Embedding of Embeddings

Data Mining Lab, SS 2013

Supervised by
Dr. Thomas Gärtner
Daniel Paurat

Written by:
Priyanka Dank
University of Bonn
Matriculation No. 2517008

Contents

1. Introduction

Embedding is mapping from one space to the other. In data analysis, embedding is mapping of high dimensional data to a lower dimensional space. It is also known as dimensionality reduction. Embedding is useful as fewer dimensions allow better generalization; also it helps to understand the structure of the data, increases the efficiency as it compresses the data and reduces noise. There are number of different embedding techniques available. All of those try to keep underlying structure of the data in the space that data gets embedded into. Different techniques emphasize different aspects of the data. We wonder how redundant these techniques are which emerges the idea of "***Embedding of embeddings***". It is placing the embedding techniques on two-dimensional map such that the techniques that deliver similar results are placed closer together. Keeping the focus on this goal, we carried out an experiment on number of different datasets which is being discussed in this study.

2. Related Work

The section discusses about some of the more common embedding techniques that are considered in our experiment.

Principal Component Analysis (PCA) [Pearson 1901] finds orthogonal axes with highest variance in high dimensional space. It uses the n most dominant ones to embed the data into a low dimensional space. This is the basic idea of PCA. The detailed description of the method is given below.

Consider there are m-dimensional n data points arranged in the matrix as given below.

$$\mathrm{X} = \begin{pmatrix} x_1^1 & \cdots & x_1^n \\ \vdots & \ddots & \vdots \\ x_m^1 & \cdots & x_m^n \end{pmatrix}$$

Initially data is evaluated with respect to its mean. It is achieved by subtracting the mean from each of the data dimensions

Calculating mean: $\overline{x_i} = \frac{1}{n}\sum_{k=1}^{n} x_i^k \quad where\ i = 1,2, \ldots . m$

Subtracting mean from data point $x_i^j \leftarrow x_i^j - \overline{x_i}$

Due to the above steps the data is now centered on origin. We calculate the variance of the data using the equation below. This represents the projected data.

$$\text{variance(i)} = \frac{1}{n-1}\sum_{k=1}^{n}(x_i^k)^2$$

Next step in analysis is to formulate the covariance matrix such that the diagonal entries are the variances of the row data and non-diagonal entries are the co-variance values between rows. The matrix can be calculated using the formula

$$S(X) = \frac{1}{n-1} X X^T$$

Then eigenvalues λ_i and eigenvectors q^j of the covariance matrix are calculated. The eigen-vectors are put in matrix Q in descending order of the respective eigenvalues. It is possible to discard some eigenvectors with small values and select the eigenvectors with high variance i.e. principal components. Final data will have corresponding dimensions. The final data is calculated by

$$Y^{(k)}=Q^T x^{(k)}$$

Thus with the PCA, we can achieve dimensionality reduction.

Independent Component Analysis (ICA) [Hyvärinen 2000] computes the embedding diffe-rently. It represents multidimensional random vector as a linear combination of non-gaussian random variables i.e. independent components.

ICA is based on statistical principles. Consider there are m-dimensional n data points ar-ranged in the *m x n* matrix as given below.

$$X = \begin{pmatrix} x_1^1 & \cdots & x_1^n \\ \vdots & \ddots & \vdots \\ x_m^1 & \cdots & x_m^n \end{pmatrix}$$

Let S be the source signal *m x n* matrix containing independent components with non-gaussian distribution. We assume linear independence between X and S which is represented as

$$X = WS$$

Where w is *m x m* matrix and is called as mixing matrix.

The matrices W and S are unknown and ICA estimates these matrices using statistical inde-pendence.

Fast Independent Component Analysis (FastICA) [Hyvärinen 2000] is based on ICA. It uses a fixed-point iteration scheme for finding a maximum of non-gaussianity of $W^T X$ where W is weight vector and X is data matrix. FastICA is a computationally highly efficient method.

FastICA preprocesses the data by centering and whitening. Centering of the data is achieved by calculating the mean of the components of each vector and subtracting the mean from each of the components. After centering of the data, the eigenvalue decomposition (EVD) of the covariance matrix is computed as

$$E\{XX^T\} = EDE^T$$

Where E is the matrix of eigenvectors and D is the diagonal matrix of eigenvalues. The whitening of the data is then performed using

$$\widetilde{X} = ED^{-1/2}E^TX$$

Single component extraction is done using iterative algorithm. The goal of the algorithm is to find the direction for the weight vector W by maximizing the non-gaussianity of the projection vector W^TX. The algorithm starts with randomizing the initial weight vector W. Next step is to calculate the average over all column-vectors of matrix X. The weight vector is then divided by the average. The process is repeated till it converges.

The algorithm of multiple component extraction is based on the above mentioned FastICA algorithm of single component extraction. The input is several units weight vectors $W_1, \ldots, W_n$. It is necessary to decorrelate the output vectors so that they should not reach the same maxima. It is achieved by iterative algorithm [Hyvärinen 1999a] given below

1. Let $W = W / \sqrt{WW^T}$
   Repeat 2. until convergence:
2. Let $W = \frac{3}{2}W - \frac{1}{2}WW^TW$

The independent component matrix is achieved by the product WX. The matrix represents reduced dimensions of input data.

