Cooperation between automatic algorithms, interactive algorithms and visualization tools for Visual Data Mining

François Poulet

ESIEA Recherche
38, rue des Docteurs Calmette et Guérin
Parc Universitaire de Laval-Changé
53000 Laval, France
poulet@esiea-ouest.fr

Abstract. Visual data-mining strategy lies in tightly coupling the visualizations and analytical processes into one data-mining tool that takes advantage of the strengths from multiple sources. This paper presents concrete cooperation between automatic algorithms, interactive algorithms and visualization tools. The first kind of cooperation is an interactive decision tree algorithm called CIAD+. It allows the user to be helped by an automatic algorithm based on a support vector machine (SVM) to optimize the interactive split performed at the current tree node or to compute the best split in an automatic mode. This algorithm is then modified to perform an unsupervised task, the resulting clustering algorithm has the same kind of help mechanism based on another automatic algorithm (the k-means). The last effective cooperation is a visualization algorithm used to explain the results of SVM algorithm. This visualization tool is also used to view the successive planes computed by the incremental SVM algorithm.

1 Introduction

Knowledge Discovery in Databases (or KDD) can be defined [1] as the non-trivial process of identifying patterns in the data that are valid, novel, potentially useful and understandable. In most existing data mining tools, visualization is only used during two particular steps of the data mining process: in the first step to view the original data, and in the last step to view the final results. Between these two steps, an automatic algorithm is used to perform the data-mining task. The user has only to tune some parameters before running his algorithm and wait for its results.

Some new methods have recently appeared [2], [3], [4], trying to involve more significantly the user in the data mining process and using more intensively the visualization [5], [6], this new kind of approach is called visual data mining. In this paper we present some tools we have developed, which integrate automatic algorithms, interactive algorithms and visualization tools. These tools are two interactive classification algorithms and a visualization tool created to show the

results of an automatic algorithm. The classification algorithms use both human pattern recognition facilities and computer calculus power to perform an efficient user-centered classification. This paper is organized as follows.

In section 2 we briefly describe some existing interactive decision tree algorithms and then we present our new interactive algorithms, the first one is an interactive decision tree algorithm called CIAD+ (Interactive Decision Tree Construction) using support vector machine (SVM) and the second is derived from the first one and performs unsupervised classification (clustering).

In section 3 we present a graphical tool used to explain the results of support vector machine algorithms. These algorithms are known to be efficient but they are used as "black boxes", there is no explanation of their results. Our visualization tool graphically explains the results of the SVM algorithm. The implemented SVM algorithm can modify an existing linear classifier by both retiring old data and adding new data. We visualize the successive separating planes computed by this algorithm.

Section 4 concludes the paper and lists some future work.

2 Interactive decision tree construction

Some new user-centered manual (i.e. interactive or non-automatic) algorithms inducing decision trees have appeared recently: Perception Based Classification (PBC) [7], Decision Tree Visualization (DTViz) [8], [9] or CIAD [10]. All of them try to involve the user more intensively in the data-mining process. They are intended to be used by a domain expert not the usual statistician or data-analysis expert. This new kind of approach has the following advantages:

- the quality of the results is improved by the use of human pattern recognition capabilities,
- using the domain knowledge during the whole process (and not only for the interpretation of the results) allows a guided search for patterns,
- the confidence in the results is improved, the KDD process is not just a "black box" giving more or less comprehensible results.

The technical part of these algorithms are somewhat different: PBC and DTViz use an univariate decision tree by choosing split points on numeric attributes in an interactive visualization. They use a bar visualization of the data: within a bar, the attribute values are sorted and mapped to pixels in a line-by-line fashion according to their order. Each attribute is visualized in an independent bar (cf. fig.1). The first step is to sort the pairs ($attr_i$, class) according to attribute values, and then to map to lines colored according to class values. When the data set number of items is too large, each pair ($attr_i$, class) of the data set is represented with a pixel instead of a line. Once all the bars have been created, the interactive algorithm can start. The classification algorithm performs univariate splits and allows binary splits as well as n-ary splits.

CIAD is described in the next section and, in section 2.2, we present a new version of CIAD (called CIAD+) with a help tool added to the interactive algorithm allowing the user to perform an automatic computation of the best bivariate split.

[9] uses a two dimensional polygon or more precisely, an open-sided polygon (i.e. a polyline) in a two dimensional matrix. It is interactively drawn in the matrix. The display is made of one 2D matrix and one-dimensional bar graphs (like in PBC).

