# Distance Metric Learning by Greedy, Exhaustive and Hybrid Approaches

Aparna Varde* Stephen Bique† David Brown‡

**Abstract**

This paper focuses on distance metric learning for graphical data. We deal with graphs plotting scientific functions in the form of a dependent versus an independent variable. To compare such graphs it is often important to consider various features, e.g., critical regions, statistical observations and absolute position of points. These features could potentially be defined by individual metrics (components), a weighted sum of which defines a notion of distance for the graphs. Selecting appropriate components for the given data is a significant challenge. This is the problem addressed in the paper. We consider greedy and exhaustive selection approaches. Greedy selection aims to learn a notion of distance favoring fewer components. Exhaustive selection refers to considering all possible combinations of components. In addition, we propose hybrid greedy and hybrid exhaustive approaches by merging greedy and exhaustive selection in a more greedy and more exhaustive manner respectively. We compare all these approaches based on the complexity of the learning algorithm, accuracy of the learned distance and efficiency of execution.

**Keywords:** Feature Selection, Domain Semantics, Distance Metrics, Clustering, Data Preprocessing, Learning Algorithms

## 1 Introduction.

Learning distance metrics for complex data involves interesting issues pertaining to factors such as the nature of the data, the semantics of the domain and the requirements of specific problems. Accordingly, distance metric learning has been approached using several methods, e.g., [1, 2, 3, 10, 12, 14]. In our work, the focus is on graphical data, more specifically, two-dimensional graphs plotting the results of scientific experiments as a dependent versus an independent variable. In order to compare and analyze such graphs analogous to domain experts, it is important to capture their semantics. Although various distance metrics exist in the literature [4, 6, 13], it is often not known apriori which of these best fits the given data.

In dealing with the graphs in the context of our problem, various issues need to be considered [12]. For example, some regions of the graphs represent critical phenomena in the domain and are more significant than other regions. In addition, there are statistical observations on the graphs depicting behavior of process parameters. Also, it is important to take into account the absolute position of points on the graphs. Thus, it seems advisable to capture these features using individual distance metrics, a combination of which can be used as a notion of distance for the graphs. The relative importance of the components can then be defined by weights that can be learned [12]. However, a crucial issue is the selection of individual metrics, referred to in our work as components. It is not always feasible for domain experts to determine which components are the best to use in the metric. Factors such as accuracy of the learned notion of distance and efficiency of the learning process also need to be taken into account.

Given the above issues we address the problem of component selection for distance metric learning. We consider two main approaches to select components, namely, the greedy and the exhaustive approach [5, 8]. In greedy selection, the goal is to keep the learned distance metric simple. The heuristic for greedy selection is based on the accuracy of each individual component. The other extreme is exhaustive selection which considers all possible combinations of components. In addition to these two approaches, we propose two hybrid approaches for selection. The hybrid greedy approach follows the basic principle of greedy selection but considers more combinations than greedy. The hybrid exhaustive approach considers all possible combinations as in the exhaustive approach but after pruning certain components. In all approaches, an initial set of components is identified by a knowledge of distance metrics and with the help of domain experts. The weights of the components are learned using our LearnMet algorithm [12].

We compare the component selection approaches in terms of three criteria: the computational complexity of the learning algorithm, the accuracy of the learned distance metric in preserving semantics and the efficiency of learning in terms of execution time. Experimental evaluation is conducted with real data from the domain of Materials Science [9]. It is found that all the approaches have their pros and cons in distance metric learning.

*Assistant Professor, Department of Math and Computer Science, Virginia State University, Petersburg, VA

†Associate Professor, Department of Math and Computer Science, Virginia State University, Petersburg, VA

‡Professor, Department of Computer Science, Worcester Polytechnic Institute, Worcester, MA

The rest of this paper is organized as follows. Section 2 gives an overview of distance metric learning using the LearnMet technique. Sections 3 and 4 explain the greedy and exhaustive selection, respectively. Section 5 describes the two hybrid selection approaches. Section 6 presents a comparison of the computational complexity of the selection approaches. Section 7 summarizes the comparative evaluation of accuracy and efficiency for all the approaches. Section 8 outlines related work. Section 9 gives the conclusions.

## 2 Technique for Distance Metric Learning.

In our prior work, we propose a technique called LearnMet [12] to learn domain-specific distance metrics for graphs. We define a *LearnMet Distance Metric* $D$ as a weighted sum of components, where each component is an individual metric describing a particular aspect of the graphs and the weight of each component is a numerical value giving its relative importance in the domain. Thus $D = \Sigma_{i=1}^{m} w_i D_i$ where each $D_i$ is a component, $w_i$ is its weight, and $m$ is the number of components.

The input to LearnMet is a training set with actual clusters of graphs provided by domain experts. These are iteratively compared with clusters over the same graphs predicted using an arbitrary but fixed clustering algorithm. Using a guessed initial metric $D$ 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 [12]. The details of this technique are described in the next subsection.

### 2.1 Details of Technique.

In order to guess the initial metric in LearnMet, domain experts are asked to identify components applicable to the graphs. If experts have a subjective notion about relative importance of components, then this notion is used to assign initial weights using the following heuristic [12]:

**Initial Weight Heuristic:** *Assign initial weights to components in the LearnMet 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 is assigning equal weights to all components. Weights are typically assigned on a scale of 0 to 10.

An arbitrary but fixed clustering algorithm, e.g., k-means [7] is selected. Using $D = \Sigma_{i=1}^{m} w_i D_i$ 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.

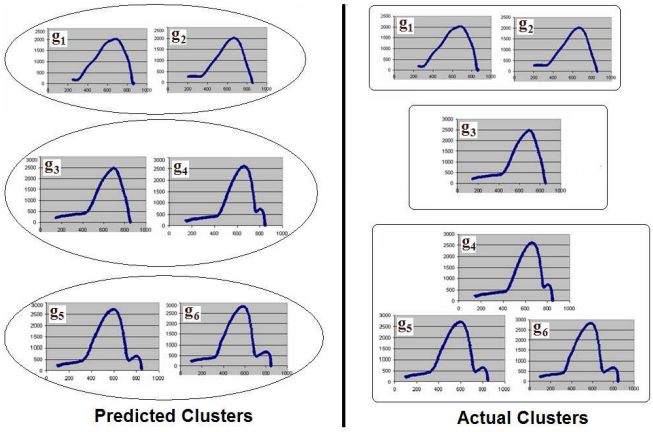


Figure 1: Predicted and Actual Clusters

Figure 1 shows an example of predicted and actual clusters of graphs. In order to compute error, we consider pairs of graphical plots and introduce the following notion of correctness [12]:

**Notion of Correctness:** Given a pair of graphical plots $g_a$ and $g_b$, we say that:

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

Thus, the true positives and true negatives are considered to be correct pairs while false positives and false negatives are error pairs. In each epoch, a randomly selected subset of pairs is used for evaluation and weight adjustment referred to as pairs per epoch or *ppe*. The error measure of interest to us is failure rate which is explained next [13]. Let $TP$, $TN$, $FP$ and $FN$ denote the number of true positive, true negative, false positive and false negative pairs respectively for a given *ppe* value. 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, $FR = \frac{FP+FN}{TP+TN+FP+FN}$. A domain-specific error threshold $t$ is used, where $t$ is the extent of error allowed per epoch in the domain. If the

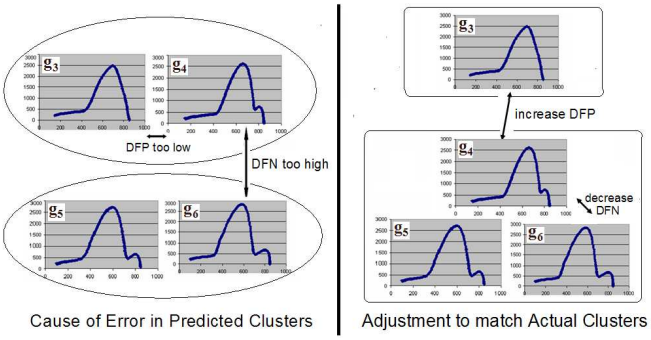


Figure 2: Distances used in Weight Adjustment

error is less than or equal to the threshold then the final metric is output. However, if the error is greater than the threshold in a given epoch, then the metric is adjusted based on this error.

