Semi-supervised Network Alignment 6

6.1 Overview

As mentioned before, in the real-world online social networks, the anchor links are extremely difficult to label manually. The training set we can obtain is usually of a small size compared with the network scale, and most of the potential anchor links are unlabeled actually. For instance, given the Facebook and Twitter networks containing millions or billions of users, identifying a very small training set merely with hundreds of correct anchor links is however not an easy task. Therefore, it is not realistic to achieve a large set of labeled anchor links as required by the *supervised network alignment* models introduced in Chap. 4. On the other hand, completely ignoring the (small) set of labeled anchor links, just like the *unsupervised network alignment* models introduced in Chap. 5, may also create lots of problems, since these labeled anchor links can provide important signals for the network alignment model building. In this chapter, we will introduce another category of network alignment models based on the *semi-supervised learning* setting [8, 23], where both the (small) labeled and (large) unlabeled sets will be utilized in the model building process.

However, significantly different from the traditional semi-supervised learning problems, the anchor link instances studied in the network alignment problem are not independent. The *one-to-one* constraint on the anchor links actually limits the number of existing anchor links incident to each user node across networks, which can also introduce extra information for inferring the anchor links. For instance, given an identified anchor link $(u_i^{(1)}, u_j^{(2)})$ between networks $G^{(1)}$ and $G^{(2)}$, we can know that the remaining unlabeled anchor links incident to either $u_i^{(1)}$ or $u_j^{(2)}$ should not exist.

Given two heterogeneous online social networks $G^{(1)}$ and $G^{(2)}$, we can represent the small set of positively labeled anchor links and large number of unlabeled anchor links as $\mathcal{A}_{train}$ and $\mathcal{A}_{unlabeled} = \mathcal{U}^{(1)} \times \mathcal{U}^{(2)} \setminus \mathcal{A}_{train}$, respectively. The main objective of semi-supervised network alignment task is to build a model to infer the existence labels of these unlabeled anchor links with both sets $\mathcal{A}_{train}$ and $\mathcal{A}_{unlabeled}$. In our network alignment task, the test set is actually identical to the unlabeled set, i.e., $\mathcal{A}_{unlabeled} = \mathcal{A}_{test}$. The built model will be further applied to the test set to infer the potential labels of these anchor links.

In this chapter, we will focus on studying the network alignment problem based on the semi-supervised learning setting. This chapter will be organized as follows: At the very beginning, we will provide an introduction to the semi-supervised learning task, which is very different from the supervised and unsupervised learning tasks introduced before. With such a new learning setting, we

© Springer Nature Switzerland AG 2019
J. Zhang, P. S. Yu, *Broad Learning Through Fusions*,
https://doi.org/10.1007/978-3-030-12528-8_6

will introduce three different network alignment models based on the semi-supervised learning [8,23], active learning [16], and positive-unlabeled (PU) learning [12] settings, respectively.

6.2 Semi-supervised Learning: Overview

Semi-supervised learning [8, 23] is halfway between supervised and unsupervised learning. Besides the unlabeled data, the learning algorithms are also provided with some supervision information from a small set of labeled training data. In this section, we will provide a basic introduction to the classic semi-supervised learning task. In addition, the *semi-supervised learning* problem also has several special types, including *active learning* [16] and *positive-unlabeled learning* [12], which will be introduced in this section as well.

6.2.1 Semi-supervised Learning Problem Setting

Semi-supervised learning [8,23] is a new type of learning tasks which were not covered in the machine learning overview provided in Chap. 2. In semi-supervised learning tasks, besides the set of labeled training data instances $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_l, y_l)\}$, there also exists a large-sized unlabeled data instance set $\{\mathbf{x}_{l+1}, \mathbf{x}_{l+2}, \ldots, \mathbf{x}_{l+u}\}$. Semi-supervised learning tasks attempt to make use of this combined information from both the labeled and unlabeled sets to surpass the performance that could be obtained by either the supervised learning merely with the labeled instances or unsupervised learning merely with the unlabeled instances.

Existing *semi-supervised learning tasks* can be generally divided into two main categories, i.e., *transductive semi-supervised learning* [23] and *inductive semi-supervised learning* [23]. The *transductive semi-supervised learning* tasks aim at inferring the correct labels of instances in the unlabeled data set $\{\mathbf{x}_{l+1}, \mathbf{x}_{l+2}, \ldots, \mathbf{x}_{l+u}\}$, while the *inductive semi-supervised learning* tasks want to infer the correct mapping from the feature space to the label space instead (not just limited to the data instances in the unlabeled set).

By reading here, a question may naturally arise in the readers' mind: "Is semi-supervised learning useful?" To answer the question from the mathematical perspective, the "semi-supervised learning is useful" iff the knowledge obtained from the unlabeled instance feature vector distribution $P(\mathbf{x})$ is helpful for the inference of posterior probability $P(y|\mathbf{x})$. If this is not the case, semi-supervised learning will not yield any improvement over supervised learning (merely with the labeled set). Otherwise, the involvement of the unlabeled instances will lead to a great improvement in learning the probability function $P(y|\mathbf{x})$, i.e., semi-supervised learning will be useful.

To ensure the effectiveness of semi-supervised learning, certain assumptions need to hold, like the *smoothness assumption* [8], *cluster assumption* [8], and *manifold assumption* [8]. These assumptions will provide the way to use the unlabeled instances in improving the learned models with the labeled instances. Subject to these different assumptions, various *semi-supervised learning* models have been proposed already, some of which will be introduced in this part as well.

6.2.1.1 Smoothness Assumption

The *smoothness assumption* [8] is the most popular assumption used in semi-supervised learning tasks, which goes as follows:

"*Given the feature vectors of two instances,* $\mathbf{x}_1$ *and* $\mathbf{x}_2$*, if their feature vectors are close, so will be their corresponding labels.*"

The *smoothness assumption* implies that if two data instances are similar in their feature representations, they will be more likely to share common labels. Clearly, with such an assumption, we can generalize the finite training set by incorporating the unlabeled instances. The *smoothness assumption* can be applied in both semi-supervised classification and regression models.

6.2.1.2 Cluster Assumption

In the feature space, the data instances tend to form clusters, where the data instances with similar feature vectors tend to lie in the same cluster, while those which are different in feature representations will be partitioned into different clusters instead. In the semi-supervised learning, the *cluster assumption* [8] denotes

"*For the data instances in the same cluster, they are more likely to have similar labels.*"

Based on the *cluster assumption*, the unlabeled data instances can be used to help identify the boundaries of the clusters in a more accurate way. We could run a clustering algorithm and use the labeled instances to assign a class label to each cluster. In this way, depending on the belonging relationships of the unlabeled instances, we can determine the potential labels of these unlabeled instances. Here, we also want to clarify that the *cluster assumption* doesn't imply that each class will only form one single cluster, and it only denotes that the instances within the same cluster will have the same label. For the same class, the data instances are also possible to form multiple clusters.

6.2.1.3 Manifold Assumption

Another frequently used assumption in *semi-supervised learning* is called the *manifold assumption* [8]:

"*The high dimensional data instances lie on a lower-dimensional manifold.*"

In this case we can attempt to learn the manifold [8] using both the labeled and unlabeled data to avoid the *curse of dimensionality* [5]. The manifold assumption is practical when high-dimensional data are being generated by some processes that may be hard to model directly but only have a few degrees of freedom. If the data happen to lie on a low-dimensional manifold, however, then the learning algorithm can essentially operate in a space of the corresponding dimension, thus avoiding the curse of dimensionality.

6.2.2 Semi-supervised Learning Models

Several existing models can be adjusted to be applied in the semi-supervised learning tasks. In this part, we will introduce some existing semi-supervised learning models, which incorporate the unlabeled instances in the model training in different ways.

6.2.2.1 Semi-supervised Support Vector Machine (S3VM)

For the *smoothness assumption* introduced in the previous subsection, another revised version [4] is that

"*Given the feature vectors of two instances,* $\mathbf{x}_1$ *and* $\mathbf{x}_2$*, if the feature vectors are close in the high-density regions, so will be their corresponding labels.*"

Generally, in semi-supervised learning, the decision boundary of learning models is assumed to be situated in a low-density region (in terms of unlabeled data). Let notation $f(\mathbf{x}) \in \mathcal{Y}$ ($\mathcal{Y} = \{-1, +1\}$) denote the inferred label of the data instance with feature vector representation $\mathbf{x}$, where $f(\cdot)$ is the built model and $\mathcal{Y} = \{-1, +1\}$ is the binary label space. The decision boundary inferred by the model $f(\cdot)$ can be denoted as $f(\mathbf{x}) = 0$. Furthermore, the loss function on an unlabeled instance $\mathbf{x}$ can be represented as the following loss function:

$$l(f, \mathbf{x}) = \max(1 - |f(\mathbf{x})|, 0), \tag{6.1}$$

where the loss term $l(f, \mathbf{x}) > 0$ when $-1 < f(\mathbf{x}) < +1$; otherwise, it will be 0.

By considering all the unlabeled instances in set $\{\mathbf{x}_{l+1}, \mathbf{x}_{l+2}, \ldots, \mathbf{x}_{l+u}\}$, the average loss term of these instance will become

$$l(f, \{\mathbf{x}_{l+1}, \mathbf{x}_{l+2}, \ldots, \mathbf{x}_{l+u}\}) = \frac{1}{u} \sum_{i=l+1}^{l+u} \max(1 - |f(\mathbf{x}_i)|, 0). \tag{6.2}$$

Generally, the average loss term counts the violations in the margin separation, which will lead to a ranking score of potential mapping $f \in \mathcal{F}$. The top ranked mapping f denotes the one whose decision boundary avoids most unlabeled instances by a large margin.

