LearnMet: Learning Domain-Specific Distance Metrics for Plots of Scientific Functions *

Aparna Varde[1,2], Elke Rundensteiner[1], Carolina Ruiz[1], Mohammed Maniruzzaman[2], and Richard Sisson Jr.[2]

[1] Department of Computer Science
Worcester Polytechnic Institute, Worcester, MA 01609
{aparna, rundenst, ruiz}@cs.wpi.edu
[2] Center for Heat Treating Excellence
Metal Processing Institute, Worcester, MA 01609
{maniruzz, sisson}@wpi.edu

Abstract. Scientific experimental results are often depicted as plots of functions to aid their visual analysis and comparison. In computationally comparing these plots using techniques such as similarity search and clustering, the notion of similarity is typically distance. However, it is seldom known which distance metric(s) best preserve(s) semantics in the respective domain. It is thus desirable to learn such domain-specific distance metrics for the comparison of plots. This paper describes a technique called LearnMet proposed to learn such metrics. The input to LearnMet is a training set with actual clusters of plots. These are iteratively compared with clusters over the same plots predicted using an arbitrary but fixed clustering algorithm. Using a guessed initial metric for clustering, adjustments are made to the metric in each epoch based on the error between the predicted and actual clusters until the error is minimal or below a given threshold. The metric giving the lowest error is output as the learned metric. The proposed LearnMet technique and its enhancements are discussed in detail in this paper. The primary application of LearnMet is clustering plots in the Heat Treating domain. Hence it is rigorously evaluated using Heat Treating data. Given distinct test sets for evaluation, clusters of plots predicted using the learned metrics are compared with given actual clusters over the same plots. The extent to which the predicted and actual clusters match each other denotes the accuracy of the learned metrics.

Keywords: Distance Metrics, Clustering, Domain Semantics, Parameter Learning, Curve Comparison

* This work is supported by the Center for Heat Treating Excellence (CHTE) and by the Department of Energy - Industrial Technology Program (DOE-ITP) Award Number DE-FC-07-01ID14197.

1 Introduction

Motivation. In domains such as Materials Science the results of experiments are often depicted as graphical plots of scientific functions. This facilitates their visual analysis by the human eye as well as by computational techniques. In our earlier work [12, 13], we also refer to such plots as graphs. Computational analysis of these graphical plots involves several processes such as comparison between plots [9], similarity search [7] and clustering [12]. The inferences drawn from such analysis in a given domain are used in many applications such as process parameter selection [9] and computational estimation [12]. Hence the analysis should be such that it closely reflects the notion of correctness in the given domain so that the inferences are precise. In order to achieve this it is important to capture the semantics in the scientific plots. Processes such as clustering involve a notion of similarity for comparison which typically refers to the distance between the plots [4]. Although several distance metrics exist in the literature [4, 7, 11], it is often not known a priori which of these metrics best preserves domain semantics. Often none of the metrics in the literature may alone be suitable. We may need to work with a specially-tuned metric that is a combination of several metrics. Hence there is a need for distance metric learning in such plots. State-of-the-art machine learning techniques such as neural networks [1], genetic algorithms [3], and others [5, 16] are found to be either inappropriate or inaccurate in this context [14]. This inspires the development of a technique to learn semantics-preserving metrics for plots.

Proposed Approach. We propose a technique called LearnMet [13] to learn domain-specific distance metrics for plots of scientific functions. The input to LearnMet is a training set with actual clusters of such plots provided by domain experts. The steps of LearnMet are: (1) guess an initial metric D as a weighted sum of metrics applicable to the domain; (2) use that metric D for clustering with an arbitrary but fixed clustering algorithm to get predicted clusters; (3) evaluate clustering accuracy by comparing predicted and actual clusters to obtain the error between them; (4) adjust the metric D based on the error, and re-execute clustering and evaluation until error is minimal or below a threshold; (5) output the metric D giving lowest error as the learned metric.

Experimental Evaluation. LearnMet is thoroughly evaluated in the domain of Heat Treating of Materials [2] that motivated this research. Its main application is to learn distance metrics for clustering plots called heat transfer curves. Experts provide actual clusters over distinct test sets of heat transfer curves not used for training. These are compared with clusters predicted by any fixed clustering algorithm over the same curves using the learned metrics. The extent to which the predicted and actual clusters match each other measures the clustering accuracy of the respective metric. Comparative evaluation is also done using the default Euclidean distance as the notion of similarity in clustering.

Approach Enhancement. Evaluation of the basic LearnMet approach is found to yield metrics with higher accuracy in clustering compared to the default Euclidean distance [13]. However, there is scope for further enhancement in terms of the the clustering accuracy of the learned metrics. Also important are

the efficiency of the learning process, and the simplicity of the learned metrics which have not been addressed in the basic approach. Learning efficiency refers mainly to training time. The simplicity of the hypothesis is measured by the number and type of individual metrics in the learned metric D. In this paper we propose three approaches for enhancement. The first one focuses on methods to avoid overfitting for learning a more generic hypothesis. This aims to improve clustering accuracy. The second one deals with enhancing both accuracy and efficiency by exploiting fundamental domain knowledge in initial metric selection and weight adjustment. This is likely to provide faster convergence and learn metrics closer to the notion of correctness in the domain. The third approach involves using Occam's Razor [15] to learn metrics preferring simpler to complex ones taking into account the goal of accuracy.

Paper Organization. Section 2 of this paper gives a background on graphical plots and distance metrics. Section 3 presents the details of LearnMet. Section 4 explains its refinements. Section 5 summarizes experimental evaluation. Section 6 outlines the related work. Section 7 gives the conclusions.

2 Background

2.1 Graphical Plots in Heat Treating

Figure 1 shows a plot in the Heat Treating domain called a heat transfer curve that depicts the reaction of a part to a rapid cooling process called quenching [2]. The curve is a plot of heat transfer coefficients h versus temperature T of a material during quenching. Heat transfer coefficients represent the heat extraction capacity in the process. Certain regions on this curve correspond to physical processes in the domain. The Boiling Point BP marks the temperature of the part being reduced to the boiling point of the cooling medium. The Leidenfrost Point LF denotes the breaking of a vapor blanket resulting in rapid cooling. The maximum heat transfer MAX achieved in a quenching process serves to separate the curves, and hence the corresponding experiments, statistically into different categories. The mean heat transfer achieved in the process is also a statistical distinguishing factor. Other regions on the curve are MIN, the point of minimum heat transfer, and SC, the point where slow cooling ends [2].

2.2 Distance Metrics for Plots

We describe the distance metrics relevant to our problem with respect to n-dimensional objects $A(A_1, A_2 \ldots A_n)$ and $B(B_1, B_2 \ldots B_n)$.