Only PBC provides the user with an automatic algorithm to help him choosing the best split in a given tree node. The other algorithms can only be run in a 100% manual interactive way.

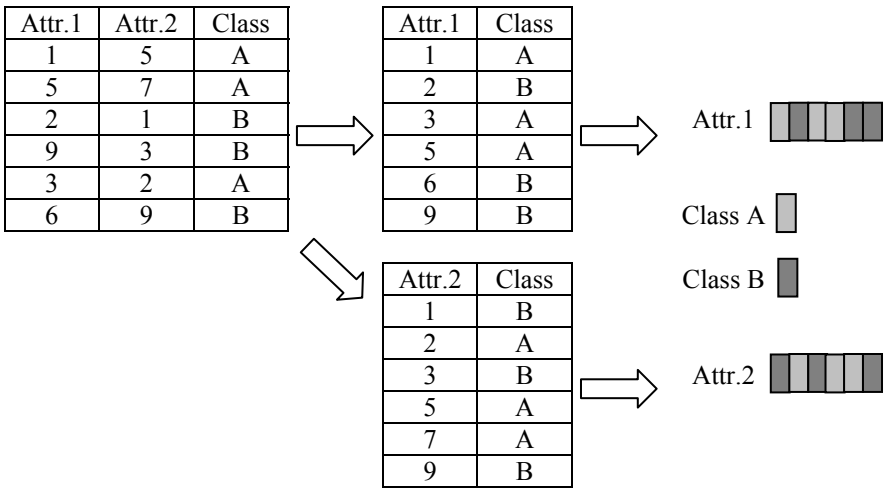

Fig. 1. Creation of the visualization bars with PBC

2.1 CIAD

CIAD uses a bivariate decision tree using line drawing in a set of two-dimensional matrices (like scatter plot matrices [11]). The first step of the algorithm is the creation of a set of $(n-1)^2/2$ two-dimensional matrices (n being the number of attributes). These matrices are the two dimensional projections of all possible pairs of attributes, the color of the point corresponds to the class value. This is a very effective way to graphically discover relationships between two quantitative attributes. One particular matrix can be selected and displayed in a larger size in the bottom right of the view (as shown in figure 2 using the Segment data set from the UCI repository [12], it is made of 19 continuous attributes, 7 classes and 2310 instances). Then the user can start the interactive decision tree construction by drawing a line in the selected matrix and performing thus a binary, univariate or bi-variate split in the current node of the tree. The strategy used to find the best split is the following. We try to find a split giving the largest pure partition, the splitting line (parallel to the axis or oblique) is interactively drawn on the screen with the mouse. The pure partition is then removed from all the projections. If a single split is not enough to get a pure partition, each half-space created by the first split will be treated alternately in a recursive way (the alternate half-space is hidden during the current one's treatment).

At each step of the classification, some additional information can be provided to the user like the size of the resulting nodes, the quality of the split (purity of the resulting partition) or overall purity. Some other interactions are available to help the user: it is possible to hide / show / highlight one class, one element or a group of elements.

