{S_{\lambda(2),1}}\right) & \left(X_{S_{\lambda(2),2}}, \widetilde{B}_{S_{\lambda(2),2}}\right) & \cdots & \left(X_{S_{\lambda(2),m_2}}, \widetilde{B}_{S_{\lambda(2),m_2}}\right) \\ \vdots & \vdots & \ddots & \vdots \\ \left(X_{S_{\lambda(n),1}}, \widetilde{B}_{S_{\lambda(n),1}}\right) & \left(X_{S_{\lambda(n),2}}, \widetilde{B}_{S_{\lambda(n),2}}\right) & \cdots & \left(X_{S_{\lambda(n),m_n}}, \widetilde{B}_{S_{\lambda(n),m_n}}\right) \end{bmatrix}. \tag{6.48}$$

$$|\mathcal{G}^{S_{\lambda(i),*}}| \propto \lambda(i). \tag{6.49}$$

Proof The proof can be derived by means of $\lambda(i) = \lambda$, and Definition 6.35. □

In the above discussion, we only consider the situation that there exists one concept cluster in FCLM. However, in the real-life world, studying the situation of multiple concept clusters with the label information is also highly desirable, as there are at least two concept clusters for classification tasks.

We denote by $G = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m\}$ a set of instances and $\mathcal{K} = \{1, 2, \ldots, l\}$ the label space. There does exist a partition of the instances into l clusters $C_1, C_2, \ldots, C_l$ by means of the label information such that they can cover all the instances, and formally, $C_1 \cup C_2 \cup \cdots \cup C_l = G$, where $C_i \cap C_j = \emptyset$ $(\forall i \neq j)$. Meanwhile, we denote the corresponding fuzzy conceptual clusters by $\mathcal{G}_1^{S_{\lambda(i),*}}, \mathcal{G}_2^{S_{\lambda(i),*}}, \ldots, \mathcal{G}_l^{S_{\lambda(i),*}}$ with $\lambda(i)$. Moreover, the set of all fuzzy conceptual clusters with $\lambda(i)$ is denoted by $\mathcal{C}^{S_{\lambda(i)}}$, namely $\mathcal{C}^{S_{\lambda(i)}} = \{\mathcal{G}_1^{S_{\lambda(1),*}}, \mathcal{G}_2^{S_{\lambda(1),*}}, \cdots, \mathcal{G}_l^{S_{\lambda(1),*}}\}$. For different object-oriented FCS thresholds, we further denote

$$\begin{bmatrix} \mathcal{C}^{S_{\lambda(1)}} \\ \mathcal{C}^{S_{\lambda(2)}} \\ \vdots \\ \mathcal{C}^{S_{\lambda(n)}} \end{bmatrix} = \begin{bmatrix} \mathcal{G}_1^{S_{\lambda(1),*}} & \mathcal{G}_2^{S_{\lambda(1),*}} & \cdots & \mathcal{G}_l^{S_{\lambda(1),*}} \\ \mathcal{G}_1^{S_{\lambda(2),*}} & \mathcal{G}_2^{S_{\lambda(2),*}} & \cdots & \mathcal{G}_l^{S_{\lambda(2),*}} \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{G}_1^{S_{\lambda(n),*}} & \mathcal{G}_2^{S_{\lambda(n),*}} & \cdots & \mathcal{G}_l^{S_{\lambda(n),*}} \end{bmatrix}. \tag{6.50}$$

Our aim is to select an optimal $\lambda(i)$ in the interval [0,1] for each dataset. Let $(X_r, \widetilde{B}_r)$ $(r \in \{1, 2, \ldots, m\})$ be a fuzzy granular concept. Then, the objective function can be formulated as

$$
\begin{aligned}
E(\lambda(i), j) = &\min_{i\in\mathcal{I}, j\in\mathcal{J}, k'} \sum_{r=1}^{m} ||(X_r, \widetilde{B}_r) - \mathcal{G}_{k'}^{S_{\lambda(i),j}}||_2^2 - \\
&\max_{i\in\mathcal{I}} \min_{j\in\mathcal{J}} \sum_{k''\in\overline{\mathcal{K}}} \sum_{r=1}^{m} ||(X_r, \widetilde{B}_r) - \mathcal{G}_{k''}^{S_{\lambda(i),j}}||_2^2 \\
&\text{s.t. } m_i \propto \lambda(i), 0 \le \lambda(i) \le 1,
\end{aligned}
\tag{6.51}
$$

where $\mathcal{I} = \{1, 2, \ldots, n\}$, $\mathcal{J} = \{1, 2, \ldots, m_i\}$, $\overline{\mathcal{K}} = \mathcal{K} \setminus \{k'\}$, and k' represents the real class label of the fuzzy granular concept $(X_r, \widetilde{B}_r)$. Hence, in Eq. (6.51), the first item denotes that samples are classified into the ground truth conceptual subcluster, while the second item indicates the opposite situation.

Let $(X_{S_{\lambda(i),j}}^k, \widetilde{B}_{S_{\lambda(i),j}}^k)$ $(k \in \mathcal{K})$ be the representation of the conceptual subcluster $\mathcal{G}_k^{S_{\lambda(i),j}}$. For any fuzzy granular concept $(X_r, \widetilde{B}_r)$, it can be considered as an instance $\boldsymbol{x}_r$ with M-dimensional features. Therefore, according to Definition 6.31 and Eq. (6.48), the objective function can be reformulated as

$$
\begin{aligned}
E(\lambda(i), j) = &\min_{i\in\mathcal{I}, j\in\mathcal{J}, k'} \sum_{r=1}^{m} ||\widetilde{B}_r - \widetilde{B}_{S_{\lambda(i),j}}^{k'}||_2^2 - \\
&\max_{i\in\mathcal{I}} \min_{j\in\mathcal{J}} \sum_{k''\in\overline{\mathcal{K}}} \sum_{r=1}^{m} ||\widetilde{B}_r - \widetilde{B}_{S_{\lambda(i),j}}^{k''}||_2^2 \\
&\text{s.t. } m_i \propto \lambda(i), 0 \le \lambda(i) \le 1.
\end{aligned}
\tag{6.52}
$$

Based on Eq. (6.48) and Property 6.6, we know that the variable j is dependent on another variable $\lambda(i)$. Hence, we can optimize the objective function of our FCLM by means of updating $\lambda(i)$:

$$
\begin{aligned}
\widehat{\lambda}(i) = &\arg\min_{i\in\mathcal{I}, j\in\mathcal{J}} E(\lambda(i), j) \\
&\text{s.t. } m_i \propto \lambda(i), 0 \le \lambda(i) \le 1.
\end{aligned}
\tag{6.53}
$$

In theory, we can obtain an optimal $\widehat{\lambda}(i)$ by solving Eq. (6.53) directly. Unfortunately, it is quite difficult to obtain analytical solutions due to lacking of a concrete functional expression between m_i and $\lambda(i)$. Hence, we select an approximate optimal $\widehat{\lambda}(i)$ by a method similar to grid search. The complete procedure for selecting an approximate optimal solution (see Algorithm 6.14 for details) is proposed based on the above discussion.

Algorithm 6.14 Select $\widehat{\lambda}(i)$ for FCLM

```
1: Input: Training set $\overline{G}$, validation set $V$, and step size $\varepsilon$.
2: Output: An approximate optimal $\widehat{\lambda}(i)$.
3: Construct a fuzzy conditional concept space $\mathcal{G}_{\widetilde{\mathcal{F}}^c\widetilde{\mathcal{H}}^c}$ and classical decision concept space $\mathcal{G}_{\mathcal{F}^d\mathcal{H}^d}$ by Algorithm 6.12.
4: for $\lambda(i) = 0$ to 1 do
5:     Get $C^{S_{\lambda(i)}}$ by Algorithm 6.13.
6:     for $x_r \in V$ do
7:         Compute $E(\lambda(i), j)$ by Eq. (6.52).
8:     end for
9:     $\lambda(i)=\lambda(i)+\varepsilon$.
10: end for
11: Get $\widehat{\lambda}(i)$ by Eq. (6.53).
12: Return $\widehat{\lambda}(i)$.
```

6.2 Label Proportion for Learning

6.2.1 A Fast Algorithm for Multi-Class Learning from Label Proportions

Learning from label proportions (LLP) is a new kind of learning problem which has attracted wide interest in machine learning. Different from the well-known supervised learning, the training data of LLP is in form of bags and only the proportion of each class in each bag is available. In this subsection, we propose a fast algorithm called multi-class learning from label proportions by extreme learning machine (LLP-ELM), which takes advantage of extreme learning machine with fast learning speed to solve multi-class learning from label proportions.

6.2.1.1 Background

In this section, we give a brief introduction of the traditional extreme learning machine [21, 22]. Figure 6.5 shows the architecture of ELM. In detail, it is a single-hidden layer feed-forward networks with three parts: input neurons, hidden neurons and output neurons. In particular, $\mathbf{h}(\mathbf{x}) = [h_1(x), \ldots, h_L(x)]$ is nonlinear feature mapping of ELM with the form of $h_j(x) = g(\mathbf{w_j}.\mathbf{x} + b_j)$ and $\boldsymbol{\beta_j} = [\beta_{j1}, \ldots, \beta_{jc}]^T$, $j = 1, \ldots, L$ is the output weights between the jth hidden layer and the output nodes.

Given N samples $(x_i, t_i), i = 1, \ldots, N$, where $\mathbf{x_i} = [x_{i1}, \ldots, x_{id}]^T$ denotes the input feature vectors and $\mathbf{t_i} = [t_{i1}, \ldots, t_{ic}]^T$ is the corresponding label in a one-hot fashion. In particular, c and d respectively represent the total classes and feature

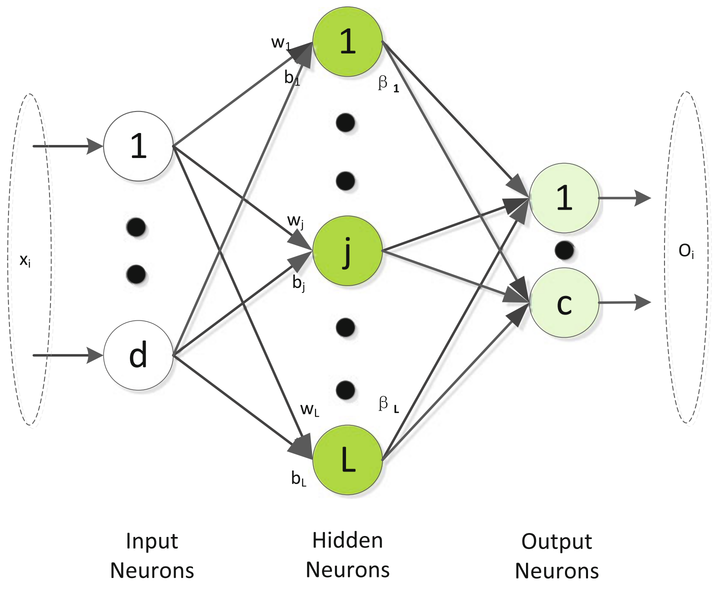


Fig. 6.5 The architecture of ELM. In detail, it is a single-hidden layer feed-forward network with three parts: input neurons, hidden neurons and output neurons. In particular, $\mathbf{h}(\mathbf{x}) = [h_1(x), \ldots, h_L(x)]$ is nonlinear feature mapping of ELM with the form of $h_j(x) = g(\mathbf{w_j}.\mathbf{x} + b_j)$ and $\boldsymbol{\beta_j} = [\beta_{j1}, \ldots, \beta_{jc}]^T, j = 1, \ldots, L$ is the output weights between the jth hidden layer and the output nodes

number. Consequently, a standard feed-forward neural network with L hidden nodes can be expressed as:

$$\sum_{j=1}^{L} \boldsymbol{\beta_j} g(\mathbf{w_j}.\mathbf{x_i} + b_j) = \mathbf{o_i}, i = 1, \ldots, N, \tag{6.54}$$

where $\mathbf{w_j} = [w_{j1}, w_{j2}, \ldots, w_{jL}]^T$ is the weight vector between the jth hidden neuron and the input neurons, and $\boldsymbol{\beta_j} = [\beta_{j1}, \beta_{j2}, \ldots, \beta_{jc}]^T, j = 1, \ldots, L$ is the weight vector connecting the output neuron and the jth hidden neurons. According to [21], the ELM can approximate those N samples to zero error with the equation $\sum_{i=1}^{N} \|\mathbf{o_i} - \mathbf{t_i}\| = 0$. Thus, the above equations can be expressed as:

$$\sum_{j=1}^{L} \boldsymbol{\beta_j} g(\mathbf{w_j}.\mathbf{x_i} + b_j) = \mathbf{t_i}, i = 1, \ldots, N. \tag{6.55}$$

In particular, we can use matrix to express the above N equations with form of:

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{T}, \tag{6.56}$$

where $\mathbf{H}$ is the hidden layer output matrix of the single-hidden layer feed-forward network and $\mathbf{T}$ is output matrix. More specifically, $\mathbf{H}$ and $\mathbf{T}$ have the form of:

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}(\mathbf{x_1}) \\ \vdots \\ \mathbf{h}(\mathbf{x_N}) \end{bmatrix} = \begin{bmatrix} h_1(\mathbf{x_1}) & \cdots & h_L(\mathbf{x_1}) \\ \vdots & \vdots & \vdots \\ h_1(\mathbf{x_N}) & \vdots & h_L(\mathbf{x_N}) \end{bmatrix} \tag{6.57}$$

and

$$\mathbf{T} = \begin{bmatrix} \mathbf{t_1^T} \\ \vdots \\ \mathbf{t_N^T} \end{bmatrix} = \begin{bmatrix} t_{11} & \cdots & t_{1c} \\ \vdots & \vdots & \vdots \\ t_{N1} & \vdots & t_{Nc} \end{bmatrix} \tag{6.58}$$

In practice, the hidden node parameters ($\mathbf{w}$,b) of ELM are randomly generated and then fixed without iteratively tuning, which is different to the traditional BP neural networks [21]. As a result, training an ELM is equivalent to find the optimal solution to $\boldsymbol{\beta}$, which is in defined as:

$$\boldsymbol{\beta} = \begin{bmatrix} \boldsymbol{\beta}_\mathbf{1}^\mathbf{T} \\ \vdots \\ \boldsymbol{\beta}_\mathbf{L}^\mathbf{T} \end{bmatrix} = \begin{bmatrix} \beta_{11} & \cdots & \beta_{1c} \\ \vdots & \vdots & \vdots \\ \beta_{L1} & \vdots & \beta_{Lc} \end{bmatrix} \tag{6.59}$$

Furthermore, $\boldsymbol{\beta}$ can computed by the following expression:

$$\boldsymbol{\beta}^* = \mathbf{H}^\dagger\mathbf{T} \tag{6.60}$$

where $\mathbf{H}^\dagger$ is the Moore-Penrose generalized inverse of matrix $\mathbf{H}$.

6.2.1.2 The LLP-ELM Algorithm

In this section, we propose a fast method for multi-class learning from label proportions algorithm called LLP-ELM, which employs extreme learning machine to solve multi-class LLP problem. In order to leverage extreme learning machine to LLP, we reshape the hidden layer output matrix $\mathbf{H}$ and the training data target matrix $\mathbf{T}$ to new forms, such that $\mathbf{H}$ is in bag level and $\mathbf{T}$ contains the proportion information instead of a label one.

