Learning a Distance Metric to cluster E-Mails

Mitul Saha
Computer Science Deptt.
Stanford University.

Gauhar Wadhera
Computer Science Deptt.
Stanford University.

ABSTRACT:

Application of document clustering techniques to cluster e-mails is an interesting application. Techniques like k-means, EM etc can be used to achieve this. However, the selection of a good distance metric is the key issue involved. Often people manually tweak the chosen distance metric to achieve desirable/good clusters/results that in all certainty do not provide a generic solution. Hence it would be very useful to automatically learn the distance metric from some training set before clustering. In [1] a technique for learning distance metrics has been proposed for clustering. Our first task is to apply this technique to document (specifically e-mails) clustering.

The main issues covered in our work are two-fold: to decide on the learning strategy to be adopted and secondly, to design a feature space for the emails. Classical email related problems as in [2] have sought to classify them based on their text usually resulting in a space either as big as the collection vocabulary or proportional to the document length. Even existing clustering mechanisms like k-Means have utilized the term-document space for emails in order to cluster. We approach the problem in a way as to avoid the dimensionality problem as well.

I. INTRODUCTION:

Most work in text processing can easily be broadly categorized into two areas: clustering and classification.

Text Classification methods have majorly used classifiers as Naïve Bayes and SVMs (for example, [2]) in order to classify text as {spam/non-spam}, {newsgroup/ non-newsgroup} etc. These classifiers have mostly worked in the term space representation for documents relying primarily on the occurrence/non-occurrence/frequency of terms in the documents to establish their classification. The term space is defined by terms in the collection vocabulary serving as the dimensions of the input space with various measures as tf*idf (term frequency, inverse document frequency) values filling in the magnitudes. The literature for this is available in [3]. Even though such supervised learning methods optimize the classification error the metrics learnt are not necessarily generic enough to be plugged into other learning algorithms as K-Means. This particular limitation is easily highlighted in the case of less structured data.

Text clustering techniques have principally utilized the same representation Even though such representations have served a lot of applications in text clustering and information retrieval they inherently suffer from the drawback of high dimensionality. There exist methods as Principal Component Analysis [4] and Multi-Dimensional Scaling [5] that find an optimal embedding of the inputs in a subspace (not necessarily strict) of the input feature space. However, despite providing a legitimate and often useful workaround to the dimensionality problem they seem to lack generalization over non-training data.

Our effort mainly concentrates on the concept of learning a distance metric as proposed by Xing et al. [1] with the training data in the form of pairs of points labeled as similar or dissimilar giving the objective function a criteria for optimization to put similar pairs in the same cluster while assigning dissimilar points to different ones. The above methods strictly work on the concept of similarity and dissimilarity as a binary outcome with no notion of degrees of similarity and dissimilarity and

shall be an aspect to be researched by us beyond the scope of this term project.
Our project deals specifically with the task of clustering emails into folders wherein the above algorithms would map onto being given a training set of similar and dissimilar emails. We strongly feel that the semantics of emails are much richer than simple unstructured text documents, which can be capitalized upon in the design of our feature space. As a preliminary research step we had run K-Means clustering on a set of emails with each emails represented in the term space with its tf*idf values. The resulting clusters were compared to the original email folders that formed the data set and the level of matching was not very significant. The clusters were a clear indication that the term-document space of operations was not completely successful in capturing the email semantics in order to learn a notion of similarity/dissimilarity. The above results form a part of the result comparisons in the Results section.

Besides deciding on appropriate learning strategies and evaluating them, the second phase of our work concentrates on designing, testing and evaluating various features for representing the emails as "Sender" field, "Bcc", "Subject" etc. We hope to learn the significance as well as correlation between all these features via a distance metric learnt through the methods described in [1].

II. PROBLEM DEFINITION:

The primary objective of the project is to build a module for an office agent that clusters emails into folders. The primary question that arises is whether an agent can do the above since such a clustering (as any other clustering operation) requires a metric of similarity/dissimilarity as the centric operator. However, there is no gold standard available in such a scenario. In fact, each user decides his own gold standard; in other words a user himself decides as to what he wants to be considered as similar/dissimilar. For example, I might want my emails to be foldered based on the month of receipt whereas someone else would want the clustering to be done based on subject and content. List emails provide another such example.
As input the agent shall receive emails of its client, along with a set of examples, both similar and dissimilar, in order to develop a sense of what the user requires in his clustering. The agent has the task to group the emails into folders based on the similarity between them in the form of common subjects, common email threads etc. Our goal is to design a learning strategy for the agent to learn how to cluster. The input vectors are represented in the feature space with training data comprising of the two sets:

i) S: $(x_i,x_j) \in S$ if x_i,x_j are similar
ii) D: $(x_i,x_j) \in D$ if x_i,x_j are dissimilar

Learning methods are then applied to the training data in order to learn a distance metric. Our notion of a distance metric has been derived from [1].