If we use support vector machine (SVM) as the base model, i.e., mapping $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$, where $\mathbf{w}$ and b are the variables involved in the model. By adding this loss term with the objective function of support vector machine (SVM), we can represent the joint optimization objective function as follows:

$$\min_{\mathbf{w}, b} \frac{1}{l} \sum_{i=1}^{l} \max(1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b), 0) + c \,\|\mathbf{w}\|_2^2 + \lambda \cdot \frac{1}{u} \sum_{i=l+1}^{l+u} \max(1 - |\mathbf{w}^\top \mathbf{x}_i + b|, 0), \tag{6.3}$$

where the first term denotes the introduced loss on the labeled data instances, and $\|\mathbf{w}\|_2^2$ denotes the regularization on the model variables. The parameters c and λ are the weights of the last two terms, respectively.

By solving the objective function, the variables of the model can be learned. Actually, the loss term is non-convex, and the learning process of the objective function can be hard. Some existing algorithms, like *deterministic annealing* [15], *continuation method* [1], and *concave–convex procedure (CCCP)* [17], can be used to handle such a challenge. Formally, the above support vector machine model with the semi-supervised learning setting is also named as the S3VM model [4].

6.2.2.2 Semi-supervised Graph Based Model

Many of the semi-supervised learning models are based on graphs [24], where the data instances are represented as the nodes in the graph and the links denote the pairwise distance of the instances. For instance, given all the data instances, $\mathcal{V} = \mathcal{A}_{train} \cup \mathcal{A}_{unlabeled}$, we can represent it as a weighted graph $G = (\mathcal{V}, \mathcal{L}, w)$, where link set $\mathcal{L} \subset \mathcal{V} \times \mathcal{V} \setminus \{(u, u)\}_{u \in \mathcal{V}}$. The mapping function $w : \mathcal{L} \to \mathbb{R}$ denotes the weight of the links. For instance, given a link $e = (u, v) \in \mathcal{L}$, its weight $w(e)$ denotes how similar u and v are. If there exists no link between nodes u and v, then the weight of the potential link between them will be 0 instead.

The graph based semi-supervised learning models [24] can be viewed as estimating a projection function f to project the nodes in the graph to the label set. Generally, the projection function f needs to meet two requirements: (1) the inferred labels of the labeled nodes (i.e., data instances with labels) should be close to their true labels, and (2) it should be smooth on the whole graph. The first requirement can be viewed as minimizing the learning loss on labeled data, and the second requirement can be viewed as a regularization term about the model. So far, the existing various graph based semi-supervised models [24] mainly differ with each other in three aspects: (1) graph construction, (2) the loss function, and (3) the regularization term.

Graph construction is the key point of the graph based semi-supervised models, and it is still an open question to this context so far. Several different approaches have been proposed to construct the graphs already, which include:

- *Graph construction with domain knowledge*
- *Neighbor graph construction*
- *Graph construction with local fit*

More information about these graph construction approaches is provided in [24]. Besides graph construction, choosing different loss functions and regularization terms will lead to different semi-supervised models. We will introduce some of them as follows.

1. MinCut Model In the binary case, the classification of the data instances can be viewed as a cut problem to partition nodes in the graph into two disjoint subsets. We can treat the positive labels as the sources and the negative labels as the sinks. The objective of MinCut model [6] is to find a set of edges, removal of which can block all the flow between the source and sink nodes. In the cut result, nodes connected to the source nodes will be classified as the positive instances, and those connected to the sink nodes are classified as the negative instances.

Formally, the loss term introduced in the MinCut model can be represented as

$$loss = \sum_{i=1}^{l} (f(\mathbf{x}_i) - y_i)^2, \tag{6.4}$$

where $f(\mathbf{x}_i)$ denotes the inferred label for the data instance $\mathbf{x}_i$.

Meanwhile, the regularization term in the *MinCut model* can be represented as

$$reg = \frac{1}{2} \sum_{i,j=l+1, i \neq j}^{l+u} w(\mathbf{x}_i, \mathbf{x}_j) \cdot \left(f(\mathbf{x}_i) - f(\mathbf{x}_j)\right)^2. \tag{6.5}$$

The regularization terms can ensure the smoothness of the model. By minimizing the regularization term, the data instance pairs with closer representations, i.e., $w(\mathbf{x}_i, \mathbf{x}_j)$ is large, should have closer labels, i.e., $\left(f(\mathbf{x}_i) - f(\mathbf{x}_j)\right)^2$ will be small.

To ensure that the labeled instances are classified correctly, the loss term is usually assigned with a very large weight. The joint optimization function can be represented as

$$\begin{aligned} &\min \alpha \cdot \sum_{i=1}^{l} (f(\mathbf{x}_i) - y_i)^2 + \frac{1}{2} \sum_{i,j=l+1, i \neq j}^{l+u} w(\mathbf{x}_i, \mathbf{x}_j) \cdot (f(\mathbf{x}_i) - f(\mathbf{x}_j))^2 \\ &s.t. \;\; f(\mathbf{x}_i) \in \{+1, -1\}, \forall i \in \{1, 2, \ldots, l, l+1, \ldots, l+u\}, \end{aligned} \tag{6.6}$$

where α is assigned with a very large value, like $\alpha = \infty$. For simplicity, in the learning process, instead of learning the function $f(\cdot)$, we can treat $f(\mathbf{x}) = f_i$ as a variable instead, which takes values from the label space $\mathcal{Y} = \{+1, -1\}$. The learned variables will be outputted as the inferred labels for the data instances.

2. Local and Global Consistency Model The links in the weighted graph G can be represented as a weighted adjacency matrix $\mathbf{W} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$, where entry $W(i, j)$ denotes

$$W(i, j) = \begin{cases} w((u_i, u_j)), & \text{if } (u_i, u_j) \in \mathcal{L}, \\ 0, & \text{otherwise.} \end{cases} \tag{6.7}$$

Based on the weight matrix $\mathbf{W}$, we can define its corresponding normalized and unnormalized Laplacian matrix to be

$$\mathbf{L}_n = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}} \mathbf{W} \mathbf{D}^{-\frac{1}{2}}, \text{ and } \mathbf{L} = \mathbf{D} - \mathbf{W}, \text{ respectively,} \tag{6.8}$$

where the diagonal matrix $\mathbf{D}$ has value $D(i, i) = \sum_j W(i, j)$ on its diagonal.

The *local and global consistency* model [22] uses the following loss function:

$$loss = \sum_{i=1}^{l} (f_i - y_i)^2, \tag{6.9}$$

where f_i denotes the inferred label of the input data instance featured by vector $\mathbf{x}_i$. Meanwhile, the regularization function used in the *local and global consistency* model can be represented as

$$reg = \mathbf{f}^\top \mathbf{L}_n \mathbf{f}. \tag{6.10}$$

Besides the regularization term used above, many other regularization functions, like Tikhonov regularizer $\mathbf{f}^\top \mathbf{L} \mathbf{f}$ [2], or manifold regularizer $\|\mathbf{f}\|_K^2 + \|\mathbf{f}\|_I^2$ [3], can all be used to define the objective function of the semi-supervised learning models, which will lead to different learning performance. In the equations, K denotes a base kernel, where $\|\mathbf{f}\|_K^2$ denotes an "intrinsic norm" on $\mathbf{f}$ in the reproducing Kernel Hilbert space (RKHS), and $\|\mathbf{f}\|_I^2 = \frac{1}{(l+k)^2} \mathbf{f}^\top \mathbf{L} \mathbf{f}$.

6.2.2.3 Semi-supervised Generative Model

In the case that the base model used is a generative model, like

$$f_{\boldsymbol{\theta}}(\mathbf{x}) = \arg\max_y P(y|\mathbf{x}, \boldsymbol{\theta}) = \arg\max_y \frac{P(\mathbf{x}, y|\boldsymbol{\theta})}{\sum_{y'} P(\mathbf{x}, y'|\boldsymbol{\theta})}, \tag{6.11}$$

where $\boldsymbol{\theta}$ denotes the parameter vector involved in the generative model. The term $P(\mathbf{x}, y|\boldsymbol{\theta})$ denotes the joint probability of instance's feature vector and label in the generative model.

For the unlabeled instances, the likelihood for them to fit in the model can be represented as

$$L(f_{\boldsymbol{\theta}}, \{\mathbf{x}_{l+1}, \mathbf{x}_{l+2}, \ldots, \mathbf{x}_{l+k}\}) = \sum_{i=l+1}^{l+u} \log \left(\sum_{y \in \mathcal{Y}} P(\mathbf{x}_i, y|\boldsymbol{\theta}) \right). \tag{6.12}$$

To learn the model that can fit well for both the labeled instances and unlabeled instances, we can represent the objective function as follows:

$$\arg\max_{\boldsymbol{\theta}} \log\left(\sum_{i=1}^{l} P(y_i|\mathbf{x}_i, \boldsymbol{\theta})\right) + \lambda \cdot \sum_{i=l+1}^{l+u} \log\left(\sum_{y \in \mathcal{Y}} P(\mathbf{x}_i, y|\boldsymbol{\theta})\right). \tag{6.13}$$

The parameter $\boldsymbol{\theta}$ can be learned with the expectation–maximization (EM) algorithm [9] and some existing numerical optimization methods.

Besides these three models introduced in this section, there also exist many other types of *semi-supervised learning* models, like *semi-supervised co-training models* [7]. For the readers who are interested in *semi-supervised learning* works, please refer to [8, 23] for more information. These models introduced in this part can all be applied to solve the network alignment problem to infer the anchor links between different social networks. By utilizing the unknown anchor links, more accurate decision boundary can be determined with a small number of labeled instances. More information about the network alignment method based on semi-supervised learning setting is available in Sect. 6.3.

6.2.3 Active Learning

Active learning [16] is a special case of semi-supervised learning tasks, in which a learning algorithm is able to interactively query an oracle (denoting an information source) to obtain the desired labels of some unlabeled data instances. In the situation where manual labeling of the data instances is extremely hard, if the learning algorithm can actively query for the labels of some data instances, the learning process will be more efficient and effective. In active learning, the number of required labeled instances will be much smaller than the number of labels required by the normal supervised learning models. Meanwhile, in the data instance label query process, choosing the most informative instances [16] will be crucial for active learning, which can also reduce the number of required queries significantly to determine a good decision boundary.