Position-based Distances. They refer to distances based on the absolute position of the objects [4]. In the context of the given problem, the position-based distance is *Euclidean Distance*, i.e., the *as-the-crow-flies* distance between objects calculated as $D_{Euclidean}(A, B) = \sqrt{\Sigma_i^n (A_i - B_i)^2}$

Statistical Distances. These refer to distances based on statistical observations in the objects [10]. Examples of statistical distances are the *Mean Distance*, i.e., distance between mean values of the objects, given as $D_{Mean}(A, B) =$

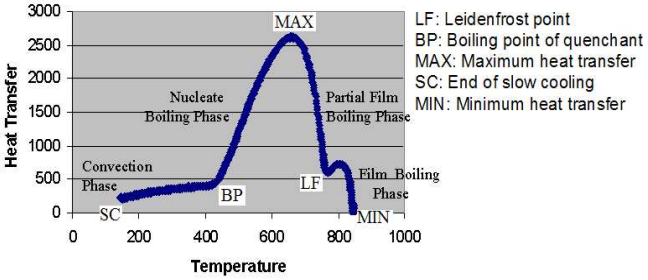

Fig. 1. Heat Transfer Curve

$|Mean(A) - Mean(B)|$, the *Maximum Distance*, i.e., distance between maximum values of the objects, given as $D_{Max}(A, B) = |Max(A) - Max(B)|$ and the *Minimum Distance*, i.e., the distance between minimum values of the objects, given as $D_{Min}(A, B) = |Min(A) - Min(B)|$.

Critical Distances. In addition to the distance metrics reviewed above, we introduce the concept of critical distances for graphical plots [13].

Critical Distance Metric: Given two graphical plots A and B, a critical distance metric represents the distance between critical regions of A and B where a critical region depicts the occurrence of a significant physical phenomenon. Each such metric is calculated in a domain-specific manner.

Examples of critical distances are given here with reference to Figure 1. The *Leidenfrost distance* is distance between the Leidenfrost points [2] on two heat transfer curves, given as $D_{LF}(A, B) = \sqrt{(A_{TLF} - B_{TLF})^2 + (A_{hLF} - B_{hLF})^2}$ where TLF is the temperature at Leidenfrost Point and hLF is the heat transfer coefficient at that point. Another critical distance is the *Boiling Point distance* given as $D_{BP}(A, B) = \sqrt{(A_{TBP} - B_{TBP})^2 + (A_{hBP} - B_{hBP})^2}$ where TBP and hBP are the temperature and heat transfer coefficient at that point respectively. Likewise, given that TSC and hSC are temperature and heat transfer coefficient at Slow Cooling respectively, the *Slow Cooling* distance is given as $D_{SC}(A, B) = \sqrt{(A_{TSC} - B_{TSC})^2 + (A_{hSC} - B_{hSC})^2}$.

3 Proposed Approach: LearnMet

3.1 Overview of LearnMet

In order to describe the learning strategy, a LearnMet metric is first defined.

A *LearnMet Distance Metric* D is a weighted sum of components, where each component can be a position-based, a statistical, or a critical distance. The weight of each component is a numerical value giving its relative importance in the domain. Thus a LearnMet metric is of the form $D = \Sigma_{i=1}^{m} w_i Dc_i$ where each Dc_i is a component, w_i is its weight, and m is the number of components.

It is important for D to be a metric for reasons such as those listed below.

- Clustering algorithms requiring the notion of similarity to be a distance metric can be used [6].
- Indexing structures such as $B+$ trees [4] can be applied.
- Pruning in similarity search can be performed using triangle inequality [15].

Sufficient conditions for D to be a metric are stated in Theorem 1.

Theorem 1: If each Dc_i is a distance metric and each $w_i >= 0$ then $D = \Sigma_{i=1}^{m} w_i Dc_i$ is a distance metric, i.e., it satisfies metric properties.

For convenience the distance metric properties are listed here as follows [4].

1. Distance is non-negative.
2. Distance of an object to itself is zero.
3. Distance is commutative, thus $Distance(P, Q) = Distance(Q, P)$ for any objects P, Q in n-dimensional space.
4. Distance satisfies triangle inequality, i.e., if 3 objects P, Q and R form a triangle in n-dimensional space, then sum of any two sides is greater than the third, e.g., $Distance(P, Q) + Distance(Q, R) > Distance(P, R)$.

The proof of this theorem is straightforward and can be found in [14]. In our targeted applications, conditions in Theorem 1 are satisfied. Since the plots have interval-scaled variables, distances applicable to them are metrics [4], this is sufficient to say that each is component a metric. Also, we consider only non-negative weights since negative weights do not have a semantic interepretation in our targeted applications [14].

The LearnMet technique is summarized in the flowhart Figure 2.

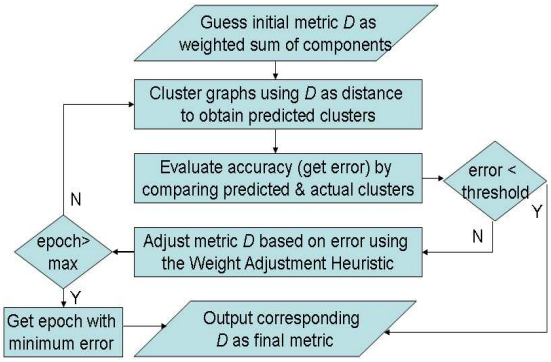

Fig. 2. Flowchart on LearnMet

3.2 Details of Technique

1. Initial Metric Step. Experts are asked to identify components (i.e., distance metrics) applicable to the plots to serve as building blocks for learning

a new metric. If experts have a subjective notion about relative importance of components, this is used to assign initial weights as follows [13].

Initial Weight Heuristic: Assign initial weights to components in the Learn-Met metric based on relative importance of components in the domain.

If this relative importance is unknown then random weights are assigned to all components. A special case of this is assigning equal weights to all components. Weights are typically assigned on a scale of 0 to 10.

2. Clustering Step. Actual clusters over a training set of plots are provided by domain experts. An arbitrary but fixed clustering algorithm, e.g., k-means [8] is selected. Using $D = \Sigma_{i=1}^{m} w_i Dc_i$ as the notion of distance, k clusters are constructed using the selected algorithm, where k is the number of actual clusters in the training set. The clusters obtained from the algorithm using the metric D are called the predicted clusters.

3. Cluster Evaluation Step. Cluster evaluation involves comparing the predicted and actual clusters over the training set with each other. An example of predicted and actual clusters of plots is shown in Figure 3.

