A New Kernel for Classification of Networked Entities

Dell Zhang DELL.Z@IEEE.ORG
SCSIS, Birkbeck, University of London, Malet Street, London WC1E 7HX, UK

Robert Mao ROBMAO@MICROSOFT.COM
Microsoft Corp., EPDC5/2352, South County Business Park, Leopardstown, Dublin, Ireland

Keywords: graph mining, community discovery, semi-supervised learning, kernel methods.

Abstract

Statistical machine learning techniques for data classification usually assume that all entities are i.i.d. (independent and identically distributed). However, real-world entities often interconnect with each other through explicit or implicit relationships to form a complex network. Although some graph-based classification methods have emerged in recent years, they are not really suitable for complex networks as they do not take the degree distribution of network into consideration. In this paper, we propose a new technique, Modularity Kernel, that can effectively exploit the latent community structure of networked entities for their classification. A number of experiments on hypertext datasets show that our proposed approach leads to excellent classification performance in comparison with the state-of-the-art methods.

1. Problem

We are in a connected age: real-world entities often interconnect with each other through explicit or implicit relationships to form a *complex network* (Newman, 2003), such as social networks, information networks, technological networks and biological networks.

Given a complex network (graph) G that consists of n entities (nodes) and m links (edges), we can describe the network structure using its $n \times n$ adjacency matrix $\mathbf{A}$ with elements $A_{ij} \geq 0$ representing the number or weight of edges between node $\mathbf{x}_i$ and node $\mathbf{x}_j$. Since real-world complex networks are usually sparse, most elements in $\mathbf{A}$ should be 0. In this paper we focus on undirected networks, so $\mathbf{A}$ is symmetric. The degree of a node $\mathbf{x}_i$, i.e., the number of edges connected to a node $\mathbf{x}_i$, is given by $k_i = \sum_j A_{ij}$.

Without Loss of generality, suppose that there are two classes of entities in the network: the entities in one class are labelled with $+1$ and the entities in the other class are labelled with -1. In the given set of networked entities $X = \{\mathbf{x}_i\}_{i=1}^{n}$, there are l entities $X_l := \{\mathbf{x}_i\}_{i=1}^{l}$ for which the labels $Y_l := \{y_i\}_{i=1}^{l}$ are provided, and u entities $X_u := \{\mathbf{x}_j\}_{j=l+1}^{l+u}$ for which the labels are absent. Obviously $X = X_l \bigcup X_u$ and $n = l + u$. Our goal is to learn a classification function f so that the class of any networked entity $\mathbf{x}$ can be accurately predicted by the sign of $f(\mathbf{x})$. This is actually *semi-supervised learning*, i.e., learning from both labelled and unlabelled data (Chapelle et al., 2005).

2. Approach

Let us consider the problem of classifying networked entities in the framework of *kernel methods* (Scholkopf & Smola, 2002). A prominent advantage of kernel methods is that they typically lead to a *convex optimization* problem so the global optimal solution can be computed efficiently.

For a Mercer kernel $K : X \times X \rightarrow \mathbb{R}$, there is an associated Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}_K$ of functions $X \rightarrow \mathbb{R}$ with the corresponding norm $\| \ \|_K$. Given a set of labelled examples X_l as well as a set of unlabelled examples X_u, typical kernel methods for semi-supervised learning estimate the classification function to be

$$f^* = \arg\min_{f \in \mathcal{H}_K} \frac{1}{l} \sum_{i=1}^{l} V(x_i, y_i, f) + C\|f\|_K^2,$$

where the first term V is a loss function defined on X_l, the second term $\|f\|_K^2$ is a regularizer defined

Appearing in the 6^{th} *International Workshop on Mining and Learning with Graphs*, Helsinki, Finland, 2008.

on $X_l \bigcup X_u$ and the parameter C controls its relative weight. The regularizer $\|f\|_K^2$ imposes smoothness conditions on f to avoid overfitting.

Different choices of the loss function V in the above optimization problem lead to different learning algorithms. For example, substituting the hinge loss $(1 - y_i f(\mathbf{x}_i))_+ = \max(0, 1 - y_i f(\mathbf{x}_i))$ for V we get Support Vector Machine (SVM).

In real-world complex networks, entities (nodes) tend to group into communities, with a high density of links (edges) within communities and a low density of links (edges) between them. The latent *community structure* of a complex network must contain valuable clues about the right classification of entities (nodes) in the network, because entities are likely to connect to other members of their own class. A "good" classification function f for networked entities should align well not only with the labelled data but also with the community structure — entities in the same community should have a high probability to be classified into the same class.

