Quantifying the Impact of Learning Algorithm Parameter Tuning

Niklas Lavesson and **Paul Davidsson**
Dept. of Software and Systems Engineering
Blekinge Institute of Technology
Box 520, SE-372 25, Ronneby, Sweden

Abstract

The impact of learning algorithm optimization by means of parameter tuning is studied. To do this, two quality attributes, sensitivity and classification performance, are investigated, and two metrics for quantifying each of these attributes are suggested. Using these metrics, a systematic comparison has been performed between four induction algorithms on eight data sets. The results indicate that parameter tuning is often more important than the choice of algorithm and there does not seem to be a trade-off between the two quality attributes. Moreover, the study provides quantitative support to the assertion that some algorithms are more robust than others with respect to parameter configuration. Finally, it is briefly described how the quality attributes and their metrics could be used for algorithm selection in a systematic way.

Introduction

It has been argued that many of the well-known learning algorithms often generate classifiers of comparable quality (Mitchell 1997; Weideman *et al.* 1995; Witten & Frank 2000). However, the relation between algorithm parameter tuning and classifier performance is seldom discussed.

In order to solve a learning task there is typically only a limited amount of resources available and different situations put emphasis on different requirements. For instance, when the time available for preparation is short, a robust learning algorithm may be useful, capable of generating a reasonably good classifier with a minimum amount of parameter tuning. For other problems it could be more important to select an algorithm capable of generating a classifier with the best classification accuracy possible (using careful parameter tuning), with the possible drawback of getting worse results if less care is taken when setting the parameters.

We present a systematic comparison between four algorithms. One rationale behind this study is to investigate the impact of the configuration of algorithms on classifier performance. Instead of just finding the algorithm that generates the most accurate classifier on some data, it may be important to look at the variation of performance, or sensitivity of a learning algorithm (sensitivity is here seen as the inverse of robustness). Obviously this is a complex issue and it has been argued that, theoretically, no algorithm is superior on all possible induction problems (Schaffer 1994; Wolpert 2001). In fact, it is argued that any particular configuration of an algorithm is as good as any other taken over all possible problems. In order to investigate this issue, quantifiers of performance as well as more systematic ways to compare algorithms and algorithm configurations are needed. This paper presents two quality attributes for algorithm assessment; sensitivity and classification performance. As this study will show, these two attributes reflect different aspects of quality related to learning problems and there exist several possible candidate metrics for assessing the quality attributes.

In the next section, we describe the quality attributes and the metrics used to assess them. Then the experiment procedure and results are presented and this is followed by a discussion about quality attributes and how they may be used for algorithm selection in a systematic way. The last sections feature related work as well as conclusions and pointers to future work.

Quality Attribute Metrics

We now introduce sensitivity and classification performance as learning algorithm quality attributes and present formal definitions of two metrics for each attribute. The metrics for classification performance capture the average and best performance and the metrics for sensitivity capture the average impact, and the degree of impact, of parameter tuning.

Given a learning algorithm a, let C_a be the set of all possible parameter configurations and $c_a \in C_a$ a particular configuration. In addition, D is a set of known instances and f an evaluation function $f(c_a, D)$. Since C_a is usually very large it is not feasible to evaluate all elements in practise. Instead a subset, $Q_a \subset C_a$, is used as an estimate. One should be careful when choosing Q_a since this set has a large impact on the assessment of the quality attributes.

Classification performance is probably the most used quality attribute in current practice. It is typically measured in terms of the classification accuracy of a classifier generated by a learning algorithm. The average performance metric and its estimate are defined as:

$$p_1(C_a, D) = \frac{\sum_{x \in C_a} f(x, D)}{|C_a|} \quad (1)$$

$$\hat{p}_1 = p_1(Q_a, D) = \frac{\sum_{x \in Q_a} f(x, D)}{|Q_a|} \quad . \tag{2}$$

