Chapter 10
Case Study of Network-Based Semi-Supervised Learning: Stochastic Competitive-Cooperative Learning in Networks

Abstract Information reaches us at a remarkable speed and the amount of data it brings is unprecedented. In many situations, only a small subset of data items can be effectively labeled. This is because the labeling process is often expensive, time consuming, and requires intensive human involvement. As a result, partially labeled data sets are more frequently encountered. In order to get a better characterization of partially labeled data sets, semi-supervised classifiers are designed to learn from both labeled and unlabeled data. It has turned out to be a new topic of machine learning research that has received increasing attention in the past years. In this chapter, the semi-supervised classification with focus on methods based on complex networks is explored. In special, the particle competition model that we have introduced in the previous chapter is adapted to this new learning paradigm. Specifically, this enhancement is achieved by introducing the idea of cooperation among the particles and by changing the inner mechanisms of the original algorithm so as to fit it into a semi-supervised environment. In contrast to the unsupervised learning model, where the particles are randomly spawned in the network because no prior analysis of the groups is available, the semi-supervised learning version does have some external knowledge by definition. This knowledge is represented by the labeled data items, usually offered as a small fraction of the entire data set. In this scenario, the objective is to propagate the labels from the labeled set to the unlabeled set. Likewise the previous chapter, a mathematical formalization of the model, as well as a theoretical analysis, is also provided. A great portion of this analysis is based on the model that we have studied in the last chapter. A validation is also presented linking the numerical and theoretical results. An application in imperfect data learning is also presented, where the particle competition model is employed to detect and prevent error propagation in the learning process due to noisy or wrongly labeled data.

10.1 A Quick Overview of the Chapter

As we have seen in Chap. 3, the semi-supervised learning differs from the unsupervised learning by the fact that the former has some external knowledge incorporated into the learning process, by means of pre-labeled data items. Moreover, semi-

© Springer International Publishing Switzerland 2016

T.C. Silva, L. Zhao, *Machine Learning in Complex Networks*,
DOI 10.1007/978-3-319-17290-3_10

supervised learning also differs from supervised learning because it uses both labeled and unlabeled data in the learning process.

In the initial part of this chapter, the semi-supervised particle competition model is presented [3, 11]. Once the basic concepts are properly introduced, we deal with the interesting problem of learning with imperfect data. In this case, the labeled data set is not totally reliable. Situations in which the training data is not perfectly reliable may arise when noises are embedded in the labeling source procedure or in the acquisition of the data items, or even when an external professor incorrectly labels data items (human error). Though being prone to all of these kinds of error sources, most semi-supervised learning algorithms assume a perfectly reliable training data. As we will see, their performances are largely impacted when that assumption is violated.

Due to the practicability in the real-world, the semi-supervised model of multiple particles is further enhanced to deal with this uncertainty in the training data. For that, the model is equipped with mechanisms to detect and to prevent wrongly labeled training data. These mechanisms only use information that the dynamical system of multiple particles generates. Therefore, the procedure of detection and prevention of imperfect data is embedded within the model. We show that the modified particle competition model can really provide good results even in environments where the training data reliability is low. We analyze how the model's accuracy rate behaves as we increase the percentage of noise or wrongly labeled items in the training data set. In this analysis, we find critical points that are characterized by border regions in which small increases in the noise in the labeled set produces large downfalls in the model's accuracy. We compare the critical points of the semi-supervised model of multiple particles with other competing techniques and show that the former can withstand much more noisy environments. We also show that, provided that the fraction of correctly labeled vertices is the majority, the model can in general correctly re-label the wrongly labeled vertices. This constraint is intuitive and confirms that the competitive model uses some kind of "natural selection" in the learning process that forces the majority to overwhelm the minority using the network topology.

10.2 Description of the Stochastic Competitive-Cooperative Model

In this section, we describe the semi-supervised version of the particle competition model in detail [11].

10.2.1 Differences of the Semi-Supervised and the Unsupervised Versions

The main difference in the semi-supervised version of the model of multiple interacting particles is that now each particle represents a labeled data item. The goal of each particle is to spread the associated label of its labeled vertex to other unlabeled vertices by visiting and dominating them in a competitive way. The labeled vertex that each particle represents is termed as the *home vertex* of that particle. Particles always start the dynamical process at their home vertices. In the reanimation procedure of particles, once exhausted, they always regress to their corresponding home vertices and not to random dominated vertices. Moreover, particles are guaranteed to always dominate their home vertices in such a way that the domination levels imposed on home vertices are not sensitive to visits of rival particles.

Figure 10.1 illustrates the initial condition of the semi-supervised version of the particle competition model. Note that the initial location of particles is deterministically established in accordance with their home vertices. Moreover, the number of particles equals the number of labeled data items.

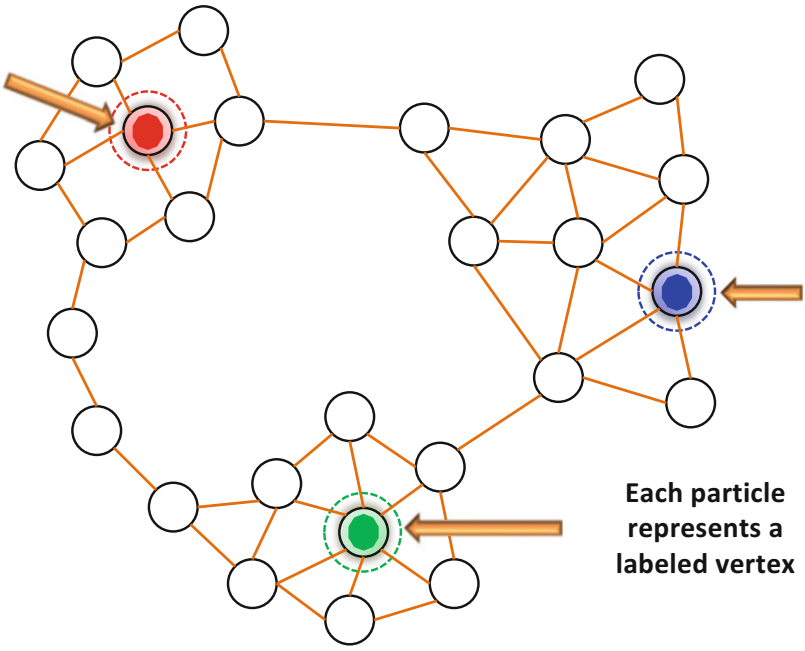

Fig. 10.1 Illustration of the initial conditions and the reanimation procedure of the semi-supervised model with multiple particles

This modification forces the model to work in a local label-spreading behavior, since each particle is now probabilistically bounded within a small region of the network. As a consequence, due to the competitive mechanism, each particle only visits a portion of vertices potentially having the same label as the particle's home labeled vertex. This concept can be roughly conceived as a "divide-and-conquer" effect embedded into the competitive scheme.

Recall that particles only have exploratory behavior due to the random movement rule when $\lambda = 0$. Conversely, they only present defensive characteristics due to the preferential movement rule when $\lambda = 1$. In addition, we have seen that a mixture of these two walking policies promotes better results for the model. Due to the deterministic behavior of the reanimation procedure, particles get probabilistically confined in regions potentially centered at their corresponding home vertices. The width of these regions is determined by the counterweighting factor λ. Figure 10.2 provides an intuitive schematic of this concept. Note that, as λ decreases, the larger are the regions that particles can potentially visit in that we are giving more importance to the exploratory in detriment to the defensive behavior. In this way, particles' dominated territories are expected to collide more often as λ decreases.

One interesting feature of the semi-supervised version is of the emergence of potential cooperation among particles. Frequently, we may have more than one labeled vertex from the same class. As a consequence, more than one particle may represent the same class or label of their respective home vertices. In this case, these

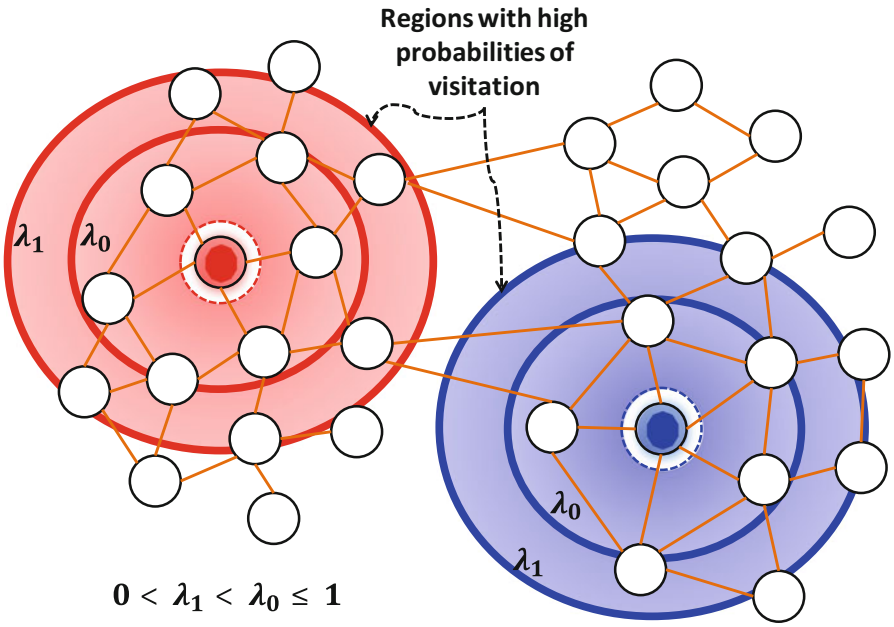

Fig. 10.2 Influence of the counterweighting factor λ in shaping regions with high probabilities of visitation

representative particles act together as a team because they propagate the same label to unlabeled vertices. In this way, several teams can potentially compete with each other to establish their class borders, while cooperating with their teammates.

10.2.2 Familiarizing with the Semi-Supervised Environment

In the semi-supervised learning scheme, denote $\mathscr{Y}$ as the set of possible discrete classes that can be predicted by the semi-supervised classifier. A set of data items $\mathscr{X} = \{x_1, \ldots, x_L, x_{L+1}, \ldots, x_{L+U}\}$ is supplied, in which each entry is a P-dimensional attribute vector of the form $x_i = (x_{i1}, \ldots, x_{iP})$. The first L data items are initially labeled and compose the labeled set $\mathscr{L}$. The remainder U data items are the unlabeled instances and comprise the unlabeled set $\mathscr{U}$. Note that $\mathscr{X} = \mathscr{L} \cup \mathscr{U}$. For each $x_i \in \mathscr{L}$, a label $y_i \in \mathscr{Y}$ is given. In contrast, no labels are supplied for data items in $\mathscr{U}$. The objective is to propagate the labels from $\mathscr{L}$ to $\mathscr{U}$, while preserving the data distributions. In practice, the proportion of unlabeled data items far surpasses the proportion of labeled data items, such that $U \gg L$ is observed in several practical occasions.