In order to adjust the metric $D$ using error between predicted and actual clusters, we introduce the concept of false positive and false negative distances $DFP$ and $DFN$ respectively [12] as shown in Figure 2. $DFP$ is defined as the average distance using the metric $D$ of the false positive pairs. Likewise, $DFN$ is defined as the average distance using the metric $D$ of the false negative pairs.

Given this, we now consider the error due to the false positive pairs. To reduce this error it is desirable to increase the distance $DFP$. In order to increase $DFP$ the weights of one or more components in the metric used to calculate the distance in the present epoch is increased. Thus new weight $w_i^{''} = w_i + \frac{DFP_i}{DFP}$ where $DFP_i = DFP$ for component $D_i$ alone. Conversely, to reduce error due to the $FN$ pairs we decrease $DFN$ by decreasing the weights of one or more components in metric $D$. Hence new weight $w_i^{'} = w_i - \frac{DFN_i}{DFN}$ where $DFN_i = DFN$ for component $D_i$ alone. Combining these two we get the weight adjustment heuristic below.

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

The new metric obtained after adjustment is likely to minimize error due to both false positive and false negative pairs. Clustering in the next epoch is done with this new metric. The process is repeated until error is below threshold or maximum number of epochs is reached. The final metric is then output.

Based on the above discussion, we give the LearnMet algorithm that we use for distance metric learning [12].

## 2.2 Algorithm for Distance Metric Learning

The main steps of the algorithm as explained above are: initial metric step, clustering step, cluster evaluation step, weight adjustment step and final metric step. These steps are outlined below.

*The LearnMet Algorithm:*

Given: Training set with $k$ actual clusters over $G$ graphs, error threshold $t$, domain expert input on individual distance metrics applicable to graphs.

1. **Initial Metric Step**
   (a) For each individual metric assign a component $D_i$ to $D$
   (b) If relative importance of each $D_i$ available use *Initial Weight Heuristic* to assign each $w_i$
   (c) Else assign a random $w_i$ to each $D_i$
2. **Clustering Step**
   (a) Select arbitrary but fixed clustering algorithm
   (b) Set number of clusters = k (constant)
   (c) Cluster plots using distance $D = \Sigma_{i=1}^{m} w_i D_i$
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 ($FR \leq t$) or (epoch == maxEpochs) go to Final Metric Step
4. **Weight Adjustment Step**
   (a) Calculate distances $DFN$, $DFP$
   (b) Use *Weight Adjustment Heuristic* to get new metric $D^{'''}$
   (c) Go to 2(c) in Clustering Step using $D = D^{'''}$ as distance
5. **Final Metric Step**
   (a) If ($FR \leq t$) return metric $D$
   (b) Else find epoch with minimum failure rate $FR$
   (c) Return corresponding metric $D$

Given this, we now consider the following approaches for selection of components.

## 3 Greedy Approach.

In greedy selection, the goal is to try to learn a simple metric fast, yet meeting the minimum requirements of accuracy as per the given problem. The fewer the number of components used in the metric, the simpler is the metric. The main principle applied in learning in the greedy approach is that of Occam's Razor which states that simpler theories are preferred over complex ones [8]. In our case, the theory refers to the learned metric.

Greedy selection 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. The preference of one component over another is determined by the accuracy of each individual component in preserving semantics. Accuracy is measured as the success rate $SR$ of that component used alone as the notion of distance in clustering in LearnMet. Conversely error is measured as the failure rate $FR = 1 - SR$ as described earlier. Given this, we now define the heuristic for greedy selection.

**Greedy Heuristic:** *Select each component $D_i$ such that if $FR_{D_a} < FR_{D_b}$ then $D_a$ is preferred over $D_b$, where $D_a$ and $D_b$ each denote a component $D_i$, and $FR_{D_i}$ is the failure rate of $D_i$ used alone as distance $D$ in clustering.*

Using this heuristic the process of greedy selection is explained as follows. Given a training set of graphs placed in correct clusters and an error threshold $t$ acceptable in the domain, we study the graphs using a fundamental knowledge of distance metrics and with the help of domain experts. Based on this study, we identify $m$ components, i.e., individual distance metrics potentially applicable to them. We consider each component $D_i$, one at a time as the metric $D$. Using this as the notion of distance, we execute the clustering step of LearnMet and evaluate error and accuracy (failure and success rates). If error with any component $D_i$ is less than or equal to the threshold then we output that component alone as the learned metric $D$ and stop. Otherwise, we consider the two best components, i.e., the two giving highest accuracy individually as the initial metric and execute all the steps of LearnMet. If the learned metric yields error below the given threshold then this is output as the final metric. If error is still above threshold, then we repeat this process with the best three, best four components and so forth until error is below threshold or all best combinations are considered.

The logic behind this approach is that if a combination of the best and second best components does not give error below threshold, then it is less likely that a combination of the best and third best will yield any better results. Hence this approach avoids the combinations that do not seem promising. Instead, as a next step it considers the best, second best and third best components since this combination would probably have a greater chance of giving error below the threshold. Based on this discussion, our algorithm for greedy selection of components is presented.

*Greedy Selection Algorithm:*

Given: Actual clusters of graphs, error threshold $t$

1. Identify all $m$ applicable components
2. For each $D_i : 1 = 1$ to $m$
   (a) Do clustering in LearnMet with $D = D_i$, calculate $FR$ and $SR$
   (b) If ($FR \leq t$) then set *final metric* $= D$ and go to step 4
3. Else $j = 2$
   (a) Execute LearnMet with $D = \Sigma_{i=1}^{j} w_i D_i$ where $D_1, \ldots, D_j$ are the $j$ best components
   (b) If ($FR \leq t$) then set *final metric* $= D$ and go to step 4
   (c) $j = j + 1$
   (d) If $j \leq m$ go to step 3(a)
4. Output *final metric*

## 4 Exhaustive Approach.

Our method for exhaustive selection of components follows the concept of exhaustive searches in the literature [5]. In these searches, typically each and every path is traversed. Exhaustive searches usually occur without heuristics. In the context of our problem, the exhaustive method involves considering all possible combinations of components. We then return the combination that gives maximum accuracy in preserving semantics as the notion of distance. As in the case of the greedy approach, accuracy is measured as the success rate of that component used alone as the notion of distance in clustering. Conversely error is measured as the failure rate. The process is as follows.