A. Learning Setting

The LLP problem is described by a set of training data, which is divided into several bags. Furthermore, compared to the traditional supervised learning, we only know the proportions of different categories in each bag instead of the ground-truth labels. In this paper, we consider the situation that different bags are disjoint, and the nth bag of the training data can be denoted as $B_n, n = 1, \ldots, h$. Consequently, the total training data is in form of:

$$D = B_1 \cup B_2 \cup \ldots . \cup B_h \qquad (6.61)$$
$$B_i \cap B_j = \emptyset, \forall i \neq j.$$

where there are n bags and N is the number of total instances. Each bag consists of m_n instances with the constraint $\sum_{n=1}^{h} m_n = N$, and can be expressed as:

$$B_n = \{x_n^1, ..., x_n^{m_n}\}, n \in \{1, 2, \ldots, h\}. \qquad (6.62)$$

Meanwhile, $\mathbf{p_n}$ is the corresponding class proportion vector of B_n and c represents the total classes number. More specifically, $\mathbf{p_n}$ can be written as a vector form:

$$\mathbf{p_n} = \begin{bmatrix} p_{n1} \\ \vdots \\ p_{nc} \end{bmatrix}, \qquad (6.63)$$

where the mth element p_n^m is the proportion of the mth class in the nth bag with the constraint $\sum_{m=1}^{c} p_n^m = 1$. Furthermore, the total proportion information can be defined in form of matrix:

$$\mathbf{P} = \begin{bmatrix} \mathbf{p_1^T} \\ \vdots \\ \mathbf{p_h^T} \end{bmatrix} = \begin{bmatrix} p_{11} & \cdots & p_{1c} \\ \vdots & \vdots & \vdots \\ p_{h1} & \vdots & p_{hc} \end{bmatrix}. \qquad (6.64)$$

B. The LLP-ELM Framework

From the above learning setting of LLP, a classifier in instance level is the final objective. To this end, we modify the original equations in ELM to the new equations in bag level. Specifically, we add all the equations in each bag straightforward, and the final equations in nth bag can be expressed as follows:

$$\sum_{j=1}^{L} \sum_{k=1}^{m_n} \boldsymbol{\beta}_j g(\mathbf{w_j} . \mathbf{x_{nk}} + b_j) = \sum_{k=1}^{m_n} \mathbf{t_{nk}}, n = 1, \ldots, h \qquad (6.65)$$

where $\mathbf{t_{nk}}$ is the real label for the kth instances in nth bag. Obviously, the real label information in the right part is inaccessible to us, with only label proportions in each bag available. To this end, we derive the right part of the above equation as the following form:

$$\sum_{k=1}^{m_n} \mathbf{t_{nk}} = m_n * \mathbf{p_n}, n = 1, \ldots, h \tag{6.66}$$

where $\mathbf{p_n}$ is the label proportion of nth bag. Substituting the formula (6.66) to (6.65), we can naturally obtain the following equations:

$$\sum_{j=1}^{L} \boldsymbol{\beta}_j \sum_{k=1}^{m_n} g(\mathbf{w_j}.\mathbf{x_{nk}} + b_j) = m_n * \mathbf{p_j}, n = 1, \ldots, h, \tag{6.67}$$

In particular, similar to the method from ELM [21], we can write the above equations in the form of matrix computing as follows:

$$\mathbf{H_p}\boldsymbol{\beta} = \mathbf{P} \tag{6.68}$$

where $\mathbf{H_p}$ is the hidden layer output matrix in the bag level, and $\mathbf{P}$ is the training data target proportion matrix. More specifically, $\mathbf{H_p}$ and $\mathbf{P}$ are given in form of:

$$\mathbf{H_p} = \begin{bmatrix} \sum_{k=1}^{m_1} \mathbf{h}(\mathbf{x_{1k}}) \\ \vdots \\ \sum_{k=1}^{m_h} \mathbf{h}(\mathbf{x_{hk}}) \end{bmatrix} = \begin{bmatrix} \sum_{k=1}^{m_1} h_1(\mathbf{x_{1k}}) & \cdots & \sum_{k=1}^{m_1} h_L(\mathbf{x_{1k}}) \\ \vdots & \vdots & \vdots \\ \sum_{k=1}^{m_h} h_1(\mathbf{x_{hk}}) & \vdots & \sum_{k=1}^{m_h} h_L(\mathbf{x_{hk}}) \end{bmatrix}$$

and

$$\mathbf{P} = \begin{bmatrix} m_1 * \mathbf{p_1^T} \\ \vdots \\ m_h * \mathbf{p_h^T} \end{bmatrix} = \begin{bmatrix} m_1 * p_{11} & \cdots & m_1 * p_{1c} \\ \vdots & \vdots & \vdots \\ m_h * p_{h1} & \vdots & m_h * p_{hc} \end{bmatrix}$$

Meanwhile, the final solution $\boldsymbol{\beta}$ is the same with the original form in ELM with dimension $L \times c$. Again, the optimal solution to (6.68) is given by

$$\boldsymbol{\beta}^* = \mathbf{H_p^\dagger P} \tag{6.69}$$

where $\mathbf{H_p^\dagger}$ is the Moore-Penrose generalized inverse of matrix $\mathbf{H_p}$.

In order to obtain a better generalization performance of ELM, we also follow the method from [22] to study the regularized ELM. In detail, the final objective function of ELM is formulated as follows:

$$\min_{\beta \in R^{L\times c}} \frac{1}{2}\|\boldsymbol{\beta}\|^2 + \frac{C}{2}\sum_{i=1}^{N}\|\mathbf{e_i}\|^2$$
$$s.t.\ \mathbf{h}(\mathbf{x_i})\boldsymbol{\beta} = \mathbf{t_i^T} - \mathbf{e_i^T}, i = 1, \ldots, N, \tag{6.70}$$

in which the first term of the objective function is a regularization term and C is a parameter to make a trade-off between the first and second term.

We equivalently reformulate the problem (6.70) as follows by substituting the constraints to its objective function:

$$\min_{\beta \in R^{L\times c}} L_{ELM} = \frac{1}{2}\|\boldsymbol{\beta}\|^2 + \frac{C}{2}\|\mathbf{T} - \mathbf{H}\boldsymbol{\beta}\|^2 \tag{6.71}$$

Note that the second term of (6.71) can be replaced by $\frac{C}{2}\|\mathbf{P} - \mathbf{H_p}\boldsymbol{\beta}\|^2$, which is the matrix form in bag level. In other words, the final unconstrained optimization problem can be written as:

$$\min_{\beta \in R^{L\times c}} L_{ELM} = \frac{1}{2}\|\boldsymbol{\beta}\|^2 + \frac{C}{2}\|\mathbf{P} - \mathbf{H_p}\boldsymbol{\beta}\|^2 \tag{6.72}$$

In practice, the final objection is widely known as the ridge regression or regularized least squares.

C. How to Solve the LLP-ELM

We follow the strategy from [22] to solve (6.72), and the final purpose is to minimize the training error as well as the norm of the output weights. Obviously, the final objective function is a convex problem, which is always solved by way of gradient. More specifically, by setting the gradient of (6.72) to zero with respect to $\boldsymbol{\beta}$, we can obtain the following expression:

$$\boldsymbol{\beta} - C\mathbf{H_p^T}(\mathbf{P} - \mathbf{H_p}\boldsymbol{\beta}) = \mathbf{0}. \tag{6.73}$$

This yields

$$(\frac{\mathbf{I}}{\mathbf{C}} + \mathbf{H_p^T}\mathbf{H_p})\boldsymbol{\beta} = \mathbf{H_p^T}\mathbf{P}, \tag{6.74}$$

where **I** is an identity matrix with dimension L.

The above equation is very intuitive, and we can obtain the final optimization result by inverting a L×L matrix directly. However, it is less efficient to directly invert a L×L matrix when the number of bag is less than the number of hidden neurons(h < L). Therefore, there are two methods which are shown in **Remark 1** and **Remark 2**. In summary, in the case where the number of bags are plentiful than hidden neurons, we use **Remark 1** to compute the output weights, otherwise we use **Remark 2**.

Remark 1 The solution for formula (6.73) when h > L.

- $\mathbf{H_p}$ has more rows than columns, which means the number of bag is larger than the number of hidden neurons.
- By inverting a L×L matrix directly and multiplying both sides by $(\mathbf{H_p^T H_p} + \frac{\mathbf{I}}{\mathbf{C}})^{-1}$, we can obtain the following expression

$$\boldsymbol{\beta} = (\mathbf{H_p^T H_p} + \frac{\mathbf{I}}{\mathbf{C}})^{-1}\mathbf{H_p^T P}, \tag{6.75}$$

which is the optimal solution of (6.73).

Remark 2 The solution for formula (6.73) when h < L.

- Notice that $\mathbf{H_p}$ is full row rank and $\mathbf{H_p H_p^T}$ is invertible when h < L.
- Restrict $\boldsymbol{\beta}$ to be a linear combination of the row in $\mathbf{H_p} : \boldsymbol{\beta} = \mathbf{H_p^T}\boldsymbol{\alpha}$
- Substitute $\boldsymbol{\beta} = \mathbf{H_p^T}\boldsymbol{\alpha}$ into (6.73), and multiply by $(\mathbf{H_p H_p^T})^{-1}\mathbf{H_p}$.
- By the above step, we can obtain the following equation:

$$\boldsymbol{\alpha} - C(\mathbf{P} - \mathbf{H_p H_p^T}\boldsymbol{\alpha}) = \mathbf{0}. \tag{6.76}$$

- As a result, the final optimal solution of (6.73) is in form of

$$\boldsymbol{\beta} = \mathbf{H_p^T}\boldsymbol{\alpha} = \mathbf{H_p^T}(\mathbf{H_p H_p^T} + \frac{\mathbf{I}}{\mathbf{C}})^{-1}\mathbf{P} = \mathbf{0}. \tag{6.77}$$

The solution process of LLP-ELM model can be concluded to the following two steps:

- Compute training data target proportion matrix **P** and the hidden layer output matrix $\mathbf{H_p}$.
- Obtain the final optional solution of $\boldsymbol{\beta}$ according to **Remark 1** or **Remark 2**. The details of the process are shown in **Algorithm 6.15**.

D. Computational Complexity

From the **Remark 1** and **Remark 2**, we can observe that the main time cost of our method is to calculate the matrix inversion. Furthermore, the dimension of matrix is

Algorithm 6.15 LLP-ELM

```
1: Input: Training datasets in bags{$B_n$}; The corresponding proportion $p_n$ of $B_n$; Activation
   function g(x) and the number of hidden nodes N.
2: Output: Classification model f(x,$\boldsymbol{\beta}$)
3: Begin
4: • Randomly initialize the value $\mathbf{w_j}$ and $b_j$ for the $j$th node, $j = 1, \ldots, L$.
5: • Compute the training data target proportion matrix $\mathbf{P}$ by the proportion information of each
   bag.
6: • Compute the hidden layer output matrix in the bag level $\mathbf{H_p}$.
7: • Obtain the weight vector according to Remark 1 or Remark 2.
8: End
```

minimum of the number of bags h and the hidden neurons L, which is determined by us. As we all know, the complexity of matrix inversion is proportional to the O^3, where O is the dimension of matrix, and is equal to Min(L,h) in this paper.

6.2.2 Learning from Label Proportions with Generative Adversarial Networks

6.2.2.1 Preliminaries

A. The Multi-Class LLP

Before further discussion, we formally describe multi-class LLP. For simplicity, we assume that all the bags are disjoint and let $\mathcal{B}_i = \{\mathbf{x}_i^1, \mathbf{x}_i^2, \cdots, \mathbf{x}_i^{N_i}\}, i = 1, 2, \cdots, n$ denote bags in training set. Then, training data is $\mathcal{D} = \mathcal{B}_1 \cup \mathcal{B}_2 \cup \cdots \cup \mathcal{B}_n, \mathcal{B}_i \cap \mathcal{B}_j = \emptyset, \forall i \neq j$, where the total number of bags is n.

In addition, $\mathbf{p}_i$ is a K-element vector where the kth element p_i^k is instance proportion in $\mathcal{B}_i$ belonging to the kth class with the constraint $\sum_{k=1}^{K} p_i^k = 1$ and K represents the total number of classes, i.e.,

$$p_i^k := \frac{|\{j \in [1 : N_i] | \mathbf{x}_i^j \in \mathcal{B}_i, y_i^{j*} = k\}|}{|\mathcal{B}_i|}. \tag{6.78}$$

Here, $[1 : N_i] = \{1, 2, \cdots, N_i\}$ and y_i^{j*} is the unaccessible ground-truth instance-level label of $\mathbf{x}_i^j$. In this way, we can denote the available training data as $\mathcal{L} = \{(\mathcal{B}_i, \mathbf{p}_i)\}_{i=1}^n$. The goal of LLP is to learn an instance-level classifier based on this kind of dataset.

B. Deep Discriminant Approach for LLP

In terms of deep learning, DLLP firstly leveraged CNNs to solve multi-class LLP problem [1]. Since CNNs can give a probabilistic interpretation for classification, it is straightforward to adapt cross-entropy loss into a bag-level version by averaging the probability outputs in every bag as the proportion estimation. To this end, inspired by [71], DLLP reshaped standard cross-entropy loss by substituting instance-level label with label proportion, in order to meet the proportion consistency.

In detail, suppose that $\tilde{\mathbf{p}}_i^j = p_\theta(\mathbf{y}|\mathbf{x}_i^j)$ is the vector-valued CNNs output for $\mathbf{x}_i^j$, where θ is the network parameter. Let $\oplus$ be element summation operator, then the bag-level label proportion in the ith bag is obtain by incorporating the element-wise posterior probability:

$$\overline{\mathbf{p}}_i = \frac{1}{N_i} \bigoplus_{j=1}^{N_i} \tilde{\mathbf{p}}_i^j = \frac{1}{N_i} \bigoplus_{j=1}^{N_i} p_\theta(\mathbf{y}|\mathbf{x}_i^j), \tag{6.79}$$

In order to smooth *max* function [5], $\tilde{\mathbf{p}}_i^j$ is in a vector-type softmax manner to produce the distribution for class probabilities. Taking *log* as element-wise logarithmic operator, objective of DLLP can be intuitively formulated using cross-entropy loss $L_{prop} = -\sum_{i=1}^n \mathbf{p}_i^\top log(\overline{\mathbf{p}}_i)$. It penalizes the difference between prior and posterior probabilities in bag-level, and commonly exists in GAN-based SSL [61].

C. Entropy Regularization for DLLP

Following the entropy regularization strategy [18], we can introduce an extra loss E_{in} with a trade-off hyperparameter λ to constrain instance-level output distribution in a low entropy accordingly:

$$L = L_{prop} + \lambda E_{in} = -\sum_{i=1}^{n} \mathbf{p}_i^\top log(\overline{\mathbf{p}}_i) - \lambda \sum_{i=1}^{n} \sum_{j=1}^{N_i} (\tilde{\mathbf{p}}_i^j)^\top log(\tilde{\mathbf{p}}_i^j). \tag{6.80}$$

This extension is similar to a KL divergence between two distributions. It takes advantage of DNN's output distribution to cater to the label proportions requirement, as well as minimizing output entropy as a regularization term to guarantee strong true-fake belief. This is believed to be linked with an inherent MAP estimation with certain prior distribution in network parameters.

6.2.2.2 Adversarial Learning for LLP

In this section, we propose LLP-GAN, which devotes GANs to harnessing LLP problem.

A. The Objective Function of Discriminator

We illustrate the LLP-GAN framework in Fig. 6.6. The generator is employed to generate images with input noise, which is labeled as fake. On the other hand, the discriminator yields class confidence maps for each class (including the fake one) by taking both fake and real data as the inputs. In particular, our discriminator is not only to identify whether it is a sample from the real data or not, but also to elaborately distinguish each real input's label assignment as a K classes classifier. This idea is fairly intuitive, and we conclude its loss as the L_{unsup} term.

Next, the main issue becomes how to exploit the proportional information to guide this unsupervised learning correctly. To this end, we replace the supervised information in semi-supervised GANs with label proportions, resulting in L_{sup}, same as L_{prop} in (6.80).

