A Novel Normalization Technique for Unsupervised Learning in ANN

Goutam Chakraborty and Basabi Chakraborty

Abstract

Unsupervised learning is used to categorize multidimensional data into a number of meaningful classes on the basis of the similarity or correlation between individual samples. In neural network implementation of various unsupervised algorithms such as Principal component analysis (PCA), competitive learning or self-organizing map (SOM), sample vectors are normalized to equal lengths so that similarity could be easily and efficiently obtained by their dot products. In general, sample vectors span the whole multidimensional feature space and existing normalization methods distort the intrinsic patterns present in the sample set. In this work, a novel method of normalization by mapping the samples to a new space of one more dimension has been proposed. The original distribution of the samples in the feature space is shown to be almost preserved in the transformed space. Simple rules are given to map from original space to the normalized space and vice versa.

keywords: Unsupervised learning, Similarity measure, Normalization, Competitive learning, Self Organizing Map

1 Introduction

Unsupervised learning [1] is mainly used to categorize a set of multidimensional input data into a number of meaningful classes. In artificial neural network implementation, the network must discover by itself patterns, features, correlations or categories in the input data without the help of any teacher. In statistical literature this is called clustering.

The objective of unsupervised learning can be multifold. Besides categorization, it is used for principle component analysis(PCA), feature mapping or data encoding. Though well developed statistical methods [2] [3] are available, for large data set they are computationally too heavy. Recently several artificial neural network architectures and learning algorithms for unsupervised analysis e.g. principal Component Analysis [4] [5] for dimensionality reduction, competitive learning [6] for clustering, self-organizing map (SOM) [7] for feature extraction and vector quantization (VQ) [8] for data encoding have been developed.

In artificial neural networks, input to the i^{th} node is $\mathbf{w}.\mathbf{x} = \Sigma_{j=1}^{d} w_{ij}\, x_j$, where $\mathbf{x}=< x_1, x_2, \ldots x_d >$ is the d-dimensional input feature vector and $\mathbf{w}$ is the weight vector for the i^{th} node. Here, w_{ij}s are the different weights connecting the i^{th} node to the different inputs. In supervised learning, this input $\Sigma_{j=1}^{d} w_{ij}\, x_j$ then passes through a non-linear transfer function (threshold, sigmoid, rbf etc.) to produce the output. Thus in general, the *inputs (features) are linearly combined with the weights* before being fed to the transfer function.

This is not true in case of most of the unsupervised learning neural networks. Unsupervised learning algorithms need to find out the *similarity/dissimilarity* between input data and a prototype for categorization by means of some *similarity/dissimilarity* measure. If the *dissimilarity* measure i.e. the distance metric is *city*

block norm or *Tchebyshev maximum* norm, this operation is linear. On the other hand if *Euclidean distance* or *Mahalanobis distance* are used for *dissimilarity* measure, they are non-linear, because they involve higher order operations on the input features. The linear operations are more efficient computationally and are easier for hardware implementation compared to measuring *Euclidean distances*.

It is again true that *Euclidean distances* as *dissimilarity* measure is preferable in many practical applications. When all the sample and weight vectors are normalized, the *dissimilarity* measure by *Euclidean distances* is equivalent to the *similarity* measure by dot product of the input feature vector ($\mathbf{x}$) and the weight vector ($\mathbf{w}$). This is further elaborated in Section 3. This linear operation on input vector is computationally more efficient compared to computing *Euclidean distances*. Its hardware implementation is simple and efficient too. Applications like unsupervised classification of NOAA satellite data [10], or data mining on retail store sales data or credit card data, where data size is enormous and needs millions of comparisons in one epoch of learning, this improvement in efficiency is very important.

Thus, for improving efficiency of unsupervised learning, it is important to normalize the input vectors before feeding to neural network. But the normalization procedure should not distort the inherent pattern present in the samples and thereby destroy some of the information contained in the data. The normalization technique should retain mutual distances between any pair of samples, even after normalization. Existing methods do not comply with that. In the present work we propose a new technique to accomplish such normalization by transforming the data to a new space where all sample vectors become of equal length, and their relative distances are maintained to a significant accuracy.

