iCaseViz: Learning Case Similarities through Interaction with a Case Base Visualizer

Debarun Kar, Anand Kumar, Sutanu Chakraborti, and Balaraman Ravindran

Department of Computer Science and Engineering,
Indian Institute of Technology Madras, Chennai- 600036, India
{debarunk,anandkr,sutanuc,ravi}@cse.iitm.ac.in

Abstract. Since the principal assumption in case-based reasoning (CBR) is that "*similar problems have similar solutions*", learning a suitable similarity measure is an important aspect in CBR. However, learning case-case similarities is often a non-trivial task and involves significant amount of domain expertise. Most techniques that arrive at a pertinent similarity measure are often incomprehensible to the domain experts. These techniques also rarely enable the user to provide expert feedback which can then be utilized to develop better similarity measures. Our work attempts to bridge this knowledge gap by developing an iterative and interactive visualization framework called iCaseViz which learns the domain experts' notion of similarity by utilizing the user feedback. This work is different from similar work in other communities in the sense that it is tailored to cater to the needs of a system built primarily based on the CBR hypothesis. The case base visualizer demonstrated in this paper is also very efficient as it has insignificant delay during real-time user interaction on large case bases. We provide preliminary results on the efficiency of the visualizer and the effectiveness of our similarity learning algorithm on UCI datasets and a real world high dimensional case base.

1 Introduction

In the past decade the emergence of vast collections of data from various sources has resulted in researchers exploring different avenues of data analysis. Most of the datasets, like the astronomical, textual and social networks data, are high-dimensional which makes interpretation and analysis difficult, even for the domain experts. Different research communities have thus focussed on developing techniques to explore the huge amount of hidden information in the data. Statistical and machine learning techniques are often employed to gain meaningful insights about the data. However, the knowledge possessed by domain experts are mostly left unutilized in these kinds of information extraction processes. This is because it is difficult for them to interpret the data while manipulating certain unknown parameters. Similar issues are encountered in case base reasoning (CBR) when trying to mine meaningful information, like the identification of important features, from large and noisy case bases, like textual case bases.

Case base visualization is therefore important to make sense of the underlying representation of the data and to facilitate analysis. A domain expert can use the knowledge acquired from the visualization to reform his understanding of the data and make informed decisions regarding certain aspects of the data, like the identification of noisy cases and attributes in a case base. The importance of case base visualization and several CBR visualization techniques has been the topic of discussion in the past. However, most of these visualizers do not support interaction with the user, in the sense that the user cannot manipulate elements in the visualization to obtain a refined representation. This is mostly due the fact that real-time interaction with a visualizer is either very slow or not supported when dealing with data sets with large number of cases and attributes. This prevents the user from further analysis and gaining deeper insights about the data. Also, it is not always clear how to elicit user feedback. In this paper we present an application that allows the user to visualize and iteratively interact with it and then utilize the user feedback for effective and efficient multi-variate data analysis.

CBR systems are primarily built on the hypothesis that "*similar problems have similar solutions*". However, identifying the most appropriate similarity measure for a given case base is often a non-trivial task and requires significant amount of domain knowledge. The failure to encode a suitable similarity measure results in incorrect retrieval of cases. This, in turn, leads to poor quality solutions and the failure of a CBR system as a whole. Therefore, it is sometimes important to involve the domain experts in the process of explicitly specifying the similarity between two cases in a case base. However, reviewing the similarity between all pairs of cases as computed by a pre-determined similarity measure and modifying the incorrect similarities is an incredibly arduous, if not infeasible, task especially if the case base consists of a large number of cases. Also, the notion of similarity between cases cannot always be explicitly stated. The qualitative notion of similarity can be captured by showing two cases to the users and allowing them to manipulate a slider, specifying the extent of similarity on a normalized scale, say 0 to 1. However, doing this for every pair in a large case base is also laborious and time consuming. Moreover, the case base may contain noisy samples due to erroneous recording of past experiences. It is therefore important to identify and eliminate such noisy cases so as to form a case base which is representative of the problem solving domain under consideration. With the help of our visualizer and the user feedback we will attempt to learn a suitable similarity measure and also recognize noisy cases and eliminate them as part of the case base maintenance procedure.

In this paper, in order to learn a new set of similarities between cases we will learn an appropriate set of feature weights. We will see in the subsequent sections how we perform feature weighting to obtain an estimate of the relative impact of each feature on a particular target variable and also to compute the suitable global similarity between cases. For large number of features, this automated learning of feature weights not only spares the domain expert from manually encoding the weight of each feature, but it is also applicable in situations where the importance of every feature is unknown even to the experts.

