An alternative method for the prediction certainty calculation in decision trees

Eduardo P. Costa[a], Sicco Verwer[b], and Hendrik Blockeel[a,c]

[a] Department of Computer Science, Katholieke Universiteit Leuven, Leuven, Belgium
[b] Institute for Computing and Information Sciences, Radboud University Nijmegen, The Netherlands
[c] Leiden Institute of Advanced Computer Science, Universiteit Leiden, Leiden, The Netherlands
eduardo.costa@cs.kuleuven.be,siccoverwer@gmail.com,hendrik.blockeel@cs.kuleuven.be

Abstract. In classification, we often need classifiers that do not only have high accuracy, but can also tell how certain they are about a prediction. In decision trees, a predictions certainty is usually estimated using the class distribution in the leaf responsible for the prediction. We introduce an alternative method that yields better estimates. For each instance to be predicted, the proposed method induces a number of decision trees based on different versions of the training set. These versions are obtained by inserting the instance in the training set with one of the possible labels for the target attribute; this procedure is repeated for each one of the labels. By comparing the outcome of the different trees, the method can identify instances that might present some difficulties to be correctly classified, and attribute some uncertainty to their prediction.

Keywords: decision trees, prediction certainty

1 Introduction

In classification, we often need classifiers that do not only have high accuracy, but can also tell how certain they are about a prediction. This is one reason for evaluating classifiers based on their area under the ROC curve (AUC), instead of only on accuracy: AUC penalizes incorrect predictions more strongly when they are made with higher certainty.

In decision trees (DTs), a prediction's certainty is usually estimated using the class distribution in the leaf responsible for the prediction. We introduce an alternative procedure that yields better estimates. Basically, for each instance to be predicted, we induce a number of trees based on different versions of the training set. These versions are obtained by inserting the instance in the training set with each one of the possible labels for the target attribute. By investigating how stable (in terms of the predicted label) the predictions are, we can identify instances for which predictions might be untrustworthy and use this information in the prediction certainty calculation.

We evaluate our method on a number of datasets, and compare its results to those of a standard DT learner and of ensembles of trees. The results show that the new method yields equally accurate results, and estimates its prediction certainty more reliably than the other methods.

Note that the ideas we introduce in this paper are not restricted to the context of DTs. In fact, the algorithm we introduce can be generally applicable to other machine learning methods that might present problems in the prediction certainty calculation.

2 Proposed method

A decision tree (DT) [ref] is a tree-shaped predictive model that maps instances from an input domain X to some output domain Y. In this model, each internal node contains a test on some attribute defined for elements of X, and each leaf is associated to a value $y \in Y$. The mapping of an instance $x \in X$ to a value y is obtained by sorting down the tree from the root to some leaf node. An example of a DT can be seen in Figure 1.

In the classification context, DTs are used to map an instance x to a nominal value y, which is usually called class y. Standard DT learners estimate a classification's certainty by using the class distribution in the leaf responsible for the prediction. For example, if an instance x ends up in a leaf that contains 80% positive instances, the positive class probability is set to 0.8. A problem with this procedure is that it tends to be overconfident in its predictions, as shown in Section 2.1. To overcome this problem, we introduce an alternative procedure for the prediction certainty calculation, which is described in Section 2.2. We present some alternatives in the implementation of the method in Section 2.3.

2.1 Intuition of the method

Consider, for example, the DT represented in Figure 1. This tree was generated for an iteration of the leave-one-out validation procedure for the well-known iris dataset [1]. The test instance x_1 belongs to class Iris-virginica, but it ends up being classified in leaf 4, where 98% of the training instances belong to class Iris-versicolor. This high confidence misclassification happens because x_1 is a "border case" (i.e, x_1 is somewhere in the border between classes Iris-versicolor and Iris-Virginica) and is therefore difficult to be correctly classified.

The same problem occurs for the other six misclassified instances of iris, which has 150 instances in total. Even though one could argue that seven instances is just a small fraction of the total number of instances, the main problem is that they are misclassified with a similar prediction certainty as the correctly classified instances. This illustrates how the prediction certainty given by DTs may be overconfident and, consequently, untrustworthy.