Since the particle competitive-cooperative model is a network-based technique, a network formation technique is employed to transform the vector-based data into a network. For this end, the data are mapped into a graph $\mathscr{G}$ using a network formation technique $g : \mathscr{X} \mapsto \mathscr{G} = \langle \mathscr{V}, \mathscr{E} \rangle$, in which $\mathscr{V} = \mathscr{L} \cup \mathscr{U}$ is the set of vertices and $\mathscr{E}$ is the set of edges. There are $V = |\mathscr{V}|$ vertices in the graph. Each vertex $v \in \mathscr{V}$ in the network corresponds to a data item $x \in \mathscr{X}$, so that $V = L + U$. Essentially, each vertex in $\mathscr{V}$ represents a data item in $\mathscr{X}$. The edges in $\mathscr{E}$ are created using a suitable network formation process, such as those explored in Chap. 4.

10.2.3 Deriving the Modified Competitive Transition Matrix

In this section, we focus on the technical differences of the unsupervised and semi-supervised learning transition matrices. If any part of the method has not been expressly indicated here, then it means that it is identical to the unsupervised transition matrix derived in Sect. 9.2.2.

The transition matrix assumes the same functional form as that in the unsupervised version, which, for convenience, we remember as follows:

$$\mathbf{P}^{(k)}_{\text{transition}}(t) \triangleq (1 - S^{(k)}(t)) \left[\lambda \mathbf{P}^{(k)}_{\text{pref}}(t) + (1 - \lambda) \mathbf{P}^{(k)}_{\text{rand}} \right] + S^{(k)}(t) \mathbf{P}^{(k)}_{\text{rean}}(t). \tag{10.1}$$

Basically, the technical differences are reflected on how each of these matrices comprising the transition matrix are defined. Specifically, the random and preferential terms do not suffer any modifications. The reanimation matrix, however, is adapted on account of two reasons:

- We must model the label diffusion process from labeled to unlabeled vertices in a local label-spreading behavior. For that, we cannot randomly transport particles from one place to another in the network as we would be creating labeling decisions that are non-smooth.
- We must now comport the idea of the existence of external information in the form of labels, which in turn are represented by labeled vertices. The idea is to use these labeled vertices, here termed as home vertices, as the proper destination for exhausted particles. As these labeled vertices are static in the network, we effectively force labeling decisions that are smooth.

Recall that each entry of $\mathbf{P}^{(k)}_{\text{rean}}(t)$ indicates the probability of bringing an exhausted particle $k \in \mathscr{K}$ back to its dominated territory. Here, we always transport the particle back to its home vertex, which is the vertex that particle k represents. Suppose that particle k is visiting vertex i when its energy becomes completely depleted. In this particular occasion, we transport the particle back in accordance with the following distribution:

$$\mathbf{P}^{(k)}_{\text{rean}}(i,j,t) \triangleq \begin{cases} 1, & \text{if } j = v_k \\ 0, & \text{otherwise} \end{cases}, \tag{10.2}$$

in which v_k indicates the home vertex of particle k, Therefore, matrix $\mathbf{P}_{\text{rean}}(t)$ only has non-zero entries for reallocations of particles to their respective home vertices. In computational terms, this can greatly enhance the process of deciding the next vertex that particle k will visit. For didactic purposes, Fig. 10.3 portrays a simple scenario of a reanimation taking place. In this case, the red or dark gray particle has its energy penalized, since it is visiting a vertex dominated by a rival particle. Supposing that its energy has been completely depleted, that particle becomes exhausted. Under these circumstances, the reanimation procedure of that particle is enabled, which will compel it to travel back to its home vertex so as to be properly recharged. Even though this is a simple mechanism, it can greatly increase the performance of the particle competitive-cooperative model, because it does not let particles go wander very far from their origions. Thus, the algorithm forces smoothness in the label diffusion process.

10.2.4 Modified Initial Conditions of the System

Recall that the internal dynamical state of system ϕ is composed of four terms: $p(t)$ is a random vector denoting the particles' locations at time t; $N(t)$ represents the number of visits that each of the vertices received up to time t; $E(t)$ indicates the

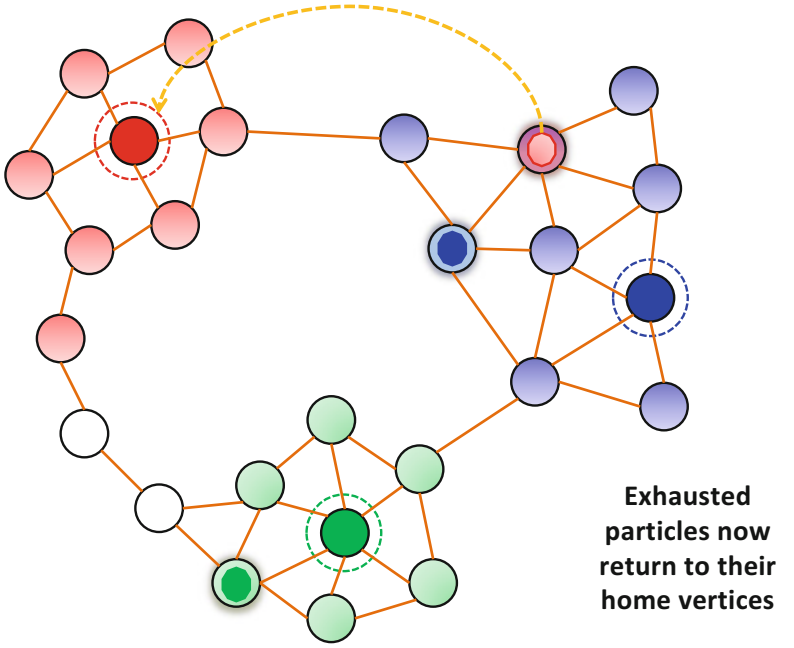

Fig. 10.3 The modified reanimation procedure performed by an exhausted particle. The surrounded vertices denote labeled instances. The color intensity in each vertex denotes the color of the particle imposing the highest domination level

particles' energy levels at time t; and $S(t)$ displays the particles' states at time t. In order to run the dynamical system ϕ, we need a set of initial conditions. In this way, we discuss how to fix the initial conditions for these four dynamical variables at $t = 0$.

For the initial particles' location $p(0)$, each particle is put at its corresponding home vertex.

We now discuss how to initialize matrix $\mathbf{N}(0)$. For those initially labeled vertices, we fix a permanent ownership to their representative particles as follows. Since the ownership is represented by the maximum actual domination level imposed on that vertex, we simply force the number of visits of the representative particle to its home vertex to be infinity at the beginning of the learning process. Thus, changes in the ownership of labeled vertices become impossible. In view of this scheme, we have that each entry of $\mathbf{N}(0)$ is now given by:

$$\mathbf{N}_i^{(k)}(0) = \begin{cases} \infty, & \text{if particle } k \text{ represents vertex } i \\ 1 + \mathbb{1}_{[p^{(k)}(0)=i]}, & \text{otherwise} \end{cases}, \tag{10.3}$$

in which we apply (10.3) to every $(i, k) \in \mathscr{V} \times \mathscr{K}$. Note that the scalar 1 is used in the second expression of (10.3) so that unlabeled and unvisited vertices at time

$t = 0$ have their calculation well-defined, according to (9.7). Usually, more than one particle (a team) is generated to represent a set of pre-labeled examples of the same class. Each of them tries to dominate vertices independently. The cooperation among the particles of the same team happens only at the end of the process. In order to do so, for each vertex, we sum up the domination levels of all of the particles of the same team on it to obtain the aggregated domination level.

With respect to the initial particles' energy levels $E(0)$ and states $S(0)$, we maintain the configurations of the unsupervised learning model according to (9.25) and (9.26).

10.3 Theoretical Analysis of the Model

In this section, a theoretical analysis of the model is discussed. A numerical validation of the theoretical results is supplied. It is worth noting that only the main analytical differences in relation to theoretical analysis previously conducted on Sect. 9.3 (unsupervised version) are provided.

10.3.1 Mathematical Analysis

Since the dynamical system of the unsupervised and semi-supervised version are virtually the same, differing only on the initial distributions of the particle locations, the transition probability function is the same. In view of this, we rewrite it for convenience matters as follows:

$$\begin{aligned} P(\mathbf{X}(t+1) \mid \mathbf{X}(t)) &= \mathbb{1}_{[\mathbf{N}(t+1)=\mathbf{N}(t)+\mathbf{Q}_N(p(t+1))]} \\ &\quad \times \mathbb{1}_{[S(t+1)=Q_S(E(t+1))]} \\ &\quad \times \mathbb{1}_{[E(t+1)=E(t)+\Delta Q_E(p(t+1),\mathbf{N}(t+1))]} \\ &\quad \times \mathbf{P}_{\text{transition}}(\mathbf{N}(t), p(t)) \\ &= \mathbb{1}_{[\text{Compliance}(t)]}\mathbf{P}_{\text{transition}}(\mathbf{N}(t), p(t)). \end{aligned} \tag{10.4}$$

Following the reasoning applied in the analytical analysis of the unsupervised dynamical system, we are required to set feasible upper and lower limits for each of the random variables. The limits for $p(t), E(t), S(t)$ derived in the unsupervised version are integrally valid for the semi-supervised learning version.

Having in mind these considerations, we now derive these limits for the random variable $\mathbf{N}(t)$. The new initialization step indicated in (10.3) takes into account both labeled and unlabeled vertices, which invalidates Lemma 9.1 that only previews the existence of unlabeled vertices. Given this fact, we reformulate the aforementioned Lemma in the following.

