 doi:10.20944/preprints201809.0197.v1

Unsupervised Metric Learning using low dimensional embedding

Parag Jain
cs13m1008@iith.ac.in
Indian Institute of Technology, Hyderabad

Unsupervised metric learning has been generally studied as a byproduct of dimensionality reduction or manifold learning techniques. Manifold learning techniques like Diffusion maps, Laplacian eigenmaps has a special property that embedded space is Euclidean. Although laplacian eigenmaps can provide us with some (dis)similarity information it does not provide with a metric which can further be used on out-of-sample data. On other hand supervised metric learning technique like ITML which can learn a metric needs labeled data for learning. In this work propose methods for incremental unsupervised metric learning. In first approach Laplacian eigenmaps is used along with Information Theoretic Metric Learning(ITML) to form an unsupervised metric learning method. We first project data into a low dimensional manifold using Laplacian eigenmaps, in embedded space we use euclidean distance to get an idea of similarity between points. If euclidean distance between points in embedded space is below a threshold t_1 value we consider them as similar points and if it is greater than a certain threshold t_2 we consider them as dissimilar points. Using this we collect a batch of similar and dissimilar points which are then used as a constraints for ITML algorithm and learn a metric. To prove this concept we have tested our approach on various UCI machine learning datasets. In second approach we propose Incremental Diffusion Maps by updating SVD in a batch-wise manner.

Keywords: unsupervised, metric learning, embedding learning, laplacian, information theoretic, diffusion maps

1 Unsupervised Information Theoretic Metric Learning using Laplacian eigenmaps

1.1 Laplacian eigenmaps

Laplacian eigenmaps learns a low dimensional representation of the data such that the local geometry is optimally preserved, this low-dimensional manifold approximate the geometric structure of data. Steps below describes the methods in detail.
Consider set of data points $X \in \mathcal{R}^N$, goal of laplacian eigenmaps is to find an embedding in m dimensional space where $m < N$ preserving the local properties of data.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 September 2018 doi:10.20944/preprints201809.0197.v1

1. Construct a graph $G(V, E)$ where E is set of edges and V is a set of vertices. Each node in the graph G corresponds to a point in X, we connect any two vertices v_i and v_j by an edge if they are close, closeness can be defined in 2 ways:

   (a) $||x_i - x_j||^2 < \epsilon$, $||.||$ is euclidean norm in $\mathcal{R}^N$ or,

   (b) x_i is in k nearest neighbour of x_j

   here ϵ & k are user defined parameters.

2. We construct a weight matrix $W(i, j)$ which assigns weights between each edge in the graph G, weights can be assigned in two ways:

   (a) Simple minded approach is to assign $W(i, j) = 1$ if vertices v_i and v_j are connected otherwise 0.

   (b) Heat kernel based, we assign weight $W(i, j)$ such that:

   $$W(i,j) = \begin{cases} exp(\frac{||x_i - x_j||^2}{t}) & \text{if } v_i \text{ and } v_j \text{ are connected} \\ 0 & \text{otherwise} \end{cases}$$

3. Construct laplacian matrix $L = D - W$ of the graph G, where D is a diagonal matrix with $D_{ii} = \Sigma_j W(i, j)$. Final low dimensional embedding can be computes by solving generalized eigen decomposition

   $$Lv = \lambda Dv$$

   Let $0 = \lambda_0 \leq \lambda_1 ... \leq \lambda_m$ be the first smallest $m + 1$ eigenvalues, choose corresponding eigenvectors $v_1, v_2 ... v_m$ ignoring eigenvector corresponding to $\lambda_0 = 0$. Embedding coordinates can be calculates as mapping:

   $$x_i \in \mathcal{R}^N \mapsto y_i \in \mathcal{R}^m$$
   $$where \quad y_i^T = [v_1(i), v_2(i), ...v_m(i)]$$

$y_i,\ i = 1, 2...n$ is the coordinates in m dimensional embedded space.

