This paper was submitted as a final project report for CS6424/ECE6424 *Probabilistic Graphical Models and Structured Prediction* in the spring semester of 2016.

The work presented here is done by students on short time constraints, so it may be preliminary or have inconclusive results. I encouraged the students to choose projects that complemented or integrated with their own research, so it is possible the project has continued since this report was submitted. If you are interested in the topic, I encourage you to contact the authors to see the latest developments on these ideas.

Bert Huang
Department of Computer Science
Virginia Tech

Predicting graph structure using local information and path-based measures in networks: A Metric Learning approach

Subhodip Biswas and Bert Huang
Department of Computer Science
Virginia Polytechnic and State University
Blacksburg, VA 24061
suhodip@vt.edu

Abstract

Nodes in a network contain local information in the form of *attributes* in addition to path-based measures available from the network. Metric learning techniques aimed at learning a distance metric $\mathbf{M}$ can help us to utilize this *local information* for predicting a graph structure. This research article proposes a metric-learning based approach that aims to align Mahalanobis distance among the nodes to have roughly the same order as determined by the path-based measures (like commute-time). Additionally we test our approach on a synthetic dataset and evaluate them on three performance metrics. To the best of our knowledge this is the first attempt to combine local information and path-based measure in learning the graph structure.

1 Introduction

Metric learning is closely associated with similarity learning in supervised machine learning [13, 2]. It is defined as the problem of learning a distance function tuned to a particular task. Metric learning aims to automate the process of learning a task-specific distance functions in a supervised manner [6]. Though metric learning has been widely used in the area of computer vision, it has also found applications in other domains as well. A recent work by Shaw *et al.* [11] has applied metric learning to learn a distance function in real-world networks. They proposed *structure preserving metric learning* (SPML) algorithm in one of the foremost attempts to learn a distance metric by capturing both local attribute and connectivity information often available in real-world networks. However, the connectivity information looks at only the absence or presence of links (or edges) between the nodes in the network. Though this binary information is useful in modelling networks, very few works have focused in going beyond this connectivity. An interesting finding with respect to the *link prediction* problem was reported in [7]. The link prediction asks the question as to "what extent can the evolution of a social network be modeled using features intrinsic to the network itself"? From a series of detailed experiment, the authors opined that *network topology does indeed contain latent information from which to infer future interactions*. In fact, it empirically demonstrated that more subtle path-based measures like commute-time [8] and Katz-measure [5] often outperform more easily computable direct measures such as shortest-path distances and numbers of shared neighbors. But it comes at a cost. These path-based measures are expensive to compute given the graph $G\left(V, E\right)$. Increase in computation time is the price paid for better expressibility. However, there has been recent works [1, 9] to ameliorate this computational problem by proposing faster approximations of commute-time and Katz measure with tighter bounds.

Motivated by the above findings and following the line of work done in [11], we make an attempt learn a distance metric $\mathbf{M}$ that incorporate the path-based information and nodal interactions (via feature of nodes). Thus we propose *commute-time metric learning* (CTML) algorithm. One of the important

Probabilistic Graphical Models and Structured Prediction Final Project Report. Spring 2016, Virginia Tech.

difference between SPML and CTML lies in incorporating the relative distance constraints (triplets) in the metric learning problem. While the former attempts to push the non-neighbor nodes farther than the neighbors of a particular node, the latter attempts to align the other nodes in feature space such that they closely follow the commute-time orderings. In relation to their different objectives, new performance metrics have been used to evaluate the CTML algorithm. In short, the contributions of our work are as follows:

1. We show that metric learning can be applied such that the commute-time orderings align with the Mahalanobis distance orderings (in feature space);
2. An artificial dataset was generated such that commute-time exactly aligns with the squared-Euclidean distance; and
3. New performance metrics are introduced to measure the degree of alignment between the commute-time orderings and the Mahalanobis distance orderings of the data-points.

The remainder of the paper are organized as follows: In Section 2, we give the background details of the metric learning problem, it's modification to network setting as was done in SPML and discuss about the path-based measures. The proposed CTML algorithm is detailed in Section 3 and is followed it up with artificial dataset generation process in Section experiment for testing our algorithm. Alongside, we also discuss the associated performance metrics for evaluating CTML. The experiments section details about the performance of CTML in different datasets and observations are analyzed in detail. Lastly we conclude our paper with in Section 5 with some possible directions for future line of research.