Compared with the classical semi-supervised learning tasks, both active learning and semi-supervised learning tasks aim at obtaining a good learning performance without demanding too many labeled instances. Meanwhile, there also exist some differences in the way they work. Semi-supervised learning focuses more on using the unlabeled data instances to assist the learning models to improve the learning results. However, the objective of active learning is to choose one part of unlabeled data instances to query for their labels, which will be involved in the model training process as the known instances instead.

Formally, let $\mathcal{T} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_l, y_l)\}$ denote the set of labeled data instances, and $\mathcal{U} = \{\mathbf{x}_{l+1}, \mathbf{x}_{l+2}, \ldots, \mathbf{x}_{l+u}\}$ represent the set of unlabeled instances. Active learning aims at partitioning the unlabeled set $\mathcal{U}$ into two disjoint subsets $\mathcal{U}_Q$ and $\mathcal{U}_U$, and will query for the labels of the instances in set $\mathcal{U}_Q$. Therefore, the crucial task in active learning is to choose the data instances for set $\mathcal{U}_Q$ from the unlabeled data instance set.

Different query strategies [16] have been proposed already to determine the instances to be selected for set $\mathcal{U}_Q$, which include

- *Uncertainty Sampling*: The *uncertainty sampling* strategy will select the data instances that the current model is least certain about what the output should be. Many measures can be adopted to measure the prediction *uncertainty*, e.g., *posterior prediction probability* for probabilistic models, prediction result *entropy* for classification tasks, and the prediction *loss* for regression tasks.

- *Query by Committee*: In this approach, a variety of models are built with the current labeled data instances, which will vote on the output labels for the unlabeled data instances. For the data instances that these current models disagree the most, they will be selected finally by the *query by committee* strategy.
- *Expected Model Change*: Labeling some unknown data instances and adding them to the training set will lead to changes of the current model. The *expected model change* strategy aims at selecting the data instances which can introduce the maximum model changes. Different metrics can be used for measuring the expected model changes, like the introduced *gradient* by the new data instance in the model loss function.
- *Expected Error Reduction*: Labeling the data instances and retraining the models with these newly labeled instances may reduce the model's generalization error. The *expected error reduction* strategy will choose to label the data instances that can lead to the maximum expected error reduction instead.
- *Variance Reduction*: Minimizing the expectation of a loss function directly is expensive, and in general this cannot be done in closed form. However, we can still reduce generalization error indirectly by minimizing the output variance, which sometimes does have a closed-form solution. The *variance reduction* strategy focuses on selecting the data instances that can lead to the maximum variance reduction in the result.
- *Balance Exploration and Exploitation*: Labeling the unlabeled data instances is seen as a dilemma between the exploration and the exploitation over the data space representation. Such a strategy manages this compromise by modeling the active learning problem as a contextual bandit problem instead.
- *Exponentiated Gradient Exploration*: This strategy uses a sequential algorithm named exponentiated gradient (EG)-active that can improve any active learning algorithm by an optimal random exploration.

Example 6.1 In Fig. 6.1, we show an example of active learning with both labeled and unlabeled data instances. The complete data distribution is provided in plot (a), where the green dots and red triangles denote the data instances belonging to two different classes, respectively. Given a few labeled data instances in the feature space as shown in plot (b), we can fit a model with these labeled data instances, whose decision boundary is denoted as the purple line. Generally, for the data instances which are far away from the decision boundary, we can know that they are more likely to be either the positive or negative instances. Meanwhile, for those lying near the decision boundary, we are less certain about their specific labels and obtaining the true labels of these instances will help to determine more correct decision boundary. For instance, in plot (c), we further query for some labels nearby the decision boundary. By adding the new labeled data instance into the labeled training set, the new decision boundary is updated, which can not only classify the queried data instances but also the remaining unlabeled data instances as well.

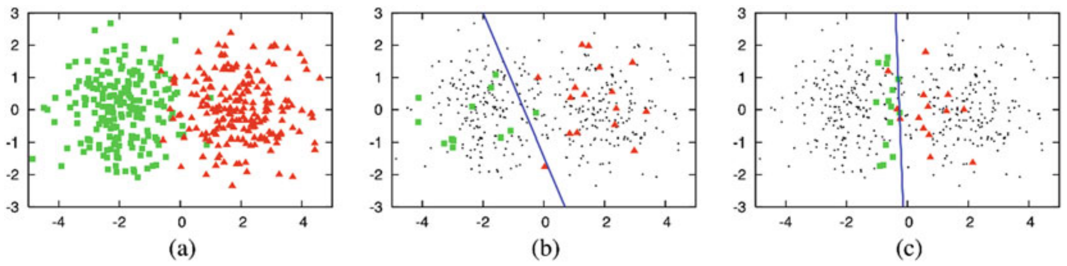

Fig. 6.1 An example of active learning

Active learning can also be applied to solve the network alignment problem when inferring the anchor links across networks. By keeping querying the labels of unlabeled instances, the model can refine the decision boundary with a very small training set. Different methods can be applied in selecting the unlabeled instances to query for their labels in the network alignment problem, which will be introduced in great detail in Sect. 6.4.

6.2.4 Positive and Unlabeled (PU) Learning

In some special case, the labeled training set may only involve the data instances belonging to one single class. For instance, in the e-commerce sites, when recommending products for the users, the training data available is merely the products that users have purchased in the past but no data about the products that the users will not purchase definitely. If we label the purchased products as the positive instances for the users, the training data available will only involve the positive instance only. Besides these positively labeled instances, there also exist a large number of instances in the site that we have no idea about whether the user is interested in or not. These remaining products will be the unlabeled instances on the other hand. Such examples are very common in the real world. On the web, the materials/contents that users are interested are relatively easy to obtain, while we have no idea about those that they dislike. Learning from these positively labeled and unlabeled data is called the *positive-unlabeled learning* (PU learning) task.

Definition 6.1 (Positive and Unlabeled (PU) Learning) Formally, the categories of learning tasks with a positive set $\mathcal{P}$ and unlabeled set $\mathcal{U}$ are called the *positive and unlabeled (PU) learning* tasks. With the $\mathcal{P}$ and $\mathcal{U}$ sets, PU learning aims at building a model to classify the unlabeled instances in $\mathcal{U}$ or some other future data.

The PU learning task is one type of the semi-supervised learning tasks as the unlabeled instances are involved in the model building. Different from classic semi-supervised learning tasks, the labeled instances in the PU learning tasks belong to one single type of class. Viewed in such a perspective, the PU learning task is also one type of one-class learning task [13], which is also known as the *unary learning* tasks aiming at identifying objects of one specific class among all the objects.

Generally speaking, it is “not learnable” merely with the positive instances. However, the addition of the unlabeled instances will make learning from the positive instances possible. Formally, let $(\mathbf{x}, y)$ be an instance tuple with feature vector $\mathbf{x}$ and label $y \in \{-1, +1\}$, the built model can be represented as a mapping $f : \mathbf{x} \to y$. We can rewrite the probability of achieving a wrong prediction as

$$P(f(\mathbf{x}) \neq y) = P(f(\mathbf{x}) = +1, y = -1) + P(f(\mathbf{x}) = -1, y = +1), \tag{6.14}$$

which denotes the cases that $f(\cdot)$ misclassify the negative (or positive) instances to be positive (or negative).

On the other hand, we know that

$$\begin{aligned} &P(f(\mathbf{x}) = +1, y = -1) \\ &\quad = P(f(\mathbf{x}) = +1) - P(f(\mathbf{x}) = +1, y = +1) \\ &\quad = P(f(\mathbf{x}) = +1) - (P(y = +1) - P(f(\mathbf{x}) = -1, y = +1)) . \end{aligned} \tag{6.15}$$

By plugging it into Eq. (6.14), we can have

$$\begin{aligned}
&P(f(\mathbf{x}) \neq y)\\
&\quad = P(f(\mathbf{x}) = +1) - P(y = +1) + 2P(f(\mathbf{x}) = -1, y = +1)\\
&\quad = P(f(\mathbf{x}) = +1) - P(y = +1) + 2P(f(\mathbf{x}) = -1|y = +1)P(y = +1). \qquad (6.16)
\end{aligned}$$

Noting that $P(y = +1)$ is a constant number, if we can also control $P(f(\mathbf{x}) = -1|y = +1)$ to be a small value, then the learning process (i.e., error minimization) is approximately the same as minimizing $P(f(\mathbf{x}) = +1)$. Meanwhile, holding $P(f(\mathbf{x}) = -1|y = +1)$ small is equivalent to ensuring $P(f(\mathbf{x}) = +1|y = +1)$ to be as large as possible while minimizing $P(f(\mathbf{x}) = +1, y = +1 \vee y = -1)$ at the same time. Here, the notation $P(f(\mathbf{x}) = +1|y = +1)$ denotes the probability of classifying positive instances correctly, and $P(f(\mathbf{x}) = +1, y = +1 \vee y = -1)$ represents the probability of classifying unlabeled instances as positive instances. Therefore, if we can ensure the positive instances are correctly classified, while the unlabeled are less likely to be classified as positive instances, the error of the model will be relatively low. Several different techniques have been proposed to ensure the low loss of the learned model, like the *spy techniques* [12] and *bridging probability inference* [18, 19], which will be introduced in Sect. 6.5.

PU learning is a good learning setting for many research problems in social networks, like link prediction, network alignment, and recommendations. By labeling the known friendship links, anchor links, and product purchase actions as the positive instances while the remaining unknown ones as the unlabeled instances, these tasks aforementioned can all be formulated as the PU learning problems. In Sect. 6.5, we will introduce more information about the network alignment model based on PU learning.

6.3 Semi-supervised Network Alignment

In the real-world online social networks involving millions even billions of users, labeling a large number of known anchor links is almost an infeasible task. In a real-world setting, we can usually have a small-sized training set (of identified anchor links), and a relatively big unlabeled set of the anchor links. Model building with the small-sized training set can hardly achieve a very good performance. How to involve the unlabeled set in the model building to improve its performance will become necessary. In this part, we will introduce a method to align the online social networks based on the semi-supervised learning setting. A new linear model similar to S3VM will be introduced first, and we will introduce how to apply the model to address the network alignment problem [20].