Definition 6.36 Suppose that $\mathcal{P}$ is a partition to divide the data space into n disjoint sections. Let $p_d^i(\mathbf{x}), i = 1, 2, \cdots, n$ be marginal distributions with respect to elements in $\mathcal{P}$ respectively. Accordingly, n bags in LLP training data spring from sampling upon $p_d^i(\mathbf{x}), i = 1, 2, \cdots, n$. In the meantime, let $p(\mathbf{x}, y)$ be the unknown holistic joint distribution.

We normalize the first K classes in $P_D(\cdot|\mathbf{x})$ into the instance-level posterior probability $\tilde{p}_D(\cdot|\mathbf{x})$ and compute $\overline{\mathbf{p}}$ based on (6.79). Then, the *ideal* optimization problem for the discriminator of LLP-GAN is:

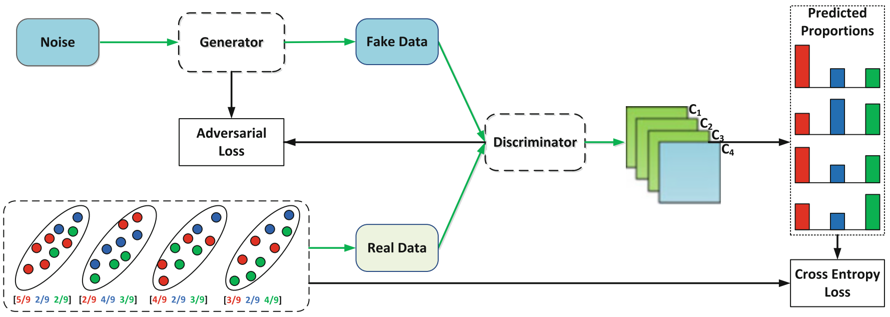


Fig. 6.6 An illustration of our LLP-GAN framework

$$\max_D \; V(G,D) = L_{unsup} + L_{sup} = L_{real} + L_{fake} - \lambda CE_{\mathcal{L}}(\mathbf{p}, \overline{\mathbf{p}})$$
$$= \sum_{i=1}^{n} E_{\mathbf{x}\sim p_d^i}\Big[log P_D(y \le K|\mathbf{x})\Big] + E_{\mathbf{x}\sim p_g}\Big[log P_D(K+1|\mathbf{x})\Big] + \lambda \sum_{i=1}^{n} \mathbf{p}_i^{\mathsf{T}} log(\overline{\mathbf{p}}_i). \tag{6.81}$$

Here, $p_g(\mathbf{x})$ represents the distribution of the synthesized data.

The normalized instance-level posterior probability $\tilde{p}_D(\cdot|\mathbf{x})$ is:

$$\tilde{p}_D(k|\mathbf{x}) = \frac{P_D(k|\mathbf{x})}{1 - P_D(K+1|\mathbf{x})}, k = 1, 2, \cdots, K. \tag{6.82}$$

Note that weight λ in (6.81) is added to balance between supervised and unsupervised terms, which is a slight revision of SSL with GANs [13, 54]. Intuitively, we reckon the proportional information is too weak to fulfill supervised learning pursuit. As a result, a relatively small weight should be preferable in the experiments. However, we fix $\lambda = 1$ in the following theoretical analysis on discriminator.

Aside from identifying the first two terms in (6.81) as that in semi-supervised GANs, the cross-entropy term harnesses the label proportions consistency. In order to justify the non-triviality of this loss, we first look at its lower bound. More important, it is easier to perform the gradient method on the lower bound, because it swaps the order of *log* and the summation operation. For brevity, the analysis will be done in a non-parametric setting, i.e. we assume that both D and G have infinite capacity.

Remark (The Lower Bound Approximation) Let $p_i(k) = p_i^k = \int p_i(y=k|\mathbf{x}) p_d^i(\mathbf{x})\mathrm{d}\mathbf{x}$ be the class k proportion in the ith bag. By applying Monte-Carlo sampling, we have:

$$-CE_{\mathcal{L}}(\mathbf{p}, \overline{\mathbf{p}}) = \sum_{i=1}^{n}\sum_{k=1}^{K} p_i(k) log\Big[\frac{1}{N_i}\sum_{j=1}^{N_i} \tilde{p}_D(k|\mathbf{x}_i^j)\Big]$$
$$\backsimeq \sum_{i=1}^{n}\sum_{k=1}^{K} p_i(k) log\Big[\int p_d^i(\mathbf{x})\tilde{p}_D(k|\mathbf{x})\mathrm{d}\mathbf{x}\Big] \geqslant \sum_{i=1}^{n}\sum_{k=1}^{K} p_i(k) E_{\mathbf{x}\sim p_d^i}\Big[log\, \tilde{p}_D(k|\mathbf{x})\Big]. \tag{6.83}$$

Similar to EM mechanism for mixture models, by approximating $-CE_{\mathcal{L}}(\mathbf{p}, \overline{\mathbf{p}})$ with its lower bound, we can perform gradient ascend independently on every sample. Hence, SGD can be applied.

Property 6.7 The maximization on the lower bound in (6.83) induces an optimal discriminator D^* with a posterior distribution $\tilde{p}_{D^*}(y|\mathbf{x})$, which is consistent with the prior distribution $p_i(y)$ in each bag.

Proof Taking the aggregation with respect to one bag, for example, the ith bag, we have:

$$\begin{aligned} E_{\mathbf{x}\sim p_d^i}[logp(\mathbf{x})] =& E_{\mathbf{x}\sim p_d^i} log\Big[\frac{p(\mathbf{x},y)}{\tilde{p}_D(y|\mathbf{x})}\frac{\tilde{p}_D(y|\mathbf{x})}{p(y|\mathbf{x})}\Big] \\ =& E_{\mathbf{x}\sim p_d^i}\int p_i(y) log\Big[\frac{p_i(y)p(\mathbf{x}|y)}{\tilde{p}_D(y|\mathbf{x})}\frac{\tilde{p}_D(y|\mathbf{x})}{p(y|\mathbf{x})}\Big]\mathrm{d}y \\ =& E_{\mathbf{x}\sim p_d^i}\int\Big[p(y_i) log\tilde{p}_D(y|\mathbf{x}) + log\frac{p(\mathbf{x}|y)}{p(y|\mathbf{x})}\Big]\mathrm{d}y \\ &+ E_{\mathbf{x}\sim p_d^i} KL(p_i(y)\|\tilde{p}_D(y|\mathbf{x})) \\ \geqslant& \sum_{k=1}^{K} p_i(k) E_{\mathbf{x}\sim p_d^i}\Big[log\tilde{p}_D(k|\mathbf{x})\Big] + \sum_{k=1}^{K} p_i(k) E_{\mathbf{x}\sim p_d^i}\Big[log\frac{p(\mathbf{x}|k)}{p(k|\mathbf{x})}\Big] \end{aligned} \tag{6.84}$$

Note that the last term in (6.84) is free of the discriminator, and the aggregation can be independently performed within every bag due to the disjoint assumption. Then, maximizing the lower bound in (6.83) is equivalent to minimizing the expectation of KL-divergence between $p_i(y)$ and $\tilde{p}_D(y|\mathbf{x})$. Because of the infinite capacity assumption on discriminator and the non-negativity of KL-divergence, we have:

$$D^* = \arg\min_D E_{\mathbf{x}\sim p_d^i} KL(p_i(y)\|\tilde{p}_D(y|\mathbf{x})) \Leftrightarrow \tilde{p}_{D^*}(y|\mathbf{x}) \overset{a.e.}{=} p_i(y), \mathbf{x}\sim p_d^i(\mathbf{x}). \tag{6.85}$$

That concludes the proof. □

Property 6.7 tells us that if there is only one bag, then $\tilde{p}_{D^*}(y|\mathbf{x}) \overset{a.e.}{=} p(y)$. However, there is normally more than one bag in LLP, the final classifier will somehow be a trade-off among all the prior proportions $p_i(y), i = 1, 2, \cdots, n$. Next, we will show how the adversarial learning on the discriminator helps to determine the formulation of this trade-off into a weighted aggregation.

B. Global Optimality

As shown in (6.83), in order to facilitate the gradient computation, we substitute cross entropy in (6.81) by its lower bound and denote this approximate objective function for discriminator by $\widetilde{V}(G, D)$.

Theorem 6.6 *For fixed G, the optimal discriminator D^* for $\widetilde{V}(G, D)$ satisfies:*

$$P_{D^*}(y=k|\mathbf{x}) = \frac{\sum_{i=1}^{n} p_i(k)p_d^i(\mathbf{x})}{\sum_{i=1}^{n} p_d^i(\mathbf{x}) + p_g(\mathbf{x})}, k = 1, 2, \cdots, K. \tag{6.86}$$

Proof According to (6.81) and (6.83) and given any generator G, we have:

$$\widetilde{V}(G,D)=\sum_{i=1}^{n}E_{\mathbf{x}\sim p_d^i}\Big[log(1-P_D(K+1|\mathbf{x}))\Big]+E_{\mathbf{x}\sim p_g}\Big[log P_D(K+1|\mathbf{x})\Big]+$$
$$\sum_{i=1}^{n}\sum_{k=1}^{K}p_i(k)E_{\mathbf{x}\sim p_d^i}\Big[log\,\tilde{p}_D(k|\mathbf{x})\Big]=\int\Big\{\sum_{i=1}^{n}p_d^i(\mathbf{x})\Big[log[\sum_{k=1}^{K}P_D(k|\mathbf{x})]+ \tag{6.87}$$
$$\sum_{k=1}^{K}p_i(k)log\frac{P_D(k|\mathbf{x})}{1-P_D(K+1|\mathbf{x})}\Big]+p_g(\mathbf{x})log\Big[1-\sum_{k=1}^{K}P_D(k|\mathbf{x})\Big]\Big\}\mathbf{dx}$$

By taking the derivative of the integrand, we find the maximum in [0, 1] as that in (6.86). □

Remark (Beyond the Incontinuity of p_g) According to [2], the problematic scenario is that the generator is a mapping from a low dimensional space to a high dimensional one, which results in the density of $p_g(\mathbf{x})$ infeasible. However, based on the definition of $\tilde{p}_D(y|\mathbf{x})$ in (6.82), we have:

$$\tilde{p}_{D^*}(y|\mathbf{x})=\frac{\sum_{i=1}^{n}p_i(y)p_d^i(\mathbf{x})}{\sum_{i=1}^{n}p_d^i(\mathbf{x})}=\sum_{i=1}^{n}w_i(\mathbf{x})p_i(y). \tag{6.88}$$

Hence, our final classifier does not depend on $p_g(\mathbf{x})$, and (6.88) explicitly expresses the weights of the aggregation.

Remark (Relationship to One-Side Label Smoothing) Notice that the optimal discriminator D^* is also related to the one-sided label smoothing mentioned in [54], which was inspirited by [64] and shown to reduce the vulnerability of neural networks to adversarial examples [73].

In our model, we only smooth labels of real data (multi-class classifier) in the discriminator by setting the targets as the holistic proportions (the prior) $p_i(y)$ in corresponding bags.

C. The Objective Function of Generator

Normally, for the generator, we should solve the following optimization problem with respect to p_g.

$$\min_G \widetilde{V}(G,D^*)=\min_G E_{\mathbf{x}\sim p_g}log P_{D^*}(K+1|\mathbf{x}). \tag{6.89}$$

If denoting $C(G)=\max_D \widetilde{V}(G,D)=\widetilde{V}(G,D^*)$, because $\widetilde{V}(G,D)$ is convex in p_g and the supremum of a set of convex function is still convex, we have the following conclusion.

Theorem 6.7 *The global minimum of $C(G)$ is achieved if and only if $p_g = \frac{1}{n}\sum_{i=1}^{n} p_d^i$.*

Proof Denote $p_d = \sum_{i=1}^{n} p_d^i$. Hence, according to Theorem 6.6, we can reformulate $C(G)$ as:

$$
\begin{aligned}
C(G) = \sum_{i=1}^{n} E_{\mathbf{x}\sim p_d^i}\Big[log\frac{p_d(\mathbf{x})}{p_d(\mathbf{x})+p_g(\mathbf{x})}\Big] + E_{\mathbf{x}\sim p_g}\Big[log\frac{p_g(\mathbf{x})}{p_d(\mathbf{x})+p_g(\mathbf{x})}\Big] + \\
\sum_{i=1}^{n}\sum_{k=1}^{K} p_i(k)E_{\mathbf{x}\sim p_d^i}\Big[log\,\tilde{p}_D(k|\mathbf{x})\Big] = 2\cdot JSD(p_d\|p_g) - 2log(2) - \\
\sum_{i=1}^{n} E_{\mathbf{x}\sim p_d^i}\Big[CE(p_i(y), \tilde{p}_{D^*}(y|\mathbf{x}))\Big],
\end{aligned}
\tag{6.90}
$$

where $JSD(\cdot\|\cdot)$ and $CE(\cdot,\cdot)$ are the Jensen-Shannon divergence and cross entropy between two distributions, respectively. However, note that p_d is a summation of n independent distributions, so $\frac{1}{n}p_d$ is a well-defined probabilistic density. Then, we have:

$$
\begin{aligned}
C(G^*) = \min_G C(G) = nlog(n) - (n+1)log(n+1) - \sum_{i=1}^{n} E_{\mathbf{x}\sim p_d^i}\Big[CE(p_i(y), \tilde{p}_{D^*}(y|\mathbf{x}))\Big] \\
\Longleftrightarrow p_{g^*} \overset{\text{a.e.}}{=} \frac{1}{n}p_d.
\end{aligned}
\tag{6.91}
$$

That concludes the proof. □

Remark When there is only one bag, the first two terms in (6.91) will degenerate as $nlog(n) - (n+1)log(n+1) = -2log2$, which adheres to results in original GANs. On the other hand, the third term manifests the uncertainty on instance label, due to the concealment in the form of proportion.

Remark According to the analysis above, ideally, we can obtain the Nash equilibrium between the discriminator and the generator, i.e. the solution pair (G^*, D^*) satisfies:

$$
\widetilde{V}(G^*, D^*) \geqslant \widetilde{V}(G^*, D), \forall D;\ \widetilde{V}(G^*, D^*) \leqslant \widetilde{V}(G, D^*), \forall G. \tag{6.92}
$$

However, as shown in [13], a well-trained generator would lead to the inefficiency of supervised information. In other words, the discriminator would possess the same generalization ability as merely training it on L_{prop}. Hence, we apply feature matching (FM) to the generator, and obtain its alternative objective by matching the expected value of the features (statistics) on an intermediate layer of the discriminator [54]: $L(G) = \|E_{\mathbf{x}\sim\frac{1}{n}p_d} f(\mathbf{x}) - E_{\mathbf{x}\sim p_g} f(\mathbf{x})\|_2^2$. In fact, FM is similar to the perceptual loss for style transfer in a concurrent work [26] and the goal of this improvement is to impede the "perfect" generator resulting in unstable training and discriminator with low generalization.

D. LLP-GAN Algorithm

So far, we have clarified the objective functions of both discriminator and generator in LLP-GAN. In particular, note that we execute Monte-Carlo sampling for the expectations. When accomplishing the training stage in GAN manner, the discriminator can be put into effect as the final classifier.

The strict proof for algorithm convergence is similar to that in [17]. Because $\max_D \widetilde{V}(G, D)$ is convex in G and the subdifferential of $\max_D \widetilde{V}(G, D)$ contains that of $\widetilde{V}(G, D^*)$ in every step, the exact line search method gradient descent converges [7]. We present the LLP-GAN algorithm as follows.