2 Background details

2.1 Metric learning problem formulation

Mahalanobis distance, as defined in the metric learning literature, is any distance function of the form:

$$D_{\mathbf{M}}(x_i, x_j) = (x_i - x_j)^{\mathrm{T}} \mathbf{M} (x_i - x_j) \tag{1}$$

where $\mathbf{M}$ is a positive semi-definite matrix, $x_i, x_j \in \mathbb{R}^d$ are two $d-$dimensional training instances. As with the original Mahalanobis distance, we can view this distance simply as applying a linear transformation of the input data. Thus, Mahalanobis distance exactly captures the idea of learning a global linear transformation. Hence the above transformation allows linear constraints on the distances to be written as linear constraints on the $\mathbf{M}$ matrix.

2.2 Structure Preserving Metric Learning (SPML) algorithm

In [12], it was shown that graph topology can be preserved using linear constraints with regard to the distance between the nodes in feature space. To understand this with regard to graph connectivity, let us consider a graph $G(V, E)$ for which we know the adjacency matrix $\mathbf{A} \in \mathbb{B}^{n \times n}$, and the feature of n nodes from $X \in \mathbb{R}^{d \times n}$. If we can learn a distance metric $\mathbf{M}$ parameterized by the linear constraints, we are basically learning a metric that preserves the graph structure. The connectivity algorithm $\mathcal{G}$ would use the given matrix of node features $\mathbf{X}$ and the distance metric $\mathbf{M}$ (learnt by imposing the constraints) to output an adjacency matrix $\tilde{\mathbf{A}}$. The graph structure is said to be preserved if the $\tilde{\mathbf{A}} = \mathbf{A}$. Kindly refer to Definition 1 in [11].

SPML builds on the above idea whereby it learns a distance metric parameterized by a PSD matrix $\mathbf{M} \in \mathbb{R}^{d \times d}$, where $\mathbf{M} \succeq 0$ that preserves the structure. Note that we talked about connectivity information previously. In SPML, these connectivity information supplies the constraints that must be enforced whie trying to learn $\mathbf{M}$. The constraint is *the distance of a node to its farthest connected neighbor should be less than the distance to nearest unconnected non-neighbor*. By taking the $k-$nearest neighbor connectivity algorithm this constraints can be enforced in the transformation (1)to learn a structure preserving $\mathbf{M}$.

The optimization problem is formulated by as the sum of two terms: a Frobenius norm (denoted by $\|.\|_F$) regularizer on $\mathbf{M}$, and hinge loss on the linear constraints. The regularization parameter λ is the trade-off between the two terms and penalizes too complex $\mathbf{M}$. To preserve the entire structure

it should consider all the possible triplets and the problem takes the form of a simple semidefinite program (SDP). However, the worst-case complexity of the SDP would be $\mathrm{O}\left(d^3 + C^3\right)$, where $C = \mathrm{O}\left(N^2\right)$ for d variables and N constraints (is of order $\mathrm{O}(n^3)$ for training set of size n). In [11], it was mentioned that we can ignore the PSD requirement on $\mathbf{M}$ since we are concerned with the relative distances. This helped to convert the problem into a one-class structural SVM and that facilitated the use of stochastic SVM algorithms like PEGASOS [10].

As we want the constraints to preserve the structure, we should incorporate a penalty in case the structure is violated by the $\mathbf{M}$ learnt so far. The hinge-loss over triplets $\max\left(D_{\mathbf{M}}\left(x_i, x_j\right) - D_{\mathbf{M}}\left(x_i, x_k\right) + 1, 0\right)$ does exactly that. It considers one reference node (say i), its neighbor (say j) and its non-neighbor (say k). The set of all such triplets is $S = (i, j, k)\,|A_{ij} = 1, A_{ik} = 0$. Given below is a unconstrained optimization problem with a hinge-loss constraints encoding the structure:

$$f\left(\mathbf{M}\right) = \frac{\lambda}{2}\left\|\mathbf{M}\right\|_F^2 - \frac{1}{|S|}\sum_{(i,j,k)\in S}\max\left(D_{\mathbf{M}}\left(x_i, x_j\right) - D_{\mathbf{M}}\left(x_i, x_k\right) + 1, 0\right) \tag{2}$$