6.3.1 Loss Function for Labeled and Unlabeled Instances

Here, we denote all the set of potential anchor links between networks $G^{(1)}$ and $G^{(2)}$ as set $\mathcal{L} = \mathcal{U}^{(1)} \times \mathcal{U}^{(2)}$. Meanwhile the set of positively labeled anchor links (i.e., the existing anchor links) can be represented as set $\mathcal{A}$, and the remaining unlabeled anchor links can be denoted as set $\mathcal{U} = \mathcal{L} \setminus \mathcal{A}$ for simplicity.

For all the links in set $\mathcal{L}$ (involving links in both $\mathcal{A}$ and $\mathcal{U}$), a set of features will be extracted. For instance, we can represent the feature vector extracted for link $l \in \mathcal{L}$ as vector $\mathbf{x}_l \in \mathbb{R}^d$. Meanwhile, we can denote the label of link $l \in \mathcal{L}$ as $y_l \in \mathcal{Y} = \{0, +1\}$ (here, we use 0 to denote the negative class label and +1 to denote the positive class label). All the links in set $\mathcal{A}$ will be assigned with known positive labels $+1$, while the labels of links in set $\mathcal{U}$ are unknown. Therefore, based on these features and labels, all the links in set $\mathcal{L}$ can be represented as a tuple set $\{(\mathbf{x}_l, y_l)\}_{l \in \mathcal{L}}$.

Depending on the separability of the anchor links in set $\mathcal{L}$, different kinds of models can be applied. Here, if we use a linear model to fit the link instances, the model $f : \mathbb{R}^d \to \{+1, 0\}$ to be learned can be represented as a linear combination of the features parameterized with weight $\mathbf{w}$. For instance, with model $f_{\mathbf{w}}(\cdot)$, we can represent the inferred label of link instance l as $f_{\mathbf{w}}(\mathbf{x}_l) = \mathbf{w}^\top \mathbf{x}_l + w_0$, where w_0 is a bias term. By adding a dummy feature 1 for all the link instances, we can also incorporate w_0 into the variable vector $\mathbf{w}$. Therefore, we will use vector $\mathbf{w}$ to represent the weights for the features as well as the bias term when referring to the model variables, and simply use $f_{\mathbf{w}}(\mathbf{x}_l) = \mathbf{w}^\top \mathbf{x}_l$ to denote the model mathematical representation. Based on the known links in set $\mathcal{A}$, we can represent the training loss term as

$$L(f_{\mathbf{w}}, \mathcal{A}) = \sum_{l \in \mathcal{A}} \max(1 - f_{\mathbf{w}}(\mathbf{x}_l) \cdot y_l, 0) = \sum_{l \in \mathcal{A}} \max\left(1 - (\mathbf{w}^\top \mathbf{x}_l) \cdot y_l, 0\right). \tag{6.17}$$

Meanwhile, for the unlabeled links, we have no idea about their true labels in the training process. By following the intuition introduced for the S3VM model, we can represent the loss introduced by the unlabeled links as

$$L(f_{\mathbf{w}}, \mathcal{U}) = \sum_{l \in \mathcal{U}} \max\left(1 - |\mathbf{w}^\top \mathbf{x}_l|, 0\right). \tag{6.18}$$

By combining the loss function defined for the labeled and unlabeled links, we can represent the combined joint optimization function as follows:

$$\begin{aligned} &\min_{\mathbf{w}, \{y_l\}_{l \in \mathcal{U}}} \frac{c_1}{2} L(f_{\mathbf{w}}, \mathcal{A}) + \frac{1}{2} \|\mathbf{w}\|_2^2 + \frac{c_2}{2} L(f_{\mathbf{w}}, \mathcal{U}) \\ &s.t.\ \ y_l \in \{+1, 0\}, \forall l \in \mathcal{U}. \end{aligned} \tag{6.19}$$

Here, $\|\mathbf{w}\|_2^2$ is a regularization term on the model variable $\mathbf{w}$, and c_1, c_2 represent the weights of loss terms of labeled and unlabeled links, respectively.

In the above objective function, the variables to be learned include the weight variable $\mathbf{w}$, as well as the labels of links in the unlabeled set $\mathcal{U}$, i.e., $\{y_l\}_{l \in \mathcal{U}}$. Some approximation methods have been introduced to solve the problem. For example, by assuming that the labels of the link instances in set $\mathcal{U}$ can be correctly inferred by the built model, i.e., $y_l = sign(\mathbf{w}^\top \mathbf{x}_l), l \in \mathcal{U}$. Depending on the value of y_l, we can rewrite the loss introduced by link l as follows:

$$L(f_{\mathbf{w}}, l) = \begin{cases} \max\left(1 - \mathbf{w}^\top \mathbf{x}_l, 0\right), & \text{if } \mathbf{w}^\top \mathbf{x}_l > 0, \\ \max\left(1 + \mathbf{w}^\top \mathbf{x}_l, 0\right), & \text{if } \mathbf{w}^\top \mathbf{x}_l < 0. \end{cases} \tag{6.20}$$

Here, we will preserve the general representation for the loss term introduced by the *unlabeled anchor links* as follows:

$$L(f_{\mathbf{w}}, \mathcal{U}) = \sum_{l \in \mathcal{U}} \max\left(1 - y_l \cdot (\mathbf{w}^\top \mathbf{x}_l), 0\right), \tag{6.21}$$

where labels y_l of these unlabeled links are the variables to be inferred in the model as well.

6.3.2 Cardinality Constraint on Anchor Links

As introduced before, the anchor links in the networks are subject to the *one-to-one* cardinality constraint [11]. Such a constraint will control the maximum number of links incident to the nodes across the networks. Subject to the link cardinality constraints, the prediction tasks of anchor links between the network are no longer independent. For instance, for the links subject to the *one-to-one* constraint, if we can know/infer that the link (u, v) is a positive link (i.e., an existing anchor link), then all the remaining links incident to u or v in the unlabeled set $\mathcal{U}$ will be negative by default. Viewed in such a perspective, the cardinality constraint on links should be incorporated into the problem definition and the result can be improved significantly with such a constraint. In this part, we will introduce the link cardinality constraint and use it to define a set of mathematical constraints on node degrees.

The anchor links studied in this book are assumed to be bi-directional and the node in and out degrees denote the number of links going into/out from them. To represent the node–link incidence relationships, we introduce the node–link in and out matrices $\mathbf{A}_i, \mathbf{A}_o \in \{0, 1\}^{|\mathcal{V}|\times|\mathcal{L}|}$. Entry $A_i(i, j) = 1$ iff the directed link $l_j \in \mathcal{L}$ ends with node n_i, while entry $A_o(i, j) = 1$ iff the directed link $l_j \in \mathcal{L}$ starts with node n_i.

According to the analysis provided before, we can represent the labels of links in $\mathcal{L}$ as vector $\mathbf{y} \in \{+1, 0\}^{|\mathcal{L}|\times 1}$, where entry $y(i)$ represents the label of link $l_i \in \mathcal{L}$. Depending on which group l_i belongs to, its value has different representations

$$y(i) = \begin{cases} +1, & \text{if } l_i \in \mathcal{A}, \\ \text{variable to be inferred}, & \text{if } l_i \in \mathcal{U} \setminus \mathcal{U}_q. \end{cases} \tag{6.22}$$

Furthermore, based on the known and inferred labels of links in $\mathbf{y}$, we can represent the node degrees according to the following theorem.

Theorem 6.1 *The in and out degrees of node u_i in either network $G^{(1)}$ or $G^{(2)}$ can be represented as $\mathbf{A}_i(i, :)\mathbf{y}$ and $\mathbf{A}_o(i, :)\mathbf{y}$, respectively, where $\mathbf{A}_i(i, :)$ and $\mathbf{A}_o(i, :)$ denote the rows corresponding to u_i.*

Proof As introduced before, for the node u_i, we can get the set of links going out from u_i from the i_{th} row of matrix $\mathbf{A}_o$, i.e., $\mathbf{A}_o(i, :)$. For the entries with value 1 in $\mathbf{A}_o(i, :)$, u_i will have a potential link from u_i to the corresponding node. Therefore, the product $\mathbf{A}_o(i, :)\mathbf{y}$ will remove the remaining links, and sum all the labels of links starting from node u_i. Considering that the labels have value either $+1$ or 0, $\mathbf{A}_o(i, :)\mathbf{y}$ will actually denote the degree of node u_i. In a similar way, we can also obtain that the node in degree d_i equals to $\mathbf{A}_i(i, :)\mathbf{y}$.

Let $\mathbf{0}$ and $\mathbf{1}$ denote the vectors with all 0s and 1s of length $|\mathcal{L}|$, respectively. According to the previous analysis, the link cardinality constraint can be applied to define the degree constraint of nodes in the network, which can be represented as follows:

$$\mathbf{0} \preccurlyeq \mathbf{A}_i\mathbf{y} \preccurlyeq \mathbf{1}, \tag{6.23}$$

$$\mathbf{0} \preccurlyeq \mathbf{A}_o\mathbf{y} \preccurlyeq \mathbf{1}. \tag{6.24}$$

6.3.3 Joint Objective Function for Semi-supervised Network Alignment

By adding the node degree constraint to the objective function introduced before, we can represent the joint optimization objective function as

$$\begin{aligned}
&\min_{\mathbf{w},\mathbf{y}} \frac{c_1}{2} L(f_{\mathbf{w}}, \mathcal{A}) + \frac{1}{2}\|\mathbf{w}\|_2^2 + \frac{c_2}{2} L(f_{\mathbf{w}}, \mathcal{U}) \\
&s.t.\;\; y_l \in \{+1, 0\}, \forall l \in \mathcal{L},\;\; y_l = +1, \forall l \in \mathcal{A}, \\
&\qquad \mathbf{0} \preccurlyeq \mathbf{A}_i \mathbf{y} \preccurlyeq \mathbf{1},\;\; \mathbf{0} \preccurlyeq \mathbf{A}_o \mathbf{y} \preccurlyeq \mathbf{1}.
\end{aligned} \tag{6.25}$$

The objective function involves two variables, and it is easy to see that it is not jointly convex in terms of these two variables. To solve the function, techniques like alternative updating can be applied here. By fixing one variable, we can keep updating the other variable. Such an alternative updating process will continue until convergence.

