Oracle Coached Decision Trees and Lists

Ulf Johansson[*], Cecilia Sönströd, and Tuve Löfström

CSL@BS Research Group
School of Business and Informatics
University of Borås, Sweden
{ulf.johansson,cecilia.sonstrod,tuve.lofstrom}@hb.se

Abstract. This paper introduces a novel method for obtaining increased predictive performance from transparent models in situations where production input vectors are available when building the model. First, labeled training data is used to build a powerful opaque model, called an *oracle*. Second, the oracle is applied to production instances, generating predicted target values, which are used as labels. Finally, these newly labeled instances are utilized, in different combinations with normal training data, when inducing a transparent model. Experimental results, on 26 UCI data sets, show that the use of oracle coaches significantly improves predictive performance, compared to standard model induction. Most importantly, both accuracy and AUC results are robust over all combinations of opaque and transparent models evaluated. This study thus implies that the straightforward procedure of using a coaching oracle, which can be used with arbitrary classifiers, yields significantly better predictive performance at a low computational cost.

Keywords: Decision trees, Rule learning, Coaching.

1 Introduction

There are situations in predictive classification where interpretable models are required, thus limiting the choice of technique to those producing transparent models. It is, however, generally known that these techniques are weaker in terms of predictive performance than techniques producing opaque models, and yet predictive performance is usually of the utmost importance, both for obtaining accurate predictions from the model and because accurate models have higher explanatory value. In real-life applications, prediction is usually a one-shot event, where a model is painstakingly built using available training data utilized in various ways, and then the model is applied once, and only once, to previously unseen, production, data. Usually, the only important evaluation criterion in this situation is accuracy on the production data. The two main ramifications of the above are that:

[*] This work was supported by the INFUSIS project (www.his.se/infusis) at the University of Skövde, Sweden, in partnership with the Swedish Knowledge Foundation under grant 2008/0502.

a) all work that goes into building the model is really only preparation for the prediction using production data
b) all means of maximizing production accuracy should be employed during model construction.

Few methods are, however, explicitly geared towards this. Various techniques, like cross-validation, hold out available data in order to estimate production accuracy, instead of trying to take advantage of all the available data. The only commonly observed constraint is that *the data on which the model is evaluated must never be used to build the model.* In many cases, however, this constraint is overly restrictive. In some situations, the only unknown data are the *target values* for the production data; all input vectors for the production data are available when building the model, and this data usually consists of a substantial number of instances. A typical example illustrating this is when predictive classification is used to target recipients of a marketing campaign and the training data consists of past response data. Here, the very same customer signatures that will later be used as production data input vectors are available to the data miner and could therefore be used for the model construction.

The purpose of this paper is to suggest a very straightforward, and yet effective, way of increasing production accuracy for transparent models when production data input vectors are available. The proposed method thus requires that bulk predictions are made, and is unsuitable when prediction instances arrive one at a time. The method, drawing inspiration from rule extraction, semi-supervised learning and transductive learning, utilizes a powerful opaque model to label production instances, yielding additional training instances which are then used to build the transparent model. In this context, we name the opaque model the *oracle*, since the target values it produces are treated as ground truth by the training regime of the transparent model.

2 Background

A predictive classification model is a function f mapping each instance $\boldsymbol{x}$ to one label in a predefined set of discrete classes $\{c_1, c_2, \ldots, c_n\}$. Since a predictive classification model should be able to assign every possible input vector to one of the predefined classes, the model must partition the input space; i.e., the classes have to be non-overlapping and exhaustive. Normally, predictive models are obtained by applying some supervised learning technique on historical (training) data. Most supervised learning techniques require that the correct value of the target variable is available for every training instance. A training instance, thus, consists of an input vector $\mathbf{x}(i)$ with a corresponding target value $y(i)$. The predictive model is an estimation of the function $y = f(\mathbf{x}; \theta)$ able to predict a value y, given an input vector of measured values $\mathbf{x}$ and a set of estimated parameters θ for the model f. The process of finding the best θ values is the core of the data mining technique.