Algorithm 6.16 LLP-GAN training algorithm

```
1: Input: The training set $\mathcal{L} = \{(\mathcal{B}_i, \mathbf{p}_i)\}_{i=1}^n$; $L$: number of total iterations; $\lambda$: weight parameter.
2: Input: The parameters of the final discriminator $D$.
3: Set $m$ to the total number of training data points.
4: for i=1:L do
5:     Draw $m$ samples $\{\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \cdots, \mathbf{z}^{(m)}\}$ from a simple-to-sample noise prior $p(\mathbf{z})$ (e.g., $N(0, I)$).
6:     Compute $\{G(\mathbf{z}^{(1)}), G(\mathbf{z}^{(2)}), \cdots, G(\mathbf{z}^{(m)})\}$ as sampling from $p_g(\mathbf{x})$.
7:     Fix the generator $G$ and perform gradient ascent on parameters of $D$ in $\widetilde{V}(G, D)$ for one step.
8:     Fix the discriminator $D$ and perform gradient descent on parameters of $G$ in $L(G)$ for one step.
9: end for
10: Return The parameters of the discriminator $D$ in the last step.
```

6.2.3 *Learning from Label Proportions on High-Dimensional Data*

6.2.3.1 Background

In this subsection, the random forests which is used for our classification is presented.

Random forests are an ensemble learning method together with a bagging procedure for classification and other tasks, where each basic classifier is a decision tree and each tree depends on a collection of random variables. More specifically, during splitting of a randomized tree, each decision node randomly selects a set of features and then picks the best among them according to some quality measurement (*e.g.*, information gain or Gini index) [53]. Furthermore, as each tree in the forest is built and tested independently from other trees, the overall training and testing procedures can be performed in parallel [31].

We denote the mth tree of random forests as $f(x, \theta_m)$, where θ_m is a random vector representing the various stochastic elements of the tree. Meanwhile, let $p_m(k|x)$ represent the estimated density of class labels for the mth tree and M be total number of the trees in the forests. In practice, the final prediction results of random forests are given by probability towards different classes. As a result, the estimated probability for predicting class k in random forests can be defined as:

$$F_k(\mathbf{x}) = \frac{1}{M} \sum_{m=1}^{M} p_m(k|\mathbf{x}), k \in \gamma = \{1, 2, \ldots, K\}, \tag{6.93}$$

where K is the total number of classes. In particular, a decision can be made by simply taking the maximum over all individual probabilities of the trees for a class k with

$$C(\mathbf{x}) = \arg \max_{k \in \gamma} F_k(\mathbf{x}), \gamma = \{1, 2, \ldots, K\} \tag{6.94}$$

where the final result of $C(\mathbf{x})$ is the index of the corresponding class.

The classification margin measures the extent to which the average number of votes for the right class exceeds the average for any other class, which is introduced by Breiman [8], and is expressed as:

$$mg(\mathbf{x}, y) = F_y(\mathbf{x}) - \max_{k \neq y} F_k(\mathbf{x}). \tag{6.95}$$

Obviously, if the classification is correct, there should be $mg(\mathbf{x}, y) > 0$. In other words, the larger the margin is, the more confidence in the classification. The generalization error of random forests is in form of:

$$GE = E_{(X,Y)}(mg(\mathbf{x}, y) < 0), \tag{6.96}$$

where the expectation is measured over the entire distribution of (X,Y).

Random forests have shown its advantages in both classification [8] and clustering [45]. In particular, experiments have shown that high accuracy can be achieved by random forests when classifying high dimensional data [3]. Meanwhile, Caruana [9] presented an empirical evaluation on high dimensional data of different methods, and found that random forests perform consistently well across all dimensions compared with other methods. Additionally, it is easy for random forests to be parallelized, which makes them very easy for multi-core and GPU implementations. Sharp [56] have show that GPU can accelerate the random forests and have great advantage compared to CPU in processing speed, which is very useful for practical applications. Recently, random forests have been applied in video segmentation [49], object detection [15], image classification [6] and remote sensing [46] due to its advantages.

6.2.3.2 The LLP-RF Algorithm

In this subsection, we present a novel learning from label proportions algorithm called LLP-RF, which use random forests to solve high-dimensional LLP problem. In order to leverage random forests to LLP, the hidden class labels insides bags are defined as the optimization variables. Meanwhile, we formulate a robust loss function based on random forests and take the corresponding proportion information into LLP-RF by penalizing the difference between the ground-truth and estimated label proportion. A binary learning setting is considered in the following.

A. Learning Setting

Similar to the standard supervised learning, the problem is also described by a set of training data. But the training data of LLP is only provided in form of bags and the ground-truth labels of training data are not available. In this paper, we assume the bags are disjoint. Let $B_i, i = 1, \ldots, n$ denote the ith bag in the training set. As a result, the total training data can be expressed as:

$$\begin{gathered} D = B_1 \cup B_2 \cup \ldots . \cup B_n \\ B_i \cap B_j = \emptyset, \forall i \neq j, \end{gathered} \tag{6.97}$$

where the total number of training data is N. The ith bag consists of m_i instances and is in form of:

$$B_i = \{x_i^1, ..., x_i^{m_i}\}\{p_i\}, i \in \{1, 2, \ldots, n\}, \tag{6.98}$$

where the associated p_i indicates the label proportion of the ith bag. As a result, the jth instance in the ith bag can be expressed as x_i^j.

The ground-truth labels of instances are modeled as $\mathbf{y} = (y_1, \ldots, y_N)^T$, where y_i is the unknown label of x_i. Furthermore, we can define the proportion of ith bag as:

$$p_i = \frac{|\{k | k \in B_i, y_k^* = 1\}|}{|B_i|}, \forall k \in \{1, 2, \ldots, N\}, \tag{6.99}$$

in which $y_k^* \in \{1, -1\}$ is the unknown ground-truth label of x_k and $|B_i|$ denotes the bag size of ith bag. In practice, the above formulation is equivalent to the following:

$$p_i = \frac{\sum_{k \in B_i} y_k^*}{2|B_i|} + \frac{1}{2}, \forall k \in \{1, 2, \ldots, N\}. \tag{6.100}$$

B. The LLP-RF Framework

The above LLP learning setting is very intuitive and the final objective is to train a classifier in the instance level. To this end, inspired by [32], we formulate a robust loss function based on random forests and take the corresponding proportion information into LLP-RF by penalizing the difference between the ground-truth and estimated label proportion. Therefore, the final objective function of LLP-RF is formulated as follows:

$$\begin{aligned} \arg\min_{F(\cdot), y_i^j} \quad & C\sum_{i=1}^{n}\sum_{j=1}^{m_i} L[F_{y_i^j}(x_i^j)] + C_p \sum_{i=1}^{n} L_p[p_i(\mathbf{y}), p_i] \\ s.t. \quad & \forall_{i=1}^{n}, \forall_{j=1}^{m_i} \quad y_i^j \in \{1, -1\}, \end{aligned} \tag{6.101}$$

where the hidden class labels $\mathbf{y}$ are defined as the optimization variables and the task is to simultaneously optimize the labels $\mathbf{y}$ and the model $F()$.

Specifically, $L()$ is a loss function which is defined over the entire set of instances and $L_p()$ is a loss function used to penalize the difference between the ground-truth label proportion and the estimated label proportion based on $\mathbf{y}$. Different weights can be added for the loss of bag proportions by changing the value of C_p.

Note that $F_k(x)$ is the confidence of classifier for the kth class, which is got from random forests.

Furthermore, our proposed framework permits choosing different loss functions for $L()$. In our paper, different loss function including hinge loss, logistic loss and entropy are tuned to obtain better classification results. In this paper, we consider $L_p()$ as the absolute loss:

$$L_p[p_i(\mathbf{y}), p_i] = |p_i(\mathbf{y}) - p_i|, \tag{6.102}$$

where p_i is the true label proportion of ith bag and $p_i(\mathbf{y})$ is the estimated label proportion of ith bag.

The above LLP-RF framework is fairly straightforward and intuitive. However, it leads to a non-convex integer programming problem because it needs to simultaneously optimize the labels y_i^j and trains a random forest. In practice, the problem is often NP-hard. Therefore, one key issue is how to solve the optimization problem efficiently. In this paper, a simple but efficient alternating optimization strategy based on annealing is employed to minimize the overall learning objective.

C. How to Solve the LLP-RF

The strategy to solve (6.101) is similar to the rule from [80]. There are two variables F and $\mathbf{y}$ in the optimization formula, where the unknown instance labels $\mathbf{y}$ can be seen as a bridge between supervised learning loss and label proportion loss.

Therefore, we solve the problem by alternating optimizing the two variables F and $\mathbf{y}$.

- We fix the $\mathbf{y}$. The optimization problem becomes a native random forests problem, which can be expressed as below:

$$\arg\min_{F(\cdot)} C \sum_{i=1}^{n} \sum_{j=1}^{m_i} L[F(x_i^j)]. \tag{6.103}$$

- Then, F is fixed. The problem can be transformed to the following:

$$\begin{aligned} &\arg\min_{y_i^j} \quad C \sum_{i=1}^{n} \sum_{j=1}^{m_i} L[F_{y_i^j}(x_i^j)] + C_p \sum_{i=1}^{n} L_p[p_i(\mathbf{y}), p_i] \\ &s.t. \;\; \forall_{i=1}^{n}, \forall_{j=1}^{m_i} \quad y_i^j \in \{1, -1\}. \end{aligned} \tag{6.104}$$

The first term of the objective is defined over the entire instances. However, the proportion information p_i of the second term is provided in the bag level. In order to use the proportion information efficiently, the above formula can be written to the following:

$$\begin{aligned} &\arg\min_{y_i^j} \quad \sum_{i=1}^{n} \left\{ C \sum_{j=1}^{m_i} L[F_{y_i^j}(x_i^j)] + C_p L_p[p_i(\mathbf{y}), p_i] \right\} \\ &s.t. \;\; \forall_{i=1}^{n}, \forall_{j=1}^{m_i} \quad y_i^j \in \{1, -1\}. \end{aligned} \tag{6.105}$$

As the bags are disjoint to each other, the contribution of each bag to the objective is independent. As a result, the objective can be optimized on each bag separately and the final result is equivalent to the summation of every bag. In particular, solving $\{y_i^j \,|\, j \in B_i\}$ yields the following optimization problem:

$$\begin{aligned} &\arg\min_{\{y_i^j | j \in B_i\}} \quad C \sum_{j \in B_i} \ell[F_{y_i^j}(x_i^j)] + C_p L_p[p_i(\mathbf{y}), p_i] \\ &s.t. \;\; \forall j \in B_i, \quad y_i^j \in \{1, -1\}. \end{aligned} \tag{6.106}$$

Obviously, the original optimization problem has changed to solve the formula (6.106), whose solution can be found by the following optimization strategy.

Remark The steps for solving formula (6.106).

- Compute all the possible values of the second term in formula (6.106), where there are total $|B_i| + 1$ values. In practice, the kth value can be expressed as:

$$F_2(k) = |\frac{k-1}{|B_i|} - p_i|, k \in \{1, 2, \dots, |B_i|, |B_i| + 1\}. \tag{6.107}$$

- Obtain all the values of first term $F_1(k)$ corresponding to the second term $F_2(k)$.
- Pick the smallest objective value from

$$C * F_1(k) + C_p * F_2(k), k \in \{1, 2, \dots, |B_i|, |B_i| + 1\}, \tag{6.108}$$

yielding the optimal solution of (6.106).

The above strategy is fairly intuitive and straightforward. The main focus is how to obtain the value of first term corresponding to the second term. In practice, there are total $|B_i| + 1$ values about the second term. For a fixed value of second term, steps can be taken as Proposition 6.4.

Proposition 6.4 *For a fixed $p_i(\mathbf{y}) = \theta$, we can find the solution of (6.106) by the iterative steps as below.*

- *Initialize $y_i^j = -1, \forall j \in \{1, 2, \dots, |B_i|\}$, where $|B_i|$ is the number of instances in ith bag.*
- *Compute the value of $\ell[F_{-1}(x_i^j)]), j \in \{1, 2, \dots, |B_i|\}$.*
- *Flip the sign of $y_i^j = 1, \forall j \in \{1, 2, \dots, |B_i|\}$.*
- *Compute the value of $\ell[F_1(x_i^j)]), j \in \{1, 2, \dots, |B_i|\}$.*
- *Let $\delta_i^j = C(\ell[F_1(x_i^j)] - \ell[F_{-1}(x_i^j)]), j \in \{1, 2, \dots, |B_i|\}$ denote the reduction of the first term in (6.106) through flipping the sign of y_i^j.*
- *Sort $\delta_i^j, \forall j \in \{1, 2, \dots, |B_i|\}$ in descending way. Then flip the signs of y_i^j of the top-R ($R = \theta|B_k|$) which have the highest reduction. For each bag, we only need to sort the $\delta_i^j, \forall j \in \{1, 2, \dots, |B_i|\}$ once.*

Obviously, the minimum value of each bag and the corresponding $\mathbf{y}$ can be obtained using the above steps. In detail, the solution process of the LLP-RF model can be concluded to the following two alternative steps: solve random forests optimization problems and renovate the labels of $\mathbf{y}$ until the objective function value is no longer changing or the reduction of objective is smaller than a threshold. The details of the process are shown in **Algorithm** 6.17.

Furthermore, in order to avoid the local solutions, similar to T-SVM [10] and SVM [80], the novelly proposed LLP-RF algorithm also takes an additional annealing loop to gradually increase C. The annealing can be seen as a step to avoid the local optimal solution. In detail, the annealing loop is achieved by the following equation $C^* = \min\{(1 + \triangle)C^*, C\}$, where $\triangle$ is a step to control the increase of C. Throughout this work, we set $\triangle = 0.5$.

In practice, the different values of initializing **y** can lead to different results. In order to reduce the randomness, we should repeat the process several times and pick the smallest objective value as the final result.

Algorithm 6.17 LLP-RF

```
1: Require: Bags$\{B_i\}$;
2: The corresponding proportion $p_i$ of $B_i$;
3: Randomly initialize $y_i^j \in \{1, -1\}, \forall_{i=1}^n, \forall_{j=1}^{m_i}$;
4: $C^* = 10^{-5}C$.
5: while $C^* < C$ do
6:     $C^* = \min\{(1+\triangle)C^*, C\}$.
7:     repeat
8:         Fix y to solve F(Train the Random Forests: $trainRF(y_i^j)$).
9:         Fix F to solve y(using the strategy discussed in the above Remark).
10:        Update $y_i^j, \forall_{i=1}^n, \forall_{j=1}^{m_i}$.
11:    until the decrease of the objective is smaller than a threshold or reach the setting iteration.
12: end while
```

6.2.4 *Learning from Label Proportions with Pinball Loss*

6.2.4.1 Preliminary

In this subsection, we introduce the basic formulation of learning from label proportions and give corresponding symbol description.

In learning from label proportions, although the proportion of each bag is given, the label of each instance is unknown. Suppose we are given a sample set $\{x_i, y_i^*\}_{i=1}^N$, where $x \in \mathcal{R}^n$ and $y_i^* \in \{1, 1\}$ denotes the unknown ground truth label of x_i. The sample set is grouped into K bags. In this subsection, we assume that the bags are disjoint.

The ground truth label proportion of the k-th bag S_k can be defined as

$$P_k := \frac{|\{i|i \in S_k, y_i^* = 1\}|}{|S_k|}.$$

The goal is to find a decision function $f(x) = sign(w^T\phi(x) + b)$ such that the label y for any instance x can be predicted, where $\phi(\cdot)$ is a map of the input data.

Assume the instance labels are explicitly modeled as $\{y_i\}_{i=1}^N$, where $y_i \in \{1, 1\}$. The modeled label proportion of the k-th bag can be defined as

$$P_k = \frac{|\{i|i \in S_k, y_i = 1\}|}{|S_k|}.$$

The Learning from label proportions model can be formulated as below:

$$\begin{aligned} &\min_{y,w,b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} L_\tau + C_2 \sum_{k=1}^{K} |p_k(y) - P_k|, \\ &\text{s.t. } y_i \in \{-1, 1\}, \end{aligned} \tag{6.109}$$

in which $L_\tau(\cdot)$ is the supervised loss function. Notice the instance labels y is also a variable, which can be seen as a bridge between empirical loss and label proportion loss.