Learning the distance metric over the set of all constraints is the main bottle-neck. Not all triplets (i, j, k) also satisfy the structural condition $A_{ij} = 1, A_{ik} = 0$. So we can make SPML scalable via the projected stochastic subgradient descent. The authors in [11] have shown a fast way to evaluate these by using sparse structure. The $|S|$ constraints can be written using the distance transformation in (1) as a sparse matrix $\mathbf{C}^{(i,j,k)}$, where

$$\mathbf{C}_{jj}^{(i,j,k)} = 1, \quad \mathbf{C}_{ik}^{(i,j,k)} = 1, \quad \mathbf{C}_{ki}^{(i,j,k)} = 1, \quad \mathbf{C}_{ij}^{(i,j,k)} = -1, \quad \mathbf{C}_{ji}^{(i,j,k)} = -1, \quad \mathbf{C}_{kk}^{(i,j,k)} = -1, \quad .$$

The other entries are zero. The matrix $\mathbf{C}^{(i,j,k)}$ ensures that the expression $\mathrm{tr}\left(\mathbf{C}^{(i,j,k)}\mathbf{X}^{\mathrm{T}}\mathbf{M}\mathbf{X}\right)$ is equal to hinge-loss penalty function. The subgradient of $f\left(\mathbf{M}\right)$ at $\mathbf{M}$ is then

$$\nabla f = \lambda\mathbf{M} + \frac{1}{|S|}\sum_{(i,j,k)\in S_+}\left(\mathbf{X}\mathbf{C}^{(i,j,k)}\mathbf{X}^{\mathrm{T}}\right) \tag{3}$$

where $S_+ = \{(i, j, k)\,|D_{\mathbf{M}}\left(x_i, x_j\right) - D_{\mathbf{M}}\left(x_i, x_k\right) + 1 > 0\}$. As mentioned earlier, the set of all possible triplets is the reason for computational burder. Therefore SPML uses stochastic subgradient descent to optimize this objective function via a batch of triplets instead of using the full set S. Thus S in the objective function is replaced with a random subset of S of B triplets each of which satisfies the property $A_{ij} = 1, A_{ik} = 0$. Interested readers are referred to [11] for detailed understanding. We will revisit this section while describing the CTML algorithm.

2.3 Path-based measures

As mentioned earlier Liben-Nowell and Kleinberg [7] identified a variety of topological measures as features for link prediction. The measures they studied fall into two categories – neighborhood-based measures and path-based measures. The former are cheaper to compute, yet the latter are more effective at link prediction. In fact, Katz measure and commute-time were found to be among the most effective ones. Here we will briefly go over the two measures for both

Katz score: Katz score computes a weighted sum of the number of paths between them and is the measure the affinity between nodes. Formally, the Katz score between node i and j is

$$K_{i,j} = \sum_{l=1}^{\infty}\alpha^l \mathrm{paths}_l\left(i, j\right)$$

where $\mathrm{paths_l\left(i, j\right)}$ denotes the number of paths from node i to j of length l and $\alpha < 1$ is an attenuation parameter. For a symmetric adjacency matrix $\mathbf{A}$ of a undirected and connected graph, $A_{i,j}^l$ is the number of paths between node i and j. Then the Katz scores for all pairs of nodes can be expressed as:

$$\mathbf{K} = \alpha\mathbf{A} + \alpha^2\mathbf{A}^2 + +\alpha^2\mathbf{A}^2..... = \left(\mathbf{I} - \alpha\mathbf{A}\right)^{-1} - \mathbf{I} \tag{4}$$

$\mathbf{K}$ refers to the Katz matrix. The restriction here is $\mathbf{I} - \alpha\mathbf{A}$ must be positive definite for it to be invertible. This happens when $\alpha < {}^{1}/_{\sigma_{\max}(\mathbf{A})}$ ($\sigma_{\max}$ is the largest singular value of $\mathbf{A}$) and also corresponds to the case where the series expansion converges.

