Learning the Consistent Behavior of Common Users for Target Node Prediction across Social Networks

Shan-Hung Wu SHWU@CS.NTHU.EDU.TW
Hao-Heng Chien HHCHIEN@NETDB.CS.NTHU.EDU.TW
Kuan-Hua Lin KHLIN@NETDB.CS.NTHU.EDU.TW
National Tsing Hua University, ROC

Philip S. Yu PSYU@CS.UIC.EDU
University of Illinois at Chicago, USA

Abstract

We study the *target node prediction* problem: given two social networks, identify those nodes/users from one network (called the *source network*) who are likely to join another (called the *target network*, with nodes called target nodes). Although this problem can be solved using existing techniques in the field of cross domain classification, we observe that in many real-world situations the cross-domain classifiers perform sub-optimally due to the *heterogeneity* between source and target networks that prevents the knowledge from being transferred. In this paper, we propose learning the consistent behavior of common users to help the knowledge transfer. We first present the *Consistent Incidence Co-Factorization* (CICF) for identifying the *consistent users*, i.e., common users that behave consistently across networks. Then we introduce the *Domain-UnBiased* (DUB) *classifiers* that transfer knowledge *only through* those consistent users. Extensive experiments are conducted and the results show that our proposal copes with heterogeneity and improves prediction accuracy.

1. Introduction

With the popularity of online social networking and its immense influence, more and more online services (including new startups like Foursquare and old players such as Gmail, Flicker, YouTube, etc.) "socialize" themselves by allowing their users to link to each other either directly or through existing social networks (e.g., Facebook, Twitter, Google+, etc.), aiming to strengthen the coherence amongst existing users and to attract newcomers via virus marketing. One important task to the owner of such a service is to identify those users in other social networks who are willing to join in, so that the advertisements can be placed more precisely and economically. Meanwhile, to another social network, the owner may identify existing common users either a) explicitly (e.g., Foursquare and Twitter, given that Foursquare allows their users to fill in Twitter IDs in their account pages); b) through common e-mail addresses (aided by the single-sign-one protocols such as OpenID); c) by affiliation (e.g., YouTube and Google+, Flickr and Yahoo! Answers, and between many online travel agencies); or d) via other data mining techniques (Narayanan & Shmatikov, 2009).

Define a *content-rich social network* as $G = (V, E)$, where $V = \{v_i\}_{i=1}^{n}$ is the set of vertices/nodes representing users, $E = \{e_i\}_{i=1}^{m}$ is the set of edges representing user relationship (e.g., friendship, following, coauthorship, etc.), and each vertex v (and/or edge e) is associated with a content/feature vector $\mathbf{content}(v) \in \mathbb{R}^w$, $w \in \mathbb{Z}^+$ (and/or $\mathbf{content}(e)$), denoting, for example, the bag of words of a user profile (and/or interaction logs between the endpoint users). We study the *target node prediction problem*:

Problem 1. Given two snapshots of content-rich social networks $G^{(t)} = (V^{(t)}, E^{(t)})$ and $G^{(s)} = (V^{(s)}, E^{(s)})$ with identifiable common users $C = V^{(t)} \cap V^{(s)} \neq \emptyset$, where $G^{(t)}$ and $V^{(t)}$ (resp. $G^{(s)}$ and $V^{(s)}$) are called the *target* (resp. *source*) *network* and *nodes/users* respectively, find a function $f : V^{(s)} \to \mathbb{R}$ that scores a node in $V^{(s)} \backslash C$ higher if it is more likely to become a target node in the future.

Note that different communities (e.g., fans of a Page in Facebook) in a single social network can also be treated as different "networks." Then, the target node prediction can be applied to identify users who would like to join a

Proceedings of the 31^{st} International Conference on Machine Learning, Beijing, China, 2014. JMLR: W&CP volume 32. Copyright 2014 by the author(s).

new community (and receive the ads from that community, which is an important revenue source of Facebook now).

The cross-domain classification, which aims to classify the unlabeled instances in one domain using the labeled ones in another domain, can be employed to solve the above problem if we regard the target users and users in $V^{(s)} \backslash C$ as the positive and unlabeled instances in two respective domains.[1] The works in this field can be divided into those minimizing empirical error (Long et al., 2012a; Zhuang et al., 2011) and those regularizing geometry properties (He et al., 2009; Ling et al., 2008; Pan et al., 2011; Wang & Mahadevan, 2009; 2011). Zhuang et al. (Zhuang et al., 2011) exploit the association between the latent factors of instance features and factors of labels to align distributions of instances in different domains. Long et al. (Long et al., 2012a) find "common factors of factors" which associate the feature factors and label factors to transfer knowledge. Ling and Pan et al. (Ling et al., 2008; Pan et al., 2011) regularize geometry properties in each network using spectral clustering and Wang et al. (Wang & Mahadevan, 2009; 2011) map the features of all domains into a low dimensional manifold while preserving the geometric structure in each domain. He et al. (He et al., 2009) propagate the label information across graphs using the graph Laplacian. Recent, Long et al. (Long et al., 2012b) take the two tasks simultaneously: minimizing the empirical error and regularizing geometric structures, and demonstrate improved performance.

However, we find that in many real-world situations the cross-domain classifiers are outperformed by some much simpler classifiers taking only the inputs from a single domain $V^{(s)}$ (e.g., Semi-Supervised Learning (SSL)-based methods, by labeling nodes in C as positive) due to the following reasons. Since the target and source networks may be formed by different reasons, have different focuses, and evolve distinctively, they may be *heterogeneous*, *either extrinsically or intrinsically*. The two networks may be heterogeneous extrinsically in that their link structures and contents may be very different. Nevertheless, existing cross-domain classifiers find the common latent factors using the *overlap of contents* (of vertices), thereby rendering sub-optimal results in the presence of extrinsic heterogeneity. On the other hand, the two networks may be heterogeneous intrinsically in that their users may sign/participate in for different reasons. Each network may reveal only a small fraction of a user's latent interests/character, and the knowledge on the user is *not necessarily transferable* across the networks. If we learn from the target users equally, those "inconsistent" users may introduce bias during the knowledge transfer. Moreover, due to the heterogeneity of networks, a pair of latent factors that help knowledge transfer is not necessarily a common latent factor. It is also possible that they are *complementary* to each other.