Here, we first discuss the noise generated in the framework of learning from label proportions, and introduce pinball loss to address this issue. Next, we give the learning from label proportions model with pinball loss. Also, the dual problem is given. Then, an alternating optimization method is applied to solve the proposed model. Finally, the complexity of our method is discussed.

6.2.4.2 Noise and Pinball Loss

Unlike traditional hinge loss, pinball loss pushes the surfaces that define the margin to quantile positions by penalizing also the correctly classified sampling points [24]. The distance between these two classes is easily affected by the noise on feature x. Also, improper initialization of label y causes noise as well. As a result, the classifier with hinge loss is sensitive to feature noise. The pinball loss is related to quantiles and has been well studied in regression (parametric methods [52] and nonparametric methods [12, 63]. And it is also used for binary classification recently [23].

The pinball loss is defined as follows:

$$L_\tau(u) = \begin{cases} u, & u \geq 0, \\ -\tau u, & u < 0. \end{cases} \tag{6.110}$$

Particularly, when $\tau = 0$, the pinball loss $L_\tau(u)$ reduces to the hinge loss. When a positive τ is used, minimizing the pinball loss results in the quantile value.

To intuitively show the properties of pinball loss, we are going to compare the classifiers based on the hinge loss and the pinball loss, respectively. Here, let's consider a two dimensional example: points are generated from two Gaussian distribution $N(\mu_1, \sigma)$ and $N(\mu_2, \sigma)$, where $\mu_1 = [0.5, -3]^T$, $\mu_2 = [0.5, 3]^T$ and $\sigma = [0.1, 0; 0, 2]$. As shown in Fig. 6.7, the solid lines indicate the classification hyperplane achieved by classifier based on the hinge loss and the dashed lines represent the hyperplane obtained by pinball loss. The data points are generated from the same distribution. However, the hinge loss classifier obtains the significantly different results while the pinball loss hyperplane achieve more stable results. It is mainly because that the hinge loss classifier measures the distance between two sets by the nearest points. But pinball loss takes the nearest τ (e.g. 35%) points

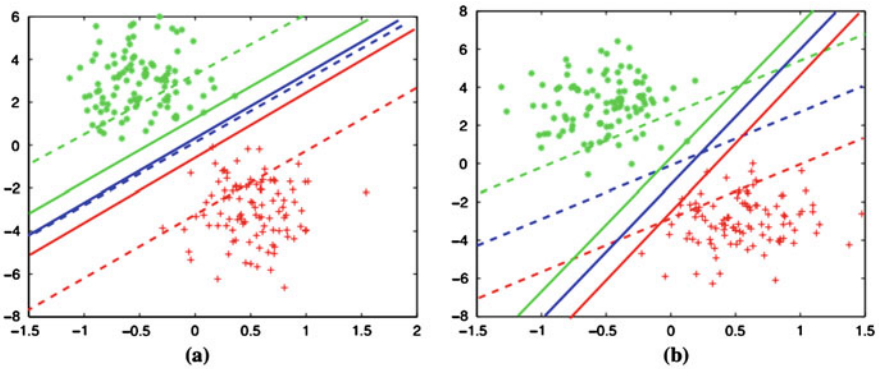


Fig. 6.7 Comparison between the classifiers based on hinge loss and pinball loss. As it is shown, the results of pinball loss classifier are more stable

to measure this distance, which makes its result less sensitive to noise around the boundary.

6.2.4.3 Learning from Label Proportions Model with Pinball Loss

With pinball loss, we can formulate the learning from label proportions model as below:

$$\min_{y,w,b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} L_\tau(1 - y_i(w^T\phi(x_i) + b)) + C_2\sum_{k=1}^{K}|p_k(y) - P_k|,$$
$$\text{s.t. } y_i \in \{-1, 1\}. \tag{6.111}$$

As the instance labels y is also a variable, one natural way for solving Eq. (6.111) is via alternating optimization.

Step 1 For a fixed y, the optimization of Eq. (6.109) w.r.t w and b becomes a classic SVM with pinball loss:

$$\min_{w,b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} L_\tau(1 - y_i(w^T\phi(x_i) + b)). \tag{6.112}$$

Step 2 When w and b are fixed, the problem becomes:

$$\min_{y} \sum_{i=1}^{N} L_\tau(1 - y_i(w^T\phi(x_i) + b)) + \frac{C_2}{C}\sum_{k=1}^{K} |p_k(y) - P_k|, \qquad \text{s.t. } y_i \in \{-1, 1\}. \tag{6.113}$$

By taking the strategy presented in [80], we show that the second step above can be solved efficiently. Since the influence of each bag on the objective is independent, we can optimize Eq. (6.113) on each bag separately. For a fixed $p_k(y) = \theta$, Eq. (6.113) can be optimally solved by the steps below.

- Initialize $y_i, i \in B_k$.
- Suppose the reduction of the first term in (6.113) is δ_i. Sort $\delta_i, i \in B_k$.
- Flip the signs of the top-R y_i which have the highest reduction δ_i, where $R = \theta|B_k|$.

  By conducting Step 1 and Step 2 alternately until the decrease of objective is smaller than a threshold (e.g. 10^{-4}), we can obtain the optimal solution.

6.2.4.4 Dual Problem

The problem in Eq. (6.112) can be transformed into:

$$\begin{aligned} &\min_{w,b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} \xi_i, \\ &\text{s.t. } y_i(w^T\phi(x_i) + b) \geq 1 - \xi_i, i = 1, 2, \cdots, N, \\ &\quad y_i(w^T\phi(x_i) + b) \leq 1 + \frac{1}{\tau}\xi_i, i = 1, 2, \cdots, N. \end{aligned} \tag{6.114}$$

According to the Karush-Kuhn-Tucker (KKT) sufficient and necessary optimality conditions, the dual problem of Eq. (6.114) is obtained as follows,

$$\begin{aligned} &\max_{\alpha,\beta} -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(\alpha_i - \beta_i)y_i\phi(x_i)^T\phi(x_j)y_j(\alpha_j - \beta_j) + \sum_{i=1}^{N}(\alpha_i - \beta_i), \\ &\text{s.t. } \sum_{i=1}^{N}(\alpha_i - \beta_i)y_i = 0, \\ &\quad \alpha_i + \frac{1}{\tau}\beta_i = C, \ i = 1, 2, \cdots, N, \\ &\quad \alpha_i \geq 0, \ i = 1, 2, \cdots, N, \\ &\quad \beta_i \geq 0, \ i = 1, 2, \cdots, N. \end{aligned} \tag{6.115}$$

Introduce the variables γ_i, α_i and β_i. Let $\gamma_i = \alpha_i - \beta_i$. The dual problem Eq. (6.115) has the same solution set w.r.t. α as that to the following convex quadratic programming problem:

$$\begin{aligned} &\min_{\gamma,\beta} \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\gamma_i y_i \phi(x_i)^T\phi(x_j) y_j \gamma_j - \sum_{i=1}^{N}\gamma_i, \\ &\text{s.t.} \sum_{i=1}^{N}\gamma_i y_i = 0, \\ &\quad -\tau C \le \gamma_i \le C,\ i = 1, 2, \cdots, N. \end{aligned} \tag{6.116}$$

Suppose $\gamma^* = (\gamma_1^*, \gamma_2^*, \ldots, \gamma_l^*)$ is the solution to problem Eq. (6.116). We can have

$$w^* = \sum_{i=1}^{N}\gamma_i^* y_i \phi(x_i), \text{ and}$$

$$b^* = y_j - \sum_{i=1}^{N} y_i \gamma_i^* \phi(x_i)^T\phi(x_j),$$

where $\forall j : -\tau C < \gamma_j^* < C$.

Then the obtained function can be represented as

$$f(x) = \sum_{i=1}^{N} y_i \gamma_i^* \phi(x_i)^T\phi(x_j) + b^*,$$

where $\forall j : -\tau C < \gamma_j^* < C$.

6.2.4.5 Overall Optimization Procedure

Based on the detailed explanation above, the overall optimization procedure is summarized in Algorithm 6.18.

By alternating between solving w^*, b^* and y, the objective is guaranteed to converge, for the reason that the objective function is lower bounded, and non-increasing. Empirically, the alternating optimization typically terminates fast within ten iterations.

In practice, the stopping criterion of the overall optimization procedure is that the objective function does not decrease any more (or if its decrease is smaller than a threshold).

Algorithm 6.18 Optimization procedure of learning from label proportions

```
1: Input: $\{x_i\}_{i=1}^N, \{P_k\}_{k=1}^K, C, C_2$ and $C^* = 10^{-5}C$.
2: Output: $w^*, b^*$ and $y$.
3: Randomly initialize $y_i \in \{-1, 1\}$.
4: while $C^* < C$ do
5:     $C^* = \min\{(1 + \triangle)C^*, C\}$.
6:     while not converged do
7:         % Fix y.
8:         $w^* = \sum_{i=1}^N \gamma_i y_i \phi(x_i)$.
9:         $b^* = y_i - \sum_{i=1}^N y_i \gamma_i^*(x_i \cdot x_j)$.
10:        % Fix $w^*$ and $b^*$.
11:        Solve y (Eq.(6.113) with $C \leftarrow C$).
12:    end while
13: end while
```

6.2.4.6 Complexity

Step 1 takes the complexity of SVM with pinball loss. As described in the paper, the bags are disjoint, the influences of the bags are independent. In Step 2, for each bag S_k, sorting takes $O(|S_k|log(|S_k|))$, which is same with [80]. Overall, the complexity is $O(\sum_{k=1}^K |S_k|log(|S_k|))$. We know that $\sum_{k=1}^K |S_k| = N$ and denote $J = \max_{k=1,2,\ldots,K} |S_k|$. The complexity is $O(Nlog(J))$ time.

6.3 Other Enlarged Learning Models

6.3.1 Classifying with Adaptive Hyper-Spheres: An Incremental Classifier Based on Competitive Learning

6.3.1.1 Basic Theory

A. Basic Theory of Supervised Competitive Learning

We partially borrow the topological structure of CPN to introduce our model. CPNs are a combination of competitive networks and Grossberg's outstar networks [19]. The topological structure of CPN has three layers: input layer, hidden layer, and output layer (Fig. 6.8).

Suppose there are N elements in the input layer, M neurons in the hidden layer, and L neurons in the output layer. Let vector $V_i = (v_{i1}, \ldots, v_{iN})^T$ denote the weights of neuron i in the hidden layer connecting to each of the elements of the input layer. Then $V = (V_1, \ldots, V_M)$ denotes weight matrix of the instars. If the training in stage 1 can be viewed as a clustering process, then neuron i is cluster c_i and V_i is the centroid of cluster c_i.

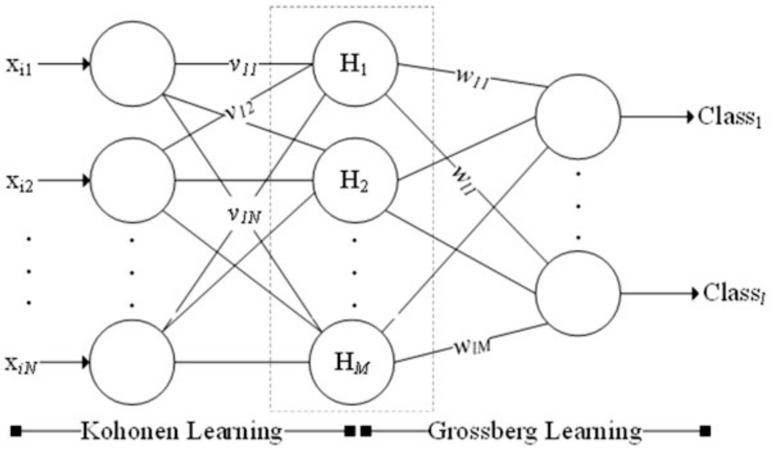


Fig. 6.8 Topological structure of CPN.

When an instance is coming, it will compute the proximity between the instance and each V_i in the weight matrix, i.e., the centroid of cluster c_i. Here, proximity can be measured by computing inner product $net_j = V_j^T x$, $(j = 1, 2, \ldots, m)$. It adopts a winner-takes-all strategy to determine which neuron's weights are to be adjusted. The winner is $net_{j^*} = \max\{net_j\}$. In other words, the winner is c_{j^*} whose centroid is the closest to the incoming instance. The winning neuron's weights would be adjusted as follows:

$$V_{j^*}(t+1) = V_{j^*}(t) + \alpha[x - V_{j^*}(t)],$$

where α is the learning rate, indicating that the centroid of the winning cluster will move in the direction of x. As instances keep coming, the weights vector—i.e., the centroid of the hyper-spheres—tend to move toward the densest region of the space. This first stage of the CPN's training algorithm is a process of self-organizing clustering, although it is structured using a network.

The second part of the structure is a Grossberg learning [19]. We will redesign a different hidden layer and different connection from the hidden layer to the output layer.

B. Advantages and Disadvantages of the Original Model

To illustrate the advantage and disadvantage of original model, a set of two-dimensional artificial data were created and visualized in Fig. 6.9.

In Fig. 6.9a, instances can be grouped into six clusters. Setting the number of neurons in the hidden layer to six, the first training stage of the model in Fig. 6.8 can automatically find the centroids of the six clusters, which are represented by

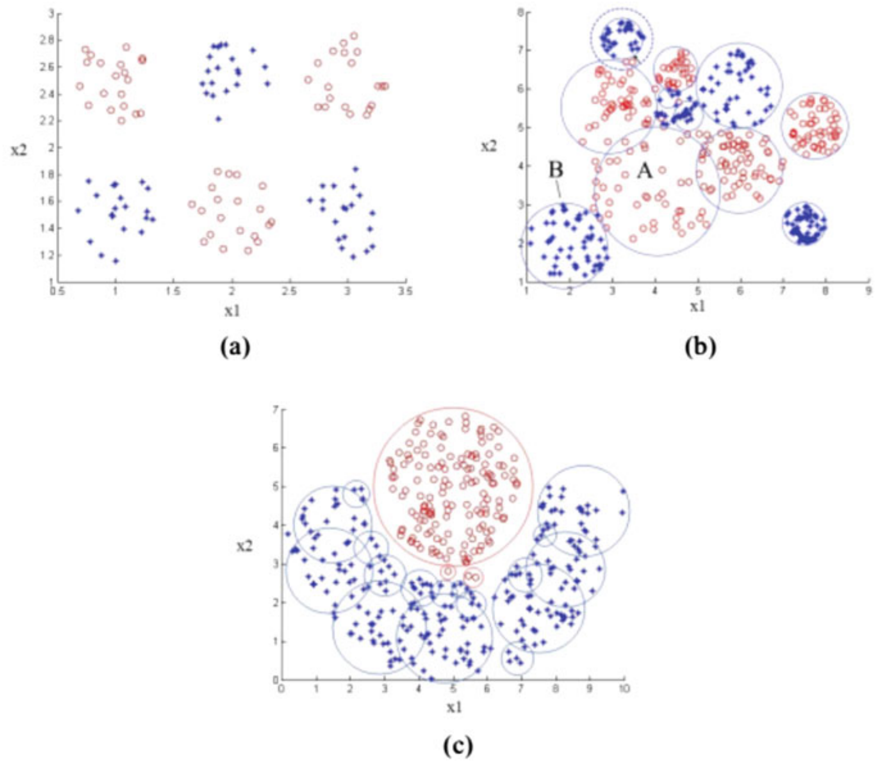


Fig. 6.9 Artificial datasets and the proposed clustering solutions

the weights of the six neurons. The second training stage can learn each cluster's connection to the right class. The distance from each instance in Fig. 6.9a to its cluster centroid is smaller than the distances to the centroids of other clusters. The dataset shown in Fig. 6.9 is ideal for CPN to classify.

Data distribution in Fig. 6.9a is simplified and idealistic. Data with distribution similar to Fig. 6.9b will cause two kinds of problems to the original model.