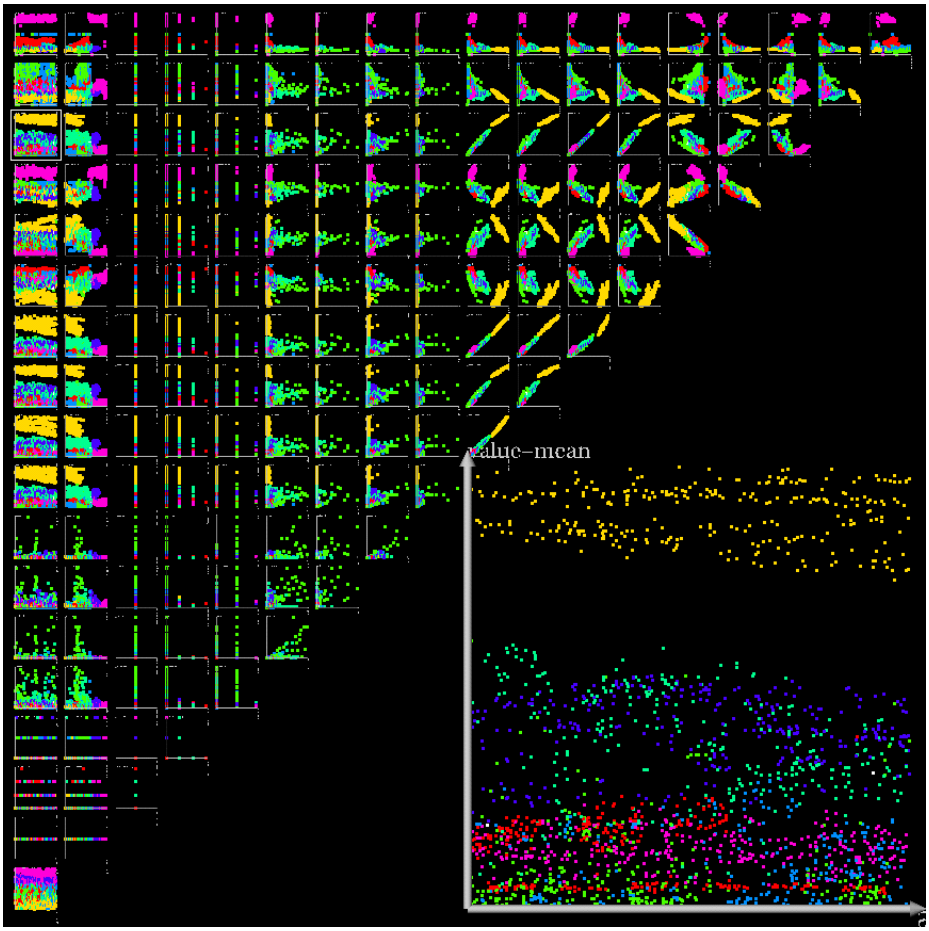

Fig. 2. The Segment data set displayed with CIAD

2.2 CIAD+

The first version of the CIAD algorithm was only an interactive algorithm. No help was available for the user, and sometimes, it was difficult to find the best pure partition in the set of two-dimensional matrices. We have decided to provide such help. Our first intention was to use a modified OC1 (Oblique Classifier 1) algorithm [13]: OC1 performs real oblique cuts (we have a real n-dimensional hyperplane with n-dimensional data) and in CIAD, the cuts are only "oblique" in two dimensions. The plane coefficients are null in all the other dimensions. We have made another choice:

we use a support vector machine (SVM). This algorithm is equivalent to OC1 in its simplest use, and will allow us to benefit from all its other possibilities for further developments.

2.2.1 The SVM algorithms

The SVM algorithms are kernel-based classification methods. They can be seen as a geometric problem: to find the best separating plane of a two classes data set. A lot of methods can be used to find this best plane, and a lot of algorithms have been published. A review of the different algorithms can be found in [14]. They are used in a wide range of real-world applications such as text categorization, hand-written character recognition, image classification or bioinformatics. We briefly describe here the basis of the algorithm, from the geometrical point of view.

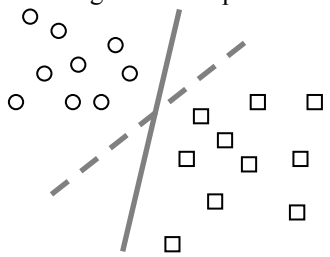

Fig. 3. Two possible separating planes

The aim of the SVM algorithm is to find the best separating plane between the n-dimensional elements of two classes. There are two different cases according to the nature of the data: they are linearly separable or not.

In the first case, the data are linearly separable i.e. there exists a plane that correctly classifies all the points in the two sets. But there are infinitely many separating planes as shown in Fig.3. Geometrically, the best plane is defined as being furthest from both classes (i.e. small perturbations of any point would not introduce misclassification errors). The problem is to construct such a plane. It has been shown in [15] that this problem is equivalent to finding the convex hull (i.e. the smallest convex set containing the points) of each class, and then to finding the nearest two points (one from each convex hull); the best plane bisects these closest points.

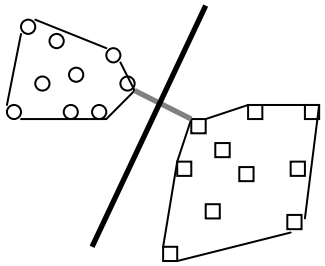

Fig. 4. The best separating plane bisects the closest points

In the second case, the data are not linearly separable (i.e. the intersection of the two convex hulls is not empty). There is no clear definition of what is the "best" plane. The solution is to create a misclassification error, and to try to minimize this error.

