Concept Tree Based Ordering for Shaded Similarity Matrix

Article · January 2002

3 authors, including:

Jun Wang
19 PUBLICATIONS   **761** CITATIONS

SEE PROFILE

Bei Yu
Syracuse University
40 PUBLICATIONS   **1,184** CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Health News Quality View project

Brain Simulations View project

Concept Tree Based Ordering for Shaded Similarity Matrix

Jun Wang   Bei Yu   Les Gasser

Graduate School of Library and Information Science
University of Illinois at Urbana-Champaign
501 E. Daniel St., Champaign, IL 61820, USA
{junwang4, beiyu, gasser}@uiuc.edu

Abstract

Shaded similarity matrix is one of the oldest graphic techniques that has long been used in data visualization. The key problem of shaded similarity matrix is how to order the data in the matrix such that similar objects are put adjacent. Based on the assumption that "similar objects share same concepts or patterns", we propose a novel general ordering approach: transforming the object ordering problem to concept organization. In this paper, we adopt concept tree as concept organization means, and hence our method is called "concept tree based ordering". Using concept tree based ordering has some advantages over existing approaches: 1) it greatly enhances the interpretation of the visualization results since concept trees are easy to understand; 2) it can avoid the negative impact from irrelevant features since concept acquisition tends to select relevant features; 3) it is very natural to integrate shaded similarity matrix with some popular data mining methods for better visualization such as nearest neighbor classification, decision tree, regression tree, and clustering tree. This paper investigates the application of concept tree based ordering with shaded similary matrix to visualizing two classification methods: nearest neighbor and decision tree. We also introduce ensemble visualization for handling large data sets.

Keywords: Visual Data Mining, Shaded Similarity Matrix, Concept Tree, Decision Tree, Nearest Neighbor, Ensemble Classification.

1   Introduction

Shaded similarity matrix has long been used in visual cluster analysis[11, 8, 18]. In shaded similarity matrix, the relationship between very similar objects are represented by dark shading, while less similar items by light shading. The key problem of shaded similarity matrix is how to order the data or objects in the matrix such that similar objects are put adjacent. There are exponentially many ways to order a set of objects. For example, on 100 objects, the number of ordering or permutation is 100!. It should be noted that the ordering problem is not only for shaded similarity matrix, many data visualization approaches need to consider it if there is no natural ordering (e.g., categorical data don't have natural ordering like numerical data)[2].

Previous work on the ordering problem has mainly focused on how to design an efficient algorithm to get a near-optimal ordering. For example, [8] discuessed an algorithm which first generates a hierarchical agglomerative clustering and then uses it as a starting point to reorder the objects through the exchange of adjacent pairs. To handle large categorical datasets, [2] proposed an efficient multi-level method by coarsening the original dataset without computing most pairwise similarities.

In this paper, we propose a novel approach to the ordering problem: *concept tree based ordering*. (Concept tree is also called decision tree when the task of data mining is classification.) The basic idea behind our approach is that we believe: *similar objects share same concepts or patterns*. Therefore, if we can discover concepts or patterns from a data set and organize these concepts in some way, then the problem of object ordering is solved. As an initial effort to this idea, we are interested in using concept tree as the concept organization means.

Using concept tree based ordering has following benefits over existing approaches: 1) it greatly enhances the interpretation of the visualization results since concept trees are easy to understand; 2) it can avoid the negative impact from irrelevant features since concept acquisition tends to select relevant features; 3) it is very natural to integrate shaded similarity matrix with some popular data mining methods for better visualization such as nearest neighbor classification, decision tree, regression tree, and clustering tree; 4) there are many efficient concept tree construction algorithms ready for use.

After the introduction of shaded similarity matrix and concept tree based ordering, we will focus on their ap-

plications to visualizing two popular classification methods: nearest neighbor and decision tree. We will also explore how to attack the scalability problem using ensemble classification and sampling techniques. Experimental performance evaluation on *Statlog* data sets is also reported, which shows that the ensemble approach achieves a smaller tree while retaining the accuracy compared to an existing visual classification method.

2 Shaded Similarity Matrix

This section is a brief introduction to shaded similarity matrix . Over the past 40 years, shaded similarity matrix has been mainly used in visual cluster anaylsis. [11, 8] have comprehensive introduction to the early work, and [18] involves some recent related work.

In shaded similarity matrix [1], similarity in each cell is represented using a shade to indicate the similarity value: greater similarity is represented by dark shading, while lesser similarity by light shading. The dark and light cells may initially be scattered over the matrix. To bring out the potential clusterings, the rows and columns need to reorganize so that the similar objects are put on adjacent positions. If "real" clusters exist in the data, they should appear as symmetrical dark squares along the diagonal [8].