Step 1: By fixing $\mathbf{y}$, the objective function will be reduced to the objective function of traditional SVM model involving variable $\mathbf{w}$ only:

$$\min_{\mathbf{w}} \frac{c_1}{2} \sum_{l \in \mathcal{A}} \max\left(1 - y_l \cdot (\mathbf{w}^\top \mathbf{x}), 0\right) + \frac{1}{2}\|\mathbf{w}\|_2^2 + \frac{c_2}{2} \sum_{l \in \mathcal{U}} \max\left(1 - y_l \cdot (\mathbf{w}^\top \mathbf{x}), 0\right). \tag{6.26}$$

The learning methods for the SVM model introduced in Sect. 2.3.3 can be applied to learn the optimal model variable, which will not be introduced here again.

Step 2: By fixing $\mathbf{w}$, the objective function will be reduced to the link selection problem we introduced before in Sect. 4.5. Let $\hat{y}_l = \mathbf{w}^\top \mathbf{x}_l$ denote the inferred label of link l, and the objective function will be reduced to

$$\begin{aligned}
&\min_{\mathbf{y}} \frac{c_1}{2} \sum_{l \in \mathcal{A}} \max\left(1 - \hat{y}_l \cdot y_l, 0\right) + \frac{1}{2}\|\mathbf{w}\|_2^2 + \frac{c_2}{2} \sum_{l \in \mathcal{A}} \max\left(1 - \hat{y}_l \cdot y_l, 0\right) \\
&s.t.\;\; y_l \in \{+1, 0\}, \forall l \in \mathcal{L},\;\; y_l = +1, \forall l \in \mathcal{A}, \\
&\qquad \mathbf{0} \preccurlyeq \mathbf{A}_i \mathbf{y} \preccurlyeq \mathbf{1},\;\; \mathbf{0} \preccurlyeq \mathbf{A}_o \mathbf{y} \preccurlyeq \mathbf{1}.
\end{aligned} \tag{6.27}$$

Some algorithms like greedy link selection introduced in Sect. 4.5 can be applied to determine the label vector $\mathbf{y}$. Here, we will not talk about that algorithm again, and more information about the selection algorithm is provided in Algorithm 3 in Sect. 4.5.

6.4 Active Network Alignment

In this section, we will introduce an algorithm to address the network alignment problem based on active learning [16]. Different from the traditional active learning problems, due to the one-to-one constraint on anchor links, if an unlabeled anchor link $a = (u, v)$ is identified as positive (i.e., existing), all the other unlabeled anchor links incident to u or v will be negative (i.e., non-existing) automatically. Viewed in such a perspective, querying for the labels of potential positive anchor links in the unlabeled set will be much more rewarding in the active network alignment problem, since the identification of one positive anchor link will help identify a bunch of negative anchor links simultaneously. Various novel anchor link information gain measures will be defined in this section, based on which several active network alignment methods will be introduced.

Active learning aims at minimizing the labeling cost of the training set by asking the model to choose which examples to query for the labels. We can represent the set of labeled anchor links as set $\mathcal{A} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_l, y_l)\}$, which involves the positively labeled anchor links existing between the networks. The active learning algorithm will train an anchor link prediction model M with the training set $\mathcal{A}$. During the training process, what the active learner needs to do is to select a query pool of unlabeled anchor links from the unlabeled anchor link set $\mathcal{U} = \mathcal{U}^{(1)} \times \mathcal{U}^{(2)} \setminus \mathcal{A}$. The selection strategy is to pick the most valuable anchor link(s) according to the values computed by applying M on $\mathcal{U}$, which can be represented as set $\mathcal{U}_Q \subset \mathcal{U}$. The data instances in $\mathcal{U}_Q$ together with their labels will be added to $\mathcal{U}$ to update the model M. The training process and query process will be repeated until the limit of query cost has been reached.

In this section, we will first introduce several *anchor link query strategy* [25] for the network alignment problem first, based on which we will introduce the objective function of *active network alignment* [14] afterwards and provide the solutions.

6.4.1 Anchor Link Label Query Strategy

The main challenge in the active network alignment problem will be the query process of the unlabeled data instances. In each round of query process, traditional active learning methods usually just add the newly queried samples to the training set. However, via the one-to-one constraint, the constrained active learning methods will be able to infer the labels of some unlabeled anchor links after identifying one positive anchor link, and thus the samples to be added to the training set can be more than the queried samples.

Example 6.2 As shown in Fig. 6.2, there are 4 unlabeled anchor links in the query pool, i.e., $\{(u_1^{(1)}, u_1^{(2)}), (u_1^{(1)}, u_2^{(2)}), (u_2^{(1)}, u_1^{(2)}), (u_2^{(1)}, u_2^{(2)})\}$. Let's assume after querying an oracle, we get the label for link $(u_1^{(1)}, u_1^{(2)})$ to be $+1$ (i.e., $u_1^{(1)}$ and $u_1^{(2)}$ are the same user). Traditional active learners will just add $(u_1^{(1)}, u_1^{(2)})$ to the positive training set. However, a constrained active learner will firstly infer that $(u_1^{(1)}, u_2^{(2)})$ and $(u_2^{(1)}, u_1^{(2)})$ to be "negative" according to the one-to-one constraint, and then add $(u_1^{(1)}, u_1^{(2)})$ to the positive training set, as well as $(u_1^{(1)}, u_2^{(2)})$ and $(u_2^{(1)}, u_1^{(2)})$ to the negative training set. In this way, the constrained active learning methods can incorporate two more negatively labeled data instances (i.e., anchor links) than the traditional active learning methods under the same

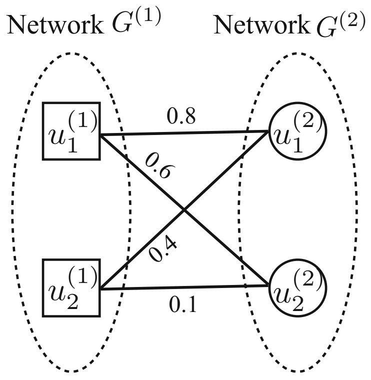

Fig. 6.2 An example of active anchor link query

query limitation (where the query limitation means the cost of achieving anchor link labels within one round of query).

6.4.1.1 Regular Active Network Alignment

Many active learning methods usually involve the evaluation of the informativeness of unlabeled instances. However, due to the challenges created by the *one-to-one* cardinality constraint, many query methods used in active learning cannot be applied to the anchor link prediction task. Among the existing query methods [16], the simplest and most commonly used query strategy is uncertainty sampling, where the learner will query the labels of instances that it is the least certain about. There exist several commonly used sampling strategies in uncertainty sampling, including least confidence sampling, the margin sampling, and the entropy based sampling. Compared with the former two sampling strategies, the entropy based sampling generalizes more easily to complex structured instances. It is because by computing the entropy, we can compare the amount of information contained in different multi-structured samples in a uniform metric. The active network alignment method to be introduced here is based on the entropy theory, and aims to calculate the potential entropy $H(l)$ for each unlabeled link $l \in \mathcal{U}$. Here we define $H(l)$ as the evaluated amount of information that the active network alignment model can gain by identifying the label of anchor link l.

Here, we use notation $\Gamma(l)$ to represent the related anchor link set of a given anchor link l, i.e., the set of all anchor links in $\mathcal{U}$ that are incident to the nodes forming l. The major idea of the regular active learning method is to calculate $H(l)$ for each of the unlabeled anchor link $l \in \mathcal{U}$, and select the anchor link with the highest score to query for its label. If the label for the link is "negative," the link will be added to the training set. Meanwhile, if the label of the link is "positive," besides this link, we will also extract the remaining incident anchor links, i.e., those in $\Gamma(l)$, and add them as "negative" instances into the training set.

Formally, the information entropy of the anchor link $l \in \mathcal{A}$ can be represented as

$$H(l) = -\sum_{y \in \mathcal{Y}} P_M(y|\mathbf{x}_l) log P_M(y|\mathbf{x}_l), \tag{6.28}$$

where the term $P_M(y|\mathbf{x}_l)$ denotes the posterior probability of anchor link $l \in \mathcal{A}$ inferred by an anchor link inference model M. Literally, for the anchor link that can introduce a larger entropy, the learned model M is less certain about its label. Querying for the label of such links can actually introduce the maximum information gain.

6.4.1.2 Biased Constrained Active Network Alignment

For the network alignment task, generally identifying the potentially positive instances can lead to more information, since the identification of one positive instance can lead to a bunch of identified negative anchor links at the same time due to the *one-to-one* constraint. As we discussed before, because of the sparsity of anchor links, acquiring enough informative positive anchor links under a limited cost is very important. However, in the regular active network alignment method introduced in the previous subsection, there may not be enough mechanism to increase the probability of each identified link to be positive. So if we can explore such a mechanism, and integrate it into the active network alignment model, we will be able to achieve better results. In this part, we will present the biased constrained active network alignment method, which prefers the potential positive links over the negative ones in the query process.

According to [11], under different circumstances, when predicting the existing anchor links, by incorporating the *one-to-one* cardinality constraint into the learning model, it will bring about a much higher accuracy. So in the biased constrained network alignment approach to be introduced here, the learner should firstly apply the existing non-active network alignment models (e.g., MNA [11] as introduced in Sect. 4.3) to predict the potentially positive anchor links in $\mathcal{U}$, which can be recognized as set $\mathcal{U}_+ \subset \mathcal{U}$. In each round, links from set $\mathcal{U}_+$ will be selected to query for their labels, where those which can introduce the maximum information gain can be the optimum here. Different strategies can be applied to rank the anchor links in set $\mathcal{U}_+$. Meanwhile, considering that the anchor links to be inferred are correlated, ranking of the candidates in set $\mathcal{U}_+$ depends on not only these links themselves but also the other links incident to them.