Lemma 10.1. *The maximum reachable value of* $\mathbf{N}_i^{(k)}(t)$, $\forall (i,k) \in \mathscr{V} \times \mathscr{K}$, $t \in \mathbb{N}$, *is:*

- *If* $i \in \mathscr{U}$*:*

$$\mathbf{N}_{i_{\max}}^{(k)}(t) = \begin{cases} \left\lceil \frac{t+1}{2} \right\rceil + 1, & \text{if } t > 0 \text{ and } a_{ii} = 0 \\ t+2, & \text{if } t > 0 \text{ and } a_{ii} > 0 \end{cases}. \tag{10.5}$$

- *If* $i \in \mathscr{L}$*:*

$$\mathbf{N}_{i_{\max}}^{(k)}(t) = \infty. \tag{10.6}$$

in which $a_{ii} = 0$ if there are no self-loops starting at vertex i, and $a_{ii} > 0$ otherwise.

Proof. With regard to unlabeled vertices, i.e., which belong to $\mathscr{U}$, the proof supplied in Lemma 9.1 can be invoked *ipsis litteris*. This is valid because the movement policy of each particle remains the same in relation to the original unsupervised learning version.

With regard to labeled vertices, i.e., which belong to $\mathscr{L}$, this quantity can be inferred in a straightforward manner through the initial conditions of the system. According to (10.3), if i is a pre-labeled vertex and k is its representative particle, then $\mathbf{N}_i^{(k)}(t) = \infty, \forall t \geq 0$. ■

Since the upper and lower limits of the random variable $\mathbf{N}(t)$ have changed, the analysis of $\bar{\mathbf{N}}(t)$ needs some adjustments. Similarly to the previous case, Lemma 9.3 presented in the original version of the algorithm may only be applied to unlabeled vertices. In view of this, we reformulate this Lemma as follows:

Lemma 10.2. *The following assertions hold* $\forall (i,k) \in \mathscr{V} \times \mathscr{K}$*:*

- *If* $i \in \mathscr{U}$*:*

  (a) The minimum value of $\bar{\mathbf{N}}_i^{(k)}(t)$ *is:*

$$\bar{\mathbf{N}}_{i_{\min}}^{(k)}(t) = \frac{1}{1 + \sum_{u \in \mathscr{K} \setminus \{k\}} \mathbf{N}_{i_{\max}}^{(u)}(t)}. \tag{10.7}$$

  (b) The maximum value of $\bar{\mathbf{N}}_i^{(k)}(t)$ *is:*

$$\bar{\mathbf{N}}_{i_{\max}}^{(k)}(t) = \frac{\mathbf{N}_{i_{\max}}^{(k)}(t)}{\mathbf{N}_{i_{\max}}^{(k)}(t) + (K-1)}. \tag{10.8}$$

- *If* $i \in \mathscr{L}$*:*

  (a) The minimum value of $\bar{\mathbf{N}}_i^{(k)}(t)$ *is:*

$$\bar{\mathbf{N}}_{i_{\min}}^{(k)}(t) = 0. \tag{10.9}$$

(b) The maximum value of $\bar{\mathbf{N}}_i^{(k)}(t)$ *is:*

$$\bar{\mathbf{N}}_{i_{\max}}^{(k)}(t) = 1. \tag{10.10}$$

Proof. With regard to unlabeled vertices, the proof supplied in Lemma 9.3 can be invoked *ipsis litteris*.

With regard to labeled vertices, Eq. (10.9) can be reached as follows: consider that particle k is not a representative from the labeled vertex i. However, by the initial conditions shown in (10.3), since vertex i is labeled by hypothesis, $\exists k' \in \mathscr{K} : \mathbf{N}_i^{(k')}(t) = \infty, \forall t \geq 0$. Now, as k does not represent i, via (10.3) again, we know that $\mathbf{N}_i^{(k)}(t)$ may only take on finite values $\forall t \geq 0$. Finally, applying (9.7) using this setup yields (10.9). Equation (10.10) can be achieved as follows: consider that particle k now represents the labeled vertex i. In this case, $\mathbf{N}_i^{(k)}(t) = \infty$. Considering that a labeled vertex may only be represented by one kind of particle, then all the remaining entries of $\mathbf{N}_i^{(k')}(t), k' \in \mathscr{K}, k' \neq k$ are finite. Using (9.7) under these circumstances, we arrive at (10.10). ■

The final step before calculating the marginal distribution of the vertices' domination levels, i.e., $P(\bar{\mathbf{N}}(t))$, is to find all the possible irreducible fractions that an arbitrary entry of $\bar{\mathbf{N}}(t)$ can assume. Lemma 9.4 fails to provide us with enough information about the labeled vertices, since it only delimits the irreducible fractions for unlabeled vertices. Next, a reformulation of such Lemma is provided.

Lemma 10.3. *Denote* $^{\text{num}}/_{\text{den}}$ *as an arbitrary irreducible fraction. Consider that the set* $\mathscr{I}_t$ *retains all the reachable values of* $\bar{\mathbf{N}}_i^{(k)}(t)$, $\forall (i,k) \in \mathscr{V} \times \mathscr{K}$, *for a fixed* t. *Then, the elements of* $\mathscr{I}_t$ *are composed of all elements satisfying the following constraints:*

(i) With regard to unlabeled vertices:

- *(a) The minimum element is given by the expression in (9.60).*
- *(b) The maximum element is given by the expression in (9.61).*
- *(c) All the irreducible fractions within the interval delimited by (a) and (b) such that:*
  - *I.* $\text{num}, \text{den} \in \mathbb{N}^*$.
  - *II.* $\text{num} \leq \mathbf{N}_{i_{\max}}^{(k)}(t)$.
  - *III.* $\text{den} \leq \sum_{u \in \mathscr{K}} \mathbf{N}_{i_{\max}}^{(u)}(t)$.

(ii) With regard to labeled vertices:

- *(a)* 0, *if particle* k *does not represent vertex* i.
- *(b)* 1, *if particle* k *represents vertex* i.

Proof. Regarding item (i): Straightforward from Lemma 9.4.

Regarding item (ii): (a) As vertex i is labeled, $\exists u \in \mathscr{K} : \mathbf{N}_i^{(u)}(t) = \infty$. In view of (9.7) and (10.3), we obtain $\bar{\mathbf{N}}_i^{(k)}(t) = 0$; (b) Similarly, using (9.7) and (10.3), we get $\bar{\mathbf{N}}_i^{(k)}(t) = 1$. ■

Finally, the expression for calculating the domination matrix distribution remains the same as the one derived in the unsupervised version of the algorithm. In the following, we reiterate it for convenience:

$$P\big(\bar{\mathbf{N}}(t) = \mathbf{U} : \mathbf{U} \in \mathscr{M}_t\big) = \sum_{u=1}^{t} P\big(f(u\mathbf{N}(t)) = \mathbf{U}\big). \tag{10.11}$$

As $t \to \infty$, $P(\bar{\mathbf{N}}(t))$ provides enough information for classifying the unlabeled vertices. In this case, they are labeled according to the team of particles that is imposing the highest domination level. Since the domination level is a stochastic variable, the output of this model is fuzzy.

10.3.2 A Numerical Example

In this section, we show how the theoretical results derived in the previous section can be employed in a simple example. We limit the demonstration for a single iteration, namely for transition from $t = 0$ to $t = 1$. Consider the exemplificative network as a trivial 3-vertex regular network, identical to that in Fig. 9.10a. Consider that vertex 1 has been labeled as pertaining to class 1 and vertex 2, to class 2, i.e., $\mathscr{V} = \{1, 2, 3\}$, $\mathscr{L} = \{1, 2\}$, and $\mathscr{U} = \{3\}$. Clearly, we can see that the unlabeled vertex 3 possesses overlapping characteristics in relation to classes 1 and 2. We theoretically show this behavior through this illustrative example. Consider the arbitrary initial settings: we insert $K = 2$ particles into the network, i.e., $\mathscr{K} = \{1, 2\}$. Let the particle 1 represent vertex 1 and particle 2, vertex 2. In this setup, particle 1 propagates the label of vertex 1 and particle 2 diffuses labels of vertex 2. Suppose also that we have a certainty about the locations of the particles at $t = 0$, which satisfy the following distribution:

$$P\left(\mathbf{N}(0) = \begin{bmatrix} \infty & 1 \\ 1 & \infty \\ 1 & 1 \end{bmatrix}, p(0) = [1 \;\; 2], E(0), S(0)\right) = 1, \tag{10.12}$$

i.e., there is 100 % (certainty) that the particles 1 and 2 are generated at vertices 1 and 2, respectively. Observe that $\mathbf{N}(0)$, $E(0)$, and S(0) are chosen such as to satisfy (10.3), (9.25), and (9.26), respectively; otherwise the probability should be 0, in view of (10.4).

From Fig. 9.10a, we can deduce the adjacency matrix $\mathbf{A}$ of the graph and, therefore, determine the transition matrix associated to the random movement term for a single particle. Applying (9.2), we get:

$$\mathbf{P}_{\text{rand}} = \begin{bmatrix} 0 & 0.50 & 0.50 \\ 0.50 & 0 & 0.50 \\ 0.50 & 0.50 & 0 \end{bmatrix}. \tag{10.13}$$

Given $\mathbf{N}(0)$, we can readily establish $\bar{\mathbf{N}}(0)$ with the aid of (9.7):

$$\bar{\mathbf{N}}(0) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0.50 & 0.50 \end{bmatrix}. \tag{10.14}$$

Using (9.8) we are able to calculate the matrices associated to the preferential movement policy for each particle in the network:

$$\mathbf{P}^{(1)}_{\text{pref}}(0) = \begin{bmatrix} 0 & 0 & 1 \\ 0.67 & 0 & 0.33 \\ 1 & 0 & 0 \end{bmatrix}, \tag{10.15}$$

$$\mathbf{P}^{(2)}_{\text{pref}}(0) = \begin{bmatrix} 0 & 0.67 & 0.33 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}. \tag{10.16}$$

To simplify calculations, let us assume $\lambda = 1$, so that (10.1) reduces to $\mathbf{P}_{\text{transition}}(0) = \mathbf{P}^{(1)}_{\text{pref}}(0) \otimes \mathbf{P}^{(2)}_{\text{pref}}(0)$ at time 0, which is a matrix with dimensions 9×9. Instead of building this matrix, we make use of Remark 9.1 to build the next particles localization vector $p(1)$ with the collection of two matrices 3×3, as given in (10.15) and (10.16). Note that, in the special case when $\lambda = 1$, the preferential movement matrix is the transition matrix itself, provided that all the particles are active, which indeed are at time 0, according to (9.26). For the first particle, one can see from (10.15) that, starting from vertex 1 (row 1), there can only be one next possible localization for particle 1, namely vertex 3. For the second particle, starting from the vertex 2 (row 2), one can state that the next localization of particle 2 can only be vertex 3, too. With that in mind, we have that:

$$P\left(\mathbf{N}(1) = \begin{bmatrix} \infty & 1 \\ 1 & \infty \\ 2 & 2 \end{bmatrix}, p(1) = [3\ 3], E(1), S(1) \mid \mathbf{X}(0)\right) = 1, \tag{10.17}$$

in which $\mathbf{X}(0)$ is given by (10.12). Furthermore, as we have fixed $\lambda = 1$, it is expected that the transition will be heavily dependent on the domination levels of the neighborhood vertices. Therefore, given that the labeled vertices constitute strong

repulsive forces that act against rival particles, the preferential or defensive behavior of these particles will never adventure in these type of vertices. This provides a natural explanation for the reason that the state $p(1) = [3\ 3]$ is the only possible next particles localization vector.