In this section, we will briefly show how a shaded similarity matrix is constructed and how it looks through an example. The data used in the example is part of the *Iris* data from the UCI repository [13]. There are 150 instances in the original Iris data set, which evenly distributed in 3 classes: *setosa, virginica*, and *versicolor*. For each class, we fetch its first 5 instances from the data file, and thus obtaining 15 instances (see Table 1). Table 2 is the distance matrix[2] computed using Euclidean distance after normalizing the data in Table 1 to $[0, 1]$. For the categorical attribute *Type*, the distance is 1 for different value and 0 otherwise.

The shaded distance matrix is illustrated in Fig. 1. Usually, we use seriation or unidimensional scaling to reorganize the matrix. The right figure in Fig. 1 is generated using the algorithm proposed in ClustanGraphics [18]. It works by weighting each similarity using the distance of the similarity cell to the diagonal. The algorithm tries to minimize the sum of the weighted similarities in the similarity matrix by reordering the pre-computed clusters in a hierarchical clustering such as a dendrogram. It should be noted that different seriation algorithms might generate different results.

---

[1]Some researchers called it *Shaded Proximity Matrix* or *Trellis Diagram*.

[2]For convenience, we use *distance matrix* and *similarity matrix* interchangeably since similarity can be defined in terms of distance.

Table 1. Data matrix extracted from the Iris data set. *Abbreviations:* **sl: sepal-length, sw: sepal-width, pl: petal-length, pw: petal-width.**

objects	sl	sw	pl	pw	type (class)
e_1	5.1	3.5	1.4	0.2	setosa
e_2	6.3	2.9	5.6	1.8	virginica
e_3	7.0	3.2	4.7	1.4	versicolor
...	...	...	...	...	...
e_{15}	6.5	2.8	4.6	1.5	versicolor

Table 2. Distance matrix of the instances in Table 1.

0.0	1.7	1.5	0.4	1.4	1.8	...	1.5
1.7	0.0	1.1	1.6	1.1	0.4	...	1.0
1.5	1.1	0.0	1.6	0.2	1.2	...	0.4
...	...						
1.5	1.0	0.4	1.5	0.3	1.2	...	0.0
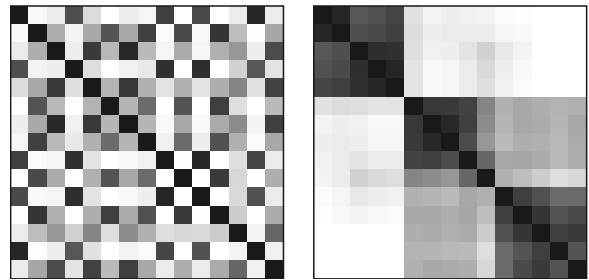
Figure 1. LEFT: *Randomly ordered* **shaded distance matrix corresponding to the distance matrix in Table 2;** RIGHT: *Reordered* **shaded distance matrix using a seriation algorithm.**

2.1 Two variants: k-nearest neighbor setting and distance threshold setting

To display a similarity matrix of n objects, we need n^2 cells or $\frac{n^2}{2}$ cells in the case of half matrix. In practice, usually it is not necessary to display all cells in a matrix. To this end, we should be able to have some parameters to control whether a cell is displayed or not. Depending on what parameter setting will be used, we have two variants of nearest neighbor visualization: *k-nearest neighbor setting* and *distance threshold setting*.

In the k-nearest neighbor setting, for an object whose position is i, suppose the positions of its k nearest neighbors are $j_1, \cdots, j_k$, then only the cells located at $(i, j_1), \cdots, (i, j_k)$ are displayed. Note that in this setting, the matrix is not symmetrical because that object A is the nearest neighbor of object B does not imply B is the nearest neighbor of A. Figure a) in Fig. 9 (on the last page) is an illustration of this setting. The Zoo data set used in the figure comes from the UCI repository. It contains 101 instances with 7 classes $\{mammal, bird, reptile, fish, amphibian, insect,$ and $invertebrate\}$. The concept tree used in Fig.9 is a trivial concept tree. We will introduce *concept tree* in next section.

In the distance threshold setting, only those cells are displayed whose distance values are less than a prespecified threshold. In this setting, the matrix is symmetrical. Figure b) in Fig. 9 (on the last page) is an example of this setting.

## 3 Concept Tree Based Ordering

3.1 Concept tree

A *concept tree* (also known as *concept hierarchy*) is composed of nodes and links with each node representing a concept. The links connecting node to its children specify an 'IS-A' or 'subset' relation. Concept tree is also called *decision tree* when the data mining task is classification. Fig.2 is an illustration of a concept tree. For details on concept tree or concept hierarchy, please refer to Pat Langley's book "Elements of Machine Learning"[10].