The main motivation for our work comes while trying to find an important set of features and a suitable similarity measure for the soil nutrients' prediction task using a CBR system called InfoChrom [1]. In our previous work [2], we attempted to solve the problem of feature weighting by using alignment as a guiding measure. However, we were still unable to exploit the domain experts' knowledge of similarity between the cases present in the case base of InfoChrom. Due to the presence of a large number of cases (15167) in a high dimensional (176) feature space, the problem of manually encoding the relative similarities between cases became an uphill task. In this paper we attempt to address this problem by focussing on two important issues. Firstly, we want to emphasize the need for an interactive case base visualization tool which is efficient for large case bases. Secondly, we want to present a way to utilize user feedback from the visualizer to learn the domain experts' notion of similarity by finding a suitable set of feature weights. In Section 2, we discuss related work regarding visualization and similarity learning. We present the algorithm used by iCaseViz to revise the similarity measure in Section 3. In Section 4, we describe our application framework, its various components and the process of interaction with the visualizer. This is followed by preliminary experimental results and analysis on UCI data sets and the case base of InfoChrom in Section 5. Finally, we conclude by discussing how our work is different from a similar existing work and by describing a few possible extensions of our work in Section 6.

2 Related Work

Encoding a suitable similarity measure for a given data set has been an active research topic in several AI communities, like machine learning, case-based reasoning and data mining. Some papers focus on learning similarity measures by computing a set of relevant feature weights and combining the weighted feature level similarities to arrive at a global similarity measure. Our approach to learning similarity is based on the above idea. Research on feature weight learning primarily focusses on eliminating the *"curse of dimensionality"* problem for k-nearest neighbor (k-NN) based approaches by finding features which are more important than others for a particular prediction task [3–7]. An informative survey of similarity mechanisms in case base reasoning can be found in [8].

Past work on information visualization focuses mainly on visualizing high dimensional data with the help of Parallel Coordinates (PC) and Multi-Dimensional Scaling (MDS) plots. PC [9] draws features or coordinates as lines, parallelly and equidistant to each other (Fig. 1). A point in the n-dimensional coordinate space is drawn as a polyline in the PC plot, where the polyline connects all the parallel axes. Therefore, this parallel coordinates representation in 2-d space enables us to view and find complex patterns in multivariate datasets. However, the biggest drawback of this technique is that the PC plot becomes cluttered as the number of data points and the number of coordinates increase. This hinders the user from gaining any insightful patterns about the data being viewed. PC has been used in case base visualization in [10]. This work also addressed the problem of axes reordering by utilizing feature similarities. Falkman [11] presents a visualization tool that projects cases onto a three dimensional PC plot.

Work involving lower dimensional projections in visual data analytics mostly involves projection using MDS [12]. This includes several spring layout based visualizations for clustering problems as well as applications to manipulate certain parameters to modify the visualization [13–18]. The task of feature selection and distance function learning using 2-d projection has been most recently addressed in [19]. Other past research on case base visualization which are closely aligned with our work include [20, 21], which uses a force-directed graph drawing algorithm for visualizing an evolving case base. Cbtv [22] allows the user to visualize the effect of several similarity measures on a case base. Other related work on the same topic include [23–25].

3 Our Approach to Learning Similarity

Learning similarity with the help of our visualizer is an iterative process and the steps are explained in detail in Sections 3.1, 3.2, 3.3 and 3.4. Let us assume that we are given a set of N cases C_1, C_2,..., C_N, with the problem space P consisting of the values of q features F_1, F_2, ..., F_q and the solution space S consisting of the target variable T. Also, let the weight of feature F_i be w_i $(1 \leq i \leq q)$, where $0 \leq w_i \leq 1$. Then we calculate the dissimilarity in the problem space between two cases C_a and C_b as the weighted sum of their feature space dissimilarities:

$$DisSimP(C_a, C_b, \boldsymbol{w}) = \sum_{i=1}^{q} w_i * DisSimF(C_{ai}, C_{bi}) \tag{1}$$

where, *DisSimF*(C_{ai},C_{bi}) denotes the dissimilarity between the corresponding values of feature F_i for the cases $\mathrm{C}_a$and C_b, and is computed as follows:

$$DisSimF(C_{ai}, C_{bi}) = \frac{DistF(C_{ai}, C_{bi}) - Min(DistF_i)}{Max(DistF_i) - Min(DistF_i)} \tag{2}$$

Here, *DistF*(C_{ai},C_{bi}) denotes the city-block distance between the values C_{ai} and C_{bi}. *Max*$(DistF_i)$ and *Min*$(DistF_i)$ are the maximum and minimum distances between all pairs of values for feature F_i respectively. The target space dissimilarity between any two cases is calculated in the same way and is represented as $\mathrm{DisSimS}(\mathrm{C}_a,\mathrm{C}_b)$. Please note that our formulation of feature space dissimilarity is such that the feature space similarity between two cases C_{ai} and C_{bi} for feature F_i will be (1-*DisSimF*(C_{ai},C_{bi})). The overall similarity between two cases can then be computed as the weighted sum of these feature space similarities. In our application, we work with dissimilarities between cases so that we can use in-built Matlab functions which takes a dissimilarity matrix as an argument, to perform the projection in Section 3.1.

3.1 Projecting the Case Base in Two Dimensions

We use classical multi-dimensional scaling (MDS) to project the high-dimensional case base onto two dimensions and show the output to the user in the form of a scatter plot. The MDS algorithm is provided with a dissimilarity matrix containing the dissimilarities between case pairs, computed in the original high dimensional space based on the current measure of dissimilarity. It then attempts to find a projection in lower dimensions that best preserves the relative dissimilarity between the cases. We use the MDS plot to help the user visualize the relative dissimilarities between the cases in the original high dimensional space. We also color the points in this MDS plot for effective visualization and analysis. The coloring scheme for the cases is discussed in Section 4.1.