In this paper, we propose learning the consistent behavior of common users to help the knowledge transfer between social networks. We first present the *Consistent Incidence Co-Factorization* (CICF), a regularized co-factorization of the incidence matrices of the source and target networks, that finds the pairs of latent factors in respective networks *explaining the consistent behavior of common users*. The factors in each pair can be either common or complementary with each other. More importantly, they can be used to identify the *consistent users*, i.e., common users that behave consistently across networks. Then we introduce the *Domain-UnBiased* (DUB) *classifiers* that allow the knowledge to be transferred *only through* those consistent users. A DUB classifier can be either augmentation of an existing classifier such that it labels common users positively according to their consistency scores, or a completely new classifier that makes use of the found latent factors directly.

The proposed learning technique copes with both the extrinsic and intrinsic heterogeneity. In CICF, the pairs of latent factors are found using the *overlap of vertices* (i.e., common nodes C), rather than the overlap of feature spaces of vertices/edges as in traditional cross-domain classifiers. So, our work can be readily applied to two networks where the link structures and contents are very different. Furthermore, by focusing on the consistent behavior, the DUB classifiers are robust to the bias of individual networks during the knowledge transfer.

Extensive simulations are conducted on both synthetic and real datasets and the results show that a DUB classifier augmenting LapRLS (Belkin et al., 2006) can improve the performance of original LapRLS up to about 35% in both precision and recall. And another DUB classifier which directly makes use of the latent factors found by CICF outperforms GCMF (Long et al., 2012b) up to 60% in precision and 58% in recall.

Further related work. Different from most existing work on transfer learning, we do *not* assume the extrinsic/intrinsic overlapping of link structure/feature spaces in two domains. Although not directly comparable, some other work on graph-based cross-domain learning are relevant to our study. Tang et al. (Tang et al., 2009) lay the fundamentals of clustering on multiple graphs. However, this work assumes that the vertex set V in different networks are identical and therefore is not applicable to our problem. Tang et al. (Tang et al., 2012) predict the next coauthorship between authors from different domains in the future. If we treat the authors, coauthorship, and documents as the vertices, edges, and edge contents respectively, the problem in this study is to predict new edges between existing vertices

[1] Be aware of the naming conflict: the source and target networks are indeed the target and source domains by convention of cross-domain classification respectively.

	No.	Target	Source	HLS	HF	IH
DBLP	1	A&T	DM	0.932	0.912	0.452
	2	DB	DM	0.263	0.288	0.439
	3	D&PC	DM	0.909	0.895	0.303
	4	HCI	DM	0.912	0.912	0.184
	5	ML&PR	AI	0.323	0.342	0.211
	6	NL&S	AI	0.848	0.88	0.214
	7	NL&S	ML&PR	0.696	0.736	0.472
	8	PL	SE	0.407	0.448	0.367
	9	WWW	DM	0.776	0.82	0.386
Amazon	10	BM	HB	0.442	0.334	0.567
	11	LF	CB	0.728	0.72	0.433
	12	SF	T	0.58	0.559	0.552
	13	SB	T	0.961	0.93	0.478
	14	CG	HB	0.87	0.829	0.403

Figure 1. Datasets and degree of extrinsic/intrinsic heterogeneity.

across networks, which is different from our problem (to predict vertices). Moreover, the input contains the cross-network edges and contents that are not available in our settings. Similarly, Lu et al. (Lu et al., 2010) focus on predicting edges. Jiang et al. (Jiang et al., 2012) recommend items from different item networks to users. Again, the assumed edges between the items and users are not available in our problem. Zhong et al. (Zhong et al., 2012) use the link structure and historical user behavior data (which records the user-item pairs) to predict user behavior on items in the future. Both the input and problem definition are different from our settings. Studies (Duan et al., 2012; Glorot et al., 2011) investigate the preprocessing of data from different domains, either by feature augmentation or unsupervised deep learning, to help the knowledge transfer. However, these works assume the overlapping of feature spaces, which may not hold in our settings. Furthermore, they do not consider the link structures nor common users in social networks.

2. Evidence of Heterogeneity and Challenges

We use two real datasets, DBLP citation network (Tang et al., 2008) and Amazon product co-purchasing network (Leskovec et al., 2007), to demonstrate the extrinsic and intrinsic heterogeneity.[2]

[2]The DBLP citation network contains 1,572,277 papers and 2,084,019 citation relations. Each paper is associated with abstract, authors, year, venue, and title. We use the categories listed in Microsoft Academic Search to divide the venues into different networks: Algorithm & Theory (A&T), Data Mining (DM), Databases (DB), Distribution & Parallel Computing (D&PC), Human-Computer Interaction (HCI), Machine Learning & Pattern Recognition (ML&PR), Artificial Intelligence (AI), Natural Language & Speech (NL&S), Programming Language (PL), Software Engineering (SE), World Wide Web (WWW). For each category, we extract a coauthor-network formed by all papers with corresponding authors published during 2006 to 2010. The nodes and edges are the authors and coauthorship respectively. We assign the content of an edge as the term frequencies of abstracts of all papers coauthored by the two endpoint users. Then, 9 pairs of networks are selected to be the target and the source networks.