2.2.2 SVM in CIAD+

We use two different SVM algorithms in CIAD+. The first one is the geometric version described in section 2.2.1 for the linearly separable case. The convex hulls are computed in two dimensions with the quick hull algorithm [16], then the two closest points of the convex hulls are computed with the rotating calipers algorithm [17].

For the linearly inseparable case, a lot of solutions have been developed and compared, we have chosen one of these algorithms: the RLP (Robust Linear Programming) algorithm [18] because it is the best one when the data are not linearly separable.

The RLP algorithm will compute the separating plane wx=γ minimizing the average violations:

$$\frac{1}{m}\sum_{i=1}^{m}(-A_i w+\gamma+1)_+ + \frac{1}{k}\sum_{i=1}^{k}(B_i w-\gamma+1)_+ \qquad (\mathbf{1})$$

of points of *A* lying on the wrong size of the plane wx=γ+1, and of points of *B* lying on the wrong side of the plane wx=γ-1 as shown in Fig.5.

This algorithm computes the n-dimensional hyperplane, we have modified it to compute the best two-dimensional plane.

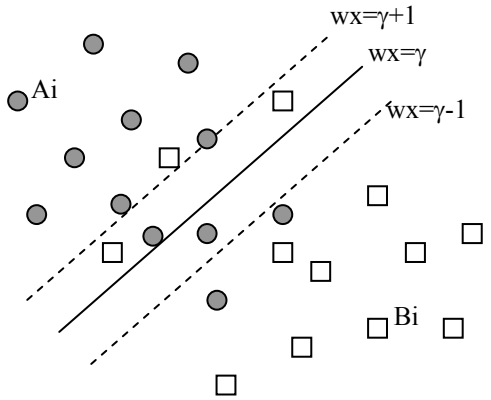

Fig. 5. Optimal plane for linearly inseparable data

SVMs in CIAD+ are used to help the user. The first kind of help is when the user draws interactively the separating line on the screen with a pure partition on one side, the optional help optimizes the line position to reach the best line position (furthest from both groups) with the computation of the closest points of the convex hulls. The second kind of help is the same case except there is no pure partition, the best line position is computed with the RLP algorithm in the two dimensions corresponding to the selected matrix. The last kind of help is used when the user cannot find a separating plane, the help algorithm has to compute the best separating plane among all the ones corresponding to the projections along pairs of attributes. So we compute all the separating lines in the 2D projections and we keep the best one.

This help mechanism can be turned on / off by the user. It slightly improves the accuracy of the results on the training sets (this result may be more significant according to the kind of user), more on the test set, and it considerably reduces the time needed to perform the classification and increases the ease of use. An example is shown on figure 6, the left part is the original line drawn interactively by the user on the screen and the right part shows the transformed line (the best separating plane computed with the convex hulls).

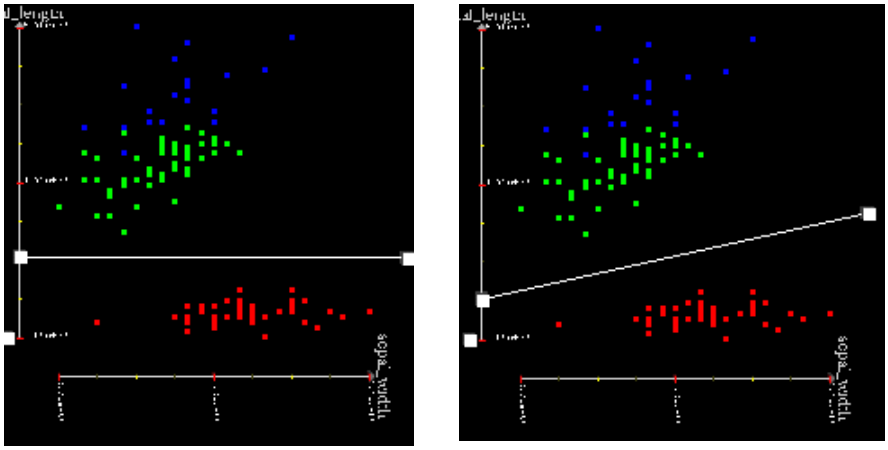

Fig. 6. An example of the automatic best separating plane on iris data set

2.3 Clustering

The interactive algorithm described in the previous section can also be used for unsupervised classification. The computation of the convex hulls and the nearest points can be computed with or without the class information. The proposed algorithm can perform either usual decision tree (supervised classification) or clustering (unsupervised classification). This kind of approach allows the user to perform clustering of the dataset easily using its pattern recognition capabilities and avoiding the usual complexity of the other algorithms. The same kind of help as for decision tree construction is provided to the user: the separating line drawn in a 2D projection can be optimized to be the furthest from the two clusters.

