

Seminar on Semi-supervised Learning using Graph Kernels

Sandeepkumar B. Satpal
Guide: Prof. Sunita Sarawagi
KReSIT, IIT Bombay

March 29, 2006



Motivation

- Traditional classifier need labeled data which is time consuming and expensive to obtain
- How to use unlabeled data?
- Semi-supervised learning addresses these problem
- Classical kernel and Graph kernels



Introduction

- Semi-supervised learning uses both labeled and unlabeled instances to build a classifier
- Kernel functions returns the similarity between two instances
- Graph kernel use structured information to find similarity
- Approximate the difference
- Object decomposed into sub-structures, find similarity and finally combine it.



Outline

- Variants of graph kernels
- Types of graph kernels
 - Count
 - Random walk
 - Subtree pattern
 - Shortest path
- Application on web graph
- Conclusion
- Further references

Variants of Graph Kernels

- First variant
 - Kernel on two graphs
 - Each graph is either labeled or unlabeled
 - Each graph represent an instance
- Second variant
 - Kernel on two nodes of same graph
 - Each node is either labeled or unlabeled
 - Each node represent an instance

Types of Graph kernels

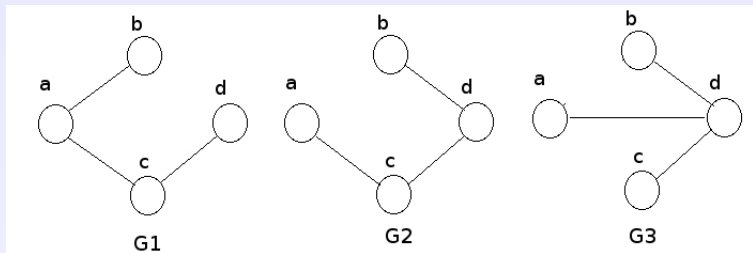
- Graph kernel based on ..
 - Count
 - Random walk
 - Subtree pattern
 - Shortest path

Graph kernel based on count

- Each element in document vector is a count of a particular label
 - $X_G = \left(\frac{\#label(l_1, G)}{|V|}, \frac{\#label(l_2, G)}{|V|}, \dots, \frac{\#label(l_m, G)}{|V|} \right)$
- $K(G_i, G_j) = \frac{1}{|V_i||V_j|} \sum_{v_1 \in V_i} \sum_{v_2 \in V_j} k(v_1, v_2)$
- $k(v_1, v_2) = I(v_1, v_2)$
- Advantages
 - Simple to calculate
 - Complexity is (nm) , where n and m are the number of nodes in G_1 and G_2 respectively.



Count (contd...)



• Problem

- Same similarity value for $K(G_1, G_2)$ and $K(G_1, G_3)$
- Does not consider local connectivity

Graph kernel based on Random walk

- 1st method

- Redefine $k(v_1, v_2)$

- $k(v_1, v_2) = pk_0(v_1, v_2) + p(1 - p) \sum_{\substack{e_1 \in A(v_1) \\ e_2 \in A(v_2)}} k_1(v_1, v_2, e_1, e_2) + \dots$

- $k_0(v_1, v_2) = I(v_1, v_2)$

- $k_1(v_1, v_2, e_1, e_2) = k_0(v_1, v_2) * \frac{I(e_1, e_2) * I(\delta(v_1, e_1), \delta(v_2, e_2))}{|A(v_1)| |A(v_2)|}$

- $K(G_1, G_2) > K(G_1, G_3)$

- Problem

- For each v_1 and v_2 , compute $k(v_1, v_2)$
- Each $k(v_1, v_2)$ needs lots of computation

Random walk (contd...)

- 2nd method
 - Assumes if two graphs are similar then random walk matches with high probability
 - $h = (h_1, h_2, \dots, h_l)$
 - $p_s(h_1)$: Initial probability (uniform)
 - $p_t(h_i|h_{i-1})$: Transition probability
 - $p_q(h_{i-1})$: Probability that it ends at $i - 1$
 - Probability of random walk for a given graph
 - $p(h|G) = p_s(h_1) * \prod_{i=2}^l p_t(h_i|h_{i-1})p_q(h_l)$

Random walk (contd...)

- Kernel function

- $K(G_1, G_2) = \sum_{i=1}^{\infty} \sum_h \sum_{h'} p(h|G)p(h'|G')K_r(z, z')$

-

$$K_r(z, z') = \begin{cases} 0 & (l \neq l') \\ K_v(l(h_1), l(h'_1)) & \\ \prod_{i=2}^l K_e(l(h_{i-1}, h_i), l(h'_{i-1}, h'_i)) * K_v(l(h_l), l(h'_l)) & (l = l') \end{cases}$$

- where, $z = (G, h)$ and $l(h_i)$ is label associated with node h_i

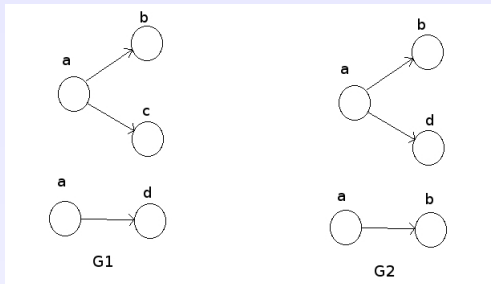
- Advantage over count

- It considers local connectivity of graphs

- Advantage over 1st method

- Consumes less time

Random walk (contd...)