3.2 Concept tree based object ordering

Given a tree with k leaves from left to right: $< L_1, L_2, \cdots, L_k >$. For each leaf, it has a set of objects: $L_i = \{e_{i_1}, \cdots, e_{i_n}\}$. Then the ordering of all objects will be:

$$< \{e_{1_1}, \cdots, e_{1_n}\}, \cdots \cdots, \{e_{k_1}, \cdots, e_{k_n}\} >$$

We don't care about the internal object ordering within a leaf. This is due to an assumption that the objects within
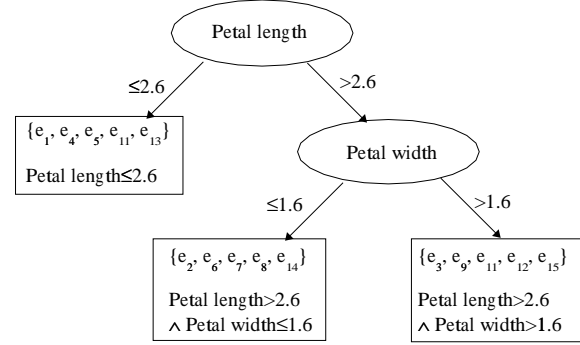
Figure 2. A concept tree on the Iris data set (see Table 1) obtained by using a clustering tree construction algorithm.

a leaf are similar enough, otherwise the leaf node can be partitioned into smaller leaves until the objects within a leaf are similar enough.

As an illustration, the object ordering based on the concept tree in Fig.2 can be:

$$< \{e_1, e_4, e_5, \cdots\}, \{e_2, e_6, e_7, \cdots\}, \{e_3, e_9, e_{10}, \cdots\} >$$

3.3 Concept tree construction framework

Table 3 is a general framework for constructing a concept tree which can be used in building decision trees, clustering trees, and etc[10]. One of the main differences between different algorithms is how to generate a *best* partition on a data set using some metric such as information entropy. Usually, decision tree (supervised learning) algorithms need to take *Class* attribute into consideration, while clustering tree algorithms don't have *Class* information.

4 Nearest Neighbor Visualization

Nearest neighbor [6] has been reported to have excellent results on many real world classification tasks. The basic approach involves storing training instances and their associated classes in memory and then, when given a test instance, finding the training instances nearest to that test instance and using them to predict the class.

When applying shaded similarity matrix to nearest neighbor visualization , the concept tree is a trivial tree with tree depth being 2. For example, Fig.3 is a concept tree for nearest neighbor on the Iris data set.

We use a distinct color to represent a different class. The shade of a color indicates the distance or similarity. The larger the distance, the lighter the shade. Since each cell represents the distance between two objects, what if the two

**Table 3. Concept tree construction algorithm.**

Inputs: The current node N of the concept tree.
       An object set Set.
Output: A concept tree.
Top-level call: $CTree(root, Set)$.
Procedure: $CTree(N, Set)$
  If the object set Set is not empty,
  Then:
    Generate a best partition π for Set using some metric.
    For each subset Set_i in the partition π,
      Form a node C with Set_i as its members.
      Form a conceptual description $Desc$ for Set_i.
      Associate the description $Desc$ with node C.
      Make node C a child of node N.
      $CTree(C, Set_i)$.
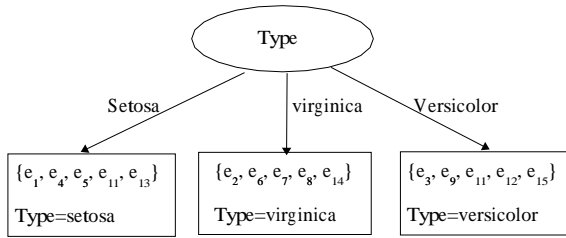
Figure 3. A trivial concept tree for nearest neighbor classification visualization on the Iris data set.

objects belongs to different classes? We just set the color to one class. This is not a problem because the similarity matrix is usually symmetrical.

In k-nearest neighbor setting, it is easy to compute the leave-one-out cross-validation accuracy, in which the set of m training instances is repeatedly divided into a training set of size $m-1$ and test set of size 1, in all possible ways [14]. In this paper, if **K-NN** appears on the top of a figure along with its leave-one-out errors and accuracy, it means we are using the k-nearest neighbor setting.
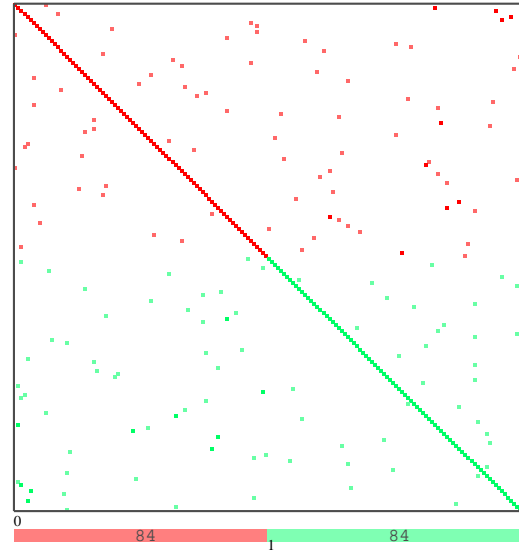
4.1 Applications of nearest neighbor visualization

We will show two possible applications of nearest neighbor visualization. First, nearest neighbor visualization can help users detect how regularities and outliers dynamically emerge by gradually changing the control parameter value. For example, a series of figures in Fig. 10 (on the last page) show the emergence of regularities. Also, from the figure c) in Fig. 10, we can see an outlier around the *Bird* block emerges. It is *Ostrich*! Here, we did not use the color for

representing the different classes because we want to show that the regularity is so clear and natural that it is almost unnecessary to use colors.