The second metric, best performance and its estimate are defined as:

$$p_2 = max_{x \in C_a} f(x, D) \tag{3}$$

$$\hat{p}_2 = max_{x \in Q_a} f(x, D) \quad . \tag{4}$$

We define algorithm sensitivity as being inverse proportionate to robustness, i.e., the impact of tuning a sensitive algorithm is high, while the opposite holds for a robust algorithm. Two metrics are defined for evaluating the sensitivity attribute. s_1 uses the standard deviation of the evaluation scores to capture the average impact of tuning. The metric s_1 and its estimate $\hat{s}_1$ are defined as follows:

$$s_1(C_a, D) = \sqrt{\frac{\sum_{x \in C_a} \left(f(x, D) - p_1(C_a, D)\right)^2}{|C_a|}} \tag{5}$$

$$\hat{s}_1 = s_1(Q_a, D) = \sqrt{\frac{\sum_{x \in Q_a} \left(f(x, D) - \hat{p}_1(Q_a, D)\right)^2}{|Q_a| - 1}} \quad . \tag{6}$$

The metric s_2 uses the statistical range, i.e., the difference between the highest and lowest scores, to capture the degree of impact. s_2 and its estimate are defined as:

$$s_2 = p_2 - min_{x \in C_a} f(x, D) \tag{7}$$

$$\hat{s}_2 = \hat{p}_2 - min_{x \in Q_a} f(x, D) \quad . \tag{8}$$

Experiment Design

The aim is to investigate the impact of algorithm parameter tuning by assessing learning algorithm quality attributes and to find out whether there is a trade-off between these attributes. The experiment is run using the WEKA machine learning workbench (Witten & Frank 2000).

Featured Algorithms

The four algorithms; Back-Propagation (BP) (Rumelhart, Hinton, & Williams 1986), K-Nearest Neighbor (KNN) (Aha, Kibler, & Albert 1991), Support Vector Machines (SVM) (Cortes & Vapnik 1995), and C4.5 (Quinlan 1993) were selected for this study. The motivation is that these four algorithms are all widely used and, in addition they belong to four different families of learning algorithms; neural network learners, instance-based learners, kernel machines and decision tree learners. Many studies on supervised learning include one or more of these algorithms and they are all discussed extensively in the literature, cf. (Mitchell 1997; Russell & Norvig 2003).

The number of parameters and the extent to which it is possible to influence the performance of the resulting classifier vary between different algorithms. Some allow very extensive tuning to adjust to a specific problem while others are completely unconfigurable. The parameter intervals were chosen by selecting limits close to, and symmetrically around, the default values of WEKA. In case WEKA used the lowest possible setting as the default value this setting was chosen as a lower bound. One of the most complex algorithms in this respect is BP (MultilayerPerceptron in WEKA) and many researchers have suggested configuration guidelines for different problems (Bebis & Georgiopoulos 1995).

A subset of the BP algorithm parameters and the network structure were selected for this particular study; the training time, hidden layers and neurons in each hidden layer. The justification for choosing these parameters is that they all are relative to the data set on which the algorithm operates; hence they should be optimized for a particular problem. Other parameters like momentum and learning rate are often set to very similar values independent of the data set. Common values for momentum and learning rate are 0.2 and 0.3 respectively (Mitchell 1997; Witten & Frank 2000). The WEKA default value for training time is 500 epochs and the number of hidden neurons is defined by n:

$$n = (\alpha + \beta)/2 \quad , \tag{9}$$

where α is the number of classes and β is the number of attributes for a particular data set. Table 1 describes the configurations of BP used in the experiment. The time and layers parameters both have static intervals but the neurons parameter interval is dynamic and varies with the number of classes and attributes of different data sets. As a constraint the lower bound of the hidden neurons interval is 1. This means that even if n is lower than 7 the lower bound will not be lower than 1, e.g., if $\alpha = 2$ and $\beta = 3$ then $n = 2$ which would make the lower limit $n - 6 = 2 - 6 = -4$ if it was not restricted.

Table 1: BP parameter configurations.

Parameter	Setting(s)
Training time	[500...27500] step size: 3000
Hidden neurons	[n-6...n+6] step size: 2
Hidden layers	[1...5] step size: 1
Sample size	300±50

Table 2: KNN parameter configurations.