The key concept in any clustering method is the notion of the distance between two points in the sample space. The distance metric is used to establish the concept of similarity and dissimilarity between any two points. Intuitively, similar points are closer to each other than a pair of dissimilar points. However, the issue still remains as to how do we formally learn a distance metric. Herein, we could define our objectives that we expect the distance metric to achieve:

i) Learn the importance of individual features in the input vectors. This could be easily be done by learning a relative weighting/importance given to each feature.
ii) Learn the correlation between features if there exists any.

[1] discusses the concept of a distance metric A as:

$d(x,y) = d_A(x,y)$ which can be expressed as:
$\| x - y \|_A = [(x-y)^T A(x-y)]^{1/2}$

Here A is clearly of the form $\Re^{n \times n}$. Certain Properties that A needs to satisfy are non-negativity and the triangle inequality which require A to be positive semi-definite. It

can be seen that in order to satisfy the first objective a diagonal matrix A would suffice since it would assign different weight to the different axes. In order to learn various correspondences and correlations in the feature space we learn the full matrix A.

The next three sections discuss the three stages of development of our work:

i) Collection and parsing of data.
ii) The learning strategy used for learning the distance metric.
iii) Design of a feature space for the email documents.

III. DATA COLLECTION AND PARSING:

The data used for this project comes from the CALO initiative (URL: http://www.calo.sri.com) in the form or collections of emails of people. We gratefully acknowledge Ben Taskar for helping us out with the data accumulation. This serves us well in two contexts, giving us ample data to learn/test on as well as possible labeling since they are organized in folders for each person. The folder organization for each person has been used to test the accuracy of our system.

Even though availability of data has been a non-issue, cleanliness has definitely been a major hurdle. Even though email headers impart a lot of structure and semantics to emails, even the MIME headers do not have all standard features and only all-present headers (e.g. sent to:, cc:, bcc:) have been used in order to impart scalability to the system. However, it is not the headers that pose the biggest challenge.

Email text does not possess any standard structure and is not even restricted to the English grammar. Text features as punctuations, emoticons etc. occur as frequently as normal words in emails. To our credit, the above have been converted from a hurdle in standardization to assets in context detection by indexing them along with other words in the mails. However, numerous other problems do exist. For example, HTML in emails (the html body parts in multi-part MIME messages) has been avoided and the ASCII portions used for analysis since the tags tend to dominate the top ranking tokens of the mails.
Even though we have handled most issues the hampered our progress in training, we cannot claim complete success in the objective. However, we do not see a solution apart from having a knowledge base of patterns (e.g. HTML tags), which are consequently excluded by the agent from its parsing and training schedule.

IV. LEARNING STRATEGY:

Our learning methodologies are primarily derived from the work of Xing et al. [1]. The representation of a distance metric is as described in section II. We learn the distance metric in 2 ways, which shall be analyzed and compared as part of the result analysis of our work. These 2 strategies are as:

i) Learning a diagonal A:

Xing et al. [1] have defined the following objective function in order to learn a diagonal A = diag(A_{11}, A_{22}, …., A_{nn}).

$$g(A) = \sum_{S} ||x_i - x_j||_A^2 - log \sum_{D} ||x_i - x_j||_A$$

where,
S and D retain the same specifications i.e. set of similar and dissimilar points respectively.

This is done by performing a Newton's search over the search space. The optima are found by minimizing the above objective function given the constraint that A remains positive semi definite. The above constraint is imposed by performing a line search before the update step in the optimization to obtain the step size (in addition to the learning rate) alpha that ensures A remaining p.s.d. The log function over the dissimilar set ensures that the summation does not below zero since this is a minimization.

The derivations for the gradient and Hessian equations used in our learning code have been provided as Appendix A.

ii) Learning a full matrix A:

The learning of the full matrix happens in three stages:

- Maximization of cumulative distance between the dissimilar points. This is achieved by using a gradient ascent on the following objective.

$$\max_{A} g(A) = \sum_{(x_i,x_j)\in D} ||x_i - x_j||_A$$

- The above optimum is then projected using the concept of iterative projection [6] onto the set of matrices satisfying the following constraint on the set of similar points. This is done by via the following constrained optimization.

$$A' = arg\min_{A'} h(A') = arg\min_{A'} ||A' - A||_F^2$$

$$s.t. f(A) = \sum_{(x_i,x_j)\in S} ||x_i - x_j||_A^2 \leq 1$$

- The result is finally projected onto the set of all p.s.d matrices by removing all negative eigen values to derive the result.

The above steps are repeated in order to converge to the global optimum. The equations for the above optimizations are derived in Appendix B.