But there is one difference with the decision tree construction algorithm, when the user does not perceive clearly a separating line, this line can be computed automatically (with a modified SVM algorithm). This SVM algorithm cannot be used without the class information, so we have chosen a k-means algorithm. All possible partitions into two clusters are searched for in each matrix and the best one is kept. Then we compute the convex hulls and nearest points to find the best separating line in the same way as for decision tree construction.

As shown in [19], this kind of algorithm may not be very efficient for high dimensional data sets because axis-parallel projections may lead to an important loss of information. This restriction exists anyway because of the graphical representation we use (we cannot display a very large quantity of scatter plot matrices). On the other hand, the results are more comprehensible because we only use one attribute on each axis and not a linear combination of various number of attributes for the axes.

2.4 Some results of interactive algorithms

The characteristics of the different algorithms are summarized in table 1. Some results of interactive algorithms compared to automatic ones have been presented by their authors. To summarize them, we can say their results concerning efficiency, are generally at least as good as automatic decision tree algorithms such as CART (Classification And Regression Trees) [20], C4.5 [21], OC1 [13], SLIQ (Supervised Learning In Quest) [22] or LTree (Linear Tree) [23]. The main difference between the two kinds of algorithms is the tree size. Most of the time, interactive algorithms have smaller tree sizes. This tree size reduction can significantly increase the result comprehensibility. Furthermore, as the user is involved in the tree construction, his confidence in the model is increased too. This may be a little less significant for the LTree algorithm because of the open-sided polygon used in the classification: they are easy to understand during the visual construction step of the tree, but without this information, the resulting equations may be not so easy to understand.

	Vis. technique	split	+	-
Ware	bar graphs + 1 scatter plot	poly-line binary	large dataset tree size	result comprehensibility plot of qual. x qual. attr.
PBC	bar graphs	univariate n-ary	large datasets help	loss of information
DTViz	bar graphs	univariate n-ary	large datasets	loss of information
CIAD+	set of scatter plot	bivariate binary	tree size help	plot of qual. x qual. attr.

Table 1. Comparison of interactive decision tree algorithms

If we compare the interactive algorithms, we can say PBC and DTViz are particularly interesting for large data sets (because of the pixelisation technique used), but their pixelisation technique introduces some bias in the data: for example two classes very far one from the other have the same representation as the same two classes very near one to one other. We lose the distance information in this kind of representation. Ware and CIAD+ will provide smaller trees because they can use bivariate splits, but their kind of visualization tool (scatterplot matrices) are not at all suitable for the display of two qualitative attributes (a lot of points have the same projection). Only two algorithms provide the user with a help, PBC and CIAD+, this is a significant advantage of these two algorithms because during the decision tree

construction, there is often a least one particular step where the best split is difficult to visually detect.

The kind of cooperation between automatic and manual algorithms in PBC and CIAD+ shows the interest of mixing the human pattern recognition facilities and the computer processing power. The human pattern recognition facilities reduce the cost of the computation of the best separating plane, and the computer processing power can be used at low cost (for a single step, instead of the whole process) when the human pattern recognition fails.

3 Visualization of SVM results

Another kind of cooperation is between automatic algorithms and visualization tools used to show the results. As described in the previous section, SVM are today widely used because they give high quality results, but they are used as a "black box". They give high quality results, but there is no explanation of these results.

One paper [24] talks about SVM results visualization: they use projection-based tour [25] method to visualize the results. They use a visualization of the distribution of the data predicted class (by the way of histograms), a visualization of the data and the support vectors in 2D projection, and examine the weights of the plane coordinates to find the most important attributes for the classification.

The authors recognize that their approach is very "labor intensive for the analyst. It cannot be automated because it relies heavily on the analyst's visual skills and patience for watching rotations and manually adjusting projection coefficients."

3.1 Visualization of the SVM separating plane

Our approach is to visualize all the intersections of the 2D planes (of the scatter plot matrices) with the separating plane computed by the SVM algorithm. We have chosen to use the incremental SVM algorithm from [26]. This algorithm gives the coefficient values of the separating hyperplane and the accuracy of the algorithm. We visualize the intersection of this hyperplane with the 2D scatter plot matrices, i.e. a line in each matrix (as shown in Figure 7 with the diabetes data set, from the UCI repository).