In the next section we discuss briefly about general normalization techniques. In section 3 we introduce competitive learning network as the representative of unsupervised learning and further elaborate the need for normalization. In section 4, we explain our normalization algorithm and in section 5 the algorithm is analyzed. The final section contains discussion and conclusion.

2 Normalization of Data

Generally normalization is a preprocessing technique to accommodate different ranges and units of values in different dimension of a multidimensional data by translation and scaling, so that the sample values in each dimension have zero mean and unit variance. This or similar preprocessing is in many situations an important step before any analysis of the input samples. For categorization of the preprocessed normalized data one has to partition the sample space into separate groups of similar samples. Now the obvious question is how to measure the similarity between two samples. One possible measure is the Euclidean distance $\|\mathbf{x} - \mathbf{x}'\|$ in the feature space. A more robust way [3] is to introduce a nonmetric similarity function $\sigma(\mathbf{x}, \mathbf{x}')$, which is large when vectors $\mathbf{x}$ and $\mathbf{x}'$ are similar. Thus one possibility is

$$\sigma(\mathbf{x}, \mathbf{x}') = \frac{\mathbf{x}^{\mathbf{t}}.\mathbf{x}'}{\|\mathbf{x}\|\|\mathbf{x}'\|}$$

which is the cosine of the angle between $\mathbf{x}$ and $\mathbf{x}'$. This distance measure is invariant to rotation and dilation. But this similarity measure can not distinguish for example cluster 2 from cluster 3 as shown in Fig. 1.

In fact, the two distance measures, the Euclidean distance and the cosine of the angle, are equivalent when the vectors are normalized in the sense that the length of vectors are all equal.

Most of the artificial neural network algorithms for clustering, which measure the closeness of input and the prototype weight vectors by their dot product i.e. in terms of the cosine of the angle between them, need those vectors to be normalized. Usually the normalization is carried out by dividing every individual vector by its modulus, which destroys some of the spatial information of the samples in the feature space. After such normalization the sample distribution as shown in Fig. 1(a) will become like in Fig. 1(b), where cluster 2 and cluster 3 got merged.

The above mentioned problem, with respect to unsupervised neural network models, is explained more formally in the next section.

3 Unsupervised Neural Network and Normalization Problem

We take the competitive learning network as the representative for unsupervised learning. We shall discuss here one of its several variants, known as the winner take all network, to explain the necessity for normalizing samples.

Let $\mathbf{x}^\mu$ denote the μth d-dimensional sample vector, whose components are $x_1^\mu, x_2^\mu, \ldots, x_d^\mu$. μ is the label for the pattern and has values from 1 to N, where N is the number of patterns.

In simple competitive learning network, as shown in Fig. 2, there is a single layer of output units $O_1, O_2, \ldots, O_p$, each fully connected to the different components of the input pattern $\mathbf{x}^\mu$ via excitatory connections w_{ij}. Output unit O_i represents vector $\mathbf{w_i}$ in the feature space with components $w_{i1}, w_{i2}, \ldots, w_{id}$. Different samples are presented to the input units (in random order) and one of the outputs is declared as winner. The winner is decided as the output unit which is closest to the input pattern i.e.

$$\|\mathbf{w_{i^*}} - \mathbf{x}^\mu\| \leq \|\mathbf{w_i} - \mathbf{x}^\mu\| \quad \text{for all i} = 1 \text{ to p} \tag{1}$$

Here, i^* denotes the winner output. In Eq. (1) Euclidean distance is used as the measure of *dissimilarity* and is desirable in many applications. When all patterns and weight vectors are normalized, it is easy to see that Eq. (1) is equivalent to

$$\mathbf{w_{i^*}}.\mathbf{x}^\mu \geq \mathbf{w_i}.\mathbf{x}^\mu \quad \text{for all i} = 1 \text{ to p} \tag{2}$$

Compared to Eq. (1), Eq. (2) is computationally more efficient and easier for hardware implementation. When the number of samples is large the gain in efficiency is significant. Once the winner output O_{i^*} for input pattern $\mathbf{x}^\mu$ is decided, its weight vector components are trained (modified) according to the following learning rule,

$$\Delta w_{i^*j} = \eta \times (x_j^\mu - w_{i^*j}) \tag{3}$$

where η is the learning rate. Eq. (3) is often called as standard competitive learning rule and works best for normalized inputs [9](chapter 9). After learning for several epochs the weight vectors stabilize to the

locations of cluster centers.