For predictive classification, the most important criterion is accuracy, i.e., the number of correct classifications made by the model when applied to novel data should be as high as possible. For situations where a black-box prediction machine is sufficient, most data miners will use techniques like *artificial neural networks* (ANNs), *ensembles* or *support vector machines*, which all have a very good track record of producing accurate models, in a variety of domains. All these techniques produce, however, opaque models, making it impossible to assess the model, or even to understand the reasoning behind individual predictions. When models need to be comprehensible, the most straightforward option is to sacrifice accuracy by using techniques producing less accurate but transparent models, most typically decision trees like C4.5/C5.0 [1] and CART [2], or rule sets like RIPPER [3].

In this paper, we study the situation where transparent models are mandatory and the input vectors in the production set are already determined and available when building the model. In real-world applications, this is actually a very common scenario, meaning that the predictive model is explicitly built for the prediction task at hand. In this exact situation, the unlabeled production instances, i.e., the very same instances that later will be used for the actual prediction, could also be used for building the model.

2.1 Related Work

Semi-supervised learning uses both labeled and unlabeled data when building the models. The main motivation for semi-supervised learning is the fact that labeled data is hard or costly to obtain in many applications, while unlabeled data is often cheap and easily accessible. The overall goal is, of course, to produce better (more accurate) models by using both labeled and unlabeled data, compared to using only labeled data. In most situations, the number of available labeled instances is much smaller than the number of unlabeled instances. It should be noted that semi-supervised classification is still inductive since a model is ultimately built and used for the actual classification.

Naturally, several fundamental approaches to semi-supervised learning, as well as numerous variations, exist; for a good survey see [4]. The simplest approach is *self training*, where a model is first built using the labeled data only, and is then used to label unlabeled instances, thus increasing the size of the labeled data set. This process is normally repeated several iterations, but eventually the final classifier would be trained on a combination of initially labeled instances and instances labeled by the intermediate models. So, simply put, the classifier uses its own predictions to teach itself.

Our approach is somewhat similar to self training, but there are two main differences: First we use one (stronger) classifier to label the instances, and another (weaker but transparent) classifier for the final model. Technically, we refer to this technique as *coaching*. Second, since the purpose is increased accuracy on a specific production set, we explicitly utilize the fact that we have the corresponding production input vectors available.

Transductive learning also utilizes both labeled and unlabeled data, and the overall purpose is similar to our approach; i.e., to obtain high accuracy on specific instances; see e.g. [5]. But the main difference is again that we explicitly focus on situations where the final model must be transparent, leading to a process where a stronger model coaches a weaker.

Rule extraction is the process of generating a transparent model based on a corresponding opaque predictive model. Rule extraction has been heavily investigated in the neural network domain, and the techniques have been applied mainly to ANN models; for an introduction and a good survey of traditional methods, see [6]. There are two fundamentally different extraction strategies, *decompositional* (*open-box* or *white-box*) and *pedagogical* (*black-box*). Decompositional approaches focus on extracting rules at the level of individual units within a trained ANN, while pedagogical approaches treat the opaque model as a black box. The core pedagogical idea is to view rule extraction as a learning task, where the target concept is the function originally learnt by the opaque model. Black-box rule extraction is, consequently, an instance of predictive modeling, where each input-output pattern consists of the original input vector $\boldsymbol{x}_i$ and the corresponding prediction $f(\boldsymbol{x}_i;\theta)$ from the opaque model. Two typical and well-known black-box rule extraction algorithms are TREPAN [7] and VIA [8]. Naturally, extracted models must be as similar as possible to the opaque models. This criterion, called fidelity, is therefore a key part of the optimization function in most rule extraction algorithms. Most, if not all, rule extraction algorithms targeting fidelity use 0/1 fidelity, i.e., maximize the number of identical classifications.