As we can see on figure 7, the resulting lines do not necessarily separate the two classes, the hyperplane does separate the data (the accuracy of the incremental SVM is 77,8% on this dataset), but not its "2D projections". It is only an approximate interpretation of the results.

All 2D representations of one n-dimensional feature will lead to a loose part of the information like the lines we get here or the support vectors displayed in [24]. This kind of representation seems more comprehensible than the support vectors, but it can only be used with a linear kernel function.

For large data sets, it is possible to only display the separating plane and not the data. The SVM algorithm used is able to classify 1 billion points, and such a quantity of points cannot be displayed in a reasonable time (furthermore this kind of representation is not at all suitable for such data set size).

For other kinds of kernel functions (not linear), this method cannot be used.

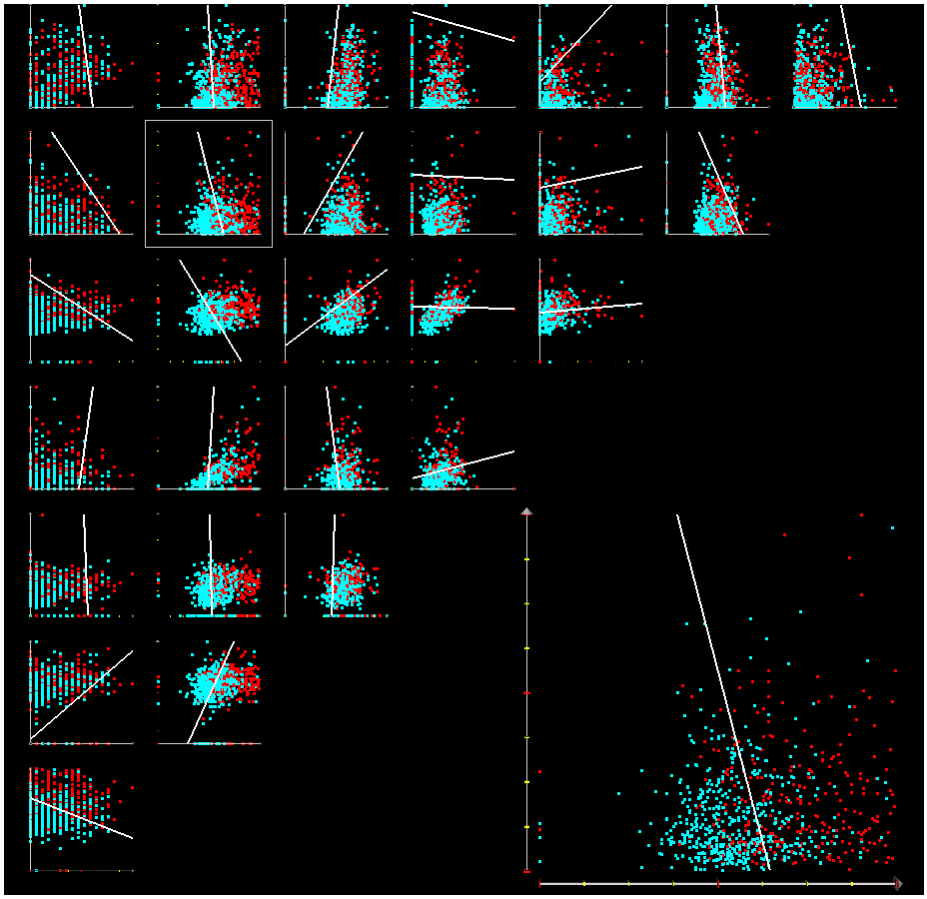

Fig. 7. Visualization of the separating hyperplane of the diabetes dataset.

3.2 Visualization of the evolution of SVM separating planes

A very interesting feature of the incremental support vector machine we have used is its capability to modify an existing linear classifier by both withdrawing old data and adding new data. We use our visualization tool to show the successive modifications of the separating plane projections. The first plane is calculated (and projected on the 2D matrices) and then blocks of data are successively added and withdrawn. The modification of the plane is computed and the corresponding projections are displayed in the matrices.

