String Kernels for Matching Seriated Graphs

Hang Yu
Computer Science Department
University of York

Edwin R. Hancock
Computer Science Department
University of York

Abstract

Graph seriation allows the nodes of a graph to be placed in a string order, and then matched using string alignment algorithms. Prior work has used Bayesian methods to derive the string edit costs required in matching. The aim in this paper is to demonstrate how the matching of seriated graphs can be kernelised. To do this we make use of string kernels and show how the parameters of the kernels can be linked to edge density. We illustrate that the graph edit distances computed using the string kernel can be used for graph clustering.

1 Introduction

Machine learning with graphs has proved to be an elusive problem. There are a number of reasons for this. First, graphs are not vectorial in nature, and most of the available methodology in the machine learning literature focuses on data that exist in a vectorial form. Second, the modes of variation that exist in a sample of graphs are structural in nature, being due to the changes in edge or node structure.

Recently, however, there have been a number of advances which bring the analysis of structural variations in graphs within grasp. First, it has been shown how to construct kernels over non-vectorial structures including graphs, trees and strings. In fact, the simplest of these structures is the string due to the serial ordering of the nodes. Second, techniques have been developed which allow the nodes of a graph to be placed in a string order. This process is known as graph seriation and involves finding a permutation of the nodes of a graph that satisfies ordering constraints provided by the edges of the graph. The recovery of the permutation order can be posed as an optimisation problem. It has been shown that when the cost-function is harmonic, then an approximate solution is given by the Fiedler vector of the Laplacian matrix for the graph under study [7]. In a recent paper, Robles-Kelly and Hancock [2] have reformulated the problem as that of recovering the node permutation order subject to edge connectivity constraints, and have provided an approximate spectral solution to the problem. Although spectral methods are elegant and convenient, they are only guaranteed to locate solutions that are locally optimal. To overcome this shortcoming, Yu and Hancock [5] have shown how semidefinite programming (SDP) [9] can be used to convexify the seriation problem, and have demonstrated advantages over the earlier method of Robles-Kelly and Hancock.

One of the shortcomings of the method for computing edit distances developed by Robles-Kelly and Hancock [1] is that it relies of Dijsktra's algorithm to match via string alignment. To render the method more efficient, in this paper we turn to kernel methods. In particular, we aim to investigate whether string kernels can be used to measure the similarity of seriated graphs. Kernel methods operate using inner products between pattern vectors [11]. String kernels have been successfully used for text classification problem by Lodhi and his coworkers [6]. We show how to represent seriated graphs in a symbolic manner so that the computation of the distance between graphs can be effected by applying a string kernel mapping to the associated symbol strings. By comparing the expression for distance with that obtained for edit distance by Robles-Kelly and Hancock, we relate the kernel decay parameter to the edge density of the graphs being matched. We show how the resulting kernel distances can be used for the purposes of graph clustering.

2 Graph Seriation

We are concerned with the undirected graph $G = (V, E)$ with node index-set V and edge-set $E =\subseteq V \times V$. The adjacency matrix A for the graph is the $V \times V$ matrix with elements

$$A(i,j) = \begin{cases} 1 & \text{if}(i,j) \in E \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

The graph seriation problem has been posed as one of optimisation in the work of Atkins *et al* [7]. Formally, the problem can be stated as finding a path sequence for the nodes in the graph using a permutation π which will minimize the penalty function

The 18th International Conference on Pattern Recognition (ICPR'06)
0-7695-2521-0/06 $20.00 © 2006 IEEE

$$g(\pi) = \sum_{i=1}^{|V|} \sum_{j=1}^{|V|} A(i,j)(\pi(i) - \pi(j))^2 \qquad (2)$$

Since the task of minimizing g is NP-hard due to the discrete nature of the permutation, a relaxed solution is sought using a function h of continuous variables x_i. The relaxed problem can be posed as seeking the solution of the constrained optimisation problem $x = \arg\min_{x^*} h(x^*)$ where $h(x) = \sum_{(i,j)} f(i,j)(x_i - x_j)^2$ subject to the constraints $\sum_i x_i = 0$ and $\sum_i x_i^2 = 1$. Using graph-spectral methods, Atkins and his coworkers showed that the solution to the above problem can be obtained from the Laplacian matrix of the graph. The Laplacian matrix is defined to be $L_A = D_A - A$ where D_A is a diagonal matrix with $d_{i,i} = \sum_{j=1}^{n} A_{i,j}$. The solution to the relaxed seriation problem equation (2) is given by the Fiedler vector, i.e. the vector associated with the smallest non-zero eigenvalue of L_A. The required serial ordering is found by sorting the elements of the Fiedler vector into rank-order. Recently, Robles-Kelly and Hancock [2] have extended the graph seriation problem by adding edge connectivity constraints. The graph seriation problem is restated as that of minimising the cost-function

$$h_E(x) = \sum_{i=1}^{|V|-1} \sum_{k=1}^{|V|} (A(i,k) + A(i+1,k))x_k^2 \qquad (3)$$