Parameter	Setting(s)
Neighbors	[1...40] step size: 1
Weighting	{equal, inverse-distance, similarity}
Sample size	120

Regarding the KNN algorithm (IBk in WEKA), the number of neighbors, k, to use for a particular problem can be difficult to know beforehand and a number of methods for finding an optimal value have been proposed. The WEKA default value of k is 1. The KNN algorithm can also be configured to weight instances differently and two weighting techniques included in the WEKA implementation are inverse-distance weighting ($w = 1/\text{distance}$) and similarity weighting ($w = 1 - \text{distance}$). Equal weighting ($w =$distance) is the default technique. The KNN configurations are shown in Table 2.

Table 3: C4.5 parameter configurations.

Parameter	Setting(s)
Confidence threshold	[0.02...0.5] step size: 0.02
Minimum instances per leaf	[1...4] step size: 1
Folds for pruning	[2...5] step size: 1
Pruning	{no, yes}
Reduced-error pruning	{no, yes}
Subtree raising	{no, yes}
Sample size	808

The C4.5 algorithm (J48 in WEKA) induces pruned or unpruned decision tree classifiers. Configuration possibilities include the type of pruning, the confidence threshold for pruning, the number of folds used for reduced error pruning and the minimum number of instances per leaf. Subtree raising is a post-pruning technique that raises the subtree of the most popular branch. The most popular branch is defined as the branch that has the highest number of training instances. Reduced-error pruning is performed by splitting the data set into a training and validation set. The smallest version of the most accurate subtree is then constructed by greedily removing the node that less improves the validation set accuracy. The J48 default configuration can be summarized as follows. Pruning is on and the technique used is subtree raising. The confidence threshold for pruning is 0.25, the minimum instances per leaf parameter is set to 2 and the number of folds for reduced error pruning value is set to 3. The configuration possibilities of J48 are further detailed in Table 3.

Table 4: SVM parameter configurations.

Parameter	Setting(s)
Complexity	[0.1...1.0] step size: 0.1
Kernel function	{polynomial, radial basis}
Gamma	[0.005...0.060] step size: 0.005
Lower order terms	{true, false}
Exponent	[0.5...2.5] step size: 0.5
Sample size	200

For Support Vector Machines, SMO in WEKA supports two different kernels: polynomial and radial basis function. These kernels have different sets of parameters that can be configured for a specific problem. Configurations have been chosen so that each parameter interval lies in proximity of the WEKA defaults. One parameter is the kernel, which can be set to either polynomial or radial basis function. The gamma parameter is only used by the radial basis function whereas the exponent and lower order terms parameters are only used by the polynomial kernel. The complexity constant determines the trade-off between the flatness and the amount by which misclassified samples are tolerated (Howley & Madden 2004). The SVM parameters are shown in Table 4.

Procedure

Eight data sets from UCI machine learning repository were used in the experiments (Newman *et al.* 1998). The number of instances (I), classes and attributes (A) of each data set are shown in Table 5. The last column of this table, $I * A$, could be regarded as a measure of problem size. For each

Table 5: Data set overview.

Data set	Instances (I)	Classes	Attributes (A)	$I * A$
Breast-cancer	286	2	9	2574
Contact-lenses	24	3	4	96
Iris	150	3	4	600
Labor	57	2	16	912
Lymph	148	4	19	2812
Soybean	683	19	35	23905
Weather	14	2	4	56
Zoo	101	7	18	1818

of the four algorithms, a large number of different parameter configurations were evaluated using 10-fold cross-validation (10CV) on each of the eight data sets. 10CV is performed by dividing the data set into 10 folds or partitions. Classifiers are then generated by training on nine folds and validated on the tenth until all folds have been used for validation. The 10CV score is defined as the mean of all validation test scores. We use 10CV as evaluation function for the metrics and the justification is that it is perhaps the most common classifier evaluation method and extensive tests on different data sets with different techniques have shown that ten is about the right number of folds to get the best estimate of accuracy. Other versions of cross-validation exist, e.g. leave-one-out, but the drawbacks of this procedure include a high computational cost and it also implies a non-stratified sample (Witten & Frank 2000) because the folds will not have the same class distribution as the complete data set. For instance, if a data set has 100 instances and the class distribution is 50 class A and 50 class B instances, this distribution (50% A and 50% B) will be very different from a leave-one-out testing fold which will be either 100% class A or 100% class B.

Results

The classification performance and sensitivity of each algorithm for each data set, measured according to the four metrics, are presented in Table 6.