V. EMAIL FEATURE SPACE:

Selection of informative and distinguishing features is core and crucial to our end objectives. Below is a list of features that we have used in our feature space. These features do not represent the perfect scenario as shall be seen in our results and other possibilities could result in better performances. Since we did not want the features to have any initial bias based on the range of values each takes, we have implemented all the features (except distances between tf*idf vectors) on a scale of 0 (completely similar) to 1.0 (completely dissimilar).

i) Subject Line Vector:

The subject line usually conveys a lot about the context of e-mail. Common examples of the same include chains of emails arising from a discussion in a group having similar subject lines with 'Re:'/'Fwd:' tags added to it. We create an index of all subjects in the collection and create vectors in the term space for each subject line. The Euclidean distance between the tf*idf vectors of two subject lines gives the value of the distance between the two vectors in the subject dimension.

ii) Subject Line String:

We have a separate feature for the longest common sub string between two subject lines since it was observed that the high dimensional tf*idf vectors were not able to highlight the syntax of list emails. The [cs-229-qa] emails were easily missed since the '[' were getting excluded from the index. The sub string score substantially helped in capturing the list emails and the similarity between them.

iii) Sender's e-mail id:

It is highly probable that the context of the e-mails sent by a particular person would be similar. Exception to this is when we hold multiple relationships with a person. For example, if your secretary is also your girlfriend. However, we do not cover for such exceptions and expect the training process to downplay such inconclusive features. The distance between any two vectors is given as a function of the length of the longest common subsequence between the two email IDs.

Example,
LCS[{andrew_ng} – {ngandrew}] = {andrew}.

iv) Domain of the e-mail:

Useful information could also be hidden in the domain name of the sender's e-mail. For example working in Enron we could reasonably expect emails from 'enron.com' to be official emails. As another example, emails from '.com' domains might serve a different purpose as compared to mails

from '.edu' domains. Even though this might be a very weak signal we shall still be plugging this in as a feature. An LCS score is used to measure the distance in this dimension as well. This feature has been excluded from our test runs since the entire test set was from CALO and ENRON as a result of which all the distances (almost all) were zero in the feature resulting in a singular matrix while training.

v) CC:

If some e-mails have the same recipients in the CC components of the MIME headers then it is likely that they are related. We compute similarity based on the number of common recipients in the CC/BCC/TO header of the two emails.

vi) List Email Detection:

Certain identifying characteristics for list emails include different addresses in the "To" field and the client's email address. However, an exception to this is email redirection. The subject line heuristic was able to detect the list emails pretty comfortably and hence we have used that for list detection rather than relying on email addresses.

vii) Body:

We create an index of all email bodies in the collection. The Euclidean distance between the tf*idf vectors of two subject lines gives the value of the distance between the two vectors in the subject dimension. It can be clearly seen that even though this feature involves a very high dimensionality it does not involve training in these dimensions as a result of which it does not suffer from the problems of clustering in the term-document space. An interesting observation was that for almost all emails the highest tf*idf valued token was junk (e.g. _____00934031****). We introduced a threshold and excluded terms with a document frequency below it. For example, with the threshold 1 we did not include the terms that occurred only in 1 email. This intuitively makes sense for similarity (not so for dissimilarity) measurement. The above heuristic led to a big qualitative improvement in the performance of our feature space.

viii) Top 'N' Tokens in Body/Subject:

Even though this feature is similar to the Euclidean distance between the two Body/Subject vectors we keep it to ensure that actual token level similarities do not get under expressed due to the distance between two vectors being computed after converting them to unit vectors for length normalization (also known as the cosine distance between two vectors). The feature performs pretty well in our data sets. We have ranked the tokens in a Body based on the tf*idf values and see the number of common tokens in the top 'N' tokens in each Body/Subject to account for their similarities. The values used for 'N' in our space are 50, 20 and 10 for the email Body and 5 and 2 for the Subject.

ix) Date of receipt:

This feature is included in the space to allow the agent to be able to learn a metric for mails sorted by date as well. Even though we have not tested the learning by giving it synthetic data stressing on the date, the theory behind the learning inspires enough confidence in the positive results expected.

VI. IMPLEMENTATION:

i) Email Parsing:

The code for parsing the emails is in Java and uses the JMime library (http://www.hunnysoft.com) for extracting the MIME structure. The code for building the index/lexicon for the email collection containing the term frequencies and document frequencies uses Lucene (http://jakarta.apache.org) for the backend with the interfacing code in Java. The implementation for the parsing infrastructure include s a lot of text preprocessing and cleaning code in order to ensure that the actual signal in the emails is not diluted by unimportant material as HTML tags etc.