By introducing the matrix

$$\Omega = \begin{bmatrix} 1 & 0 & 0 & 0 \cdots 0 & 0 \\ 0 & 2 & 0 & 0 \cdots 0 & 0 \\ \vdots & & & & \\ 0 & 0 & 0 & 0 \cdots 2 & 0 \\ 0 & 0 & 0 & 0 \cdots 0 & 1 \end{bmatrix}$$

the path connectivity requirement is made more explicit. The minimiser of $h_E(\vec{x})$ satisfies the condition

$$\lambda = arg\min_{x_*} \frac{\vec{x_*}^T \Omega A \vec{x_*}}{\vec{x_*}^T \Omega \vec{x_*}} \qquad (4)$$

Although elegant and convenient, spectral methods are only guaranteed to find a locally optimal solution to the problem. For this reason in this paper we turn to the more general method of semidefinite programming to locate an optimal solution which utilizes the convexity properties of the matrix representation.

3 Semidefinite Programming

Semidefinite programming (SDP) is convex optimisation technique that is efficient since it uses interior-point methods. The method has been applied to a variety of optimisation tasks in combinatorial optimization. Semidefinite programming is essentially an extension of ordinary linear programming, where the vector variables are replaced by matrix variables and the nonnegativity elementwise constraints are replaced by positive semidefiniteness. The standard form for the problem is: $X = \arg\min_{X^*} traceCX^*$, such that $traceF_iX = b_i$, $i = 1...m$, $X \succeq 0$. Here C, F_i and X are real symmetric $n \times n$ matrices and b_i is a scalar. The constraint $X \succeq 0$ means that the variable matrix must lie on the closed convex cone of positive semidefinite solutions. To solve the graph seriation problem using semidefinite programming, we denote the quantity $\Omega^{1/2} A \Omega^{-1/2}$ appearing in equation (4) by B and $\Omega^{1/2} x_*$ by y. With this notation the optimisation problem can be restated as $\lambda = \arg\min_{y^T y=1} y^T B y$. Noting that $y^T B y = trace(Byy^T)$ by letting $Y = yy^T$ in the semidefinite programming setting the seriation problem becomes $Y = \arg\min_{Y^*} traceBY^*$ such that $traceEY^* = 1$, where the matrix E is the unit matrix, with the diagonal elements set to 1 and all the off-diagonal set to 0. Note that $Y = yy^T$ is positive semidefinite and has rank one. As a result it is convex and we can add the positive semidefinite condition $Y \in S_n^+$ where S_n^+ denotes the set of symmetric $n \times n$ matrices which are positive semidefinite.

Interior Point Algorithm : To compute the optimal solution Y^*, a variety of iterative interior point methods can be used. By using the SDP solver developed by Fujisawa et.al [8], a primal solution matrix Y^* can be obtained. Using the solution Y^* to the convex optimization problem, we must find an ordered solution y to the original problem. To do this we use the randomized-hyperplane technique proposed by Goemans and Williamson [10].

Since $Y^* \in S_n^+$, by using the Cholesky decomposition we have that $Y = V^T V, V = (v_1, ...v_n)$.Recalling the constraint $y^T y = 1$, the vector y must lie on the unit sphere in a high dimensional space. This means that we can use the randomized hyperplanes approximation. This involves choosing a random vector r from the unit sphere. An ordered solution can then be calculated from $Y^* = V^T V$ by ordering the value of $v_i^T r$. We repeat this procedure multiple times for different random vectors. The final solution y_* is the one that yields the minimum value for the objective function $y^T B y$. This technique can be interpreted as selecting different hyperplanes through the origin, identified by their normal r, which partition the vectors $v_i, i = 1....n$.