The Amazon product co-purchasing network contains product metadata and review information about 548,552 different products. For each product, we use the following information: detailed product categorization and product reviews including time, customer, rating, etc. To fit the dataset for the target node prediction problem, we only extract the products under *Books* category and treat the following subcategories as different networks: Biographies and Memoirs (BM), Children's Books (CB), History Books (HB), Literature & Fiction Books (LF), Science fiction and Fantasy books (SF), Teens (T), Science Books (SB), and Comics & Graphic novels (CG). Each network contains all reviewers that have reviewed at least one book of the corresponding subcategory during 2003 to 2005 and two reviewers have a link if they have co-reviewed at least one book. We assign the content of an edge as the term frequencies of the book description crawled from the Amazon website. Finally, 5 pairs of networks are selected to be the target and the source networks.

The two networks $G^{(t)}$ and $G^{(s)}$ may be heterogeneous extrinsically in that their link structures $E^{(t)}$ and $E^{(s)}$ may be very different, and the $\mathbf{content}^{(t)}(e) \in \mathbb{R}^{w^{(t)}}$ and $\mathbf{content}^{(s)}(e) \in \mathbb{R}^{w^{(s)}}$ of a common edge e, $e \in E^{(t)} \cap E^{(s)}$, may consist of different features. We define the *Heterogeneity of Link Structures* (HLS) as the average degree of neighbor overlap for each common user:

$$1 - \frac{1}{|C|} \sum_{v \in C} \frac{|N^{(t)}(v) \cap N^{(s)}(v)|}{|N^{(t)}(v) \cup N^{(s)}(v)|},$$

where $C = V^{(t)} \cap V^{(s)}$ is the set of common users and $N^{(t)}(v)$ (resp. $N^{(s)}(v)$) is the set of neighbors of a common user v in the target (resp. source) network. We also define the *Heterogeneity of Feature spaces* (HF) as

$$1 - \frac{|F^{(t)} \cap F^{(s)}|}{|F^{(t)} \cup F^{(s)}|},$$

where $F^{(t)}$ (resp. $F^{(s)}$) is the feature set of the target (resp. source) network.

On the other hand, the two networks may be heterogeneous intrinsically in that their users may join in for different reasons. We measure the degree of *Intrinsic Heterogeneity* (IH) as follows: (1) adopt the spectral clustering on two networks separately;[3] and then (2) calculate IH by using the 1-*cluster purity*,

$$1 - \frac{1}{|C|} \sum_{i} \max_{j} |U_i^{(t)} \cap U_j^{(s)}|,$$

[3]We use spectral clustering because 1) it is based on the latent factors like our baseline GCMF; 2) it is popular and has different explanations (from either the graph-cut, random walk, or perturbation points of view (Von Luxburg, 2007)) for the objective. We set the number of clusters $k = 10$.

where $U_i^{(s)}$ and $U_j^{(t)}$ are the clusters found by the spectral clustering (nodes only in a single network are excluded). The more common users that are grouped into different clusters in the two networks, the higher the IH.

The degree of heterogeneity in different datasets are summarized in Table 1. As we can see, the HLS, HF, and IH are rather high in many real datasets. However, most of the existing studies on cross-domain classification (He et al., 2009; Ling et al., 2008; Long et al., 2012a;b; Pan et al., 2011; Wang & Mahadevan, 2009; 2011; Zhuang et al., 2011) find common latent factors by assuming the *overlap of feature spaces*, therefore result in sub-optimal performance. In addition, a high IH implies that users may join different networks with different reasons, hence their behaviors may not be consistent across networks. However, existing studies on cross-domain classification learn from the positive labels of target users *equally*, and those "inconsistent" users introduce bias during the knowledge transfer.

In fact, we observe that in many situations the existing cross domain predictors are outperformed by simple classifiers such as the Semi-Supervised Learning (SSL)-based methods that utilize the information in only the source network by regarding the common and source users as the positive and unlabeled examples respectively. We choose the GCMF (Long et al., 2012b) and LapRLS (Belkin et al., 2006) as the representative cross-domain and SSL-based classifiers due to their superior performance. Figs. 3 and 4 show the accuracy of GCMF and LapRLS on synthetic networks with different degrees of extrinsic and intrinsic heterogeneity (see Section 5 for more details about the synthetic networks). In Fig. 3, the performance of GCMF keeps decreasing when the degree of extrinsic heterogeneity gets larger. We believe this is because that the extrinsic heterogeneity prevents GCMF from finding the correct latent factors of content features. In Fig. 4, the GCMF performs well when the common users in two networks are highly consistent but is outperformed by LapRLS when the degree of intrinsic heterogeneity becomes larger than 0.3. Since the heterogeneity is common in practice, it is crucial to have a new technique to improve the knowledge transfer between the content-rich social networks.

3. Consistent Incidence Co-Factorization

In the presence of extrinsic and intrinsic heterogeneity, it is hard to transfer knowledge via links, contents, or common latent factors. The only parts the two networks $G^{(t)}$ and $G^{(s)}$ are in common are the common users $V^{(t)} \cap V^{(s)}$. This motivates us to first identify the knowledge transferable via the common users, and then, based on the knowledge, make better predictions. In this section, we introduce the *Consistent Incidence Co-Factorization* (CICF) for the first step. Section 4 explains the second.