We first identify the applicable components as in the greedy approach. We then consider each individual component alone as the notion of distance in clustering and record its failure rate as calculated in the LearnMet algorithm. Regardless of the values of the failure rates of the individual components, we next consider all possible combinations of two components as initial metrics in LearnMet. After executing the LearnMet steps, the failure rates of the corresponding learned metrics are recorded. Then all possible combinations of three components are considered and the process is repeated. Likewise, we proceed until all possible combinations of $m$ components are considered as initial metrics. In each case, we record the failure rates of the corresponding learned metric. Among all the combinations the one with the lowest failure rate is returned as the final metric. In case of ties, the one with minimal components is returned. Note that this selection could proceed in any order, i.e., we could start with combinations of all $m$ components, then consider combinations of $m - 1$ components and so forth. Our algorithm for exhaustive selection of components is shown below.

*Exhaustive Selection Algorithm:*

Given: Actual clusters of graphs, error threshold $t$

1. Identify all $m$ applicable components
2. $j = 1$
   (a) Execute LearnMet with all possible combinations of $j$ components as metric $D$
   (b) For each combination record failure rate $FR$
   (c) $j = j + 1$
   (d) If $j \leq m$ go to step 2(a)
3. Output *final metric* with lowest $FR$

## 5 Hybrid Approaches.

While greedy approaches are based on the Occam's Razor principle of keeping things simple, exhaustive approaches stem from the Epicurean philosophy, i.e., the other extreme of experiencing everything [8]. However, the greedy selection does not find the most accurate solution nor does it yield the simplest possible metric. The exhaustive selection is not practical except for cases involving only a small number of components. We therefore investigate two hybrid approaches using principles from both greedy and exhaustive search. We describe these approaches next.

- **Hybrid Greedy Selection:** This approach is basically greedy in principle but considers more combinations, thus tending towards the exhaustive approach.
- **Hybrid Exhaustive Selection:** In this approach the selection is basically exhaustive but after pruning components using ideas from the greedy heuristic.

The details of these two hybrid approaches for component selection are explained below. Before we proceed further, we first define the concept of a Tolerance Limit Heuristic as follows.

**Tolerance Limit Heuristic:** *The tolerance limit $L$ is defined as the upper limit of failure rate (or lower limit of success rate). If the failure rate of component $D_i$ used alone as the notion of distance is higher that the tolerance limit then this component is not considered to be useful and is hence discarded as per this heuristic.*

Note that the tolerance limit $L$ is different from the error threshold $t$. Threshold $t$ defines satisfactory performance, i.e., if error drops below this threshold then the clustering is considered to be good. The tolerance limit on the other hand measures the other extreme, i.e., to what extent unsatisfactory performance can be tolerated. This is used to prune unwanted components during selection.

We are now ready to explain hybrid greedy and hybrid exhaustive approaches for selection.

### 5.1 Hybrid Greedy Approach.

In this approach the basic idea is to select the components in a greedy manner. Thus it follows the greedy heuristic of preferring components with higher individual accuracies. However, we consider not only the best components at each stage but also consider the rest of them in descending order of their accuracies. This approach tends towards exhaustive selection. The stopping criterion is the same as for greedy selection, i.e., stop when the failure rate drops below a given threshold. However, we do not consider all $m$ components (as in pure exhaustive selection), since we apply the Tolerance Limit Heuristic. The process is described next.

We first use each component individually as the notion of distance in clustering and calculate its failure rate using LearnMet. If the failure rate of any single component is less than or equal to the threshold then we output that component alone as the final metric and stop. If not, then as a next step we apply the Tolerance Limit Heuristic to prune $s$ components that give individual failure rates greater than the tolerance limit $L$. For the remaining $(m - s)$ components, the selection proceeds as follows. We consider combinations of two components as the initial metric and execute LearnMet. This time, however, instead of considering only the best two components, i.e,, the two with individually lowest failure rates, we consider other combinations also. Thus, if the best and the second best components do not give failure rate below threshold, then instead of considering the best three as a next step, we consider the best and the third best. This is repeated until all combinations are considered with components ordered in descending order of accuracies or until the failure rate drops below the threshold. This approach aligns with the logic of exhaustive selection. The main argument is that other paths need to be explored rather than directly choosing the path that seems likely to give minimal error. The algorithm that we propose for hybrid greedy selection is outlined as follows.

*Hybrid Greedy Selection Algorithm:*

Given: Actual clusters of graphs, error threshold $t$, tolerance limit $L$

1. Identify all $m$ applicable components
2. For each component $D_i : i = 1$ to $m$
   (a) Do clustering in LearnMet with $D = D_i$, calculate $FR$ and $SR$
   (b) If $(FR \leq t)$ then set *final metric* $= D$ and go to step 5
3. Use Tolerance Limit Heuristic to prune $s$ components that give $FR > L$
4. $j = 2$

(a) Execute LearnMet with combinations of $j$ components ordered from highest to lowest accuracies

(b) If ($FR \leq t$) for any combination then set *final metric* = $D$ and go to step 5

(c) Else $j = j + 1$

(d) If $j \leq s$ go to step 4(a)

5. Output *final metric*

**5.2 Hybrid Exhaustive Approach.** This approach follows the exhaustive method of considering all possible combinations even if failure rate drops below threshold and finally returning the combination with the lowest failure rate. However, we consider only those components whose individual failure rates are greater than or equal to the given tolerance limit $L$. This stems from the greedy heuristic of preferring components with higher accuracies. The method is explained below.

We consider each component individually and evaluate its failure rate using LearnMet. Using the Tolerance Limit Heuristic, we discard $s$ components whose failure rate is greater than the tolerance limit $L$. For the remaining $(m - s)$ components, we consider all possible combinations of components as the initial metric in LearnMet and execute all of its steps. This part is analogous to the pure exhaustive approach. Regardless of error dropping below threshold, all combinations are considered as the initial metric. The corresponding learned metric that gives the lowest failure rate is returned as the final metric. Hence this approach is exhaustive in the sense of traversing all possible paths. However, the Tolerance Limit Heuristic is applied in order to prune the number of paths to be traversed. The algorithm that we propose for hybrid exhaustive selection is shown below.

*Hybrid Exhaustive Selection Algorithm:*

Given: Actual clusters of graphs, error threshold $t$, tolerance limit $L$

1. Identify all $m$ applicable components
2. For each component $D_i : i = 1$ to $m$
   (a) Do clustering in LearnMet with $D = D_i$, calculate $FR$ and $SR$
3. Use Tolerance Limit Heuristic to prune $s$ components that give $FR > L$
4. $j = 1$
   (a) Execute LearnMet with all possible combinations of $j$ components
   (b) For each combination record failure rate $FR$
   (c) $j = j + 1$
   (d) If $j \leq (m - s)$ go to step 4(a)
5. Output *final metric* with lowest failure rate $FR$

## 6 Comparison of Computational Complexity.

**6.1 Greedy Selection.** In greedy selection, we first need to execute LearnMet $m$ times in order to determine the individual failure rates of the $m$ components. In the next execution, we consider only the best and second best. Then, we consider only the best, second best and third best. Thus, in every subsequent execution we consider only the best combinations, not all possible combinations. Hence this involves at most $(m - 1)$ executions of LearnMet, in addition to the $m$ executions with single components. Thus the total number of executions is at most $m+(m-1)$. Thus greedy selection even in the worst case involves only $2m-1$ executions of the distance metric learning algorithm, thus providing a search space that is linear with respect to the number of components $m$. This gives a low computational complexity and hence faster learning.