1.2 Information Theoretic Metric Learning

Information Theoretic Metric Learning(ITML)[3] learns a mahalanobis distance metric that satisfy some given similarity and dissimilarity constraints on input data. Goal of ITML algorithm is to learn a metric of form $d_A = (x_i - x_j)' A (x_i - x_j)$ according to which similar data point is close relative to dissimilar points.
ITML starts with an initial matrix d_{A_0} where A_0 can be set to identity matrix(I) or inverse of covariance of the data and eventually learns a metric d_A which is close to starting metric d_{A_0} and satisfies the the defined constraints. To measure distance between metrics it exploits the bijection between Gaussian distribution with fixed mean μ and Mahalanobis distance,

$$\mathcal{N}(x|\mu, A) = \frac{1}{Z} exp\Big(-\frac{1}{2} d_A(x, \mu) \Big)$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 September 2018 doi:10.20944/preprints201809.0197.v1

Using the above connection, the problem is formulated as:

$$\begin{aligned} \min_{A} & \int \mathcal{N}(x,\mu,A_0) log\left(\frac{\mathcal{N}(x,\mu,A_0)}{\mathcal{N}(x,\mu,A)}\right) dx \\ \text{subject to} \quad & d_A(x_i,x_j) \leq u \;\; \forall (x_i,x_j) \in S, \\ & d_A(x_i,x_j) \geq l \;\; \forall (x_i,x_j) \in D, \\ & A \succeq 0 \end{aligned} \tag{1}$$

Above formulation can be simplified by utilizing the connection between KL-divergence and LogDet divergence which is given as,

$$\begin{aligned} & \int \mathcal{N}(x,\mu,A_0) log\left(\frac{\mathcal{N}(x,\mu,A_0)}{\mathcal{N}(x,\mu,A)}\right) dx = \frac{1}{2} D_{ld}(A,A_0) \\ & \text{where,} \quad D_{ld}(A,A_0) = tr(AA_0^{-1}) - logdet(AA_0^{-1}) - d \end{aligned} \tag{2}$$

Using 2 and 1 problem can be reformulated as:

$$\begin{aligned} \min_{A} \quad & D_{ld}(A,A_0) \\ \text{subject to} \quad & d_A(x_i,x_j) \leq u \;\; \forall (x_i,x_j) \in S \\ & d_A(x_i,x_j) \geq l \;\; \forall (x_i,x_j) \in D \\ & A \succeq 0 \end{aligned} \tag{3}$$

Above formulation can be solved efficiently using bregman projection method as described in [3].

1.3 Proposed algorithm

We propose a method which combines Laplacian eigenmaps and ITML to form an unsupervised metric learning method. Laplacian eigenmaps as described in 1.1 can be used to recover underlying low dimensional manifold of data where we can use euclidean distance to get (dis)similarity information using which we can learn a metric using supervised metric learning settings like ITML. In box 1.3 we describe the details of proposed algorithm.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 September 2018 doi:10.20944/preprints201809.0197.v1

Manifold + supervised metric learning

Input:
$X \in N \times k$, is input data in k dimensional space
t_s: threshold for similarity
t_d: threshold for dissimilarity
ϵ, m: parameters for laplacian eigenmaps algorithm, $m < k$
Output: Learned metric $A_{k \times k}$
Steps:

1. Construct low dimensional embedding:
   $\mathcal{E} = \text{laplacianEigenmaps}(X, \epsilon, m)$

2. Construct similarity and dissimilarity pairs:
   for each pair $(x_i, x_j) \in \mathcal{E}$:
   $p = ||x_i - x_j||^2$
   if $p \leq t_s$ then $S \leftarrow S \cup (x_i, x_j)$
   if $p \geq t_d$ then $D \leftarrow D \cup (x_i, x_j)$

3. Apply ITML 1.2 procedure to learn metric:
   $A = \text{itml}(X, S, D)$

We reduce the time complexity of above algorithm by limiting number of similar and dissimilar points.

Calculating threshold: Manually setting thresholds for similarity and dissimilarity can be difficult in real case, but we can set these thresholds with a simple procedure of calculating the distance extremes for data in embedded space $\mathcal{E}$. A safe way for setting threshold is to compute histogram of distances and set t_s & t_d to be the 5^{th} and 95^{th} percentiles respectively.

1.4 Results

We have evaluated our method on different UCI datasets[4], to best of our knowledge there is no other method that does exactly what we tried we compare our algorithm with euclidean distance, we construct similar and dissimilar pairs using euclidean distance and then learn a metric using ITML.
We split each dataset randomly into two parts 80% for training and 20% for testing, to evaluate the learned metric with k-NN classification using learned metric as distance measure. All results presented are the average of 5 runs.