Second, it can help users to see whether nearest neighbor is suitable for classification, or whether the distance definition needs modification such as using weighted distance to reduce the interference of irrelevant attributes. Fig. 4 is an example showing that nearest neighbor is not fit for the Monks-2 problem: one of three Monks problems used in a comparison of different learning techniques which was performed at the 2nd European Summer School on Machine Learning in 1991 [17]. The target function of the Monks-2 problem is: EXACTLY TWO of $\{a_1 = 1, a_2 = 1, a_3 = 1, a_4 = 1, a_5 = 1, a_6 = 1\}$.

Figure 4. Nearest neighbor visualization on Monks-2 problem. The graph shows nearest neighbor does not suit this kind of data.
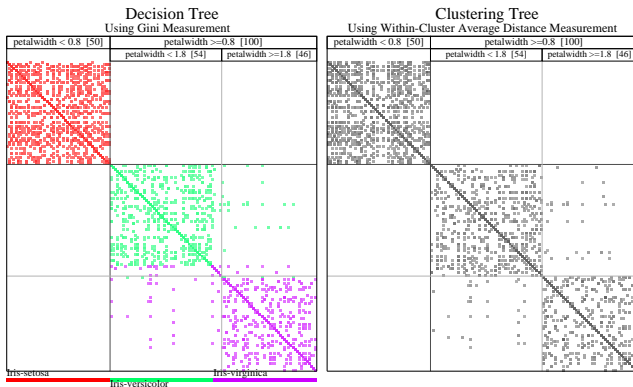
5 Decision Tree Visualization

Decision trees are one of the most popular classification models. A decision tree is constructed by partitioning an initial data set into disjoint subsets, and then recursively partitioning each of the subsets, until the subsets cannot be partitioned further. When a subset or a node stops to grow, it is called tree leaf and will be assigned a class label. Here we consider only the one attribute (univariate) partitioning rule. Typical attribute selection methods measure whether the partition on an attribute can lead to more purity on the subsets.

When applying shaded similarity matrix to decision tree visualization , the concept tree is simply a decision tree.
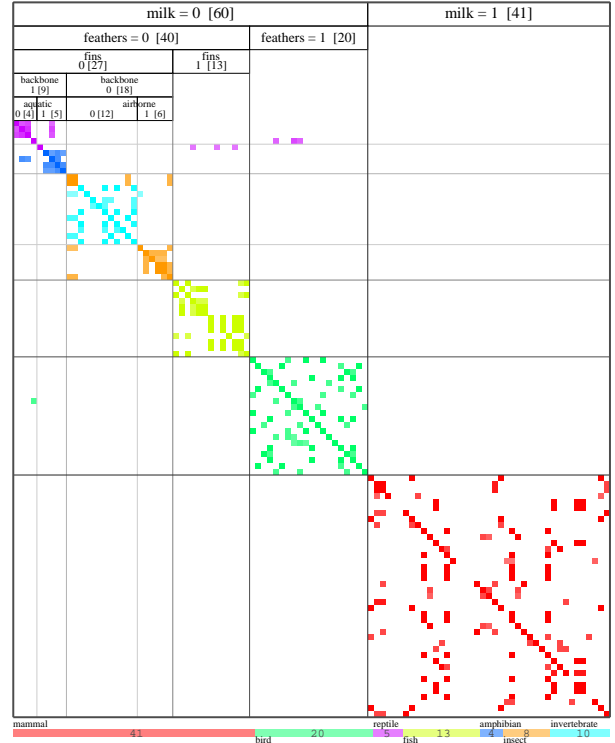
Again, similar to nearest neighbor visualization, whether or not shaded similarity matrix is appropriate for being applied to decision tree visualization depends on the definition of distance. It can be shown (see Appendix **A**) that when distance is defined only on class information, within-cluster average distance is equivalent to the Gini index, a popular attribute selection measurement in constructing decision trees [4]. In other words, if we build a clustering tree the same way as build a decision tree, we may obtain the same result. Of course, when generating a clustering tree, we do not have the class information. Fig. 5 is a demonstration on Iris data showing that we can build two same trees with or without class information. The decision tree is constructed using Gini index to select good attributes. The clustering tree is by using within-cluster average distance, and is displayed with gray shading because we do not have class information. In this paper, all decision trees are constructed using Gini measurement.

