A Fast Method to Predict the Labeling of a Tree

Sergio Rojas Galeano and Mark Herbster

Department of Computer Science
University College London
Gower Street, London WC1E 6BT, UK
{M.Herbster,S.Rojas}@cs.ucl.ac.uk

Abstract. Given an n vertex weighted tree with (structural) diameter $S_{\mathbf{G}}$ and a set of ℓ vertices we give a method to compute the corresponding $\ell \times \ell$ Gram matrix of the pseudoinverse of the graph Laplacian in $O(n+\ell^2 S_{\mathbf{G}})$ time. We discuss the application of this method to predicting the labeling of a graph. Preliminary experimental results on a digit classification task are given.

1 Introduction

Classification methods which rely upon the graph Laplacian (see [2, 10, 5] and references therein), have proven to be useful for transductive and semi-supervised learning. A key insight of these methods is that unlabeled data can be used to improve the performance of supervised learners. These methods reduce to the problem of labeling a graph whose vertices are associated to the data points and the edges to the similarity between pairs of data points. The labeling of the graph can be achieved either in a batch [2, 10] or in a online manner [5]. These methods can all be interpreted as different kernel methods: ridge regression in the case of [2], minimal semi-norm interpolation in [10] or the perceptron algorithm in [5]. Therefore, in all cases an important preprocessing step is the computation of the kernel[1], which is the pseudoinverse of the graph Laplacian or a matrix related to it [5]. This computation usually scales cubically with the quantity of unlabeled data, which may prevent the use of these methods on large graphs. In the case of unbalanced bipartite graphs [7] presents a method which significantly improves the computation time of the pseudoinverse. Finally [3] presents a non-Laplacian-based method for predicting the labeling of tree based on computing the exact probabilities of a Markov random field.

In this paper, we are concerned with the practical scenario in which only a relatively small number of vertices of a large graph need to be labeled. Specifically, we divide the vertices into three sets, $V^{(\ell)}, V^{(p)}$, and $V^{(u)}$, which are the set of labeled vertices, the set of unlabeled vertices for which we will give predictions, and the set of remaining unlabeled vertices (no predictions is required on them), respectively. Therefore, if $V^{(n)}$ denotes the set of n vertices of the graph, we have that $n = \ell + u + p$, where ℓ is number of labeled vertices, p is the number

[1] Computation in the primal is also possible with a comparable time expenditure.

of *predictive* unlabeled vertices and u is the number of nonpredictive unlabeled vertices. Typically ℓ is much smaller than both p and u, and p is much smaller than u.

In this paper, we propose a technique to improve the computational complexity of Laplacian-based learning methods. The method is based on approximating the original graph with a tree. Computationally our method requires an $O(n)$ initialization step and after that any element of the pseudoinverse of the Laplacian of a tree may be computed in $O(S_{\mathbf{G}})$ time, where $S_{\mathbf{G}}$ is the structural diameter of the tree $\mathbf{G}$. The pseudoinverse of the Laplacian may then be used as a kernel [6, 5] for a variety label prediction methods. We consider as specific examples the kernel perceptron and ridge regression (see e.g., [9]). Thus if we are given an n vertex tree and assume that the time to compute the inverse of a matrix is cubic in the dimension then with ℓ labeled vertices and p predictive vertices the total time required to predict with the kernel perceptron is $O(n+\ell^2 S_{\mathbf{G}}+p\ell S_{\mathbf{G}})$ and with kernel ridge regression is $O(n+\ell^2 S_{\mathbf{G}}+\ell^3+p\ell S_{\mathbf{G}})$. The promise of our technique is that if $(l+p+S_{\mathbf{G}}) \ll n$ and a tree is given, our method requires $O(n)$ time versus $O(n^3)$ for standard methods and when only a similarity function on the data is given then our method requires $O(n^2)$ time and $O(n)$ space.