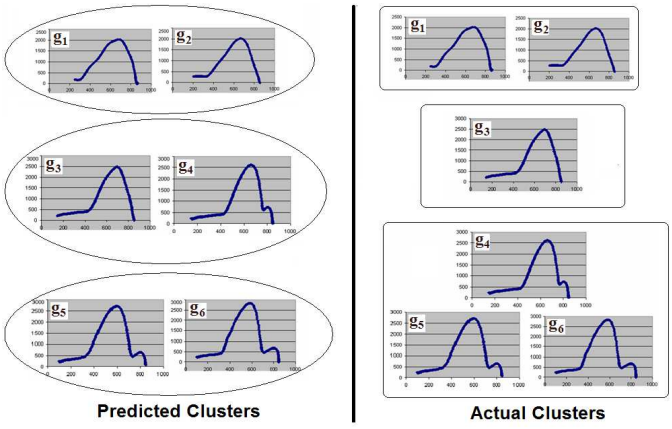

Fig. 3. Predicted and Actual Clusters

Ideally, predicted and actual clusters should match perfectly. Any difference between them is considered an error. To compute error, we consider pairs of graphical plots and introduce a notion of correctness [14].

Notion of Correctness: Given a pair of graphical plots A and B, we say that:

– (A, B) is a True Positive (TP) pair if A and B are in the same predicted cluster and in the same actual cluster.

- (A, B) is a True Negative (TN) pair if A and B are in different predicted clusters and in different actual clusters.
- (A, B) is a False Positive (FP) pair if A and B are in the same predicted cluster but in different actual clusters.
- (A, B) is a False Negative (FN) pair if A and B are in different predicted clusters but in the same actual cluster.

Figure 3 includes examples of each of these kinds of pairs. The pair (g_1, g_2) is a true positive; (g_2, g_3) is a true negative pair; (g_3, g_4) is a false positive pair; and (g_4, g_6) is a false negative pair. The error measure of interest to us is failure rate which is explained below [15].

Success and Failure Rates: Let TP, TN, FP and FN denote the number of true positive, true negative, false positive and false negative pairs respectively. Also let SR be the Success Rate and $FR = (1 - SR)$ be the Failure Rate.

Then, $SR = \frac{TP+TN}{TP+TN+FP+FN}$ and thus, $FR = \frac{FP+FN}{TP+TN+FP+FN}$

In our context, false positives and false negatives are equally undesirable. Hence, our definition of failure rate weighs them equally. Given a number G of graphical plots in the training set, the total number of pairs of plots P is given by *G choose 2*, i.e., $C_2^G = P = \frac{G!}{2!(G-2)!}$ [10]. Thus, for 25 plots there are 300 pairs, for 50 graphs, 1225 pairs, etc. We define an epoch in LearnMet as one run of all its steps, i.e., a complete training cycle.

Overfitting: To avoid overfitting in LearnMet, we use an approach analogous to incremental gradient descent [1]. Instead of using all pairs of plots for evaluation, a subset of pairs is used called *ppe* or pairs per epoch. In each epoch, a randomly selected subset of pairs is used for evaluation and weight adjustment. Thus there is enough randomization in every epoch. If $ppe = 25$, then we have a total of $C_2^{300} = \frac{300!}{25!275!}$ distinct pairs for learning [10]. Thus in each epoch 25 randomly selected pairs can be used. This still gives a large number of epochs with distinct pairs for learning. This incremental approach reduces the time complexity of the algorithm and helps avoid overfitting. Determining good *ppe* values is an enhancement issue and will be discussed in Section 4. Also in LearnMet, the random seed is altered in the clustering algorithm in different epochs as an additional method to avoid overfitting.

Error Threshold: Ideally, the error i.e., in an epoch should be zero. However, in practice a domain-specific error threshold t is used, where t is the extent of error allowed per epoch in the domain, error being measured by failure rate.

If the error is below threshold then the final metric is output as explained in step 5. However, if the error is not below threshold in a given epoch, then the metric is adjusted based on this error.

4. Weight Adjustment Step. In order to give the details of weight adjustment the following terminology on distances is explained since the cause of error can be traced to certain distances between pairs of plots.

Distance between a Pair of Graphical Plots: The distance $D(g_a, g_b)$ between a pair of plots g_a and g_b is the weighted sum of components in the plots using metric D. Thus, $D(g_a, g_b) = w_1 Dc_1(g_a, g_b) + \ldots + w_m Dc_m(g_a, g_b)$.

(DFN) and (DFP) Distances: The distances DFN and DFP are defined as the average distance using the metric D of the false negative pairs and of the false positive pairs respectively. These are calculated as:

$DFN = (1/FN)\Sigma_{j=1}^{FN} D(g_a, g_b)$ where (g_a, g_b) denotes each FN pair.

$DFP = (1/FP)\Sigma_{j=1}^{FP} D(g_a, g_b)$ where (g_a, g_b) denotes each FP pair.

Given this notion of distances refer to the example in Figure 4. Consider first the false negative pairs, e.g., (g_4, g_5) and (g_4, g_6). These pairs are in the same actual cluster. However they are predicted to be in different clusters. Since predicted clusters are obtained with the metric D the cause of the error is that the (average) distance DFN between these pairs with the given metric is greater than it should be. Hence these pairs are incorrectly pushed far apart to be in different predicted clusters although they in reality they should have been closely placed in the same actual cluster. Conversely, for false positive pairs in different actual but same predicted clusters, e.g., (g_3, g_4) in Figure 4, the cause of the error is that the (average) distance DFP is smaller than it should be. These distances are now used in altering weights as follows.

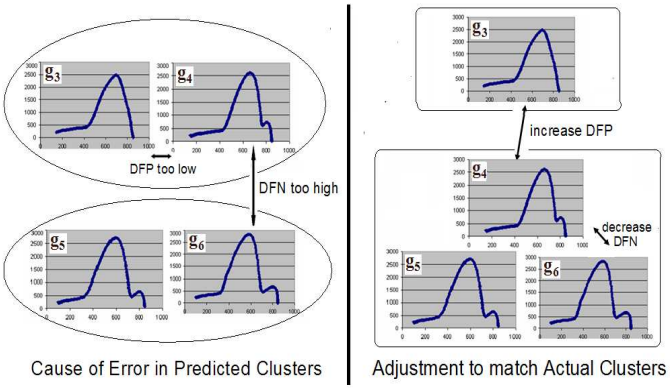

Fig. 4. Distances used in Weight Adjustment

Consider the error due to the false negative pairs. To reduce this error it is desirable to decrease the distance DFN. In order to reduce DFN the weights of one or more components in the metric used to calculate the distance in the present epoch is decreased. For this we propose the FN Heuristic.

FN Heuristic: Decrease the weights of components in metric D in proportion to their contributions to distance DFN. That is, for each component:

New weight $w_i^{'} = w_i - \frac{DFNc_i}{DFN}$

where $DFNc_i = DFN$ for component Dc_i alone

Conversely, to reduce error due to the FP pairs we increase DFP by increasing the weights of one or more components in metric D using the FP Heuristic.