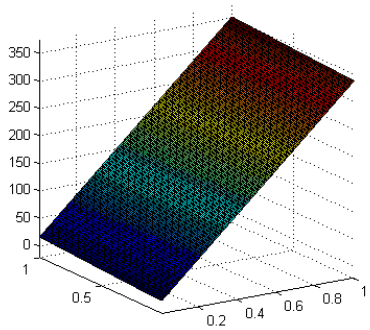


Fig 1a. Unscaled Search space with 2 parameters.

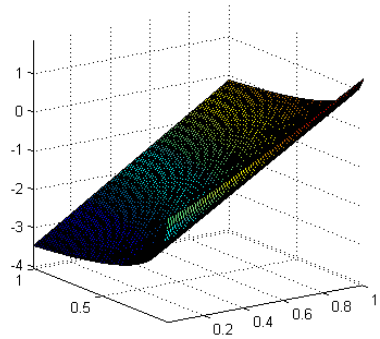


Fig1b Search space scaling factor C = 0.01.

ii) Learning Code:

- Diagonal Learning

The two methods of learning as described earlier and in [1] have been coded in MATLAB. We are deeply grateful for the constant help from Pieter Abbeel in advising us on fixing the nuances of the optimization functions.

The search space was almost a continuous slope for the objective function mentioned earlier with a very narrow optimum band. The solution was easily being missed by the Newton Raphson method. A contant factor C (between 0 and 1.0) was multiplied to the summation over similar points in order to make the search space more amenable to optimization. The result was only a scaled version of the original optimum found. An example created from near ideal synthetic data (easiest to optimize) with two features and hence two parameters (1 for each) to be learnt had a search space as Fig. 1a, which was transformed to Fig. 1b as a result of the scaling done.

- Full Matrix Learning

The code – as pseudo coded in Xing et al [1] had a considerable time overhead to its credit as a result of which its applicability in an office agent seemed doubtful. We now have (with help from Pieter Abbeel) an optimization code using the SeDuMi package, which runs extremely efficiently to learn the full matrix in seconds. The results are completely in tune with the iterative projection code written earlier.

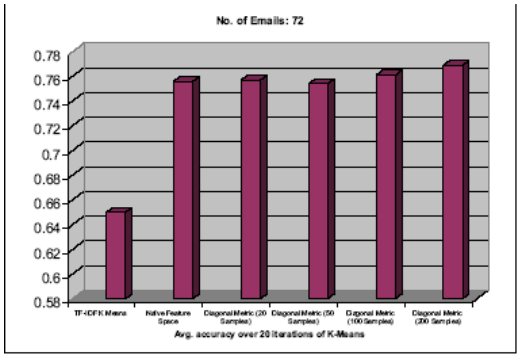
iii) Clustering Code:

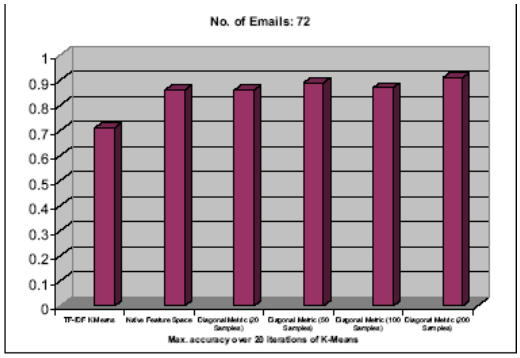
The clustering code is written in Java. We have implemented two principle clustering routines. The first one is for running kMeans on the tf*idf vectors for the emails in order to simulate and evaluate the fundamental method behind a majority of current text clustering techniques. The second routine evaluates distances based on the learnt distance metric as explained in [1] and in section III. Our feature space does not model each email as a vector of real numbers. Rather, it models the distance between two emails as $\chi = \Re^n$. Consequently, kMeans does not run normally since there is no such point as a centroid for each cluster. Hence, we locate "representative" points for each cluster C according to the objective function:

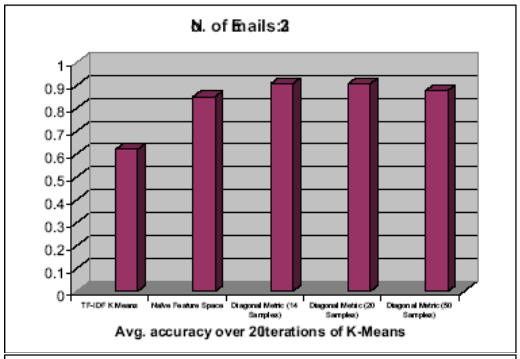
$P = \text{argmin}_{P \text{ in } C} (\Sigma \| X - P \|^2_A)$ where,

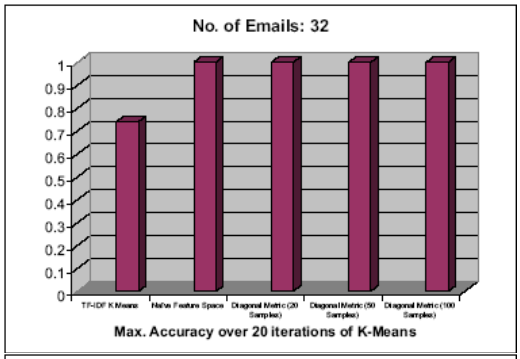
X is a point in cluster C.