Isomap [Tenenbaum 2000] finds the neighbors of each data point in high dimensional data space. The neighborhood relations are represented as a weighted graph G. It computes the pairwise distances between all pairs of points based on nearest neighbor graph G by computing their shortest path distances $d_G(i,j)$. It embeds the data by applying Multidimensional Scaling [Trosset 1993] to the matrix of graph distances $D_G = \{d_G(i,j)\}$ and preserving the geometry of the manifold. The cost function to minimize is given by

$$E = \| \tau(D_G) - \tau(D_Y) \|_{L^2}$$

Where D_Y: matrix of Euclidean distances, τ operator converts distances to inner products such that inner products give optimized representation of the geometry of the data.

The cost function is computed by taking the L^2 matrix norm of the subtraction of the matrices D_G and D_Y which contain the distanced in the form of inner product that uniquely determines the geometry of the data. The solution contains set of points with respect to top d eigenvectors of matrix D_G.

Simplex Volume Maximization (SiVM) [Thurau, Kersting, Bauckhage 2010] selects opposing points of the dataset´s convex hull so that points couldn´t be further apart. The embedding is done by expressing all data points as convex combination. This is the basic idea of the method. The detailed description is as follows.

Let V be the *d x n* data matrix containing n samples and W be the *d x k* matrix containing the centroids of the data points in matrix V. Let H be the *k x n* matrix of coefficients which is restricted to the convexity and G be *n x k* matrix restricted to unary column vectors. The matrix W is called basis vector which is obtained by selecting some data points from the matrix V and is expressed as W= VG. The method aims to represent the data as convex combination of certain data points in V which can be given by V=VGH by minimizing Frobenius norm:

$$E = \|V - WH\|^2$$

If we add a vertex w_{k+1} sampled from a data matrix V to the *d x k* basis vector W, the Frobenius norm will not increase i.e.

$$\left\|V^{d\,x\,n} - W^{d\,x\,(k+1)} H^{(k+1)\,x\,n}\right\|^2 \;\leq\; \|V^{d\,x\,n} - W^{d\,x\,k} H^{k\,x\,n}\|^2$$

This is the idea of simplex volume maximization. To find the vertex that maximizes the simplex volume, consider the simplex with lengths $d_{i,j}$ of the edges between n+1 vertices of an n-simplex. The vertex v is selected from the matrix V and the basis vector is computed using following equation

$$w_k = \underbrace{argmax}_{k} \left[\sum_{i=1}^{n} d_{i.k} \left[a + \sum_{j=i+1}^{n} d_{j,k} - \frac{n-1}{2} \sum_{i=1}^{n} d_{i,k}^2 \right] \right]$$

With this iterative distance computation in SiVM, we get the new basis vector as the dimensionality reduction.

CUR matrix decomposition [Drineas 2006] decomposes given *m x n* matrix A into three smaller matrices C, U and R such that product of these three matrices approximates the given matrix A. The matrix C is *m x c* matrix consisting of randomly picked c columns of A, matrix R is *r x n* matrix consisting of randomly picked r rows of A and matrix U is c *x r* matrix computed from matrices C and R. It has been proven that for the column and row selections, the resultant matrix A'=CUR should fulfill following criteria:

$$\|A - A'\|_\varepsilon \;\leq\; \underbrace{min}_{D:rank\,(D)\leq k} \;\; \|A - D\|_\varepsilon + poly(k, {}^{1}/_{c})\|A\|_F$$

Where $\varepsilon = 2, k = 1, \dots, rank(A)$, for any matrix X: $\|X\|_2$ is its spectral norm and $\|X\|_F$ is its Frobenius norm.

The idea behind ***Singular Volume Decomposition (SVD)*** [Yang, Ma, Buja 2011] is identifying and ordering the dimensions along the data points that show the most variations. This enables to represent high dimensional data using fewer dimensions. Any m x n matrix A, with m > n, can be written using a singular value decomposition as A = USV^T where U is m x m matrix and contains the columns which are orthonormal eigenvectors of AA^T. S is a diagonal matrix of singular values which are ranked from greatest to least so that data points showing most variations are captured. V is n x n matrix and contains the columns which are orthonormal eigenvectors of A^{T}A.

Non-negative Matrix factorization (NMF) [Cho, Saul 2011] is a computational method used for dimensionality reduction. It factorizes data matrix into a low rank and non-negative factors A = BW where B and W have non-negative elements. B is of dimension n x r and is called the basis matrix as its row contains set of basis vectors and W is of dimension r x m, which is called a weight matrix as its row contains coefficient sequences. The result (BW) can be interpreted as weighted sum of each of the basis vectors in B, the weights are the corresponding columns of W.

In ***Kmean matrix factorization*** [Ding 2007], n data points are classified randomly into k clusters. The initial cluster seeds are chosen arbitrarily. The squared Euclidean distance from each point to each cluster is computed and each data point is assigned to the cluster closest to it. For each cluster, the new centroid is computed. This is repeated until the process stabilizes. Thus the goal is to minimize the squared distance given by

$$\underbrace{\arg min}_{S} \sum_{i=1}^{k} \sum_{x_j \in S_i} \|x_j - \mu_i\|^2$$

Where μ_i is the centroid of the respective cluster S$_i$ .It is can be calculated using the formula $\frac{1}{|S_i|} \sum_{x_i \in S_i} x_i$

Archetypal Analysis (AA) [Cutler 1994] estimates the principal convex hull of the dataset and represents each data point by a convex combination of a collection of prototypes. The details are as mentioned below.