3.2 User Action

The application provides the user with the capability to select any case from the scatter plot and move it, thus changing the relative similarities between the selected case and other cases in the case base. The facility to select multiple cases together and move them in the plot is also provided. This will affect the relative dissimilarities between the selected set of cases and all other cases in the case base, while the relative dissimilarities between the untouched cases remains the same. More details about the features that are incorporated in the application can be found in Section 4.1.

3.3 Capturing the Experts' Notion of Similarity

Once the user has changed the relative dissimilarities between the cases, we obtain the new locations of the cases in the 2-d scatter plot and compute the Euclidean distance between every pair of cases in the case base. This gives us an estimate of the domain experts' notion of similarity between the cases.

3.4 Feature Weighting and Similarity Learning

Once the user has modified the relative dissimilarities between the cases, the system solves an optimization problem to learn an appropriate feature weight vector $\boldsymbol{w}$. These relative importance values for each feature denoted as w_i $(1 \leq i \leq q)$ are then used to derive new dissimilarity values with the help of Eqn. 1.

In order to obtain the revised feature weight set $\boldsymbol{w^t}$ at time t, we solve the optimization problem as specified by Equation 3. For every pair of cases C_i and C_j, we take the squared difference between the user specified dissimilarities $DisSimP^{user}(C_i, C_j)$ as obtained by the users' modification of the scatter plot and the dissimilarities $DisSimP(C_i, C_j, \boldsymbol{w})$ in the high dimensional space obtained for a particular weight vector $\boldsymbol{w}$. Alignment measures the extent to which the similarity hypothesis holds good in a particular case base. Local alignment measures this property of the case base in the neighbourhood of a particular case while global alignment is measured across the entire case base. Now, consider the scenario where the user moves all cases which are highly similar both in the problem and solution spaces to a single point. This is clearly not desirable even though this increases the local as well as the global alignment. Thus the need to preserve the structure in the original data in terms of inter-case similarities might pose a threat to the goal of improving the alignment. We devise the objective function to capture the tradeoff between these conflicting requirements. Now, consider another situation where the user moves two cases, which are highly similar both in the problem and solutions spaces, too far apart from each other. This will decrease the alignment and is also undesirable. To discourage the user movements as explained in the above two scenarios, we give weights to each of the squared difference terms in Equation 3. This is an important design step in our application. To do this, we first compute the complexity between two cases C_i and C_j as the product of their problem and solution side dissimilarities (Equation 6), normalize it (Equation 5) and replace the value thus obtained in Equation 4. The hyperbolic tangent function is used in Equation 4 as a smoothing function. The trade-off term will ensure that the squared difference term in Equation 3 receives less importance if the value of complexity between two cases is low and vice versa. Thus, if the user moves two cases, which are already highly similar both in the problem and solutions spaces, too close or too far apart from each other, the tradeoff term (Eqn. 4) will ensure that the movement receives less weight. Therefore, by incorporating the tradeoff term, random movements in the case space is prevented and the global structure of the case base is maintained in accordance with the CBR hypothesis.

$$\boldsymbol{w^t} = \underset{\boldsymbol{w}}{argmin}(\sum_{i \le j \le N}(tanh(Comp_{ij}^{norm})*(DisSimP^{user}(C_i, C_j) - DisSimP(C_i, C_j, \boldsymbol{w}))^2)) \tag{3}$$

subject to $\mathrm{w}_i \geq 0$, $(1 \leq i \leq q)$

$$tanh(Comp_{ij}^{norm}) = (2/(1 + \exp(-2 * Comp_{ij}^{norm}))) - 1 \tag{4}$$

$$Comp_{ij}^{norm} = \frac{Comp_{ij} - Min(Comp_{ij})}{Max(Comp_{ij}) - Min(Comp_{ij})} \tag{5}$$

$$Comp_{ij} = DisSimP(C_i, C_j) * DisSimS(C_i, C_j) \tag{6}$$

4 The iCaseViz Application Framework and Its Components

This section describes the different components of our application and the features provided to the user. We also discuss the implementation details and how we obtain fast responses while loading and interacting with the data as compared to similar existing tools. It is divided into two sections: the visualizer (Sec. 4.1) and the analyzer (Sec. 4.2).