FP Heuristic: Increase the weights of components in metric D in proportion to their contributions to distance DFP. That is, for each component:

New weight $w_i^{''} = w_i + \frac{DFPc_i}{DFP}$
where $DFPc_i = DFP$ for component Dc_i alone
Combining these two we get the weight adjustment heuristic below.

Weight Adjustment Heuristic: For each component Dc_i, its new weight is $w_i^{'''} = max(0, w_i - \frac{DFNc_i}{DFN} + \frac{DFPc_i}{DFP})$.

Thus, the new metric is: $D^{''} = \Sigma_{i=1}^{m} w_i^{'''} Dc_i$

The new metric obtained after weight adjustments is likely to minimize error due to both false positive and false negative pairs. If the weight of a component becomes negative it is converted to zero as we consider only non-negative weights [14]. Clustering in the next epoch is done with the new metric.

5. Final Metric Step. If the learning terminates because the error is below the threshold then the metric in the last epoch is considered accurate and is output as the final metric. However if termination occurs because the maximum number of epochs is reached then the most reasonable metric to be output is the one corresponding to the epoch with the minimum error among all epochs.

Convergence: LearnMet is not guaranteed to converge or yield an optimal distance metric. However, thorough experimental evaluation in our application domain has shown convergence to errors below the required threshold [14]. Proof of convergence in theory will be addressed as future work.

3.3 Algorithm for LearnMet

Given the overview and detailed discussion on the LearnMet technique, we now present the LearnMet algorithm based on each of the steps discussed.

The LearnMet Algorithm

Given: Training set with k actual clusters over G graphical plots, error threshold t, domain expert input on distances applicable to plots

1. **Initial Metric Step**
   (a) For each distance on plots assign a component Dc_i to D
   (b) If relative importance of each Dc_i available
   (c) Then use *Initial Weight Heuristic* to assign each w_i
   (d) Else assign a random w_i to each Dc_i
   (e) Thus set metric $D = \Sigma_{i=1}^{m} w_i Dc_i$
2. **Clustering Step**
   (a) Select arbitrary but fixed clustering algorithm
   (b) Set number of clusters = k (constant)
   (c) Cluster plots using distance $D = \Sigma_{i=1}^{m} w_i Dc_i$
3. **Cluster Evaluation Step**
   (a) Set *ppe* = Number of pairs per epoch
   (b) Select randomly *ppe* pairs of graphical plots
   (c) Calculate TP, TN, FP, FN for *ppe* pairs
   (d) Calculate failure rate $FR = (FP + FN)/(TP + TN + FP + FN)$
   (e) If ($FR < t$) or (epoch == maxEpochs)
   (f) Then go to 5. Final Metric Step
4. **Weight Adjustment Step**
   (a) Calculate distances DFN, DFP
   (b) Apply *Weight Adjustment Heuristic* to get new metric $D^{'''}$
   (c) Go to 2(d) in Clustering Step using $D = D^{'''}$ as distance
5. **Final Metric Step**
   (a) If ($FR < t$) then return metric D
   (b) Else find epoch with minimum failure rate FR
   (c) Return corresponding metric D

4 Enhancement to LearnMet

The basic LearnMet approach yields metrics that provide higher clustering accuracy than the default Euclidean distance [13]. This is elaborated in the section on experimental evaluation. However, there is scope for further enhancement in our targeted domains. The three primary goals in enhancement are:

- *Quality:* This goal refers to improving the accuracy of the distance metrics in processes such as clustering.
- *Efficiency:* This involves reducing the number of epochs needed for convergence and the total training time.
- *Simplicity:* This deals with learning simple metrics that meet the requirements in the domain. Simplicity is mainly concerned with the number of components in the distance metric.

The following approaches are used to meet one or more of the above goals:

1. *Selecting Pairs Per Epoch:* This refers to selecting a suitable number of pairs of graphical plots in each epoch, denoted as pairs per epoch or *ppe*. This is to avoid overfitting to achieve the quality goal. It also impacts efficiency due to execution time with the given number of pairs in each epoch.
2. *Using Domain Knowledge in Weight Selection and Adjustment:* This involves considering semantics of distance components in adjusting weights. It also involves intelligently guessing the initial weights to enhance the learning. This aims to learn metrics closer to the notion of correctness in the domain and to converge faster. Thus it addresses the goals of quality and efficiency.
3. *Learning Simple Metrics:* In this approach we apply the Occam's Razor principle [15] in preferring simpler theories over complex ones. This involves first considering metrics with a single component, then with two components, then three and so forth until convergence occurs or the training times out.

4.1 Selecting Pairs Per Epoch

The basic LearnMet approach uses a default value of pairs per epoch, $ppe = G$ where G is the number of graphical plots in the training set [13]. The enhanced approach involves selecting a suitable number of pairs per epoch such that there are sufficient pairs in every epoch and yet enough randomization. Given that the number of graphical plots in the training set is G, the total number of pairs available for learning is $P = \frac{G!}{2!(G-2)!}$ [10]. For example, if $G = 25$, $P = 300$. In every epoch of LearnMet, a random combination of *ppe* pairs is selected for evaluation and weight adjustment. The total number of distinct combinations of *ppe* pairs that can be made from P pairs is given as $R = \frac{P!}{ppe!(P-ppe)!}$ where R denotes the extent of randomization. For example, if $G = 25$ and $ppe = 15$, the total number of combinations available for learning (extent of randomization) is $R = 7.68 \times 10^{24}$. We now consider different ranges for *ppe*.

Low Range of *ppe*: In order to learn a hypothesis that does not overfit the training data, it is desirable to have more randomization. This would suggest

using low *ppe* values, e.g., $ppe <= G$. These are likely to give a wider range of distinct combinations. However, consider for example an extreme of $ppe = 1$. If one distinct pair is used in each epoch for evaluation and weight adjustment, then it could happen that convergence to error below threshold occurs over the first few epochs. However the learned metric fits only the few pairs that got considered in these epochs. The resulting hypothesis, namely, the learned metric would possibly not give high accuracy over unseen test data, since it is not generic. Likewise, *ppe* values higher than 1 but still in the low range are likely to yield a similar, though perhaps less serious problem of overfitting.

High Range of *ppe*: Consider the argument that a hypothesis is likely to be stronger if it is learned over a larger volume of data. This would suggest the other extreme, i.e., using *ppe* values closer to $P = \frac{G!}{2!(G-2)!}$ Mathematically this would yield a fairly large number of distinct combinations. In fact the number of distinct combinations for $ppe = x$ and $ppe = P - x$ are the same [10]. For example, if $G = 25$ and $P = 300$, the number of distinct combinations for $ppe = 15$ are the same as for $ppe = 285$. For $ppe = 15$, $R = \frac{300!}{15!(300-15)!}$, and for $ppe = 285$, $R = \frac{300!}{285!(300-285)!}$. Both give $R = \frac{300!}{15!285!} = 7.68 \times 10^{24}$. However, there is a major difference between the two. For $ppe = 285$, there is a danger of the same pairs getting selected in each epoch, with only a few pairs distinct, e.g., if pairs 2 through 286 are selected in one epoch, and pairs 3 to 287 in another, only 2 pairs are distinct in these epochs. Thus there is not really enough randomization. Moreover, with high *ppe* values, there is also a huge overhead in each epoch. An extreme of this is $ppe = P$, i.e., all pairs per epoch.