Before doing the calculation of the marginal distribution $P(\mathbf{N}(1))$, we are required to find an upper limit for an arbitrary entry of a specific unlabeled vertex of the matrix $\mathbf{N}(1)$. This is readily evaluated from (10.5), which results in $\mathbf{N}_{i_{\max}}^{(j)}(1) = 2$, $\forall i \in \mathscr{V}$, implying that we are only needed to take all numerical combinations of the matrix $\mathbf{N}(0)$ such that each entry may only take the values $\{1, 2\}$, since larger values would yield probability 0 according to Lemma 10.1. Moreover, we need to iterate through every feasible value of every entry of $E(0)$ and $E(1)$. In order to do so, we fix $\Delta = 0.25$, $\omega_{\min} = 0$, and $\omega_{\max} = 1$. With that, we are able to make use of Lemma 9.2, which yields $E(t) \in \{0, 0.25, 0.5, 0.75, 1\}$. The limits of the remaining system variables $S(0)$ and $S(1)$ are straightforward. In the present conditions, we have enough information to calculate the marginal distribution $P(\mathbf{N}(1))$, according to (9.57):

$$P\left(\mathbf{N}(1) = \begin{bmatrix} \infty & 1 \\ 1 & \infty \\ 2 & 2 \end{bmatrix}\right) = 1 \times 1 = 1. \tag{10.18}$$

As the last goal, the task is to determine the distribution $P(\bar{\mathbf{N}}(1))$. According to the specified steps in the previous section, we need to find all irreducible fractions that lie within the interval $[0, 1]$ with the constraints derived in the previous section. This means that we only have to consider entries of matrix $\bar{\mathbf{N}}(t)$ that contain elements of $\mathscr{I}_t$; the remainder $\bar{\mathbf{N}}(t)$ are infeasible and, thus, occur with probability 0. In view of the constraints previously enumerated, $\mathscr{I}_t = \{0, 1/4, 1/3, 1/2, 2/3, 3/4, 1\}$. It is worth commenting that the labeled vertices (vertices 1 and 2) can only assume the values $\{0, 1\} \subset \mathscr{I}_t$, as we have previously stated. Observing that we have the complete distribution of $\mathbf{N}(1)$, it is an easy task to apply (9.64), as follows:

$$P\left(\bar{\mathbf{N}}(1) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0.5 & 0.5 \end{bmatrix}\right) = 1. \tag{10.19}$$

It is noteworthy to reinforce that the mapping between the probabilities of $\mathbf{N}(t)$ and $\bar{\mathbf{N}}(t)$ is not bijective: in this special simple case that we are studying, we did not have distinct $\mathbf{N}(t)$ that could generate $\bar{\mathbf{N}}(t)$, but as t increases, this is likely to happen quite frequently. This process is repeated for a sufficiently large t or until the system converges to a quasi-stationary state $\bar{\mathbf{N}}(t)$. A detailed look at the system's behavior that we have derived suggests that (10.19) holds for every $t \geq 1$, and particles 1 and 2 will visit vertex 3 with period 2. Hence, this shows the overlapping nature of vertex 3, as it can be naturally stated only by the topological structure of the graph. Ideally, for networks with presence of distinct classes, $\bar{\mathbf{N}}(t)$ varies as we iterate

the dynamical system. We can then check the most probable classification of each of the vertices by looking at the domination levels of those particles with highest probability $P\left(\tilde{\mathbf{N}}\right)$ in the corresponding rows of each vertex.

For example, in Ref. [11], a simulation is performed to illustrate that the theoretical results really approximate the empirical behavior of the stochastic competitive model for a large number of independent runs of the algorithm. In addition, extensive numerical analyses are conducted to show the good performance of the particle competitive-cooperative model.

10.4 Numerical Analysis of the Model

In this section, we present simulation results in order to show the effectiveness of the semi-supervised particle competitive-cooperative model.

10.4.1 Simulation on a Synthetic Data Set

Here, we investigate the performance of the algorithm when applied to a network consisted of $V = 15$ vertices split into 3 unbalanced communities, as depicted in Fig. 10.4. $K = 3$ particles are inserted into the network at the initial positions $p(0) = [2\ 8\ 15]$, meaning the first particle (representing the red or "circle" class) starts at vertex 2, the second particle (representing the blue or "square" class) starts at vertex 8, and the third particle (representing the green or "triangle" class) starts at vertex 15. All the remaining vertices in the network are initially unlabeled (in the

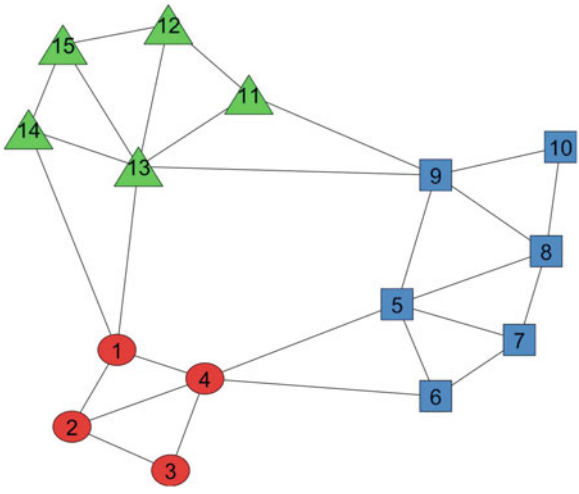

Fig. 10.4 A simple networked data set. The *red* or "*circle*" class is composed by vertices 1–4, the *blue* or "*square*" class comprises the vertices 5–10, and the *green* or "*triangle*" class encompasses the vertices 11–15. Initially, only the vertices 2 (*red* or "*circle*" particle), 8 (*blue* or "*square*" particle), and 15 (*green* or "*triangle*" particle) are labeled

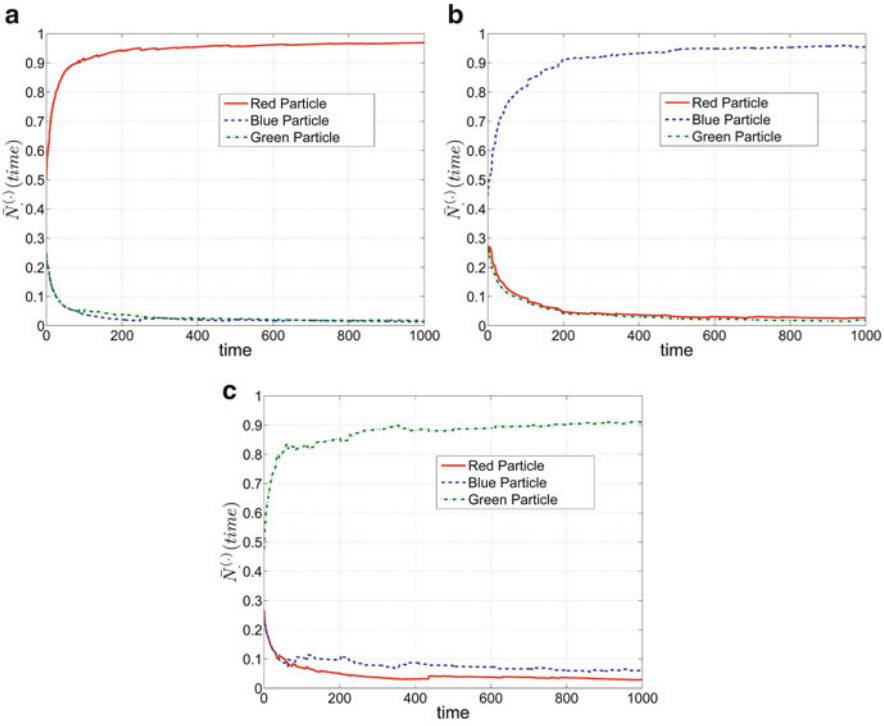

Fig. 10.5 Evolutional behavior of the average domination level imposed by the 3 particles on the existing classes in the network. (**a**) Red or "circle" class (vertices 1–4). (**b**) Blue or "square" class (vertices 5–10). (**c**) Green or "triangle" class (vertices 11–15)

figure, they are colored for the sake of easily identifying the classes). The competitive system is iterated until $t = 1,000$ and the predicted label for each of the unlabeled vertices is given by the particle's label that is imposing the highest domination level. Figure 10.5a–c show the evolutional behavior of the domination levels imposed by the three particles on the red or "circle" class, the blue or "square" class, and the green or "triangle" class, respectively. Specifically, from Fig. 10.5a, we can verify that red or "circle" particle dominates vertices 1–4 (red or "circle" class), due to the fact that the average domination level on these vertices approaches 1, whereas the average domination levels of the other two rival particles decay to 0. Considering Fig. 10.5b, c, we can use the same logic to confirm that the blue or "square" particle completely dominates the vertices 5–10 (blue or "square" class) and the green or "triangle" particle dominates vertices 11–15 (green or "triangle" class).

Table 10.1 Brief meta-information of the UCI data sets

Data set	# Instances	# Dimensions	# Classes
Heart	303	75	2
Heart-statlog	270	13	2
Ionosphere	351	34	2
Vehicle	946	18	4
House-votes	435	16	2
Wdbc	569	32	2
Clean1	476	168	2
Isolet	7797	617	26
Breastw	569	32	2
Australian	690	14	2
Diabetes	768	8	2
German	1000	20	2
Optdigits	5620	64	10
Sat	6435	36	7

10.4.2 Simulations on Real-World Data Sets

In this section, computer simulations on real-world data sets gathered from the UCI Machine Learning Repository are presented. A brief meta-information of the 14 real-world data sets that are going to be studied here is supplied in Table 10.1. For a detailed description, refer to [7].