Dataset	**Proposed method**	**Euclidean + ITML**
Letter recognition	**95.25**	93.75
Iris	**83.3**	76.6
Scale	**84.7**	82.6
Yeast	**60.7**	59.4
Wine	**75.9**	74.1

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 September 2018 doi:10.20944/preprints201809.0197.v1

The method we proposed in this section is general and the metric learned can be used for clustering dataset. From results it is clear that performing low dimensional embedding to obtain similar and dissimilar pairs performs better than directly apply a general measure than euclidean distance. *It is important to note that the comparison is not been made directly between euclidean distance and proposed method rather in both the cases we learned a metric using ITML .*

2 Incremental Diffusion Maps

2.1 Diffusion maps

Diffusion maps[2] are non-linear dimensionality reduction technique. It achieves dimensionality reduction by exploiting relation between Markov chains and heat diffusion. Diffusion map embeds data into low dimensional space such that euclidean distance between points in embedded space is approximated as diffusion distance in the original feature space.
Consider a graph $G = (\Omega, W)$ where $\Omega = \{x_i\}_{i=1}^{N}$ are data samples and W is a similarity matrix with $W(i,j) \in [0,1]$. W is obtained by applying Gaussian kernel on distances,

$$W(i,j) = exp\left\{\frac{-d^2(i,j)}{\sigma^2}\right\} \tag{4}$$

where d(i,j) is the distance between x_i and x_j and σ is kernel size. Using W we can obtain a transition matrix by row wise normalizing the similarity matrix:

$$P(i,j) = \frac{W(i,j)}{d_i} \text{ where, } d_i = \sum_{j=1}^{N} W_{ij} \tag{5}$$

Transition matrix reflects the local geometry of the data, where $p(x,y)$ is the probability of transition from x to y in one step. If we look forward in time than P^t gives the probability of transition from x to y in t time steps. Intuitively what that means is running the diffusion in time will reveal the geometric structure of data at different scales.

Diffusion process We define a new kernel L using normalized laplacian:

$$L^{(\alpha)} = D^{-\alpha} W D^{-\alpha} \tag{6}$$

where, D is diagonal matrix suct that $D_{ii} = \sum_j W_{i,j}$ and $\alpha \in \mathbb{R}$. Apply weighted graph Laplacian normalization to this kernel:

$$M = (D^{(\alpha)})^{-1} L^{(\alpha)} \tag{7}$$

where $D^{(\alpha)}$ is a diagonal matrix such that $D^{(\alpha)}_{ii} = \sum_j L^{(\alpha)}_{i,j}$

Diffusion distance can be defined in terms of eigenvectors of matrix M^t

$$D_t(i,j) = ||\Psi_t(x_i) - \Psi_t(x_j)||^2 \tag{8}$$

ψ_l is right eigenvector and λ_i are eigenvalues of M^t and $\Psi_t(x) = (\lambda_1^t \psi_1(x), \lambda_2^t \psi_2(x) ..., \lambda_k^t \psi_k(x))$.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 September 2018 doi:10.20944/preprints201809.0197.v1

2.2 Updating SVD

Given matrix $A_{m\times n}$ and $\hat{A}_{m\times n} = U\Sigma V'$ where $\hat{A}_{m\times n}$ is rank-k approximation of $A_{m\times n}$, [5] describes the procedure to get the approximate rank-k approximation of $\left[\hat{A}_{m\times n}, B_{m\times r}\right]$ and $\left[\hat{A}_{m\times n}; B_{r\times n}\right]$. Method proposed by [5] is summarized below,

Updating Columns

1. Let the QR decomposition of $(I - UU')B$ be $(I - UU')B = QR$ where R is upper triangular.
2. Get SVD decomposition of $\begin{bmatrix}\Sigma & U'B\\ 0 & R\end{bmatrix} = \hat{U}\hat{\Sigma}\hat{V}'$
3. Then best rank-k approximation of $\left[\hat{A}_{m\times n}, B_{m\times r}\right]$ is given as $\big([U, Q]\hat{U}\big)\hat{\Sigma}\big(\begin{bmatrix}V & 0\\ 0 & I\end{bmatrix}\hat{V}\big)'$