**6.2 Exhaustive Selection.** In the exhaustive approach, the search space is exponential $(2^m - 1)$ in the number of subsets from $m$ components. In the first $m$ executions of LearnMet, each component would be considered individually. Then all possible combinations of 2 components would be considered The number of combinations of 2 components that can be made from $m$ components is $\binom{m}{2}$. Next, it would consider all possible combinations of 3, i.e, $\binom{m}{3}$ number of combinations. Each combination is selected as the initial metric in LearnMet and hence one combination corresponds to one execution of LearnMet. Thus for $m$ components, the total number of combinations considered is $\Sigma_{j=1}^{m}\binom{m}{j} = 2^m - 1$. This gives the total number of executions of LearnMet. Clearly, the exhaustive approach involves a huge number of executions and hence a huge learning complexity, which is not practical except for small values of $m$.

**6.3 Hybrid Greedy Selection.** The proposed hybrid greedy algorithm reduces the search space from exponential $(2^m - 1)$ to quadratic in the number of subsets from $m$ components. In particular, at most $\frac{m(m-1)}{2}$ executions of LearnMet are needed in the worst case. After considering $c$ components, the next step of the hybrid greedy algorithm involves executing LearnMet at most $m - c$ times. If the components are considered in descending order of accuracies, it is reasonable to expect that only a fraction of the $(m - c)$ executions will be sufficient to find a solution with $(c+1)$ components. To compute the Distance Metric $D$ or execute to LearnMet with $c \leq m$ components, it is expected that the execution time will be reduced proportionally by a factor of $\frac{c}{m}$ of the time needed using all $m$ components.

**6.4 Hybrid Exhaustive Selection.** In the hybrid exhaustive approach, the search space is reduced to approximately $2^{m-s}$ where $s$ is the number of components that get pruned out by the tolerance limit heuristic. In the first $m$ executions of LearnMet, each component is considered individually to determine its failure rate. Among these, $m-s$ components are retained after the pruning is done. Next, all possible combinations of 2 components would be considered from a total of $(m-s)$ components. The number of combinations of 2 components that can be made from $(m-s)$ components is $\binom{m-s}{2}$. Likewise, in the next few iterations all combinations of 3 components would be considered. Hence, the total number of combinations considered is: the first $m$ combinations using all components to determine failure rates and the next $\binom{m-s}{2} + \binom{m-s}{3} \ldots + \binom{m-s}{m-s}$ combinations with the remaining $(m-s)$ components. Thus, the total number of executions of LearnMet in hybrid exhaustive selection is $m + \Sigma_{j=2}^{m-s}\binom{m-s}{j}$. Even though this complexity is high, it is still likely to be less than that of the pure exhaustive approach if some components get pruned out. The hybrid exhaustive approach is therefore suitable for small values of $(m-s)$.

## 7 Evaluation of Accuracy and Efficiency.

Evaluation of the selection approaches is conducted with real data from the domain of Heat Treating of Materials [9] that motivated this research. Its main application is to learn distance metrics for clustering graphs 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. Learning efficiency is recorded as the number of epochs required for training and the training time in milliseconds [11].

**7.1 Data Description.** In order to proceed with the details of the experiments, we give a brief overview of the heat transfer curves and distance metrics that apply to them.

Figure 3 shows a heat transfer curve. It plots the heat transfer coefficient $h$ versus temperature $T$ of a material where the heat transfer coefficient measures the heat extraction capacity in a rapid cooling process called quenching [9]. Some regions on the graph are more significant than others because they correspond to physical phenomena in the domain. Boiling Point region $BP$ shows the temperature of the part being reduced to the boiling point of the cooling medium.

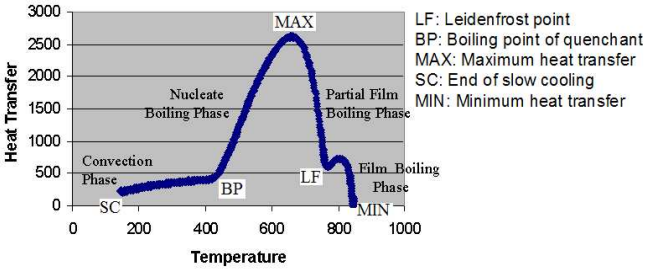


Figure 3: Heat Transfer Curve

Leidenfrost Point $LF$ denotes the breaking of the vapor blanket resulting in rapid cooling. Slow Cooling region $SC$ is where the quenching process ends [9]. Maximum and minimum heat transfer regions, $MAX$ and $MIN$ respectively, are statistical distinguishing factors.

Different metrics from the literature can be used to compare these graphs. Euclidean distance $D_{Euclidean}$ compares them based on absolute position of points. Statistical distances $D_{Max}$, $D_{Min}$ and $D_{Mean}$ compare them based on the statistical observations of maximum value, minimum value and mean value of the dependent variable, i.e., heat transfer coefficient. In addition, we define *Critical Distances* [12], $D_{LF}$, $D_{BP}$ and $D_{SC}$ as the distances between the respective Leidenfrost points, Boiling points and Slow Cooling points of the graphs respectively. With this discussion, we give the details of our experimental evaluation.

**7.2 Experimental Details.** The experiments below show the impact of selecting the components on clustering accuracy and learning efficiency. Several experiments have been conducted [11], a summary of which is presented here. The number of graphs 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 number was observed to be a good setting from earlier experiments [11]. We conducted experiments with different values for error threshold and maximum number of epochs. For the experiments shown here, error threshold $t$ is maintained at 0.1 and maximum number of epochs is constant at 1000. For the hybrid approaches, we use a tolerance limit of $L = 0.75$, i.e., components giving more than 75% error (less than 25%) accuracy individually are not considered useful. The test set used is of size $G = 15$, with $k = 3$ actual clusters provided. The components applicable to the graphs as identified with the help of domain experts are: $D_{SC}$, $D_{Euclidean}$, $D_{Mean}$, $D_{Max}$, $D_{LF}$, $D_{BP}$, and $D_{Min}$. We alter the seeds in the clustering algorithm

| Components | Epochs | Time (ms) | Accuracy |
|---|---|---|---|
| D_Euclidean | 1 | 132 | 0.7247 |
| D_SC | 1 | 99 | 0.2473 |
| D_LF | 1 | 76 | 0.5867 |
| D_BP | 1 | 85 | 0.6367 |
| D_Max | 1 | 97 | 0.6734 |
| D_Mean | 1 | 83 | 0.5 |
| D_Min | 1 | 79 | 0.2167 |
| D_Euclidean, D_Max | 1000 | 4125 | 0.7733 |
| D_Euclidean, D_Max, D_BP | 1000 | 4236 | 0.8867 |
| **D_Euclidean, D_Max, D_BP, D_LF** | **375** | **2726** | **0.92** |

Figure 4: Greedy Selection

for randomization.

The results of our experiments are shown below for greedy, exhaustive, hybrid greedy and hybrid exhaustive approaches. For each experiment, we record the components used, the number of epochs for training, the training time in milliseconds and the accuracy of the learned metric over the test set. Each experiment shows the average of four experiments altering clustering seeds in LearnMet for randomization.