Archetypes are the original patterns from which things are copied. Consider X is a data matrix with data values {x$_i$} and the matrix Z with the values {z$_i$} which contains archetypes approximating the data values {x$_i$}. The coefficient matrix β is the matrix that satisfies the following equation

$$z_k = \sum_i \beta_{ki} x_i$$

Where $\beta_{ki} \geq 0, \sum_i \beta_{ki} = 1$ and k=1,…,p

The problem is to find the matrix Z and the matrices α and β such that the residual sum of squares (RSS) given below is to be minimized

$$RSS = \sum_i \left\| x_i - \sum_{k=1}^{p} \alpha_{ik} z_k \right\|^2$$

Where $\alpha_{ik} \geq 0, \sum_k \alpha_{ki} = 1$

As the archetypes lay on the convex hull of the data, the data can be well represented by the convex combination of the archetypes. Let N be the number of data points that define the boundary of the convex hull of the data. There are three cases for the values of k. For k =1, the sample mean of the data minimizes RSS. For k=N, archetypes are exactly the data points that define the boundary of the convex hull and the RSS for these archetype is 0. For 1<k<N, there are k archetypes on the boundary of the convex hull that minimizes RSS.

Locally Linear Embedding (LLE) [Roweis, Saul 2000] is useful for non-linear manifolds. First step of the LLE is to compute the neighbors of each data point. Let the data contains N real valued vectors$\overrightarrow{X_i}$. The neighbors are computed using either K-nearest-neighbors or epsilon radius. It is considered that the data point and its neighbors lie on the locally linear patch of the manifold. So, the original data points are reconstructed using linear combination of its neighbors by minimizing the reconstruction error given by

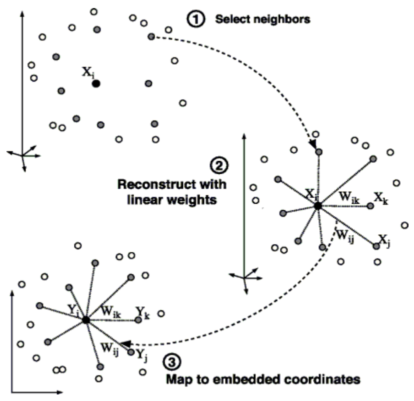
$$\varepsilon(W) = \sum_i \left| \overrightarrow{X_i} - \sum_j W_{ij} \overrightarrow{X_j} \right|^2$$

Where W is weight matrix with two constraints if $\overrightarrow{X_j}$ is not the neighbor of $\overrightarrow{X_i}$ then $W_{ij} = 0$ and $\sum_j W_{ij} = 1$ for the rows of the weight matrix. The linear combination is computed using an eigenvector-based optimization technique. The weights minimizing reconstruction error are invariant to scale, rotation and translation of the data points and its neighbors.

Then the high dimensional coordinates are mapped to the manifold coordinates by preserving neighborhood. Resulting Y$_i$ coordinates are computed by minimizing the cost function given below.

$$\phi(Y) = \sum_i \left| \vec{Y_i} - \sum_j W_{ij} \vec{Y_j} \right|^2$$

Thus the same weights W_{ij} are used to reconstruct the original data point in D dimensional space and the same point in d-dimensional space where D>>d.



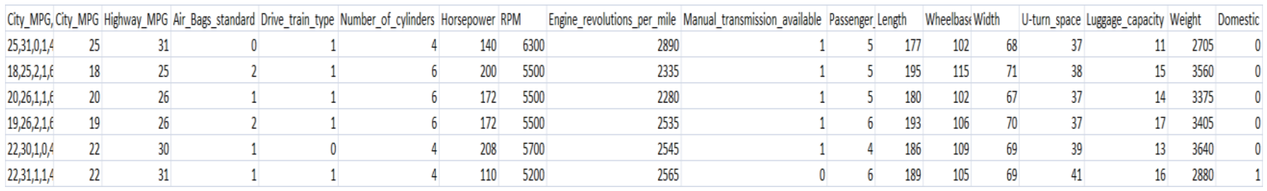
The whole process can be summarized in the above figure by [Roweis, Saul 2000].

3. **Experimental Details**

In this section, the experiment we performed for the comparison of the embedding techniques is explained in detail.

3.1 Application of Embedding Technique

Initially we considered the dataset *auto93* (figure 1) from UCI repository that stores some automobile data. The dimension of the dataset is 82 x 21.



City_MPG,	City_MPG	Highway_MPG	Air_Bags_standard	Drive_train_type	Number_of_cylinders	Horsepower	RPM	Engine_revolutions_per_mile	Manual_transmission_available	Passenger_	Length	Wheelbase	Width	U-turn_space	Luggage_capacity	Weight	Domestic
25,31,0,1,4	25	31	0	1	4	140	6300	2890	1	5	177	102	68	37	11	2705	0
18,25,2,1,6	18	25	2	1	6	200	5500	2335	1	5	195	115	71	38	15	3560	0
20,26,1,1,6	20	26	1	1	6	172	5500	2280	1	5	180	102	67	37	14	3375	0
19,26,2,1,6	19	26	2	1	6	172	5500	2535	1	6	193	106	70	37	17	3405	0
22,30,1,0,4	22	30	1	0	4	208	5700	2545	1	4	186	109	69	39	13	3640	0
22,31,1,1,4	22	31	1	1	4	110	5200	2565	0	6	189	105	69	41	16	2880	1

Figure 1: auto93 dataset

We have applied some of the more common embedding techniques mentioned in the section 2 on this dataset. With the application of each of these embedding techniques, we successfully achieved the reduction in dimensionality from 82 x 21 to 82 x 2. Figure 2 illustrates the results achieved by the application of each technique on the dataset *auto93*.