(1) First, the self-organized clustering process depends on the similarity measures between data points and hyper-sphere's centroid. Points closer to one cluster's centroid may belong to another cluster. Therefore, every cluster should have a definite scope or radius, and the scope should be as far away from others as possible.
(2) Second, the number of clusters in the hidden layer is fixed in the original model. However, it is difficult to estimate the number of clusters in advance. Given different numbers of neurons in the hidden layer, the accuracy varies dramatically. The training of the instar layer-i.e., the clustering process-is contingent on this fixed number.

C. Building of the DMZ

To solve the first aforementioned problem, we should have a general knowledge of the scope of the clusters. For example, points of cluster A (in Fig. 6.9b) near the border may be closer to the centroid of cluster B, so these points will be considered belong to cluster B in the original model. We must identify the decision border that separates clusters according to their labels. When two instances with conflicting labels fall into the same cluster, it gives us an opportunity to identify the border point that is somewhere between the two conflicting instances (as long as the instance is not an outlier). To maintain the maximum margin and for the sake of simplicity, the median point of two instances could be selected as a point in a zone called a Demilitarized Zone (DMZ), and clusters should be as far away from the DMZ as possible. As the number of conflicting instances increases, a general zone gradually forms as the DMZ. This mechanism can find borders of any shapes that are surrounded by many hyper-spheres.

To solve the second problem, the number of clusters should not be predetermined. The clusters should be formed dynamically and merged or split if necessary. The scope of the hyper-spheres, represented by the corresponding radii, should be adjusted on demand. As an example, consider the situation presented in Fig. 6.9b: with instances of conflicting labels found in the top cluster, the original cluster should tune its radius. After training, a new cluster would be formed beneath the top cluster containing instances of different labels from the ones in the top cluster. The radii of the two clusters should be tuned according to their distance to the borders.

One single hyper-sphere may not enclose an area whose shape is not hyper-spherical [51]. However, any shape could be enclosed as long as the number of the formed hyper-spheres is unlimited. Consider the clusters represented by the two-dimensional circles in Fig. 6.9c. All of the instances can be clustered no matter what the data distribution is and what the shape of the border is, as long as there are enough hyper-spheres of varying radii and are properly arranged.

D. Proposed Topological Structure

Given the solutions above, the structure of our improved model is as follows (Fig. 6.10):

The first difference is that our model has an adaptive dynamic hidden layer and the number of neurons in hidden layer is adaptive. The second difference is that each neuron H_i connects to only one particular neuron in the output layer, and w_{ij} is used to record the radius of neuron H_i .

E. Kernelization

It is challenging for competitive learning models to apply kernel methods because they cannot be denoted in inner-product forms. Some previous studies use approx-

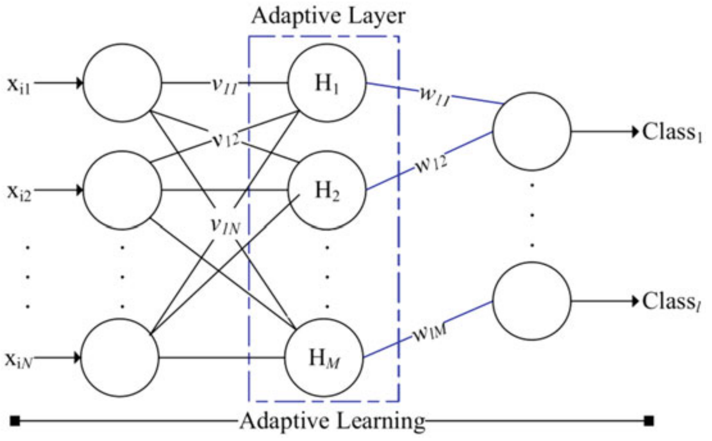


Fig. 6.10 Topological structure of the proposed model

imation methods for the kernelization of competitive learning [29, 76]. This paper uses Nyström method to kernelize the proposed model [28, 40].

Let the kernel matrix written in blocks form:

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix},$$

Let $C = [A_{11}\ A_{12}]^T$, Nyström method uses A_{11} and C to approximate large matrix A. Suppose C is a uniform sampling of the columns, Nyström method generates a rank-k approximation of $A(k \le n)$ and is defined by:

$$A_k^{nys} = CA_{11}^{+}C^T = \begin{bmatrix} A_{11} & A_{21} \\ A_{21} & A_{21}A_{11}^{+}A_{21}^{T} \end{bmatrix} \approx A,$$

where A_{11}^{+} denotes the generalized pseudo inverse of A_{11}.

There exists an Eigen decomposition $A_{11}^{+} = V\Lambda^{-1}V^T$ such that each element $A_k^{nys}{}_{ij}$ in A_k^{nys} can be decomposed as:

$$\begin{aligned} A_k^{nys}{}_{ij} &= (C_i^T V\Lambda^{-1}V^T C_j) \\ &= (\Lambda^{-1/2}V^T C_i)^T(\Lambda^{-1/2}V^T C_i) \\ &= (\Lambda^{-1/2}V^T(\kappa(x_i, x_1), \ldots, \kappa(x_i, x_m)))^T \bullet (\Lambda^{-1/2}V^T(\kappa(x_j, x_1), \ldots, \kappa(x_j, x_m))), \end{aligned}$$

where $\kappa(x_i, x_j)$ is the base kernel function, $x_1, x_2, \ldots, x_m$ are representative data points and can be obtained by uniform sampling or clustering methods such as K-means and SOFM.

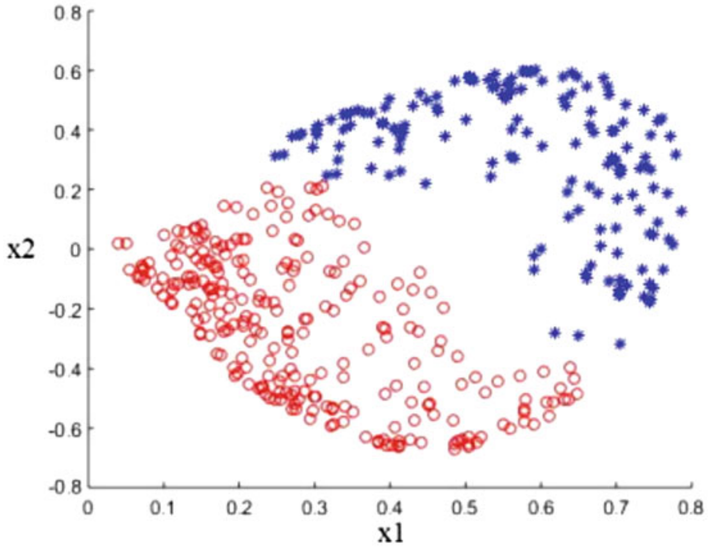


Fig. 6.11 Artificial dataset 3 after Nyström and SVD transformation

Let $\phi_m(x) = \Lambda^{-1/2}V^T(\kappa(x, x_1), \ldots, \kappa(x, x_m))^T$, such that ${A_k^{nys}}_{ij} = \phi_m(x_i)^T\phi_m(x_j) = \kappa(x_i, x_j)$.

With Nyström method, we can get an explicit approximation of the nonlinear projection $\phi_m(x)$, which is:

$$x \rightarrow \phi_m(x). \tag{6.117}$$

To justify why we use kernel methods for our model, we first used Nyström method to raise the dimension of dataset 3 to 403, then used Singular Value Decomposition (SVD) to reduce the dimension to 2 for the purpose of visualization. Figure 6.11 illustrates the transformed dataset 3 from Fig. 6.9c.

Compared with Fig. 6.9c, the data in Fig. 6.11 can be covered with less hyper-spheres, or each hyper-sphere can enclose more data points. Because the sampling points in Nyström methods can be obtained dynamically, the projection of Eq. (6.117) can be used for every single instance in competitive learning and can be applied directly to our incremental model.

Without loss of generality, we use $\phi_m(x)$ to denote a potential projection of x in the reminder of this paper. If it works in the original space, the projection of x is to itself.

6.3.1.2 Proposed Classifier: ADA-HS

The main characteristic of the proposed model is to adaptively build hyper-spheres. Therefore, we call the model Adaptive Hyper-Spheres (AdaHS), and the version after Nyström projection is called Nys-AdaHS.

A. Training Stages

Our algorithms are trained in three stages, which are described below.

Stage 1. Forming Hyper-Spheres and Adjusting Centroids and Radii

(1) Forming hyper-spheres and adjusting centroids

Given that instances are read dynamically, there is no hyper-sphere at the beginning. The first instance inputted forms a hyper-sphere whose centroid is itself and initial radius is set to a large value. When a new instance is inputted and does not fall into any existing hyper-spheres, a new hyper-sphere will be formed in the same way. If a new instance falls into one or more existing hyper-spheres, the winner is the one whose centroid is the closest to the new instance. The winning cluster's centroid is recalculated as:

$$c_i(t+1) = c_i(t) + \alpha[\phi(x) - c_i(t)],$$

where x is the new inputted instance, $c(t)$ is the original centroid of the hyper-sphere, $c(t+1)$ is the new centroid, and α is the learning rate.

When the number of instances that fall within a particular hyper-sphere grows, its centroid tends to move toward the densest zone.

In order to speed up the search of the winner, we build simple k-dimension trees for all hyper-spheres. With the knowledge of the radius, it is easy to figure out the upper and lower bounds of the selected k dimensions. In this way, it avoids extensive computation of all Euclidean distance of instance and hyper-sphere pairs.

(2) Building decision border zone: DMZ

The goal of this step is to find the DMZ's median points that approximate the shape of the DMZ.

We find the points using the following technique. The first time a labeled instance falls into a hyper-sphere, the hyper-sphere will be labeled using the label of this instance. If another instance with a conflicting label falls into the same hyper-sphere, it indicates that the hyper-sphere has entered the DMZ. We identify the nearest data point in the hyper-sphere to the newly inputted conflicting instance, and let p_i represent the median point as follows:

$$p_i = \frac{1}{2}(\phi(x_{conflicting}) + c_i),$$

where $\phi(x_{conflicting})$, $p_i \in c_i$ and p_i is recorded and used in the posterior clustering process.

(3) Adjusting the radii of hyper-spheres

Once a DMZ point is found in a hyper-sphere, the radius of the hyper-sphere should be updated such that it does not enter the DMZ. The new radius of hyper-sphere c_i should therefore be set as:

$$r_i = d(p_i, c_i) - d_{safe},$$

where d_{safe} represents a safe distance at which a hyper-sphere should be from the closest DMZ point. And the logics of this stage are outlined in Algorithm 6.19 below.

Algorithm 6.19 The forming of hyper-spheres and the adjusting of the centroids and radii

```
1:  Input: $x$, the newly read instance.
2:  Output: $C$: A set of hyper-spheres whose centroids and radii are tuned properly;
            $DMZ$: A set of points who shape the decision border approximately.
3:  Method:
4:  $c_t = Null, len = -1$.
5:  for each $c_t$ in $C$ do
6:      if $\phi(x)$ falls into $C$ then
7:          % Find the winner of the hyper-spheres.
8:          if $label(x) = label(c_i)$ and ($len = -1$ or $d_E(\phi(x), c_i) < len$) then
9:              $c_t = c_i$.% Store the present temporary nearest hyper-sphere
10:             $len = d_E(\phi(x), c_i)$.%Store the present temp nearest distance
11:         else if $label(x) \neq label(c_i)$ then %//Split the hyper-sphere
12:             $p_i = \frac{1}{2}(\phi(x_{conflicting}) + c_i), \phi(x_{conflicting}), p_i \in c_i$.
13:             Add $p_i$ to $DMZ$.
14:             $r_i = d(p_i, c_i) - d_{safe}$.% Adjusting radii $r_j$ of hyper-sphere $c_j$
15:             Mark $c_i$ as "support hyper-sphere".
16:         end if
17:     end if
18: end for
19: if $c_t \neq Null$ then
20:     % Adjust the winning hyper-sphere's centroid
21:     $c_i(t+1) = c_i(t) + \alpha[\phi(x) - c_i(t)]$.
22: else
23:     Form a new hyper-sphere, and make $\phi(x)$ be the centroid.
24:     Let the label of the new hyper-sphere be $label(x)$.
25: end if
```

Stage 2. Merging Hyper-Spheres

Hyper-spheres may overlap with one another or even be contained in others. Therefore, after certain period of training, a merging operation should be performed. Suppose that we have two hyper-spheres, c_A and c_B, and the radii of them are not the same. Let $c_{big} = \max_{radius}(c_A, c_B)$, $c_{small} = \min_{radius}(c_A, c_B)$, $d_t = d(c_{big}, c_{small})$, and θ be the merging coefficient. If $d_t + r_{small} \leq r_{big} + \theta \times r_{small}$, the prerequisite to merge is met. Then let $r_{temp} = d_t + r_{small}$, and the new radius of the c_{big} will be $r_{new} = \max(r_{temp}, r_{big})$.

The details of this stage are outlined in Algorithm 6.20.

Algorithm 6.20 Merging of hyper-spheres

```
1: **Input:** $C$: A set of hyper-spheres which are formed in stage 1.
2: **Output:** $C$: The remaining hyper-spheres after merging.
3: **Method:**
4: **for** each $c_i$ in $C$ **do**
5:     **for** each $c_j$ in $C$ except $c_i$ **do**
6:         $c_{big} = \max_{radius}(c_i, c_j)$, $c_{small} = \min_{radius}(c_i, c_j)$, $d_t = d(c_{big}, c_{small})$.
7:         **if** $d_t + r_{small} \le r_{big} + \theta \times r_{small}$ **then** % $\theta$ is the merging coefficient
8:             Merge $c_i$ and $c_j$.
9:         **end if**
10:    **end for**
11: **end for**
```

Stage 3. Selecting Hyper-Spheres

Since the training process is entirely autonomous, the number of generated hyper-spheres could be large. Therefore, the final stage needs to select hyper-spheres.

There are three types of hyper-spheres that are prominent, which are described as follows:

(1) The first type of hyper-spheres includes large number of instances. Because these are the fundamental hyper-spheres that contain most data points, they are marked as "Core Hyper-spheres".
(2) The second type of hyper-spheres has less instances but locates near the border. They are marked as "Support Hyper-spheres" because such hyper-spheres can be found by measuring the distance between hyper-spheres and the nearest DMZ points.
(3) The third type of hyper-spheres has small number of instances and is far away from the border. These hyper-spheres can be discarded.

    To achieve high classification accuracy, both core hyper-spheres and support hyper-spheres should be selected. The logic of the third stage is outlined in Algorithm 6.21.

B. Mini-Batch Learning and Distributed Computing

To make it applicable in large scale applications, we encapsulate the proposed algorithms into a Map-Reduce framework. We can collect the incoming instances as mini-batch set and then train them in MapReduce tasks. The computing model of the algorithms is illustrated in Fig. 6.12.

The collected mini-batch instances can be encapsulated in key-value pairs and mapped into mapper tasks.