**7.2.1 Greedy Selection.** In applying the greedy approach for selection of components, we first executed LearnMet with each individual component, $D_{Euclidean}$, $D_{SC}$, $D_{LF}$, $D_{BP}$, $D_{Max}$, $D_{Mean}$ and $D_{Min}$. Then based on their individual accuracies, we used a combination of the best two components, i.e., $D_{Euclidean}$ and $D_{Max}$. This was followed by a combination of the best three, i.e., $D_{Euclidean}$, $D_{Max}$ and $D_{BP}$. Finally a combination of the best four, i.e., $D_{Euclidean}$, $D_{Max}$, $D_{BP}$ and $D_{LF}$ gave failure rate below threshold. The observations are shown in Figure 4. We highlight the combination that gives the greatest accuracy.

It is seen that for seven components, the greedy approach requires only ten executions of LearnMet before converging to error below threshold. The convergence occurs in as few as 375 epochs for a combination of the four best components. The final metric in this case yields a clustering accuracy of 0.92, i.e., 92% over the test set. Hence this approach is efficient and yet gives clustering accuracy above the required threshold of 90%. However, it does not consider all prior combinations of fewer components before moving on to a combination with more components. Hence it is not guaranteed to find a solution with the minimal number of components, although it converges quickly to a fairly simple metric, i.e., favoring few components with high accuracies.

**7.2.2 Exhaustive Selection.** In the exhaustive approach, we consider all possible combinations of the seven components, namely, $D_{Euclidean}$, $D_{SC}$, $D_{LF}$, $D_{BP}$, $D_{Max}$, $D_{Mean}$ and $D_{Min}$. Figure 5 shows executions with single components and with combinations of two components. The executions with combinations of three components and four components are shown in Figures 6 and 7 respectively. Figure 8 displays all the remaining combinations, i.e., with five or more components.

| Components | Epochs | Time (ms) | Accuracy |
|---|---|---|---|
| D_Euclidean | 1 | 128 | 0.7182 |
| D_SC | 1 | 84 | 0.2008 |
| D_LF | 1 | 68 | 0.6261 |
| D_BP | 1 | 77 | 0.5896 |
| D_Max | 1 | 84 | 0.6178 |
| D_Mean | 1 | 92 | 0.5221 |
| D_Min | 1 | 99 | 0.2074 |
| D_Euclidean, D_SC | 1000 | 4447 | 0.7235 |
| D_Euclidean, D_LF | 1000 | 4356 | 0.7435 |
| D_Euclidean, D_BP | 1000 | 4687 | 0.7432 |
| D_Euclidean, D_Max | 1000 | 3756 | 0.7352 |
| D_Euclidean, D_Mean | 1000 | 3984 | 0.7282 |
| D_Euclidean, D_Min | 1000 | 3902 | 0.7208 |
| D_SC, D_LF | 1000 | 3867 | 0.6223 |
| D_SC, D_BP | 1000 | 4561 | 0.5996 |
| D_SC, D_Max | 1000 | 4456 | 0.6198 |
| D_SC, D_Mean | 1000 | 3879 | 0.5376 |
| D_SC, D_Min | 1000 | 3882 | 0.2118 |
| D_LF, D_BP | 1000 | 3986 | 0.6895 |
| D_LF, D_Max | 1000 | 3829 | 0.6745 |
| D_LF, D_Mean | 1000 | 3902 | 0.6533 |
| D_LF, D_Min | 1000 | 4003 | 0.6278 |
| D_BP, D_Max | 1000 | 4173 | 0.6993 |
| D_BP, D_Mean | 1000 | 4256 | 0.6785 |
| D_BP, D_Min | 1000 | 3897 | 0.5945 |
| D_Max, D_Mean | 1000 | 4267 | 0.6203 |
| D_Max, D_Min | 1000 | 3896 | 0.6202 |
| D_Mean, D_Min | 1000 | 4672 | 0.5287 |

Figure 5: Exhaustive Selection with one and two components

It is found that none of the combinations of two components converge to error below threshold. Nor do any of the combinations of three components, although they require relatively longer training times. Among the combinations of four components, the only one that converges is with the components, $D_{Euclidean}$, $D_{LF}$, $D_{BP}$ and $D_{Max}$ (highlighted in Figure 7). These happen to be the components with individually highest accuracies. Among the combinations of five and six components also, the ones that converge are those that contain these 4 components. However, combinations containing $D_{Mean}$ also give fairly high accuracies, even if one among the four best components is missing. The best combinations of five and six components are highlighted in Figure 8. It is found that the best combination of six gives greater accuracy than the best combination of five which in turn gives greater accuracy than the best combination of four. The combination of all seven com-

| Components | Epochs | Time (ms) | Accuracy |
|---|---|---|---|
| D_Euclidean, D_SC, D_LF | 1000 | 4785 | 0.7532 |
| D_Euclidean, D_SC, D_BP | 1000 | 4897 | 0.7498 |
| D_Euclidean, D_SC, D_Max | 1000 | 4986 | 0.7513 |
| D_Euclidean, D_SC, D_Mean | 1000 | 4788 | 0.7362 |
| D_Euclidean, D_SC, D_Min | 1000 | 4789 | 0.7393 |
| D_Euclidean, D_LF, D_BP | 1000 | 4873 | 0.7101 |
| D_Euclidean, D_LF, D_Max | 1000 | 4889 | 0.7587 |
| D_Euclidean, D_LF, D_Mean | 1000 | 4788 | 0.7442 |
| D_Euclidean, D_LF, D_Min | 1000 | 4927 | 0.7455 |
| D_Euclidean, D_BP, D_Max | 1000 | 4993 | 0.7782 |
| D_Euclidean, D_BP, D_Mean | 1000 | 4896 | 0.7623 |
| D_Euclidean, D_BP, D_Min | 1000 | 4783 | 0.7589 |
| D_Euclidean, D_Max, D_Mean | 1000 | 4893 | 0.7689 |
| D_Euclidean, D_Max, D_Min | 1000 | 4826 | 0.7572 |
| D_Euclidean, D_Mean, D_Min | 1000 | 4893 | 0.7443 |
| D_SC, D_LF, D_BP | 1000 | 4787 | 0.6934 |
| D_SC, D_LF, D_Max | 1000 | 4674 | 0.6802 |
| D_SC, D_LF, D_Mean | 1000 | 4668 | 0.6634 |
| D_SC, D_LF, D_Min | 1000 | 4447 | 0.6387 |
| D_SC, D_BP, D_Max | 1000 | 4482 | 0.6997 |
| D_SC, D_BP, D_Mean | 1000 | 4467 | 0.6813 |
| D_SC, D_BP, D_Min | 1000 | 4563 | 0.5923 |
| D_SC, D_Max, D_Mean | 1000 | 4782 | 0.6321 |
| D_SC, D_Max, D_Min | 1000 | 4472 | 0.6285 |
| D_SC, D_Mean, D_Min | 1000 | 4437 | 0.5384 |
| D_LF, D_BP, D_Max | 1000 | 4356 | 0.7098 |
| D_LF, D_BP, D_Mean | 1000 | 4264 | 0.6872 |
| D_LF, D_BP, D_Min | 1000 | 4412 | 0.6899 |
| D_LF, D_Max, D_Mean | 1000 | 4352 | 0.6803 |
| D_LF, D_Max, D_Min | 1000 | 4376 | 0.6716 |
| D_LF, D_Mean, D_Min | 1000 | 4343 | 0.6623 |
| D_BP, D_Max, D_Mean | 1000 | 4662 | 0.7003 |
| D_BP, D_Max, D_Min | 1000 | 4261 | 0.6992 |
| D_BP, D_Mean, D_Min | 1000 | 4352 | 0.6798 |
| D_Max, D_Mean, D_Min | 1000 | 4271 | 0.6345 |