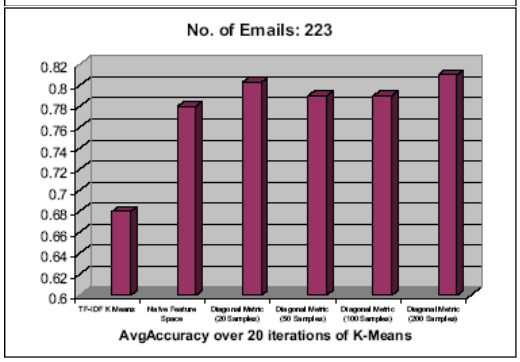
VII. Results and Conclusions:

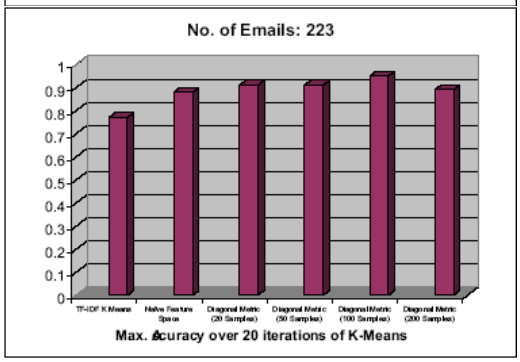


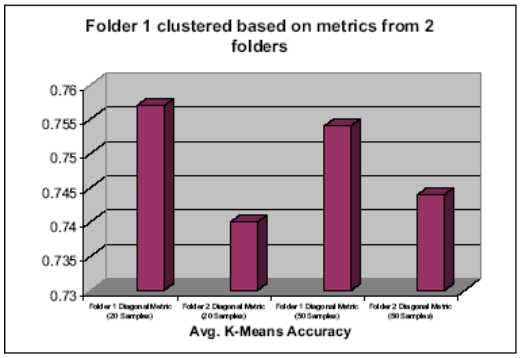


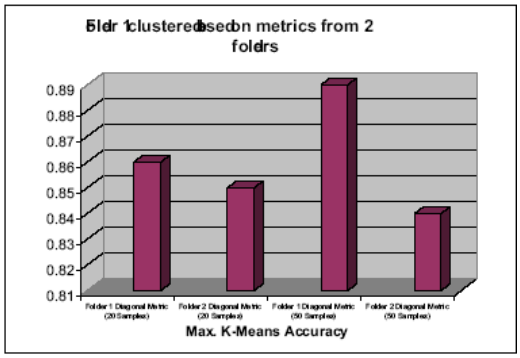












Our entire result analysis has been done by learning a metric over a given user's email folders and observing as to how well our clustering maps onto the original clustering. Testing has basically been on the basis of running K-Means multiple numbers of times over the folder to observe the average accuracy over the clustering. Maximum accuracies earned through the cycles have also been reported. However, these are of less significant value.

It was very clearly observable that the single biggest gain was being accrued due to the feature space representation adopted by us. Average accuracies were observed for learning the diagonal metric by giving the system 20, 50, 100 and 200 dissimilar/similar pairs. A consistent observation was the increase in accuracy from the normal term-document space clustering to the feature space. The other significant improvement was when we used 20 samples. As the number of samples was increased the average accuracy more or less stayed stagnant and in certain cases even dropped showing a sense of over fitting. Our philosophy of learning a given user's notion of similarity/dissimilarity was further substantiated by the observation that average accuracy notably reduced when we clustered a user's emails based on learning a metric for another user.

Even though constrained K-Means would have positively given us better results we have decided to opt out of it and give the points the freedom to organize themselves. It also reduces the dependence of the learning on the users training data as a gold standard.

The results obtained have been healthy and predictable. They encourage us to further delve into better training algorithms as well as newer features.

VIII. Future Work:

Even though the results clearly show a two-fold gain as compared to the conventional text clustering in the context of emails a lot of scope for improvement remains. We do believe that there is an urgent need to build a knowledge base with every agent to store patterns deemed unfit for parsing and learning on. This shall help counter most data anomalies.

