Metric Nearness: Problem Formulation and Algorithms

Inderjit S. Dhillon **Suvrit Sra**
Dept. of Computer Sciences
The University of Texas at Austin
Austin, TX 78712.
{inderjit,suvrit}@cs.utexas.edu

Joel A. Tropp
ICES
The University of Texas at Austin
Austin, TX, 78712.
jtropp@ices.utexas.edu

Abstract

Many problems in machine learning, data mining, databases and statistics involve the pairwise dissimilarities among a set of objects. It is often desirable for these dissimilarities to satisfy the properties of a metric—especially the triangle inequality. Applications where metric data are crucial include clustering, classification, metric-based indexing, query processing, and approximation algorithms. This paper presents the *Metric Nearness Problem*: Given a dissimilarity matrix, find the "nearest" matrix of distances that satisfy the triangle inequalities. For ℓ_p nearness measures, the paper develops efficient algorithms that compute globally optimal solutions by exploiting the deep structure of the problem. Empirically, the algorithms have time and storage costs that are linear in the number of triangle constraints. The methods can easily be parallelized for additional speed.

1 Introduction

Imagine that a lazy graduate student has been asked to measure the pairwise distances among a group of objects in a metric space. He doesn't complete the experiment, and he must figure out the remaining numbers before his adviser returns from her conference. Obviously, all the distances need to be consistent, but the student does not know very much about the space in which the objects are embedded. One way to solve his problem is to find the "nearest" complete set of distances that satisfy the triangle inequalities. This procedure respects the measurements that have already been taken while forcing the missing numbers to behave like distances.

More charitably, suppose that the student has finished the experiment, but—measurements being what they are—the numbers do not satisfy the triangle inequality. The student knows that they must represent distances, so he would like to massage the data so that it corresponds with his *a priori* knowledge. Once again, the solution seems to require the "nearest" set of distances that satisfy the triangle inequalities.

Matrix nearness problems [6] offer a natural framework for developing this idea. If there are n points, we may collect the measurements into an $n \times n$ symmetric matrix whose (j, k) entry represents the dissimilarity between the j-th and k-th points. Then, we seek to

approximate this matrix by another whose entries satisfy the triangle inequalities. That is, $m_{ik} \leq m_{ij} + m_{jk}$ for every triple (i, j, k). Any such matrix will represent the distances among n points in some metric space. We measure the approximation error with a distortion measure that depends on how the corrected matrix should relate to the input matrix. For example, one might prefer to change a few entries significantly or to change all the entries a little.

We call the problem of approximating general dissimilarity data by metric data the *Metric Nearness (MN) Problem*. This simply stated problem has never been studied, although the literature does contain some related topics (see Section 1.1). This paper contains two major contributions. First, we present a rigorous statement of the Metric Nearness Problem (Section 2), and we argue that every locally optimal solution is globally optimal. Second, we present algorithms that take advantage of the structure of the problem to produce globally optimal solutions. It is computationally prohibitive, both in time and storage, to solve the MN problem without making these efficiencies.

1.1 Related Work

The Metric Nearness (MN) problem is novel, but the literature contains some related work.

The most relevant research appears in a recent paper of Roth et al. [11]. They observe that machine learning applications often require metric data, and they propose a technique for metrizing dissimilarity data. Their method, constant-shift embedding, increases all the dissimilarities by an equal amount to produce a set of Euclidean distances (i.e., a set of numbers that can be realized as the pairwise distances among an ensemble of points in a Euclidean space). The size of the translation depends on the data, so the relative and absolute changes to the dissimilarity values can be huge. Our approach to metrizing data is completely different. We seek a consistent set of distances that *deviates as little as possible* from the original measurements. Using our approach the resulting set of distances can arise from an arbitrary metric space. There is no restriction on obtaining Euclidean distances. In consequence, we expect metric nearness to provide superior denoising. Moreover, our techniques can also learn distances that are missing entirely.