Middle Range of *ppe*: Let us now consider using *ppe* values close to $P/2$, i.e., half the total number of pairs in each epoch. This mathematically gives a large extent of randomization $R = \frac{P!}{(P/2)!(P/2)!}$. For example, for $P = 300$, $ppe = P/2 = 150$ and $R = \frac{300!}{150!150!}$. Moreover, this is likely to yield a genuinely distinct combinations of pairs in every epoch, since it does not consider almost all P pairs together. In addition, this gives a fairly large number of pairs in every epoch. Thus the learned metrics are likely to be more generic. Also, overhead in each epoch is not as high as with *ppe* values close to P. Thus it is good to select middle range *ppe* values.

4.2 Using Domain Knowledge in Weight Selection and Adjustment

The basic LearnMet approach [13] proposes a weight adjustment heuristic that considers all individual metrics at par when assigning the blame for the error. This heuristic is based solely on the distance contribution of each component to the average false positive and false negative distances DFP and DFN respectively. However, a contribution of $DFNc_i/DFN = 0.4$ for Euclidean distance may be more or less crucial than the same for Leidenfrost distance. Likewise, it is desirable to incorporate semantics of components and scale the weight adjustment. Based on this, we define term scaling factor.

A *Scaling Factor* for a distance component in the LearnMet metric is a number that gives the extent to which the weight of that component should be altered based on its semantics.

Scaling factors are calculated as follows. Consider a single component in the LearnMet metric used as the notion of distance in clustering. The greater the accuracy of the clusters given by that component alone, the greater is the significance of the component in the domain. This is because clustering accuracy in LearnMet is the extent to which the predicted clusters over a given data set match the notion of correctness depicted by actual clusters. Thus if a component alone used as the distance metric yields high clustering accuracy, it implies that this component by itself is a significant feature in preserving domain semantics. Now, if a component is more significant, then it is advisable to alter its weight to a greater extent in making adjustments. This is in line with the logic of exploiting the good and making it better to get the best results. We propose the following heuristic for scaling factors.

Scaling Factor Heuristic: Assign a scaling factor to each component in the LearnMet metric directly proportional to the clustering accuracy with that component alone as the notion of distance.

With this discussion, we give the algorithm for computing scaling factors.

Scaling Factor Algorithm

Given: Training sets with actual clusters of plots, Distance components for plots

1. For each component Dc_i in LearnMet metric do
   (a) Repeat z times (where $z = a \times b$, such that a = number of training sets and b = number of clustering seeds)
      i. Do clustering with distance metric $D = Dc_i$
      ii. Calculate TP, TN, FP, FN for all P pairs
      iii. Accuracy of component Dc_i = success rate $SR_i = \frac{TP+TN}{TP+TN+FP+FN}$
   (b) Scaling factor of component $Dc_i = sf_i = (1/z)\Sigma_{j=1}^{z} SR_i$
2. Return scaling factor sf_i for each component Dc_i

With the use of scaling factors, the Weight Adjustment Heuristic gets modified. We thus propose the Scaled Weight Adjustment Heuristic as follows.

Scaled Weight Adjustment Heuristic: For each component Dc_i, its new weight is: $w_i^{'''} = max(0, w_i - sf_i \times \frac{DFNc_i}{DFN} + sf_i \times \frac{DFPc_i}{DFP})$.

Thus, the new metric is: $D^{'''} = \Sigma_{i=1}^{m} w_i^{'''} Dc_i$.

Initial Weight Selection. Scaling factors are likely to boost the learning if weights in the initial metric are assigned according to the initial weight heuristic, i.e., based on the relative importance of each component. Then though learning the weight of a component with a low scaling factor may take longer, this is counterbalanced by fast convergence of components with high scaling factors. Since components with high scaling factors are more important, overall convergence to the notion of correctness is quicker. Also quality of the learned metric is likely to be better. In the basic LearnMet approach [13], relative importance of components is determined by subjective notions of the domain experts and

initial weights are assigned accordingly. However, if this relative importance is not known in advance then weights are assigned randomly, or equal weights are assigned to all components. In the enhanced approach, we assign initial weights proportional to the accuracy of each individual component in clustering which is effectively proportional to its relative importance [14].

4.3 Learning Simple Metrics

The goal of simplicity refers to learning a simple metric that meets the requirements in the given problem. The main requirement in our problem is quality. Simplicity is measured in terms of:

- *Number of components:* The fewer the number of components used in the metric, the simpler is the metric.
- *Amount of data for each component:* The less the amount of data needed to represent each component, the simpler is that component and hence the corresponding metric.

The reasons for learning a simple metric are as follows.

1. Simple metrics are more efficient in terms of time complexity when used as the notion of distance in processes such as clustering.
2. Less storage space is required for simple metrics.
3. Experts cannot always identify components applicable to the plots. A brute force combination of components is not practical.

Principle in Learning. The main principle applied here is that of Occam's Razor which states that simpler theories are preferred over complex ones [15]. In our case, the theory refers to the learned metric. However, in our problem simple metrics are considered better than complex metrics only if both achieve the same quality. Considering the two criteria of quality and simplicity, the process of learning, analagous to greedy search [15] is outlined below.

Process of Learning Simple Metrics

Given: Training set with actual clusters of plots, error threshold t

1. Identify all m components $Dc_i : i = 1$ to m in the domain
2. For each Dc_i
   (a) Do clustering in LearnMet with $D = Dc_i$, get FR and SR (failure and success rates)
   (b) If $(FR <= t)$ then set *final metric* $= D$ and go to step 5
3. If $(FR > t)$ then $m^{'} = 2$; Repeat
   (a) Execute LearnMet with $D = \Sigma_{i=1}^{m^{'}} w_i Dc_i$ where $Dc_1 \ldots Dc_m^{'}$ are the $m^{'}$ components with highest clustering accuracies
   (b) If $(FR <= t)$ then set *final metric* $= D$ and go to step 5
   (c) Set $m^{'} = m^{'} + 1$ and go to step 3(a)
4. Until $m^{'} = m$
5. Output *final metric*

5 Experimental Evaluation

LearnMet has been rigorously evaluated in the Heat Treating domain [2]. The source code used for experimentation is our LearnMet software tool [14] developed in Java, implementing k-means [8] for clustering. The platform used for running the experiments is a Mobile Intel Celeron (R) PC with a CPU Speed of 2 GHz, 192 MB of RAM and the Microsoft Windows XP Professional Version 2002 operating system. A summary of our evaluation is presented here.