Here, we would like to introduce two new strategies, i.e., *biased likelihood* and *biased entropy*, for ranking the links in $\mathcal{U}_+$. Given two links $l, l' \in \mathcal{L}$, we can use notation $l \cap l'$ to denote the set of shared nodes by l and l'. Given a link $l \in \mathcal{U}_+$, we can represent the links incident to l (i.e., sharing a common node) as set $\Gamma(l) = \{l' | l' \in \mathcal{L}, l \cap l' \neq \emptyset\}$.

Biased Likelihood: By labeling link l to be positive, we can know that links in set $\Gamma(l)$ will be negative by default. The likelihood of such a case can be represented as

$$P(l, \Gamma(l)) = P(y_l = +1 | \mathbf{x}_l) \cdot \prod_{l' \in \Gamma(l)} P(y_{l'} = -1 | \mathbf{x}_{l'}). \tag{6.29}$$

Links in set $\mathcal{U}_+$ can be sorted according to the probability $P(l, \Gamma(l))$ and those with higher probability can be queried for the labels.

Biased Entropy: Besides the likelihood, a similar measure like the entropy can be defined based on the intuition as well, which considers not only the positive link labels but also the uncertainty. For instance, after querying for the label of link l, we can have two scenarios:

- *l is positive*: If link l is positive, links in $\Gamma(l)$ will be negative for sure.
- *l is negative*: If link l is negative, links in $\Gamma(l)$ can be either positive or negative.

Therefore, the uncertainty about the labels of link l and its incident set $\Gamma(l)$ can be represented as

$$\begin{aligned} &H(l, \Gamma(l)) \\ &\quad = P(y_l = +1 | \mathbf{x}_l) \cdot H(l, \Gamma(l) | y_l = +1) + P(y_l = -1 | \mathbf{x}_l) \cdot H(l, \Gamma(l) | y_l = -1). \end{aligned} \tag{6.30}$$

In the above equation, we have $H(l, \Gamma(l) | y_l = +1)$ and $H(l, \Gamma(l) | y_l = +1)$ denotes the conditional entropy as follows:

$$\begin{aligned} H(l, \Gamma(l) | y_l = +1) = &- P_M(y_l = +1 | \mathbf{x}_l) log P_M(y_l = +1 | \mathbf{x}_l) \\ &- \sum_{l' \in \Gamma(l)} P_M(y_{l'} = -1 | \mathbf{x}_{l'}) log P_M(y_{l'} = -1 | \mathbf{x}_{l'}) \end{aligned} \tag{6.31}$$

and

$$\begin{aligned} H(l, \Gamma(l) | y_l = -1) = &- P_M(y_l = -1 | \mathbf{x}_l) log P_M(y_l = -1 | \mathbf{x}_l) \\ &- \sum_{l' \in \Gamma(l)} \sum_{y \in \mathcal{Y}} P_M(y_{l'} = y | \mathbf{x}_{l'}) log P_M(y_{l'} = y | \mathbf{x}_{l'}). \end{aligned} \tag{6.32}$$

For the link l with a larger *biased entropy*, we will be less sure about the results of l and its incident neighbor set $\Gamma(l)$. All the potentially positive anchor links in set $\mathcal{U}_+$ can be sorted according to their entropy scores, and those with larger *biased entropy* scores can be picked for labeling.

6.4.2 Active Network Alignment Objective Function

Here, we will use the loss function for labeled anchor links and cardinality constraints introduced in Sect. 6.3, but for the unlabeled anchor links, we propose to further query for the labels of a subset of the data instances. Given the unlabeled anchor link set $\mathcal{U}$, we can denote the subset of anchor links to be selected for querying the labels as $\mathcal{U}_Q$. The true label of link $l \in \mathcal{U}_Q$ after query can be represented as $\tilde{y}_l \in \{+1, -1\}$. The remaining links in set $\mathcal{U}$ can be represented as $\mathcal{U} \setminus \mathcal{U}_Q$, whose labels are still unknown. Based on the loss functions introduced before, depending on whether the labels of links are queried or not, we can further specify the loss function for set $\mathcal{U}$ as

$$\begin{aligned} L(f_{\mathbf{w}}, \mathcal{U}) &= L(f_{\mathbf{w}}, \mathcal{U}_Q) + L(f_{\mathbf{w}}, \mathcal{U} \setminus \mathcal{U}_Q) \\ &= \sum_{l \in \mathcal{U}_Q} (\mathbf{w}^\top \mathbf{x}_l - \tilde{y}_l)^2 + \sum_{l \in \mathcal{U} \setminus \mathcal{U}_Q} (\mathbf{w}^\top \mathbf{x}_l - y_l)^2. \end{aligned} \tag{6.33}$$

Here, we need to add more remarks that notation $\tilde{y}_l$ denotes the queried label of link $l \in \mathcal{U}_Q$ which will be a known value, while y_l will be a variable to be inferred in the model for all the links $\mathcal{U} \setminus \mathcal{U}_Q$.

By combining the loss functions for links in different subsets together with the anchor link cardinality constraint, we can represent the objective function for *active network alignment* to be

$$\begin{aligned} \min_{\mathbf{w}, \mathbf{y}, \mathcal{U}_Q} \; & \frac{c_1}{2} L(f_{\mathbf{w}}, \mathcal{A}) + \frac{1}{2} \|\mathbf{w}\|_2^2 + \frac{c_2}{2} L(f_{\mathbf{w}}, \mathcal{U}_Q) + \frac{c_3}{2} L(f_{\mathbf{w}}, \mathcal{U} \setminus \mathcal{U}_Q) \\ s.t. \; & |\mathcal{U}_Q| \leq b, \\ & y_l \in \{+1, -1\}, \forall l \in \mathcal{L}, \; y_l = +1, \forall l \in \mathcal{A}, \; y_l = \tilde{y}_l, \forall l \in \mathcal{U}_Q, \\ & \mathbf{0} \preccurlyeq \mathbf{A}_o \mathbf{y} \preccurlyeq \mathbf{1}, \; \mathbf{0} \preccurlyeq \mathbf{A}_i \mathbf{y} \preccurlyeq \mathbf{1}, \end{aligned} \tag{6.34}$$

where c_1, c_2, and c_3 denote the weights of the loss terms and b represents the available query budget in the learning process.

As shown in the above objective function, besides the variable $\mathbf{w}$ of the model and the link labels $\mathbf{y}$ to be inferred, we also need to select the optimal node set $\mathcal{U}_Q$ to query for the labels in active learning. The selection of different node subsets can affect the link prediction result greatly, and the selection of the optimal query node set renders the problem to be much more challenging. In the above objective function, the optimal query node selection is actually a combinatorial problem, which is NP-hard with a search space involving $\binom{|\mathcal{U}|}{|\mathcal{U}_Q|}$ different options.

For simplicity, we assume the weights c_1, c_2, c_3 all to be c, i.e., all the links in the networks are assumed to be of similar importance in training. And the new loss term of all the links in $\mathcal{A}, \mathcal{U}_Q$ and $\mathcal{U} \setminus \mathcal{U}_Q$ can be simplified as

$$\begin{aligned} & \frac{c_1}{2} L(f_{\mathbf{w}}, \mathcal{A}) + \frac{c_2}{2} L(f_{\mathbf{w}}, \mathcal{U}_Q) + \frac{c_3}{2} L(f_{\mathbf{w}}, \mathcal{U} \setminus \mathcal{U}_Q) \\ & = \frac{c}{2} L(f_{\mathbf{w}}, \mathcal{L}) \\ & = \frac{c}{2} \|\mathbf{w}\mathbf{X} - \mathbf{y}\|_2^2, \end{aligned} \tag{6.35}$$

where matrix $\mathbf{X} = [\mathbf{x}_{l_1}^\top, \mathbf{x}_{l_2}^\top, \ldots, \mathbf{x}_{l_{|\mathcal{L}|}}^\top]^T$ denotes the feature matrix about all these anchor links in the potential anchor link set $\mathcal{L}$.

Here, we can see that the objective function involves multiple variables, like **w**, **y**, and the query set $\mathcal{U}_Q$, and the objective is not jointly convex with regarding these variables. What's more, the inference of the label variable **y** and the query set $\mathcal{U}_Q$ are both combinatorial problems. In this section, we propose to update the variables alternatively, while fixing the remaining ones, and design a hierarchical alternative variable updating process for solving the problem instead:

1. fix $\mathcal{U}_Q$, and update **y** and **w**,
   (1–1) with fixed $\mathcal{U}_Q$, fix **y**, update **w**,
   (1–2) with fixed $\mathcal{U}_Q$, fix **w**, update **y**,
2. fix **y** and **w**, and update $\mathcal{U}_Q$.

A remark to be added here: we can see that variable $\mathcal{U}_Q$ is different from the remaining two, which involves the label query process with the oracle subject to the specified budget. To differentiate these two iterations, we call the iterations (1) and (2) as the *external iteration*, while we call (1–1) and (1–2) as the *internal iteration*. Next, we will illustrate the detailed alternative learning algorithm as follows.