One idea to overcome this problem is to identify instances that might present some difficulties to be correctly classified, and to attribute some uncertainty to

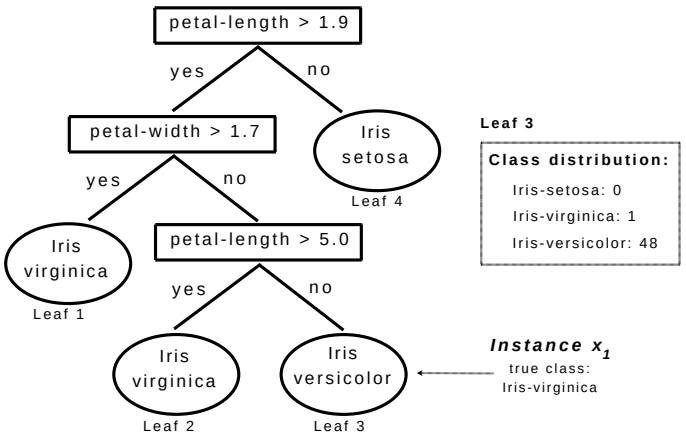

Fig. 1. Example of a misclassification with high prediction certainty, which was obtained from an iteration of the leave-one-out validation procedure for the iris dataset.

their prediction. This can be achieved by introducing small changes in the training data and checking how this affects the outcome of the DT; the underlying idea being that these changes should not have a strong effect on the outcome of the DT, unless we are dealing with cases for which the classification is more challenging, such as border cases. More specifically, we are interested in investigating the following questions: (1) "If we add the test instance to the training set with a different label than the correct one, will the DT learner find a tree that explains this instance according to the wrong label?"; (2) "How certain will the tree be about this prediction?"; (3) "How does this situation compare to the one where the instance is added to the training set with the correct label?".

In Figure 2 we show two DTs built for the same data used in Figure 1, with the difference that the test instance x_1 was included in the training data. For the left tree, x_1 was included with the correct label, Iris-virginica, while for the right one, x_1 was included with the wrong label Iris-versicolor. Observe that the trees make different predictions based on the pre-defined label for x_1. This shows that the learned DT is highly dependent on the label of x_1. Intuitively, we would like to have confident predictions only for cases where the predicted label is independent of the actual label.

2.2 Description of the method

Using the intuition just described, the proposed method estimates the prediction certainty by comparing trees generated for the test instance x using different labels. As the correct label of x is not known during training, the method tries all possibilities for the target attribute. More specifically, to classify an instance x, the method builds k trees, where k is the number of possible labels for the

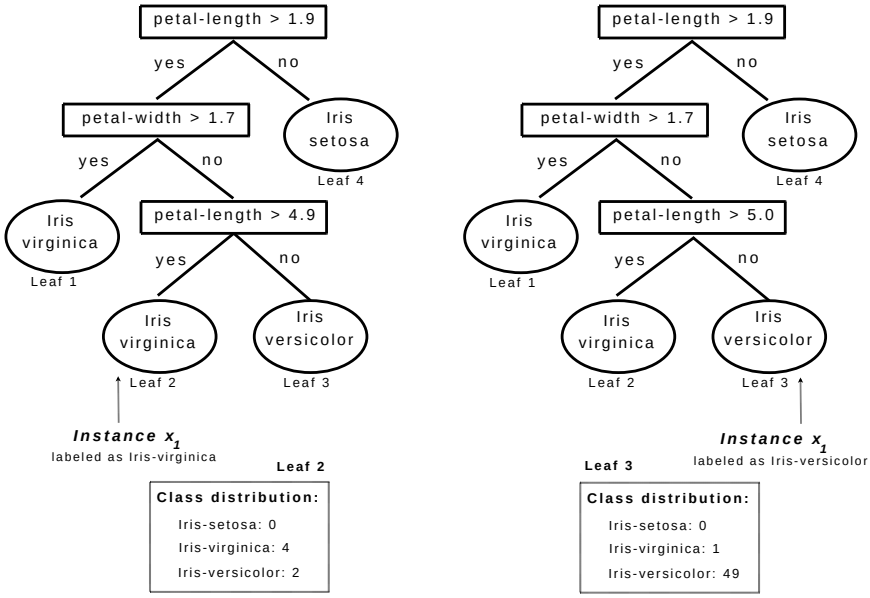

Fig. 2. DTs built for the same data used in Figure 1, with the difference that the test instance x was included in the training data. Left: x was included with the correct label, Iris-virginica. Right: x was included with the wrong label Iris-versicolor.

target attribute. For each label, we insert x in the training set with that label and induce a DT. In the end, the final prediction and its certainty for x are obtained by combining the prediction of all the trees.