While learning rule is different, Kohonen's SOM algorithm determines the winner in the same fashion and the efficiency gain is similar when pre-normalized data is used.

4 Proposed Normalization Algorithm

In this section we propose a normalization algorithm that preserves the relative distances of the original samples. For ease of explanation, we shall first introduce the concept for one and two-dimensional data and then extend it to the general d-dimensional case. We shall show that the algorithm is a one-to-one mapping and the relative distance of any pair of samples is preserved to a high order of accuracy in the mapped space.

4.1 The proposed plan

The basic idea of this normalization is to map elements of the d-dimensional multivariate sample vector set to the surface of a unit hypersphere of $d+1$-dimension. All samples are now of unit length and therefore normalized. Once the sample vectors are normalized, the unsupervised NN algorithms can work efficiently on them. The prototypes obtained in the normalized space could then be reverse mapped to the original feature space. The reverse mapping is unnecessary when only categorization of patterns is needed.

Let us start with the original set of d-dimensional sample vectors $\mathbf{x}_{org} = \{x^{\mu}_{i_{org}} | i = 1, \cdots, d \text{ and } \mu = 1, \cdots, N\}$ of N samples.

The steps involved for normalization shown in Fig. 3 are explained below:

Step 1: Find the length of the longest component of the sample vectors i.e. the longest of all x^{μ}_{i} for $\mu = 1$ to N, and $i = 1$ to d and let it be l. For one-dimensional case it is the length of the longest sample.

Step 2: Divide individual components of all the sample vectors by l. Thus the length of any component of any sample vector will be ≤ 1. This uniform scaling in all dimensions will not introduce any distortion in patterns existing in the original sample set. The sample set in this compressed form is denoted as $\mathbf{x}^{\mu}_{com}$. For any d-dimensional vector, $\mathbf{x}^{\mu}_{com}$ is obtained by $x^{\mu}_{i_{com}} = x^{\mu}_{i_{org}}/l$ for all μ and i.

Step 3: In the final step normalize the compressed sample vectors by mapping them to the surface of a unit hypersphere to get normalized sample vectors $\mathbf{x}_{nor}$. The detail follows:

- For One Dimensional Samples:

  Here we map the scaled sample points on a two dimensional plane along the circumference of a circle of radius 1, as shown in Fig. 4. The $x^{\mu}_{1_{com}}$ will be mapped such that it subtends an angle of $x^{\mu}_{1_{com}}$ to the X-axis. Thus the polar coordinates of the point $x^{\mu}_{1_{com}}$ will become $(1, x^{\mu}_{1_{com}})$. The new coordinates (components) of the normalized sample vectors will be

$$\begin{aligned} x^{\mu}_{1_{nor}} &= cos\, x^{\mu}_{1_{com}} \\ x^{\mu}_{2_{nor}} &= sin\, x^{\mu}_{1_{com}} \end{aligned} \tag{4}$$

– For Two Dimensional Samples:

For normalization, two-dimensional sample vectors are to be mapped to the surface of a unit sphere in 3-dimension as shown in Fig. 5. Here, a sample point $(x^{\mu}_{1_{com}}, x^{\mu}_{2_{com}})$ is mapped to $(1, x^{\mu}_{1_{com}}, x^{\mu}_{2_{com}})$ in polar coordinates. The point is shown as P in Fig. 5. Thus the new coordinates in the normalized space is

$$\begin{array}{rcl} x^{\mu}_{1_{nor}} & = & cos\ x^{\mu}_{2_{com}}\ cos\ x^{\mu}_{1_{com}} \\ x^{\mu}_{2_{nor}} & = & cos\ x^{\mu}_{2_{com}}\ sin\ x^{\mu}_{1_{com}} \\ x^{\mu}_{3_{nor}} & = & sin\ x^{\mu}_{2_{com}} \end{array} \tag{5}$$

– For general d-Dimensional Samples:

Extending the same mapping principle, a d-dimensional compressed sample vector, $(x^{\mu}_{1_{com}}, x^{\mu}_{2_{com}}, \ldots x^{\mu}_{d_{com}})$ will be mapped to the surface of $d+1$-dimensional unit hypersphere, and the components of the normalized vector are:

$$\begin{array}{rcl} x^{\mu}_{1_{nor}} & = & cos\ x^{\mu}_{d_{com}}\ cos\ x^{\mu}_{(d-1)_{com}} \ldots cos\ x^{\mu}_{3_{com}}\ cos\ x^{\mu}_{2_{com}}\ cos\ x^{\mu}_{1_{com}} \\ x^{\mu}_{2_{nor}} & = & cos\ x^{\mu}_{d_{com}}\ cos\ x^{\mu}_{(d-1)_{com}} \ldots cos\ x^{\mu}_{3_{com}}\ cos\ x^{\mu}_{2_{com}}\ sin\ x^{\mu}_{1_{com}} \\ x^{\mu}_{3_{nor}} & = & cos\ x^{\mu}_{d_{com}}\ cos\ x^{\mu}_{(d-1)_{com}} \ldots cos\ x^{\mu}_{3_{com}}\ sin\ x^{\mu}_{2_{com}} \\ \vdots & & \\ x^{\mu}_{(i+1)_{nor}} & = & cos\ x^{\mu}_{d_{com}}\ cos\ x^{\mu}_{(d-1)_{com}} \ldots cos\ x^{\mu}_{(i+1)_{com}}\ sin\ x^{\mu}_{i_{com}} \\ x^{\mu}_{(i+2)_{nor}} & = & cos\ x^{\mu}_{d_{com}}\ cos\ x^{\mu}_{(d-1)_{com}} \ldots cos\ x^{\mu}_{(i+2)_{com}}\ sin\ x^{\mu}_{(i+1)_{com}} \\ \vdots & & \\ x^{\mu}_{d_{nor}} & = & cos\ x^{\mu}_{d_{com}}\ sin\ x^{\mu}_{(d-1)_{com}} \\ x^{\mu}_{(d+1)_{nor}} & = & sin\ x^{\mu}_{d_{com}} \end{array} \tag{6}$$

All these new $(d+1)$-dimensional vectors will be of unit length and therefore normalized and ready for further processing with unsupervised NN algorithms .

4.2 Reverse Mapping

We have shown above how to normalize a d-dimensional vector by mapping it to the surface of a unit hypersphere of $(d+1)$-dimension. Once the sample vectors are normalized, unsupervised algorithms are used to find cluster centers (prototype vectors). If finding those prototype vectors is the motivation for unsupervised learning, one would probably need to know their values in the original feature space. If only clustering of the data is the aim, all computations could be confined in the normalized space. When necessary the reverse mapping to the original feature space could easily be done as follows.

x^{μ}_{d} is obtained using the following expression obtained from the last line of Eq. (6).

$$x^{\mu}_{d_{com}} = sin^{-1}\ x^{\mu}_{(d+1)_{nor}} \tag{7}$$

Any other $x^{\mu}_{i_{com}}$ could be expressed in terms of $x^{\mu}_{(i+2)_{nor}}$ and $x^{\mu}_{(i+1)_{nor}}$ if we consider the following forward mapping expressions in Eq. (6)

$$
\begin{aligned}
x^{\mu}_{(i+1)_{nor}} &= cos\ x^{\mu}_{d_{com}}\ cos\ x^{\mu}_{(d-1)_{com}} \ldots cos\ x^{\mu}_{(i+2)_{com}}\ cos\ x^{\mu}_{(i+1)_{com}}\ sin\ x^{\mu}_{i_{com}} \\
x^{\mu}_{(i+2)_{nor}} &= cos\ x^{\mu}_{d_{com}}\ cos\ x^{\mu}_{(d-1)_{com}} \ldots cos\ x^{\mu}_{(i+2)_{com}}\ sin\ x^{\mu}_{(i+1)_{com}}
\end{aligned} \tag{8}
$$

whence we get,

$$
x^{\mu}_{i_{com}} = sin^{-1}\left(\frac{x^{\mu}_{(i+1)_{nor}}}{x^{\mu}_{(i+2)_{nor}}\ cot\ x^{\mu}_{(i+1)_{com}}}\right) \tag{9}
$$

5 Analysis of the Algorithm

We have shown how to normalize multidimensional sample vectors and then reverse map to the original space. Our proposed algorithm also satisfies following three useful properties for being used as a practical normalization method.