2 Preliminairies

In this paper any graph $\mathcal{G}$ is assumed connected, to have n vertices, and to have edge weights. The set of vertices of $\mathcal{G}$ is denoted $V = \{1, \ldots, n\}$. Let $\mathbf{A}$ denote the $n \times n$ symmetric nonnegative weight matrix of the graph such that $A_{ij} \geq 0$, and define the edge set $E(\mathcal{G}) := \{(i,j) : 0 < A_{ij},\, i < j\}$. We say that $\mathcal{G}$ is a *tree* if it is connected and has $n-1$ edges. The graph Laplacian $\mathbf{G}$ is the $n \times n$ matrix defined as

$$\mathbf{G} := \mathbf{D} - \mathbf{A},$$

where $\mathbf{D} = \mathrm{diag}(d_1, \ldots, d_n)$ and d_i is the *weighted* degree of vertex i, $d_i = \sum_{j=1}^{n} A_{ij}$. The Laplacian is positive semidefinite and induces the semi-norm

$$\|\mathbf{w}\|_{\mathbf{G}}^2 := \mathbf{w}^\top \mathbf{G} \mathbf{w} = \sum_{(i,j)\in E(\mathbf{G})} A_{ij}(w_i - w_j)^2. \tag{1}$$

The reproducing kernel [1] associated with the above semi-norm is $\mathbf{G}^+$, where "+" denotes pseudoinverse (see [6, 5] for further details). As the graph is connected, it follows from equation (1) that the null space of $\mathbf{G}$ is spanned by the constant vector $\mathbf{1}$ only. The weighted graph may be seen as a network of resistors where edge (i,j) is a resistor with resistance $\pi_{ij} = A_{ij}^{-1}$. Then the *effective resistance* $r_{\mathbf{G}}(i,j)$ may be defined as the resistance measured between vertex i and j in this network and may be calculated using Kirchoff's circuit laws or directly from $\mathbf{G}^+$ using [8]

$$r_{\mathbf{G}}(i,j) = \mathbf{G}^+_{ii} + \mathbf{G}^+_{jj} - 2\mathbf{G}^+_{ij}\,. \tag{2}$$

The effective resistance is a metric distance on the graph [8] as well as the geodesic $d_{\mathbf{G}}$ and structural $s_{\mathbf{G}}$ distances. The structural (geodesic) distance between vertices $i, j \in V$ is

$$s_{\mathbf{G}}(i,j) := \min\{|P(i,j)| : P(i,j) \in \mathcal{P}\},$$
$$d_{\mathbf{G}}(i,j) := \min\{\sum_{(p,q)\in\mathcal{P}(i,j)} \pi_{pq} : P(i,j) \in \mathcal{P}\},$$

where $\mathcal{P}$ is the set of all paths in $\mathcal{G}$ and $P(i,j)$ is the set of edges in a particular path from i to j. The diameter is the maximum distance between any two points on the graph hence the resistance, structural, and, geodesic diameter are

$$R_{\mathbf{G}} = \max_{i,j\in V} r_{\mathbf{G}}(i,j), \quad S_{\mathbf{G}} = \max_{i,j\in V} s_{\mathbf{G}}(i,j), \text{ and } \quad D_{\mathbf{G}} = \max_{i,j\in V} d_{\mathbf{G}}(i,j),$$

respectively.

3 Computing the Pseudoinverse of a Tree Quickly

In the following we give our method to compute the pseudoinverse of a tree. The principle of the method is that when a graph is a tree, there is a unique path between any two vertices hence the *effective resistance* is simply the sum of resistances along that path (see for example [8, 5]) and hence on trees the geodesic distance is equivalent to the resistance distance. We now additionally assume that $\mathcal{G}$ is a tree, the root vertex of the tree is indexed as 1. The parent of vertex i is denoted $\uparrow(i)$ while the children of i are $\downarrow(i)$ and the descendants of i are

$$\downarrow^*(i) := \begin{cases} \downarrow(i) \cup \sum_{j\in\downarrow(i)} \downarrow^*(j) & \text{if } \downarrow(i) \neq \emptyset \\ \emptyset & \text{if } \downarrow(i) = \emptyset \end{cases}.$$

We define $Z := \sum_{i=1}^n \mathbf{G}^+_{ii}$, $R(i) := \sum_{j\neq i} r_{\mathbf{G}}(i,j)$ and $R := \sum_{i=1}^n R(i)$. In equations (3) and (5) we give the formulas which we use to compute $\mathbf{G}^+$.The off-diagonal elements are computed with

$$\mathbf{G}^+_{ij} = \frac{\mathbf{G}^+_{ii} + \mathbf{G}^+_{jj} - r_{\mathbf{G}}(i,j)}{2}. \tag{3}$$

as follows from (2). Observe that

$$\mathbf{G}^+_{ii} = -\sum_{j\neq i} \mathbf{G}^+_{ij} \tag{4}$$

since the null space of $\mathbf{G}$ is spanned by the constant vector $\mathbf{1}$. Thus we may substitute (3) into (4) to obtain

$$\mathbf{G}^+_{ii} = -\frac{1}{2}\left[(n-1)\mathbf{G}^+_{ii} + \sum_{j\neq i} \mathbf{G}^+_{jj} - \sum_{j\neq i} r_{\mathbf{G}}(i,j)\right],$$

thus

$$\mathbf{G}^+_{ii} = \frac{R(i) - Z}{n} \text{ and } Z = \frac{R}{2n}. \tag{5}$$