In addition, Figure 1 contains eight box plots (one for each data set) showing the distribution of 10CV values for the four algorithms. Each box plot indicates the highest and lowest 10CV value as well as the median and the upper and lower quartiles.

We see that both the highest mean, $\hat{p}_1$, (Table 6) and the highest median (Figure 1) are lower than the lowest max value, $\hat{p}_2$, for all eight data sets. Assuming that the mean and median values are approximations of the performance of the default configuration of an algorithm (that is, given that little effort is spent on parameter tuning the mean/median would be the expected value), then this would suggest that

Table 6: Experiment results.

Algorithm	Data set	Performance		Sensitivity	
		$\hat{p}_1$	$\hat{p}_2$	$\hat{s}_1$	$\hat{s}_2$
BP	Breast-cancer	0.69	0.74	0.02	0.14
	Contact-Lenses	0.67	0.83	0.05	0.38
	Iris	0.79	0.98	0.27	0.65
	Labor	0.83	0.95	0.10	0.30
	Lymph	0.75	0.87	0.11	0.36
	Soybean	0.81	0.95	0.16	0.81
	Weather	0.64	0.86	0.07	0.43
	Zoo	0.76	0.97	0.21	0.57
C4.5	Breast-cancer	0.70	0.75	0.02	0.10
	Contact-lenses	0.76	0.96	0.09	0.46
	Iris	0.94	0.96	0.01	0.07
	Labor	0.78	0.90	0.04	0.23
	Lymph	0.75	0.83	0.03	0.19
	Soybean	0.88	0.93	0.01	0.10
	Weather	0.61	0.86	0.05	0.43
	Zoo	0.89	0.96	0.02	0.14
KNN	Breast-cancer	0.71	0.75	0.01	0.07
	Contact-lenses	0.72	0.83	0.09	0.25
	Iris	0.95	0.97	0.01	0.07
	Labor	0.85	0.95	0.05	0.21
	Lymph	0.82	0.87	0.02	0.08
	Soybean	0.83	0.92	0.06	0.25
	Weather	0.67	0.79	0.09	0.36
	Zoo	0.92	0.96	0.02	0.08
SVM	Breast-cancer	0.71	0.75	0.02	0.12
	Contact-lenses	0.64	0.79	0.04	0.25
	Iris	0.92	0.97	0.03	0.17
	Labor	0.80	0.97	0.12	0.32
	Lymph	0.78	0.89	0.10	0,34
	Soybean	0.86	0.94	0.14	0.78
	Weather	0.63	0.86	0.06	0.36
	Zoo	0.78	0.97	0.19	0.56

it is more important to tune the parameters of an arbitrary algorithm than choosing a particular algorithm. At least this holds for the classification tasks and algorithms studied in this experiment.

We have investigated the relation between the two quality attributes, e.g., whether there is a trade-off between classification performance and robustness, i.e., if sensitive algorithms have high performance. If this trade-off was present there should be a strong positive correlation between the classification performance and sensitivity metrics. However this is not the case. With a significance level of 0.05 (95% confidence) and a sample size of 8 the correlation coefficient must be greater than 0.707 to establish a nonzero correlation. Using this level of significance we could establish a correlation between the two classification performance metrics (0.73) and the two sensitivity metrics (0.84) but no correlation could be established between a classification performance metric and a sensitivity metric.

When comparing the highest sensitivity and classification performance scores between the algorithms across all data sets, it can be observed that KNN has the highest $\hat{p}_1$ on a majority of the data sets while SVM only has the highest $\hat{p}_1$ on one data set. The opposite holds for $\hat{p}_2$. BP has the highest $\hat{s}_1$ and $\hat{s}_2$ on a majority of the data sets, followed by C4.5 and SVM.

In six out of eight data sets, the range ($\hat{s}_2$) is greater for BP and SVM than for KNN and C4.5. This would suggest that BP and SVM are more sensitive to the choice of parameter setting, which in turn indicates that BP and SVM are harder to tune. For at least four out of eight data sets, the best scoring algorithm is also the worst scoring. This strengthen the argument made that there is little, or no correlation between sensitivity and performance.