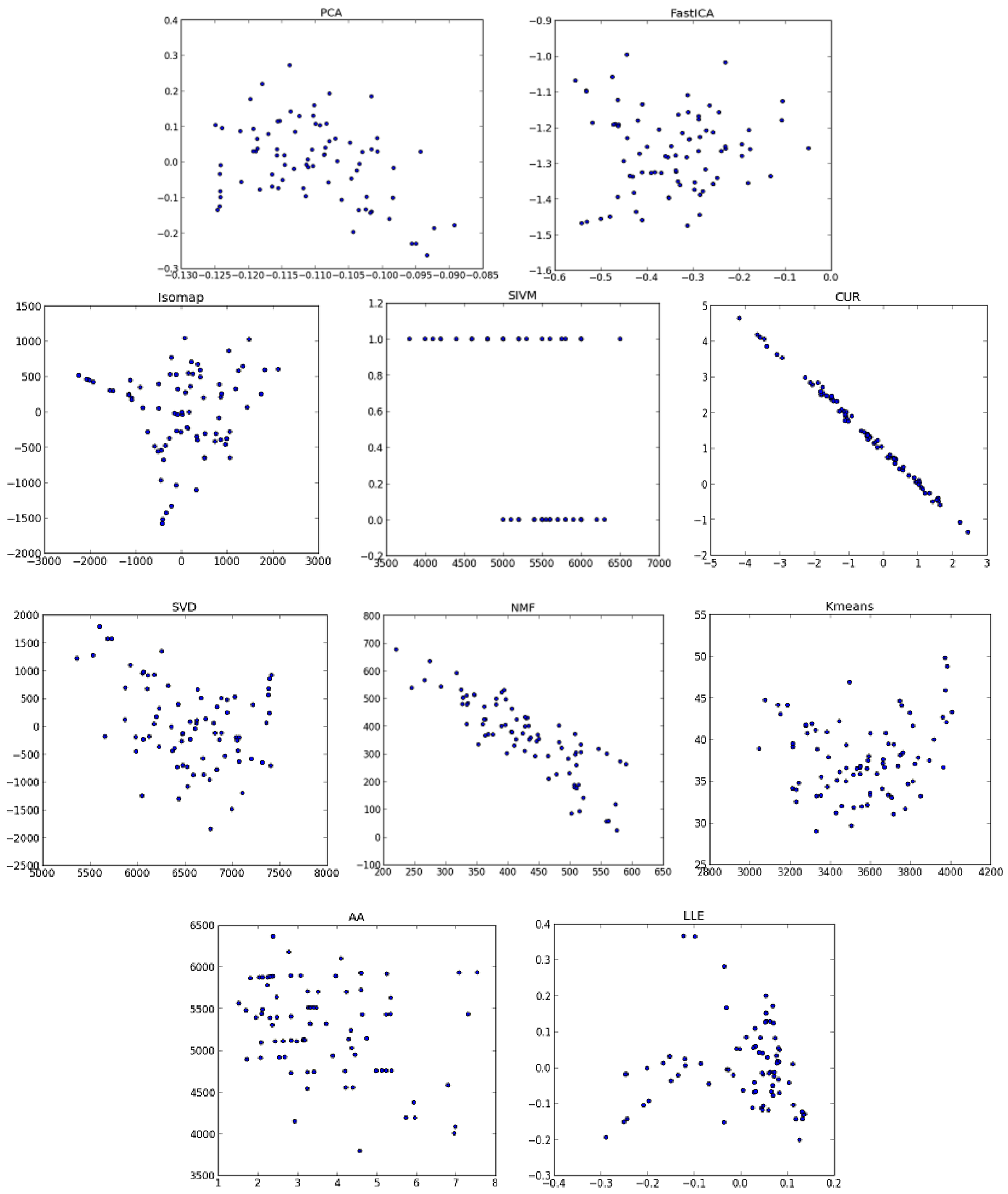


Figure 2: Results obtained by the applying embedding techniques on auto93

3.2 Finding Similarity Measure

Next crucial step in comparison of embedding techniques was to find *the similarity measure* between the different two dimensional embeddings. It can be seen from the figure 2 that embeddings are done on the different scales for example PCA [-0.3:0.4], SiVM [-0.2:1.2], Isomap [-2000:1500], FastICA [-1.6:-0.9]. In case of scale-dependent measure like epsilon-radius of each data point, we would have required to bring all embeddings to one scale in order to be able to use same epsilon. To avoid this, we decided to use the measure which is independent of the scale. We found that k-nearest-neighbors technique is not bound to the scale of embedding and rotation differences between the embeddings. Hence we decided to build a similarity measure based on ***k-nearest neighbors.***

We simplified the problem of comparing the techniques by formulating an example with less number of data points. For example, Consider three embedding techniques A, B, C which map 10 datapoints {X,Y,Z,P,Q,R,S,U,V,W} to lower dimensions. Let us assume that the mappings obtained using techniques A, B, C would look like as shown in figure 3.