Fig. 6 is an illustration of decision tree visualization on Zoo data in the k-nearest neighbor setting. Zoo data set includes 16 non-category attributes: {hair, *feathers*, eggs, *milk*, *airborne*, *aquatic*, predator, toothed, *backbone*, breathes, venomous, *fins*, legs, tail, domestic, catsize}, where the highlighted ones are used by the decision tree. Using shaded similarity matrix to visualize decision tree can help us see the tree structure very clearly. Note that each square block indicates a tree leaf. In the Zoo data set, there are seven squares corresponding to seven classes. In most cases the nearest neighbors of each instance share the same square block with that instance.

Figure 6. Decision tree visualization on Zoo.

6   Ensemble Classifier Visualization

An ensemble classifier consists of a collection of individual classifiers, where each individual can make contribution to the final classification. Bagging and boosting are two popular ensemble approaches [3, 16]. Previous work shows that ensemble classifier can greatly improve the accuracy when small training data variation can lead to a quite different classification model [12]. For example, decision tree models are sensitive to the training data.

There are several reasons for developing visualization approaches for ensemble classifiers. First, scalability is a problem we can not avoid in any visualization. One popular method to handle large scale data is to use sampling technique. Much work has been done to deal with the reliability issue raised by sampling [15]. One solution is to develop an ensemble classifier, where each individual is generated from one sample data. Second, ensemble classifier was thought as an approach to improve the accuracy *at the cost of simplicity*. However, as we may think, with different classification models generated from different samples, we could have a deeper understanding of the data from different perspectives.

In this paper, we concern only about the ensemble decision tree visualization.

Figure 5. Decision tree and clustering tree visualization on *Iris*. Distance threshold setting with threshold=0.1**.**

6.1 Ensemble decision tree visualization

To understand data from as many perspectives as possible, one approach is to acquire all different individual classifiers. However, this is not feasible. A better way might be to heuristically compute the individual classifiers which satisfy some constraints. For example, given a training data set and some constraints such as training error threshold, the goal is to search a set of decision trees satisfying the constraints in the space of all possible decision trees.

Here we apply the boosting algorithm AdaBoost to generate different tree [7]. It works by repeatedly running a given weak learning algorithm (e.g., a simple decision tree) on various distributions over the training data, and then combining the classifiers produced by the weaker learner into a single composite classifier. It should be noted that any algorithm can not guarantee to find all good different trees.

We use Iris data set as an example for showing the visualization of ensemble classifier. Fig. 7 shows four individual classifiers whose training error is below a pre-defined threshold 0.1. From the attribute-class value mapping bars (refer to [1]) in Fig. 8, it is easy to see that the most related attributes are pw and pl (i.e., $petalwidth$ and $petallength$). That explains why four pairs of attributes (pl, pl), (pw, pl), (pw, pw), (pl, pw) can grow the best trees.
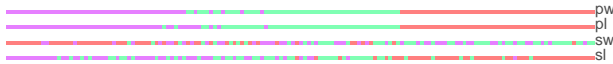
Figure 8. Attribute-class value mapping bars. There are 4 bars corresponding to 4 attributes. Each bar consists of 150 dots (150 examples in the set) where each dot represents an attribute value, and the 150 dots are sorted according to their attribute values. The dot color indicates the class label. It is apparent that the attributes pw and pl have strong correlation with the class.

6.2 Ensemble decision tree visualization on large data sets

To handle large data sets, our basic method is again to use sampling technique. Currently we implement random sampling only. More sophisticated sampling algorithm that can help find decision trees as different as possible, given the number of trees, will be explored in the future.

The purpose of this section is to see if it is effective to use simple random sampling with very small sample size. To this end, we test the ensemble classifier on

5 *Statlog* data sets: *Satimage, Segment, Shuttle, Australian*, and *DNA*. For data description, please see Table 4. The reason to use these 5 Statlog data sets is because Ankerst used them as benchmark in his PBC system [1]. The experimental results in Table 5 show that even with very small sample size, the ensemble classification accuracy is comparable with PBC using complete training data set. And the average tree size of ensemble classifier using small sample is significantly smaller than PBC. For an example of ensemble classifier visualization on *Satimage* (satellite image) data, please refer to: http://www.uiuc.edu/~junwang4/publications/infovis/satimage.eps

Table 4. Data description of 5 Statlog data sets. For *Segment* and *Australian*, we use 10-fold cross-validation to evaluate the accuracy. For *Satimage*, *Shuttle* and *DNA*, the size only indicates that of the train set. There are separate test set for these three data sets.

Dataset	size	classes	categorical	numerical
Satimage	4435	6	0	36
Segment	2310	7	0	19
Shuttle	43500	7	0	9
Australian	690	2	6	8
DNA	2000	3	60	0

Table 5. Comparison of accuracy and tree size of Ankerst's PBC system and our ensemble algorithm using random sampling technique. The sample size is set to 100 and the number of individual classifiers in an ensemble is 9.