Commute time: It is defined as the sum of hitting times from i to j and from j to i, where the hitting time from node i to j is the expected number of steps for a random walk started at i to visit j for the first time. The hitting time is computed using the random walk transition matrix (first-transition analysis) associated with a graph. Let $\mathbf{A}$ be the symmetric adjacency matrix from which we can get the diagonal matrix of degrees $\mathbf{D}$ as

$$D_{i,j} = \begin{cases} \sum_v A_{i,v} & i = j \\ 0 & \text{otherwise} \end{cases}$$

The random walk transition matrix $P = \mathbf{D}^{-1}\mathbf{A}$. If $H_{i,j}$ be the hitting time from node i and j, it must satisfy

$$H_{i,j} = 1 + \sum_v H_{i,v} P_{v,j} \quad \text{and} \quad H_{i,i} = 0$$

based on the Markovian nature of a random walk. The hitting time matrix is the minimum non-negative solution $\mathbf{H}$ that satisfies this equation . The commute time between node i and j is then: $C_{i,j} = H_{i,j} + H_{j,i}$. Then the commute-time matrix $\mathbf{C}$ can be calculated as $\mathbf{C} = \mathbf{H} + \mathbf{H}^T$. An equivalent expression can be obtained based on the combinatorial graph Laplacian matrix: $\mathbf{L} = \mathbf{D} - \mathbf{A}$ [4]. Each element $C_{i,j} = Vol(G)(L^\dagger_{i,i} - 2L^\dagger_{i,j} + L^\dagger_{j,j})$ where $Vol(G)$ is the number of edges in the graph and $\mathbf{L}^\dagger$ is the pseudo-inverse of $\mathbf{L}$. The commute time between nodes in different connected components is infinite. So we use the largest connected component of a graph for analysis.

In this article we deal with commute time only. However for a connected component of the graph Katz measure can also be incorporated into the algorithmic framework similarly. Since we use synthetic dataset we don't consider the problem of scalability.

3 Commute-time based Metric Learning (CTML)

The CTML algorithm builds on the idea of SPML by tweaking the triplet selection process. Remember in SPML we chose the set of all such triplets as $S = \{(i,j,k)\,|A_{ij} = 1, A_{ik} = 0\}$. In CTML, we look beyond the connectivity information provided by the adjacency matrix $\mathbf{A}$ and use higher level of information provided by the commute-time matrix $\mathbf{C}$. We deal with the set of all such triplets $S = \{(i,j,k)\,|C_{ij} < C_{ik}\}$ and the associated penalty function $\max\left(D_{\mathbf{M}}\left(x_i, x_j\right) - D_{\mathbf{M}}\left(x_i, x_k\right) + 1, 0\right)$ is kept the same. The expectation is that the distance to node j (that has less commute-time) should be less than node k (with higher commute time). If this expectation is not satisfied we penalize the constraint using the penalty function defined in Eq. (2). The pseudocode of the algorithm with commute-time constraints is given in Algorithm 1.

input : Commute time matrix $\mathbf{C} \in \mathbb{R}^{n,n}$, dataset $\mathbf{X} \in \mathbb{R}^{d,n}$, and parameters λ, T,B
output : $\mathbf{M}_t$

for $t \leftarrow 1$ **to** $T-1$ **do**
  $\eta_t \leftarrow \frac{1}{\lambda t}$;
  $\mathbf{C}^{(i,j,k)} \leftarrow \mathbf{0}_{n,n}$;
  for $b \leftarrow 1$ **to** B **do**
    $(i,j,k) \leftarrow$ Sample random triplet from $S = \{(i,j,k)\,|C_{ij} < C_{ik}\}$;
    if $D_{\mathbf{M}}\left(x_i, x_j\right) - D_{\mathbf{M}}\left(x_i, x_k\right) + 1 > 0$ **then**

$$\mathbf{C}^{(i,j,k)}_{jj} = \mathbf{C}^{(i,j,k)}_{jj} + 1, \mathbf{C}^{(i,j,k)}_{ik} = \mathbf{C}^{(i,j,k)}_{ik} + 1, \mathbf{C}^{(i,j,k)}_{ki} = \mathbf{C}^{(i,j,k)}_{ki} + 1,$$
$$\mathbf{C}^{(i,j,k)}_{ij} = \mathbf{C}^{(i,j,k)}_{ij} - 1, \mathbf{C}^{(i,j,k)}_{ji} = \mathbf{C}^{(i,j,k)}_{ji} - 1, \mathbf{C}^{(i,j,k)}_{kk} = \mathbf{C}^{(i,j,k)}_{kk} - 1,$$