An interesting discussion about the purpose of rule extraction is found in [9], where Zhou argues that rule extraction really should be seen as two very different tasks; rule extraction *for* neural networks and rule extraction *using* neural networks. While the first task is solely aimed at understanding the inner workings of a trained neural network, the second task is explicitly aimed at extracting a comprehensible model with higher accuracy than a comprehensible model created directly from the data set. The motivation for that rule extraction using neural networks may work is that a highly accurate opaque model often is a better representation of the underlying relationships than the set of training instances. One example is that training instances misclassified by the opaque model may very well be atypical, i.e., learning such instances could reduce the generalization capability.

But the opaque model is also a very accurate model of the function between input and output, so it could be used to label novel instances with unknown target values, as they become available. Naturally, the rule extraction algorithm could then use these newly labeled instances as learning examples. Despite this, no rule extraction algorithm that we are aware of use anything but training data (and possibly artificially generated instances) when extracting the transparent model. We have previously showed that the use of *oracle data*, i.e., production input vectors labeled by an opaque oracle model, could be beneficial for rule extraction [10]. In that study, we used our specialized rule extraction algorithm G-REX [11], but in this paper we extend the suggested methodology to general predictive classification. Specifically, only well-known and readily available classifiers are used in this study.

3 Method

As mentioned in the introduction, the purpose of this study was to evaluate whether the use of a high-accuracy opaque model (serving as a coaching oracle) may be beneficial for creating transparent predictive models. More specifically, decision trees and rule sets induced directly from training data only were compared to decision trees and rule sets built using different combinations of training data and oracle data. For simplicity, and to allow easy replication of the experiments, the Weka workbench [12] was used for all experiments[1]. In this study, two kinds of ensemble models were used as oracles, a large Random Forest [13] and a number of RBF neural networks, trained and combined using standard bagging [14]. For the actual classification, J48 and JRip were used since they represent what probably are the most famous tree inducer C4.5 [1] and rule inducer RIPPER [3], respectively. J48 obviously builds decision trees while JRip produces ordered rule sets. In the experimentation, all Weka settings were left at the default values for the different techniques.

For the evaluation, 4-fold cross-validation was used. The reason for not using the more standard value of ten folds was the fact that the use of only four folds results in what we believe to be a more representative proportion between training and production data. On each fold, the ensemble (the Random Forest or the bagged RBFs) was first trained, using training data only. This ensemble (the oracle) was then applied to the production instances, thereby producing production predictions; i.e., labeling the instances. This resulted in three different data sets:

– The *training* data: this is the original training data set, i.e., original input vectors with corresponding correct target values.
– The *ensemble* data: this is the original training instances but with ensemble predictions as target values instead of the always correct target values.
– The *oracle* data: this is the production instances with corresponding ensemble predictions as target values.

In the experimentation, all different combinations of these data sets were evaluated as training data for the techniques producing transparent models; i.e., J48 and JRip. In practice, this setup means that J48 and JRip will optimize different combinations of training accuracy, training fidelity and production fidelity. More specifically, we had the following seven different setups:

– Induction (I): Standard induction using original training data only. This maximizes training accuracy.
– Extraction (E): Standard extraction, i.e., using ensemble data only. Maximizes training fidelity.
– Explanation (X): Uses only oracle data, i.e., maximizes production fidelity.
– Indanation[2] (IX): Uses training data and oracle data, i.e., will maximize training accuracy and production fidelity.

[1] Our Weka OracleClassifier will be made available on our webpage.
[2] These describing names, combining the terms induction, extraction and explanation in different ways, are of course made-up.

- Exduction (IE): Uses training data and ensemble data. This means that if a specific training instance is misclassified by the ensemble, there will be two training instances for J48 and JRip, having identical inputs but different target values. So, here training accuracy and training fidelity are simultaneously maximized.
- Extanation (EX): Uses ensemble data and oracle data, i.e., will maximize fidelity towards the ensemble on both training and production data.
- Indextanation (IEX): Uses all three data sets, i.e., will try to maximize training accuracy, training fidelity and production fidelity simultaneously.