Algorithm 6.21 Selection of hyper-spheres

```
1: Input: $C$: The set of hyper-spheres which are formed in preceding stages.
2: Output: $C$: The remaining hyper-spheres after selection.
3: Method:
4: for each $c_i$ in $C$ do
5:     % $T$ is the threshold of the instances number which one hyper-sphere must at least have.
6:     % $num(c)$ is a function computing the number of instances in a hyper-sphere.
7:     if $num(c_i) < T$ then
8:         % Let $d(c_i, DMZ)$ be the distance from the centroid of $c_i$ to the nearest data point in $DMZ$.
9:         if $r_i < d(c_i, DMZ)$ then
10:            Discard $c_i$.
11:        end if
12:    else
13:        Mark $c_i$ as "core hyper-sphere".
14:    end if
15: end for
```

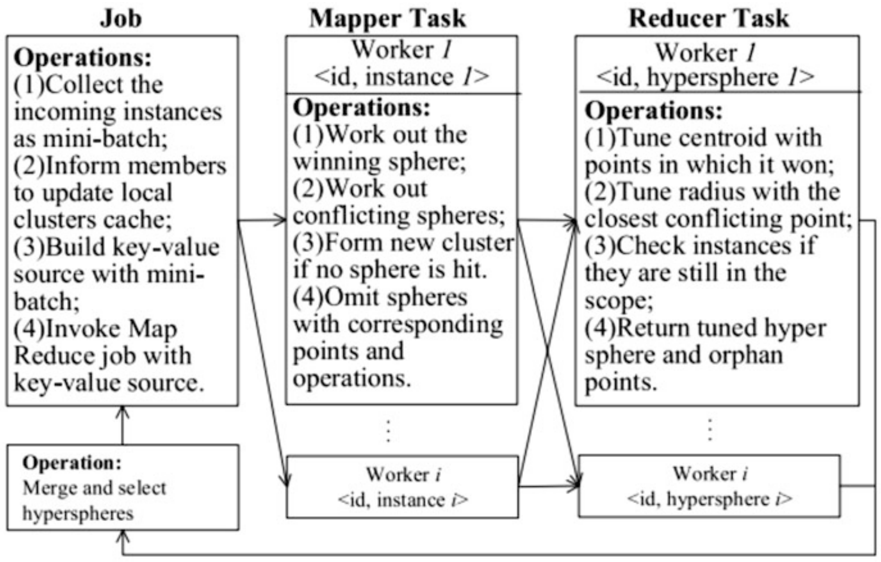

Job
Operations:
(1)Collect the incoming instances as mini-batch;
(2)Inform members to update local clusters cache;
(3)Build key-value source with mini-batch;
(4)Invoke Map Reduce job with key-value source.
Operation:
Merge and select hyperspheres
Mapper Task
Worker 1
<id, instance 1>
Operations:
(1)Work out the winning sphere;
(2)Work out conflicting spheres;
(3)Form new cluster if no sphere is hit.
(4)Omit spheres with corresponding points and operations.
Worker i
<id, instance i>
Reducer Task
Worker 1
<id, hypersphere 1>
Operations:
(1)Tune centroid with points in which it won;
(2)Tune radius with the closest conflicting point;
(3)Check instances if they are still in the scope;
(4)Return tuned hyper sphere and orphan points.
Worker i
<id, hypersphere i>

Fig. 6.12 MapReduce computing model

In each mapper tasks, the operations are based on instances. It queries local cache for every instance to find out in which hyper-spheres the instance falls, marks the winning hyper-sphere and the conflicting ones, and sents the hyper-spheres along with the description of the needed operations in another form of key-value<id, hyper-sphere> pairs.

In each reducer task, the operations are based on every hyper-sphere, which is aggregated according to the hyper-sphere id emitted from mapper tasks. The competitive learning can be conducted collectively with the aggregated instances.

The tuning of a radius can be performed for only once with the closest conflicting instance, and it should find out the orphan points and return the tuned hyper-sphere at the end.

After a turn of the MapReduce tasks, the merging and selection of the hyper-spheres should be performed. After all of the operations, the tuned hyper-spheres should be saved to the cache. The orphan points should be retrained in the next turn. In the whole MapReduce process, sub-tasks do not coordinate with each other. Thus the hyper-spheres and DMZ are not updated in real time in a mini-batch turn, and they are updated collectively after all reducer tasks return.

C. Predicting Labels

Just like other supervised competitive neural networks, AdaHS must determine the winning hyper-sphere in the hidden layer to predict the label of a new instance. There are two situations. In the first situation, the new instance falls into an existing hyper-sphere and the label of the instance is determined by the label of the hyper-sphere. In the second situation, the new instance does not fall into an existing hyper-sphere, and the label of the new instance is coordinated by the k nearest hyper-spheres' labels:

$$y = \arg\max_{l_j} \sum_{c_i \in N_k(x)} w_j I(y_i = l_j),$$

where $w_j = exp(-([d_E(\phi(x), c_j)^2]/[2r_j^2]))$; $i = 1, 2, \ldots, L$; $j = 1, 2, \ldots, k$; $N_k(x)$ is the k nearest hyper-spheres; and I is the indicator function. The default value of k is set to 3.

6.3.2 A Construction of Robust Representations for Small Data Sets Using Broad Learning System

6.3.2.1 Review of Broad Learning System

This subsection is mainly a simple introduction to the BLS. The details of this system can be found in [11]. The BLS is designed based on the random vector functional-link neural network (RVFLNN) [25, 47]. In the BLS, the mapped features and enhancement features instead of the original features are used to feed into a single layer neural network. Figure 6.13 shows the structure of the BLS.

In Fig. 6.13, X means the input features, and Y means the corresponding labels. The label Y uses one-hot encoding, which means all neurons are set to 0 except the one that belongs to the label is set to 1. The mapped features can be represented as

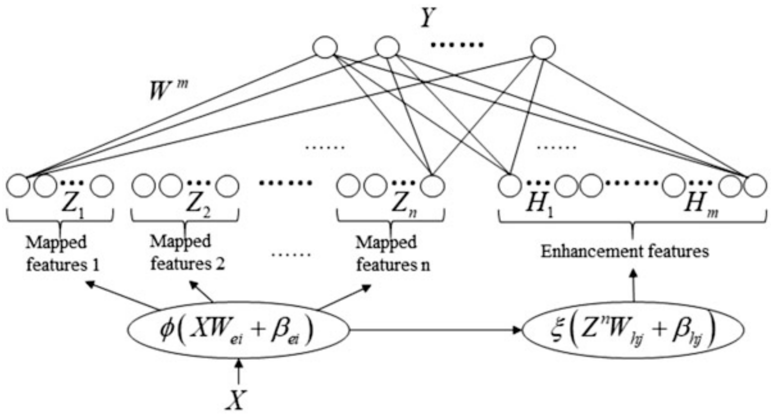


Fig. 6.13 The structure of the BLS

follows:

$$Z_i = \phi_i(XW_{ei} + \beta_{ei}), \tag{6.118}$$

where Z_i is the i-th mapped features and W_{ei} is the random weights. All the mapped features are concatenated as $Z^n \equiv [Z_1, Z_2, \dots, Z_n]$, then the enhancement features can be represented as follows:

$$H_j = \xi_j(Z^n W_{hj} + \beta_{hj}). \tag{6.119}$$

All the enhancement features are concatenated as $H^m \equiv [H_1, H_2, \dots, H_m]$. Therefore, the broad model can be represented as follows:

$$Y = [Z^n|H^m]W^m, \tag{6.120}$$

where $W^m = [Z^n|H^m]^+ Y$ is the weights of the single-layer neural network and can be easily calculated through the ridge regression approximation of $[Z^n|H^m]^+$ using the following equation:

$$A^+ = \lim_{\lambda \to 0}(\lambda I + AA^T)^{-1}A^T. \tag{6.121}$$

Theoretically, the $\phi_i(\cdot)$ and $\xi_j(\cdot)$ used in mapped features and enhancement features can be different functions. The sparse autoencoder is applied to fine-tune the W_{ei} of mapped features, and the sigmoid function is used to generate enhancement features in [11].

6.3.2.2 Proposed BLS Framework and BLS with RLA

A. BLS Framework

To extend the BLS to a framework of transforming inputs into robust representations, feature extraction methods instead of random mapping are used to generate mapped features. Let $Z_i = \phi_i(X)$ denote the i-th mapped features, where $\phi_i(\cdot)$ can be any feature extraction method. Different feature extraction methods can generate different mapped features. Even if all mappings of mapped features use the same AE method, the mapped features are different due to the randomness of neural networks. All the mapped features are concatenated as $Z^n \equiv [Z_1, Z_2, \ldots, Z_n]$, and the ensemble of mapped features Z^n can provide a robust representation of inputs.

The setting of a large number of enhancement nodes in the original BLS is removed. Deep representations, called enhancement features, are learned from the ensemble mapped features Z^n. The enhancement features can be denoted as $H_j = \xi_j(Z^n)$, where $\xi_j(\cdot)$ can be any feature extraction method. All the enhancement features are concatenated as $H^m \equiv [H_1, H_2, \ldots, H_m]$. The concatenation of mapped features Z^n and enhancement features H^m can provide more robust representations to enhance the performance of downstream tasks.

Figure 6.14 shows the structure of the BLS framework. It should be noted that w and β used in (6.118) and (6.119) are random, so their method is random mapping. The mappings $\phi(\cdot)$ and $\xi(\cdot)$ in Fig. 6.14 can be any feature extraction method, including random mapping, autoencoder, convolution feature extraction, recursive feature extraction, etc. Therefore, w and β are omitted in the new equations.

Further, to generate more different mapped features and enhancement features in the BLS framework, samples and features can be randomly selected for each mapping. Figure 6.15 shows the structure of a random version of the BLS framework.

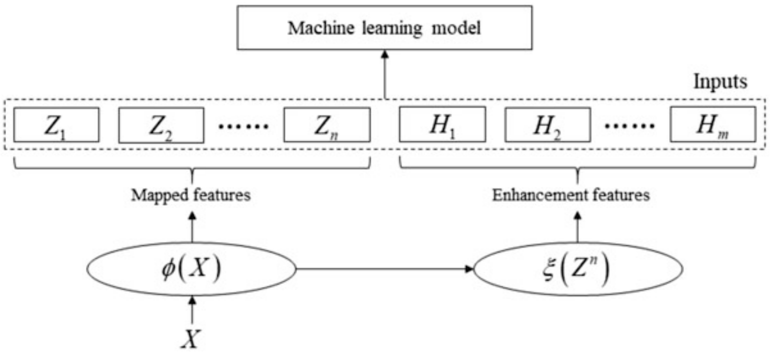


Fig. 6.14 The structure of the BLS framework

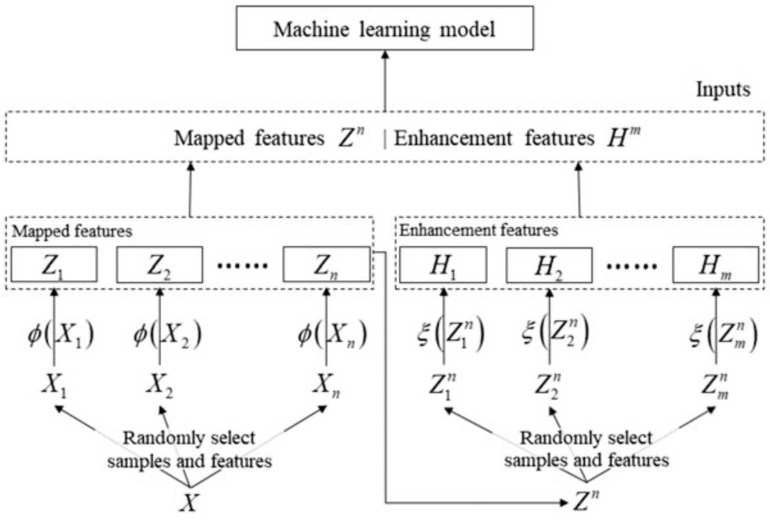


Fig. 6.15 The structure of a random version of the BLS framework

B. BLS with RLA

Deep autoencoder (DA) is a nonlinear dimensionality reduction approach and usually works much better than PCA [20]. Instead of the unsupervised architecture used in DA, LA uses supervised architecture to connect the features and the labels together. The representation features learned from the LA not only contain the information of original features but also contain the estimated label belonging to the sample. In that case, the representation features can provide more information to the machine learning models and may promote the performance of these models. Figure 6.16 shows the structure of the LA.

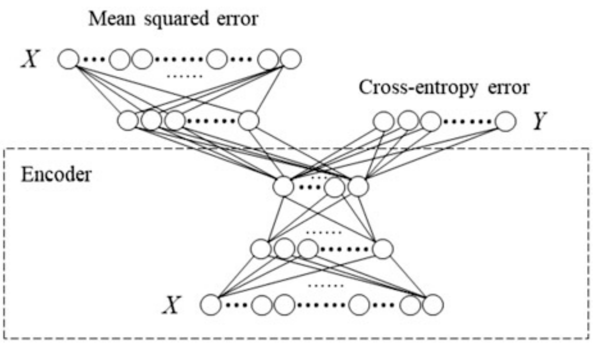


Fig. 6.16 The structure of the LA

In Fig. 6.16, X means the input features, and Y means the corresponding labels using one-hot encoding. The label layer is added on the top of the representation layer, and a softmax function is used to forecast the label. The loss of the reconstruction of X is measured by the mean squared error (MSE), and the loss of forecasting label Y is measured by the cross-entropy error (CEE). Therefore, the loss function of LA can be represented as follows:

$$loss = \frac{1}{n}\sum_{i=1}^{n}(\alpha(x_i - \hat{x}_i)^2 - \beta y_i log \hat{y}_i),$$

where n is the number of samples, x_i is the i-th sample, $\hat{x}_l$ is the i-th reconstruction sample, y_i is the i-th sample's one-hot label, $\hat{y}_l$ is the i-th sample's softmax output, α and β are the scale factors.

To illustrate how the BLS framework works, LA is embedded in the BLS framework as an example. More specifically, the mappings $\phi_i(\cdot)$ and $\xi_j(\cdot)$ in the BLS framework is the same feature extraction method, LA. The mapped feature Z_n is learned by using the original data X as the input and output of the LA. After learning n mapped features, all the mapped features are concatenated as Z^n. The enhancement feature H_m is learned by using the Z^n as the input and output of the LA, then all the mapped features and enhancement features are concatenated as the final input and fed into any machine learning model.

Because of the random initialization of the weights of LA, each mapped feature and enhancement feature will be different. Further, randomly picked samples and randomly picked features can be used as inputs for each LA, and the random label-based autoencoder (RLA) can generate more different mapped features and enhancement features in the BLS. The randomness of RLA is controlled by two parameters: selected sample size and selected feature size. If the selected sample size and the selected feature size are less than 1, samples and features are randomly picked according to these two selected sizes. If the selected sample size and the selected feature size are equal to 1, all samples and features are used to train RLA. Therefore, LA is a special case of RLA.

Given a two-layer encoder structure, the input fed into a machine learning model is as follows:

$$\begin{aligned} input &= [Z^n | H^m] \\ &= [Z_1, \ldots, Z_n | H_1, \ldots, H_m] \\ &= \begin{bmatrix} \sigma(w_{11}\sigma(w_{12}x + b_{12}) + b_{11}), \ldots, \sigma(w_{n1}\sigma(w_{n2}x + b_{n2}) + b_{n1})| \\ \sigma(w'_{11}\sigma(w'_{12}Z^n + b'_{12}) + b'_{11}), \ldots, \sigma(w'_{m1}\sigma(w'_{m2}Z^n + b'_{m2}) + b'_{m1}) \end{bmatrix}, \end{aligned}$$

where w is the weight, b is the bias, and $\sigma(\cdot)$ is the activation function.

The number of mapped features n and the number of enhancement features m are different and depend on the complexity of modeling problems. Additional mapped features and enhancement features can be added to achieve a better performance when the setting (n, m) cannot reach the desired accuracy. The pseudocode of the BLS with RLA is shown in Algorithms 6.22 and 6.23.