Figure 6: Exhaustive Selection with three components

| Components | Epochs | Time (ms) | Accuracy |
|---|---|---|---|
| D_Euclidean, D_SC, D_LF, D_BP | 1000 | 4567 | 0.7157 |
| D_Euclidean, D_SC, D_LF, D_Max | 1000 | 4452 | 0.7596 |
| D_Euclidean, D_SC, D_LF, D_Mean | 1000 | 4517 | 0.7478 |
| D_Euclidean, D_SC, D_LF, D_Min | 1000 | 4332 | 0.7423 |
| D_Euclidean, D_SC, D_BP, D_Max | 1000 | 4320 | 0.7552 |
| D_Euclidean, D_SC, D_BP, D_Mean | 1000 | 4451 | 0.7432 |
| D_Euclidean, D_SC, D_BP, D_Min | 1000 | 4402 | 0.7367 |
| D_Euclidean, D_SC, D_Max, D_Mean | 1000 | 4467 | 0.7338 |
| D_Euclidean, D_SC, D_Max, D_Min | 1000 | 4434 | 0.7229 |
| D_Euclidean, D_SC, D_Mean, D_Min | 1000 | 4427 | 0.7207 |
| **D_Euclidean, D_LF, D_BP, D_Max** | **459** | **2664** | **0.9234** |
| D_Euclidean, D_LF, D_BP, D_Mean | 1000 | 4456 | 0.7768 |
| D_Euclidean, D_LF, D_BP, D_Min | 1000 | 4567 | 0.7568 |
| D_Euclidean, D_LF, D_Max, D_Mean | 427 | 2578 | 0.9067 |
| D_Euclidean, D_LF, D_Max, D_Min | 1000 | 4456 | 0.7621 |
| D_Euclidean, D_LF, D_Mean, D_Min | 1000 | 4462 | 0.7524 |
| D_Euclidean, D_BP, D_Max, D_Mean | 433 | 2589 | 0.9103 |
| D_Euclidean, D_BP, D_Max, D_Min | 1000 | 4462 | 0.7536 |
| D_Euclidean, D_BP, D_Mean, D_Min | 1000 | 4502 | 0.7632 |
| D_Euclidean, D_Max, D_Mean, D_Min | 1000 | 4397 | 0.7499 |
| D_SC, D_LF, D_BP, D_Max | 1000 | 4698 | 0.7052 |
| D_SC, D_LF, D_BP, D_Mean | 1000 | 4724 | 0.7045 |
| D_SC, D_LF, D_BP, D_Min | 1000 | 4762 | 0.6892 |
| D_SC, D_LF, D_Max, D_Mean | 1000 | 4677 | 0.7004 |
| D_SC, D_LF, D_Max, D_Min | 1000 | 4703 | 0.6803 |
| D_SC, D_LF, D_Mean, D_Min | 1000 | 4772 | 0.6724 |
| D_SC, D_BP, D_Max, D_Mean | 1000 | 4687 | 0.6883 |
| D_SC, D_BP, D_Max, D_Min | 1000 | 4679 | 0.6834 |
| D_SC, D_BP, D_Mean, D_Min | 1000 | 4722 | 0.6745 |
| D_SC, D_Max, D_Mean, D_Min | 1000 | 4711 | 0.6824 |
| D_LF, D_BP, D_Max, D_Mean | 1000 | 4261 | 0.7191 |
| D_LF, D_BP, D_Max, D_Min | 1000 | 4683 | 0.7003 |
| D_LF, D_BP, D_Mean, D_Min | 1000 | 4719 | 0.6905 |
| D_LF, D_Max, D_Mean, D_Min | 1000 | 4774 | 0.6893 |
| D_BP, D_Max, D_Mean, D_Min | 1000 | 4789 | 0.6815 |

Figure 7: Exhaustive Selection with four components

| Components | Epochs | Time (ms) | Accuracy |
|---|---|---|---|
| D_Euclidean, D_SC, D_LF, D_BP, D_Max | 347 | 2478 | 0.9269 |
| D_Euclidean, D_SC, D_LF, D_BP, D_Mean | 1000 | 5001 | 0.8009 |
| D_Euclidean, D_SC, D_LF, D_BP, D_Min | 1000 | 5023 | 0.7682 |
| D_Euclidean, D_SC, D_LF, D_Max, D_Mean | 1000 | 5017 | 0.8086 |
| D_Euclidean, D_SC, D_LF, D_Max, D_Min | 1000 | 5124 | 0.7623 |
| D_Euclidean, D_SC, D_LF, D_Mean, D_Min | 1000 | 5028 | 0.7734 |
| D_Euclidean, D_SC, D_BP, D_Max, D_Mean | 1000 | 5092 | 0.8132 |
| D_Euclidean, D_SC, D_BP, D_Max, D_Min | 1000 | 5102 | 0.7683 |
| D_Euclidean, D_SC, D_BP, D_Mean, D_Min | 1000 | 5006 | 0.7529 |
| D_Euclidean, D_SC, D_Max, D_Mean, D_Min | 1000 | 5082 | 0.7426 |
| **D_Euclidean, D_LF, D_BP, D_Max, D_Mean** | **302** | **2014** | **0.9412** |
| D_Euclidean, D_LF, D_BP, D_Max, D_Min | 348 | 2461 | 0.9254 |
| D_Euclidean, D_LF, D_BP, D_Mean, D_Min | 1000 | 5027 | 0.8334 |
| D_Euclidean, D_LF, D_Max, D_Mean, D_Min | 1000 | 5111 | 0.8262 |
| D_Euclidean, D_BP, D_Max, D_Mean, D_Min | 1000 | 5004 | 0.8345 |
| D_SC, D_LF, D_BP, D_Max, D_Mean | 1000 | 5002 | 0.7802 |
| D_SC, D_LF, D_BP, D_Max, D_Min | 1000 | 5342 | 0.7023 |
| D_SC, D_LF, D_BP, D_Mean, D_Min | 1000 | 5045 | 0.7011 |
| D_SC, D_LF, D_Max, D_Mean, D_Min | 1000 | 5267 | 0.7002 |
| D_SC, D_BP, D_Max, D_Mean, D_Min | 1000 | 5189 | 0.6985 |
| D_LF, D_BP, D_Max, D_Mean, D_Min | 1000 | 5297 | 0.7015 |
| **D_Euclidean, D_SC, D_LF, D_BP, D_Max, D_Mean** | **486** | **3067** | **0.9476** |
| D_Euclidean, D_SC, D_LF, D_BP, D_Max, D_Min | 495 | 3325 | 0.9203 |
| D_Euclidean, D_SC, D_LF, D_BP, D_Mean, D_Min | 1000 | 5672 | 0.8567 |
| D_Euclidean, D_SC, D_LF, D_Max, D_Mean, D_Min | 1000 | 5545 | 0.8646 |
| D_Euclidean, D_SC, D_BP, D_Max, D_Mean, D_Min | 1000 | 5672 | 0.8532 |
| D_Euclidean, D_LF, D_BP, D_Max, D_Mean, D_Min | 503 | 3367 | 0.9327 |
| **D_Euclidean, D_SC, D_LF, D_BP, D_Max, D_Mean, D_Min** | **556** | **3674** | **0.9512** |