    end
  end
  $\nabla_t \leftarrow \mathbf{X}\mathbf{C}^{(i,j,k)}\mathbf{X}^T + \lambda \mathbf{M}_t$;
  $\mathbf{M}_{t+1} \leftarrow \mathbf{M}_t - \eta_t \nabla_t$;
end

Algorithm 1: CTML algorithm

4 Experiments

In this section we compare the performance of SPML and CTML on synthetic datasets on three metrics. The synthetic dataset generation is elucidated in Section 4.1 followed by discussions on performance metrics in Section 4.2. Finally we discuss the results in Section 4.3 and bring out observations that may be helpful in guiding future research.

4.1 Synthetic dataset generation

The synthetic dataset $\mathbf{X}$ is generated such that the squared Euclidean distance between any two nodes exactly resemble with the commute-time between them. In short the dataset should satisfy the property $(x_i - x_j)^{\mathrm{T}} (x_i - x_j) \approx C_{ij}$. We briefly discuss the generation process. Remember in Section 2.3 we have shown that the commute-time between pair of nodes can be expressed as $C_{i,j} = Vol(G)(L^{\dagger}_{i,i} - 2L^{\dagger}_{i,j} + L^{\dagger}_{j,j})$, where $Vol(G)$ is the sum of elements in $\mathbf{A}$ and $\mathbf{L}^{\dagger}$ is the pseudo-inverse of $\mathbf{L}$. This enables us to construct a Kernel matrix for commute time embeddings as $\mathbf{K}_C = Vol(G) \cdot \mathbf{L}^{\dagger}$. Symmetric matrix factorization of $\mathbf{K}_C = \mathbf{X}^{\mathrm{T}}\mathbf{X}$ leads to the formation of the dataset $\mathbf{X}$. We use eigen-decomposition to obtain $\mathbf{X}$. Mathematically the dataset generation can be written as

$$\mathbf{X}^{\mathrm{T}}\mathbf{X} = VDV^{\mathrm{T}} = K_{\mathbf{C}} \quad => \mathbf{X} = D^{1/2}\ V^{\mathrm{T}} \tag{5}$$

where D is a digonal matrix containing the eigenvalues and V is the matrix of eigenvectors. The dot-product Kernel $\mathbf{K}$ exactly resembles $\mathbf{K}_C$. Since the commute-time embeddings exactly resemble the Euclidean distance embeddings, we make the problem more challenging by scrambling the dataset using an invertible matrix $W \in \mathbb{R}^{d \times d}$ containing uniform random numbers lying in the range $[0, 1]$. So now we have a scrambled matrix $\tilde{\mathbf{X}} = W \cdot \mathbf{X}$. The metric learning algorithm should ideally learn the Mahalanobis metric $\mathbf{M}_{opt} = W^{-1}$.

Note that we use a connected graph $G(V, E)$ by artificially generating the adjacency matrix $\mathbf{A}$. The commute-time matrix $\mathbf{C}$ is computed from $\mathbf{A}$ and then we assign the features to the node using Eq. (5). As mentioned earlier there exists an optimal $\mathbf{M} = \mathbf{M}_{opt}$. The scrambling of the dataset does not affect the commute-time between the nodes. We will now discuss two performance metrics that is based on the learning problem in hand and has direct relation to this generation process.