4.1 The Visualizer

The visualizer portion of the application is written in C++ using the Qt open source library. It has two main modules: the Parallel Coordinates (PC) visualizer

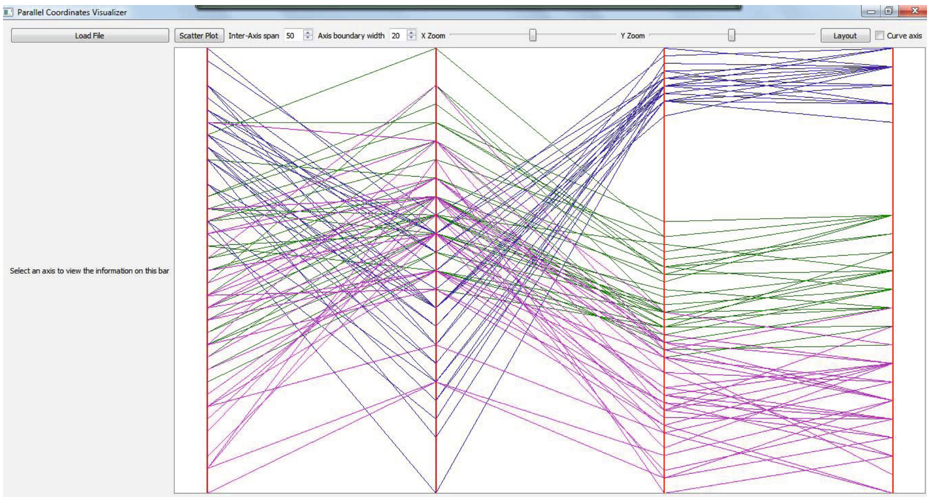

Fig. 1. The Parallel Coordinates visualizer

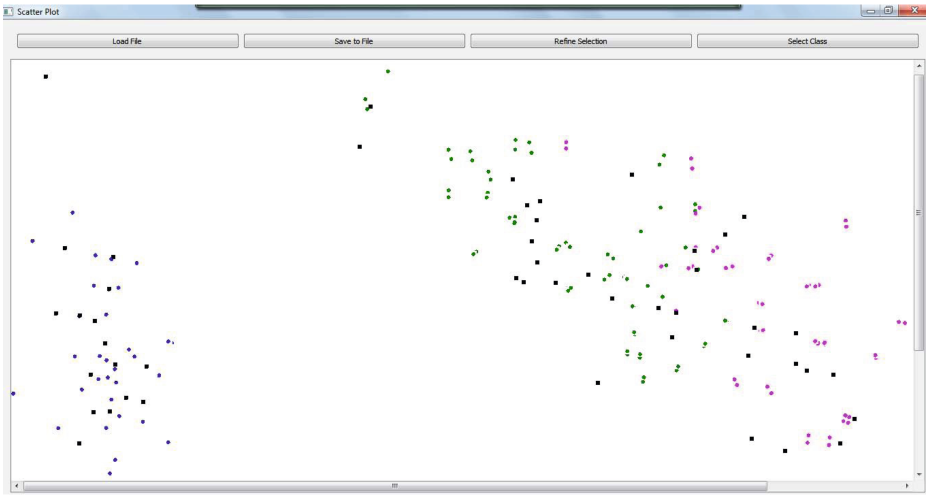

Fig. 2. The MDS Scatter Plot visualizer

(Figure 1) and the Scatter Plot visualizer (Figure 2). The PC module allows one to reposition axis, change the spacing between axes, zoom along the X and Y axes independently and scroll horizontally and vertically. It is also capable of displaying axis specific information like the axis name when one selects an axis. The Scatter Plot module displays an interactive two dimensional plot. One can move the cases around and save the resulting plot data to a file. The Scatter Plot gives the user multiple options to select the cases:

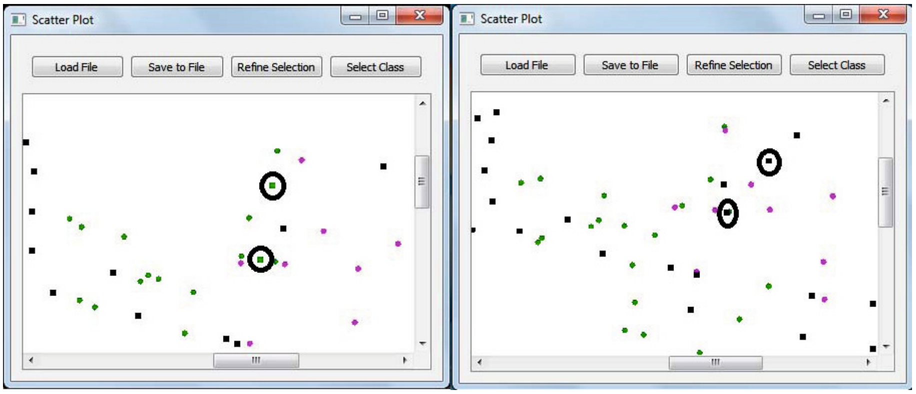

Fig. 3. A zoomed view of the MDS Scatter Plot visualizer. In the left view, two test cases (green in the visualizer) are circled to show that these were misclassified (into the class represented by the color green). The right view shows the MDS plot after a few iterations. The same circled cases (represented in black color in the visualizer) have now been correctly classified.

- Click on a case and it gets highlighted.
- Hold down the CTRL key and click on multiple cases to select.
- Select by dragging a selection box around a group of cases.
- Select all cases that are of the same colour.
- Select cases of only a given colour from currently selected set of cases.