Figure 8: Exhaustive Selection with five or more components

| Components | Epochs | Time (ms) | Accuracy |
|---|---|---|---|
| D_Euclidean | 1 | 128 | 0.7265 |
| D_SC | 1 | 95 | 0.2265 |
| D_LF | 1 | 83 | 0.5777 |
| D_BP | 1 | 78 | 0.6534 |
| D_Max | 1 | 84 | 0.6544 |
| D_Mean | 1 | 90 | 0.5345 |
| D_Min | 1 | 81 | 0.1167 |
| D_Euclidean, D_Max | 1000 | 3978 | 0.7645 |
| D_Euclidean, D_BP | 1000 | 3990 | 0.7757 |
| D_Euclidean, D_LF | 1000 | 4520 | 0.7434 |
| D_Euclidean, D_Mean | 1000 | 4672 | 0.7956 |
| D_Max, D_BP | 1000 | 4267 | 0.6867 |
| D_Max, D_LF | 1000 | 3867 | 0.6782 |
| D_Max, D_Mean | 1000 | 4178 | 0.6162 |
| D_BP, D_LF | 1000 | 4209 | 0.6544 |
| D_BP, D_Mean | 1000 | 4256 | 0.6688 |
| D_LF, D_Mean | 1000 | 4378 | 0.6002 |

Figure 9: Hybrid Greedy Selection with one and two components

ponents gives the greatest accuracy of all and is also highlighted. However, we find that simply adding components to the metric does not lead to convergence. The good components, i.e., the ones with individually high accuracies do have a significant impact on improving the accuracy of the learned metric. Exhaustive selection goes through a much higher number of executions than greedy selection. It does however find a solution that gives a learned metric with the highest accuracy.

**7.2.3 Hybrid Greedy Selection.** In hybrid greedy selection, the first 7 executions are conducted with the individual components $D_{Euclidean}$, $D_{SC}$, $D_{LF}$, $D_{BP}$, $D_{Max}$, $D_{Mean}$ and $D_{Min}$. Then the two components $D_{SC}$ and $D_{Min}$ that individually give errors greater than the tolerance limit $L = 0.75$ are pruned out. For the remaining five components, execution proceeds with combinations of two, three, four and all five components respectively. The components are considered in descending order of accuracies. Figure 9 shows the results for executions with one and two components. The remaining ones, i.e., with three or more components are shown in Figure 10.

We find that convergence does not occur for any of the combinations of two or three components. The first combination that converges to error below threshold is the one with the four components $D_{Euclidean}$, $D_{LF}$, $D_{BP}$ and $D_{Max}$. This is highlighted in Figure 10. At this point we do not continue any further. It is thus found that hybrid greedy selection requires far less executions than exhaustive selection but relatively more than greedy. It considers all prior combinations of fewer components before adding components to the metric.

| Components | Epochs | Time (ms) | Accuracy |
|---|---|---|---|
| D_Euclidean, D_Max, D_BP | 1000 | 5023 | 0.8523 |
| D_Euclidean, D_Max, D_LF | 1000 | 5099 | 0.8256 |
| D_Euclidean, D_Max, D_Mean | 1000 | 4978 | 0.8111 |
| D_Euclidean, D_BP, D_LF | 1000 | 4902 | 0.8325 |
| D_Euclidean, D_BP, D_Mean | 1000 | 5015 | 0.8097 |
| D_Euclidean, D_LF, D_Mean | 1000 | 5067 | 0.8023 |
| D_Max, D_BP, D_LF | 1000 | 4798 | 0.7867 |
| D_Max, D_BP, D_Mean | 1000 | 4867 | 0.7775 |
| D_Max, D_LF, D_Mean | 1000 | 4773 | 0.7623 |
| D_BP, D_LF, D_Mean | 1000 | 4782 | 0.7601 |
| **D_Euclidean, D_Max, D_BP, D_LF** | **367** | **3143** | **0.9123** |

Figure 10: Hybrid Greedy Selection with three or more components

| Components | Epochs | Time (ms) | Accuracy |
|---|---|---|---|
| D_Euclidean | 1 | 112 | 0.7111 |
| D_SC | 1 | 91 | 0.1856 |
| D_LF | 1 | 87 | 0.6413 |
| D_BP | 1 | 80 | 0.6478 |
| D_Max | 1 | 77 | 0.6395 |
| D_Mean | 1 | 83 | 0.6002 |
| D_Min | 1 | 85 | 0.1967 |
| D_Euclidean, D_LF | 1000 | 4067 | 0.7468 |
| D_Euclidean, D_BP | 1000 | 3899 | 0.7745 |
| D_Euclidean, D_Max | 1000 | 4178 | 0.7832 |
| D_Euclidean, D_Mean | 1000 | 3888 | 0.6777 |
| D_LF, D_BP | 1000 | 4576 | 0.6834 |
| D_LF, D_Max | 1000 | 4234 | 0.6956 |
| D_LF, D_Mean | 1000 | 4207 | 0.7002 |
| D_BP, D_Max | 1000 | 4389 | 0.6911 |
| D_BP, D_Mean | 1000 | 4686 | 0.6906 |
| D_Max, D_Mean | 1000 | 3976 | 0.6934 |

Figure 11: Hybrid Exhaustive Selection with one and two components

Hence it finds a solution with the fewest number of components that give the desired level of accuracy.

**7.2.4 Hybrid Exhaustive Selection.** In the hybrid exhaustive selection approach, after the first seven executions with all components, i.e., $D_{Euclidean}$, $D_{SC}$, $D_{LF}$, $D_{BP}$, $D_{Max}$, $D_{Mean}$ and $D_{Min}$, the components $D_{SC}$ and $D_{Min}$ are discarded due to errors above tolerance limit. Then the executions proceed as in the pure exhaustive approach. The results for executions with one and two components are shown in Figure 11, while those with three or more components are shown in Figure 12.