Table 1 below summarizes the different setups.

Table 1. Setups

	Data			Maximizes		
Setup	Train	Ensemble	Production	Train Acc	Train Fid.	Prod. Fid.
I	x			x		
E		x			x	
X			x			x
IE	x	x		x	x	
IX	x		x	x		x
EX		x	x		x	x
IEX	x	x	x	x	x	x

3.1 Experiments

This study contains two experiments. In the first experiment, J48 trees and JRip rule sets were built using an oracle ensemble consisting of 30 bagged RBF neural networks. In the second experiment, a Random Forest with 300 trees was used as the oracle. In the experiments, both accuracy and area under the ROC curve (AUC) were used for evaluation. While accuracy is based only on the final classification, AUC measures the ability to rank instances according to how likely they are to belong to a certain class; see e.g. [15]. AUC can be interpreted as the probability of ranking a true positive instance ahead of a false positive. For the evaluation, 10x4-fold cross-validation was used. The reported accuracies and AUCs were therefore averaged over the 4x10 folds. The 26 data sets used are all well-known and publicly available from the UCI Repository [16].

4 Results

Table 2 below shows the accuracy results for J48 in Experiment 1; i.e., when using bagged RBFs as the oracle. Comparing mean accuracies and mean ranks, it is very obvious that the use of production data paid off since IX, EX and IEX (combining production data with training or ensemble data) clearly outperformed the other setups. As a matter of fact, if these setups are explicitly

Table 2. Experiment 1 - Accuracy J48

	I	E	X	IE	IX	EX	IEX
Balance S	.783	.807	.862	.813	.868	.860	.858
Bcancer	.721	.718	.727	.723	.731	.725	.727
Bcancer W	.949	.953	.962	.951	.961	.963	.963
CMC	.508	.514	.515	.521	.528	.520	.526
Colic	.851	.827	.807	.847	.857	.843	.855
Credit-A	.854	.802	.795	.840	.847	.799	.820
Credit-G	.717	.721	.728	.731	.750	.739	.745
Diabetes	.743	.739	.757	.746	.759	.759	.754
Ecoli	.820	.820	.831	.819	.850	.846	.848
Glass	.676	.651	.682	.667	.715	.702	.706
Haberman	.707	.740	.743	.740	.731	.744	.737
Heart-C	.764	.774	.823	.765	.823	.816	.815
Heart-H	.787	.814	.828	.816	.823	.835	.836
Heart-S	.782	.791	.825	.783	.824	.837	.822
Hepatitis	.794	.813	.855	.801	.843	.848	.841
Iono	.894	.881	.911	.888	.924	.914	.918
Iris	.938	.941	.954	.944	.955	.955	.956
Labor	.766	.771	.858	.798	.852	.847	.855
Liver	.632	.621	.638	.631	.659	.646	.641
Lymph	.765	.777	.775	.764	.807	.799	.800
Sonar	.714	.691	.761	.694	.772	.771	.764
Tae	.548	.497	.503	.511	.525	.509	.516
Vehicle	.722	.654	.665	.691	.693	.666	.677
Vote	.963	.946	.946	.955	.960	.950	.956
Wine	.924	.923	.959	.925	.974	.973	.965
Zoo	.933	.933	.911	.942	.949	.949	.959
Mean	**.779**	**.774**	**.793**	**.781**	**.807**	**.800**	**.802**
Avg Rank	**5.37**	**5.92**	**3.94**	**5.00**	**2.02**	**2.98**	**2.77**

compared to standard J48 tree induction (the I setup), the results show that IX and IEX win 22 and lose 4 against standard J48, while EX wins 21 and loses 5.

To determine if there are any statistically significant differences, we use the statistical tests recommended by Demšar [17] for comparing several classifiers over a number of data sets, i.e., the Friedman test [18], followed by the Nemenyi post-hoc test [19]. With seven classifiers and 26 data sets, the critical distance (for $\alpha = 0.05$) is 1.77, so based on these tests, all three approaches utilizing the production data together with training or ensemble data obtained significantly higher accuracies than the three approaches using only training or ensemble data. Furthermore, it is interesting to observe that augmenting the production data with training data only is the best setup overall, clearly outperforming even the other setups also using production data.