Dataset	Accuracy		Tree size	
	PBC	Ensemble	PBC	Ensemble
Satimage	83.5	80.5	33	6-7
Segment	94.8	91.2	21.5	7-8
Shuttle	99.9	99.5	8.9	4-5
Australian	82.7	84.7	9.3	5-6
DNA	89.2	91.9	18	7-8

6.3 Discussion

Though ensemble classifier visualization has the above advantages, there are several issues needed to mention. First, the number of different good trees might be huge. For example, when there are lots of redundant or strongly correlated attributes, the number of different trees can be exponential to the number of related attributes. Digit recognition data set [4] is such an example. All seven attributes
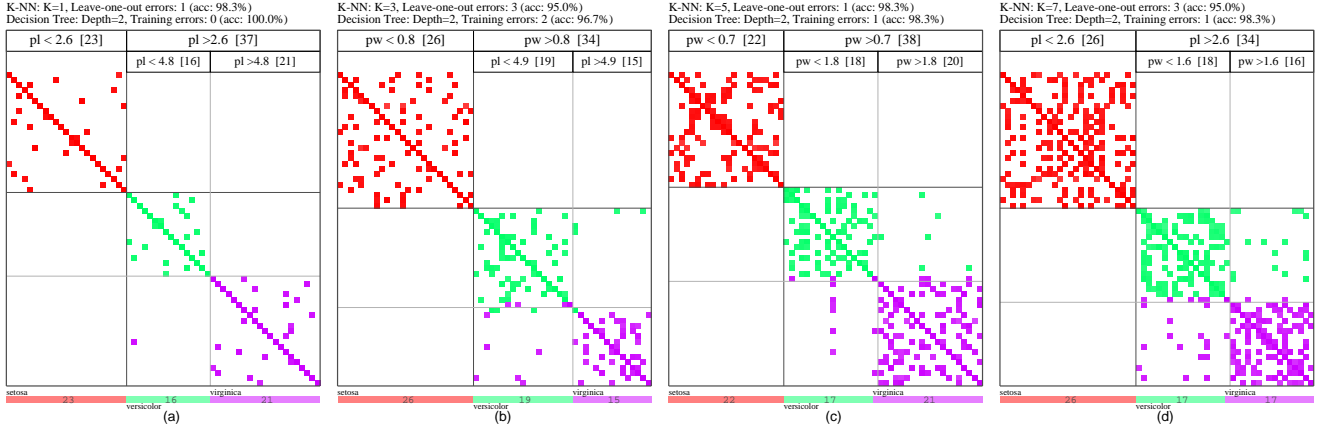
Figure 7. Ensemble decision tree visualization on *Iris*. The setting is k-nearest neighbor, where k increases from 1 to 7 step by 2. The sample size is 60.

(horizontal and vertical lights in on-off combinations) are strongly related. In this case, we need alternative tools to help user understand why the tree structure might be so unstable. *Pixel bar charts* technique could be such a tool [9].

Second, we may need to have a measurement of the difference or distance of two tree structures. If two trees are slightly different according to the measurement, we would like to remove it from the ensemble and try to find a new different tree. The distance of two trees can be defined as the prediction error on a common test set. If the two trees are generated from different samples, we can use the union of the two samples as the test set.

Third, having more than one trees implies that the organization of trees should be considered. One simple way is to set a priority for each attribute, and then apply some mechanisms such as sorting with combinatory attributes or fields. Another interesting approach might be to apply some clustering techniques to group similar trees together according to the definition of tree distance.

7 Summary and Future work

This paper proposed a new approach for ordering objects such that similar objects have adjacent positions in a visual space. Based on the assumption that "similar objects share same concepts or patterns", we transform the object ordering problem to concept organization. Due to many benefits of concept tree (e.g., easy to understand), we adopt concept tree as our concept organization means. As two applications, we investigate how to use concept tree based ordering with shaded similarity matrix in visualizing two popular classification methods: nearest neighbor and decision tree. We also explored a method to solve large data sets by using ensemble classification and sampling techniques.

This paper has not explored the following interesting issues. First, it deserves to apply concept tree based ordering to other domains rather than shaded similarity matrix. Second, can weighted distance definition or nonlinear distance definition improve the grouping quality when using typical Euclidean distance failed to reveal the within-class clusterings? And how to use visualization to help find a better distance definition?

Third, can we design a sampling approach which can guarantee the convergence of an ensemble classifier whose individual classifiers satisfy some constraints such as the training error threshold? In other words, after a convergence point, taking more rounds of sampling will not change the ensemble. This seems to be a theoretical question related more to the (online) computational learning instead of visualization. However, the solution to this problem can provide users a more reliable ensemble classifier visualization.

Acknowledgments

References

[1] M. Ankerst, M. Ester, and H.-P. Kriegel. Towards an effective cooperation of the user and the computer for classification. In *Proceedings of the Sixth International Conference on Knowledge Discovery and Data Mining*, pages 179–188, 2000.