On the feature space front, we intend to evaluate a new feature in the form of a "sliding window tf*idf distance" between two emails. This is based on the principle that any contextual similarity between two documents does not necessarily span the entire length. Relations based on text can be highlighted through regions of emails talking about the same subject. The feature shall traverse both emails and find regions of high similarity based on tf*idf values in both. The use of tf*idf does not imply its functionality as a gold standard and it would be interesting to evaluate other metrics as simple tf's etc as well.
On the learning front, Shai et al. [7] suggest a pseudo-metric online learning algorithm (POLA) that updates the distance metric incrementally based on new inputs about similar/dissimilar pairs of data points. This fits well with our ultimate objective since it can easily map onto an agent that incrementally learns and improvises on the clustering of emails for its client.

IX. Acknowledgements:

We would like to sincerely thank Pieter Abbeel, Ben Taskar and Andrew Ng for their constant help throughout the project on all fronts.

X. References:

[1] Distance metric learning, with application to clustering with side-information; Eric Xing, Andrew Y. Ng, Michael Jordan, and Stuart Russell. NIPS 15, 2003.

[2] A Bayesian approach to filtering junk email; M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz. AAAI Workshop on Learning for Text Categorization, July 1998, Madison, Wisconsin. AAAI Technical Report WS-98-05

[3] A Theory of Indexing; Salton, G. The Society for Industrial and Applied Mathematics.

[4] Principal Component Analysis; I.T. Jolliffe. Springer-Verlag, New York, 1989.

[5] Multidimensional Scaling; T. Cox and M. Cox. Chapman & Hall, London, 1994.

[6] Convex Analysis; R. Rockafellar. Princeton Univ. Press, 1970.

[7] Online and Batch Learning of Pseudo-Metrics, Shai et al.

1 Appendix A

The following is the derivation for the Gradient and Hessian to be used in the Newton Raphson optimzation for learning the Diagonal Matrix A. Substitutions for long expressions have been made to symbols as indicated in the derivation. The objective function g(A) is given as:

$$g(A) = \sum_S ||x_i - x_j||_A^2 - log \sum_D ||x_i - x_j||_A \tag{1}$$
$$= \alpha - \beta(say)$$

Taking partial derivatives:

$$\frac{\partial \alpha}{\partial a_k} = \sum_S (x_{ik} - x_{jk})^2 \tag{2}$$
$$\frac{\partial \beta}{\partial a_k} = \frac{1}{r}\frac{\partial r}{\partial a_k} \tag{3}$$
$$where, \tag{4}$$
$$r = \sum_D ||x_i - x_j||_A = \sum_D \sqrt{\sum_l a_l (x_{il} - x_{jl})^2} = \sum_D r_{ij} \tag{5}$$
$$\frac{\partial r}{\partial a_k} = \sum_D \frac{\partial r_{ij}}{\partial a_k} \tag{6}$$
$$where, \tag{7}$$
$$r_{ij} = \sqrt{\sum_l (x_{il} - x_{jl})^2} \tag{8}$$

$$\frac{\partial r_{ij}}{\partial a_k} = \frac{1}{2sqrt\tau_{ij}}\frac{\partial \tau_{ij}}{\partial a_k} \tag{9}$$
$$where, \tag{10}$$
$$\tau_{ij} = \sum_l a_l (x_{ij} - x_{jl})^2 \tag{11}$$
$$\frac{\partial \tau_{ij}}{\partial a_k} = (x_{ik} - x_{jk})^2 \tag{12}$$
$$\frac{\partial r_{ij}}{\partial a_k} = \frac{(x_{ik} - x_{jk})^2}{2\sqrt{\sum_l (x_{il} - x_{jl})^2}} \tag{13}$$
$$= \frac{(x_{ik} - x_{jk})^2}{2||x_i - x_j||_A} \tag{14}$$
$$\frac{\partial r}{\partial a_k} = \sum_D \frac{(x_{ik} - x_{jk})^2}{2||x_i - x_j||_A} \tag{15}$$

$$\frac{\partial \beta}{\partial a_k} = \frac{1}{\sum_D ||x_i - x_j||_A} \sum_D \frac{(x_{ij} - x_{jk})^2}{2||x_i - x_j||_A} \tag{16}$$

$$\frac{\partial g}{\partial a_k} = \frac{\partial \alpha}{\partial a_k} - \frac{\partial \beta}{\partial a_k} \tag{17}$$

$$= \sum_S (x_{ik} - x_{jk})^2 - \frac{1}{\sum_D ||x_i - x_j||_A} \sum_D \frac{(x_{ij} - x_{jk})^2}{2||x_i - x_j||_A} \tag{18}$$

$$\tag{19}$$

The above expression is used to obtain the gradient.
The following equations are useful in deriving the expression for the Hessian

$$\frac{\partial^2 g}{\partial a_k \partial a_u} = \frac{\partial^2 \alpha}{\partial a_k \partial a_u} - \frac{\partial^2 \beta}{\partial a_k \partial a_u} \tag{20}$$