Our 2-dimensional Scatter Plot has an additional feature in terms of the projection, which is missing from the earlier work on the same topic. In addition to the cases in the case base, we also project the test cases onto this 2-d space. The Scatter Plot visualizer represents cases in the case base as circles and the test cases as squares. The cases in the case base are colored both in the PC and Scatter Plot visualizer and their color is determined by the solution side similarity between the cases. Test cases whose solutions are satisfactory to the user are colored black while the other cases are colored according to the color of the case whose solution is most similar to the proposed solution of that case. For example, in the case of classification tasks, each class can be represented by a particular color as shown in Figures 1 and 2, which displays the Iris dataset from the UCI repository. The correctly classified test samples are colored black while the incorrectly classified test cases has the same color as the class it has been classified into. This enhances the visualization by providing additional information to the domain expert in the form of relative dissimilarities between the cases in the case base and the test cases. He gets an idea about the quality of the current similarity metric by not only viewing the relative positions of the cases but also the quality of the solutions produced for the test cases as can be seen by the color of a particular case. However, the test cases are only for viewing purpose and cannot be moved around in the scatter plot. Note that even if the expert is aware of the quality of solutions of the test cases, improving it cannot be enforced by arbitrarily moving the cases around. This is because of the tradeoff term in Section 3.4. So the cases in the case base have to be moved around so that some latent notion of similarity is captured. Hence this is a valuable technique even though the test data is shown to the user. Two zoomed views of the MDS Scatter Plot are given in Figure 3. The view on the left highlights two misclassified test cases and the right view highlights the same cases after a few iterations, when they have been correctly classified. Note that one iteration corresponds to a single user interaction with the MDS plot. This figure also displays the various components as tabs which the user can use during the interaction. The two visualizer modules are linked to each other. So when the user selects case(s) on the Scatter Plot, the corresponding polyline(s) on the PC plot is highlighted. Our PC visualizer thus provides a consolidated view of the entire case base and cases of interest across multiple dimensions and aids the domain expert in his interaction with the MDS Scatter Plot. Discussion of the implementation details of the two visualization components is provided below.

Parallel Coordinates Visualizer: The PC Visualizer reads data in CSV format. The first row is expected to be the headings row, for example the names of the features. Each subsequent row is expected to have an extra column that gives the quality of the solution for the corresponding case, represented by a particular colour. Figure 4 shows the sequence of activities that lead to the image getting drawn on screen. Once the data is loaded, a layout is created. The height of the layout is determined by the axis that has the largest range. The inter-axis interval and the number of axis determine the length of the canvas. This canvas is logically split into canonical rectangles the size of the viewing area (dimensions of the viewing windows scaled according to zoom factors). This means the visible rectangle on screen (determined by zoom setting and position of the scroll bars) can be made of at most four adjacent canonical rectangles. The canonical rectangles' dimensions change when the view is zoomed in or out. For large datasets, to make the application responsive, these canonical rectangles are cached. This allows the application to save time spent on repetitive drawing.

MDS Plot Visualizer: The scatter plot is realized using the capabilities of QGraphicsScene and QGraphicsView. The Qt Graphics View framework allows

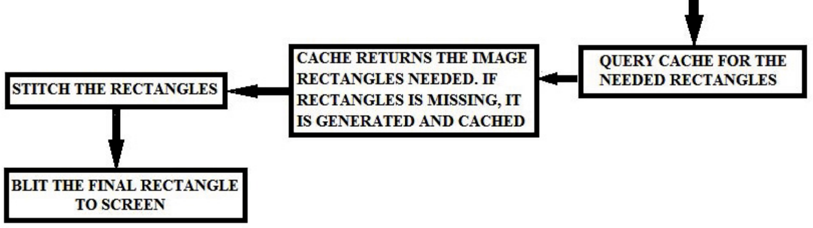
DETERMINE CURRENT VIEW RECTANGLE
QUERY TILE MANAGER FOR THE RECTANGLE
COMPUTE THE CANONICAL RECTANGLES NEEDED
QUERY CACHE FOR THE NEEDED RECTANGLES
CACHE RETURNS THE IMAGE RECTANGLES NEEDED. IF RECTANGLES IS MISSING, IT IS GENERATED AND CACHED
STITCH THE RECTANGLES
BLIT THE FINAL RECTANGLE TO SCREEN

Fig. 4. Sequence of activities for the Parallel Coordinates visualizer

the developer to describe a scene in terms of its constituent items. The framework allows for easy implementation of user interaction with object like selection and moving objects through Qt's event system.

In our implementation, the Scatter Plot modules load data from the files into a QGraphicsScene. This scene is attached to a view that displays the contents of the scene. The Scatter Plot is linked to the PC visualizer through the event system. Whenever the user selects cases in the scatter plot, a signal is passed to the PC module indicating the change in status. This in turn marks the appropriate polyline as selected, resulting in a change in its colour. The image cache is purged and the view is redrawn to show the change.

4.2 The Analyzer

The analyzer module uses Matlab to perform the data analysis task. This involves, among other things, computing the dissimilarity between pairs of cases and finding the most similar cases to suggest a solution. We have used in-built Matlab functions to perform MDS. To solve the optimization problem in Equation 3, we use the default solver (SDPT3) in CVX, a Matlab-based convex optimization tool for specifying and solving convex programs [26, 27]. We have found that SDPT3 is slow when we are working with large case bases and we are currently looking for fast solvers to enhance the performance of our application.