We model CICF using the framework of regularized co-factorization. Given a constant $k \in \mathbb{Z}^+$, our goal is to embed the spaces of vertices and edges in target and source networks into four low-dimensional latent spaces respectively of the same dimension k, and obtain four latent factor matrices $\mathbf{V}^{(t)} \in (\mathbb{R}^+)^{k \times n^{(t)}}$, $\mathbf{V}^{(s)} \in (\mathbb{R}^+)^{k \times n^{(s)}}$, $\mathbf{E}^{(t)} \in (\mathbb{R}^+)^{k \times m^{(t)}}$, and $\mathbf{E}^{(s)} \in (\mathbb{R}^+)^{k \times m^{(s)}}$. We require the elements of $\mathbf{V}^{(g)}$ and $\mathbf{E}^{(g)}$, $g = t$ or s, to be non-negative such that $\mathbf{V}^{(g)}_{i,j}$ (resp. $\mathbf{E}^{(g)}_{i,j}$) denotes the degree that user (resp. edge) j *holds* latent factor i, or equivalently, degree that latent factor i *covers* user (resp. edge) j. Without loss of generality, we assume that the columns of $\mathbf{V}^{(t)}$ and $\mathbf{V}^{(s)}$ from left correspond to the common users. We formulate the objective of CICF as follows:

$$\arg\min_{\Theta} \sum_g \|\mathbf{G}^{(g)} - \mathbf{E}^{(g)\top}\mathbf{V}^{(g)}\|^2_{\mathcal{F}} + \alpha\Omega_c(\Theta) + \beta \sum_g \Omega_l(\Theta^{(g)}), \tag{1}$$

where $\Theta = \bigcup_g \Theta^{(g)}$, $\Theta^{(g)} = \{\mathbf{V}^{(g)}, \mathbf{E}^{(g)} : \mathbf{V}^{(g)} \geq \mathbf{O} \text{ and } \mathbf{E}^{(g)} \geq \mathbf{O}\}$, are parameters to solve, $\mathbf{G}^{(g)} \in \{0,1\}^{m^{(g)} \times n^{(g)}}$ is the incidence matrix such that $\mathbf{G}^{(g)}_{i,j} = 1$ if $e_i^{(g)}$ is incident to $v_j^{(g)}$ and 0 otherwise, and $\|\cdot\|_{\mathcal{F}}$ is the Frobenius norm. The term $\Omega_l(\Theta^{(g)})$ is a regularizer that penalizes the variations of latent factors in an individual network along its structure and/or contents; while $\Omega_c(\Theta)$ is another regularizer penalizing the variations of the latent factors across the two networks along some consistency measure for common users (to be explained later). The hyperparameters α and β control the trade-off between the empirical error (the first term) and the effect of regularizers.

The regularizer $\Omega_l(\Theta^{(g)})$ for single network has been well-studied (for example, in (Belkin et al., 2006; Qi et al., 2012)). Here we adopt a simple form:

$$\sum_{i=1}^{k} \mathbf{E}^{(g)}_{i,:}\mathbf{L}(\mathbf{K}(V^{(g)}))\mathbf{E}^{(g)\top}_{i,:} = \mathrm{tr}(\mathbf{E}^{(g)}\mathbf{L}(\mathbf{K}(V^{(g)}))\mathbf{E}^{(g)\top}), \tag{2}$$

where $\mathbf{L}(\mathbf{K}(V^{(g)}))$ is the Laplacian matrix of a kernel matrix $\mathbf{K}(V^{(g)}) \in \mathbb{R}^{n^{(g)} \times n^{(g)}}$, and $\mathbf{K}(V^{(g)})_{i,j}$ equals to 0 if $v_i^{(g)}$ and $v_j^{(g)}$ are not adjacent; otherwise $\exp(-\frac{\|\mathbf{content}(v_i) - \mathbf{content}(v_j)\|^2}{2\sigma^2})$, a Gaussian RBF with σ constant. This makes each latent factor of users $\mathbf{V}^{(g)}_{i,:}$, $1 \leq i \leq k$, smooth along similar users who are connected. Note that since we factorize the incidence matrices, whenever the contents $\mathbf{content}(e^{(g)})$ of edges are available, we can write $\Omega_l(\Theta^{(g)})$ as

$$\mathrm{tr}(\mathbf{E}^{(g)}\mathbf{L}(\mathbf{K}(E^{(g)}))\mathbf{E}^{(g)\top}) \tag{3}$$

analogously.[4] Therefore, CICF can make use of the edge contents, which usually encode peer-specific information

[4]The contents of edges are usually available to the owner

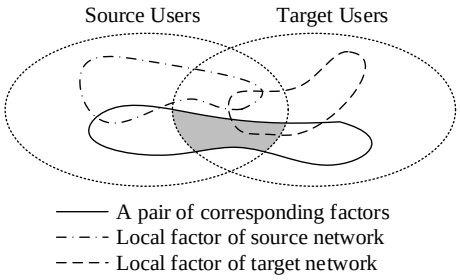

Figure 2. CICF aims to find the pairs of corresponding latent factors in the two networks, each explaining some consistent behavior of the covered common users (shaded).

not available in the contents of individual nodes (Qi et al., 2012; Yang et al., 2009; Zhou et al., 2009), to find better latent factors.

We write $\Omega_c(\Theta)$ as

$$\|\mathbf{V}^{(s)}\mathbf{P}^{(s)} - \mathbf{V}^{(t)}\mathbf{P}^{(t)}\|_{\mathcal{F}}^2, \tag{4}$$

where $\mathbf{P}^{(g)} \in \{0,1\}^{n^{(g)} \times |V^{(g)} \cap V^{(g)}|}$ is a rectangular diagonal matrix, $\mathbf{P}_{i,i} = 1$, used to extract the columns of $\mathbf{V}^{(g)}$ corresponding to the common users. The effect of Eq. (4) is two-fold. First, it can be written as $\sum_{i=1}^{k} \|\mathbf{V}_{i,:}^{(t)}\mathbf{P}^{(t)} - \mathbf{V}_{i,:}^{(s)}\mathbf{P}^{(s)}\|^2$. Minimizing this term requires two latent factors with the same index i in respective networks to cover the same set of common users, and therefore creates their mutual correspondence, as shown in Fig. 2. Each $(\mathbf{V}_{i,:}^{(t)}, \mathbf{V}_{i,:}^{(s)})$ pair, denoted as φ_i, can be seen as a pair of *corresponding latent factors* from respective networks. Moreover, the two corresponding latent factors can be either one *common* latent factor spanning through the two networks or two *complementary* factors. For example, researchers who are active in two research fields (networks) may hold either the same or complementary expertise (factors). It is important to note that, since the k pairs of corresponding latent factors are found using the overlap of vertices rather than the overlap of feature spaces of vertices/edges, the CICF can cope with extrinsic heterogeneity and be readily applied to two networks where the link structures and contents are very different.