Algorithm 6.22 Broad learning: increment of additional enhancement features

```
 1: for $i = 0, i < n$ do
 2:     Train RLA model with training data set.
 3:     Generate $Z_i$ using the RLA.
 4: end for
 5: Concatenate the mapped features $Z^n \equiv [Z_1, Z_2, \ldots, Z_n]$.
 6: for $j = 0, j < m$ do
 7:     Train RLA model with data set $Z^n$.
 8:     Generate $H_j$ using the RLA.
 9: end for
10: Concatenate the mapped features and enhancement features $[Z^n|H^m]$ as inputs.
11: Train the machine learning model.
12: while VALIDATION ERROR is not satisfied do
13:     Train RLA model with data set $Z^n$.
14:     Generate $H_{m+1}$ using the RLA.
15:     Concatenate the mapped features and enhancement features $[Z^n|H^{m+1}]$ as inputs.
16:     Train the machine learning model.
17:     $m = m + 1$.
18: end while
```

Algorithm 6.23 Broad learning: increment of additional mapped features

```
 1: for $i = 0, i < n$ do
 2:     Train RLA model with training data set.
 3:     Generate $Z_i$ using the RLA.
 4: end for
 5: Concatenate the mapped features $Z^n \equiv [Z_1, Z_2, \ldots, Z_n]$.
 6: for $j = 0, j < m$ do
 7:     Train RLA model with data set $Z^n$.
 8:     Generate $H_j$ using the RLA.
 9: end for
10: Concatenate the mapped features and enhancement features $[Z^n|H^m]$ as inputs.
11: Train the machine learning model.
12: while VALIDATION ERROR is not satisfied do
13:     Train RLA model with training data set.
14:     Generate $Z_{n+1}$ using the RLA.
15:     Concatenate the mapped features $Z^{n+1} \equiv [Z^n, Z_{n+1}]$.
16:     for $j = 0, j < m$ do
17:         Train RLA model with data set $Z^{n+1}$.
18:         Generate $H_j$ using the RLA.
19:         Concatenate the mapped features and enhancement features $[Z^{n+1}|H^m]$ as inputs.
20:         Train the machine learning model.
21:     end for
22:     $n = n + 1$.
23: end while
```

In addition, it should be noted that the BLS is not conflicted with feature selection methods. The BLS can be used before or after the feature selection methods, and the selected features can also be concatenated with the mapped features and enhancement features.

References

1. Ardehaly, E.M., Culotta, A.: Co-training for demographic classification using deep learning from label proportions. In: 2017 IEEE International Conference on Data Mining Workshops (ICDMW) (2017)
2. Arjovsky, M., Bottou, L.: Towards principled methods for training generative adversarial networks. Stat **1050** (2017)
3. Banfield, R.E., Hall, L.O., Bowyer, K.W., Kegelmeyer, W.P.: A comparison of decision tree ensemble creation techniques. IEEE Trans. Pattern Anal. Mach. Intell. **29**(1), 173–180 (2006)
4. Belohlávek, R., Sklenar, V., Zacpal, J.: Crisply generated fuzzy concepts. In: Formal Concept Analysis, Third International Conference, ICFCA 2005, Lens, France, February 14–18, 2005, Proceedings (2005)
5. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer (2006)
6. Bosch, A., Zisserman, A., Munoz, X.: Image classification using random forests and ferns. In: 2007 IEEE 11th International Conference on Computer Vision, pp. 1–8. IEEE (2007)
7. Boyd, S., Boyd, S.P., Vandenberghe, L.: Convex Optimization. Cambridge University Press (2004)
8. Breiman, L.: Random forests. Machine Learning **45**(1), 5–32 (2001)
9. Caruana, R., Karampatziakis, N., Yessenalina, A.: An empirical evaluation of supervised learning in high dimensions. In: Proceedings of the 25th International Conference on Machine Learning, pp. 96–103 (2008)
10. Chapelle, O., Sindhwani, V., Keerthi, S.S.: Optimization techniques for semi-supervised support vector machines. J. Mach. Learn. Res. **9**(2) (2008)
11. Chen, C.P., Liu, Z.: Broad learning system: An effective and efficient incremental learning system without the need for deep architecture. IEEE Trans. Neural Networks Learn. Syst. **29**(1), 10–24 (2017)
12. Christmann, A., Steinwart, I.: How svms can estimate quantiles and the median. In: Advances in Neural Information Processing Systems, pp. 305–312 (2007)
13. Dai, Z., Yang, Z., Yang, F., Cohen, W.W., Salakhutdinov, R.: Good semi-supervised learning that requires a bad gan. Preprint (2017). arXiv:1705.09783
14. Feldman, J.: Minimization of boolean complexity in human concept learning. Nature **407**(6804), 630–633 (2000)
15. Gall, J., Lempitsky, V.: Class-specific hough forests for object detection. In: Decision Forests for Computer Vision and Medical Image Analysis, pp. 143–157. Springer (2013)
16. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer Science & Business Media (2012)
17. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial networks. Preprint (2014). arXiv:1406.2661
18. Grandvalet, Y., Bengio, Y., et al.: Semi-supervised learning by entropy minimization. In: CAP, pp. 281–296 (2005)
19. Grossberg, S.: Adaptive resonance theory: How a brain learns to consciously attend, learn, and recognize a changing world. Neural Networks **37**, 1–47 (2013)
20. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. Science **313**(5786), 504–507 (2006)

21. Huang, G.B., Zhu, Q.Y., Siew, C.K.: Extreme learning machine: theory and applications. Neurocomputing **70**(1-3), 489–501 (2006)
22. Huang, G.B., Zhou, H., Ding, X., Zhang, R.: Extreme learning machine for regression and multiclass classification. IEEE Trans. Syst. Man Cybern. B (Cybern.) **42**(2), 513–529 (2011)
23. Huang, X., Shi, L., Suykens, J.A.: Support vector machine classifier with pinball loss. IEEE Trans. Pattern Anal. Mach. Intell. **36**(5), 984–997 (2013)
24. Huang, X., Shi, L., Suykens, J.A.: Sequential minimal optimization for svm with pinball loss. Neurocomputing **149**, 1596–1603 (2015)
25. Igelnik, B., Pao, Y.H.: Stochastic choice of basis functions in adaptive function approximation and the functional-link net. IEEE Trans. Neural Netw. **6**(6), 1320–1329 (1995)
26. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: European Conference on Computer Vision, pp. 694–711. Springer (2016)
27. Kang, X., Miao, D.: A study on information granularity in formal concept analysis based on concept-bases. Knowl. Based Syst. **105**, 147–159 (2016)
28. Kumar, S., Mohri, M., Talwalkar, A.: Sampling methods for the nyström method. J. Mach. Learn. Res. **13**(1), 981–1006 (2012)
29. Lai, J., Wang, C.: Kernel and graph: Two approaches for nonlinear competitive learning clusterin. Front. Electr. Electron. Eng. **7**(1), 134–146 (2012)
30. Lake, B.M., Salakhutdinov, R., Tenenbaum, J.B.: Human-level concept learning through probabilistic program induction. Science **350**(6266), 1332–1338 (2015)
31. Leistner, C., Saffari, A., Santner, J., Bischof, H.: Semi-supervised random forests. In: 2009 IEEE 12th International Conference on Computer Vision, pp. 506–513. IEEE (2009)
32. Leistner, C., Saffari, A., Bischof, H.: Miforests: Multiple-instance learning with randomized trees. In: European Conference on Computer Vision, pp. 29–42. Springer (2010)
33. Li, J., Laird, J.: Spontaneous retrieval from long-term memory for a cognitive architecture. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 29 (2015)
34. Li, J., Mei, C., Lv, Y.: Knowledge reduction in decision formal contexts. Knowl. Based Syst. **24**(5), 709–715 (2011)
35. Li, J., Huang, C., Xu, W., Qian, Y., Liu, W.: Cognitive concept learning via granular computing for big data. In: International Conference on Machine Learning & Cybernetics (2015)
36. Li, J., Mei, C., Xu, W., Qian, Y.: Concept learning via granular computing: A cognitive viewpoint. Information Sciences **298**, 447–467 (2015)
37. Li, J., Huang, C., Qi, J., Qian, Y., Liu, W.: Three-way cognitive concept learning via multi-granularity. Information Sciences **378**, 244–263 (2017)
38. Li, T., Kou, G., Peng, Y., Shi, Y.: Classifying with adaptive hyper-spheres: An incremental classifier based on competitive learning. IEEE Trans. Syst. Man Cybern. Syst. **50**(4), 1218–1229 (2017)
39. Liu, J., Wang, B., Qi, Z., Tian, Y., Shi, Y.: Learning from label proportions with generative adversarial networks. Preprint (2019). arXiv:1909.02180
40. Lu, J., Hoi, S.C., Wang, J., Zhao, P., Liu, Z.Y.: Large scale online kernel learning. J. Mach. Learn. Res. **17**(47), 1 (2016)
41. Macdonald, B.A., Witten, I.H.: A framework for knowledge acquisition through techniques of concept learning. IEEE Trans. Syst. Man Cybern. **19**(3), 499–512 (1989)
42. Mi, Y., Liu, W., Shi, Y., Li, J.: Semi-supervised concept learning by concept-cognitive learning and concept space. IEEE Trans. Knowl. Data Eng. (2020). https://doi.org/10.1109/TKDE.2020.3010918
43. Mi, Y., Shi, Y., Li, J., Liu, W., Yan, M.: Fuzzy-based concept learning method: exploiting data with fuzzy conceptual clustering. IEEE Trans. Cybern., 1–12 (2020)
44. Modha, D.S., Ananthanarayanan, R., Esser, S.K., Ndirango, A., Sherbondy, A.J., Singh, R.: Cognitive computing. Commun. ACM **54**(8), 62–71 (2011)
45. Moosmann, F., Triggs, B., Jurie, F.: Fast discriminative visual codebooks using randomized clustering forests. In: Twentieth Annual Conference on Neural Information Processing Systems (NIPS'06), pp. 985–992. MIT Press (2006)

46. Pal, M.: Random forest classifier for remote sensing classification. Int. J. Remote Sens. **26**(1), 217–222 (2005)
47. Pao, Y.H., Park, G.H., Sobajic, D.J.: Learning and generalization characteristics of the random vector functional-link net. Neurocomputing **6**(2), 163–180 (1994)
48. Pei, D., Li, M.Z., Mi, J.S.: Attribute reduction in fuzzy decision formal contexts. In: 2011 International Conference on Machine Learning and Cybernetics, vol. 1, pp. 204–208. IEEE (2011)
49. Perbet, F., Stenger, B., Maki, A.: Random forest clustering and application to video segmentation. In: BMVC, pp. 1–10. Citeseer (2009)
50. Quan, T.T., Hui, S.C., Cao, T.H.: A fuzzy fca-based approach to conceptual clustering for automatic generation of concept hierarchy on uncertainty data. In: CLA, pp. 1–12 (2004)
51. Rehman, M.Z.u., Li, T., Yang, Y., Wang, H.: Hyper-ellipsoidal clustering technique for evolving data stream. Knowl. Based Syst. **70**, 3–14 (2014)
52. RogerKoenker: Quantile Regression. Cambridge University Press (2005)
53. Saffari, A., Leistner, C., Santner, J., Godec, M., Bischof, H.: On-line random forests. In: 2009 Ieee 12th International Conference on Computer Vision Workshops, ICCV Workshops, pp. 1393–1400. IEEE (2009)
54. Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X.: Improved techniques for training gans. Preprint (2016). arXiv:1606.03498
55. Shao, M.W., Leung, Y., Wang, X.Z., Wu, W.Z.: Granular reducts of formal fuzzy contexts. Knowl. Based Syst. **114**, 156–166 (2016)
56. Sharp, T.: Implementing decision trees and forests on a gpu. In: European Conference on Computer Vision, pp. 595–608. Springer (2008)
57. Shi, Y., Liu, J., Qi, Z., Wang, B.: Learning from label proportions on high-dimensional data. Neural Networks **103**, 9–18 (2018)
58. Shi, Y., Mi, Y., Li, J., Liu, W.: Concept-cognitive learning model for incremental concept learning. IEEE Trans. Syst. Man Cybern. Syst. (2018). https://doi.org/10.1109/TSMC.2018.2882090
59. Shi, Y., Cui, L., Chen, Z., Qi, Z.: Learning from label proportions with pinball loss. Int. J. Mach. Learn. Cybern. **10**(1), 187–205 (2019)
60. Shi, Y., Mi, Y., Li, J., Liu, W.: Concurrent concept-cognitive learning model for classification. Information Sciences **496**, 65–81 (2019)
61. Springenberg, J.T.: Unsupervised and semi-supervised learning with categorical generative adversarial networks. Preprint (2015). arXiv:1511.06390
62. Srivastava, N., Vul, E.: A simple model of recognition and recall memory. In: Proceedings of the 31st International Conference on Neural Information Processing Systems, pp. 292–300 (2017)
63. Steinwart, I., Christmann, A., et al.: Estimating conditional quantiles with the help of the pinball loss. Bernoulli **17**(1), 211–225 (2011)
64. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2818–2826 (2016)
65. Tadrat, J., Boonjing, V., Pattaraintakorn, P.: A new similarity measure in formal concept analysis for case-based reasoning. Expert Syst. Appl. **39**(1), 967–972 (2012)
66. Tang, H., Dong, P., Shi, Y.: A construction of robust representations for small data sets using broad learning system. IEEE Trans. Syst. Man Cybern. Syst., 1–11 (2019)
67. Tho, Q.T., Hui, S.C., Fong, A.C.M., Cao, T.H.: Automatic fuzzy ontology generation for semantic web. IEEE Trans. Knowl. Data Eng. **18**(6), 842–856 (2006)
68. Wang, Y.: On cognitive computing. Int. J. Softw. Sci. Comput. Intell. **1**(3), 1–15 (2011)
69. Wang, Y., Wang, Y.: Cognitive informatics models of the brain. IEEE Trans. Syst. Man Cybern. C (Appl. Rev.) **36**(2), 203–207 (2006)
70. Wang, Y., Chiew, V.: On the cognitive process of human problem solving. Cogn. Syst. Res. **11**(1), 81–92 (2010)

71. Wang, Z., Feng, J.: Multi-class learning from class proportions. Neurocomputing **119**, 273–280 (2013)
72. Wang, Y., Howard, N., Kacprzyk, J., Frieder, O., Sheu, P., Fiorini, R.A., Gavrilova, M.L., Patel, S., Peng, J., Widrow, B.: Cognitive informatics: Towards cognitive machine learning and autonomous knowledge manipulation. Int. J. Cogn. Inf. Nat. Intell. (IJCINI) **12**(1), 1–13 (2018)
73. Warde-Farley, D., Goodfellow, I.: Adversarial perturbations of deep neural networks. Perturbat. Optim. Stat. **311** (2016)
74. Wei, L., Qi, J., Zhang, W.: Attribute reduction theory of concept lattice based on decision formal contexts. Sci. China F Inf. Sci. **51**(7), 910–923 (2008)
75. Wille, R.: Restructuring lattice theory: an approach based on hierarchies of concepts. In: International Conference on Formal Concept Analysis, pp. 314–339. Springer (2009)
76. Wu, J.S., Zheng, W.S., Lai, J.H.: Approximate kernel competitive learning. Neural Networks **63**, 117–132 (2015)
77. Xiao, C., Wang, W., Lin, X., Shang, H.: Top-k set similarity joins. In: 2009 IEEE 25th International Conference on Data Engineering, pp. 916–927. IEEE (2009)
78. Yahia, S.B., Jaoua, A.: Discovering knowledge from fuzzy concept lattice. In: Data Mining and Computational Intelligence, pp. 167–190. Springer (2001)
79. Yu, J.: Machine learning: From axioms to algorithms (2017)
80. Yu, F., Liu, D., Kumar, S., Tony, J., Chang, S.F.: $\propto$svm for learning with label proportions. In: International Conference on Machine Learning, pp. 504–512. PMLR (2013)
81. Zadeh, L.A.: Fuzzy sets and information granularity. Adv. Fuzzy Set Theory Appl. **11**, 3–18 (1979)
82. Zhang, W.X., Qiu, G.F.: Uncertain Decision Making Based on Rough Sets. Publishin of Tsinghua University, Beijing (2005)
83. Zhang, W.X., Ma, J.M., Fan, S.Q.: Variable threshold concept lattices. Information Sciences **177**(22), 4883–4892 (2007)
84. Zhang, F., Liu, J., Wang, B., Qi, Z., Shi, Y.: A fast algorithm for multi-class learning from label proportions. Electronics **8**(6), 609 (2019)