- **External Iteration Step (1)**: Fix $\mathcal{U}_Q$, update **y**, **w**.
  - **Internal Iteration Step (1–1)**: Fix $\mathcal{U}_Q$, **y**, update **w**.
    With **y**, $\mathcal{U}_Q$ fixed, we can represent the objective function involving variable **w** as

$$\min_{\mathbf{w}} \frac{c}{2} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 + \frac{1}{2} \|\mathbf{w}\|_2^2 . \tag{6.36}$$

    The objective function is a quadratic convex function, and its optimal solution can be represented as

$$\mathbf{w} = \mathbf{H}\mathbf{y} = c(\mathbf{I} + c\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}, \tag{6.37}$$

    where $\mathbf{H} = c(\mathbf{I} + c\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top$ is a constant matrix. Therefore, the weight vector **w** depends only on the **y** variable.
  - **Internal Iteration Step (1–2)**: Fix $\mathcal{U}_Q$, **w**, update **y**.
    With $\mathcal{U}_Q$, **w** fixed, together with the constraint, we know that terms $L(f_{\mathbf{w}}, \mathcal{A})$, $L(f_{\mathbf{w}}, \mathcal{U}_Q)$, and $\|\mathbf{w}\|_2^2$ are all constant. And the objective function will be

$$\begin{aligned} &\min_{\mathbf{y}} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 \\ &s.t.\ y_l \in \{+1, 0\}, \forall l \in \mathcal{U} \setminus \mathcal{U}_Q, \\ &\quad y_l = \tilde{y}_l, \forall l \in \mathcal{U}_Q \text{ and } y_l = +1, \forall l \in \mathcal{A}, \\ &\quad \mathbf{0} \preccurlyeq \mathbf{A}_i\mathbf{y} \preccurlyeq \mathbf{1}, \text{and } \mathbf{0} \preccurlyeq \mathbf{A}_o\mathbf{y} \preccurlyeq \mathbf{1}. \end{aligned} \tag{6.38}$$

    It is an integer programming problem, which has been shown to be NP-hard and no efficient algorithm exists that leads to the optimal solution. Here, we will introduce the greedy link selection algorithm proposed in [20] based on values $\hat{\mathbf{y}} = \mathbf{X}\mathbf{w}$, which has been proven to achieve $\frac{1}{2}$-approximation of the optimal solution.

- **External Iteration Step (2)**: Fix $\mathbf{w}$, $\mathbf{y}$, update $\mathcal{U}_Q$.
  Selecting the optimal set $\mathcal{U}_Q$ at one time involves the search of all the potential b link instance combinations from the unlabeled set $\mathcal{U}$, whose search space is $\binom{|\mathcal{U}|}{b}$, and there is no known efficient approach for solving the problem in polynomial time. Therefore, instead of selecting them all at one time, we propose to choose several link instances greedily in each iteration. Due to the one-to-one constraint, the unlabeled anchor links no longer bear equal information, and querying for labels of potential positive anchor links will be more "informative" compared with negative anchor links. Formally, the strategies introduced in Sect. 6.4.1 can all be applied to rank the links either based on their *biased likelihood* or their *biased entropy*, and we will not introduce them again here.

6.5 Positive and Unlabeled (PU) Network Alignment

In the previous sections, the anchor links to be inferred are subject to the *one-to-one* cardinality constraint, where each user is assumed to have at most one account within one social network. However, in some scenarios, as mentioned in Sect. 4.5, users may create multiple accounts in the same social networks, where each account will be for different purposes, e.g., personal socialization vs professional socialization, or family socialization vs external socialization. In such a case, each user can be connected with multiple anchor links across networks, and the cardinality constraint on the anchor links will become *many-to-one* or *many-to-many* instead. The identification of positive anchor links can no longer help to infer the other potential negative anchor links.

In this section, we will introduce a network alignment to address the aforementioned problem based on the PU learning settings [12]. Before we talk about the detailed information about the PU network alignment model [18, 19], we will first introduce the formulation and the preliminary used in the studied problem.

6.5.1 PU Network Alignment Problem Formulation and Preliminary

Formally, given the partially aligned online social networks $\mathcal{G} = ((G^{(2)}, G^{(1)}), (\mathcal{A}))$ with the set of anchor links $\mathcal{A}$ connecting the shared users across network $G^{(1)}$ and $G^{(2)}$, we can represent the set of existing and non-existing anchor links between these two networks as $\mathcal{A}$ and $\mathcal{U} = \mathcal{U}^{(1)} \times \mathcal{U}^{(2)} \setminus \mathcal{A}$. If these existing and non-existing anchor links are treated as the "*positive*" and "*unlabeled*" anchor links, the task of building a model to infer the existence of anchor links across networks will be formulated as a PU learning problem.

As introduced in [19], across these two networks, we can extract the set of both existing and unidentified anchor links. To differentiate these links, a term named "*connection state*": $z \in \{-1, +1\}$ was introduced in [19]. If a certain link (u, v) is an existing anchor link across the networks, then $z(u, v) = +1$; if (u, v) is an unidentified anchor link, then $z(u, v) = -1$. Meanwhile, besides the "*connection state*," all the anchor links can also have their own *labels*, i.e., $y \in \{-1, +1\}$. In this section, if an anchor link (u, v) is/will be identified to be existing, then $y(u, v) = +1$; if (u, v) is not an anchor link, then $y(u, v) = -1$. As shown in Fig. 6.3, for all the existing anchor links across the networks, their connection states z and labels y are all $+1$, while the connection states z of all initially unidentified anchor links are -1 but the labels y of these anchor links can be either $+1$ or -1, as these unidentified anchor links include both anchor links that should either exist or not exist. These unidentified anchor links are referred to as the unlabeled anchor links in the PU network alignment problem.

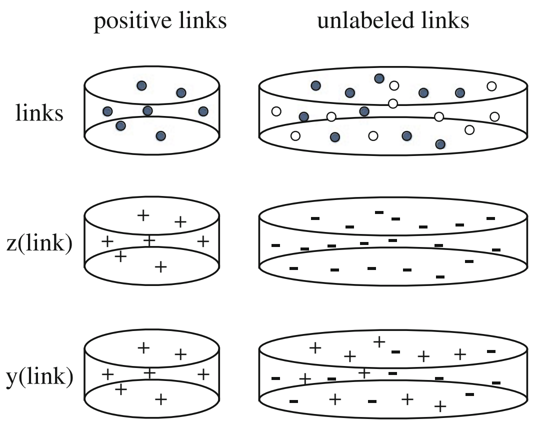

Fig. 6.3 Example of connection states and labels of links in PU link prediction

Based on the problem formulation and preliminary concepts introduced above, we will introduce the PU network alignment model in the following part, which will illustrate the relationships of the anchor link *connection state* and their *labels*.

6.5.2 PU Network Alignment Model

For each anchor link, a set of features can be extracted from the networks. For instance, the feature vector extracted for certain anchor/social link (u, v) can be represented as $\mathbf{x}(u, v)$. As a result, each anchor link (u, v) across the networks can be denoted as a tuple $\langle \mathbf{x}(u, v), y(u, v), z(u, v) \rangle$. Let $p(\mathbf{x}, y, z)$ be the joint distribution of $\mathbf{x}$, y, and z. As shown in Fig. 6.3, all the existing links $(z = 1)$ are positive links $(y = 1)$. In other words, we have

$$p(y = 1|\mathbf{x}, z = 1) = p(y = 1|z = 1) = 1.0. \tag{6.39}$$

A basic assumption in the PU network alignment model is that *the existing positive links are randomly sampled from the whole positive link set* [10], which means that for two arbitrary positive links (u_1, v_1) and (u_2, v_2) we have

$$\begin{aligned} &p(z(u_1, v_1) = 1|\mathbf{x}(u_1, v_1), y(u_1, v_1) = 1) \\ &\quad = p(z(u_2, v_2) = 1|\mathbf{x}(u_2, v_2), y(u_2, v_2) = 1). \end{aligned} \tag{6.40}$$

Based on such an assumption, the conditional distribution $p(z = 1|\mathbf{x}, y = 1)$ is independent of variable $\mathbf{x}$, i.e.,

$$\begin{aligned} p(z = 1|y = 1) = &\sum_{link \in \mathcal{L}} p(z = 1|\mathbf{x}(link), y = 1) \cdot p(\mathbf{x}(link)|y = 1) \\ =&p(z = 1|\mathbf{x}, y = 1) \cdot \sum_{link \in \mathcal{L}} p(\mathbf{x}(link)|y = 1) \\ =&p(z = 1|\mathbf{x}, y = 1), \end{aligned} \tag{6.41}$$

where $\mathcal{L} = \mathcal{A} \cup \mathcal{U}$ denotes all the potential anchor links across networks.

Meanwhile, the probabilities that anchor link l is predicted to be "existing" ($z = +1$) and "positively labeled" ($y = +1$) can be defined as the "*existence probability*" (i.e., $p(z = 1|\mathbf{x})$) and "*positively labeled probability*" (i.e., $p(y = 1|\mathbf{x})$), respectively [19]. The relationship between links' "*existence probability*" and "*positively labeled probability*" can be formally represented as follows:

$$\begin{aligned} p(z = 1|\mathbf{x}) &= p(z = 1|\mathbf{x}) \cdot p(y = 1|\mathbf{x}, z = 1) = p(y = 1, z = 1|\mathbf{x}) \\ &= p(y = 1|\mathbf{x}) \cdot p(z = 1|\mathbf{x}, y = 1) \\ &= p(y = 1|\mathbf{x}) \cdot p(z = 1|y = 1). \end{aligned} \tag{6.42}$$

Based on the above equation, the links' *positively labeled probabilities* can be inferred from their *existence probabilities* if we can know $p(z = 1|y = 1)$ in advance, where $p(z = 1|y = 1)$ is called the *bridging probability* formally in the PU network alignment model.

Definition 6.2 (Bridging Probability) The term $p(z = 1|y = 1)$ is formally defined as the *bridging probability* between the existence probability and the positively labeled probability.

The bridging probability can actually be inferred with the binary classification models built with the existing ($z = +1$) and unconnected ($z = -1$) links [10]. To achieve such a goal, we can split all the existing and unconnected links into "training set" and "validation set" via cross validation. Classification models built based on the training set can be applied to the validation set. Let Pos be the subset of links that are positive in the validation set. We have

Bridging Probability Inference Equation

$$\begin{aligned} p(z = 1|y = 1) &= \frac{1}{|\mathrm{Pos}|} \sum_{link \in \mathrm{Pos}} p(z = 1|y = 1) \\ &= \frac{1}{|\mathrm{Pos}|} \sum_{link \in \mathrm{Pos}} p(z = 1|\mathbf{x}, y = 1), \end{aligned} \tag{6.43}$$

where $p(z = 1|y = 1) = p(z = 1|\mathbf{x}, y = 1)$ can hold according to proof in the previous part. For links in Pos, we have $p(y = 1|\mathbf{x}) = 1$, $p(z = 1|\mathbf{x}, y = -1) = 0$ and $p(y = -1|\mathbf{x}) = 0$. Therefore, we have

$$\begin{aligned} p(z = 1|y = 1) = \frac{1}{|\mathrm{Pos}|} \sum_{link \in \mathrm{Pos}} &(p(z = 1|\mathbf{x}, y = 1)p(y = 1|\mathbf{x}) \\ &+ p(z = 1|\mathbf{x}, y = -1)p(y = -1|\mathbf{x})) \\ = \frac{1}{|\mathrm{Pos}|} \sum_{link \in \mathrm{Pos}} &p(z = 1|\mathbf{x}). \end{aligned} \tag{6.44}$$

As a result, the average existence possibility of links in Pos works as an estimator of the bridging probability, which clearly clarifies the correlation between link's existence probability and positively labeled probability. Based on the inferred bridging probability $p(z = 1|y = 1)$, we can predict the positively labeled probabilities of anchor and social links based on their existence probabilities. Meanwhile, inferring the anchor links' *existence probability* (i.e., $p(z = 1|\mathbf{x})$) is an easy task, which can be addressed with the supervised models for anchor link prediction introduced in Sect. 4 perfectly. Based on the links' inferred *existence probability* together with the above *bridging probability*, we will be able to compute the anchor links' *positively labeled probability* as the final output.