5.1 Evaluation of Basic LearnMet Approach

Effect of Initial Metrics. These experiments are conducted to observe the impact of the initial metrics on the learning. The experiments shown below are for $G = 25$ graphical plots giving 300 pairs. The number of clusters is $k = 5$ since this is the equal to number of actual clusters given by experts. The test set consists of 40 distinct graphical plots placed in $k = 7$ clusters giving 780 pairs. Experts give an error threshold of 10%, i.e., 0.1 as acceptable in the domain for evaluation over test sets. As a default, we use the same threshold for learning over the training set. The maximum number of epochs is maintained at a constant value of 1000. Initial components in the metric are given by experts. Two distinct assignments of initial weights are given by two different experts. The corresponding two metrics are denoted by $DE1$ and $DE2$ respectively. A third initial metric EQU is obtained by assigning equal weights to all components. Several experiments are run by assigning random weights to components in the initial metric [14]. We show two experiments with randomly generated metrics called $RND1$ and $RND2$. The initial metrics are shown in Figure 5. Each experiment is the average of 10 experiments conducted by altering parameters such as clustering seeds. The learned metrics are shown in Figure 6.

EXPT	INITIAL METRIC
DE1	$5D_{Euclidean} + 2D_{Mean} + 5D_{Max} + 1D_{LF} + 3D_{BP}$
DE2	$5D_{Euclidean} + 1.5D_{Mean} + 4.5D_{Max} + 3D_{LF} + 3D_{BP}$
EQU	$1D_{Euclidean} + 1D_{Mean} + 1D_{Max} + 1D_{LF} + 1D_{BP}$
RND1	$6D_{Euclidean} + 2.75D_{Mean} + 3.5D_{Max} + 3.25D_{LF} + 1.85D_{BP}$
RND2	$4D_{Euclidean} + 3.2D_{Mean} + 4.3D_{Max} + 1.86D_{LF} + 2.9D_{BP}$

Fig. 5. Initial Metrics in Experiments

EXPT	LEARNED METRIC
DE1	$4.0645D_{Euclidean} + 1.7784D_{Mean} + 2.8428D_{Max} + 1.9625D_{LF} + 3.0238D_{BP}$
DE2	$4.2688D_{Euclidean} + 1.6114D_{Mean} + 2.9998D_{Max} + 2.2778D_{LF} + 2.9239D_{BP}$
EQU	$4.0578D_{Euclidean} + 1.8657D_{Mean} + 2.7749D_{Max} + 2.3257D_{LF} + 2.8481D_{BP}$
RND1	$4.1301D_{Euclidean} + 1.7345D_{Mean} + 2.8514D_{Max} + 2.1187D_{LF} + 3.1402D_{BP}$
RND2	$4.2204D_{Euclidean} + 1.8175D_{Mean} + 2.7968D_{Max} + 2.0581D_{LF} + 3.0864D_{BP}$

Fig. 6. Learned Metrics in Experiments

The clustering accuracy of each metric over the test set is shown in Figure 7. Clustering accuracy as stated earlier is measured by comparing clusters obtained from learned metrics with actual clusters over the test set. Comparative evaluation is also performed using Euclidean distance (shown as ED). Learning efficiency, i.e, number of epochs for convergence, is shown in Figure 8.

Discussion on experiments with different initial metrics:

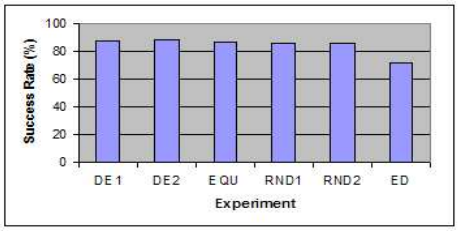

Fig. 7. Clustering Accuracy with Metrics

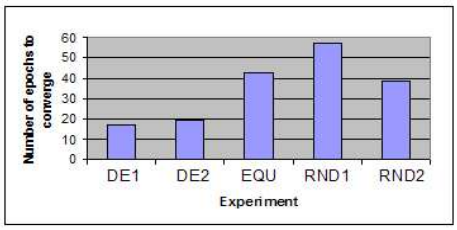

Fig. 8. Learning Efficiency with Metrics

1. Convergence to error below threshold occurs in all the experiments.
2. The experiments with initial metrics provided by experts converge faster.
3. All the experiments converge to approximately the same learned metrics.
4. Accuracies of the learned metrics are higher than those of Euclidean distance.

Effect of the Error Threshold. The parameter of interest in these experiments is the value of the threshold used for error. In the experiments shown here the training set is of size $G = 15$ with number of clusters $k = 3$. The test set is of size $G = 25$ with $k = 5$. Thresholds are altered from 0.1 to 0.01. The initial metrics in these experiments are the same as in Figure 5. Each experiment here shows the average of 10 experiments performed by altering clustering seeds [14]. Figures 9 and 10 show the clustering accuracy over the test set and the learning efficiency over the training set respectively.

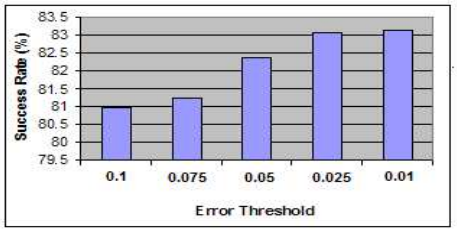

Fig. 9. Accuracy with Thresholds

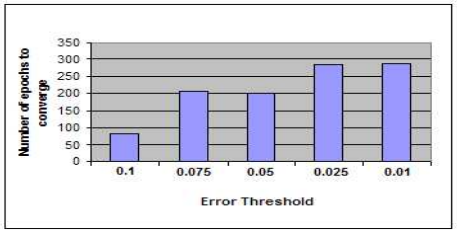

Fig. 10. Efficiency with Thresholds

Discussion on the impact of varying error thresholds:

1. Clustering with the learned metrics gives higher accuracy than clustering with Euclidean distance (as observed from earlier experiments).
2. As the error threshold is reduced, the number of epochs to converge tends to increase. Thus the learning efficiency reduces with a decrease in threshold.
3. With reduced thresholds however, the clustering accuracy over the test set is higher. As the threshold is reduced from 0.1 to 0.01, accuracy increases from approximately 81% to 83%.