4.2 Performance measures

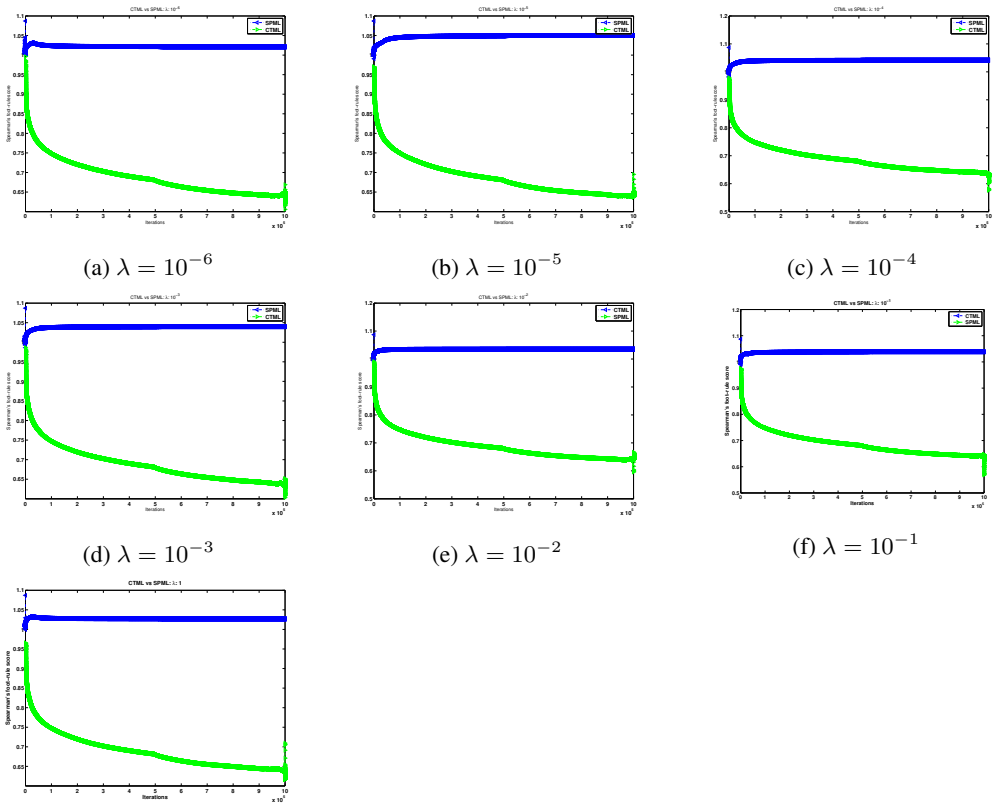
As the optimization is running we are trying to attain two objectives simultaneously. The Mahalanobis metric $\mathbf{M}_t$ should ideally approach the $\mathbf{M}_{opt} = W^{-1}$ and the relative distance-based rankings between the node should approach the relative commute-time based rankings. The performance metrics discussed below are designed to measure this two objectives.

Spearman's footrule score : This is a more strict measure that averages the absolute sum of the commute-time rankings and Mahalanobis distance-based ranking (as determined by $\mathbf{M}_t$) over all the nodes. Suppose $r^{i,j}_{\mathbf{M}}$ is the rank of node j when all nodes are sorted according to the Mahalanobis distance from node i. Similarly $r^{i,j}_{\mathbf{C}}$ be the rank of node j when sorting done based on commute-time with respect to node i. Then the Spearman's footrule score is approximated as

$$S_{\mathbf{M},\mathbf{C}} = \frac{1}{n^3} \sum_{i=1}^{n} \sum_{j=1}^{n} \left| r^{i,j}_{\mathbf{M}} - r^{i,j}_{\mathbf{C}} \right| \tag{6}$$

As the $\mathbf{M}_t$ approaches the optimal solution, we should expect the rankings to align and the score $S_{\mathbf{M}_t,\mathbf{C}}$ should approach 0.

The above measure is very exact and measures exact alignment. If the algorithm is made to run infinitely long, we can get perfect scores. However with regard to the stochasticity of the sub-gradient descent algorithm, we should use a comparatively less stringent metric to evaluate the objective. Area under the curve (AUC) scores have been used in [11] to determine the performance of the SPML