The solution vector x_* can be obtained using the equation $\Omega^{1/2} x_* = y$, and the elements of the vector x_* then can be used to construct the serial ordering of the nodes in the graph. Commencing from the node associated with the largest component of x_*, we sort the nodes in so that the nodes are ordered so that the components of x_* are of de-

creasing magnitude and also satisfy edge connectivity constraints on the graph. We iteratively proceed in the following. Let us denote the list of the visited nodes by S_k at the kth iteration. Initially $S_1 = i_1 = \arg\max_i x_*(i)$. We proceed by searching the set of the first neighbours of i_1, i.e. $N_{i_1} = \{j|(i_1, j) \in E\}$, to locate the node which is associated with the largest remaining component of x_*. This node is then appended to the list of nodes visited list and satisfies the condition $i_2 = \arg\max_{l \in N_{i_1}} x_*(l)$. This process is repeated until every node in the graph is visited. At termination the sorted list of nodes is the string S_G.

4 String Kernels

With the converted string at hand, we are then able to solve the graph clustering problems using the string kernel techniques. The idea is to compare the strings by means of the substrings they contain: the strings have greater similarity if they share more common substrings. The substrings do not need to be contiguous which however the contiguity measure its weight during the comparison. For example, the substring '**e-f-c** ' is present both in string sequence 'a-**e-f-c** -d' and 'b-**e**-d-a-**f-c**' but with different weighting. Here we give an example in Figure 1 to illustrate the idea more clearly. Let's first introduce a decay factor $\lambda \in (0, 1)$ which can be used to weight a certain feature in a string.

Example Consider the graphs G_a,G_b,G_c. If we consider the case which two characters features are present, the strings are mapped as follows:

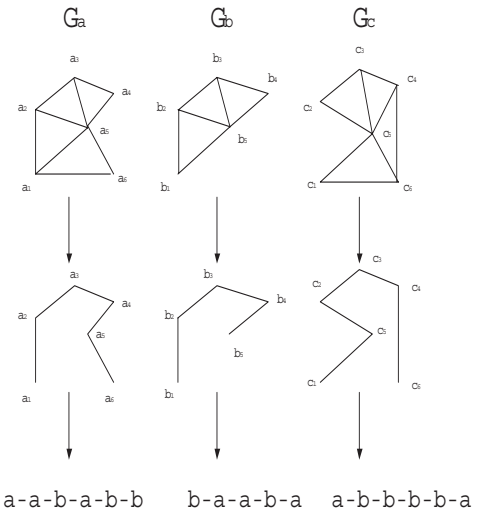

Figure 1. Example

	$a-a$	$a-b$	$b-a$	$b-b$
$\phi(aababb)$	λ^2+ $\lambda^3+\lambda^4$	$2\lambda^2+\lambda^3$ $+\lambda^4+2\lambda^5$	λ^2	λ^2+ $\lambda^3+\lambda^4$
$\phi(baaba)$	λ^2+ $\lambda^3+\lambda^4$	$\lambda^2+\lambda^3$	$2\lambda^2+$ $\lambda^3+\lambda^5$	λ^4
$\phi(abbbba)$	λ^6	$\lambda^2+\lambda^3$ $+\lambda^4+\lambda^5$	$\lambda^2+\lambda^3$ $+\lambda^4+\lambda^5$	$3\lambda^2+$ $2\lambda^3+\lambda^4$

Hence, the kernel between the string G_a and G_b is $K(G_a, G_b) = K(G_b, G_a) = (\lambda^2+\lambda^3+\lambda^4)^2+(2\lambda^2+\lambda^3+\lambda^4+2\lambda^5)(\lambda^2+\lambda^3)+\lambda^2(2\lambda^2+\lambda^3+\lambda^5)+\lambda^4(\lambda^2+\lambda^3+\lambda^4)$.

4.1 Node Attributes

According to the method of Robles-Kelly and Hancock [1], the serial ordering of the nodes can be recovered from the components of the leading eigenvector ϕ^* of the marix ΩA. We attribute the nodes of the seriated graphs using the components of this eigen-vector since it relates to their serial order in the string. In the graph seriation step, we denote the serial ordering of the nodes of the graph by S_T , then the node indexed i_T. We define a finite alphabet: $\{\Sigma = a_1, a_2, a_3, ...a_n\}$. The symbols are assigned to the nodes of the graph using the leading eigenvector components using the rule

$$SS_k(i) = \begin{cases} a_1 & L_{min} \leq \phi^*(i) \leq \\ & L_{min} + 1/n * (L_{max} - L_{min}) \\ a_2 & L_{min} + 1/n * (L_{max} - L_{min}) \\ & \leq \phi^*(i) \leq L_{min} + 2/n * (L_{max} - L_{min}) \\ \vdots & \\ a_n & L_{min} + (n-1)/n * (L_{max} - L_{min}) \\ & \leq \phi^*(i) \leq L_{max} \end{cases} \quad (5)$$

where the $\phi^*(i)$ is element of the leading eigenvector corresponding to the node indexed i, and L_{min} and L_{max} are the minimum and maximum components of the leading eigenvector. In this way, we are able to map the seriated graph into symbol sequence $\{a_1, a_2, ...a_n\}$. Here we will be interested in a sample of graphs $G_1, G_2, ...G_k, ...$ We denote the mapped symbol string sequence for the graph indexed k by SS_k.