1. All vectors after normalization are of equal length.
2. Mapping from d to $d+1$-dimensional is one-to-one.
3. The mutual distance between any pair of sample vectors is preserved.

The proof of item 1 trivially follows from Eq. (6). The proof for item 2 and 3 follows.

5.1 Proof of one-to-one mapping

In the compressed space the value of different d-components are ≤ 1. Now when they are mapped to the normalized $d+1$-dimensional space, it is mapped simply to a point where the different components are measures of angles (in circular measure). A point $(x^{\mu}_1, x^{\mu}_2, \ldots, x^{\mu}_d)$ becomes $(1, x^{\mu}_1, x^{\mu}_2, \ldots, x^{\mu}_d)$ (in polar coordinates) in the normalized space. As all the components are ≤ 1 i.e. none of the angles are $\geq 2\pi$, it is trivial that this mapping will be one-to-one. The same argument holds for the reverse mapping.

5.2 Preservation of Mutual distance

We already mentioned that the mapping algorithm preserves the mutual distance between any pair of vectors. For simplicity's sake we consider the 2-dimensional case where a point (x^{μ}_1, x^{μ}_2) is mapped to $(cos\ x^{\mu}_2\ cos\ x^{\mu}_1,\ cos\ x^{\mu}_2\ sin\ x^{\mu}_1,\ sin\ x^{\mu}_2)$. Without loss of generality, we can consider $\mu = 1$ and $\mu = 2$ as any two points. The square of the distance between the two points in normalized space is

$(cos\ x^1_2\ cos\ x^1_1 - \ cos\ x^2_2\ cos\ x^2_1)^2 + (cos\ x^1_2\ sin\ x^1_1 - \ cos\ x^2_2\ sin\ x^2_1)^2 + (sin\ x^1_2 - sin\ x^2_2)^2$
$= 2 - 2\ cos\ x^1_2\ cos\ x^2_2\ cos\ (x^1_1 - x^2_1) - 2\ sin\ x^1_2\ sin\ x^2_2$
$= 2 - 2\ cos\ (x^1_2 - x^2_2) + (x^1_1 - x^2_1)^2$ expanding $\cos(\mathrm{x}^1_1 - \mathrm{x}^2_1)$, neglecting $(\mathrm{x}^1_1 - \mathrm{x}^2_1)^4$ and higher order terms
$= (x^1_2 - x^2_2)^2 + (x^1_1 - x^2_1)^2$ neglecting $(\mathrm{x}^1_2 - \mathrm{x}^2_2)^4$ and higher order terms

Thus we see that the distance is preserved to a high order of accuracy. As the components values are < 1 and therefore their differences too, the fourth and higher order terms are negligibly small and the mutual distance is nearly preserved in the normalized space. Numerical simulations are done with several pairs of points at various regions of the original space, and then they are mapped to normalized space. All calculations were

rounded to six decimal digits. It is observed that distance between a pair of points as close as 0.000001 are maintained exactly, whereas greater distances are slightly changed in their 5th. or 6th. decimal places.

6 Conclusion

Unsupervised Neural Network algorithms are very strong tools for clustering or grouping multivariate input data. With the increasing computational power and memory of the machines, it is now possible to collect and store enormous data sets from sources like retail stores, credit card company, satellites etc. Unsupervised neural network models are suitable to discover patterns from these vast data. For unsupervised clustering we need to measure *similarity* between samples and prototypes and this comparisons could be done more efficiently when sample data are normalized. Here we proposed a novel way to do this normalization without distorting the intrinsic pattern. We have shown that the algorithm preserves the mutual distance between any two pair of samples to a high degree of accuracy. Though the algorithm is computationally heavy, it has to be done only once, whereas the *similarity* comparisons has to be done over several epochs facilitating ultimate gain in efficiency.

An increased dimension is often a curse. But the *curse of dimensionality* is a curse only when the data size is small. For the above mentioned applications that is not true.

7 Acknowledgement

The author would like to thank the editor and the reviewers for their useful comments to improve the quality of presentation.

References

[1] H. B. Barlow, "Unsupervised Learning", Neural Computation, vol. 1, pp. 295-311, 1989.