There is at least one other method for inferring a metric. An article of Xing et al. [12] proposes a technique for learning a Mahalanobis distance for data in $\mathbb{R}^s$. That is, a metric $\mathrm{d}(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{(\boldsymbol{x} - \boldsymbol{y})^T \boldsymbol{G} (\boldsymbol{x} - \boldsymbol{y})}$, where $\boldsymbol{G}$ is an $s \times s$ positive semi-definite matrix. The user specifies that various pairs of points are similar or dissimilar. Then the matrix $\boldsymbol{G}$ is computed by minimizing the total *squared* distances between similar points while forcing the total distances between dissimilar points to exceed one. The article provides explicit algorithms for the case where $\boldsymbol{G}$ is diagonal and when $\boldsymbol{G}$ is positive semi-definite. In comparison, the metric nearness problem is not restricted to Mahalanobis distances; it can learn a general discrete metric. It also allows us to use specific distance measurements and to indicate our confidence in those measurements (by means of a weight matrix), rather than forcing a binary choice of "similar" or "dissimilar."

The Metric Nearness Problem may appear similar to metric Multi-Dimensional Scaling (MDS) [8], but we emphasize that the two problems are *distinct*. The MDS problem endeavors to find an ensemble of points in a *prescribed* metric space (usually a Euclidean space) such that the inter-point distances of these points are close to the set of input distances. In contrast, the MN problem does not seek to find any such embedding. In fact MN does not impose any hypotheses on the underlying space other than requiring it to be a metric space.

Section 2 formally describes the MN problem. In Section 3, we present algorithms that allow us to solve MN problems with ℓ_p nearness measures. Some applications and experimental results follow in Section 4. Section 5 discusses our results, some interesting

connections, and possibilities for future research.

2 The Metric Nearness Problem

Let us begin with some basic definitions. A *dissimilarity matrix* is a nonnegative, symmetric matrix with a zero diagonal. Meanwhile, a *distance matrix* is a dissimilarity matrix whose entries satisfy the triangle inequalities. That is, $\boldsymbol{M}$ is a distance matrix if and only if $m_{ik} \leq m_{ij} + m_{jk}$ for every triple of distinct indices (i, j, k). Distance matrices arise from measuring the distances among n points in a pseudo-metric space (i.e., two distinct points can lie at zero distance). A distance matrix contains $N = n\,(n-1)/2$ free parameters, so we denote the collection of all distance matrices by $\mathscr{M}_N$. The set $\mathscr{M}_N$ is a closed, convex cone.

The metric nearness problem requests a distance matrix $\boldsymbol{M}$ that is closest to a given dissimilarity matrix $\boldsymbol{D}$ with respect to some measure of divergence. Specifically, we seek a distance matrix $\boldsymbol{M}$ so that,

$$\boldsymbol{M} \in \left\{ \operatorname*{argmin}_{\boldsymbol{X} \in \mathscr{M}_N} \left\| \boldsymbol{W} \odot (\boldsymbol{X} - \boldsymbol{D}) \right\| \right\}, \tag{2.1}$$

where $\boldsymbol{W}$ is a symmetric non-negative weight matrix and '$\odot$' denotes the element-wise (Hadamard) product of two matrices. The weight matrix reflects our confidence in the entries of $\boldsymbol{D}$. When each d_{ij} represents a measurement with variance σ_{ij}^2, we might set $w_{ij} = 1/\sigma_{ij}^2$. If an entry of $\boldsymbol{D}$ is missing, one sets the corresponding weight to zero. For ease of exposition, we shall generally assume that all weights equal one.

Theorem 2.1. *The function $\left\| \boldsymbol{W} \odot (\boldsymbol{X} - \boldsymbol{D}) \right\|$ always attains its minimum on $\mathscr{M}_n$. Moreover, every local minimum is a global minimum. If, in addition, the norm is strictly convex and the weight matrix has no zeros or infinities off its diagonal, then there is a unique global minimum.*

Proof. The basic idea is that the objective function has no directions of recession, so it must attain a finite minimum. Details appear on pages 3 and 4 of [4]. □