6.6 Summary

In this chapter, we introduced a new type of learning tasks using both labeled and unlabeled data instances for model building, which are formally named as the semi-supervised learning tasks. Based on three different semi-supervised learning settings, we talked about the network alignment problem and introduced several different network alignment approaches to address the problem.

To provide readers with the background knowledge about semi-supervised learning, we used one section to introduce the semi-supervised learning problems and algorithms. Based on different assumptions, e.g., smoothness assumption, cluster assumption, and manifold assumption, existing semi-supervised learning models adopt different ways to use the unlabeled data instances. Three different semi-supervised learning models, i.e., S3VM, graph based models, and generative models, were introduced in this section. In addition to regular semi-supervised learning, active learning and PU learning were introduced as two special cases of the semi-supervised learning problem.

We introduced a model similar to S3VM to resolve the network alignment task with both labeled and unlabeled anchor link instances. However, slightly different from the regular semi-supervised learning tasks, the anchor links studied in the network alignment problem are not independent, which are strongly correlated with each other due to the one-to-one cardinality constraint. By modeling the anchor link cardinality constraint as a mathematical constraint on node degrees, the semi-supervised network alignment problem was formulated as an optimization problem instead.

We also introduced a network alignment model based on active learning in this section, which can help address the lack of training data problem. Instead of requiring a large number of training data initially, the introduced active network alignment model adopted an active query strategy to get the labels of unlabeled anchor links from an oracle subject to a pre-specified query budget. However, due to the one-to-one cardinality constraint on anchor links, the information that can be provided by the positive and negative anchor link is no longer balanced. Two query strategies, i.e., biased likelihood and biased entropy, were introduced for anchor link selection.

At the end of this chapter, we introduced to model the network alignment problem as a positive and unlabeled (PU) learning problem. In the PU network alignment tasks, the anchor links will be represented as tuples, involving the feature representation, connection state, and label, respectively. Based on the correlation between the existence probability and positively labeled probability, the introduced model is able to infer the anchor links positively labeled probabilities from their existence probabilities effectively.

6.7 Bibliography Notes

For the readers who are interested in the semi-supervised learning, the textbook [8] is highly recommended, which provides a comprehensive introduction about semi-supervised learning problems and algorithms. The readers can also take a look at the survey article [23], which covers the literature review of the semi-supervised learning algorithms and potential applications. A survey about active learning is available in [16], and the PU learning papers include [10, 12].

The PU network alignment problem was initially proposed in [18], which proposed a semi-supervised method to infer both anchor links and social links simultaneously across aligned social networks. The proposed method analyzes the correlation between links existence probability and formation probability (i.e., positively labeled probability introduced in Sect. 6.5). Active network alignment is studied in [14, 25] for the first time. Via a step-wise selection of anchor links for label query, active network alignment models are capable to achieve very good performance with a small amount of labeling efforts. The semi-supervised network alignment problem formulation was originally introduced in [14, 20, 21], and the readers may also refer to these academic papers for more information about relating works.

6.8 Exercises

1. (Easy) In the S3VM model, we use Eq. (6.1) to denote the loss for the unlabeled data instances. Please briefly explain why it can work well to utilize information of unlabeled data instances in learning the models.
2. (Easy) In the *local and global consistency model*, please briefly talk about the physical meaning of the regularization term $reg = \mathbf{f}^\top \mathbf{L}_n \mathbf{f}$ (as indicated in Eq. (6.10)), and explain why it can work well in regularizing the model.
3. (Easy) Please briefly introduce what is active learning, and provide the advantages of active learning compared against other learning tasks.
4. (Easy) According to Sect. 6.2.4, please briefly explain the ideas of PU learning and the circumstances where it can work well.
5. (Medium) Please read article [9], and briefly introduce how to learn the semi-supervised generative model based on the EM algorithm.
6. (Medium) When introducing the active network alignment model in Sect. 6.4, we mention that "positive and negative anchor links may have different amounts of information." Please explain what does that sentence mean, and briefly talk about why the introduced biased query strategies we introduced in Sect. 6.4.1.2 can work better than regular query strategies, e.g., the *entropy* based strategy introduced in Sect. 6.4.1.1.
7. (Medium) Please briefly introduce the *bridging probability inference equation* used in Sect. 6.5, and explain why it can work in inferring the potential *bridging probability*.
8. (Hard) Please use your preferred programming language to implement the semi-supervised learning algorithm introduced in Sect. 6.3, and compare its advantages over the regular supervised network alignment approach that we introduced in Sect. 4.3 with experiments on some toy data sets.
9. (Hard) Please try to implement the active network alignment algorithm introduced in Sect. 6.4 with one of your preferred programming language, and test its effectiveness on a toy network data set.
10. (Hard) Please try to implement the PU network alignment algorithm introduced in Sect. 6.5 with one of your preferred programming language, and test its effectiveness on a toy network data set.

References

1. E. Allgower, K. Georg, *Numerical Continuation Methods: An Introduction* (Springer, Berlin, 1990)
2. M. Belkin, I. Matveeva, P. Niyogi, Regularization and semi-supervised learning on large graphs, in *Learning Theory*, ed. by J. Shawe-Taylor, Y. Singer (2004)
3. M. Belkin, P. Niyogi, V. Sindhwani, Manifold regularization: a geometric framework for learning from labeled and unlabeled examples. J. Mach. Learn. Res. **7**, 2399–2434 (2006)

4. K. Bennett, A. Demiriz, Semi-supervised support vector machines, in *Proceedings of the 1998 Conference on Advances in Neural Information Processing Systems II* (1999)
5. S. Berchtold, C. Bohm, H. Kriegel, The pyramid-technique: towards breaking the curse of dimensionality, in *Proceedings ACM SIGMOD International Conference on Management of Data* (1998)
6. A. Blum, S. Chawla, Learning from labeled and unlabeled data using graph Mincuts, in *Proceedings of the Eighteenth International Conference on Machine Learning (ICML '01)* (2001)
7. A. Blum, T. Mitchell, Combining labeled and unlabeled data with co-training, in *Proceedings of the Eleventh Annual Conference on Computational Learning Theory (COLT' 98)* (1998)
8. O. Chapelle, B. Schlkopf, A. Zien, *Semi-Supervised Learning*, 1st edn. (The MIT Press, Cambridge, 2010)
9. A. Dempster, N. Laird, D. Rubin, Maximum likelihood from incomplete data via the EM algorithm. J. R. Stat. Soc. Ser. B **39**(1), 1–38 (1977)
10. C. Elkan, K. Noto, Learning classifiers from only positive and unlabeled data, in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '08)* (2008)
11. X. Kong, J. Zhang, P. Yu, Inferring anchor links across multiple heterogeneous social networks, in *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management (CIKM '13)* (2013)
12. B. Liu, Y. Dai, X. Li, W. Lee, P. Yu, Building text classifiers using positive and unlabeled examples, in *Third IEEE International Conference on Data Mining* (2003)
13. L. Manevitz, M. Yousef, One-class SVMs for document classification. J. Mach. Learn. Res. **2**, 139–154 (2002)
14. Y. Ren, C.C. Aggarwal, J. Zhang, Meta diagram based active social networks alignment. CoRR (2019). http://arxiv.org/abs/1902.04220
15. K. Rose, Deterministic annealing for clustering, compression, classification, regression, and related optimization problems. Proc. IEEE **86**(11), 2210–2239 (1998)
16. B. Settles, Active learning literature survey. Computer sciences technical report, University of Wisconsin–Madison (2009)
17. A. Yuille, A. Rangarajan, The concave-convex procedure (CCCP), in *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic (NIPS'01)* (2002)
18. J. Zhang, P. Yu, Integrated anchor and social link predictions across partially aligned social networks, in *Proceedings of the 24th International Conference on Artificial Intelligence (IJCAI'15)* (2015)
19. J. Zhang, P. Yu, Z. Zhou, Meta-path based multi-network collective link prediction, in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '14)* (2014)
20. J. Zhang, J. Chen, J. Zhu, Y. Chang, P. Yu, Link prediction with cardinality constraints, in *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining (WSDM '17)* (2017)
21. J. Zhang, J. Chen, S. Zhi, Y. Chang, P. Yu, J. Han, Link prediction across aligned networks with sparse low rank matrix estimation, in *2017 IEEE 33rd International Conference on Data Engineering (ICDE)* (2017)
22. D. Zhou, O. Bousquet, T. Lal, J. Weston, B. Schölkopf, Learning with local and global consistency, in *Proceedings of the 16th International Conference on Neural Information Processing Systems (NIPS'03)* (2003)
23. X. Zhu, Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison (2005)
24. X. Zhu, Semi-supervised Learning with Graphs. PhD thesis, Pittsburgh, PA, USA, 2005. AAI3179046
25. J. Zhu, J. Zhang, Q. Wu, Y. Jia, B. Zhou, X. Wei, P. Yu, Constrained active learning for anchor link prediction across multiple heterogeneous social networks. Sensors **17**(8), 1786 (2017)