Table 3 below shows the AUC results for J48 utilizing bagged RBFs as the oracle.

Table 3. Experiment 1 - AUC J48

	I	E	X	IE	IX	EX	IEX
Balance S	.86	.88	.93	.88	.93	.94	.94
Bcancer	.60	.61	.62	.61	.62	.61	.62
Bcancer W	.96	.95	.97	.96	.97	.96	.97
CMC	.69	.66	.66	.71	.71	.66	.71
Colic	.83	.81	.80	.84	.85	.83	.84
Credit-A	.87	.84	.81	.91	.88	.84	.90
Credit-G	.67	.69	.65	.71	.70	.71	.72
Diabetes	.74	.69	.71	.76	.76	.71	.76
Ecoli	.96	.96	.96	.96	.97	.97	.97
Glass	.81	.76	.80	.81	.81	.78	.84
Haberman	.54	.57	.57	.57	.56	.57	.57
Heart-C	.77	.77	.83	.78	.83	.83	.84
Heart-H	.76	.78	.81	.79	.81	.82	.82
Heart-S	.78	.80	.83	.79	.82	.84	.85
Hepatitis	.66	.71	.76	.69	.76	.77	.76
Iono	.88	.87	.91	.88	.92	.92	.92
Iris	.99	.99	1.00	.99	1.00	1.00	1.00
Labor	.73	.73	.85	.77	.85	.84	.83
Liver	.62	.59	.62	.62	.64	.62	.63
Lymph	.79	.78	.81	.80	.83	.80	.82
Sonar	.72	.69	.77	.71	.77	.77	.77
Tae	.72	.69	.69	.71	.73	.69	.71
Vehicle	.77	.66	.68	.79	.73	.68	.79
Vote	.98	.97	.96	.97	.97	.97	.97
Wine	.96	.95	.98	.96	.99	.99	.98
Zoo	.99	.99	.98	.99	.99	.99	1.00
Mean	**.79**	**.78**	**.81**	**.81**	**.82**	**.81**	**.83**
Avg Rank	**5.13**	**5.90**	**4.35**	**4.19**	**2.65**	**3.62**	**2.15**

Although the mean AUCs over all data sets appear to indicate that all setups performed similarly, the mean ranks give a more detailed and completely different picture. Most importantly, the results clearly show that the use of production data is beneficial also when considering the ranking ability of the models. Looking for statistically significant differences, the differences in mean ranks show that IX and IEX both obtained significantly higher AUCs than standard J48 (I). A closer inspection shows that even if the differences are quite small, the successful approaches have slightly higher AUCs on almost all data sets compared to normal induction. As an example, a pairwise comparison between IX and I shows 23 outright wins for IX and only 2 for I.

Table 4 below shows a summary of accuracy and AUC results for JRip using bagged RBFs as the oracle. Looking at the average ranks on accuracy, IX and IEX are clearly superior to all the other setups. A further analysis shows that IX and IEX obtained significantly higher accuracies than the three setups not utilizing the production data, also when using JRip as classifier. In addition, the two setups combining production data with training data were actually significantly more accurate than setup X, i.e., utilizing only production data. Finally, the difference between EX and I is also statistically significant. The AUC results when using JRip for the classification are quite similar to the J48 results. Here, however, all three setups combining the production data with either training or ensemble data are significantly better than standard JRip (I). In addition, IX, EX and IEX also obtained significantly higher AUCs than both E and X.