With respect to the experimental setup, for each data set, 10 examples are randomly selected to compose the labeled set and the labels of the remainder of the examples are dropped. The labeled set is formed in such a way that each class has at least a labeled example. It is worth observing that in a semi-supervised learning task, the quantity of labeled examples is often too few to afford a valid cross validation, and therefore hold-out tests are usually used for the evaluation process. Also, for the purpose of comparing the performance of different algorithms, we use two competitive semi-supervised data classification techniques: Transductive SVM (TSVM) and low density separation (LDS). For a description and the default parameters used, one can refer to [5].

Since the particle competitive-cooperative model relies on a networked environment, we need to construct a graph from the vector-based data sets shown in Table 10.1. For this end, we use the k-nearest neighbor technique with $k = 3$. We fix $\lambda = 0.6$ and $\Delta = 0.07$, which respects the parameter sensitivity analysis in Sect. 9.2.6.

The simulation results are reported in Table 10.2. In this table, the average test error and the corresponding standard deviation achieved by each algorithm are supplied. We can note that the semi-supervised algorithm based on multiple particles achieves better results most of the time.

The average rank attained by each of the algorithms is also provided. We estimate the average rank as follows: (i) for each data set, the algorithms are ranked according

Table 10.2 Test errors (%) with 10 labeled training points and the corresponding average rank and standard deviation of each technique

Data set	TSVM	LDS	Proposed technique
Heart	27.4 ± 10.4	22.9 ± 9.6	$\mathbf{21.3 \pm 9.9}$
Heart-statlog	26.1 ± 5.9	21.7 ± 6.1	$\mathbf{20.5 \pm 5.4}$
Ionosphere	$\mathbf{23.9 \pm 8.2}$	24.1 ± 10.9	24.7 ± 9.0
Vehicle	36.8 ± 7.8	33.7 ± 8.5	$\mathbf{31.8 \pm 8.8}$
House-votes	16.0 ± 5.3	11.6 ± 4.0	$\mathbf{11.4 \pm 3.7}$
Wdbc	$\mathbf{11.1 \pm 3.7}$	15.0 ± 8.7	11.9 ± 5.1
Clean1	46.7 ± 4.8	43.2 ± 3.7	$\mathbf{40.2 \pm 2.9}$
Isolet	13.3 ± 9.5	$\mathbf{8.0 \pm 11.4}$	12.7 ± 8.8
Breastw	11.1 ± 8.8	$\mathbf{9.6 \pm 7.6}$	10.5 ± 9.4
Australian	$\mathbf{31.4 \pm 11.4}$	34.0 ± 14.5	31.6 ± 12.2
Diabetes	34.2 ± 4.6	33.8 ± 4.8	$\mathbf{32.1 \pm 4.6}$
German	36.5 ± 5.1	$\mathbf{35.3 \pm 4.2}$	35.9 ± 4.3
Optdigits	8.6 ± 7.6	$\mathbf{3.6 \pm 11.1}$	5.4 ± 8.9
Sat	13.5 ± 10.8	$\mathbf{5.8 \pm 14.2}$	9.3 ± 10.1
Average rank	2.6	1.9	1.6

The experiments are repeated 30 times for each labeled set and the average test error and standard deviations are recorded. The smallest test errors for each data set are in bold

to their average performance, i.e., the best algorithm (the smallest test error) is ranked as 1st, the second best one is ranked as 2nd, and so on; and (ii) for each algorithm, the average rank is given by the average value of its ranks scored on all the data sets.

With the purpose of examining these simulation results in a statistical manner, the procedure outlined in [6] is adopted. The methodology described therein uses the calculated average rank of each algorithm for the statistical inference. Specifically, the Friedman Test is used to check whether the measured ranks are significantly distinct from the mean value of the ranks. In this case, the mean value of the ranks is 2, since there are 3 algorithms. The null-hypothesis considered here is that all the algorithms are equivalent, so their ranks should be the same. Here, we fix a significance level of 0.05. For our experiments, according to [6], we have that $N = 14$ and $k = 3$, resulting in a critical value given by $F(2, 26) \approx 3.37$, in which the two arguments are derived from the degrees of freedom defined as $k - 1$ and $(N - 1)(k - 1)$, respectively. In our case, we get a value $F_F \approx 5.32$ that is higher than the critical value, so the null-hypothesis is rejected at a 5 % significance level.

As the null hypothesis is rejected, one can advance to post-hoc tests which aim at verifying the performance of the proposed algorithm in relation to others. For this task, we opt to use the Bonferroni-Dunn Test, for which the control algorithm is fixed as the proposed technique. According to [6], one should not make pairwise comparisons when we test whether a specific method is better than others. Basically, the Bonferroni-Dunn Test quantifies whether the performance between an arbitrary algorithm and the reference is significantly different. This is done by verifying

whether the corresponding average ranks of these algorithms differ by at least a critical difference (CD). If they do differ that much, then it is said that the better ranked algorithm is statistically superior to the worse ranked one. Otherwise, they do not present a significant difference for the problem at hands. Thus, if we perform the evaluation of the CD for our problem, we encounter CD $\approx$ 0.8. The average rank of the proposed method is 1.6. By virtue of that, if any rank does lie in the interval 1.6 ± 0.8, the control algorithm and the compared algorithms are statistically equivalent. We conclude that our algorithm is superior to Transductive SVM for the simulations performed on these data sets. However, the comparison of the LDS to the control algorithm does not surpass the CD, meaning that the differences among them are statistically insignificant.

10.5 Application: Detection and Prevention of Error Propagation in Imperfect Learning

In this section, we tackle the problem of learning with imperfect data, i.e., some of the labeled samples are incorrectly labeled, via the competitive model described in the previous section.

Section 10.5.1 motivates the importance and real practicability of detecting and preventing error propagation in semi-supervised tasks in imperfect data training data. Section 10.5.2 enhances the particle competitive-cooperative model to deal with imperfect learning by providing a detection mechanism of possible wrong labeled instances. Section 10.5.3 shows a mechanism to prevent error propagation by flipping labels from those vertices detected as possibly wrong labeled by the detection module. Section 10.5.4 formally presents the modified particle competitive-cooperative model to withstand imperfect training data. Section 10.5.5 investigates the sensitivity of the model's parameters related to detecting and preventing error propagation. Finally, Sect. 10.5.6 tests the error detection and prevention mechanisms on synthetic and real-world data.

10.5.1 Motivation

The quality of the training data is a fundamental issue in machine learning. It becomes more critical in semi-supervised learning, because fewer labeled data are available and errors (wrong labels) may easily propagate to a portion of or to the entire data set. Up to now, there are still few works devoted to studying semi-supervised learning from imperfect data [1, 2, 8]. Usually, in machine learning, the input label information of the training data set is supposed to be completely reliable. Intuitively, this is not always true and mislabeled samples are commonly found in the data sets due to instrumental errors, corruption from noise, or even

human mistakes in the labeling process. If these kinds of wrong labels are used to further classify new data (in the supervised learning case) or are propagated to the unlabeled data (in the semi-supervised learning), severe consequences may occur. Therefore, designing mechanisms to prevent error propagation is important in the machine learning area. Specifically, the prevention of error propagation can benefit the learning systems from two complementary aspects:

i. Improvement of the performance of the learning system, permitting the system to learn from errors;
ii. Avoidance of a system's catastrophe by limiting the spread of wrong labels from imperfect training data.

In the next section, a mechanism for preventing error propagation embedded in the particle competition-cooperation model is presented [4, 10].

10.5.2 Detecting Imperfect Training Data

The idea of mislabeled vertex identification is described in the following. In the competition-cooperation model, for each labeled vertex, a representative particle is generated. For simplicity, we here use the term *correctly labeled particle* to denote the representative particle of a correctly labeled vertex and the *mislabeled particle* to represent the representative particle of a mislabeled vertex. In this way, the vertices in the vicinity of mislabeled vertices are expected to be in constant competition among the correctly labeled and the mislabeled particles. Therefore, the mislabeled particles will be stranded in the small region centered at the mislabeled vertex. Since the number of mislabeled particles in each region is generally much smaller than the number of correctly labeled ones, the surrounding region of the mislabeled vertex tends to be heavily dominated by the correctly labeled particle team. By virtue of the combination of random and preferential walking rules, particles will eventually try to venture far away from their home vertices. Once a mislabeled particle goes far away from its home vertex, it has a high probability to get exhausted. Hence, the number of times that a particle becomes exhausted is a good indicator of whether or not the home vertex that it represents is mislabeled. If the associated particle is constantly getting exhausted, it is possibly representing an imperfect labeled vertex. Otherwise, it is probably representing a correctly labeled vertex.

In order to detect possible mislabeled vertices, consider the random vector $D(t) = [D^{(1)}(t), \ldots, D^{(K)}(t)]$, in which the k-th entry, $D^{(k)}(t)$, stores the number of times that particle k has become exhausted up to time t. In view of this, the update rule of each entry of $D(t)$ is expressed by:

$$D^{(k)}(t) = D^{(k)}(t-1) + S^{(k)}(t), \tag{10.20}$$

in which $S^{(k)}(t)$ is the boolean-valued variable that indicates whether particle k is active or exhausted at time t. In brief, it yields 1 if particle k is exhausted at time t

and 0, otherwise. On account of that, Eq. (10.20) simply adds 1 or remains with the same summation, depending on the state of particle k at the current time.

In order to check whether or not a particle is getting exhausted more times than the others, we use the average number of times that particles get exhausted as a statistical descriptor/threshold. The average number of times that particles get exhausted in the network is expressed by:

$$\langle D(t) \rangle = \frac{1}{K} \sum_{u=1}^{K} D^{(u)}(t). \tag{10.21}$$

In view of the random variable introduced in (10.21), any particle $k \in \mathscr{K}$ such that:

$$D^{(k)}(t) \geq (1 + \alpha) \langle D(t) \rangle \tag{10.22}$$

holds is considered to be getting exhausted more times than the other particles. Therefore, such particle is a great candidate of representing an incorrectly labeled or imperfect home vertex. The parameter $\alpha \in [-1, \infty)$ is a confidence value that indicates the percentage above the average value $\langle D(t) \rangle$ that must occur in order to a particle to be conceived as a representative of an incorrectly labeled vertex. A small α tends to classify more vertices as incorrectly labeled ones than a large value. In the extreme case, when $\alpha \to \infty$, then the model reduces to its original form, i.e., it does not detect nor prevent incorrectly labeled vertices from propagating incorrect labels.