It is observed that convergence does not occur for combinations having less than four components. The only combination of four that converges is with the components $D_{Euclidean}$, $D_{LF}$, $D_{BP}$ and $D_{Max}$ as highlighted in Figure 12. For combinations with five components also, the ones that converge are the ones contain-

| Components | Epochs | Time (ms) | Accuracy |
|---|---|---|---|
| D_Euclidean, D_LF, D_BP | 1000 | 4996 | 0.7991 |
| D_Euclidean, D_LF, D_Max | 1000 | 4894 | 0.8547 |
| D_Euclidean, D_LF, D_Mean | 1000 | 5003 | 0.8623 |
| D_Euclidean, D_BP, D_Max | 1000 | 5268 | 0.8345 |
| D_Euclidean, D_BP, D_Mean | 1000 | 4786 | 0.8524 |
| D_Euclidean, D_Max, D_Mean | 1000 | 4676 | 0.8332 |
| D_LF, D_BP, D_Max | 1000 | 4578 | 0.8142 |
| D_LF, D_BP, D_Mean | 1000 | 4435 | 0.8002 |
| D_LF, D_Max, D_Mean | 1000 | 4689 | 0.7986 |
| D_BP, D_Max, D_Mean | 1000 | 4785 | 0.8126 |
| **D_Euclidean, D_LF, D_BP, D_Max** | **378** | **2867** | **0.9101** |
| D_Euclidean, D_LF, D_BP, D_Mean | 1000 | 5102 | 0.8234 |
| D_Euclidean, D_LF, D_Max, D_Mean | 1000 | 5123 | 0.8451 |
| D_Euclidean, D_BP, D_Max, D_Mean | 1000 | 5156 | 0.8265 |
| D_LF, D_BP, D_Max, D_Mean | 1000 | 5015 | 0.7195 |
| **D_Euclidean, D_LF, D_BP, D_Max, D_Mean** | **307** | **2005** | **0.9426** |

Figure 12: Hybrid Exhaustive Selection with three or more components

ing these four components. Among them, the one that gives highest clustering accuracy is highlighted. The overall best combination is the one with the five components $D_{Euclidean}$, $D_{LF}$, $D_{BP}$, $D_{Max}$ and $D_{Mean}$. This combination is found to give almost the same clustering accuracy as the one with all the seven components in the exhaustive approach. Hence it is likely to yield an optimal solution adhering to the goals of simplicity and accuracy.

**7.3 Discussion on Experiments.** We summarize the rssults of our experimental evaluation for the four selection approaches in terms of the most interesting observations as follows.

- Greedy selection requires 10 executions and yields a metric with four components that gives accuracy of 92%.
- Exhaustive selection needs totally 127 executions and learns the most accurate metric. This metric has all of the seven components and gives an accuracy of 95.12%.
- Hybrid greedy selection needs totally 28 executions and learns a metric with four components giving an accuracy of 91.23%.
- Hybrid exhaustive selection needs 33 executions and yields a metric with 5 components giving an accuracy of 94.26%.

In this evaluation, we find that greedy and hybrid greedy approaches both yield metrics with the fewest components, giving accuracy above the desired threshold of 90%. The exhaustive and hybrid exhaustive approaches both give metrics with almost equal accuracy in the best case. The exhaustive approach gives the most accurate metric but the difference between the accuracies of the metrics learned from exhaustive and hybrid exhaustive approaches is not much. However, the hybrid exhaustive approach is a lot faster and learns a metric with fewer components than the exhaustive approach.

## 8 Related Work.

In [6], an overview of distance metrics useful for similarity search in multimedia databases is presented. However, they do not learn a single metric encompassing various distance types. Tri-plots [10] provide a tool for multidimensional data mining using intrinsic dimensionality, cross correlations and cumulative distribution functions of pair-wise distances between n-dimensional objects. 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 graphs is similar.

Hinneburg et. al. [3] propose a learning method to find relative importance of dimensions for n-dimensional objects. Their focus is on dimensionality reduction. In [14] they learn which type of position-based distance is applicable for the given data starting from the general Mahalanobis distance. They do not consider other distance types besides position-based.

In [2], Das et. al. propose an algorithm to find the distance between time series. Two sequences are considered to be similar if there is a function using which a long subsequence of one can be approximately mapped to a long subsequence of the other. This definition of similarity allows outliers in the sequences, differences of scaling and variable rates of sampling. Chen and Ng [1] learn a new metric based on the marriage of Lp-Norm and Edit distance also for time-series data. This metric called ERP (Edit Distance with Real Penalty) satisfies metric properties and also local time shifting. However such distance measures are more suitable for continuously varying data where some of the properties involved are needed for comparison.

Zhou et. al. [15] 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 constituting an ensemble. In our context each distance metric could be viewed as a learner, thus in combining them we get an ensemble. Considering such approaches presents interesting future issues.

## 9 Conclusions.

In this paper we address the problem of distance metric learning for graphs that plot scientific functions. We define a metric as a weighted sum of components

where each component represents a feature of the graph. We consider four approaches for component selection namely, greedy, exhaustive, hybrid greedy and hybrid exhaustive. The weights are learned using our earlier approach LearnMet. Experimental evaluation is conducted with real data from Materials Science. It is found that all approaches have their pros and cons.

Greedy selection is the most efficient and learns a simple metric that gives accuracy above a desired threshold. Exhaustive selection is the most time-consuming but yields a metric with the highest accuracy. Hybrid greedy selection learns a metric with the fewest number of components that give accuracy greater the threshold. Hybrid exhaustive selection provides a good trade-off between simplicity and accuracy. It is more efficient than the exhaustive approach but less than the greedy and hybrid greedy approaches. Hence the preference of one approach over the other depends on the requirements of specific applications. Our ongoing work includes considering more approaches and conducting evaluation with data from other domains.

## References


[1] L. Chen and R.Ng, *On the Marriage of Lp-Norm and Edit Distance*, VLDB (Aug 2004), pp. 792–803.

[2] G. Das, D. Gunopulos and H. Mannila, *Finding Similar Time Series*, PKDD (Jun 1997), pp. 88–100.

[3] A. Hinneburg, C. Aggarwal and D. Keim, *What is the Nearest Neighbor in High Dimensional Spaces*, VLDB (Aug 2000), pp. 506–515.

[4] J. Han and M. Kamber, *Data Mining Concepts and Techniques*, Morgan Kaufmann Publishers, 2001.

[5] R. Johnsonbaugh and M. Schaefer, *Algorithms*, Pearson Education, 2004.

[6] D. Keim and B. Bustos, *Similarity Search in Multimedia Databases*, IEEE's ICDE (Mar 2004), pp. 873–874.

[7] J. MacQueen, *Some Methods for Classification and Analysis of Multivariate Observations*, Mathematical Statistics and Probability, 1 (1967), pp. 281-297.

[8] T. Mitchell, *Machine Learning*, WCB McGraw Hill, 1997.

[9] G. Stolz Jr., *Heat Transfer*, John Wiley and Sons, 1960.

[10] A. Traina, C. Traina, S. Papadimitriou and C. Faloutsos, *TriPlots: Scalable Tools for Multidimensional Data Mining*, ACM KDD (Aug 2001) pp. 184–193.

[11] A. Varde *Graphical Data Mining for Computational Estimation in Materials Science Applications*, Ph.D. Dissertation, Department of Computer Science, Worcester Polytechnic Institute, Worcester, MA, Aug 2006.

[12] A. Varde, E. Rundensteiner, C. Ruiz, M. Maniruzzaman and R. Sisson, *Learning Semantics-Preserving Distance Metrics for Clustering Graphical Data*, ACM KDD's MDM (Aug 2005) pp. 107–112.

[13] I. Witten and E. Frank, *Data Mining: Practical Machine Learning Algorithms with Java Implementations*, Morgan Kaufmann Publishers, 2000.

[14] E. Xing, A. Ng, M. Jordan and S. Russell, *Distance Metric Learning with Application to Clustering with Side Information*, NIPS (Dec 2003) pp. 503-512.

[15] Z. Zhou, J. Wu and W. Tang, *Ensembling Neural Networks: Many Could Be Better Than All*, Artificial Intelligence 137:1 (2002), pp. 239-263.