The following pseudocode shows the described algorithm. In the pseudocode, T is the original training data, Y is the set with the possible target labels, the prodecure induceDT builds a DT for the given training data (in this case, the modified training data T'), the vector DT stores the induced DTs, and the procedure CombinePredictions is responsible to combine the predictions of all k trees (we present three possibilities for this combination in the next section), and returns the final prediction and its certainty.

Pseudocode for the function that returns the prediction for an instance

```
function Classify_Instance (x,T,Y) returns prediction

    for i := 1 to k :
     T' := T U (x,Y[i])
     DT[i] := induceDT(T')
    pred := CombinePredictions(DT)

    return pred
```

The method just described performs a form of tranducstive learning. In this kind of learning, given a dataset and one or more specific unlabeled instances, the goal is to generate a classifier that makes predictions for that/those instance(s). In this case, it is not required that a generally applicable classifier is learned. However, differently from the standard definition of transductive learning, in the proposed method, the instance to be predicted is associated to a certain label when included in the training set.

We also have an additional constraint during the tree induction procedure. Namely, the attribute values of the instance to be predicted is not used during the selection of the best test to split the data. This constraint is especially important for instances that are close to the border of two or more classes. Consider, for example, the binary classification problem illustrated in Figure 3. In this example, the instance x, which is represented in the figure as a circle, is a positive instance. As can be seen in the figure, x is far enough from the negative class (so that a standard DT would not have problems classifying it correctly), but, among all the positive instances, x is the closest to the negative ones. If we allow our method to use the attribute values of x, it will always choose a decision boundary that perfectly separates the instances depending on the label attributed to x (as shown in Figures 3.a and 3.b). This would lead our method to conclude that x is a difficult case to be classified, while it is not. To avoid this, the attribute values of x are hidden from the method during the test selection.

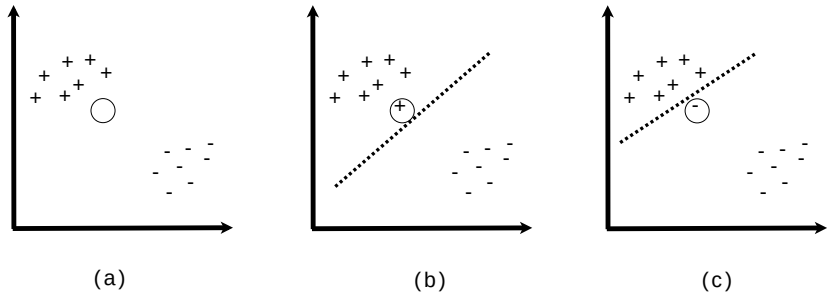

Fig. 3. Decision boundary.

2.3 Alternatives to combine the tree predictions

Different possibilities can be used to combine the predictions of the trees induced by our method. We describe three possibilities in this section, which are empirically evaluated in Section 3.

The first one consists of, for each tree, calculating the class distribution for the leaf node L responsible for the prediction and then averaging the class probabilities for each class (Equation 1). In the equation, $PredCert(a)$ is the prediction

certainty for class a, $|a|$ is the number of instances belonging to that class within L, and $|L|$ is the total number of instances in that leaf.

$$PredCert(a) = \frac{\sum_i^k \frac{|a|}{|L|}}{k} \tag{1}$$

Alternatively, we test a variant of this possibility by applying the Laplace smoothing (Equation 2). With this smoothing, we avoid that small leaves have too much effect in the final prediction, by adding some weight in the way we combine the class probabilities.

$$PredCertLap(a) = \frac{\sum_i^k \frac{|a|+1}{|L|+k}}{k} \tag{2}$$

As a third possibility, we test a strategy that first combines the leaf nodes responsible for the prediction, considering all trees, and then calculate the prediction certainty based on the class distribution in the combined leaf (Equation 3). In the equation $|a_{comb}|$ is the number of instances belonging to class a in the combined leaf and $|L_{comb}|$ is the total number of instances in that leaf. By combining leaf nodes, this calculation gives more weight to larger leaves.

$$PredCertComb(a) = \frac{|a_{comb}|}{|L_{comb}|} \tag{3}$$

3 Empirical evaluation

We implemented our method as an extension of the DT learner Clus[1]. In total we used 55 randomly selected UCI datasets [1] in the experiments. First, we used 6 datasets for fine-tuning and validation - we tested different alternatives to combine the class probabilities of the trees. We then evaluated the method on the other 49 datasets.