5 Evaluation

All our experiments have been conducted on an Intel Core i5 processor (M450 2.40 GHz) with 4GB RAM and 64-bit Windows 7 operating system. In our experiments we use k-NN based retrieval strategy to propose a solution for a test case. We calculate its dissimilarity with all the other cases in the case base using Equation 1. Then we obtain the solution for the test sample by taking a weighted average of the solutions of its k nearest neighbors, with the problem side dissimilarities acting as weights. Please note that we have considered $\mathrm{w}_i = 1$ $(1 \leq i \leq q)$ when no feature weighting is employed while calculating the dissimilarity between the cases.

5.1 Datasets

To demonstrate the efficiency of the visualizer and the effectiveness of our weight learning algorithm, we experimented on classification data sets from the UCI machine learning repository and a subset of the case base of InfoChrom. The InfoChrom case base originally contains 15167 cases represented in terms of 176 features and we have to predict the values of 15 target variables. In this paper we provide prediction results for one target variable with a subset of 250 cases in the original 176 dimensional feature space. We averaged the performance results over 5 random train-test splits with 70% of the original data used as the case base and the rest form the test cases. All results are reported with the value of k set to 3. For the data sets referred to in this paper, a brief

Table 1. Characteristics of the UCI data sets used for evaluation purposes

Data set	Kind of Prediction Task	Number of Features (all continuous)	Number of Classes (for Classification) / Range of the target variable (for Regression)
Iris	Classification	4	3
Glass Identification	Classification	9	7
Waveform-21	Classification	21	3
InfoChrom Case Base	Regression	176	26-246

description about the number and types of features and target variables are shown in Table 1. We measured performance in terms of classification accuracy and percentage error for classification and regression tasks respectively. Due to the lack of domain knowledge for the UCI data sets, the user interacted with the application based on the class information of each case, as indicated by the colour. For the InfoChrom case base, the users utilized their domain expertise to move the cases in the MDS plot.

5.2 Experimental Results and Observations

Figures 5 and 6 show the change in performance of iCaseViz on the Iris and Glass Identification data and the InfoChrom case base over several iterations. It is evident from the figures that as the domain expert interacts more with the system and the notion of similarity as captured by the system evolves, there is a noticeable improvement in performance of the system. For the case bases used to demonstrate the effectiveness of our similarity learning algorithm, we show the correlation between the dissimilarities in the original and MDS spaces in

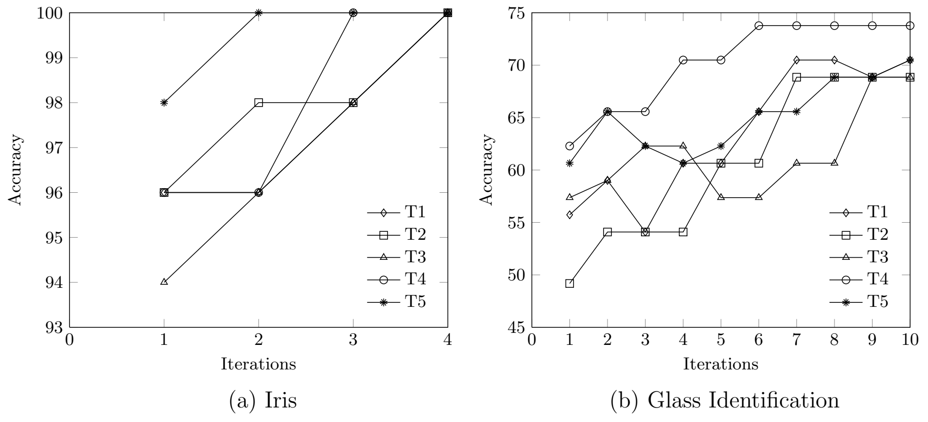

Fig. 5. Performance of iCaseViz on (a)Iris and (b)Glass Identification data sets. T1-T5 indicate 5 different random test sets. Note that in (a), the performance curve for T1 is not clearly visible as portions of it have merged with those of T3 and T4.

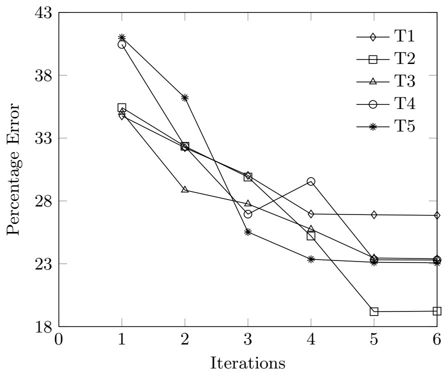

Fig. 6. Performance of iCaseViz on a subset of the InfoChrom case base for predicting the target variable 'potassium'. T1-T5 indicate 5 different random test sets.

Table 2. Correlation between dissimilarities calculated in original and MDS spaces

Data Set	Correlation
Iris	0.9905
Glass Identification	0.9188
InfoChrom Case Base	0.8563

Table 3. The time taken (in milliseconds) by various components of the visualizer. ϵ indicates that the time taken is negligible (of the order of nanoseconds).