We now describe a method to initially compute $\mathbf{G}^+_{ii}, R(i)$, $i = 1, \ldots, n, R$ and Z in $O(n)$ time and then with these precomputed values we may compute entries as needed in $\mathbf{G}^+_{ij}$ from equation (3) by computing $r_{\mathbf{G}}(i, j)$ in $O(S_{\mathbf{G}})$ time. The number of descendents of i (including i) is $\kappa(i) = 1 + |\downarrow^*(i)|$, while $T(i)$ (resp. $S(i)$) is the sum of the resistances of vertex i to each descendant (resp. non-descendent), hence,

$$T(i) = \sum_{j \in \downarrow^*(i)} r_{\mathbf{G}}(i, j)\,, \quad S(i) = \sum_{j \notin \downarrow^*(i)} r_{\mathbf{G}}(i, j)\,.$$

We compute $\kappa(i)$ and $T(i)$ (for $i = 1, \ldots, n$) with leaves-to-root recursions while we compute $S(i)$ with a root-to-leaves recursion. These $3n$ quantities are computed with the following recursions,

$$\kappa(i) := \begin{cases} 1 + \sum_{j \in \downarrow(i)} \kappa(j) & \downarrow(i) \neq \emptyset \\ 1 & \downarrow(i) = \emptyset \end{cases},$$

and

$$T(i) := \begin{cases} \sum_{j \in \downarrow(i)} (T(j) + \pi_{ij}\kappa(j)) & \downarrow(i) \neq \emptyset \\ 0 & \downarrow(i) = \emptyset \end{cases}$$

by computing $\kappa(1)$ then $T(1)$ and caching the intermediate values. We observe that $R(1) = T(1)$. We now descend the tree caching each calculated

$$S(i) := \begin{cases} S(\uparrow(i)) + T(\uparrow(i)) - T(i) + (n - 2\kappa(i))\pi_{i\,\uparrow(i)} & i \neq 1 \\ 0 & i = 1 \end{cases}.$$

Now as $R(i) = S(i) + T(i)$, the diagonal of $\mathbf{G}^+$ is calculated from (5) with a cumulative computation time of $O(n)$. We now observe to compute $\mathbf{G}^+_{ij}$ (see Equation (3)) we need $r_{\mathbf{G}}(i, j)$ which is simply the sum of resistances along the path from i to j, this path may be computing by separately computing the path from i–to–1 and j–to–1 ($O(S_{\mathbf{G}})$ time) and summing the resistances along each edge that is either in i–to–1 or j–to–1 but not both. In the full paper we will show that if we need to calculate an $\ell \times \ell$ submatrix of $\mathbf{G}^+$ this may be accomplished in $O(n + \ell^2 + \ell S_{\mathbf{G}})$ time.

4 Tree Construction

In the previous discussion, we have considered that a tree has already been given. In the following, we assume that a graph $\mathcal{G}$ or a similarity function is given and the aim is to construct an approximating tree. We will consider both the *minimum spanning tree* (`MST`) as a "best" in norm approximation; and the *shortest*

path tree (SPT) as an approximation which maintains a mistake bound [6, 5] guarantee. Moreover, we comment on the time and space complexity of constructing such trees. Given a graph with a "cost" on each edge the MST is a subgraph with $n-1$ edges such that the total cost is minimized. A SPT(i) at vertex i is a subgraph such that geodesic distance in "costs" is minimized from i to every other vertex. In our set-up the cost of edge (i,j) is π_{ij} therefore,

$$\texttt{MST}(\mathbf{G}) = \arg\min_{\mathbf{T}\subseteq\mathbf{G}}\{\sum_{(i,j)\in E(\mathbf{T})}\pi_{ij} : |\mathbf{T}| = n-1, \mathbf{T} \text{ is a tree}\},$$

$$\texttt{SPT}(\mathbf{G}, i) = \arg\min_{\mathbf{T}\subseteq\mathbf{G}}\{\sum_{j=1}^{n} r_{\mathbf{T}}(i,j) : |\mathbf{T}| = n-1, \mathbf{T} \text{ is a tree}\}.$$