Table 4. Experiment 1 - JRip Transparent Model

		I	E	X	IE	IX	EX	IEX
Accuracy	**Mean**	.775	.772	.780	.785	.800	.796	.802
	Avg Rank	**5.12**	**6.15**	**4.63**	**4.35**	**2.35**	**3.02**	**2.38**
AUC	**Mean**	.79	.78	.79	.80	.81	.81	.82
	Avg Rank	**4.85**	**5.73**	**5.00**	**4.27**	**2.62**	**3.08**	**2.46**

Summarizing Experiment 1, by looking at Table 5 below, where a + indicates a statistically significant difference based on the Nemenyi post-hoc test, the most important result is that the setups combining training data with production data (IX and IEX) obtained significantly higher accuracy and AUC compared to standard model induction (I), both when using J48 and JRip as the actual classifier. In addition, IE and IEX also had significantly better results on a large majority of the data sets compared to X, clearly indicating that building the model using only production data is an inferior choice.

Table 5. Experiment 1 - Significant differences

	I				X			
	J48		Jrip		J48		Jrip	
	Acc	Auc	Acc	Auc	Acc	Auc	Acc	Auc
IX	+	+	+	+	+		+	+
EX	+		+					+
IEX	+	+	+	+		+	+	+

Table 6 below shows a summary of the results from Experiment 2; i.e., when using a 300 tree Random Forest as the oracle. Looking at the J48 mean accuracies and mean ranks, there is a clear ordering showing that IX and EX (using training or ensemble data together with the production data) outperformed X and IEX (using only production data or all three data sets) which in turn outperformed the approaches not utilizing the production data at all, i.e., I, E and IE. Comparing I with E and IX with EX, the obtained accuracies are identical on most data sets. The explanation is that the large Random Forrest managed to obtain perfect training accuracy quite often, thus making the ensemble data identical to the training data. The average ranks for JRip accuracy show that IX and EX are clearly superior to all the other setups. As a matter of fact, the results show that IX and EX obtained significantly higher accuracies than all setups not utilizing the production data. In addition, the two setups utilizing the production data together with either training or ensemble data were significantly more accurate than setup X, i.e., utilizing only production data.

Looking at the AUC results for J48 using the Random Forest as oracle, the picture is familiar. Here, the two best setups are IX and EX, both obtaining significantly higher AUCs than all setups not using the production data, including, of course, normal use of J48 (I). The AUC results when using JRip as classifier, finally, are quite similar to the J48 results. The only difference is that IEX here outperforms both IX and EX. Again, all three setups combining the production data, with either training or ensemble data, are significantly better than setups not using the production data.

Table 6. Experiment 2 - Summary

			I	E	X	IE	IX	EX	IEX
Accuracy	**J48**	**Mean**	.779	.779	.802	.773	.813	.812	.805
		Avg Rank	**5.42**	**5.50**	**3.38**	**6.17**	**1.85**	**2.12**	**3.56**
	JRip	**Mean**	.775	.774	.787	.778	.804	.804	.807
		Avg Rank	**5.35**	**5.50**	**4.15**	**5.81**	**2.27**	**2.21**	**2.71**
AUC	**J48**	**Mean**	.79	.79	.81	.77	.82	.82	.81
		Avg Rank	**4.83**	**4.87**	**4.10**	**5.98**	**2.35**	**2.33**	**3.56**
	JRIP	**Mean**	.79	.79	.79	.78	.81	.81	**.82**
		Avg Rank	**4.90**	**4.79**	**5.08**	**5.65**	**2.73**	**2.73**	**2.12**

Table 7 below shows the statistically significant differences in Experiment 2. Again, setups using production data, together with training or ensemble data, performed significantly better than standard use of J48 or JRip. A small difference, compared to Experiment 1, is that when using J48 as classifier, building the tree using production data only actually performed relatively well.

Table 7. Experiment 2 - Significant differences

	I				X			
	J48		Jrip		J48		Jrip	
	Acc	Auc	Acc	Auc	Acc	Auc	Acc	Auc
IX	+	+	+	+			+	+
EX	+	+	+	+		+	+	+
IEX	+		+	+				+

The overall result from the two experiments is thus that the setups utilizing production data predictions obtained from the oracle almost always yield significantly better transparent models than normal induction. Furthermore, when using data from the oracle, the best results are obtained when combining production and training data.