At the beginning of the competitive process, a very small portion of the unlabeled data is expected to be dominated by the particles. In this way, a correctly labeled particle may accidentally get exhausted and (10.22) turns out to be true. In an attempt to prevent this false positive, a weighted function in (10.22) is introduced, which penalizes (10.22) at the beginning of the competition process and eliminates the effect of that penalty when t is large. In this way, the weighted version of (10.22) becomes:

$$(1 - e^{-\frac{t}{\tau}}) D^{(k)}(t) \geq (1 + \alpha) \langle D(t) \rangle, \tag{10.23}$$

in which $\tau \in (0, \infty)$ is the time constant of the exponential decaying function.

Next, we analyze the minimum number of times that a particle must get exhausted in order to (10.23) to hold. For this end, we plot the minimum $D^{(k)}(t)$, denoted as $D^{(k)}_{\min}(t)$, in a such a way that (10.23) holds. Mathematically, it satisfies the following expression:

$$D^{(k)}_{\min}(t) = \frac{(1 + \alpha) \langle D(t) \rangle}{(1 - e^{-\frac{t}{\tau}})}. \tag{10.24}$$

For the sake of clarity, let us suppose that the dynamical process produces $\langle D(t)\rangle = 1, \forall t \geq 0$. Fix $\alpha = 0$ for simplicity. With respect to the non-weighted version, even when t is very small, a $D^{(k)}(t) = 1$ is sufficient to (10.22) to be satisfied. Therefore, if a correctly labeled vertex happens to reach the "exhausted" state at the beginning of the competitive process, it is immediately marked as a possible wrongly labeled vertex. On the other hand, the weighted version penalizes $D^{(k)}(t)$ when t is small, as (10.23) reveals. Therefore, it is unlikely that labeled vertices are classified as wrongly labeled for small values of t. However, for a sufficient large t, this penalization ceases and (10.23) asymptotically approximates (10.22). In particular, when $t \to \infty$, one has:

$$\begin{aligned} \lim_{t\to\infty}(1-e^{-\frac{t}{\tau}})D^{(k)}(t) \geq \lim_{t\to\infty}(1+\alpha)\langle D(t)\rangle \Rightarrow \\ D^{(k)}(\infty)\lim_{t\to\infty}(1-e^{-\frac{t}{\tau}}) \geq (1+\alpha)\langle D(\infty)\rangle \Rightarrow \\ D^{(k)}(\infty) \geq (1+\alpha)\langle D(\infty)\rangle, \end{aligned} \tag{10.25}$$

i.e., (10.23) reduces to (10.22). Finally, parameter τ is used to control the decaying speed of the exponential function. Looking in isolation to this function, a small τ yields a large negative derivative for this function and a large τ produces a small negative derivative for this function. In other words, the speed of decaying increases as τ decreases.

10.5.3 Preventing Label Propagation from Imperfect Training Data

In the previous section, we have presented a method for detecting possible wrongly labeled vertices by means of using the information generated by the competitive process itself. Now, whenever we detect a possible imperfect labeled vertex, we need to take actions in order to prevent it from propagating wrong labels throughout its neighborhood.

In an imperfect labeled vertex, the associated particle is expected to constantly getting exhausted, as it is probably in a region with several other labeled data items from rival teams of particles. As such, the neighborhood of the imperfect labeled vertex receives a large quantity of visits by other particles, in such a way that competition is always taking place. In order to correct for the imperfectness of the training labeled data, a natural approach therefore is to drop the label from that home vertex and reset it accordingly to the most dominant class in the neighborhood. By using a local approach, we maintain the smoothness assumption of the model. Figure 10.6 portrays a schematic of this relabeling process that prevents error propagation in the model. Note that the neighborhood is mostly dominated by the red class in such a way that the home vertex has its label flipped to the red

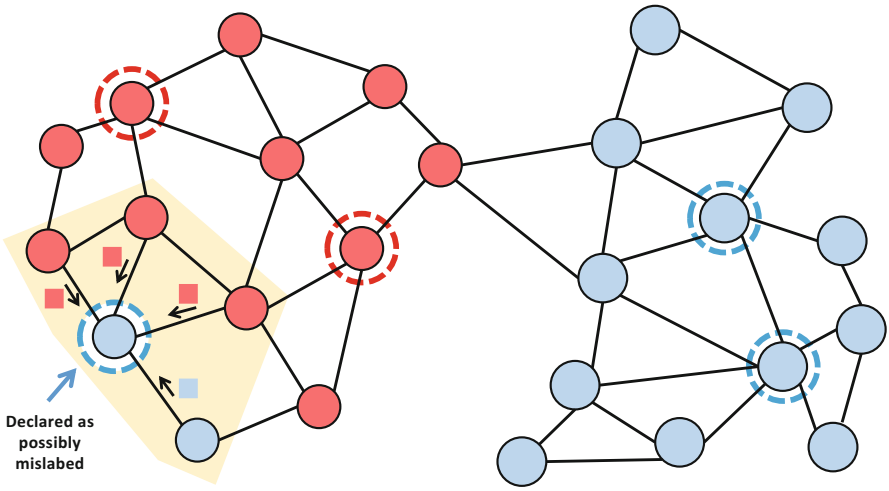

Fig. 10.6 Schematic of the mechanism of error propagation prevention. Surrounded vertices denote home vertices, which correspond to the labeled data. The *left*-most *blue* labeled item is declared as possibly mislabeled. In this case, the label is reset accordingly to the most dominant class in the neighborhood

instead of the original blue class. In this way, the training labeled data gets reshaped such as to have smoother properties. Note that we never use the information of the possibly imperfect labeled vertex, as the detection module warns that its label may be incorrect.

We now formalize that idea. Suppose vertex i has been considered as a possible imperfect labeled vertex at time t, meaning that (10.23) holds. In view of this, vertex i is going to have its random vector $\mathbf{N}_i(t)$ altered, in such a way to reflect how the neighborhood is being dominated at time t. We simply restart $\mathbf{N}_i(t)$ as the average number of visits received by its neighbors. Mathematically, for all $k \in \mathscr{K}$, we have:

$$\begin{aligned} \mathbf{N}_i^{(k)}(t) &= \frac{1}{|\mathscr{N}(i)|} \sum_{j \in \mathscr{V}} \mathbf{A}_{ij} \mathbf{N}_j^{(k)}(t) \\ &= \frac{1}{|\mathscr{N}(i)|} \sum_{j \in \mathscr{N}(i)} \mathbf{N}_j^{(k)}(t), \end{aligned} \tag{10.26}$$

in which $\mathscr{N}(i)$ is the set of neighbors of the imperfect labeled vertex i and $|\mathscr{N}(i)|$ is the corresponding number of neighbors. This idea can be further extended to encompass not only the direct neighborhood but also indirect levels of neighborhoods of the home vertex. In this chapter, we use the simple approach of relabeling according to the immediate neighborhood, which is the most conservative option in terms of smoothness in the labeling decision.

One can observe that the difference between the modified model and the original model is that the former is capable of relabeling labeled vertices, while the latter is not. This new feature is processed when (10.26) is applied.

10.5.4 Definition of the Modified Learning System to Withstand Imperfect Data

With the mechanism introduced before, the dynamical system of the original particle competitive-cooperative model presented in Sect. 10.2 is modified as follows:

$$\mathbf{X}(t) = \begin{bmatrix} p(t) \\ \mathbf{N}(t) \\ E(t) \\ S(t) \\ D(t) \end{bmatrix}. \tag{10.27}$$

If $\text{wrong(k, t)} = (1 - e^{-\frac{t}{\tau}})D^{(k)}(t) \geq (1 + \alpha)\langle D(t)\rangle$, then the new competition-cooperation system that supports detection and prevention of incorrectly labeled vertices is given by:

$$\phi : \begin{cases} p^{(k)}(t+1) = j, \quad j \sim \mathbf{P}^{(k)}_{\text{transition}}(t) \\ \mathbf{N}_i^{(k)}(t+1) = \mathbb{1}_{[\text{wrong}(k,t)]}\left[\frac{1}{|\mathcal{N}(i)|}\sum_{j\in\mathcal{N}(i)}\mathbf{N}_j^{(k)}(t)\right] \\ \qquad\qquad + \mathbb{1}_{[\neg\text{wrong}(k,t)]}\left[\mathbf{N}_i^{(k)}(t) + \mathbb{1}_{[p^{(k)}(t+1)=i]}\right] \\ E^{(k)}(t+1) = \begin{cases} \min(\omega_{\max}, E^{(k)}(t) + \Delta), \text{ if owner}(k,t) \\ \max(\omega_{\min}, E^{(k)}(t) - \Delta), \text{ if } \neg\text{owner}(k,t) \end{cases} \\ S^{(k)}(t+1) = \mathbb{1}_{[E^{(k)}(t+1)=\omega_{\min}]} \\ D^{(k)}(t+1) = D^{(k)}(t) + S^{(k)}(t+1) \end{cases} \tag{10.28}$$

We see that the update rule related to the number of visits (2nd expression) now consists of two terms: the term which is employed for vertices that have been detected to be wrongly labeled (first term) and the term for vertices that are not considered wrongly labeled (second term).

10.5.5 Parameter Sensitivity Analysis

Likewise we have performed for the original particle competitive-cooperative model in Sect. 9.2.6, here we focus on understanding the roles of parameters α and τ that deal with the error detection and prevention mechanisms in imperfect data

environments. Following the same setup in Sect. 9.2.6, we also use the benchmark of Lancichinatti et al. [9] with $V = 5{,}000$ vertices, a network connectivity of $\bar{k} = 8$, and intercommunity mixture of $\mu = 0.3$. Essentially, the benchmark process consists in varying the mixing parameter μ and evaluating the resulting model's accuracy rate.

To test for robustness against imperfect data, the benchmark of Lancichinatti et al. is altered [12]. Once the networks are generated, we label a fraction of the vertices using a stratified uniform distribution to compose the labeled training data. Now, we purposely flip some labels of the labeled data so as to introduce noise or imperfectness. In the simulations, the labeled set is fixed as 10 % of the size of the data set. Finally, we deliberately flip 30 % of the correct labels to incorrect labels $q = 0.3$ in a stratified manner, so as to maintain the proportion of labeled samples at each class.