Both $\hat{s}_1$ and $\hat{s}_2$ are very high for BP on the Iris and Soybean data sets. However, $\hat{s}_1$ is higher than $\hat{s}_2$ for Iris, and the opposite holds for Soybean. For Iris, it can be observed that the lowest value is a part of general low performance while the lowest value for Soybean seems to be an outlier. This suggests that both sensitivity metrics are important, since they can be used to measure different aspects of sensitivity.

Discussion

The metrics $\hat{p}_1$ and $\hat{p}_2$ tell little about to which extent the performance can be influenced by tuning the learning algorithm. In order to capture both the performance and to what extent an algorithm is sensitive to parameter change, two different quality attributes have to be assessed. A sensitive algorithm may be a better choice for experts or researchers searching for optimal performance, while a robust algorithm may provide less experienced users with a method that is easy to use and still produce good results.

It is hard to formulate a strict measure of the impact that algorithm parameters have on classifier performance. On the one hand, it can be a dependency problem. If one parameter is dependent on the value of another and changing the first parameter impacts classifier performance, it could be an indirect impact caused by the second parameter. On the other hand, there is a degree of randomness that affects the evaluation. For example, the weights of a neural network are initialized to random values before training. This could influence the performance of the learned classifier differently between runs. There is also a small element of randomness involved when using 10CV. Estimation bias and variance also need to be considered if the training set or the test set is small. However, there are certain measures that may indicate the degree of impact. $\hat{s}_2$ reveals the upper and lower limits of the impact of tuning. However, it does not explain which parameter had the largest impact. $\hat{s}_1$ reveals information about how results are scattered between the upper and lower limits. These two metrics capture different aspects of sensitivity, i.e., how much the output (performance) is related to the input (configuration) of an algorithm.

As shown in this study, the quality attributes and their metrics can be used to better understand the behavior of learning algorithms. In addition, they can be used for selecting which algorithm to use in a particular situation. One approach would be to apply the Analytic Hierarchy Process (AHP) (Saaty & Vargas 2001), which is a multi-criteria decision support method stemming from Management Science. One of the cornerstones in AHP is to evaluate a set of alternatives based on a particular blend of criteria, i.e., con-

sidering a specific trade-off situation. The first step in AHP is to set up a hierarchy of the criteria that are being evaluated. This means that one criterion can be broken down into several sub-criteria, and the evaluation of the different alternatives is done by weighing all levels of this decision support hierarchy. In our case, the criteria would be based on quality attributes (and metrics), and their importance for the application at hand.

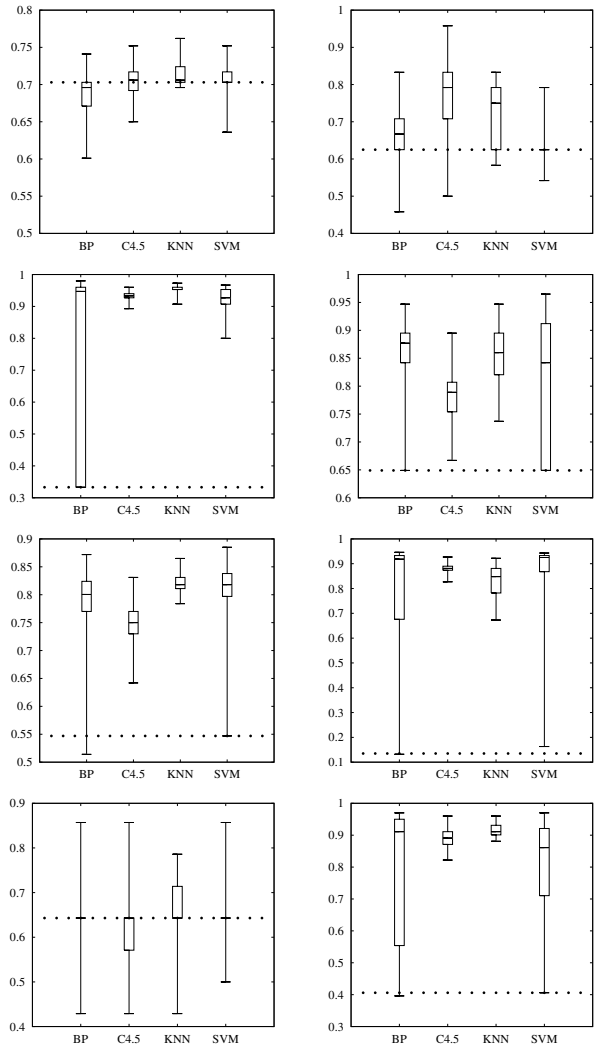

Figure 1: Box plots of 10CV scores on eight data sets: breast-cancer, contact-lenses (1st row), iris, labor (2nd row), lymph, soybean (3rd row), weather, zoo (4th row). The dotted line represents the performance of a majority class model.