Eq. (4) can also be written as $\sum_{j=1}^{|V^{(t)} \cap V^{(s)}|} \|\mathbf{V}_{:,j}^{(t)} - \mathbf{V}_{:,j}^{(s)}\|^2$. By minimizing this term, we seek only those latent factors that *explain the consistent behaviors* of common users. Notice that if we find a pair of corresponding latent factors φ_i, then a common user v_j covered by φ_i must be consistent in terms of this factor (i.e., $\mathbf{V}_{i,j}^{(t)} = \mathbf{V}_{i,j}^{(s)}$). The corresponding latent factors are learned from those consistent users only.

of a social network via logs. In case both $\mathbf{content}(v^{(g)})$ and $\mathbf{content}(e^{(g)})$ are available in a network, we adopt Eq. (3) by first pushing all features of a vertex v to the $\mathbf{content}(e)$ of all e's incident to v.

Solving the objective. The objective function Eq. (1), with the non-negative constraints on $\mathbf{V}^{(g)}$ and $\mathbf{E}^{(g)}$, is difficult to solve using traditional optimization methods such as the multiplicative update approach (Seung & Lee, 2001; Yang & Oja, 2010) due to the fluctuation problem in convergence (Yang & Oja, 2011; Zhang et al., 2012). To avoid this problem, we transform Eq. (1) into a *Projective Non-negative Matrix Factorization* (PNMF) (Yang & Oja, 2010) problem, then devise an *auxiliary function* (Seung & Lee, 2001) and obtain an iterative update rules for $\mathbf{E}^{(t)}$ and $\mathbf{E}^{(s)}$ respectively. We then update $\mathbf{E}^{(t)}$ and $\mathbf{E}^{(s)}$ alternately until convergence. Empirically, 8 to 15 iterations suffice to reach convergence. For more details, please refer to the supplementary materials.

4. Domain-Unbiased Classifiers

The reason that the intrinsic heterogeneity prevents the knowledge from being transferred is because that it introduces the *domain bias*:

Definition 2. Let $\{\mathcal{X}^{(i)}\}_i$ and $\{\mathcal{Y}^{(i)}\}_i$ be sets of instance and label spaces. We say a cross-domain classification task $\{\mathcal{Y}^{(p)}, f^{(p)}\}$, where $f^{(p)}$ is trained on different datasets $\{(\{\boldsymbol{x}^{(i,j)}\}_j, \{y^{(i,j)}\}_j)\}_i$,[5] $\boldsymbol{x}^{(i,j)} \in \mathcal{X}^{(i)}$ and $y^{(i,j)} \in \mathcal{Y}^{(i)}$, is *biased by a domain* q, $q \neq p$, if $f^{(p)}$ is systematically incorrect when predicting the correct labels for $\boldsymbol{x}^{(p)} \in \mathcal{X}^{(p)}$ due to $(\{\boldsymbol{x}^{(q,j)}\}_j, \{y^{(q,j)}\}_j)$.

Consider a local latent factor in one network shown in Fig. 2. If a cross-domain classifier makes predictions in the opposite network based on this factor, then the classifier can suffer from domain bias because the factor might not explain behavior in the other network and incur systematic errors. For example, GCMF (Long et al., 2012b) finds the "latent factors of factors" for knowledge transfer. But these latent factors of factors may be biased if they are required to explain the (linear) relationship between local factors in different networks, which does not exist indeed.

Definition 3. From the above definition, the task $\{\mathcal{Y}^{(p)}, f^{(p)}\}$ is said to be *domain-unbiased* if it is not biased by any domain q, $q \neq p$.

As Fig. 2 implies, a cross-domain classifier can eliminate the domain bias due to local factors by making predictions *only based on the corresponding latent factors* found by CICF. Depending on how the corresponding latent factors are used, we introduce two domain unbiased classifiers as follows. While other, more sophisticated, classifiers are possible, we show in Section 5 that these two classifiers suffice to make significant improvement.

[5] The labels $\{y^{(i,j)}\}_j$ can be an empty set, as in $V^{(s)} \backslash C$.

4.1. Label Augmentation

One way to make domain-unbiased predictions is to transfer knowledge through those common users with high *consistency*.

Definition 4 (Consistency)**.** Given $\mathbf{V}^{(t)}$ and $\mathbf{V}^{(s)}$. The *consistency* of a user v_c, denoted as $cnst(v_c)$, is defined as the unnormalized similarity between $\mathbf{V}^{(t)}_{:,c}$ and $\mathbf{V}^{(s)}_{:,c}$.

E.g., $cnst(v_c) = \mathbf{V}^{(t)\top}_{:,c}\mathbf{V}^{(s)}_{:,c}$. The similarity is unnormalized in order to take the absolute degrees that the user belongs to the k communities into account.

Once obtaining the consistency measure, we can pair up the CICF with an existing classifier by labeling the common user using their consistency scores. For example, we can rewrite the first term of the objective of LapRLS (Belkin et al., 2006)

$$\arg\min_{f} \sum_{v\in V^{(t)}\cap V^{(s)}} (1-f(v))^2 + \alpha\|f\|^2 + \beta\mathbf{f}^T\mathbf{L}(\mathbf{K}(V^{(s)}))\mathbf{f},$$

as $\sum_{v\in V^{(t)}\cap V^{(s)}}(cnst(v)-f(v))^2$. In this way, the CICF augments existing classifiers by enabling/improving the knowledge transfer.