4.2 Computation of String Kernel

In this section we demonstrate how to compute the distances between two symbol sequences using the string kernel.

Definition: Let Σ be a finite alphabet. A string is a finite sequence of characters from σ, including empty sequence. For strings s and t we denote with $|s|$ the length of string $s = s_1...s_{|s|}$, with st the string obtained by concatenating

the strings s and t and with $s[i:j]$ substring $s_i...s_j$. We say that u is a substring of s if there exists indices I=$(i_1, ..., i_{|u|})$ with $1 \leq i_1 < ... < i_|u| \leq |s|$such that $u = s[I]$. The length $l(I)$of the subsequence in s is $i_|u| - i_i + 1$. The kernel mapping Φ for the string s is obtained by defining the kernel feature mapping Φ_u for each $u \in \Sigma^n$ as

$$\Phi_u = \sum_{I:u=s[I]} \lambda^{l(I)} \tag{6}$$

where $\lambda \leq 1$. These kernel features measure the number of occurrences of the subsequences in the string s, by weighting them according to their lengths. Hence, the inner product of the feature vectors for two string s and t gives a sum over all common sub-sequences weighted according to their frequency of occurrence and lengths. The Euclidean distance between two strings in the feature space is given by the following equation:

$$\begin{aligned} d(s,t) &= \sqrt{\textstyle\sum_{u\in\Sigma^n}(\Phi_u(s)-\Phi_u(t))^2} \\ &= \sqrt{\sum_{u\in\Sigma^n}\Phi_u(s)^2 + \sum_{u\in\Sigma^n}\Phi_u(t)^2 - 2\sum_{u\in\Sigma^n}\Phi_u(s)\Phi_u(t)} \\ &= \sqrt{\textstyle\sum_{u\in\Sigma^n}\Phi_u(s)^2 + \sum_{u\in\Sigma^n}\Phi_u(t)^2 - 2K_n(s,t)} \end{aligned}$$

where $K_n(s,t)$ is the string kernel function

$$\begin{aligned} K_n(s,t) &= \sum_{u\in\Sigma^n}\Phi_u(s)\Phi_u(t) \\ &= \sum_{u\in\Sigma^n}\sum_{I:u=s[I]}\sum_{J:u=s[J]}\lambda^{l(I)+l(J)} \end{aligned}$$

Once the distance matrix is computed, we proceed to apply multidimensional scaling to the matrix to generate a graph clustering result. In this case, $\sum_{u\in\Sigma^n}\Phi_u(s)^2$ and $\sum_{u\in\Sigma^n}\Phi_u(t)^2$ can thought to be constant in the feature space, so, the more similar substrings they have, the large K they will have and the smaller distance will be between them.

Finally, it is interesting to note that the we can link the kernel distance to the Bayes edit distance computed by Robles-Kelly and Hancock, by comparing the expression above with the expressions for edit distance. This analysis reveals that

$$\lambda = \frac{\rho_{mean}}{\rho_{max} - \rho_{min}} \times (\lambda_{max} - \lambda_{min})$$

Here $\rho = \frac{|V|^2}{E}$is the edge density for the graph with node-set E and edge-set E. λ_{max} and λ_{min} corresponds to the maximum and minimum decay factors we can take. When two graphs are being matched ρ_{max} is the maximum edge density, ρ_{min} is the minimum edge-density and $\rho_{mean} = 1/2(\rho_{max} + \rho_{min})$ is the mean edge density.

5 Experiments

For our experimental evaluation we use the COIL image database. We have selected six objects from the database. For each object there are 20 different views. We extract corner features from the object-views using the method described in [3], and extract graphs from the images using the Delaunay triangulations of the detector corners. We apply the graph seriation process to the adjacency matrices of the extracted Delaunay graphs. Using the kernel mapping we map the strings to feature vectors in the kernel feature space.

We commence our study in Figure2 where we show the scatter plots of the string kernel distance $d(s,t)$ (SKD) versus the Bayes edit distance ED as the decay factor λ varies from 0.1 to 0.9. The main feature to note from these plots, is that as the decay factor is varied over this range, then there is a value for which the dispersion of the points in the scatter plots is minimised. In fact, the minimum dispersion occurs when $\lambda = 0.8$ which is close to the value ($\lambda = 0, 72$) predicted the edge density analysis.