10.5.5.1 Impact of α

Parameter α is employed in the module of detecting imperfect labeled data. In essence, it determines how much $D^{(k)}(t), k \in \mathscr{K}$, must deviate from $\langle D(t) \rangle$ in order to the corresponding labeled vertex to be declared as mislabeled. Observe that, when $\alpha = -1$, the detection process always accuses labeled vertices as mislabeled, because, in accordance with (10.24), one has:

$$D^{(k)}(t) \geq \lim_{\alpha \to -1} \frac{(1+\alpha)\langle D(t) \rangle}{(1 - e^{-\frac{t}{\tau}})}$$

$$\implies \quad D^{(k)}(t) \geq 0. \tag{10.29}$$

Taking into account that the domain of $D^{(k)}(t)$ is the interval $[0, \infty)$, then (10.29) is always satisfied. Therefore, (10.26) is applied to every vertex in the network for any $t \geq 0$, i.e., the detection procedure reduces to this simple strategy of locally verifying the label validity. Now, when α assumes larger values, the competition dynamics are incorporated to the detection scheme and nonlinear interactions are taken into account in the detection scheme. As α is increased, one can see that solution space of (10.24) becomes more distant from the origin, meaning that (10.24) is more difficult to be satisfied. On the extreme case, when $\alpha \to \infty$, one has that:

$$D^{(k)}(t) \geq \lim_{\alpha \to \infty} \frac{(1+\alpha)\langle D(t) \rangle}{(1 - e^{-\frac{t}{\tau}})} \implies \tag{10.30}$$

$$D^{(k)}(t) \geq \infty, \tag{10.31}$$

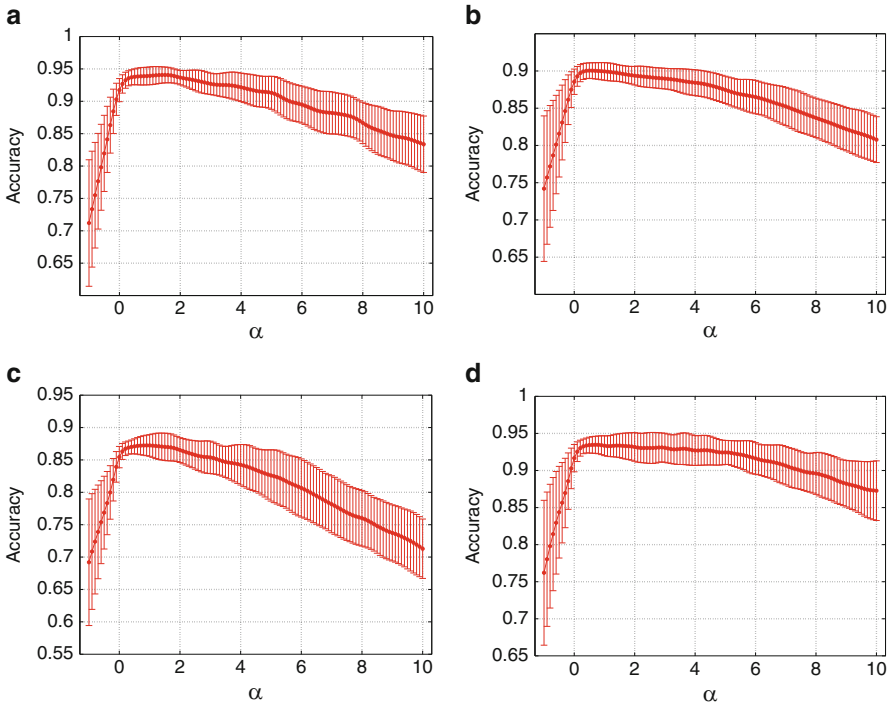

Fig. 10.7 Accuracy rate vs. α. We fix $\tau = 40$. Taking into account the steep peek that is verified and the large negative derivatives that surround it, one can see that the parameter α is sensible to the overall model's performance. Results are averaged over 30 simulations. Reproduced from [12] with permission from Elsevier. (**a**) $\gamma = 2$ and $\beta = 1$. (**b**) $\gamma = 2$ and $\beta = 2$. (**c**) $\gamma = 3$ and $\beta = 1$. (**d**) $\gamma = 3$ and $\beta = 2$

that is only satisfied when $t \to \infty$, which is empirically unattainable. Hence, (10.31) never holds for a finite t. As a result, the detection scheme is virtually turned off. In other terms, the particle competition algorithm reduces to its original form proposed in Sect. 10.2.

Taking into account that analysis, Fig. 10.7a, d display how the accuracy rate of the model behaves as we vary α from -1 to 10 in the networks constructed using the methodology described in [9] with different values of γ and β. As one can verify from the figures, this parameter is sensitive to the outcome of the technique. Oftentimes, the optimal accuracy rate is achieved when a mixture of random and preferential walks occurs. Using a conservative approach, for $0 \leq \alpha \leq 3$, the model gives decent accuracy results when applied to networks with communities.

10.5.5.2 Impact of τ

Parameter τ is employed in detection module of the error propagation mechanism. It assumes the range $\tau \in (0, \infty)$ and is responsible for adjusting the speed of the penalizing function so as to prevent vertices from being signalized as wrongly labeled ones at the beginning of the stochastic process. Next, we study the behavior of the algorithm for small and large τ in a theoretical and empirical manner.

When parameter τ assumes small values, the exponential decaying function has large-valued derivatives, meaning that its decaying speed is faster compared to larger values of τ. In the extreme case, i.e., $\tau \to 0$, one has:

$$D^{(k)}(t) \geq \lim_{\tau \to 0} \frac{(1+\alpha)\langle D(t)\rangle}{(1-e^{-\frac{t}{\tau}})} \Longrightarrow \tag{10.32}$$

$$D^{(k)}(t) \geq (1+\alpha)\langle D(t)\rangle, \tag{10.33}$$

in which we have used the fact that $\lim_{\tau \to 0}(1-e^{-\frac{t}{\tau}}) = 1$, since $e^{-\frac{t}{\tau}} \to 0$ provided that $\tau \to 0$ and t is finite. This shows that the model's behavior is dictated by the value of α, which has been studied in the previous section. This means that, in this special case, the penalizing function ceases to exist, because it decays so fast to be considered relevant in the learning process.

When τ assumes larger values, then the decaying speed of the exponential function reduces accordingly. By virtue of that, we have that:

$$D^{(k)}(t) \geq \lim_{\tau \to \infty} \frac{(1+\alpha)\langle D(t)\rangle}{(1-e^{-\frac{t}{\tau}})} \Longrightarrow \tag{10.34}$$

$$D^{(k)}(t) \geq \frac{(1+\alpha)\langle D(t)\rangle}{(1-\lim_{\tau \to \infty} e^{-\frac{t}{\tau}})} \Longrightarrow \tag{10.35}$$

$$D^{(k)}(t) \geq \infty. \tag{10.36}$$

In this case, the denominator of (10.36) approaches 0 in a quick manner, since $e^{-\frac{t}{\tau}} \to 1$ provided that $\tau \to \infty$ and t is finite. This reveals that the detection scheme is turned off, since there is no reachable solution for (10.36) when t remains finite. Therefore, in this case, the model reduces to its original form.

Bearing in mind these considerations, Figs. 10.8a, d portray the accuracy rate attained by the algorithm for distinct values of τ. We can conclude that, for intermediate values of τ, namely $30 \leq \tau \leq 60$, the model is not sensitive to τ when applied to networks with communities.

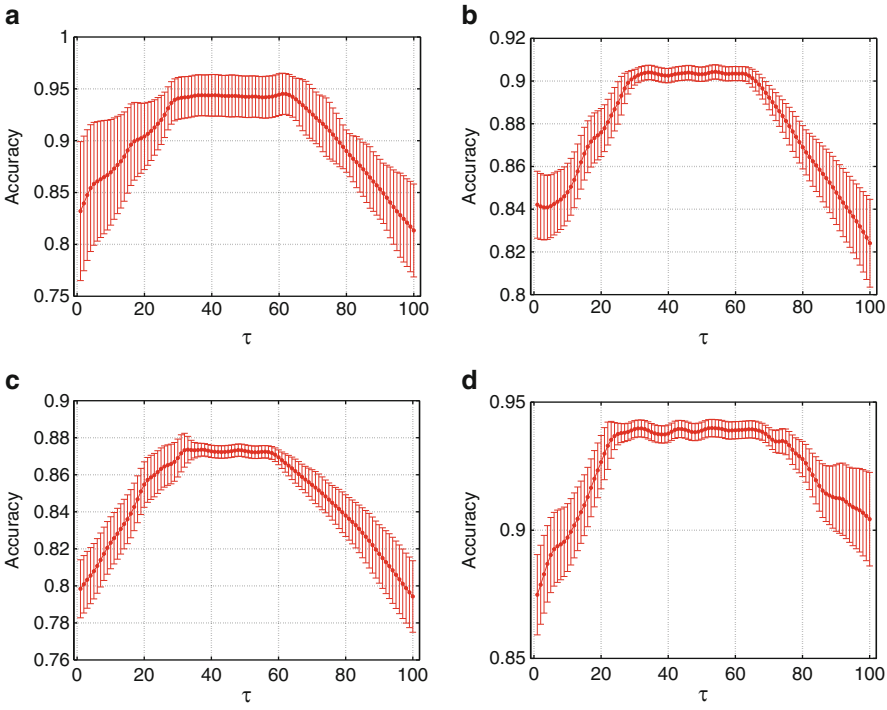

Fig. 10.8 Accuracy rate vs. τ. We fix $\alpha = 0$. Taking into account the large steady region that is verified, one can see that the parameter τ is not very sensible to the overall model's performance if one correctly uses it. Results are averaged over 30 simulations. Reproduced from [12] with permission from Elsevier. (**a**) $\gamma = 2$ and $\beta = 1$. (**b**) $\gamma = 2$ and $\beta = 2$. (**c**) $\gamma = 3$ and $\beta = 1$. (**d**) $\gamma = 3$ and $\beta = 2$

10.5.6 Computer Simulations

In this section, computer simulations are performed to show the robustness of the particle competition-cooperation model for semi-supervised learning in an error-prone environment. We use synthetic and real-world data sets to check the model's performance.

10.5.6.1 Synthetic Data Sets

We first show the behavior of the particle competition algorithm on artificial networks using the Girvan-Newman's benchmark, which we discussed in Sect. 6.2.4. Figure 10.9 depicts the model's accuracy rate as a function of the proportion of mislabeled instances or imperfect data q for three different sizes of the labeled set.