5.2 Evaluation of Enhanced LearnMet Approach

Effect of the Number of Pairs Per Epoch. The following experiments are conducted to observe the impact of the number of pairs per epoch *ppe* on the learning. The parameters varied in these experiments are the *ppe* values and seeds in clustering. These experiments are a summary of approximately 200 experiments conducted on *ppe* values [14] with different training and test set sizes. The training set shown here has $G = 40$ graphical plots with number of clusters $k = 7$. The test set has $G = 25$ with $k = 5$. The initial metric is $D = 5D_{Euclidean} + 4D_{Mean} + 3D_{Max} + 2D_{LF} + 1D_{BP}$. The error threshold is maintained at 1%, i.e., 0.01 (based on results of experiments with thresholds). Observations are summarized in terms of clustering accuracy over the test set and learning efficiency over the training set in Figures 11 and 12 respectively. In addition, Figure 13 shows some details of behavior during training. The failure rate is plotted versus the epoch for three different ranges of *ppe* values.

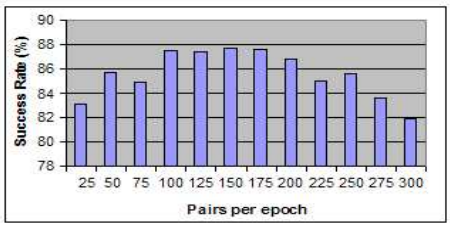

Fig. 11. Accuracy with *ppe* Values

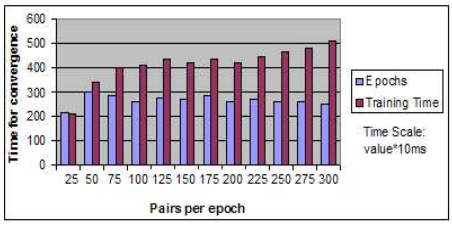

Fig. 12. Efficiency with *ppe* Values

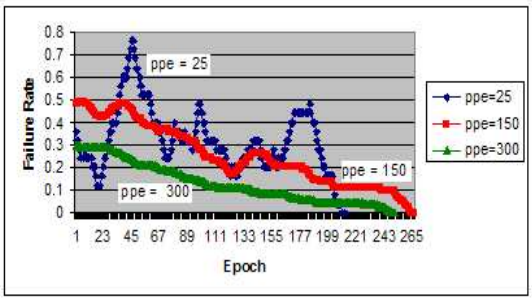

Fig. 13. Training Behavior with *ppe* Values

Discussion on the effect of ppe values:

1. Low *ppe* values, such as $ppe < G$ may converge faster but the learned metrics give relatively lower clustering accuracy over the test set. Also some exper-

iments with low *ppe* values may take as long to converge as *ppe* close to G. This depends on which *ppe* pairs get selected in each epoch.
2. Middle range *ppe* values take longer to converge but give best clustering accuracy over the test set.
3. High *ppe* values close to P, (i.e, all pairs) take still longer to converge and give accuracy over test set less than middle range values.
4. Failure rate decreases monotonously for high *ppe* values but oscillates for lower *ppe* values. This is because for low *ppe* values, a distinctly different set of pairs get used in each epoch for learning, so the metric is learned over a different set of pairs each time. For higher *ppe* values, almost the same pairs get selected in each epoch, thus causing a uniform decrease in failure rate.

Effect of Scaling Factors and Initial Weights. The experiments below are conducted to observe the impact of scaling factors (sf) on the learning. These experiments are a summary of approximately 200 experiments conducted on scaling factors [14]. The experiments shown here are for a training set of $G = 40$ with number of clusters $k = 7$. The test set is of size $G = 15$ and number of clusters $k = 3$. The distance components used are the D_{SC}, $D_{Euclidean}$, D_{Min}, D_{Mean}, D_{Max}, D_{LF} and D_{BP}. The number of pairs per epoch is maintained at $ppe = P/2 = 150$ (based on results of *ppe* experiments). Error threshold is set to 1%. The parameters varied are the initial weights, scaling factors and clustering seeds. Figure 14 illustrates the parameter settings used for the experiments shown here. Figures 15 and 16 show the test set and training set observations in terms of clustering accuracy and learning efficiency respectively.

Expt	Scaling Factors	Initial Weights
1	Equal	Random 1
2	Proportional to Accuracy	Random 2
3	Proportional to Accuracy	Random 3
4	Proportional to Accuracy	Given by Expert
5	Proportional to Accuracy	Proportional to Accuracy

Fig. 14. Parameters Settings in Scaling Factor Experiments

Discussion on scaling factor experiments:

1. With scaling factors proportional to accuracy of each individual component and with random initial metrics, convergence may occur faster or slower than with no scaling factors. This depends on the initial metric.
2. With initial metrics given by experts and with scaling factors proportional to accuracy, results are better than with random metrics.
3. Even better results are observed for initial metrics and scaling factors both proportional to accuracy of each individual component in clustering.

Effect of Components. These experiments show the impact of altering the components in the distance metric. Approximately 100 experiments have been

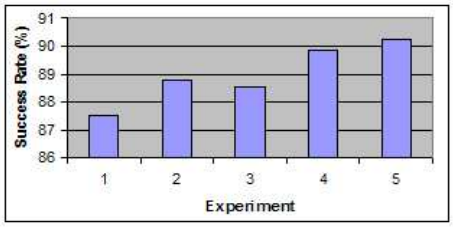

Fig. 15. Accuracy with Scaling Factors

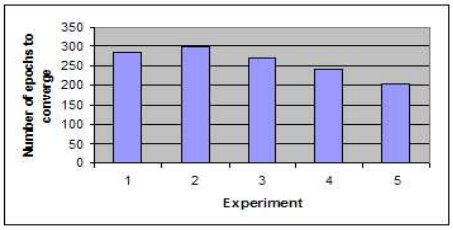

Fig. 16. Efficiency with Scaling Factors

conducted [14], a summary of which is presented below. The number of graphical plots in the training set is $G = 25$ from which $P = 300$ pairs of plots are obtained. The number of clusters is $k = 5$ from the actual clusters over the training set. The number of pairs per epoch is maintained at $ppe = P/2 = 150$, since this is learned as a good setting from previous experiments. The error threshold is maintained at 0.01 and scaling factors proportional to accuracy are used in the weight adjustment heuristic. The maximum number of epochs is constant at 1000. The test set used is of size $G = 15$, with $k = 3$ actual clusters provided. The number of components m is altered in each experiment. The possible components identified for in the domain are the individual metrics D_{SC}, $D_{Euclidean}$, D_{Mean}, D_{Max}, D_{LF}, D_{BP}, and D_{Min}. Initial weights of the components are proportional to clustering accuracy of each, for those experiments with multiple components in the metric. For metrics with single components, the weight is 1, i.e., that component alone is the distance metric. We alter the seeds in the clustering algorithm for randomization. The components in each experiment are shown in Figure 17. The observations are mainly clustering accuracy over the test set as shown in Figure 18.