4.2. Direct Predictions based on Corresponding Latent Factors

Recall that a pair of corresponding latent factors $\varphi_c = (\mathbf{V}^{(t)}_{c,:}, \mathbf{V}^{(s)}_{c,:})$ for some c can be either a common factor spanning both networks or complementary factors. A source user that is covered by $\mathbf{V}^{(s)}_{i,:}$ may be a good candidate for the target network. For example, a target network that has attracted researchers in machine learning may continue to attract other researchers in the same field. In addition, if the network has attracted researchers with expertise in both machine learning and biology, it may also attract other biologists.

Definition 5 (Correspondence)**.** Given $\mathbf{V}^{(t)}$ and $\mathbf{V}^{(s)}$. The *correspondence* of a pair of corresponding latent factors $\varphi_c = (\mathbf{V}^{(t)}_{c,:}, \mathbf{V}^{(s)}_{c,:})$, denoted as $crsp(\varphi_c)$, is defined as the unnormalized similarity between $\mathbf{V}^{(t)}_{c,:}\mathbf{P}^{(t)}$ and $\mathbf{V}^{(s)}_{c;,:}\mathbf{P}^{(s)}$.

Let $\mathbf{crsp} = [crsp(\varphi_1), crsp(\varphi_2), \cdots, crsp(\varphi_k)]^\top \in \mathbb{R}^{k\times 1}$, we define a new prediction function as $f(v_i) = \mathbf{crsp}^\top\mathbf{V}^{(s)}_{:,i}$, which scores a source user v_i higher if he/she holds more highly correspondence factors and therefore has more reasons (supported by the consistent behavior) to join the target network.

Solving the objectives of the above classifiers is straightforward therefore omitted.

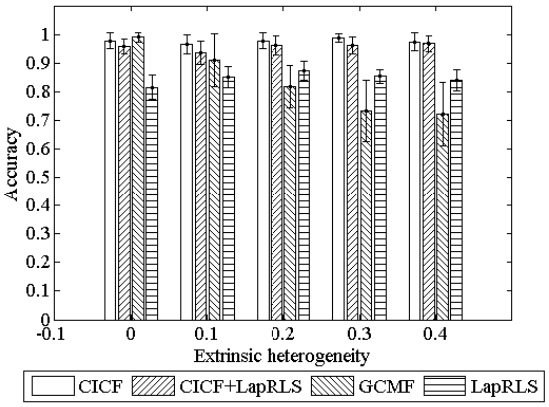

Figure 3. Accuracy with different degrees of extrinsic heterogeneity.

5. Performance Evaluation

We study the performance of the two DUB classifiers proposed in Section 4, denoted by CICF+LapRLS[6] and CICF respectively. These classifiers are compared with the LapRLS (Belkin et al., 2006) and GCMF (Long et al., 2012b), which have shown superior performance in the fields of semi-supervised learning and cross domain classification respectively.

5.1. Synthetic Networks

To understand the performance of various algorithms given different degree of heterogeneity, we generate synthetic networks. We first define the number of common users nC (that exist in both networks), and the numbers of total users $nTarget$ and $nSource$ in respective networks. We define the latent factors of users in ground truth. A factor is said to be *local* if it covers users in only one network; and *common* if it covers users in both networks and explains universal structures/contents. A common factor is a special case of a pair of corresponding factors in Fig. 2. In our settings, only the common factors can pull users from the source to target network. So here our target is to predict users in $V^{(s)}\backslash C$ that are covered by common factors.

There are cf common latent factors across the networks and lf local factors in each network. Each latent factor, either local or common, has predefined link structure between the covered users as well as predefined edge contents. Regarding the link structure, each user is connected to at least 10% of the nodes in the same latent factor to ensure the community structure. Users in the same latent fac-

[6]We do not pair the CICF with GCMF because they are both based on the latent factors, and CICF does not improve the way GCMF finds the corresponding latent factors.

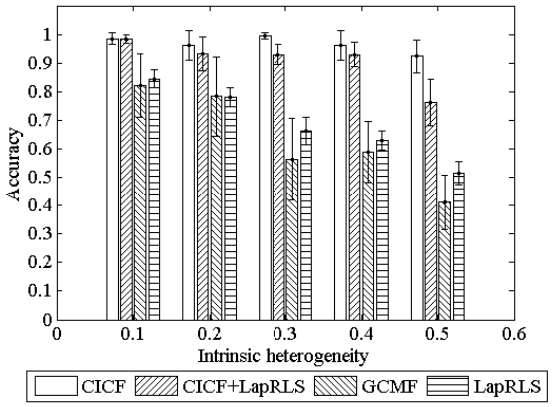

Figure 4. Accuracy with different degrees of intrinsic heterogeneity.

tor are also forced to form a connected component. Common users, if assigned to a common factor (to be discussed later), have the same link structure in the two networks. To generate the content vector $\mathbf{content}(e)$ on each edge, we define a feature pool with 10 features for each latent factor. The content of a link has all the features of a feature pool iff the two adjacent users both have the factor where the feature pool belongs to. Then, the value of each feature is sampled from a normal distribution with mean 30 and variance 4.

We simulate the extrinsic heterogeneity by introducing a degree of noises $dNoise$,[7] based on which the noise values (sampled from a normal distribution with mean 30 and variance 4) are added to $\#features \times dNoise$ features randomly sampled from all features in each network. As $dNoise$ gets larger, the feature sets of a common factor in two networks becomes more different due to the randomness.

We randomly assign k latent factors to each user, by random sampling with replacement. To simulate that a common factor can pull users from the source to target network, each sampling requires that a common user has a probability $pConsistent$ to choose a common factor. Hence the opposite, $rp = 1 - pConsistent$, is the probability that a common user behaves inconsistently and simulates the random pull. The larger the rp, the more the intrinsic heterogeneity.