It is possible to use any norm or divergence in the metric nearness problem. We restrict our attention to the ℓ_p norms, which we denote $\|\cdot\|_p$. The associated Metric Nearness Problems are

$$\min_{\boldsymbol{X} \in \mathscr{M}_N} \left[\sum_{j \neq k} \left| w_{jk}\,(x_{jk} - d_{jk}) \right|^p \right]^{1/p} \qquad \text{for } 1 \leq p < \infty, \text{ and} \tag{2.2}$$

$$\min_{\boldsymbol{X} \in \mathscr{M}_N} \max_{j \neq k} \left| w_{jk}\,(x_{jk} - d_{jk}) \right| \qquad \text{for } p = \infty. \tag{2.3}$$

Note that the ℓ_p norms are strictly convex for $1 < p < \infty$, and therefore the solution to (2.2) is unique. There is a basic intuition for choosing p. The ℓ_1 norm gives the absolute sum of the (weighted) changes to the input matrix, while the ℓ_∞ only reflects the maximum absolute change. The other ℓ_p norms interpolate between these extremes. Therefore, a small value of p typically results in a solution that makes a few large changes to the original data, while a large value of p typically yields a solution with many small changes.

3 Algorithms

This section describes efficient algorithms for solving the Metric Nearness Problems (2.2) and (2.3). At first, it may appear the one should use linear programming (LP) software when $p = 1, \infty$ and convex programming software for the remaining p. It turns out that the time and storage requirements of this approach are prohibitive. An efficient algorithm

must exploit the structure of the triangle inequalities. One approach is to use a *triangle-fixing algorithm*. This method examines each triple of points in turn, and it optimally enforces any triangle inequality that fails. (The definition of "optimal" depends on the ℓ_p nearness measure.) By introducing appropriate corrections, we can ensure that this iterative algorithm converges to a globally optimal solution of MN.

Notation We must introduce some additional notation before proceeding. To each matrix $\boldsymbol{X}$ of dissimilarities or distances, we associate the vector $\boldsymbol{x}$ formed by stacking the columns of the lower triangle, left to right. We use x_{ij} to refer to the (i, j) entry of the matrix as well as the corresponding component of the vector. Define a constraint matrix $\boldsymbol{A}$ so that $\boldsymbol{M}$ is a distance matrix if and only if $\boldsymbol{A}\boldsymbol{m} \leq \boldsymbol{0}$. Note that each row of $\boldsymbol{A}$ contains three nonzero entries, +1, -1, and -1.

3.1 MN for the ℓ_2 norm

We first develop an algorithm for solving (2.2) with respect to the ℓ_2 norm. This case turns out to be the simplest and most illuminating case. It also plays a pivotal role in the algorithms for the ℓ_1 and ℓ_∞ MN problems.

Given a dissimilarity vector $\boldsymbol{d}$, we wish to find its orthogonal projection $\boldsymbol{m}$ onto the cone $\mathscr{M}_N$. Let us introduce an auxiliary variable $\boldsymbol{e} = \boldsymbol{m} - \boldsymbol{d}$ that represents the changes to the original distances. We also define $\boldsymbol{b} = -\boldsymbol{A}\boldsymbol{d}$. The entries of $\boldsymbol{b}$ indicate how much each triangle inequality is violated. The problem becomes

$$\begin{aligned} &\text{minimize } \|\boldsymbol{e}\|_2, \\ &\text{subject to } \boldsymbol{A}\boldsymbol{e} \leq \boldsymbol{b}. \end{aligned} \tag{3.1}$$

From the minimizer $\boldsymbol{e}^\star$, we use the equation $\boldsymbol{m}^\star = \boldsymbol{d} + \boldsymbol{e}^\star$ to recover the optimal distance vector.