As the proposed method was designed to make predictions for one test instance at a time, we performed the experiments using leave-one-out validation. The results are shown in terms of accuracy and AUC. To calculate the AUC, we generate a ROC curve for each class against all the other ones and calculate the AUC for that curve. We then average all AUC values to obtain the final AUC.

3.1 Validation and fine-tuning

We investigated three possibilities for the combination of the trees, as mentioned in Section 2. We call Clus-Mod the variant of the method that uses Equation 1; Clus-ModL the variant that uses Equation 2; and Clus-ModC the variant that uses Equation 3. The results presented by the three methods for the six validation datasets are shown in Table 1. The best results are shown in bold.

[1] http://dtai.cs.kuleuven.be/clus/

Table 1. Accuracy and AUC for the three variantes of the proposed method - ClusMod, ClusModP, and ClusModC - for the six validation datasets. The best results are shown in bold.

	Accuracy			AUC		
	Clus-Mod	Clus-ModL	Clus-ModC	Clus-Mod	Clus-ModL	Clus-ModC
Iris	**0,940**	**0,940**	**0,940**	**0,983**	**0,983**	0,981
Vote	**0,966**	**0,966**	0,959	**0,975**	**0,975**	0,974
Zoo	**0,921**	0,911	0,911	**0,991**	0,989	0,990
Anneal	**0,992**	0,986	0,984	**0,993**	**0,993**	**0,993**
Vehicle	0,713	**0,733**	**0,733**	0,902	0,909	**0,911**
Sonar	0,654	**0,760**	0,750	0,354	**0,842**	0,824

In general, all variants of the method obtained similar results in terms of both accuracy and AUC. The only exception is the dataset sonar. For this dataset, Clus-Mod obtained very poor results, especially for AUC. It turned out that, for many instances in this dataset, small pure leaves, which are usually very untrustworthy, end up giving a too high confidence to the wrong class. This undesirable effect is not present in Clus-ModL and Clus-ModC. Clus-ModL avoids that small leaves have too much effect in the final prediction through the Laplace smoothing, which adds some weight in the way we combine the class probabilities. Clus-ModC obtains a similar behavior by combining leaf nodes: when a smaller node is combined with a larger one, the final class distribution is more strongly influenced by the latter.

As the results of Clus-ModL were slightly better than those of Clus-ModC, we only consider the former in the evaluation presented in the next section.

3.2 Evaluation on the test datasets

We evaluate Clus-ModL on the remaining 49 datasets selected for the experiments. We compare Clus-ModL to the original Clus (Clus-Orig). One could argue, however, that this comparison is not entirely fair since our method uses multiple trees to make the final prediction, and it has long been known that ensembles tend to perform better than single trees [REF]. Therefore, to ensure that an improvement of Clus-ModL is not simply due to "ensemble effects", we also compare to standard bagging (Clus-Ens) with the same number of trees, averaging class probabilities in the same way.

The results are summarized in tables 2, 3 and 4. Tables 2 and 3 show how many times each method wins in terms of accuracy and AUC, respectively, for the pairwise comparisons "Clus-ModL vs. Clus-Orig" and "Clus-ModL vs. Clus-Ens". Table 4 shows the average accuracy and AUC over the 49 datasets for the three methods. Additionally, we plot a graph for the pairwise comparisons for accuracy (Figure 4) and for AUC (Figure 5). The detailed results for each dataset is shown in Table5, in the appendix.

For the results regarding accuracy, Clus-ModL obtained: 22 wins, 4 ties and 23 losses in the comparison with Clus-Orig; and 23 wins and 26 losses in the

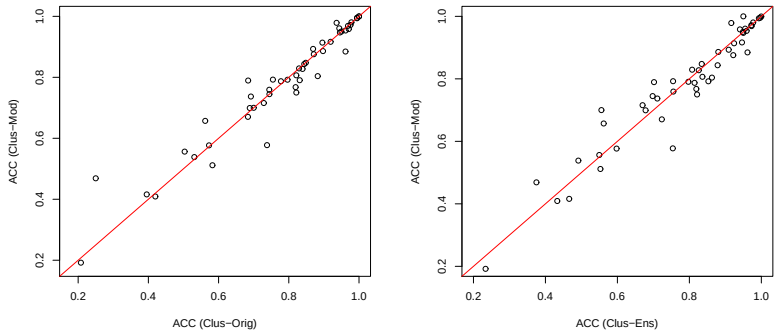

Fig. 4. Accuracy for the pairwise comparisons "Clus-ModL vs. Clus-Orig" (left) and "Clus-ModL vs. Clus-Ens" (right).