Here we fix $nC = 40$, $nTarget = 50$, $nSource = 100$, $cf = 2$, $lf = 4$, $dNoise = 0.05$, and $k = 2$, and varies $dNoise$ and rp to simulate different degrees of extrinsic and intrinsic heterogeneity respectively. Each reported result is an average of 15 runs. We also report the 90% confidence intervals (following the student's t-distribution with 14 degrees of freedom). Fig. 3 shows the accuracy (or simply the true positive rate, as we have only positive testing instances) achieved by different algorithms under different degrees of extrinsic heterogeneity. We set $pConsistent = 0.9$. As we can see, the CICF standalone consistently outperforms the other algorithms. In addition, when pairing up with LapRLS the CICF improves the performance of LapRLS by about 10%. Notice that the GCMF does not perform well because it finds the corresponding latent factors using the overlap of content features, which is biased when the the extrinsic heterogeneity is large. GCMF also gives higher variances, implying that the degradation is mainly resulted from the sharp decreases when feature clusters cannot be identified successfully.

Fig. 4 shows the performance of the algorithms under different degrees of intrinsic heterogeneity. We set $dNoise = 0.2$. Again, the CICF standalone outperforms the other algorithms in most cases, although the performances of all algorithms decrease as the degree of intrinsic heterogeneity gets larger. The CICF improves the performance of LapRLS up to 48% at $rp = 0.4$. We believe that this is because CICF can correctly identify the consistent users, and smoothing f only on these consistent users that help make better prediction.

5.2. Results from Real Networks

In this section, we demonstrate the performance of our algorithm using the real datasets introduced in Section 2 based on DBLP citation network (Tang et al., 2008) and Amazon product co-purchasing network (Leskovec et al., 2007).

Because each author may write papers and each reviewer may review books in different categories, we can identify common users in each dataset via common email addresses or Amazon IDs. We also observe in datasets that the nodes who have few publications/reviews tend to join the networks randomly. With these users, all algorithms give very poor performance because the test set contains random pulls mostly, and the behavior of a testing author who don't have enough publications can neither be inferred from link structures nor contents. Therefore, for each pair of networks we first prune the nodes having fewest publications/reviews and select the common authors with largest numbers of neighbor overlap in two networks as the test and validation sets.

Regarding the parameter tuning, for CICF we seek k in $\{3, 5, 7, 9\}$ and run the grid search on α and β. For GCMF, we set $\lambda = \gamma$ and seek them in $[10^{-1}, 10^3]$ with log step 1, k in $[1, 5]$ with step 1, and p in $[2, 10]$ with step 2. For LapRLS, we find α and β from $[2^{-8}, 2^8]$ with log step 1.

[7] Adding noise is not the only option to simulate the extrinsic heterogeneity. For example, randomly rewire some edges may do the same effect.

Table 1. Performance comparison.

		CICF		CICF+LapRLS		GCMF		LapRLS	
	Dataset	Recall	Precision	Recall	Precision	Recall	Precision	Recall	Precision
DBLP	1	0.3333	0.2667	**0.5**	**0.4**	0.1042	0.0833	0.0625	0.05
	2	**0.7424**	**0.8167**	0.6364	0.7	0.0909	0.1	0.6364	0.7
	3	0.1548	0.2167	**0.2262**	**0.3167**	0.0238	0.0333	0.0357	0.05
	4	**0.3678**	**0.5333**	0.2759	0.4	0.0805	0.1167	0.0345	0.05
	5	**0.5287**	**0.7667**	0.3793	0.55	0.2414	0.35	0.4483	0.65
	6	0.359	0.2333	**0.4615**	**0.3**	0.0769	0.05	0.2308	0.15
	7	**0.3077**	**0.4**	**0.3077**	**0.4**	0.1923	0.25	0.2308	0.3
	8	0.9394	0.5167	**1**	**0.55**	0.3636	0.2	0.7273	0.4
	9	0.7619	0.8	0.7619	0.8	0.4286	0.45	**0.8095**	**0.85**
Amazon	10	**0.6923**	**0.9**	0.5897	0.7667	0.4231	0.55	0.5	0.65
	11	0.3929	0.55	0.3929	0.55	0.3214	0.45	**0.5**	**0.7**
	12	**0.7143**	**1**	0.6071	0.85	0.4524	0.63	0.5357	0.75
	13	**0.2361**	**0.2833**	0.2083	0.25	0.0417	0.05	0.2083	0.25
	14	**0.1111**	**0.1333**	0.0972	0.1167	**0.1111**	**0.1333**	0.0833	0.1

Table 1 shows the precision and recall for the top 20 predictions. We can see that in most cases, CICF standalone and CICF+LapRLS outperform the GCMF and traditional LapRLS. In particular, CICF, when paired up with LapRLS, improves the performance of LapRLS up to about 35% in both precision and recall given dataset 1. On the other hand, CICF standalone outperforms GCMF up to 60% in precision given dataset 2 and 58% in recall given dataset 8. Generally, the performance of CICF standalone is comparable to that of CICF+LapRLS. But CICF standalone offers an advantage in fewer hyperparameters to be tuned.

Note that in datasets 9 and 11 the LapRLS performs the best. Although the extrinsic heterogeneity is high in these two datasets (see Table 1), we find that there exist many small user groups that are identical across the networks. Therefore, the extrinsic link structures and contents are already helpful to make good predictions. In this case, CICF is more conservative on only picking the consistent information. Also note that the GCMF performs well in dataset 14 when the intrinsic heterogeneity is relatively low. In the presence of intrinsic heterogeneity, we also observe that GCMF gives unstable performance during the cross-validation.

