

An Empirical Study of Hoeffding Racing for Model Selection in k -Nearest Neighbor Classification

Flora Yu-Hui Yeh and Marcus Gallagher

School of Information Technology and Electrical Engineering
University of Queensland, 4072, Australia
{flora,marcusg}@itee.uq.edu.au

Abstract. Racing algorithms have recently been proposed as a general-purpose method for performing model selection in machine learning algorithms. In this paper, we present an empirical study of the Hoeffding racing algorithm for selecting the k parameter in a simple k -nearest neighbor classifier. Fifteen widely-used classification datasets from UCI are used and experiments conducted across different confidence levels for racing. The results reveal a significant amount of sensitivity of the k -nn classifier to its model parameter value. The Hoeffding racing algorithm also varies widely in its performance, in terms of the computational savings gained over an exhaustive evaluation. While in some cases the savings gained are quite small, the racing algorithm proved to be highly robust to the possibility of erroneously eliminating the optimal models. All results were strongly dependent on the datasets used.

1 Introduction

Model selection is an important task in the application of many machine learning methods. Racing algorithms [1, 2] take a computational approach to the model selection problem. Given a set of candidate models (i.e. machine learning algorithms with fully specified parameter values), together with a set of training data for a given problem, an exhaustive search over the model set would require each model to be trained and evaluated on the entire training set. Racing works by initially implementing all models in parallel. The algorithm repeatedly estimates the performance of each model, and eliminates poorly performing candidates from the evaluation process. Hence, racing aims to efficiently find a small number of models that perform well on the given learning problem, by a process of elimination. Racing methods are generally applicable to model selection problems in that they make few assumptions about the space of models. In addition, racing can be used to search a heterogeneous set of models, where it is not clear how to define a distance metric between points in the space. All that is required is the ability to evaluate the performance of a candidate model. However, only a few applications of racing algorithms have appeared in the literature to date [1-4].

In this paper, we present an empirical study of one particular racing algorithm (Hoeffding races) for selecting the k parameter in a simple k -nearest neighbor classifier. The aim of this study is to gain a better understanding of the behaviour of this racing algorithm on a relatively simple model selection problem, and to explore the relationship between these techniques and the datasets applied. In the following Section, we briefly describe Hoeffding racing and its application to model selection in k -nearest neighbour classification. Section 3 describes our methodology and the details of our experiments. Section 4 presents the results of the experiments, and Section 5 summarizes and concludes the paper.

2 Hoeffding Races, Model Selection and Lazy Learning Algorithms

2.1 Model Selection

There are a variety of different existing approaches to the model selection problem in machine learning [5]. In some cases, a model-specific approach may be available (e.g. if model selection is restricted to a single class of models), while for other situations a general search technique could be applied, depending on the parametric form the set of models to be considered. Racing algorithms represent a very general, search-based approach to model selection. Since racing only requires the ability to evaluate a given model on the learning task at hand, it is possible to apply racing to model selection

across a set of instantiations of completely different classes of models. In this paper, we focus on Hoeffding races [1, 2], which makes a minimal assumption regarding the data used to generate statistics in evaluating the set of models used.

2.2 Hoeffding Racing

Consider a supervised learning (e.g. classification) problem where a data set of labeled examples is given. A classifier can be applied to this problem, and its performance evaluated, for example by partitioning the data into training and test sets, or by cross-validation over the data. Assume that N data points are available for testing. This yields an estimate of the true prediction error of the classifier, E_{true} . Consider an estimate of the prediction error after n points, E_{est} , where $n < N$. As n increases, E_{est} becomes closer to E_{true} . Assuming that test points are drawn independently, the Hoeffding bound states that the probability of E_{est} being more than ε away from E_{true} is

$$\Pr(|E_{\text{true}} - E_{\text{est}}| > \varepsilon) < 2e^{-2n\varepsilon^2/B^2} \quad (1)$$

where B bounds the greatest possible error that a model can make. For classification problems, B is simply 1, as a misclassification has an error of 1. A confidence parameter, δ can be introduced and (1) can be rewritten as

$$\Pr(|E_{\text{true}} - E_{\text{est}}| > \varepsilon) < \delta \quad (2)$$

denoting that, with confidence $1-\delta$, E_{est} is within ε of E_{true} . The error bound ε can be expressed as a function of n , B and δ as

$$\varepsilon(n) = \sqrt{\frac{B^2 \log(2/\delta)}{2n}} \quad (3)$$

Alternatively, (3) can be rearranged to give the number of points, n , needed to achieve accuracy, ε , with confidence, δ , as

$$n > \frac{B^2 \log(2/\delta)}{2\varepsilon^2} \quad (4)$$

Hoeffding racing is applied to model selection as follows. Consider a set of classifiers $\{C_i\}$, $i=1, \dots, p$, where each classifier (candidate model) is fully specified in terms of its model parameter values. At each iteration of the racing process, a data point is selected randomly without replacement from the original test set and added to the current test set of each classifier (which is initially empty). The classifiers use this new data to calculate a revised error estimate, E_{est} . The current error bound for each C_i is then calculated using (3). This process results in a set of error estimates for the classifiers, together with (upper and lower) error bounds (see Fig. 1).