Here is our approach. We initialize the vector of changes to zero ($\boldsymbol{e} = \boldsymbol{0}$), and then we begin to cycle through the triangles. Suppose that the (i, j, k) triangle inequality is violated, i.e., $e_{ij} - e_{jk} - e_{ki} > b_{ijk}$. We wish to remedy this violation by making an ℓ_2-minimal adjustment of e_{ij}, e_{jk}, and e_{ki}. In other words, the vector $\boldsymbol{e}$ is projected orthogonally onto the constraint set $\{\boldsymbol{x} : x_{ij} - x_{jk} - x_{ki} \leq b_{ijk}\}$. This is tantamount to solving

$$\begin{aligned} &\min \ \tfrac{1}{2}\big[(x_{ij} - e_{ij})^2 + (x_{jk} - e_{jk})^2 + (x_{ki} - e_{ki})^2)\big], \\ &\text{subject to} \quad x_{ij} - x_{jk} - x_{ki} = b. \end{aligned} \tag{3.2}$$

It is easy to check that the solution is given by

$$x_{ij} \leftarrow e_{ij} + \mu, \qquad x_{jk} \leftarrow e_{jk} - \mu, \qquad \text{and} \qquad x_{ki} \leftarrow e_{ki} - \mu, \tag{3.3}$$

where $\mu = -\frac{1}{3}(b - e_{ij} + e_{jk} + e_{ki})$. Only three components of the vector $\boldsymbol{e}$ need to be updated, and their values are given by the calculation in (3.3).

In turn, we fix each violated triangle inequality using (3.3). We must also introduce a correction term to guide the algorithm to the global minimum. The corrections have a simple interpretation in terms of the dual of the minimization problem (3.1). Each dual variable corresponds to the violation in a single triangle inequality, and each individual correction results in a decrease in the violation. We continue until no triangle receives a significant update.

Algorithm 3.1 displays the complete iterative scheme that performs triangle fixing along with appropriate corrections.

ALGORITHM 3.1: Triangle Fixing For ℓ_2 norm.

TRIANGLE_FIXING($\boldsymbol{D}, \epsilon$)
Input: $\boldsymbol{D}$: Input dissimilarity matrix, ϵ tolerance
Output: $\boldsymbol{M} = \operatorname{argmin}_{\boldsymbol{X} \in \mathscr{M}_N} \|\boldsymbol{X} - \boldsymbol{D}\|_2$.
for $1 \leq i < j < k \leq n$
  $z_{ijk} \leftarrow 0$ {Initialize correction terms for each triple (i, j, k)}
for $1 \leq i < j \leq n$
  $e_{ij} \leftarrow 0$ {Initial error values for each dissimilarity d_{ij}}
$\delta \leftarrow 1 + \epsilon$ {Parameter for testing convergence}
while $(\delta > \epsilon)$ {convergence test}
  foreach violated triangle (i, j, k)
    $b \leftarrow d_{ki} + d_{jk} - d_{ij}$
    $\mu \leftarrow -\frac{1}{3}(b - e_{ij} + e_{jk} + e_{ki})$ $(\star)$
    $\theta \leftarrow \min\{\mu, z_{ijk}\}$ {Stay within half-space of constraint}
    $e_{ij} \leftarrow e_{ij} + \theta$, $e_{jk} \leftarrow e_{jk} - \theta$, $e_{ki} \leftarrow e_{ki} - \theta$ $(\star\star)$
    $z_{ijk} \leftarrow z_{ijk} - \theta$ {Update correction term}
  end *foreach*
  $\delta \leftarrow$ sum of changes in the $\boldsymbol{e}$ values
end *while*
return $\boldsymbol{M} = \boldsymbol{D} + \boldsymbol{E}$

Remark: Algorithm 3.1 is an efficient version of Bregman's algorithm [1] that exploits the structure of the problem. By itself, Bregman's method would suffer the same storage and computation costs as a general convex optimization algorithm. Our triangle fixing operations allow us to compactly represent and compute the intermediate variables required to solve the problem. The correctness and convergence properties of Algorithm 3.1 follow from those of Bregman's method.

3.2 MN for the ℓ_1 and ℓ_∞ norms

The basic triangle fixing algorithm succeeds only when the norm used in (2.2) is strictly convex. Hence, it cannot be applied directly to the ℓ_1 and ℓ_∞ cases. These require a more sophisticated approach.