• Problems

- Different graphs may map to same space
 - It generally happens when same label appears large number of times
- Tottering
 - Similar smaller structure generates high similarity score

Graph kernel based on Subtree pattern

- Random walk does not consider tree structure
- Each element in a feature vector is number of times subtree pattern appears
- Kernel function

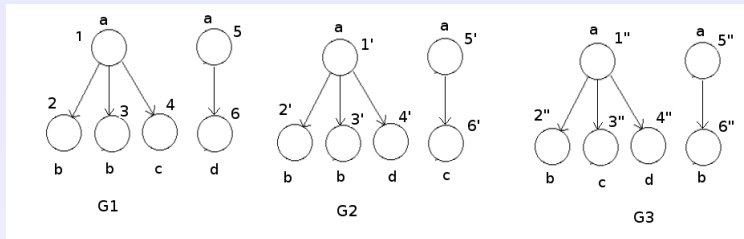
- $K(G_1, G_2) = \sum_{r \in V_1} \sum_{s \in V_2} K_{r,s,h}$

- $K_{r,s,h} = 1$, if $h = 1$ and $\text{label}(r) = \text{label}(s)$

- $K_{r,s,h} = \lambda_r \lambda_s \sum_{R \in M_{rs}} * \prod_{(r',s') \in R} K_{r',s',h-1}$

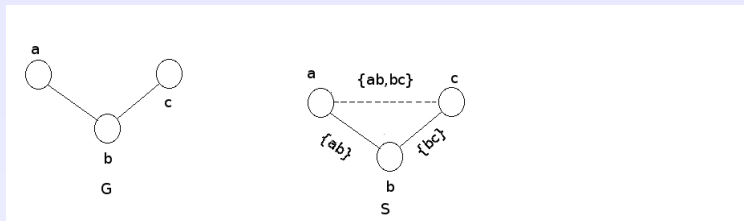
- $M_{r,s} = \{R \subseteq \delta^+(r) * \delta^+(s) \mid (\forall (a,b)(c,d) \in R : a = c \Leftrightarrow b = d) \wedge (\forall (a,b) \in R : \text{label}(a) = \text{label}(b))\}$

Subtree pattern (contd...)



- $M(1, 1') = \{\{(2, 2'), (3, 3'), (2, 3'), (3, 2')\}\}$
- $M(1, 1'') = \{\{(2, 2'')\}, \{(4, 3'')\}\}$
- Problems
 - Tottering
 - Number of nodes to consider grows exponentially with height of subtree

Graph kernel based on Shortest path



- All previous kernel were either computationally expensive or limited in their expressiveness
- Transform graph G to shortest-path graph S .
- Kernel function
 - $K(S_1, S_2) = \sum_{e_1 \in E_1} \sum_{e_2 \in E_2} k_w^1(e_1, e_2)$
- where k_w^1 is a positive definite kernel on edge walk of length 1.



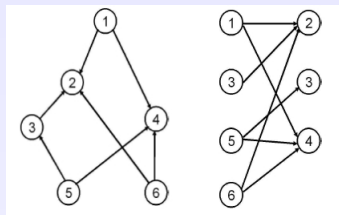
Application on Web-graph

- Web-Graph
 - Documents are represented by nodes
 - Hyperlinks are represented by edges
- Properties of web-graph
 - If there are many common words in two documents or
 - If two documents are cited by same documents
 - Then there is high probability that they belongs to same category.

Application (contd...)

- Each document is either labeled or unlabeled
- Two different kernels
 - Kernels on text
 - Each document is a vector in feature space
 - Each element in a vector is frequency of occurrence of that word
 - Term by document matrix
 - $K_w(d_1, d_2) = \langle d_1, d_2 \rangle$
 - Co-citation kernels

Co-citation kernels







- Hub - Authority model
- Each document in authority set is a vector in feature space
- Each element i is either 1 or 0 when i^{th} document from hub links to this document or not.
- $K_a(d_1, d_2) = \langle d_1, d_2 \rangle$



Conclusion

- For sparse graph, all works well, but counting methods is very fast
- If number of times same labels appear are
 - small: then random walk performs well
 - large: then subtree pattern performs well
- Both methods suffer from tottering
- Subtree pattern grows exponentially with height of subtree
- Shortest path : efficient in computation

For Further Reading I

-  Cristianini and Shawe-Taylor.
An Introduction to Support Vector Machines.
Cambridge University Press, 2000.
-  Hisashi Kashima and Akihiro Inokuchi
Kernels for Graph Classification, 2002
-  Hisashi Kashima and Akihiro Inokuchi
Marginalized Kernels Between Labeled Graphs, ICML
2003: 321-328,
-  Dengyong Zhou, Bernhard Schlkopf, Thomas Hofmann
Semisupervised Learning on Directed Graphs, NIPS 2004.

For Further Reading II



Thorsten Joachims, Nello Cristianini, John Shawe-Taylor
Composite Kernels for Hypertext Categorisation, ICML
2001: 250-257