(a) $\lambda = 10^{-6}$ (b) $\lambda = 10^{-5}$ (c) $\lambda = 10^{-4}$

(d) $\lambda = 10^{-3}$ (e) $\lambda = 10^{-2}$ (f) $\lambda = 10^{-1}$

(g) $\lambda = 1$

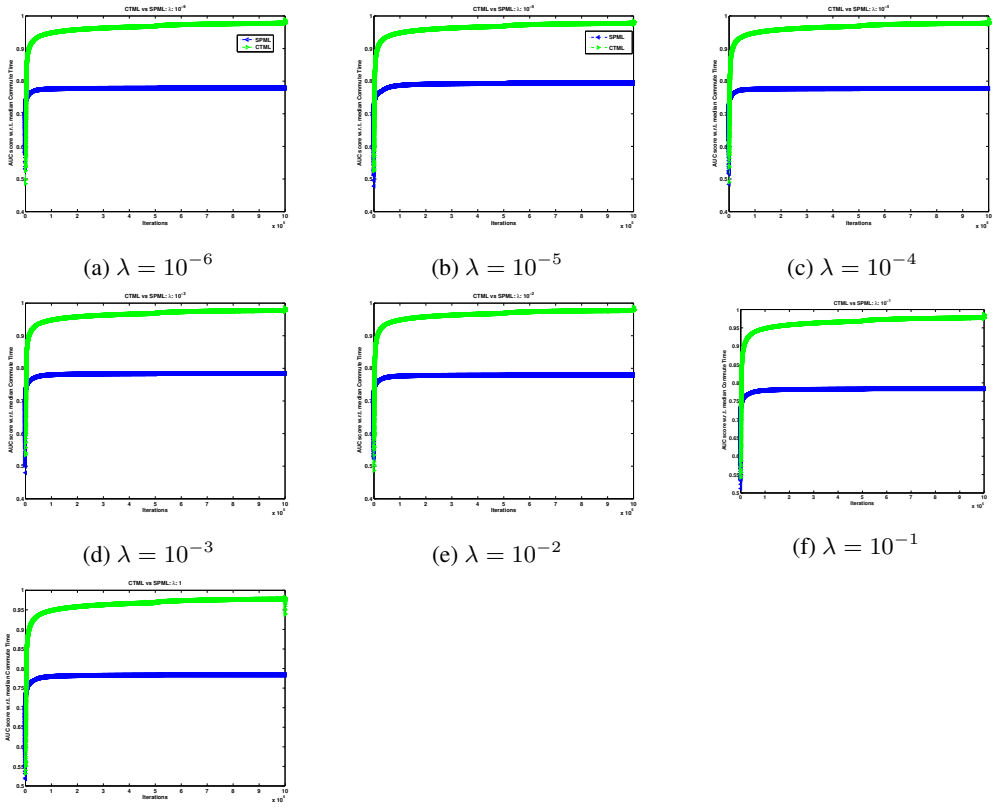
Figure 1: The Spearman's footrule score for CTML (green) and SPML (blue) for different values of the regularization parameter λ. (Zoom on the pdf to view.)

algorithm in link prediction. However CTML was designed to obtain commute-time alignment. So we must use AUC scores to consider the role of commute-time.

CTAp-measure : Thus we introduce customizable CTAp-measure (Commute-time based AUC score at p percentile). This score is based on the concept that this problem can be treated as a binary classification problem with respect to a *threshold* commute-time fixed at p-percentile with respect to each node. In this case we set $p = 50$, and take the median commute time as threshold. With respect to a node i, we find the median commute-time to all other nodes. Let this be denoted by c_i^{med}. All the nodes whose commute time is less than c_i^{med} has labels 1 and 0 otherwise. The binary label prediction is based on the analogous rule, the median distance serving as the threshold for assigning labels. The AUC scores of this binary classification is computed for all the nodes and the average measure is stored for every iteration. As the optimization iteration runs, the median commute-time is expected to approach the median distance for all the nodes leading to higher AUC scores.

4.3 Results on the synthetic dataset

Both CTML and SPML are based on the same optimization principle with the only difference between them being in triplet selection. In this section we show how this change in triplet selection affects the learning. For both the algorithms we used the synthetic dataset $\mathbf{X} \in \mathbb{R}^{d \times n} (d = 20, n = 21)$ and kept the batch-size B at 10, the margin for violation was 10^{-10}. Both the algorithms were made to run for $T = 10^6$ iterations. By keeping the other parameters fixed, the regularization parameter λ is varied in powers of 10 from -6 to 0 in steps of 1 and plot the relative performance of the algorithms are shown as per the three measures. The Spearman's footrule score and CTAp-measure at $p = 50$ are plotted in Figure 1 and 2 respectively.