Observe that for a graph with unit costs every tree is a MST but not necessarily a SPT. A MST is also the tree whose Laplacian best approximates the Laplacian of the given graph according to the trace norm, that is,

$$\texttt{MST}(\mathbf{G}) = \arg\min_{\mathbf{T}\subseteq\mathbf{G}}\{\|\mathbf{G}-\mathbf{T}\|_{\mathbf{tr}} : |\mathbf{T}| = n-1, \mathbf{T} \text{ is a tree}\}.$$

We now provide a justification for approximating the given graph by a SPT. It relies upon the analysis in [5, Theorem 4.2], where the cumulative number of mistakes of the kernel perceptron with the kernel $\mathbf{K} = \mathbf{G}^{+} + \mathbf{1}\mathbf{1}^{\top}$ was upper bounded by

$$|\mathcal{M}_A| \leq (\|\mathbf{u}\|^2_{\mathbf{G}} + 1)(R_{\mathbf{G}} + 1), \tag{6}$$

for consistent labelings $\mathbf{u} \in \{-1, 1\}^n$. To explain our argument, first we note that when we approximate the graph with a tree $\mathbf{T}$ the term $\|\mathbf{u}\|^2_{\mathbf{G}}$ is always decreasing, while the term $R_{\mathbf{G}}$ is always increasing by Rayleigh's monotonicity law (see for example [5, Corollary 3.1]). The resistance diameter $R_{\mathbf{T}}$ of an SPT subgraph is bounded by twice the geodesic diameter of the original graph,

$$R_{\mathbf{T}} \leq 2D_{\mathbf{G}}, \tag{7}$$

since for any path between p and q in the graph $\mathbf{G}$ there is in the SPT a path from p to the root and then to q which can be no longer than $2D_{\mathbf{G}}$. Thus, if the original graph was unweighted and consisted of a few dense clusters each uniquely labeled and with only a few cross-cluster edges, the tree built with a SPT would still have a non-vacuous mistake bound. This fact follows from equations (6) and (7). No such bound as (7) holds for a MST subgraph. For example, consider a bicycle wheel graph whose edge set is the union of n *spoke* edges $\{(0,i) : i = 1,\ldots,n\}$ and n *rim* edges $\{(i, i+1 \mod n) : i = 1,\ldots,n\}$ with costs on the spoke edges of 2 and on the rim edges of 1, the MST diameter is then $n+1$ while an SPT diameter is ≤ 8.

In general, we may not be given a tree rather we may be given a graph or a *similarity function* which may be then used to compute a tree. The MST and SPT trees may be constructed with Prim and Dijikstra algorithms [4] respectively in $O(n\log n + |E(\mathcal{G})|)$ time if implemented with a Fibonacci Heap. In the

general case of a non-sparse graph or similarity function the time complexity is $\Theta(n^2)$, however as both Prim and Dijikstra are "greedy" algorithms their space complexity is $O(n)$ which may be dominant consideration in a large graph.

5 Experiments

We provide results on preliminary experiments to study the feasibility of our methods. In this exploratory study we have not yet implemented the fast computation technique described. Rather the aim of these experiments is to see if there is a significant performance decrease in using a tree subgraph rather than the original graph. The initial results are promising as we find (see Figure 1) that the accuracy of the predictor with an `MST` approximation is competitive with the original graph.

We performed experiments using the USPS digits dataset, with the aim of classifying '1' vs. '2' and then '3' vs. '8'. Each OCR'd digit (a 16×16 array with 256 levels of gray) is represented as a vector $\mathbf{x} \in [-1, 1]^{256}$. The adjacency matrix for the graph was set to $A_{ij} = \exp(-a\|\mathbf{x}_i - \mathbf{x}_j\|^2)$. The parameter a was then selected by grid search as the value which optimized generalization performance of the complete weighted graph ($n = 1024$) for both digit recognition tasks ('1' vs. '2' : $a = 0.1$; '3' vs. '8' : $a = 0.05$). `MST` and `SPT` weighted subgraphs were then constructed. For each of the three graphs we computed the graph kernel $\mathbf{K} = \mathbf{G}^+ + \mathbf{1}\mathbf{1}^\top + \mathbf{I}$ (where $\mathbf{I}$ is the identity matrix) as discussed in [5, equation (10)]. A training set $V^{(\ell)}$ of 8 labeled points of 4 positives and 4 negatives was randomly selected. A kernel perceptron was then trained for 3 epochs on $V^{(\ell)}$. The accuracy of each of the three classifiers was then measured on a randomly selected predictive set $V^{(p)}$ of 50 points. This protocol was performed for $n = 128, 256, 512, 1024$. Each such experiment was repeated 50 times, the mean accuracy is accordingly reported in Figure 1.