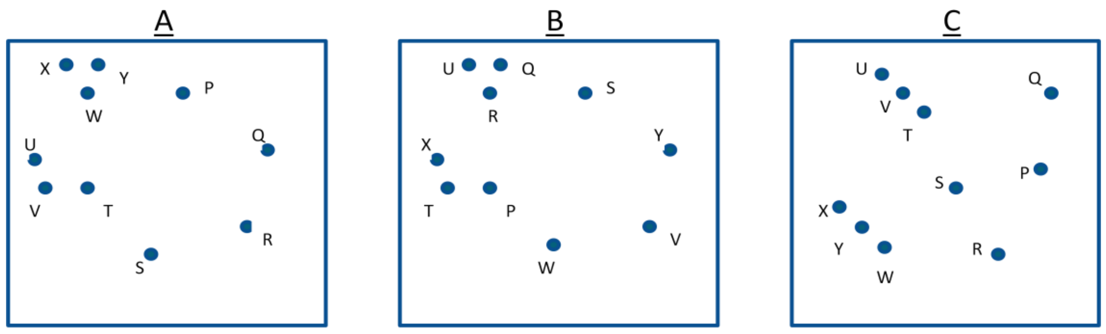


Figure 3: Example

As it can be seen from the figure 3, the output of A and B looks similar and that of A and C looks different. However after careful analysis of the outputs, we found that the neighborhood of the mapping is different in case of A and B while it is similar in case of A and C. To illustrate this, observe the data point 'X'. Let us set the criteria of 3-nearest neighbors for similarity measurement. In case of output of A, we get the data points 'Y' and 'Z' as neighbors of X; in case of output of B, the data points 'W' and 'P' are the neighbors of 'X' and in case of output of C, data points 'Y' and 'Z' are the neighbors of 'X'. This is illustrated in fig. 4.

Based on the above example and the results obtained in figure 2, we derived the formula to measure similarities between the embedding techniques. Let's call the data point and its k neighbors as a *knn* set. To find the similarity index between the methods m1 and m2, we compared each ith datapoint's knn set of one method (m1) with ith datapoint's knn set of other method (m2).

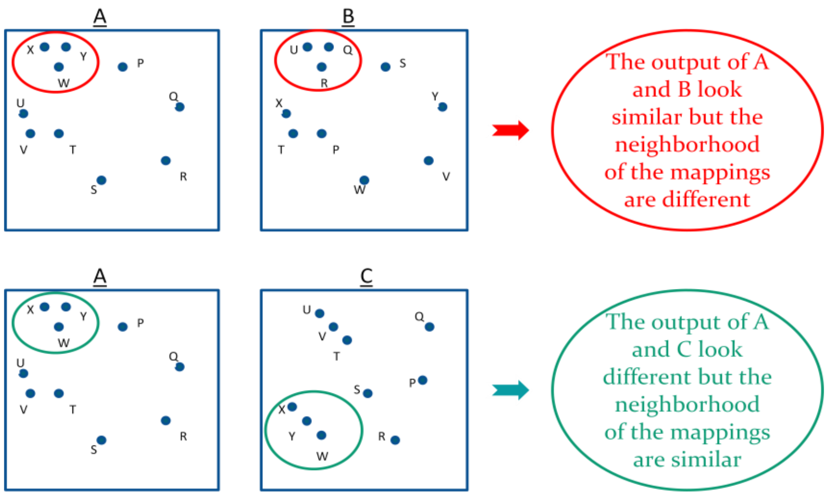

A
X
Y
W
P
U
Q
V
T
R
S
B
U
Q
R
S
X
Y
T
P
V
W
The output of A and B look similar but the neighborhood of the mappings are different
A
X
Y
W
P
U
Q
V
T
R
S
C
U
Q
V
T
S
P
X
Y
W
R
The output of A and C look different but the neighborhood of the mappings are similar

Figure 4: Example illustration for neighborhood mapping

Then we computed their intersection and union and formulated the equation for similarity measure as shown in formula 1.

$$\text{sim\_k}(\text{m1},\text{m2}) = \frac{1}{|\text{D}|}\sum_{\text{x}\in\text{D}}\frac{|(\text{knn}(\text{m1}(\text{x}))) \cap (\text{knn}(\text{m2}(\text{x})))|}{|(\text{knn}(\text{m1}(\text{x}))) \cup (\text{knn}(\text{m2}(\text{x})))|}$$

Where,

m1, m2 are compared embeddings, D is the set of datapoints

Formula 1: Similarity index of method m1 and m2

To demonstrate above equation, we can apply it to the embedding techniques A, B and C assumed in example above.

$$\text{sim\_3}(\text{A},\text{B}) = \frac{1}{10}\left[\frac{|(X,Y,W)\cap(X,T,P)|}{|(X,Y,W)\cup(X,T,P)|} + \frac{|(Y,X,W)\cap(Y,S,V)|}{|(Y,X,W)\cup(Y,S,V)|} + .. + \frac{|(R,Q,S)\cap(R,Q,U)|}{|(R,Q,S)\cup(R,Q,U)|}\right]$$

$$\text{sim\_3}(\text{A},\text{B}) = 0.2$$

$$\text{sim\_3}(\text{A},\text{C}) = \frac{1}{10}\left[\frac{|(X,Y,W)\cap(X,Y,W)|}{|(X,Y,W)\cup(X,Y,W)|} + \frac{|(Y,X,W)\cap(Y,X,W)|}{|(Y,X,W)\cup(Y,X,W)|} + .. + \frac{|(R,Q,S)\cap(R,S,P)|}{|(R,Q,S)\cup(R,S,P)|}\right]$$

$$\text{sim}(\text{A},\text{C})\_3 = 0.7$$

As the similarity index of the techniques A and C is greater than that of A and B, the techniques A and C are more similar than A and B.