This kind of visualization tool is very powerful to examine the variations of the separating plane according to the data evolution. Even if the projections used lose some amount of information, we know what the attributes involved in the

modification of the separating plane are. The evolution of the n-dimensional plane is very difficult to show in another way. The authors of the paper describing the algorithm measure the difference between planes by calculating the angle between their normals. These angle values are then displayed like circle radii.

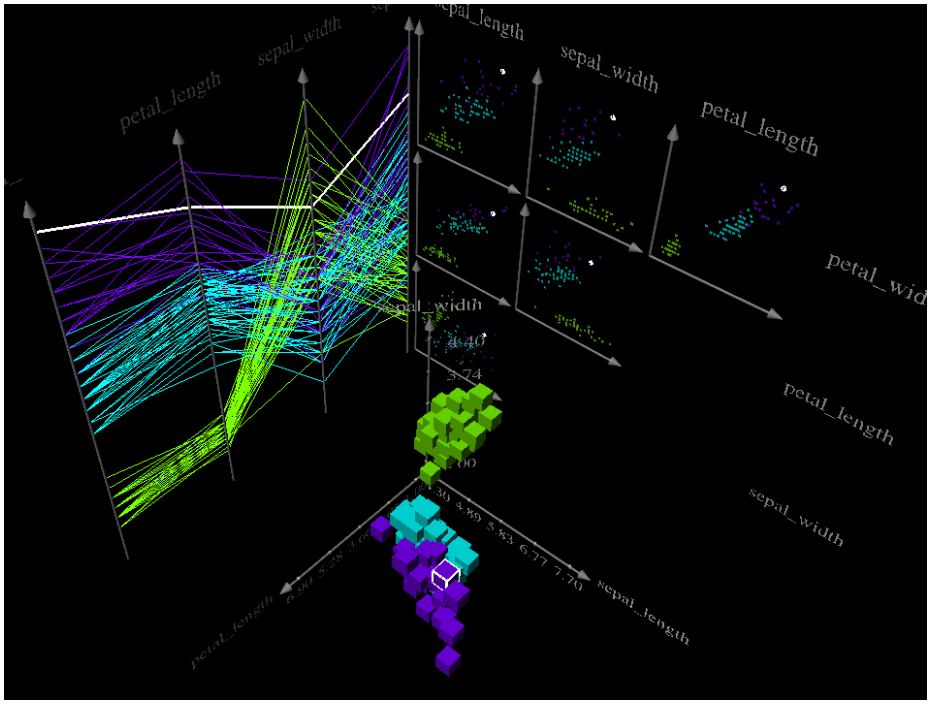

Fig. 8. Three linked tools in the FullView environment

4. Conclusion and future work

Before concluding, some words about the implementation. All these tools have been developed using C/C++ and three open source libraries: OpenGL, Open-Motif and Open-Inventor. OpenGL is used to easily manipulate 3D objects, Open-Motif for the graphical user interface (menus, dialogs, buttons, etc.) and Open-Inventor to manage the 3D scene. These tools are included in a 3D environment, described in [27], where each tool can be linked to other tools and be added or removed as needed. Figure 8 shows an example with a set of 2D scatter plot matrices, a 3D matrix and parallel coordinates. The element selected in the 3D matrix appeared selected too in the set of scatter plot matrices and in the parallel coordinates. The software program can be run on any platform using X-Window, it only needs to be compiled with a standard C++ compiler. Currently, the software program is developed on SGI O2 and PCs with Linux.

In this paper we have presented two new interactive classification tools and a visualization tool to explain the results of an automatic SVM algorithm. The classification tools are intended to involve the user in the whole classification task in order to:

- take into account the domain knowledge,
- improve the result comprehensibility, and the confidence in the results (because the user has taken part in the model construction),
- exploit human capabilities in graphical analysis and pattern recognition.

The visualization tool was created to help the user in understanding the results of an automatic SVM algorithm. These SVM algorithms are more and more frequently used and give efficient results in various applications, but they are used as "black-boxes". Our tool gives an approximate but informative graphical interpretation of these results.

A forthcoming improvement will be another kind of cooperation between SVM and visualization tools: an interactive visualization tool will be used to improve the SVM results when we have to classify more than two classes.