In the ideal situation, each class of entities in the complex network would group into a separate community. So if the value of f is restricted to be either $+1$ or -1, then $\mathbf{f} = (f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n))^T$ can be regarded as the binary indication vector for a division of the network into communities, and $\mathbf{f}^T\mathbf{f} = n$. Furthermore, let g_i denote the group to which vertex $\mathbf{x}_i$ belongs. The function $\delta(g_i, g_j) = 1$ if $g_i = g_j$ and $\delta(g_i, g_j) = 0$ otherwise. It is easy to see that $\delta(g_i, g_j) = \frac{1}{2}(1 + f(\mathbf{x}_i)f(\mathbf{x}_j))$.

Most existing graph-based classification methods such as (Herbster et al., 2005) are rooted in *graph partitioning* (Chung, 1997) that divides the network into groups by minimizing the *cut-size*, i.e., the number of edges running between different groups of nodes: $S = \frac{1}{2}\sum_{i,j} A_{ij}(1 - \delta(g_i, g_j))$. We can rewrite the cut-size for the division of network $\mathbf{f}$ in matrix form as $S = \frac{1}{4}\mathbf{f}^T\mathbf{L}\mathbf{f}$, where $\mathbf{L}$ is the *Laplacian matrix* (Chung, 1997) defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$ with $\mathbf{D} = \mathrm{diag}(k_1, \ldots, k_n)$.

Despite its success on simple graphs (such as k-nearest-neighbours graphs), Laplacian based graph partitioning is poor in detecting natural communities in real-world complex networks, because the degree distribution of network has been totally ignored. The fundamental problem of using this technique for community detection is that cut sizes are not really the right thing to optimize since they don't accurately reflect our intuitive concept of network communities. A good division of a network into communities is not merely one in which the number of edges running across communities is small. Rather, it is one in which the number of edges across communities is *smaller than expected*. It has been reported that Laplacian based graph partitioning often fails to find the right division of a complex network (Newman, 2006). Consequently the effectiveness of semi-supervised learning methods based on graph partitioning for classifying networked entities would be limited.

One proven-effective approach to community detection is maximizing the quality function known as *modularity* (Newman, 2006) over the possible divisions of a network: $Q = \frac{1}{2m}\sum_{ij}[A_{ij} - P_{ij}]\delta(g_i, g_j)$, where $P_{ij} = (k_i k_j)/(2m)$, $\delta(g_i, g_j) = 1$ if $g_i = g_j$ and $\delta(g_i, g_j) = 0$ otherwise. In fact, P_{ij} is the *expected* number of edges between node $\mathbf{x}_i$ and node $\mathbf{x}_j$ in the 'null model' — a *random* graph with the same degree distribution as the given network. Optimizing modularity reflects our intuition that the number of edges within communities should be higher than expected by chance. Only if the number of within-community edges $\frac{1}{2}\sum_{ij} A_{ij}\delta(g_i, g_j)$ is significantly higher than it would be expected purely by chance $\frac{1}{2}\sum_{ij} P_{ij}\delta(g_i, g_j)$ can we justifiably claim to have found significant community structure. Maximizing modularity has been shown to produce excellent community detection results in practice. We can rewrite the modularitye for the division of network $\mathbf{f}$ in matrix form as $Q = \frac{1}{4m}\mathbf{f}^T\mathbf{M}\mathbf{f}$, where $\mathbf{M}$ is the *modularity matrix* defined as $\mathbf{M} = \mathbf{A} - \mathbf{P}$.

If we allow the the elements of $\mathbf{f}$ to take any real value but just keep the constraint $\mathbf{f}^T\mathbf{f} = n$, the optimal division of network that maximizes the modularity Q is given by $\mathbf{f} = \mathbf{u}_1$, the eigenvector of $\mathbf{M}$ corresponding to the largest eigenvalue (Newman, 2006). The sign of $\mathbf{u}_1$'s i-th elements indicates the class to which $\mathbf{x}_i$ belongs to and the value of of $\mathbf{u}_1$'s i-th elements indicates the strength of membership.