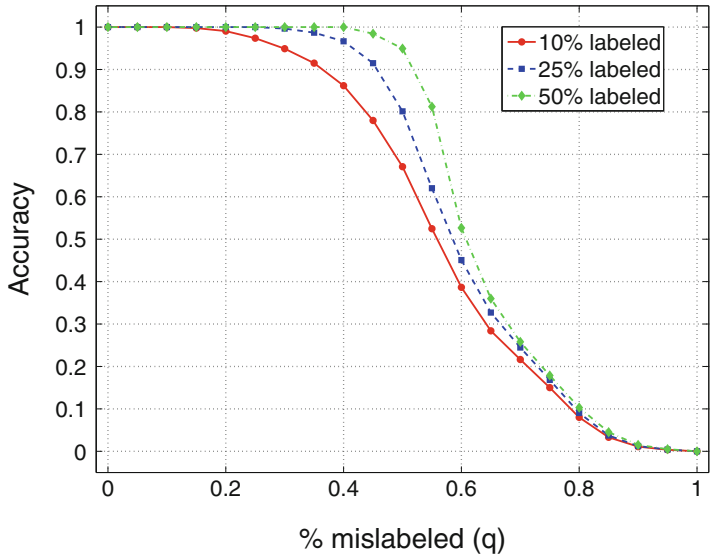

Fig. 10.9 Behavior of the model's accuracy rate of the model vs. the proportion of mislabeled instances or imperfect data q, when random clustered networks with constant mixture are used. The networks are generated using $V = 10,000$ vertices, $M = 16$ communities, and a intercommunity mixture of $z_{out}/\langle k \rangle = 0.3$. 100 independent runs are performed and the average value is reported. Reproduced from [12] with permission from Elsevier

From this figure, we can observe three different regions that are separated by two critical points of interest:

- When q is small, the effect of mislabeled vertices on the accuracy rate of the algorithm is minimal. Visually, this is translated by the plateau region in the plot depicted in Fig. 10.9 with basis near the 100 % accuracy rate. This behavior can be explained by the competition that happens in the dynamical system as it evolves in time. Since the majority of the labeled samples is correctly labeled, the propagation of the correctly labeled samples literally overwhelms that of mislabeled samples. As a consequence, the model's performance is slightly altered.
- When q's value is neither small nor large, the accuracy rate decreases. The first critical point indicates that the label propagation originated from imperfect labeled vertices start to overwhelm that of correctly labeled data. From the same figure, we see that such a phenomenon is also dependent on the size of the labeled set. As the labeled set gets bigger, the algorithm becomes more robust in error-prone environments.
- When q is large, the accuracy rate enters a new steady state with low accuracy rate values that start from the second critical point onwards. At this point, the propagation of wrongly labeled vertices by misrepresented particles completely overwhelms the particles that are spreading correct labels.

Table 10.3 Description of the semi-supervised data classification techniques used in the error propagation analysis

Abbreviation	Technique	Reference
LP	Linear propagation	Zhou et al. [14]
LNP	Linear neighborhood propagation	Wang and Zhang [13]
Original PCCM	Original particle competitive-cooperative method	Sect. 10.2
Modified PCCM	Modified particle competitive-cooperative method	Sect. 10.5

10.5.6.2 Real-World Data Sets

In this section, the algorithm is applied to real-world data sets in an imperfect labeled data environment. For comparison matters, a set of competing semi-supervised learning techniques is also employed, which is summarized in Table 10.3.

With respect to the model selection procedure, all of the parameters are tuned in accordance with the best accuracy rate reached by the algorithms. The model selection is conducted as follows:

- *LP*: σ is selected over the discretized interval $\sigma \in \{0, 1, \ldots, 100\}$ and α is fixed to $\alpha = 0.99$ (the same setup as [14]);
- *LNP*: k is evaluated over the discretized interval $k \in \{1, 2, \ldots, 100\}$, and σ, as well as α, are selected using the same process employed in the LP parameters selection;
- *Original and Modified PCCM*: the data are first transformed into a network representation using the k-nearest neighbor network formation technique. For this purpose, k is chosen over the discretized interval $k \in \{1, 2, \ldots, 10\}$. For the model's parameters, we test λ in the interval $\{0.20, 0.22, \ldots, 0.80\}$. We fix $\Delta = 0.1$. The selection of these parameters and the candidate range for λ are in accordance with the guidelines provided in the parameter sensitivity analysis supplied in Sect. 9.2.6.

Now we apply the LP, LNP, the original and modified PCCMs on two data sets from the UCI Machine Learning Repository [7]: Iris and Letter Recognition. The former is composed of three equal-sized classes, each of which comprising 50 samples, totalizing 150 samples. The latter is composed of 20,000 samples divided into 26 unbalanced classes, each representing a different letter of the English alphabet. Thus, the Letter Recognition data set can be considered as a large-scale data set.

Figure 10.10a, b show the behavior of the test error vs. the proportion of mislabeled samples q. One can verify that, as q grows, intuitively all of the algorithms' performances start to decline, producing larger test errors. However, the modified PCCM is able to outperform the compared algorithms, by virtue of the detection and prevention mechanisms embedded into the competitive model. In an error-prone environment, we can conceive the algorithm as having two types of competitions taking place simultaneously: competition of particles spreading

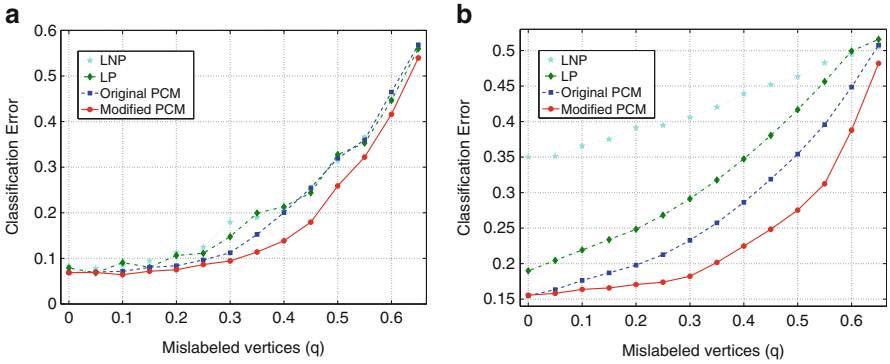

Fig. 10.10 Behavior of the test error as a function of the proportion of imperfect data on two real-world data sets. 100 independent runs are performed and the average value is reported. Reproduced from [12] with permission from Elsevier. (**a**) Iris data set. (**b**) Letter Recognition data set

correct and incorrect labels. Since the competition is always taking place indirectly,[1] then these two types of label diffusion processes are always in opposition. In practical situations, it is fair to assume that the number of correctly labeled samples is usually larger than that of incorrect labels, in such a way that the diffusion process represented by those particles that spread correct labels will eventually overwhelm or win the competition against the label propagation originated by the imperfect training data.

10.6 Chapter Remarks

This chapter presents a semi-supervised learning technique that uses mechanisms of competition and cooperation among particles, which runs in a networked environment. In this model, several particles, each of which representing a class, navigate in the network to explore their territory and, at the same time, attempt to defend their territory against rival particles. If several particles propagate the same class label, then a team is formed, and a cooperation process among these particles occurs.

Wrong label propagation is a fundamental question in machine learning because mislabeled samples are commonly found in the data sets due to several factors, such as instrumental errors, corruption from noise, or even human mistakes. In autonomous learning systems, errors are much easier to be propagated to the whole data set due to the absence or few external intervention, which makes the situation more critical.

[1]The indirect competition among particles occurs by the accumulated domination levels of each vertex and by the particles' movement policy.

To deal with that situation, in this chapter, a method is introduced for detecting and preventing error propagation embedded in the semi-supervised learning technique. The error detection mechanism is realized by weighting the total number of times a particle has become exhausted to a thresholded value, which is dependent and vary in time. When the dynamical competitive system begins, there is a penalizing factor which prevents the detection of false positives. This has been introduced in order to diminish the dependency of the error propagation model on the initial locations of the labeled samples (transient part of the dynamics). As the system evolves, this penalization ceases to exist and the plain domination level that each vertex has is used in the error propagation inference. Once a vertex is declared as mislabeled, the particle competition technique resets its domination levels as the average value of its neighborhood, so as to conform to the cluster and smoothness assumptions.

References

1. Amini, M.R., Gallinari, P.: Semi-supervised learning with explicit misclassification modeling. In: Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI), pp. 555–560. Morgan Kaufmann, San Francisco (2003)
2. Amini, M.R., Gallinari, P.: Semi-supervised learning with an imperfect supervisor. Knowl. Inf. Syst. **8**(4), 385–413 (2005)
3. Breve, F., Zhao, L., Quiles, M.G., Pedrycz, W., Liu, J.: Particles competition and cooperation in networks for semi-supervised learning. IEEE Trans. Data Knowl. Eng. **24**, 1686–1698 (2010)
4. Breve, F.A., Zhao, L., Quiles, M.G.: Particle competition and cooperation for semi-supervised learning with label noise. Neurocomputing **160**, 63–72 (2015)
5. Chapelle, O., Schölkopf, B., Zien, A. (eds.): Semi-supervised learning. Adaptive Computation and Machine Learning. MIT, Cambridge (2006)
6. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. **7**, 1–30 (2006)
7. Lichman, M.: UCI Machine Learning, Repository University of California, Irvine, School of Information and Computer Sciences, University of California, Irvine (2013)
8. Hartono, P., Hashimoto, S.: Learning from imperfect data. Appl. Soft Comput. **7**(1), 353–363 (2007)
9. Lancichinetti, A., Fortunato, S., Radicchi, F.: Benchmark graphs for testing community detection algorithms. Phys. Rev. E **78**(4), 046110(1–5) (2008)
10. Silva, T.C., Zhao, L.: Detecting error propagation via competitive learning. Procedia Comput. Sci. **13**, 37–42 (2012)
11. Silva, T.C., Zhao, L.: Network-based stochastic semisupervised learning. IEEE Trans. Neural Netw. Learn. Syst. **23**(3), 451–466 (2012)
12. Silva, T.C., Zhao, L.: Detecting and preventing error propagation via competitive learning. Neural Netw. **41**, 70–84 (2013)
13. Wang, F., Zhang, C.: Label propagation through linear neighborhoods. IEEE Trans. Knowl. Data Eng. **20**(1), 55–67 (2008)
14. Zhou, D., Bousquet, O., Lal, T.N., Weston, J., Schölkopf, B.: Learning with local and global consistency. In: Advances in Neural Information Processing Systems, vol. 16, pp. 321–328. MIT, Cambridge (2004)