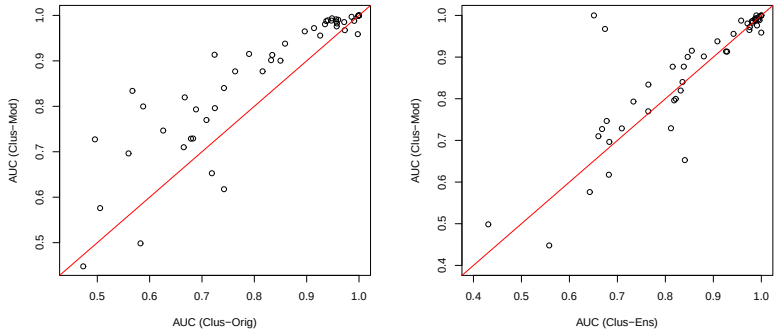

Fig. 5. AUC for the pairwise comparisons "Clus-ModL vs. Clus-Orig" (left) and "Clus-ModL vs. Clus-Ens" (right).

Table 2. Number of wins, in terms of accuracy, for each method in the pairwise comparisons "Clus-ModL vs. Clus-Orig" and "Clus-ModL vs. Clus-Ens".

Clus-ModL vs. Clus-Orig			Clus-ModL vs. Clus-Ens		
Clus-ModL	Clus-Orig	Ties	Clus-ModL	Clus-Ens	Ties
22	23	4	23	26	0

Table 3. Number of wins, in terms of AUC, for each method in the pairwise comparisons "Clus-ModL vs. Clus-Orig" and "Clus-ModL vs. Clus-Ens".

Clus-ModL vs. Clus-Orig			Clus-ModL vs. Clus-Ens		
Clus-ModL	Clus-Orig	Ties	Clus-ModL	Clus-Ens	Ties
38	7	4	29	18	2

Table 4. Average accuracy and AUC for Clus-Orig, Clus-ModL and Clus-Ens

Accuracy			AUC		
Clus-Orig	Clus-ModL	Clus-Ens	Clus-Orig	Clus-ModL	Clus-Ens
0.782	0.783	0.785	0.817	0.871	0.856

comparison with Clus-Ens. The average accuracy over the 49 test datasets is very similar for the three methods, with Clus-ModL being slightly better than Clus-Orig and slightly worse than Clus-Ens. These results show that the proposed method yields equally accurate results.

The results reported for AUC show that Clus-ModL estimates its prediction certainty more reliably than the other methods. When compared to Clus-Orig, Clus-ModL obtained 38 wins, 4 ties, 7 losses, and a better average AUC. Clus-ModL also obtains better results when compared to Clus-Ens: we observe 29 wins, 2 ties and 18 losses for Clus-ModL, and a better average AUC. Moreover, the results for the latter show that the improvement of Clus-ModL, in terms of AUC, is not simply due to use of multiple trees to determine the final prediction. This improvement is statistically significant at the level 7% according to the Wilcoxon Signed-Rank Test [REF].

4 Conclusions

In this paper we proposed a new method for estimating the prediction certainty of classifiers. The new method was implemented as an extension of the decision tree learner Clus, but the ideas investigated in this paper can also be applied to other machine learning methods.

The proposed method builds a decision tree for a data consisting of the training data plus the instance to be classified, which is associated to one of the possible target attribute labels. This procedure is repeated for every possible label, and in the end all induced trees are compared. This comparison allows us to identify instances that might present some difficulties to be correctly classified, and to attribute some uncertainty to the prediction for these instances.

We evaluated our method on a number of UCI datasets, and compared it to the original Clus and to standard bagging. The results showed that the new

method yields equally accurate results, and estimates its prediction certainty more reliably than the other methods.

As the proposed method was designed to make predictions for one instance at a time, a new classification model needs to be built every time we want to classify a new instance; while a standard decision tree learner does not have this constraints. Therefore, a possible extension of this work is to investigate how the same principles used in this paper can be applied for the case where more than one instance need to be classified.