Fig. 5 shows a typical ROC curve demonstrating how CICF and CICF+LapRLS outperform GCMF and traditional LapRLS. Both CICF and CICF+LapRLS can make correct predictions at low false positive rates, and therefore are suitable for situations where only few predictions can be made (due to, for example, limited budget for advertisement).

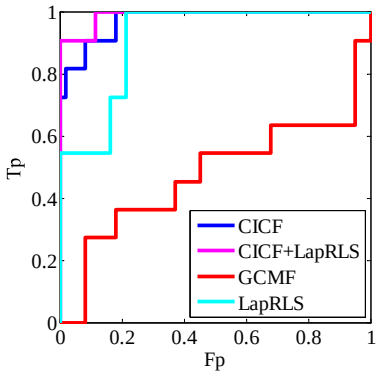

Figure 5. An ROC curve on dataset 8.

Acknowledgment

This work is supported in part by US NSF through grants CNS-1115234, DBI-0960443, and OISE-1129076, and US Department of Army through grant W911NF-12-1-0066.

References

Belkin, M., Niyogi, P., and Sindhwani, V. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *The Journal of Machine Learning Research*, 7:2399–2434, 2006.

Duan, Lixin, Xu, Dong, and Tsang, Ivor. Learning with augmented features for heterogeneous domain adaptation. In *Proc. of ICML*, pp. 711–718, 2012.

Glorot, Xavier, Bordes, Antoine, and Bengio, Yoshua. Domain adaptation for large-scale sentiment classification:

A deep learning approach. In *Proc. of ICML*, pp. 513–520, 2011.

He, Jingrui, Liu, Yan, and Lawrence, Richard. Graph-based transfer learning. In *Proc. of CIKM*, pp. 937–946, 2009.

Jiang, Meng, Cui, Peng, Wang, Fei, Yang, Qiang, Zhu, Wenwu, and Yang, Shiqiang. Social recommendation across multiple relational domains. In *Proc. of CIKM*, 2012.

Leskovec, Jure, Adamic, Lada A, and Huberman, Bernardo A. The dynamics of viral marketing. *ACM Trans. on the Web*, 1(1):5, 2007.

Ling, X., Dai, W., Xue, G.R., Yang, Q., and Yu, Y. Spectral domain-transfer learning. In *Proc. of KDD*, pp. 488–496, 2008.

Long, M., Wang, J., Ding, G., Cheng, W., Zhang, X., and Wang, W. Dual transfer learning. In *Proc. of SIAM*, 2012a.

Long, M., Wang, J., Ding, G., Shen, D., and Yang, Q. Transfer learning with graph co-regularization. In *Proc. of AAAI*, 2012b.

Lu, Zhengdong, Savas, Berkant, Tang, Wei, and Dhillon, Inderjit S. Supervised link prediction using multiple sources. In *Proc. of ICDM*, pp. 923–928, 2010.

Narayanan, Arvind and Shmatikov, Vitaly. De-anonymizing social networks. In *IEEE Symposium on Security and Privacy*, pp. 173–187, 2009.

Pan, S.J., Tsang, I.W., Kwok, J.T., and Yang, Q. Domain adaptation via transfer component analysis. *IEEE Trans. on Neural Networks*, 22(2):199–210, 2011.

Qi, G.J., Aggarwal, C.C., and Huang, T. Community detection with edge content in social media networks. In *Proc. of ICDE*, pp. 534–545, 2012.

Seung, D. and Lee, L. Algorithms for non-negative matrix factorization. *Advances in Neural Information Processing Systems*, 13:556–562, 2001.

Tang, J., Wu, S., Sun, J., and Su, H. Cross-domain collaboration recommendation. In *Proc. of KDD*, pp. 1285–1293, 2012.

Tang, Jie, Zhang, Jing, Yao, Limin, Li, Juanzi, Zhang, Li, and Su, Zhong. Arnetminer: Extraction and mining of academic social networks. In *Proc. of KDD*, pp. 990–998, 2008.

Tang, Wei, Lu, Zhengdong, and Dhillon, Inderjit S. Clustering with multiple graphs. In *Proc. of ICDM*, pp. 1016–1021, 2009.

Von Luxburg, Ulrike. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

Wang, C. and Mahadevan, S. Manifold alignment without correspondence. In *Proc. of IJCAI*, 2009.

Wang, C. and Mahadevan, S. Heterogeneous domain adaptation using manifold alignment. In *Proc. of IJCAI*, pp. 1541–1546, 2011.

Yang, T., Jin, R., Chi, Y., and Zhu, S. Combining link and content for community detection: a discriminative approach. In *Proc. of KDD*, pp. 927–936, 2009.

Yang, Z. and Oja, E. Unified development of multiplicative algorithms for linear and quadratic nonnegative matrix factorization. *IEEE Trans. on Neural Networks*, 22(12): 1878–1891, 2011.

Yang, Zhirong and Oja, Erkki. Linear and nonlinear projective nonnegative matrix factorization. *IEEE Trans. on Neural Networks*, 21(5):734–749, 2010.

Zhang, H., Yang, Z., and Oja, E. Adaptive multiplicative updates for projective nonnegative matrix factorization. In *Proc. of the Neural Information Processing*, pp. 277–284, 2012.

Zhong, E., Fan, W., Wang, J., Xiao, L., and Li, Y. Comsoc: adaptive transfer of user behaviors over composite social network. In *Proc. of KDD*, pp. 696–704, 2012.

Zhou, Y., Cheng, H., and Yu, J.X. Graph clustering based on structural/attribute similarities. *Proc. of VLDB*, 2(1): 718–729, 2009.

Zhuang, F., Luo, P., Xiong, H., He, Q., Xiong, Y., and Shi, Z. Exploiting associations between word clusters and document classes for cross-domain text categorization. *Statistical Analysis and Data Mining*, 4(1):100–114, 2011.