[2] A. Beygelzimer, C.-S. Perng, and S. Ma. Fast ordering of large categorical datasets for better visualization. In *Proceedings of the Seventh International Conference on Knowledge Discovery and Data Mining*, pages 239–244, 2001.

[3] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[4] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Chapman & Hall, 1984.

[5] A. Buja and Y.-S. Lee. Data mining criteria for tree-based regression and classification. In *Proceedings of the Seventh International Conference on Knowledge Discovery and Data Mining*, pages 27–36, 2001.

[6] B. V. Dasarathy. *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. IEEE Computer Society Press, Los Alamitos, California, 1990.

[7] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the 13th International Conference on Machine Learning*, pages 148–156, 1996.

[8] N. Gale, W. Halperin, and C. Costanzo. Unclassed matrix shading and optimal ordering in hierarchical cluster analysis. *Journal of Classification*, 1:75–92, 1984.

[9] D. A. Keim, M. C. Hao, U. Dayal, and M. Hsu. Pixel bar charts: A visualization technique for very large multi-attributes data sets. *Information Visualization*, 1, 2002.

[10] P. Langley. *Elements of Machine Learning*. Morgan Kaufmann Publishers, 1996.

[11] R. L. Ling. A computer generated aid for cluster analysis. *Communications of the ACM*, 16(6):355–361, 1973.

[12] R. Maclin and D. Opitz. An empirical evaluation of bagging and boosting. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, pages 546–551. AAAI Press/MIT Press, 1997.

[13] C. J. Merz, P. M. Murphy, and D. W. Aha. UCI repository of machine learning databases, http://www.ics.uci.edu/~mlearn/mlrepository.html, 1997. Dept. of Information and Computer Science, University of California at Irvine.

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

[15] F. Provost and V. Kolluri. A survey of methods for scaling up inductive algorithms. *Data Mining and Knowledge Discovery*, 2:1–42, 1999.

[16] R. E. Schapire. The boosting approach to machine learning: An overview. In *MSRI Workshop on Nonlinear Estimation and Classification*, 2002.

[17] S. B. Thrun, J. Bala, E. Bloedorn, I. Bratko, B. Cestnik, J. Cheng, K. D. Jong, S. D. zeroski, S. E. Fahlman, D. Fisher, R. Hamann, K. Kaufman, S. Keller, I. Kononenko, J. Kreuziger, R. S. Michalski, T. Mitchell, P. Pachowicz, Y. Reich, H. Vafaie, W. V. de Welde, W. Wenzel, J. Wnek, and J. Zhang. The MONK's problems: A performance comparison of different learning algorithms. Technical Report CS-91-197, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1991.

[18] D. Wishart. ClustanGraphics3: Interactive graphics for cluster analysis. In W. Gaul and H. Locarek-Junge, editors, *Classification in the Information Age*, pages 268–275. Springer-Verlag, 1999.

Appendix A

The purpose of this appendix is to show that when distance is defined only on class, within-cluster average distance is equivalent to the Gini index [4].

PROOF. We consider only the two-class situation[3]. The class labels are denoted 0 and 1. Given a split into left and right subsets, let $p_L^0 + p_L^1 = 1$, $p_R^0 + p_R^1 = 1$ be the probabilities of label 0 and 1 on the left and on the right, respectively. Denoting with $p_L + p_R = 1$ the probabilities of the left and the right subset given the parent data set. Then Gini index takes this form:

$$Gini = p_L p_L^0 p_L^1 + p_R p_R^0 p_R^1$$

Now suppose the distance of two examples e_1 and e_2 is defined only on class information:

$$\delta(e_1, e_2) = \begin{cases} 0 & \text{if } e_1 \text{ and } e_2 \text{ share the same class} \\ 1 & \text{otherwise} \end{cases}$$