Acknowledgments

This research was supported by projects G.0413.09 "Learning from data originating from evolution" and G.0682.11 "Declarative experimentation", funded by the Research Foundation - Flanders (FWO-Vlaanderen).

References

1. Frank, A., Asuncion, A. UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science (2010).
2. Blockeel, H., De Raedt, L., Ramon, J. Top-down induction of clustering trees. Proc. of the 15th International Conference on Machine Learning (1998).

Appendix: Detailed results

Table 5 shows the detailed results for the 49 datasets used in the evaluation described in Section 3.2.

Table 5. All results.

	Accuracy			AUC		
Dataset	Clus-Orig	Clus-Mod	Clus-Ens	Clus-Orig	Clus-Mod	Clus-Ens
balance-scale	0.778	0.787	0.814	0.763	0.877	0.815
StatlogHeart	0.744	0.759	0.756	0.742	0.840	0.836
german	0.692	0.737	0.711	0.626	0.747	0.678
car	0.976	0.972	0.972	0.986	0.997	0.996
dermatology	0.962	0.954	0.959	0.957	0.976	0.991
BreastCancerDiagnostic	0.946	0.947	0.950	0.958	0.982	0.981
chess	0.994	0.994	0.993	0.998	0.999	0.999
PrimaryTumor	0.395	0.416	0.466	0.708	0.770	0.764
Balloons	1.000	1.000	0.950	1.000	1.000	0.990
TeachingAssistant	0.503	0.556	0.550	0.679	0.729	0.709
mushroom	1.000	1.000	1.000	1.000	1.000	1.000
OpticalRecognition	0.869	0.893	0.909	0.940	0.989	0.988
SpectfHeart	0.738	0.578	0.754	0.742	0.618	0.682
ThyroidDisease	0.978	0.980	0.977	0.991	0.988	0.988
PenDigits	0.919	0.917	0.946	0.960	0.990	0.994
Hill-Valley	0.583	0.512	0.553	0.582	0.498	0.431
StatlogLandSat	0.844	0.844	0.879	0.914	0.972	0.976
hepatitis	0.819	0.768	0.819	0.725	0.796	0.818
glass	0.882	0.804	0.863	0.834	0.913	0.929
labor	0.684	0.789	0.702	0.719	0.653	0.841
lymph	0.831	0.791	0.797	0.724	0.913	0.926
diabetes	0.745	0.745	0.698	0.689	0.793	0.733
nursery	0.994	0.996	0.997	0.998	0.959	1.000
splice	0.950	0.950	0.949	0.957	0.987	0.981
abalone	0.208	0.192	0.234	0.495	0.727	0.668
bridges	0.562	0.657	0.562	0.567	0.834	0.765
cmc	0.531	0.538	0.492	0.665	0.710	0.660
ecoli	0.821	0.807	0.836	0.790	0.915	0.855
arrhythmia	0.684	0.670	0.723	0.683	0.729	0.812
lungCancer	0.250	0.469	0.375	0.560	0.696	0.683
pageBlocks	0.969	0.969	0.973	0.935	0.981	0.971
postOperativeData	0.700	0.700	0.556	0.973	0.967	0.674
haberman	0.729	0.716	0.670	0.473	0.448	0.558
spam	0.936	0.979	0.917	0.938	0.987	0.958
solarFlare	0.829	0.829	0.808	1.000	1.000	0.651
promoters	0.755	0.792	0.755	0.816	0.877	0.838
breastCancer	0.689	0.699	0.678	0.505	0.576	0.642
creditApproval	0.839	0.828	0.826	0.832	0.902	0.880
ionosphere	0.897	0.886	0.880	0.859	0.938	0.908
adult	0.848	0.848	0.834	0.850	0.900	0.846
vowel	0.420	0.409	0.433	0.588	0.800	0.822
segment	0.896	0.914	0.924	0.956	0.991	0.986
japaneseVowels	0.796	0.792	0.853	0.896	0.965	0.975
audiology	0.962	0.885	0.962	1.000	1.000	1.000
hayesRoth	0.821	0.750	0.821	0.926	0.956	0.942
tictac	0.971	0.958	0.941	0.972	0.985	0.980
yeast	0.573	0.577	0.598	0.667	0.820	0.832
wine	0.944	0.961	0.955	0.948	0.994	0.989
letter	0.871	0.876	0.922	0.947	0.989	0.996