(a) $\lambda = 10^{-6}$ (b) $\lambda = 10^{-5}$ (c) $\lambda = 10^{-4}$

(d) $\lambda = 10^{-3}$ (e) $\lambda = 10^{-2}$ (f) $\lambda = 10^{-1}$

(g) $\lambda = 1$

Figure 2: The CTAp score ($p = 50$) for CTML (green) and SPML (blue) for different values of the regularization parameter λ. (Zoom on the pdf to view.)

From our analysis of the run we see that using only binary information is not helpful when we are going beyond connections. In real-world networks, sometimes two unconnected nodes might have lower commute-time but with no connection between them. That would usually imply a big set of common neighbors and we can expect that in future a link might form between them through these common connections. Many such interesting questions can be answered if we start looking into path-based measures along with the local information available from the nodes. However as discussed earlier there is the issue of scalability that arises in real-life networks as they tend to grow in size. In that case we have to use approximations of these path-based measures. It makes more sense to use approximations rather than exact measures since real-life networks (graph structure) tends to change with time. These changes basically involve addition of new nodes or edges, as well as deletion. Recomputing the exact measures every time such a change occurs is not practical. Thanks to the recent algorithms proposed in [1], investing in metric learning algorithms like CTML has become feasible as more and more research continue to be done in network science.

Another interesting research might be in developing optimization algorithms that can lead to faster convergence using only a subset of triplets. CTML's convergence with respect to the Spearman's footrule is in Figure 1. This is based on the PEGASOS theory [10]. However more recent line of optimizations like ADAGRAD [3] remains to be explored. The bigger goal would be to formulate a line of algorithms that have a general framework for incorporating path-based measures and local information in predicting the graph structure.

5 Conclusion

In this report we propose a commute-time based metric learning algorithm that tries to learn the structure of graph by incorporating the nodal-attributes and the path-based measure. This algorithm

is based on the idea that topological measures help to learn a deeper notion of structure than is otherwise reflected by simple connectivity information. We also use two measures to show the relative performance of the two algorithms. It brings out the inadequacy of metric learning when we use the binary information and do not know use the path-based measures.

We also provide guidelines for possible future explorations in metric learning algorithms.

References

[1] Francesco Bonchi, Pooya Esfandiar, David F Gleich, Chen Greif, and Laks VS Lakshmanan. Fast matrix computations for pairwise and columnwise commute times and katz scores. *Internet Mathematics*, 8(1-2):73–112, 2012.

[2] Gal Chechik, Varun Sharma, Uri Shalit, and Samy Bengio. Large scale online learning of image similarity through ranking. *J. Mach. Learn. Res.*, 11:1109–1135, March 2010.

[3] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159, 2011.

[4] Francois Fouss, Alain Pirotte, Jean-Michel Renders, and Marco Saerens. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Trans. on Knowl. and Data Eng.*, 19(3):355–369, March 2007.

[5] Leo Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, 1953.

[6] Brian Kulis. Metric learning: A survey. *Foundations and Trends® in Machine Learning*, 5(4):287–364, 2013.

[7] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7):1019–1031, 2007.

[8] Ulrike V. Luxburg, Agnes Radl, and Matthias Hein. Getting lost in space: Large sample analysis of the resistance distance. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 2622–2630. Curran Associates, Inc., 2010.

[9] Purnamrita Sarkar and Andrew Moore. A tractable approach to finding closest truncated-commute-time neighbors in large graphs. *arXiv preprint arXiv:1206.5259*, 2012.

[10] Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical programming*, 127(1):3–30, 2011.

[11] Blake Shaw, Bert Huang, and Tony Jebara. Learning a distance metric from a network. In J. Shawe-Taylor, R.S. Zemel, P.L. Bartlett, F. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 1899–1907. Curran Associates, Inc., 2011.

[12] Blake Shaw and Tony Jebara. Structure preserving embedding. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 937–944, New York, NY, USA, 2009. ACM.

[13] Eric P. Xing, Michael I. Jordan, Stuart J Russell, and Andrew Y. Ng. Distance metric learning with application to clustering with side-information. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 521–528. MIT Press, 2003.