3.3 Applying the similarity measure on embedding techniques

We run our program for different values of k and found that for the size of our datasets, a setting of k=5 perform well. Below k=5, the effect of noise becomes more dominant and much above k=5, the overall similarities tend to become more and more similar. We applied the similarity equation on the results obtained in figure 2 in order to compare 10 most common techniques mentioned in section 2. To have a clear idea of similarity indices obtained for each pair of the techniques, we plotted the similarity graph as shown in figure 5. The figure suggests that the similar techniques have high similarity index. For example each technique is similar to itself hence have highest similarity index i.e. 1. Isomap and SVD are slightly similar to each other hence have high similarity index. PCA and CUR are totally different and hence have less similarity index i.e. close to 0.

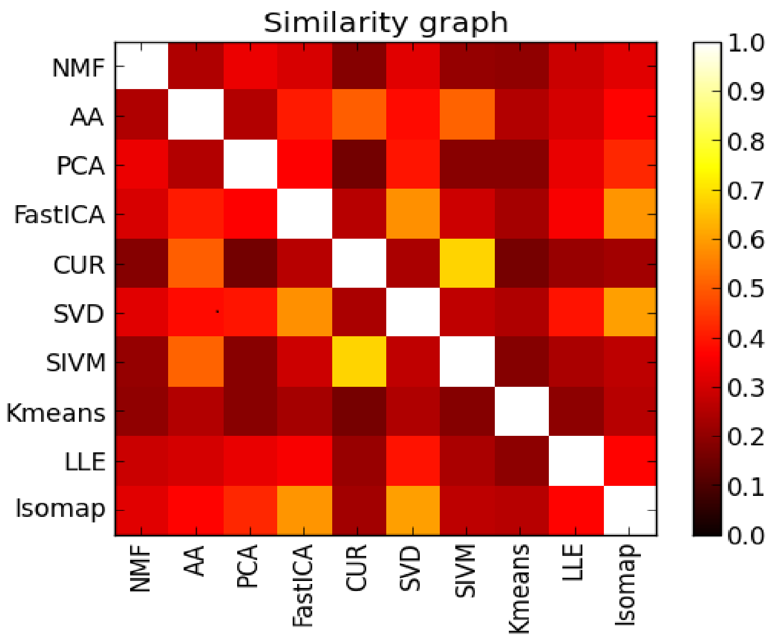


Figure 5: Similarity graph considering one dataset

3.4 Averaged and Normalized Similarity Comparisons

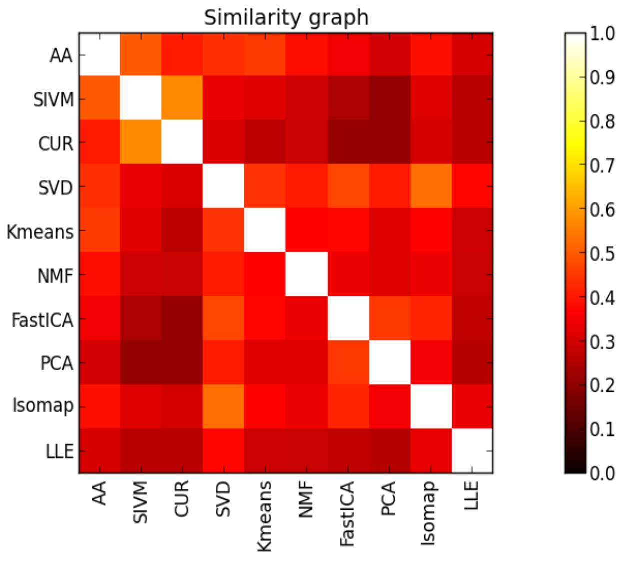
The graph obtained in figure 5 shows the similarities in the embedding techniques calculated with respect to only one dataset i.e. auto93. In order to generalize, we applied our algorithm of similarity measurement on 20 different databases from UCI repository and calculated the average of results obtained using the formula 2.

$$\boxed{\text{avg\_sim\_k}(m1, m2) = \frac{1}{no.\,of\ \ datasets} \sum_{\text{for each dataset}} \sum_{\text{for each }(m1,m2)} sim(m1, m2)}$$

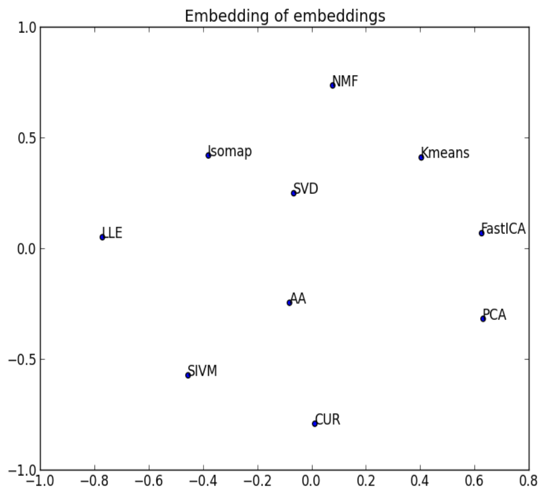
Formula 2: Average Similarity Index of methods m1 and m2

The pairwise similarity graph obtained after averaging the results of 20 databases is shown in figure 6a. Then in concordance to the title of the paper, we embedded the compared embedding techniques in two dimensional map. It is shown in figure 6b. From the figure we can say that the embedding techniques are quiet equi-distant from each other.