Then the within-cluster average distance weighted on the left and right subsets can be written as:
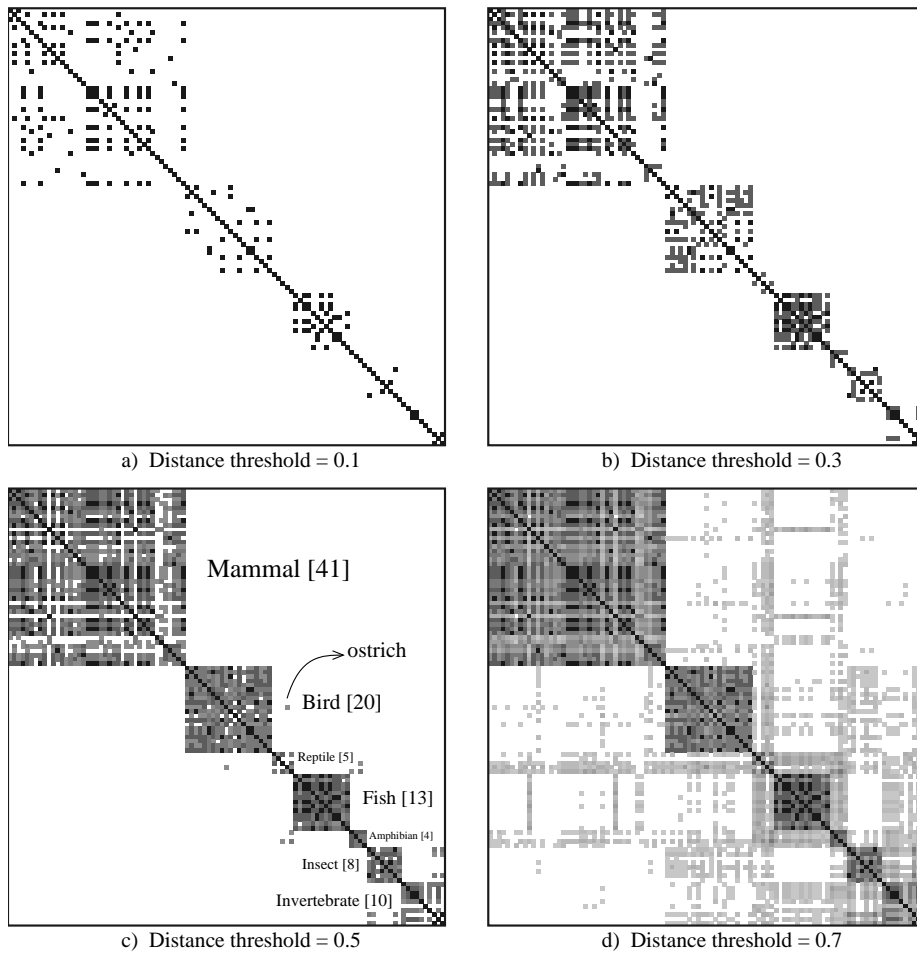
$$
\begin{aligned}
& p_L \frac{\sum_{e_i, e_j \in S_L} \delta(e_i, e_j)}{|S_L|^2} + p_R \frac{\sum_{e_i, e_j \in S_R} \delta(e_i, e_j)}{|S_R|^2} \\
=\ & p_L \frac{2|S_L^0||S_L^1|}{|S_L|^2} + p_R \frac{2|S_R^0||S_R^1|}{|S_R|^2} \\
=\ & 2(p_L p_L^0 p_L^1 + p_R p_R^0 p_R^1) \\
\propto\ & Gini
\end{aligned}
$$

Where S_L, S_R denoting the left and right subset, respectively; and $|S_L^0|$, $|S_L^1|$ denoting the number of examples with label 0 and 1 in the left subset, respectively; and so on. □

---

[3]Some notations are taken from [5]

a) k-nearest neighbor setting: k=7

b) Distance threshold setting: Threshold=0.4
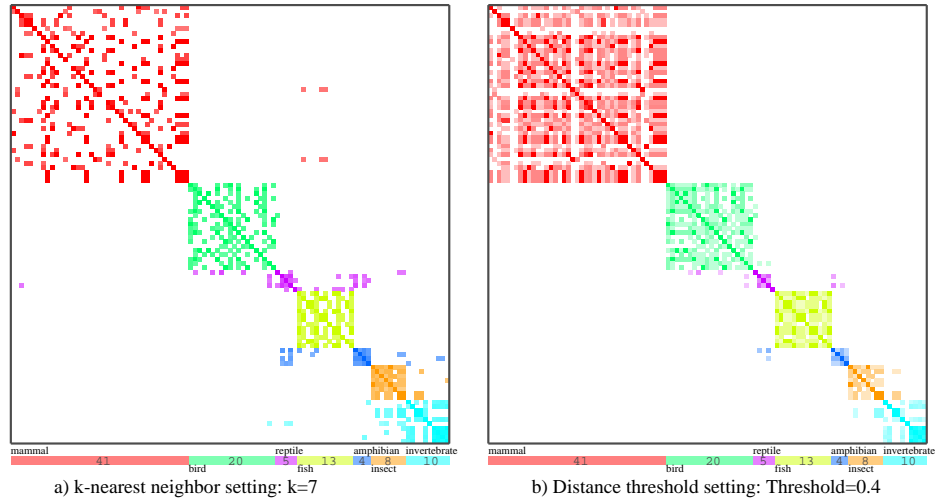
Figure 9. Two different control settings on whether to display a matrix cell or not.

a) Distance threshold = 0.1

b) Distance threshold = 0.3

Mammal [41]

ostrich

Bird [20]

Reptile [5]

Fish [13]

Amphibian [4]

Insect [8]

Invertebrate [10]

c) Distance threshold = 0.5

d) Distance threshold = 0.7

Figure 10. Gradually increasing the distance threshold shows how the regularities and outliers emerge.