Of course, the choice of the value of parameter decay parameter λ will effect the quality of clusters extracted from the pattern of pairwise distances. In Figure 3 we show the Davis-Bouldin cluster index[4] as a function λ. The value of λ computed from the edge-density is show as a red point and this is close to the minimum (i.e. best) value of the cluster index.

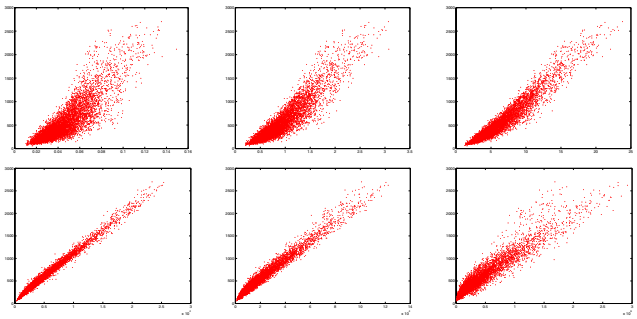

Figure 2. Scatter plot of string kernel distance versus ED for $\lambda = 0.1, 0.2, 0.3, 0.728, 0.8, 0.9$

In Figure 4 we show the pairwise distance matrix estimated using the string kernel method with the value of λ suggested by the edge-density analysis. The block structure corresponding to the different objects emerges strongly. Finally, Figure 5 shows the result of applying multidimensional scaling to the distances. The different symbols correspond to different objects, and the object clusters are well separated.

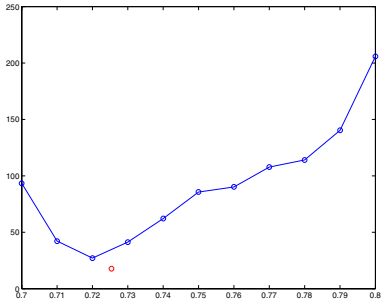

Figure 3. Davis-Bouldin index value

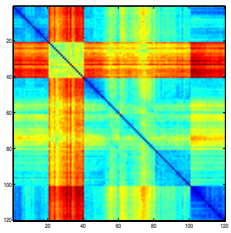

Figure 4. Edit distance matrix.

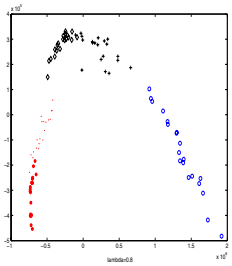

Figure 5. MDS on string kernel distances.

6 Conclusions

In this paper we have shown how string kernels can be used to match seriated graphs. We show the relationship between the decay parameter of the kernel and the edge density of the graphs being matched. Experiments show that the kernel distance is a good approximation to the Bayes graph edit distance, and that the computed distances can be used for graph-clustering.

References

[1] A.Robles-Kelly and E.R.Hancock. Graph matching using spectral seriation. *Energy Minimisation Methods in Computer Vision and Pattern Recognition*, pages 517–532, 2003.

[2] A.Robles-Kelly and E.R.Hancock. Graph Edit Distance from Spectral Seriation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, To appear, 2004.

[3] C.Harris and M.Stephens. A combined corner and edge detector. *Fourth Alvey Vision Conference*, pages 147–151, 1988.

[4] D.Davis and D.Bouldin. A cluster separation measure. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 1(2):224–227, 1979.

[5] Hang Yu and Edwin Hancock. Graph Seriation Using Semidefinite Programming. *GbRPR 2005*, pages 63–71, 2005.

[6] Huma Lodhi and Craig Saunders and John Shawe-Taylor and Nello Cristianini and Chris Watkins. Text classification using string kernels. *J. Mach. Learn. Res*, 2:419–444, 2002.

[7] Jonathan E. Atkins, Erik G. Boman and Bruce Hendrickson. A Spectral Algorithm for Seriation and the Consecutive Ones Problem. *SIAM Journal on Computing*, 28(1):297–310, 1998.

[8] K.Fujisawa,Y.Futakata,M.Kojima,K.Nakata and M.Yamashita. Sdpa-m user's manual. *http://sdpa.is.titech.ac.jp/SDPA-M*.

[9] L.Vandenberghe and S.Boyd. Semidefinite Programming. *SIAM Review*, 38(1):49–95, 1996.

[10] M.X.Goemans and D.P.WIlliamson. Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming. *J.ACM*, 42(6):1115–1145, 1995.

[11] N. Cristianini and J. Shawe-Taylor . *AN INTRODUCTION TO SUPPORT VECTOR MACHINES*. Cambridge University Press, 2000.