$$\frac{\partial^2 \alpha}{\partial a_k \partial a_u} = 0 \tag{21}$$

$$\frac{\partial^2 \beta}{\partial a_k \partial a_u} = \phi \frac{\partial \psi}{\partial a_u} + \psi \frac{\partial \phi}{\partial a_u} \tag{22}$$

$$\phi = \frac{1}{\sum_D ||x_i - x_j||_A} = \frac{1}{r} \tag{23}$$

$$\frac{\partial \phi}{\partial a_u} = -\frac{1}{r^2} \frac{\partial r}{\partial a_u} \tag{24}$$

$$= \frac{1}{(\sum_D ||x_i - x_j||_A)^2} \sum_D \frac{(x_{iu} - x_{ju})^2}{2||x_i - x_j||_A} \tag{25}$$

$$\psi = \frac{\sum_D (x_{ik} - x_{jk})^2}{2||x_i - x_j||_A} = \sum_D \psi_{ij} \tag{26}$$

$$\psi_{ij} = \frac{(x_{ik} - x_{jk})^2}{2||x_i - x_j||_A} = \frac{(x_{ik} - x_{jk})^2}{2r_{ij}} \tag{27}$$

$$\frac{\partial \psi_{ij}}{\partial a_u} = \frac{(x_{ik} - x_{jk})^2}{2||x_i - x_j||_A^2} \frac{\partial r_{ij}}{\partial a_u} \tag{28}$$

$$= -\frac{(x_{ik} - x_{jk})^2 (x_{iu} - xju)^2}{4(||x_i - x_j||_A)^3} \tag{29}$$

$$= -\frac{\sum_D (x_{ik} - x_{jk})^2}{2||x_i - x_j||_A} \frac{1}{(\sum_D ||x_i - x_j||_A)^2} \frac{\sum_D (x_{iu} - x_{ju})^2}{2||x_i - x_j||_A} \tag{30}$$

$$\frac{\partial g}{\partial a_k \partial a_u} = -\frac{\partial^2 \beta}{\partial a_k \partial a_u} \tag{31}$$

$$= H_{ku} \tag{32}$$

2 Appendix B

Equations to set up the learning of full matrix A:

Problem:
The problem can be seen as a 3 step optimization procedure iterated over till convergence.
(i) The first step involves maximizing the distance between the dissimilar pairs of points.
(ii) The second step projects the above optimum onto the set of all matrix satisfying the constraint for f(A) given below.
(iii) The final step in the iteration involves projecting the above result A' onto the set of all positive semi-definite matrices.

$$\max_A g(A) = \sum_{(x_i,x_j)\in D} ||x_i - x_j||_A \tag{33}$$

$$s.t. f(A) = \sum_{(x_i,x_j)\in S} ||x_i - x_j||_A^2 \leq 1 \tag{34}$$

$$A \geq 0 \tag{35}$$

2.1 Step I: Gradient Ascent for Maximization

$$\max_A g(A) = \sum_{(x_i,x_j)\in D} ||x_i - x_j||_A \tag{36}$$

$$\frac{\partial g(A)}{\partial A_{ij}} = \sum_{(x_k,x_l)\in D} \frac{\partial r_{kl}}{\partial A_{ij}} \tag{37}$$

$$where, \tag{38}$$

$$r_{kl} = \sqrt{(x_k - x_l)^T A (x_k - xl)} \tag{39}$$

$$\frac{r_{kl}}{\partial A_{ij}} = \frac{1}{2\sqrt{(\tau_{kl})}} \frac{\partial \tau_{kl}}{\partial A_{ij}}, \tag{40}$$

$$where, \tag{41}$$

$$\tau_{kl} = (x_k - x_l)^T A (x_k - x_l) \tag{42}$$

$$= \sum_u \sum_v (x_{ku} - x_{lu}) A_{uv} (x_{kv} - x_{lv}) \tag{43}$$

$$\frac{\partial \tau_{kl}}{\partial A_{ij}} = (x_{ki} - x_{li})(x_{kj} - x_{lj}) \tag{44}$$

$$\frac{\partial r_{kl}}{A_{ij}} = \frac{1}{2\sqrt{\tau_{kl}}} (x_{ki} - x_{li})(x_{kj} - x_{lj}) \tag{45}$$

$$= \frac{(x_{ki} - x_{li})(x_{kj} - x_{lj})}{2||x_k - x_l||_A} \tag{46}$$

$$\frac{\partial g(A)}{\partial A_{ij}} = \sum_{(x_k, x_l) \in D} \frac{(x_{ki} - x_{li})(x_{kj} - x_{lj})}{2||x_k - x_l||_A} \tag{47}$$

$$A_{ij} = A_{ij} + \alpha \frac{\partial g(A)}{\partial A_{ij}}, \tag{48}$$

The above derivation is used to obtain the gradient ascent update step in which α is the learning rate we use.