First, observe that the problem of minimizing the ℓ_1 norm of the changes can be written as an LP:

$$\begin{aligned} &\min_{\boldsymbol{e},\boldsymbol{z}} \ \mathbf{1}^T \boldsymbol{z} \\ &\text{subject to } \boldsymbol{A}\boldsymbol{e} \leq -\boldsymbol{A}\boldsymbol{d},\ -\boldsymbol{e} - \boldsymbol{z} \leq \mathbf{0},\ \boldsymbol{e} - \boldsymbol{z} \leq \mathbf{0}. \end{aligned} \tag{3.4}$$

The auxiliary variable $\boldsymbol{z}$ can be interpreted as the absolute value of $\boldsymbol{e}$. Similarly, minimizing the ℓ_∞ norm of the changes can be accomplished with the LP

$$\begin{aligned} &\min_{\boldsymbol{e},\zeta} \ \zeta \\ &\text{subject to } \boldsymbol{A}\boldsymbol{e} \leq -\boldsymbol{A}\boldsymbol{d},\ -\boldsymbol{e} - \zeta\mathbf{1} \leq \mathbf{0},\ \boldsymbol{e} - \zeta\mathbf{1} \leq \mathbf{0}. \end{aligned} \tag{3.5}$$

We interpret $\zeta = \|\boldsymbol{e}\|_\infty$.

Solving these linear programs using standard software is prohibitively expensive because of the large number of constraints. Moreover, the solutions are not unique because the ℓ_1 and ℓ_∞ norms are not strictly convex. Instead, we replace the LP by a quadratic program (QP) that is strictly convex and returns the solution of the LP that has minimum ℓ_2-norm. For the ℓ_1 case, we have the following result.

Theorem 3.1 (ℓ_1 Metric Nearness). *There exists a constant $\lambda > 0$ such that*

$$\underset{\boldsymbol{z} \in Z}{\operatorname{argmin}} \ \|\lambda \mathbf{1} + \boldsymbol{z}\|_2 \quad = \quad \underset{\boldsymbol{z} \in Z^\star}{\operatorname{argmin}} \ \|\boldsymbol{z}\|_2, \tag{3.6}$$

where Z is the feasible set for (3.4) *and $Z^\star$ is the set of optimal solutions to* (3.4)*. The minimizer is unique.*

Theorem 3.1 follows from an application of a result of Mangasarian [9, Theorem 2.1-a-i]. A similar theorem may be stated for the ℓ_∞ case.

The QP (3.6) can be solved using an augmented triangle-fixing algorithm since the majority of the constraints in (3.6) are triangle inequalities. As in the ℓ_2 case, the triangle constraints are enforced using (3.3). Each remaining constraint can be enforced by computing an orthogonal projection. We refer the reader to [5] for the details.

3.3 MN for ℓ_p norms ($1 < p < \infty$)

Next, we explain how to use triangle fixing to solve the MN problem for the remaining p in $(1, \infty)$. The computational costs are somewhat higher because the algorithm requires solving a nonlinear equation. The problem may be phrased as

$$\text{minimize } \frac{1}{p} \|\boldsymbol{e}\|_p^p \qquad \text{subject to} \qquad \boldsymbol{A}\boldsymbol{e} \leq \boldsymbol{b}. \tag{3.7}$$

To enforce a triangle constraint optimally in the ℓ_p norm, we need to compute a "non-orthogonal projection" of the vector $\boldsymbol{e}$ onto the constraint set. Define $\varphi(\boldsymbol{x}) = \frac{1}{p} \|\boldsymbol{x}\|_p^p$, and note that $\nabla\varphi(\boldsymbol{x}) = \operatorname{sgn}(\boldsymbol{x}) \, |\boldsymbol{x}|^{p-1}$. The φ-projection of $\boldsymbol{e}$ onto the (i, j, k) constraint is the solution of

$$\min \ \varphi(\boldsymbol{x} - \boldsymbol{e}) \qquad \text{subject to} \qquad x_{ij} - x_{jk} - x_{ki} = b.$$