[2] A. K. Jain, R. C. Dubes, "Algorithms for Clustering Data", Prentice Hall, 1988.

[3] R. O. Duda and P. E. Hart, "Pattern Classification and Scene Analysis", Wiley, New York, 1973.

[4] E. Oja, "Neural Networks, Principal Components, and Subspaces", International Journal of Neural Systems, vol. 1, pp. 61-68, 1989.

[5] T. D. Sanger, "Optimal Unsupervised Learning in a single-Layer Linear Feedforward Neural Network", Neural Networks, vol. 2, pp. 459-473, 1989.

[6] D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, "Parallel and Distributed Processing", Cambridge, MIT Press. 1986.

[7] T. Kohonen, "Self Organizing Maps", Springer, Berlin, 1995.

[8] J. Naylor and K. P. Li, "Analysis of a Neural Network Algorithm for Vector Quantization of Speech Parameters" Neural Networks Supplement, vol. 1, pp. 310-318, 1988.

[9] J. Hertz, A. Krogh, and R. G. Palmer, "Introduction to the Theory of Neural Computation", Addison-Wesley Publishing Company, 1991.

[10] G. Chakraborty, J. Kudoh, and N. Shiratori and S. Noguchi, "Classification of the NOAA Satellite Image Data by Unsupervised Neural Network" Trans. of the Society of Instrument and Control Engineers, Vol. 29, No. 3, pp. 281-287, 1993.