References

1. U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy, Eds, "Advances in Knowledge Discovery and Data Mining", AAAI Press, 1996.
2. P.Wong, "Visual Data Mining", in IEEE Computer Graphics and Applications, 19(5), 20-21, 1999.
3. F. Poulet, "Visualization in data mining and knowledge discovery," in Proc. HCP'99, 10th Mini Euro Conference "Human Centered Processes" ed. P. Lenca (Brest, 1999), 183-192.
4. M.Ankerst, M.Ester, H-P.Kriegel, "Toward an Effective Cooperation of the Computer and the User for Classification" in proc. of KDD'2001, 179-188.
5. C.Aggarwal, "Towards Effective and Interpretable Data Mining by Visual Interaction", in SIKDD Explorations 3(2), 11-22, accessed from www.acm.org/sigkdd/explorations/.
6. B.Schneiderman, "Inventing Discovery Tools: Combining Information Visualization with Data Mining", in Information Visualization 1(1), 5-12, 2002.
7. M.Ankerst, "Visual Data Mining", PhD Thesis, Ludwig Maximilians University of Munich, 2000.
8. J.Han, N.Cercone, "Interactive Construction of Decision Trees" in proc. of PAKDD'2001, LNAI 2035, 575-580, 2001.
9. M.Ware, E.Franck, G.Holmes, M.Hall, I.Witten, "Interactive Machine Learning: Letting Users Build Classifiers", in International Journal of Human-Computer Studies (55), 281-292, 2001.
10. F.Poulet, "CIAD: Interactive Decision Tree Construction" in proc. of VIII Rencontres de la Société Francophone de Classification", Pointe-à-Pitre, 275-282, 2001 (in french).
11. J.Chambers, W.Cleveland, B.Kleiner, P.Tukey, "Graphical Methods for Data Analysis", Wadsworth (1983).
12. C.Blake, C.Merz, UCI Repository of machine learning databases, [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science, (1998).
13. S.Murthy, S.Kasif, S.Salzberg, "A system for induction of oblique trees", Journal of Artificial Intelligence Research 2, 1-32, 1994.

14.K-R.Muller, S.Mika, G.Ratsch, K.Tsuda, B.Scholkopf, "An Introduction to Kernel-Based Learning Algorithms" in IEEE Transactions on Neural Networks, 12(2), 181-201, 2001.
15.K.Bennett, E.Bredensteiner, "Duality and Geometry in SVM Classifiers", in proc of the Seventeenth International Conference on Machine Learning, Pat Langley Editor, Morgan Kaufmann, San Francisco, 57-64, 2000.
16.C.Barber, D.Dobkin, H.Huhdanpaa, "The Quickhull algorithm for convex hulls", in ACM Transactions On Mathematical Software, 22, 469-483, 1996.
17.G.Toussaint, "Solving geometric problems with the rotating calipers", in proc. of IEEE MELECON'83, Athens, Greece, pp. A10.02/1-4, 1983.
18.K.Bennett, O.Mangasarian, "Robust Linear Programming Discrimination of Two Linearly Inseparable Sets", in Optimization Methods and Software, 1, 23-34, 1992.
19.C.Aggarwal, P.Yu, "Redifining Clustering for High-Dimensional Applications", in IEEE Transactions on Knowledge and Data Engineering, 14(2), 210-225, 2002.
20.L.Breiman, J.Friedman, R.Olsen, C.Stone, "Classification and Regression Trees", Wadsworth, (1984).
21.J.Quinlan, "C4.5: Programs for Machine Learning", Morgan-Kaufman Publishers, 1993.
22.M.Metha, R.Agrawal, J.Rissanen, "SLIQ: A fast scalable classifier for data mining", in proc. of the 5th International Conference on Extending Database Technology, Avignon, France, 1996, 18-32.
23.J.Gama, P.Brazdil, "Linear Tree" in Intelligent Data Analysis, 3, 1-22 (1999).
24.D.Caragea, D.Cook, V.Honavar, "Gaining Insights into Support Vector Machine Pattern Classifiers Using Projection-Based Tour Method", in proc. of KDD'2001 Workshop on Visual Data Mining.
25.D.Asimov, "The Grand Tour: A Tool for Viewing Multidimensional Data" in SIAM Journal of Scientific and Statistical Computing, 6(1):128-143, 1985.
26. G. Fung, O. Mangasarian, "Incremental Support Vector Machine Classification" in proc. of the 2nd SIAM International Conference on Data Mining, Airlington, USA, Apr.11-13, 2002.
27.F.Poulet, "FullView: A Visual Data-Mining Environment", in International Journal of Image and Graphics, 2(1), 127-144, 2002.