Data set	Parallel Coordinates Plot					Scatter Plot
	Reading from file	Layout the data	Decide what to put on display	Filter the polylines	Draw onto memory image and display on screen	Reading from file, creating object & adding to QGraphics Scene
Iris	ϵ	ϵ	ϵ	ϵ	5	56
Glass Identification	ϵ	ϵ	ϵ	ϵ	11	4
Waveform-21	ϵ	ϵ	ϵ	6	202	126
InfoChrom Case Base	ϵ	ϵ	ϵ	ϵ	109	94

Table 2. These correlations show the extent to which the MDS algorithm maintains the relative dissimilarities of the original space, when projecting it to two dimensions.

We also report the time taken, in milliseconds, by the various components of the visualizers to display on a screen with a resolution of 1366 x 768, when the display window is maximized. We can see from Table 3 that the time taken are negligible and therefore the system is well-suited for real-time interaction, even on large case bases. Also, we noted that the response time of our system for user movements on the PC as well as the MDS plot is of the order of nanoseconds.

6 Conclusions and Future Work

In this paper, we have introduced iCaseViz, an interactive visualization framework that not only shows the relative positions of the cases in the original high dimensional problem space by projecting them onto a lower dimensional space but also lets the user explore this space by allowing them to change the relative similarities between the cases. Both the notions of problem side similarity and solution side similarity are captured by this visualization technique. This is coupled with a CBR centric optimization function that learns the domain experts' notion of similarity by arriving at an optimal set of feature weights. This work attempts to bridge the gap between the experts' knowledge about the problem domain and the case-based reasoning methodology by providing an iterative and interactive visualization application. The weighting term in the optimization function ensures that even though the domain expert is given the freedom to change the relative similarities between cases, attempts to modify the similarities in a way which destroys the inherent structure of the case base to a large extent, are strongly discouraged by the system. This contributes towards making the system robust to any unwarranted changes. We also show that the system is able to learn a suitable set of feature weights very fast over a small number of iterations, not only for UCI datasets but also for high dimensional real world data, as is evident from the performance graphs. The visualizer in its current format can handle case bases with more than 20,000 cases and over 175 features very easily and with minimal delay in response during interactions. This is a significant development over previous applications which became unresponsive and most often even failed to load case bases with around 10,000 cases represented in terms of around 15 features.

Our work is most similar to [19] as we have a common objective of interacting with an MDS plot to learn a dissimilarity measure by finding an appropriate set of feature weights. However, our work is different from [19] in a lot of aspects. Firstly, our work is more CBR-specific in the sense that we discourage the users from making any updates that can possibly lead to a configuration where the CBR hypothesis is violated both locally and across the entire case base. We do this by introducing a tradeoff term that prevents a potential decrease in the alignment of the case base. Dis-Function [19], on the other hand, provides an inertia against any updates made by the expert to the MDS plot. Secondly, in [19], a user is only allowed to select two sets of points and change the relative distances between them, while in our work we allow the expert to select any number of cases and modify the relative dissimilarities between those and the remaining cases in the case base. So, once the cases are moved, the user has an idea about the relative positions of all the cases in the case base which he is going to see in the next update. Since we also show the test cases, colored according to the quality of their solutions, along with the cases in the case base, this gives the domain expert a global picture at any point of time and thus helps him make informed decisions.

In the future we would like to perform more experiments on large case bases with various fast optimization problem solvers. This is because the current solver we are using is slow when working with large datasets and therefore the application takes time to compute the revised similarity values once the expert is done with the modifications. Also, currently the expert can only use the PC visualizer for viewing the data at several zoom levels and can re-position the axes in any order and with arbitrary gaps between them. We are looking to develop on this by providing an innovative visualization scheme based on parallel coordinates that will enable further interaction with the PC plot. Also, other modules are being developed and integrated with the existing tool which will provide additional information about the cases selected in the MDS plot. For example, a data viewer module will show the cases in terms of the features and their corresponding values. Selected cases will be highlighted in the data viewer. For the InfoChrom case base, where each case is a chromatogram image, an additional module to show the chromatogram for a selected case is being developed. We would also like to find out the utility of the PC visualizer in terms of the extent to which it aids the user in making decisions. This can be done by comparing the performance of the system when the PC plot is shown to the user as compared to when it is not. An integrated visualization and analysis application can then be built with further options to delete noisy cases and attributes, with the changes reflecting instantaneously in the PC and MDS visualizers. We are also interested in exploring various ways to learn a suitable kernel for a case base by using the expert information obtained through interaction with the scatter plot. We believe that encoding domain knowledge in the kernel function will be the key towards developing more accurate similarity measures for a particular prediction task.