Related Work

Meta-learning research focuses on automatic ways to induce meta-knowledge from experience. The objective is learning how to learn by using algorithms to analyze and interpret prior experiments and data set characteristics in order to improve the quality of learning (Giraud-Carrier & Keller 2002). Many researchers have recognized the fact that algorithm optimization can be very time consuming. Often there is a need for domain-knowledge and expertise in order to generate good classifiers. Meta-learning studies have also brought forth different methods for enhancing the performance of existing techniques, e.g., automatic tuning of SVM kernel parameters (Chapelle *et al.* 2002) and cascade-correlation for automatically creating and training neural networks (Fahlman & Lebiere 1990). Other enhancing methods, such as bagging, boosting and stacking, combine several classifiers to increase performance, cf. (Breiman 1996; Freund & Schapire 1996; Witten & Frank 2000; Wolpert 1992). Whereas meta-learning is concerned with learning how to learn, this study addresses issues such as how to quantify the potential and need of meta-learning for different learning algorithms.

Several methods exist for both classifier and learning algorithm evaluation. For instance, classifiers can be evaluated using ROC curves (Provost, Fawcett, & Kohavi 1998), measure functions (Andersson, Davidsson, & Lindén 1999) or holdout (Duda, Hart, & Stork 2000) while algorithms are commonly evaluated using cross-validation (Stone 1974) or bootstrap (Efron & Tibshirani 1993). Large-scale empirical studies have been conducted to examine the behavior of such evaluation methods and the metrics they use, cf. (Caruana & Niculescu-Mizil 2004). Whereas this study investigates different performance metrics to measure the quality of classifiers (the output from the learning algorithm using a particular parameter setting), we define metrics which are used to measure the quality of the algorithm itself.

Some work has focused on the sensitivity of algorithms, e.g., sensitivity analysis which aims at determining how variation of the input influences the output of a system (Bousquet & Elisseeff 2002). Instead of regarding data sets as the only input for learning, we consider algorithm parameter configuration to be an important part of the input as well.

Conclusions and Future Work

We have studied the impact of learning algorithm optimization by means of parameter tuning and argue that some algorithms are more robust to parameter change than others. Two quality attributes that facilitate the impact analysis were defined. One attribute, sensitivity, captures the extent to which it is possible to affect performance by tuning parameters and the other captures the classification performance of an algorithm. We then presented two candidate metrics for each quality attribute and showed that these metrics complete each other by revealing different aspects of sensitivity and classification performance. The metrics depend on the evaluation of all possible configurations of an algorithm. In most practical cases this is impossible and we therefore suggest an estimation that can be performed by selecting a subset of the configurations symmetrically around a known default configuration, e.g., determined by meta-learning techniques. We have evaluated each configuration using cross-validation; however the metrics as such are not restricted

to this particular method. Our results indicate that parameter tuning is often more important than the choice of algorithm and we provide quantitative support to the assertion that some algorithms are more robust than others. Moreover, the results suggest that no trade-off exists between sensitivity and classification performance.

In future work, we intend to refine the method for configuration subset selection to increase the accuracy of the metrics estimations and evaluate the sensitivity and classification performance using a wider selection of algorithms and data sets. Other metrics are also considered, e.g., one plausible metric for sensitivity could be the average performance difference between adjacent configurations. An assumption related to this metric is that smooth distributions would benefit optimization of the particular algorithm. We also intend to investigate the use of AHP and other approaches that could be used for systematic selection of learning algorithms.

References

Aha, D. W.; Kibler, D.; and Albert, M. K. 1991. Instance-based Learning Algorithms. *Machine Learning* 6:37–66.