According to [1, Lemma 2.2.1], the projection may be determined by solving

$$\nabla\varphi(\boldsymbol{x}) = \nabla\varphi(\boldsymbol{e}) + \mu \, \boldsymbol{a}_{ijk} \qquad \text{so that} \qquad {\boldsymbol{a}_{ijk}}^T \boldsymbol{x} = b, \tag{3.8}$$

where $\boldsymbol{a}_{ijk}$ is the row of the constraint matrix corresponding to the triangle inequality (i, j, k). Since $\boldsymbol{a}_{ijk}$ has only three nonzero entries, we see that $\boldsymbol{e}$ only needs to be updated in three components. Therefore, in Algorithm 3.1 we may replace $(\star)$ by an appropriate numerical computation of the parameter μ and replace $(\star\star)$ by the computation of $\boldsymbol{x}$. A detailed derivation along with an implementation of Algorithm 3.1 for the ℓ_p norms is given in [5].

4 Applications and Experiments

Replacing a general graph (dissimilarity matrix) by a metric graph (distance matrix) can enable us to use efficient approximation algorithms for NP-Hard graph problems (MAX-CUT clustering) that have guaranteed error for metric data, for example, see [7]. The error from MN will carry over to the graph problem, while retaining the bounds on total error incurred. As an example, constant factor approximation algorithms for MAX-CUT exist for metric graphs, see [3], and can be used for clustering applications. See [4] for more details.

Applications that use dissimilarity values, such as clustering, classification, searching and indexing, to name a few, could potentially be sped up if the data is metric. MN is a natural candidate for enforcing metric properties on the data to enable these speedups.

We were originally motivated to formulate and solve MN by a problem that arose in connection with biological databases. This problem involves approximating mPAM matrices,

Table 1: Relative errors for mPAM dataset

Dataset	ℓ_1 error	ℓ_2 error	ℓ_∞ error
mPAM50	0.339	0.402	0.278
mPAM100	0.142	0.231	0.206
mPAM150	0.054	0.121	0.151
mPAM250	0.004	0.025	0.042
mPAM300	0.002	0.017	0.056

which are a derivative of mutation probability matrices [2] that arise in protein sequencing. They represent a certain measure of dissimilarity for an application in protein sequencing. Owing to the manner in which these matrices are formed, they tend not to be distance matrices. In turns out that query operations in biological databases have the potential to be dramatically sped up if the data were metric (using a metric based indexing scheme). Thus, one approach is to find the nearest distance matrix to each mPAM matrix and use that approximation in the metric based indexing scheme.

We approximated various mPAM matrices by their nearest distance matrices. The relative errors of the approximations $\|\boldsymbol{D} - \boldsymbol{M}\|/\|\boldsymbol{D}\|$, are reported in Table 1. We notice that the relative error, as incurred by each of the solutions ($p = 1, 2, \infty$), is comparable (for more detailed results, see http://www.cs.utexas.edu/users/suvrit/work/metric).

4.1 Experiments

The MN problem has an input of size N, and the number of constraints is roughly $N^{3/2}$. We ran experiments to ascertain the empirical behavior of the algorithm. Figure 1 shows

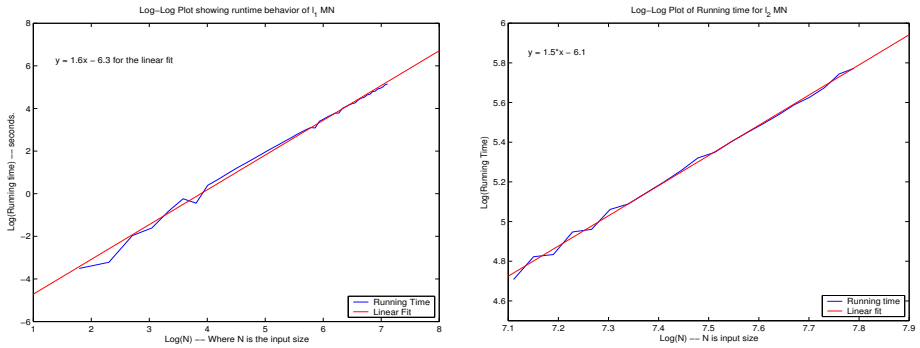

Figure 1: Running time for ℓ_1 and ℓ_2 norm solutions (plots have different scales).