We decided to further apply normalization techniques which provide an easy way to compare the values that are measured using different scales. As each embedding technique maps the data using different scales, we decided to normalize an input data before processing it to have better comparison results. We used two common normalization techniques: **Min-Max normalization** and **Mean shift & divide by variance**.



(a)



(b)

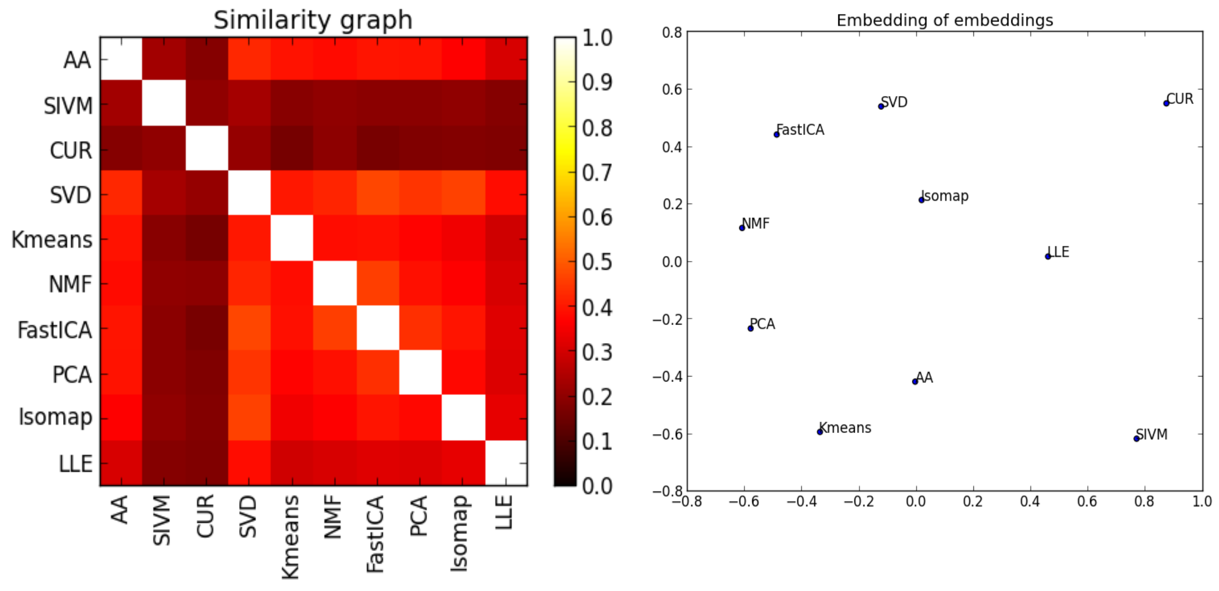
Figure 6: Averaged similarity comparisons

In Min-Max normalization, data is fitted in the scale [0, 1] using the formula 3 which shows the normalization of point A to B.

$$\boxed{B = \frac{A - \min(A)}{\max(A - \min(A))}}$$

Formula 3: Min-max Normalization

The similarity graph and the corresponding embedding of embeddings plot obtained using min-max normalization are as shown in figure 7.



(a) (b)

Figure 7: Similarity comparisons after application of Min-max normalization

From the result obtained in figure 7b, CUR and SiVM seem to be much further apart from the rest. The relevance of this can be found in the similarity graph as shown in figure 7a that the similarity index associated with CUR or SiVM is always close to 0.

Mean shift & divide by variance method uses the formula 4 which calculates the normalization of point A.

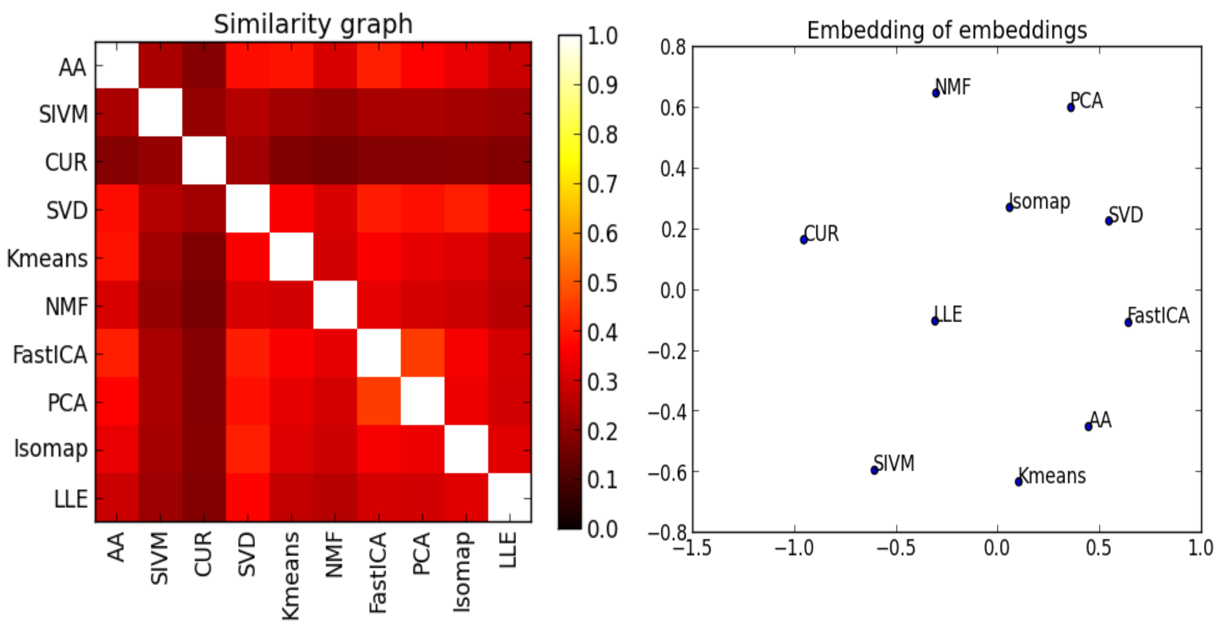
$$\boxed{B = \frac{A - \mathrm{Mean}(A)}{\mathrm{variance}(A)}}$$