Andersson, A.; Davidsson, P.; and Lindén, J. 1999. Measure-based Classifier Performance Evaluation. *Pattern Recognition Letters* 20(11-13):1165–1173.

Bebis, G., and Georgiopoulos, M. 1995. Improving Generalization by Using Genetic Algorithms to Determine the Neural Network Size. In *Southcon 95*, 392–397.

Bousquet, O., and Elisseeff, A. 2002. Stability and Generalization. *Machine Learning Research* 2:499–526.

Breiman, L. 1996. Bagging Predictors. *Machine Learning* 24(2):123–140.

Caruana, R., and Niculescu-Mizil, A. 2004. Data Mining in Metric Space: An Empirical Analysis of Supervised Learning Performance Criteria. In *10th International Conference on Knowledge Discovery and Data Mining*, 69–78.

Chapelle, O.; Vapnik, V.; Bousquet, O.; and Mukherjee, S. 2002. Choosing Multiple Parameters for Support Vector Machines. *Machine Learning* 46(1):131–159.

Cortes, C., and Vapnik, V. 1995. Support Vector Networks. *Machine Learning* 20(3):273–297.

Duda, R. O.; Hart, P. E.; and Stork, D. G. 2000. *Pattern Classification*. USA: John Wiley & Sons, 2nd edition.

Efron, B., and Tibshirani, R. J. 1993. *An Introduction to the Bootstrap*. Monographs on Statistics & Applied Probability. Chapman & Hall.

Fahlman, S. E., and Lebiere, C. 1990. The Cascade-Correlation Learning Architecture. *Advances in Neural Information Processing Systems* 2:524–532.

Freund, Y., and Schapire, R. E. 1996. Experiments with a New Boosting Algorithm. In *13th International Conference on Machine Learning*, 148–156.

Giraud-Carrier, C., and Keller, J. 2002. *Dealing With the Data Flood: Mining Data, Text and Multimedia*. The Hague, Netherlands: STT/Beweton. chapter Meta Learning.

Howley, T., and Madden, M. G. 2004. The Genetic Evolution of Kernels for Support Vector Machine Classifiers. In *15th Irish Conference on Artificial Intelligence*.

Mitchell, T. M. 1997. *Machine Learning*. Computer Science Series. Singapore: McGraw-Hill, international edition.

Newman, D. J.; Hettich, S.; Blake, C. L.; and Merz, C. J. 1998. UCI Repository of Machine Learning Databases.

Provost, F.; Fawcett, T.; and Kohavi, R. 1998. The Case Against Accuracy Estimation for Comparing Induction Algorithms. In *15th International Conference on Machine Learning*, 445–453. Madison, WI, USA: Morgan Kaufmann.

Quinlan, J. R. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

Rumelhart, D. E.; Hinton, G. E.; and Williams, R. J. 1986. Learning Internal Representations by Error Propagation. *Parallel Distributed Processing* 1:318–362.

Russell, S., and Norvig, P. 2003. *Artificial Intelligence: A Modern Approach*. Prentice Hall Series in Artificial Intelligence. Prentice Hall, 2nd edition.

Saaty, T. L., and Vargas, L. G. 2001. *Models, Methods, Concepts & Applications of the Analytic Hierarchy Process*. Kluwer Academic Publisher.

Schaffer, C. 1994. A Conservation Law for Generalization Performance. In *11th International Conference on Machine Learning*, 259–265. Morgan Kaufmann.

Stone, M. 1974. Cross-validatory Choice and Assessement of Statistical Predictions. *Royal Statistical Society* 36:111–147.

Weideman, W.; Manry, M.; Hung-Chun, Y.; and Wei, G. 1995. Comparisons of a Neural Network and a Nearest-neighbor Classifier via the Numeric Handprint Recognition Problem. *IEEE Transactions on Neural Networks* 6(6):1524–1530.

Witten, I. H., and Frank, E. 2000. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann.

Wolpert, D. H. 1992. Stacked Generalization. *Neural Networks* 5:241–259.

Wolpert, D. H. 2001. The Supervised Learning No Free Lunch Theorems. Technical report, NASA Ames Research Center, Moffett Field, CA, USA.