log–log plots of the running time of our algorithms for solving the ℓ_1 and ℓ_2 Metric Nearness Problems. The time cost appears to be $O(N^{3/2})$, which is *linear* in the number of constraints. The results plotted in the figure were obtained by executing the algorithms on random dissimilarity matrices. The procedure was halted when the distance values changed less than 10^{-3} from one iteration to the next. For both problems, the results were obtained with a simple MATLAB implementation. Nevertheless, this basic version outperforms MATLAB's optimization package by one or two orders of magnitude (depending on the problem). A more sophisticated (C or parallel) implementation could improve the running time even more, which would allow us to study larger problems.

5 Discussion

In this paper, we have introduced the Metric Nearness problem, and we have developed algorithms for solving it for ℓ_p nearness measures. The algorithms proceed by fixing violated triangles in turn, while introducing correction terms to guide the algorithm to the global optimum. Our experiments suggest that the algorithms require $O(N^{3/2})$ time, where N is the total number of distances. An immediate question is whether it is possible to obtain faster algorithms. The answer is negative because it requires $\Omega(N^{3/2})$ time just to check that N distance measurements satisfy the triangle inequality.

MN is a rich problem. Indeed, it can be shown that a special case (allowing only decreases in the dissimilarities) is *identical* to the All Pairs Shortest Path problem [10].

It is also possible to incorporate other types of constraints into the Metric Nearness Problem. Some other possibilities include putting box constraints on the distances: $\boldsymbol{l} \leq \boldsymbol{m} \leq \boldsymbol{u}$, allowing λ triangle inequalities: $m_{ij} \leq \lambda_1 m_{ik} + \lambda_2 m_{kj}$, or enforcing order constraints: $d_{ij} < d_{kl}$ implies $m_{ij} < m_{kl}$.

We plan to further investigate the application of MN to other problems in data mining, machine learning and database query retrieval.

References

[1] Y. Censor and S. A. Zenios. *Parallel Optimization: Theory, Algorithms, and Applications*. Numerical Mathematics and Scientific Computation. Oxford University Press, 1997.

[2] M. O. Dayhoff, R. M. Schwarz, and B. C. Orcutt. A model of evolutionary change in proteins. *Atlas of Protein Sequence and Structure*, 5(Suppl. 3), 1978.

[3] W. F. de la Vega and C. Kenyon. A randomized approximation scheme for Metric MAX-CUT. *J. Comput. Sys. and Sci.*, 63:531–541, 2001.

[4] I. S. Dhillon, S. Sra, and J. A. Tropp. The Metric Nearness Problems with Applications. Technical Report TR-03-23, Computer Sciences, University of Texas at Austin, 2003.

[5] I. S. Dhillon, S. Sra, and J. A. Tropp. Triangle Fixing Algorithms for the Metric Nearness Problem. Technical Report TR-04-22, Computer Sciences, University of Texas at Austin, 2004.

[6] N. J. Higham. Matrix nearness problems and applications. In M. J. C. Gower and S. Barnett, editors, *Applications of Matrix Theory*, pages 1–27. Oxford University Press, 1989.

[7] P. Indyk. Sublinear time algorithms for metric space problems. In *31st Symposium on Theory of Computing*, pages 428–434, 1999.

[8] J. B. Kruskal and M. Wish. *Multidimensional Scaling*. Number 07-011. Sage Publications, 1978. Series: Quantitative Applications in the Social Sciences.

[9] O. L. Mangasarian. Normal solutions of linear programs. *Mathematical Programming Study*, 22:206–216, 1984.

[10] C. G. Plaxton and A. Clement. Personal Communication, 2003–2004.

[11] V. Roth, J. Laub, J. M. Buhmann, and K.-R. Müller. Going metric: Denoising pariwise data. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems (NIPS) 15*, 2003.

[12] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell. Distance metric learning, with application to clustering with side constraints. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems (NIPS) 15*, 2003.