Expt	Components
1	D_SC
2	D_Euclidean
3	D_Min
4	D_Mean
5	D_Max
6	D_LF
7	D_BP
8	D_Euclidean, D_Max
9	D_Euclidean, D_Max, D_BP
10	D_Euclidean, D_Max, D_BP, D_LF
11	D_Euclidean, D_Max, D_BP, D_LF, D_Mean
12	D_Euclidean, D_Max, D_BP, D_LF, D_Mean, D_SC
13	D_Euclidean, D_Max, D_BP, D_LF, D_Mean, D_SC, D_Min

Fig. 17. Components in Experiments

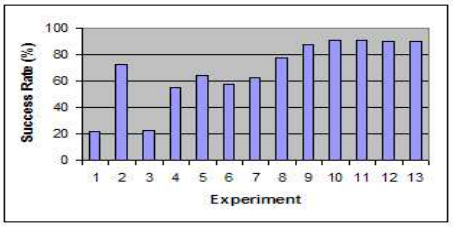

Fig. 18. Accuracy with Components

Discussion on experiments with components:

1. Among the individual metrics, Euclidean distance gives highest accuracy.
2. The highest is accuracy is in the range if 90%.
3. The simplest metric giving this accuracy has 4 components $D_{Euclidean}$, D_{Max}, D_{LF} and D_{BP}.

6 Related Work

In [7], an overview of distance metrics useful for similarity search in multimedia databases is presented. Some of these are applicable as components in the LearnMet metric. However, they do not learn a single metric encompassing various distance types. Tri-plots [11] provide a tool for multidimensional data mining using intrinsic dimensionality, cross correlations and cumulative distribution functions of pair-wise distances between n-dimensional objects. Tri-plots could possibly be useful to us in assigning the individual components to the LearnMet metric. However, the focus in Tri-plots is on the overall shape and dimensionality of objects. In our context the basic shape and dimensionality of the graphical plots is similar. The focus is on the semantics.

In [5] a learning method is proposed to find relative importance of dimensions for n-dimensional objects. However, their focus is on dimensionality reduction and not on domain semantics. In [16] they learn which type of position-based distance is applicable for the given data starting from the formula of Mahalanobis distance. However they do not consider other distance types besides position-based distances [16] and do not address semantics.

Genetic algorithms [3] can be used to select features in graphs relevant for clustering thus trying to learn a distance metric. However, they do not give enough accuracy in our applications [14]. Neural networks [1] could possibly be used for distance metric learning. However our data is such that the distance between pairs of plots is not known in advance to serve as the training set required [14]. Similar issues hold for other learning techniques such as support vector machines [15] since we do not have positive and negative training samples available in advance as required for learning.

Zhou et al [17] propose an approach for ensembling neural networks. They train a number of neural networks at first, then assign random weights to them and employ a genetic algorithm to evolve the weights to characterize the fitness of the neural network in consitituting an ensemble. Although we do not use neural networks, each distance metric in our problem could possibly be viewed as a learner, thus in combining them we get an ensemble. At present, we use an approach analogous to greedy search to learn simple metrics [15]. Considering other approaches for such sub-problems within LearnMet and comparing their computational complexity and accuracy with our present approach presents interesting future issues. However, it is beyond the scope of this paper.

7 Conclusions

The LearnMet technique is proposed to learn semantics-preserving distance metrics for graphical plots of scientific functions. Learning in this technique involves comparing predicted clusters of such plots using a guessed initial metric with actual clusters provided by experts, and refining the distance metric in every epoch based on the error between the predicted and the actual clusters, until the error is minimal or below the given threshold. The metric giving the lowest error is output as the learned metric. LearnMet is rigorously evaluated in

the Heat Treating domain that motivated its development. It is found to learn distance metrics that improve clustering results as compared to Euclidean distance. Enhancements are made to LearnMet by using approaches to select a suitable number of training samples in each epoch, incorporating domain knowledge in weight adjustment and using the Occam's Razor principle in learning. The enhancements are found to provide even better clustering accuracy. They also increase the learning efficiency and yield simpler learned metrics. LearnMet is basically an empirical approach. Future work includes theoretical discussions and proofs on various LearnMet heuristics and a comparative study on the computational complexity of LearnMet.

References

1. Bishop, C.: Neural Networks for Pattern Recognition. Oxford University Press, England (1996).
2. Boyer, H. and Cary, P.: Quenching and Control of Distortion. ASM International, Ohio (1989).
3. Friedberg, R.: A Learning Machine: Part I. In IBM Journal (1958) Vol. 2, pp. 2-13.
4. Han, J. and Kamber, M.: Data Mining Concepts and Techniques. Morgan Kaufmann, California (2001).
5. Hinneburg, A., Aggarwal, C. and Keim, D.: What is the Nearest Neighbor in High Dimensional Spaces. VLDB (Aug 2000) pp. 506-515.
6. Kaufman, L. and Rousseeuw, P.: Finding Groups in Data: An Introduction to Cluster Analysis. John Wiley, New York (1988).
7. Keim, D. and Bustos, B.: Similarity Search in Multimedia Databases. IEEE's ICDE (Mar 2004) pp. 873-874.
8. MacQueen, J.: Some Methods for Classification and Analysis of Multivariate Observations. Mathematical Statistics and Probability (1967) Vol. 1, pp. 281-297.
9. Maniruzzaman, M., Chaves, J., McGee, C., Ma, S. and Sisson, R.: The CHTE Quench Probe System. In ICFDM (Jul 2002) pp. 13-17.
10. Petrucelli, J., Nandram, B. and Chen, M.: Applied Statistics for Engineers and Scientists, Prentice Hall, New Jersey (1999).
11. Traina, A., Traina, C., Papadimitriou, S. and Faloutsos, C.: TriPlots: Scalable Tools for Multidimensional Data Mining. ACM KDD (Aug 2001) pp. 184–193.
12. Varde, A., Rundensteiner, E., Ruiz, C., Maniruzzaman, M. and Sisson, R.: Data Mining over Graphical Results of Experiments with Domain Semantics. ACM SIGART's ICICIS (Mar 2005) pp. 603-611.
13. Varde, A., Rundensteiner, E., Ruiz, C., Maniruzzaman, M. and Sisson, R.: Learning Semantics-Preserving Distance Metrics for Clustering Graphical Data. ACM KDD's MDM (Aug 2005) pp. 107-112.
14. Varde, A. Graphical Data Mining for Computational Estimation in Materials Science Applications. Ph.D. Dissertation in Progress, Worcester Polytechnic Institute, MA (Apr 2006).
15. Witten, I. and Frank, E.: Data Mining: Practical Machine Learning Algorithms with Java Implementations. Morgan Kaufmann (2000).
16. Xing, E., Ng, A., Jordan, M. and Russell, S.: Distance Metric Learning with Application to Clustering with Side Information. NIPS (Dec 2003) pp. 503-512.
17. Zhou, Z., Wu, J. and Tang, W.: Ensembling Neural Networks: Many Could Be Better Than All. Artificial Intelligence (2002) Vol. 137, No. 1-2, pp. 239-263.