Motivated by modularity based community detection, we propose to use the following matrix as the kernel matrix: $\widehat{\mathbf{M}} = \sum_{k=1}^{p} \lambda_k \mathbf{u}_k \mathbf{u}_k^T$, where $\lambda_1 \geq \ldots \geq \lambda_p > 0$ are the p positive eigenvalues of the modularity matrix $\mathbf{M}$ and $\mathbf{u}_1, \ldots, \mathbf{u}_p$ are the p corresponding eigenvectors. Since $\mathbf{M}$ is not guaranteed to be positive definite, it could not be used straightforwardly as the kernel matrix otherwise the generated kernel function would be invalid and the resulted optimization problem would no longer be convex. According to linear algebra, $\widehat{\mathbf{M}}$ is the positive definite matrix that best approximates the modularity matrix $\mathbf{M}$ therefore we use it instead and name this technique Modularity Kernel (ModKer).

The Modularity Kernel could be justified as follows. Let $\mathcal{H}(G)$ be the linear space of real-valued functions defined on the graph, i.e., an n-dimensional vec-

tor space whose elements are the real-valued vector $\mathbf{g} = (g_1, \ldots, g_n)^T$. On $\mathcal{H}(G)$ we introduce the inner-product $\langle \mathbf{f}, \mathbf{g} \rangle = \mathbf{f}^T \widehat{\mathbf{M}}^{-1} \mathbf{g}$. This inner-product function is well-defined since $\widehat{\mathbf{M}}^{-1} = \sum_{k=1}^{p} \frac{1}{\lambda_k} \mathbf{u}_k \mathbf{u}_k^T$ is symmetric and positive definite. Moreover, the function $\|g\| = \sqrt{\langle \mathbf{g}, \mathbf{g} \rangle}$, $\mathbf{g} \in \mathcal{H}(G)$ is a norm that measures the function smoothness or complexity. With a little derivation, it is easy to see that minimizing the above defined norm of function f leads to $\mathbf{f} = \mathbf{u}_1$ which gives the optimal division of network into communities according to the analysis in the above section. Therefore it is reasonable to use $\|f\|^2 = \langle \mathbf{f}, \mathbf{f} \rangle = \mathbf{f}^T \widehat{\mathbf{M}}^{-1} \mathbf{f}$ as the regularizer in the kernel methods learning framework. The reproducing kernel $\mathbf{K}$ of $\mathcal{H}(G)$ should be an $n \times n$ matrix such that for every $\mathbf{g} \in \mathcal{H}(G)$ the reproducing kernel property $g_i = \langle \mathbf{K}_i, \mathbf{g} \rangle$ holds, where $\mathbf{K}_i$ is the i-th column of $\mathbf{K}$. In fact, $\mathbf{K} = \widehat{\mathbf{M}}$, because if $\mathbf{g} \in \mathcal{H}(G)$ then $\widehat{\mathbf{M}}^{-1} \widehat{\mathbf{M}} \mathbf{g} = \mathbf{g}$, which implies that $g_i = \mathbf{e}_i \widehat{\mathbf{M}} \widehat{\mathbf{M}}^{-1} \mathbf{g} = \mathbf{K}_i \widehat{\mathbf{M}}^{-1} \mathbf{g} = \langle \mathbf{K}_i, \mathbf{g} \rangle$ where $\mathbf{e}_i$ is the i-th coordinate vector.

It should be beneficial to exploit both entity content information and entity link information for classifying networked entities. We can construct a hybrid network which is the linear combination of two networks (with equal weights in our experiments) — one the original network induced by the links among entities, the other derived from entity content information — and then use Modularity Kernel computed on the hybrid network. When deriving the content-based network, we simply connect each pair of entities with an edge that is weighted by their content similarity. To distinguish the usage of Modularity Kernel on the pure link-based network and that on the combined hybrid network, we denote the former ModKer (link) and the latter ModKer (content+link).

3. Experiments

We conduct our experiments on two collections of real-world datasets: WebKB[1] and Cora[2].

The WebKB datasets consist of about 6,000 web pages from the computer science departments of four universities: *Cornell, Texas, Washington and Wisconsin*. The web pages are classified into categories such as 'course', 'faculty' and 'student'. For each dataset, we use the *co-citation graph* derived from the original directed hyperlinks (citations) as the link-based entity network for our algorithms, i.e., two pages (nodes) are connected by an edge if there is a third page linking to or being linked to both of them, and multiple edges are allowed between one pair of nodes. This is because for Web data co-citations are usually more reliable than hyperlinks which has been reported in past research studies (Zhang et al., 2006) and also confirmed by our experiments.