References

1. Khemani, D., Joseph, M.M., Variganti, S.: Case based interpretation of soil chromatograms. In: Althoff, K.-D., Bergmann, R., Minor, M., Hanft, A. (eds.) ECCBR 2008. LNCS (LNAI), vol. 5239, pp. 587–599. Springer, Heidelberg (2008)
2. Kar, D., Chakraborti, S., Ravindran, B.: Feature weighting and confidence based prediction for case based reasoning systems. In: Agudo, B.D., Watson, I. (eds.) ICCBR 2012. LNCS, vol. 7466, pp. 211–225. Springer, Heidelberg (2012)
3. Aha, D.W.: Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms. Int. J. Man-Mach. Stud. 36(2), 267–287 (1992)
4. Wettschereck, D.: A study of distance-based machine learning algorithms. Ph.D. dissertation, Department of Computer Science, Oregon State University (1994)
5. Wettschereck, D., Aha, D.W.: Weighting features. In: Aamodt, A., Veloso, M.M. (eds.) ICCBR 1995. LNCS, vol. 1010, pp. 347–358. Springer, Heidelberg (1995)
6. Wettschereck, D., Aha, D.W., Mohri, T.: A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. Artif. Intell. Rev. 11(1-5), 273–314 (1997)
7. Stahl, A.: Learning feature weights from case order feedback. In: Aha, D.W., Watson, I. (eds.) ICCBR 2001. LNCS (LNAI), vol. 2080, pp. 502–516. Springer, Heidelberg (2001)

8. Cunningham, P.: A taxonomy of similarity mechanisms for case-based reasoning. IEEE Transactions on Knowledge and Data Engineering 21(11), 1532–1543 (2009)
9. Inselberg, A., Dimsdale, B.: Parallel coordinates: a tool for visualizing multidimensional geometry. In: Proceedings of the 1st Conference on Visualization 1990, VIS 1990, pp. 361–378. IEEE Computer Society Press (1990)
10. Massie, S., Craw, S., Wiratunga, N.: Visualisation of case-based reasoning for explanation. In: Proceedings of ECCBR Workshop, Madrid, pp. 135–144 (2004)
11. Falkman, G.: The use of a uniform declarative model in 3D visualisation for case-based reasoning. In: Craw, S., Preece, A.D. (eds.) ECCBR 2002. LNCS (LNAI), vol. 2416, pp. 103–117. Springer, Heidelberg (2002)
12. Borg, I., Groenen, P.: Modern Multidimensional Scaling: theory and applications. Springer (2005)
13. Broekens, J., Cocx, T., Kosters, W.A.: Object-centered interactive multidimensional scaling: Ask the expert. In: Proceedings of the 18th Belgium-Netherlands Conference on Artificial Intelligencem, BNAIC (2006)
14. Buja, A., Swayne, D.F., Littman, M.L., Dean, N., Hofmann, H.: Xgvis: Interactive data visualization with multidimensional scaling. Technical report (2001)
15. desJardins, M., MacGlashan, J., Ferraioli, J.: Interactive visual clustering. In: Proceedings of the 12th International Conference on Intelligent User Interfaces, IUI 2007, pp. 361–364. ACM, New York (2007)
16. May, T., Bannach, A., Davey, J., Ruppert, T., Kohlhammer, J.: Guiding feature subset selection with an interactive visualization. In: IEEE VAST, pp. 111–120 (2011)
17. Endert, A., Han, C., Maiti, D., House, L., Leman, S., North, C.: Observation-level interaction with statistical models for visual analytics. In: IEEE VAST, pp. 121–130 (2011)
18. Okabe, M., Yamada, S.: An interactive tool for human active learning in constrained clustering. Journal: Emerging Technologies in Web Intelligence 3(1) (2011)
19. Brown, E.T., Liu, J., Brodley, C.E., Chang, R.: Dis-function: Learning distance functions interactively. In: IEEE VAST, pp. 83–92 (2012)
20. Smyth, B., Mullins, M., McKenna, E.: Picture perfect: Visualisation techniques for case-based reasoning. In: ECAI, pp. 65–72 (2000)
21. McArdle, G., Wilson, D.: Visualising case-base usage. In: Workshop Proceedings ICCBR, pp. 105–114 (2003)
22. Namee, B.M., Delany, S.J.: Cbtv: Visualising case bases for similarity measure design and selection. In: Bichindaritz, I., Montani, S. (eds.) ICCBR 2010. LNCS, vol. 6176, pp. 213–227. Springer, Heidelberg (2010)
23. McKenna, E., Smyth, B.: An interactive visualisation tool for case-based reasoners. Appl. Intell. 14(1), 95–114 (2001)
24. Chakraborti, S., Cerviño Beresi, U., Wiratunga, N., Massie, S., Lothian, R., Khemani, D.: Visualizing and evaluating complexity of textual case bases. In: Althoff, K.-D., Bergmann, R., Minor, M., Hanft, A. (eds.) ECCBR 2008. LNCS (LNAI), vol. 5239, pp. 104–119. Springer, Heidelberg (2008)
25. Freyne, J., Smyth, B.: Creating visualizations: A case-based reasoning perspective. In: Coyle, L., Freyne, J. (eds.) AICS 2009. LNCS, vol. 6206, pp. 82–91. Springer, Heidelberg (2010)
26. CVX Research Inc.: CVX: Matlab software for disciplined convex programming, version 2.0 beta (September 2012)
27. Grant, M., Boyd, S.: Graph implementations for nonsmooth convex programs. In: Blondel, V.D., Boyd, S.P., Kimura, H. (eds.) Recent Advances in Learning and Control. LNCIS, vol. 371, pp. 95–110. Springer, Heidelberg (2008)