Formula 4: Mean shift & divide by variance Normalization

The similarity graph and the corresponding embedding of embeddings plot obtained using Mean shift & divide by variance normalization are as shown in figure 8. From the result obtained in figure 8b, Isomap and SVD seem to be close to each other.

4. Conclusion and Future Work

Embedding techniques in data mining map higher dimensional data to lower dimensions using different scales. Embedding techniques try to keep the underlying structure of the data in the space that the data gets embedded into. Using scale independent k-nearest-neighbor as the basis of similarity measurement, we derived a concrete formula to compute similarity index of two techniques to be compared.



(a) (b)

Figure 8: Similarity comparisons after application of Mean shift & divide by variance normalization

After applying the formula on the set of datasets, we obtained the pairwise similarity graph showing comparisons of the embedding techniques in terms of similarity index. We embedded the embedding techniques in 2D map such that similar techniques are placed close to each other. Better generalization of the comparison, we applied normalization techniques on the input data before processing it. The type of normalization seems to have a significant impact on the comparison of the different techniques.
As an outlook, we would like to investigate the unexpected far behavior of the embedding techniques like PCA & SVD, SiVM & CUR which are idea wise similar to each other but do not show much similarity over the course of our experiments. It could be interesting to find and apply other similarity measures depending on aspects of the data like its convexity, manifold geometry.

5. References

- Pearson, K. (1901), 'On Lines and Planes of Closest Fit to Systems of Points in Space', *Philosophical Magazine* 2, 559-572.
- Hyvärinen, A. & Oja, E. (2000), 'Independent component analysis: algorithms and applications', *Neural Networks* 13 (4-5), 411-430.
- Tenenbaum, J.; Silva, V. & Langford, J. (2000), 'A global geometric framework for nonlinear dimensionality reduction', *Science* 290 (5500), 2319--2323.

- Thurau, C.; Kersting, K. & Bauckhage, C., Yes We Can - Simplex Volume Maximization for Descriptive Web Scale Matrix Factorization. In CIKM, 2010.

- Drineas, P.; Kannan, R. & Mahoney, M. W. (2006), 'Fast Monte Carlo Algorithms for Matrices III: Computing a Compressed Approximate Matrix Decomposition', *SISC* **36** (1), 184-206.

- Yang, D., Ma, Z., & Buja, A. (2011) A sparse SVD method for high-dimensional data arXiv: 1112.2433.

- Cho, Y. & Saul, L. K. (2011), 'Nonnegative Matrix Factorization for Semi-supervised Dimensionality Reduction', *CoRR* abs/1112.3714.

- Ding, C. H. Q. & Li, T. (2007), Adaptive dimension reduction using discriminant analysis and K-means clustering, *in* Zoubin Ghahramani, ed., 'ICML' , ACM, pp. 521-528

- Cutler, A. & Breiman, L. (1994), 'Archetypal Analysis', *Technometrics* Vol. 36, No. 4

- Roweis, S. T. & Saul, L. K. (2000), 'Nonlinear Dimensionality Reduction by Locally Linear Embedding',*Science* 290 (5500), 2323-2326.

- van der Maaten, L. J. P.; Postma, E. O.; and van den Herik, H. J. (2007) ,'Dimensionality reduction: A comparative review', *Technical Report TiCC-TR* 2009-005, Tilburg University, Tilburg, The Netherlands

- Fodor, I. (2002), 'A Survey of Dimension Reduction Techniques' (UCRL-ID-148494) , *Technical report*, US DOE Office of Scientific and Technical Information

- Kakkonen, T.; Myller, N.; Sutinen, E. & Timonen, J. (2008), 'Comparison of Dimension Reduction Methods for Automated Essay Grading', *Educational Technology & Society* 11 (3), 275-288.

- Cao, L. J.; Chua, K. S.; Chong, W. K.; Lee, H. P. & Gu, Q. M. (2003), 'A comparison of PCA, KPCA and ICA for dimensionality reduction in support vector machine.', *Neurocomputing* 55 (1-2) , 321-336 .

- Balachander, T.; Kothari, R. & Cualing, H. (1997), 'An Empirical Comparison of Dimensionality Reduction Techniques for Pattern Classification', *in Wulfram Gerstner; Alain Germond; Martin Hasler & Jean-Daniel Nicoud, ed., 'ICANN' , Springer, pp.* 589-594 .

- M. W. Trosset, 'The formulation and solution of multidimensional scaling problems', *Rice Univ., Houston, TX, Tech.* Rep. TR93-55,1993