5 Conclusions

We have in this paper suggested and evaluated a novel method for increasing predictive performance for transparent models. This method is suitable in the very common situation where the predictive model is built for a specific prediction task, and the input vectors for the actual production data are available already when inducing the model. The procedure consists of building a powerful opaque model, called an oracle, using the available labeled training data and then applying this model on the production instances to obtain predicted values which are used as labels. This oracle data is then combined in different ways with the training data to form a new training set for a classifier producing transparent models. The method is very general in that any opaque model can serve as the oracle and any technique producing transparent models can benefit from using oracle data during training.

The results show that using an oracle coach almost always yields significantly better predictive performance for the transparent model, compared to standard induction using only training data. This result is robust in that it holds for both transparent model representations used, i.e., J48 decision trees and JRip ordered rule sets, for performance measured either as accuracy or AUC and also for both types of opaque oracle models used. Regarding how to use the oracle labeled data, the experiments show that oracle data for production instances should preferably be combined with just the original training instances to obtain the best performance, but that other combinations also utilizing the oracle data were strong alternatives.

The overall conclusion is that the suggested method virtually guarantees increased predictive performance, compared to normal training, at a low computational cost. Oracle coaching is thus recommended if production input vectors are available when building the model, and the situation requires a transparent model.

References

1. Quinlan, J.R.: C4.5: programs for machine learning. Morgan Kaufmann Publishers Inc., San Francisco (1993)
2. Breiman, L., Friedman, J., Stone, C.J., Olshen, R.A.: Classification and Regression Trees. Chapman & Hall/CRC (1984)
3. Cohen, W.W.: Fast effective rule induction. In: Proceedings of the 12th International Conference on Machine Learning, pp. 115–123. Morgan Kaufmann, San Francisco (1995)
4. Zhu, X.: Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison (2005)
5. Joachims, T.: Transductive inference for text classification using support vector machines, pp. 200–209. Morgan Kaufmann, San Francisco (1999)

6. Andrews, R., Diederich, J., Tickle, A.B.: Survey and critique of techniques for extracting rules from trained artificial neural networks. Knowl.-Based Syst. 8(6), 373–389 (1995)
7. Craven, M.W., Shavlik, J.W.: Extracting tree-structured representations of trained networks. In: Advances in Neural Information Processing Systems, pp. 24–30. MIT Press, Cambridge (1996)
8. Thrun, S., Tesauro, G., Touretzky, D., Leen, T.: Extracting rules from artificial neural networks with distributed representations. In: Advances in Neural Information Processing Systems, vol. 7, pp. 505–512. MIT Press, Cambridge (1995)
9. Zhou, Z.H.: Rule extraction: using neural networks or for neural networks? J. Comput. Sci. Technol. 19(2), 249–253 (2004)
10. Johansson, U., Niklasson, L.: Evolving decision trees using oracle guides. In: CIDM, pp. 238–244. IEEE, Los Alamitos (2009)
11. Johansson, U., König, R., Niklasson, L.: Rule extraction from trained neural networks using genetic programming. In: ICANN, supplementary proceedings, pp. 13–16 (2003)
12. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005)
13. Breiman, L.: Random forests. Machine Learning 45(1), 5–32 (2001)
14. Breiman, L.: Bagging predictors. Machine Learning 24(2), 123–140 (1996)
15. Fawcett, T.: Using rule sets to maximize roc performance. In: IEEE International Conference on Data Mining, ICDM 2001, pp. 131–138. IEEE Computer Society, Los Alamitos (2001)
16. Asuncion, A., Newman, D.J.: UCI machine learning repository (2007)
17. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. 7, 1–30 (2006)
18. Friedman, M.: The use of ranks to avoid the assumption of normality implicit in the analysis of variance. Journal of American Statistical Association 32, 675–701 (1937)
19. Nemenyi, P.B.: Distribution-free multiple comparisons. PhD-thesis. Princeton University (1963)