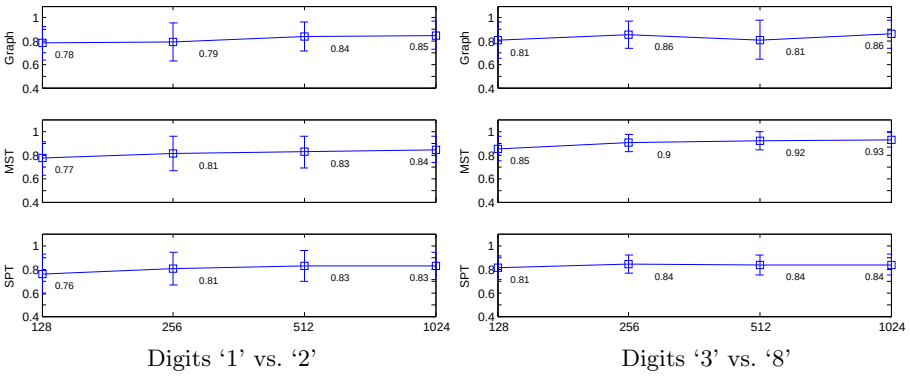

Fig. 1. Digit Classification (**Left:** Accuracy; **Bottom:** Vertex Set Size $|V^{(n)}|$)

The motivation for our methodology is the idea that accuracy can be improved by increasing the quantity of unlabeled data even if the quantity of labeled data is fixed at only a linear increase in computation cost. Experimentally this trend of increasing accuracy is seen for the graph and both the `MST` and the `SPT`; however this result is not conclusive as it may be an artifact of the tuning procedure for a and hence deserves further study. Interestingly the performance of the `MST` versus the complete weighted graph was found to be better in 5 out of 8 experiments even though the parameter a was optimally tuned for the complete graph. The `SPT` though competitive with the complete graph consistently under performed relative to the `MST`. We believe that these initial results show promise for our technique and we plan to implement the methods of Section 3 to compute the graph Laplacian pseudoinverse in order to scale our experiments to larger datasets.

Acknowledgements: We would like to thank José Luis Balcázar for valuable discussions and Massimiliano Pontil for valuable discussions and detailed comments on this manuscript. Finally, we would also like to thank Dimitrios Athanasakis and Sudhir Shankar Raman for useful preliminary experimentation. This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778 and Sergio Rojas Galeano is supported under the DHPA Research Councils UK Scheme.

References

1. N. Aronszajn. Theory of reproducing kernels. *Trans. Amer. Math. Soc.*, 68:337–404, 1950.
2. M. Belkin and P. Niyogi. Semi-supervised learning on riemannian manifolds. *Machine Learning*, 56:209–239, 2004.
3. A. Blum, J. Lafferty, M. R. Rwebangira, and R. Reddy. Semi-supervised learning using randomized mincuts. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, page 13, New York, NY, USA, 2004. ACM Press.
4. T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 1990.
5. M. Herbster and M. Pontil. Prediction on a graph with a perceptron. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 577–584. MIT Press, Cambridge, MA, 2007.
6. M. Herbster, M. Pontil, and L. Wainer. Online learning over graphs. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 305–312, New York, NY, USA, 2005. ACM Press.
7. N.-D. Ho and P. V. Dooren. On the pseudo-inverse of the laplacian of a bipartite graph. *Appl. Math. Lett.*, 18(8):917–922, 2005.
8. D. Klein and M. Randić. Resistance distance. *Journal of Mathematical Chemistry*, 12(1):81–95, 1993.
9. J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.
10. X. Zhu, J. Lafferty, and Z. Ghahramani. Combining active learning and semi-supervised learning using gaussian fields and harmonic functions. In *In Proc. of the ICML 2003 workshop on The Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*, pages 58–65, 2003.