List of Figures

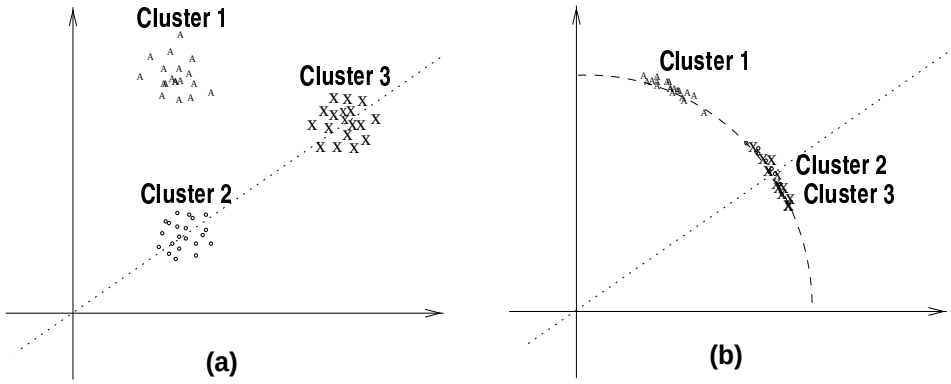

Figure 1: Merging of clusters due to Normalization

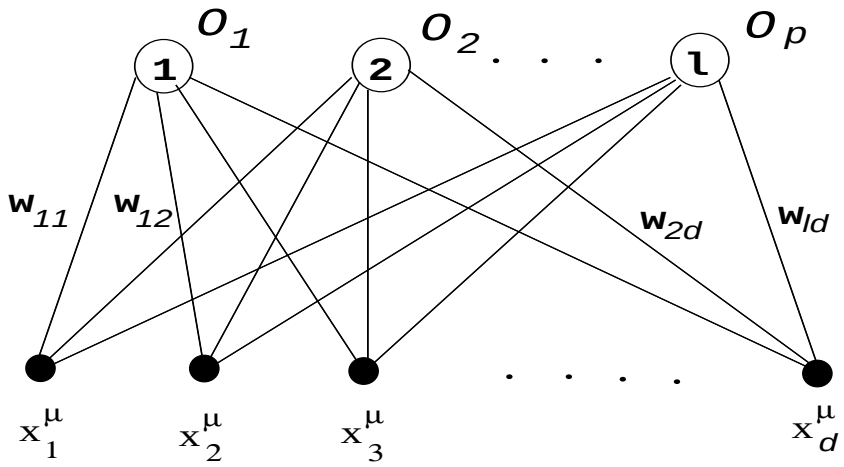

Figure 2: Simple competitive learning Neural Network

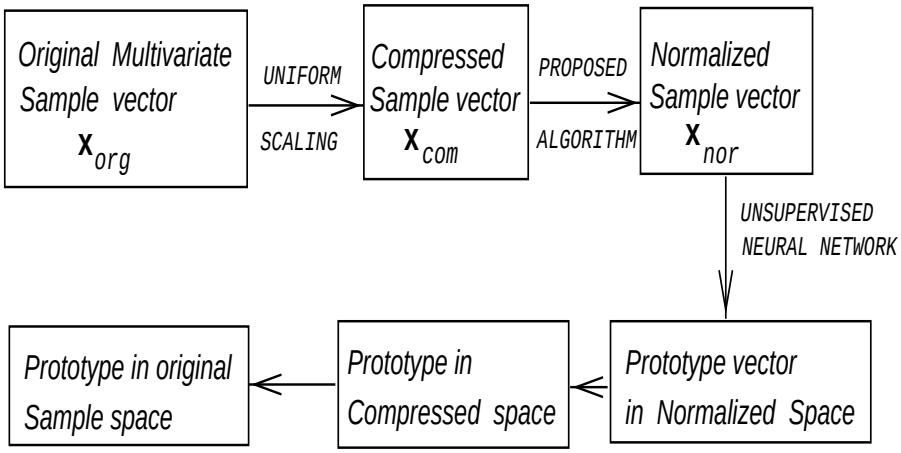

Figure 3: The Normalization Scheme

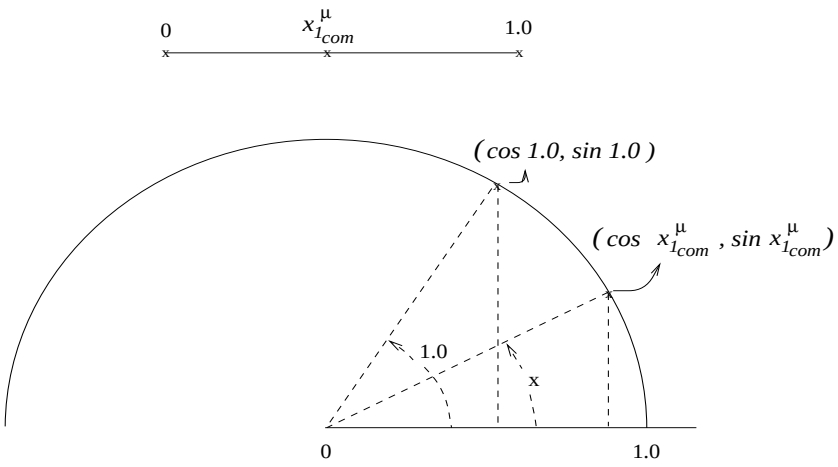

Figure 4: Mapping of 1-dim data for normalization

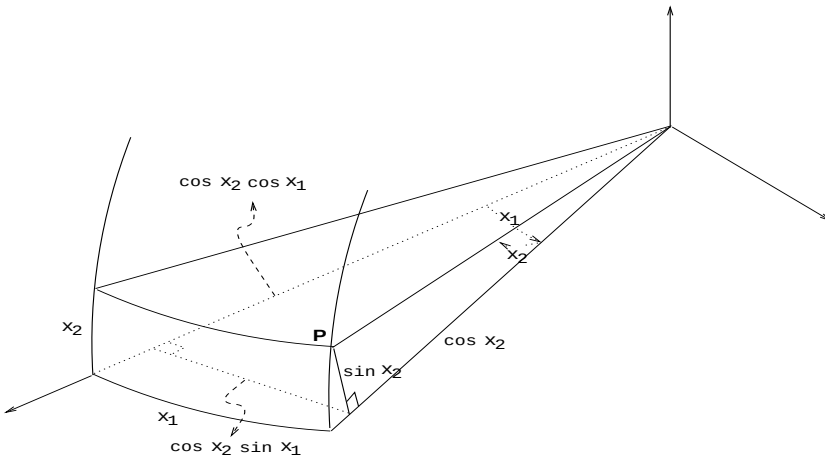

Figure 5: Mapping of 2-dim data for normalization. For clarity of figure, sample points $x^{\mu}_{1_{com}}$ and $x^{\mu}_{2_{com}}$ are simply written as x_1 and x_2