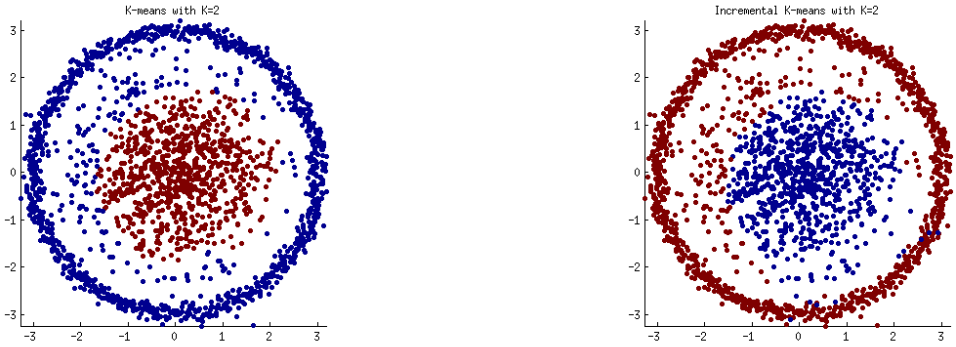
Updating Rows

1. Let the QR decomposition of $(I - VV')B'$ be $(I - VV')B' = QL'$ where L' is lower triangular.
2. Get SVD decomposition of $\begin{bmatrix}\Sigma & 0\\ BV & L\end{bmatrix} = \hat{U}\hat{\Sigma}\hat{V}'$
3. Then best rank-k approximation of $\left[\hat{A}_{m\times n}; B_{m\times r}\right]$ is given as $\big(\begin{bmatrix}U & 0\\ 0 & I\end{bmatrix}\hat{U}\big)'\hat{\Sigma}\big([V, Q]\hat{V}\big)$

2.3 Proposed algorithm

In box 2.3 we have described the basics steps of proposed incremental diffusion maps algorithm. Calculation can be done in a very efficient way by storing the previous values of row sum at step 2 and step 4 of the algorithm.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 September 2018 doi:10.20944/preprints201809.0197.v1



(a) Clusters using batch mode. Run time: 27.5 sec

(b) Clusters using incremental mode. Run time 6.4 sec

Figure 1: Result incremental diffusion maps

Incremental Diffusion map algorithm

Input:
D: distance matrix of size $N \times N$
T: distance matrix of new points $N + p \times p$
n: dimension of embedding
U,S,V : SVD of markov transition matrix M^t obtained using old points
Diffusion maps paramenets: σ, α
Output: $\hat{U}, \hat{S}, \hat{V}$, DD: Diffusion distance matrix of size $N + p \times N + p$
Steps:

1. Update D by adding new rows and columns from T to get $\hat{D}(N + p \times N + p)$
2. Apply gaussian kernel to get W on $\hat{D}$
3. Compute new kernel by applying laplacian normalization, $L^{(\alpha)} = D^{-\alpha} W D^{-\alpha}$
4. Calculate $M = (D^{(\alpha)})^{-1} L^{(\alpha)}$
5. Get new rows R and columns C from M
6. Get $\hat{U}, \hat{S}, \hat{V}$ by using update SVD procedure 2.2
7. Use $\hat{U}, \hat{S}$ to get new diffusion distance matrix $DD_t(i, j) = ||\Psi_t(x_i) - \Psi_t(x_j)||^2$

2.4 Results

To check the performance of proposed method we have used a toy dataset having two non-linearly separable clusters, total number of points in dataset are 2000, to test the effectiveness of incremental procedure we divide the dataset into 2 parts of 1600 and 400 points. Initial 1600 points were used in batch mode to get initial embedding then we update the embedding

using proposed incremental diffusion maps procedure. To simulate real world setting we update the embedding 10 points at a time, calling incremental update 40 time in this case. Once we get the final embedding after updating all 400 points for both batch and incremental setting we use k-means with k = 2 to visualize the effectiveness of diffusion distance learned. Final results are plotted in figures 1.

All the experiment was done on Sony machine with i3 processor and 4GB RAM. It can be observed that the results are very close with very less run time than batch mode. We have tested this methods on other datasets but due to high approximation error in markov matrix calculation and update svd procedure results are not appreciable.

2.5 Conclusion and future work

From our results we conclude that approximation in markov matrix and svd is not very appreciable, however, the current method used a basic svd update procedure and using recent advanced procedures [1] could lead to further improvements.

References

[1] Xilun Chen and K Selcuk Candan. "LWI-SVD: low-rank, windowed, incremental singular value decompositions on time-evolving data sets". In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2014, pp. 987–996.

[2] Ronald R Coifman and Stephane Lafon. "Diffusion maps". In: *Applied and computational harmonic analysis* 21.1 (2006), pp. 5–30.

[3] Jason V Davis et al. "Information-theoretic metric learning". In: *Proceedings of the 24th international conference on Machine learning*. ACM. 2007, pp. 209–216.

[4] M. Lichman. *UCI Machine Learning Repository*. 2013. URL: `http://archive.ics.uci.edu/ml`.

[5] Hongyuan Zha and Horst D Simon. "On updating problems in latent semantic indexing". In: *SIAM Journal on Scientific Computing* 21.2 (1999), pp. 782–791.