2.2 Step II: Projection onto set of matrices satisfying constarint $f(A) < 1$

Method: Langrange's Optimization
In this projection step we optimize in order to find a matrix A' closest to A in terms of $||A' - A||_F^2$ which also satisfies the above constraint regarding f(A).

Objective function (quadratic):

$$A' = arg \min_{A'} h(A') = arg \min_{A'} ||A' - A||_F^2 \tag{49}$$

Linear Constraint:

$$y(A') = \sum_{(x_i, x_j) \in S} ||x_i - x_j||_{A'}^2 - 1 \leq 0 (from f(A') \leq 1) \tag{50}$$

Primal Problem:

$$\mathcal{L}(A', \alpha) = ||A' - A||_F^2 + \alpha \{ \sum_{(x_i, x_j) \in S} ||x_i - x_j||_A^2 - 1 \} \tag{51}$$

$$\theta_P(A') = max_{\alpha \geq 0} \mathcal{L}(A', \alpha) \tag{52}$$

$$A'^* = \min_{A'} \theta_P(A') \tag{53}$$

$h(A')$ and $y(A')$ are both convex. As a result the solution of the dual problem is equal to the primal problem.

Dual Problem:

$$A' = \max_{\alpha \geq 0} \theta_D(A') \tag{54}$$

$$\theta_D(A') = \min_{A'} \mathcal{L}(A', \alpha) \tag{55}$$

In order to compute $\theta_D(A')$ we use:

$$\mathcal{L}(A',\alpha) = ||A'-A||_A^2 + \alpha\{\sum_{(x_i,x_j)\in S} ||x_i - x_j||_{A'}^2 - 1\} \tag{56}$$

$$= \sum_k \sum_l (A'_{kl} - A_{kl})^2 \tag{57}$$

$$+ \alpha\{\sum_{(x_i,x_j)\in S} \sum_k \sum_l ()A'_{kl}(x_{ul} - x_{vl})\} - \alpha \tag{58}$$

$$\frac{\partial \mathcal{L}(A',\alpha)}{\partial A_{ij}} = 2(A'_{ij} - A_{ij}) + \alpha \sum_{(x_u,x_v)\in S} (x_{ui} - x_{vi})(x_{uj} - x_{vj}) \tag{59}$$

$$Let,\ \beta_{ij} = \sum_{(x_u,x_v)\in S} (x_{ui} - x_{vi})(x_{uj} - x_{vj}) \tag{60}$$

$$\tag{61}$$

setting the derivative to zero we get,

$$\frac{\partial \mathcal{L}(A',\alpha)}{\partial A'_{ij}} = 0 \tag{62}$$

$$2(A'_{ij} - A_{ij}) + \alpha\beta_{ij} = 0 \tag{63}$$

$$A'_{ij} = A_{ij} - \frac{\alpha}{2}\beta_{ij}. \tag{64}$$

Substituting into the equation for $\mathcal{L}(A',\alpha)$ we get

$$\theta_D(\alpha) = \sum_i \sum_j (A_{ij} - \frac{\alpha\beta_{ij}}{2} - A_{ij})^2 \tag{65}$$

$$+ \alpha\{\sum_{(x_u,x_v)\in S} \sum_i \sum_j (x_{ui} - x_{vi})(A_{ij} \tag{66}$$

$$- \frac{\alpha\beta_{ij}}{2})(x_{uj} - x_{vj}) - 1\} \tag{67}$$

$$= \frac{\alpha^2}{4}||\beta||_F^2 + \alpha f(A) - \frac{\alpha^2}{2} f(B) - \alpha \tag{68}$$

$$\frac{\partial \theta_D(\alpha)}{\partial \alpha} = \frac{\alpha}{2}||\beta||_F^2 + f(A) - \alpha f(\beta) - 1 \tag{69}$$

$$\tag{70}$$

Setting the derivative to zero we get,

$$\frac{\partial \theta_D(\alpha)}{\partial \alpha} = 0 \tag{71}$$

$$\alpha = \frac{2(f(A) - 1)}{2f(\beta) - ||\beta||_F^2} \tag{72}$$

Hence, our update rule becomes:

$$A'_{ij} = A_{ij} - \frac{\alpha}{2}\beta_{ij} \tag{73}$$

and in general,

$$A' = A - \frac{\alpha\beta}{2} \tag{74}$$

2.3 Step III: Projection of A' onto set of all p.s.d. matrices.

We first perform diagnolization of A':
$A' = X^T \wedge X$ and eliminate all -ve eigen values of A' as $\wedge' = diag(max0, \lambda_1, ..., max0, \lambda_2)$ and compute $A'' = X^T \wedge' X$.