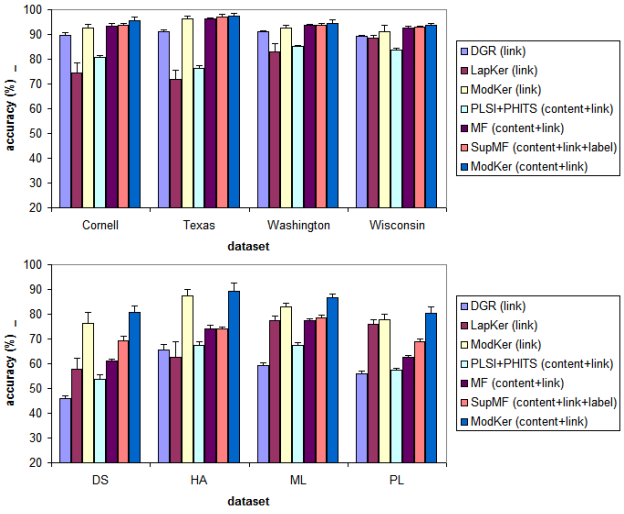

Figure 1. The comparison of classification accuracy on the WebKB datasets (upper) and the Cora datasets (lower).

The Cora datasets consist of the abstracts and references of about 34,000 computer science papers. In our experiments, we use only the papers in four research areas, *Data Structure (DS), Hardware and Architecture (HA), Machine Learning (ML) and Programming Language (PL)*, and we discard the papers without reference to other papers in the same area. The papers are classified according to their subfields in the research area. For each dataset, we use the *undirected graph* derived from the original directed references as the link-based entity network for our algorithms, i.e., two papers (nodes) are connected by an edge if one of them cites the other or vice versa, and multiple edges are allowed between one pair of nodes.

The employed learning algorithm for our Modularity Kernel technique is SVM as implemented in LIBSVM[3] (with default parameter values). The classification performance is measured by the 5-fold cross-validation accuracy.

We compare our proposed approach Modularity Kernel (**ModKer**) with some state-of-the-arts methods for hypertext classification, among which Directed Graph Regularization (**DGR**) (Zhou et al., 2005) and Laplacian Kernel (**LapKer**) (Herbster et al., 2005) use only entity link information while **PLSI+PHITS**

[1] http://www.cs.cmu.edu/∼webkb/

[2] http://www.cs.umass.edu/∼mccallum/code-data.html

[3] http://www.csie.ntu.edu.tw/∼cjlin/libsvm/

(Cohn & Hofmann, 2000), Matrix Factorization (**MF**) and Supervised Matrix Factorization (**SupMF**) (Zhu et al., 2007) use both entity content and entity link information. On all datasets, using only entity link information ModKer works better than LapKer and DGR, while using both entity content and entity link information ModKer works better than PLSI+PHITS, MF and SupMF, as shown in Figure 1.

Acknowledgements

We would like to thank Dr. Shenghuo Zhu for sharing his pre-processed datasets. Thanks also to the anonymous reviewers for their helpful comments.

References

Chapelle, O., Scholkopf, B., & Zien, A. (Eds.). (2005). *Semi-supervised learning*. MIT Press.

Chung, F. R. K. (1997). *Spectral graph theory*. American Mathematical Society.

Cohn, D. A., & Hofmann, T. (2000). The missing link - a probabilistic model of document content and hypertext connectivity. *Advances in Neural Information Processing Systems (NIPS)* (pp. 430–436). Denver, CO, USA.

Herbster, M., Pontil, M., & Wainer, L. (2005). Online learning over graphs. *Proceedings of the 22nd International Conference on Machine Learning (ICML)* (pp. 305–312). Bonn, Germany.

Newman, M. E. J. (2003). The structure and function of complex networks. *SIAM Review*, *45*, 167–256.

Newman, M. E. J. (2006). Finding community structure in networks using the eigenvectors of matrices. *Physical Review E*, *74*, 036104.

Scholkopf, B., & Smola, A. J. (2002). *Learning with kernels*. Cambridge, MA: MIT Press.

Zhang, T., Popescul, A., & Dom, B. (2006). Linear prediction models with graph regularization for web-page categorization. *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)* (pp. 821–826). Philadelphia, PA.

Zhou, D., Huang, J., & Scholkopf, B. (2005). Learning from labeled and unlabeled data on a directed graph. *Proceedings of the 22nd International Conference on Machine Learning (ICML)* (pp. 1036–1043). Bonn, Germany.

Zhu, S., Yu, K., Chi, Y., & Gong, Y. (2007). Combining content and link for classification using matrix factorization. *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)* (pp. 487–494). Amsterdam, The Netherlands.