Using this information, the racing algorithm may attempt to eliminate classifiers. A model will only be eliminated when its lower error bound is higher than the upper error bound of the best model at that iteration. For example, in Fig. 1, model C_3 is currently the best learning box. Hence, its upper bound is the threshold for elimination. Since the lower bounds of models C_4 and C_5 are higher than the threshold, they can be eliminated in this iteration. This racing process will iterate repeatedly until only one of the learning box left or the accuracy, ε , or the confidence, δ , has reached the threshold. In another word, the ending state of the racing is either when there is only one model left in the racing or when all models left satisfy requirements we set for the racing. Classifiers with their current estimated errors within $2\varepsilon(n)$ range cannot be distinguished; hence, it is unlikely that many models will be eliminated in the first few iterations of racing. As n gets larger, the bound, ε gets smaller and bad models start to be eliminated. In practice, for computational efficiency, it is possible to only consider eliminating boxes periodically (rather than at every single iteration), since the error estimate is likely to vary slowly as n increases.

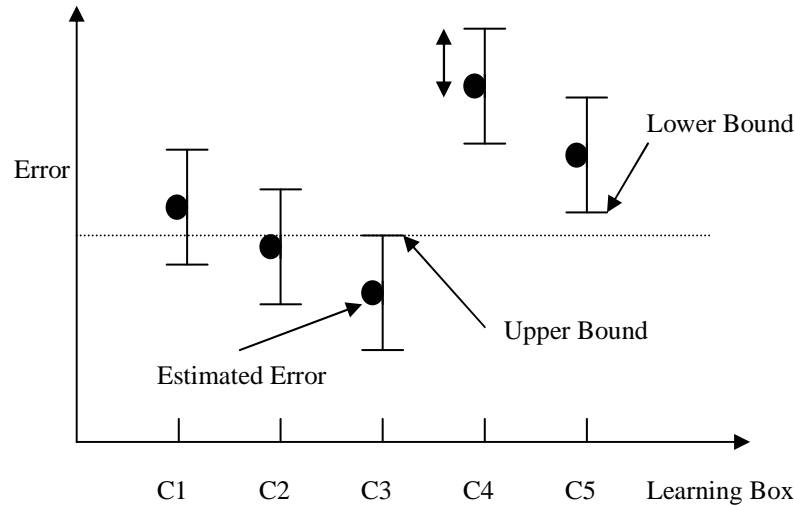


Fig. 1. Hoeffding racing learning boxes: estimated error and error bounds

The Hoeffding racing algorithm attempts to combine the strength of exhaustive search with the computational efficiency of descent methods. It assumes that all learning boxes are independent of each other. Also, the error of current iteration point does not affect the error of next iteration point at all. Hoeffding racing never does worse than brute force and it is not concerned with the models' structure and complexity.

This racing algorithm has better performance when there are some clear winners in the initial set of models, so the elimination begins early and proceeds quickly. On the other hand, when initial set of models are equally good, the racing algorithm is not very effective. However, unlike descent methods, the racing algorithm guarantees the true best model will never be eliminated even though for some problems, it cannot distinguish the best one out from other good ones left. The speed of racing can be accelerated by using "Bayesian" bound, which assumes the errors are normally distributed and gives a tighter bound than Hoeffding bound [2]. However, Bayesian racing is not considered in this paper.

2.3 Lazy Learning

Lazy learning (or memory-based learning) refers to a class of machine learning algorithms that make predictions based on computations over the stored training set, rather than, for example, fitting a model to the data via an iterative process [6]. Well known examples include k -nearest neighbor classifiers and locally weighted regression techniques. In lazy learners, very little work is needed to perform training and most of the computational expense comes when a prediction query is made.

The k -nearest neighbor (k -nn) classifier is a well-known and widely applied technique, due at least partly to its simplicity and its demonstrated good performance on a range of benchmark and practical problems [7]. Given a test data point to be classified, k -nn returns the class (output) of the points with the most similar attributes (input). The similarity between examples and the queried case is typically measured by using the Euclidean or some other distance metric, depending on the type of the attributes. k -nn has a single adjustable model parameter, k , which represents the number of nearest neighbors selected to determine the class prediction. The predicted class output is obtained via a majority vote rule of all contributed example points. When the votes are equal (for an even value of k), the tie might be broken at random, or (as done in this paper) the closest neighbor's class is selected as the predicted class. Generally speaking, it is expected that as k gets larger, the stability against noisy data increases, but at the same time the locality advantage of k -nn is destroyed. However, the sensitivity of the performance of the algorithm will also be problem-dependent.

3 Experimental Methodology

In this research, we are interested in studying the behaviour of the Hoeffding racing algorithm for model selection in k -nn, on a large set of real-world benchmark classification problems. Experimental results will be dependent on the variable factors in the racing algorithm, classifier and dataset used. Nevertheless, we have attempted to consider an experimental scenario that is as simple as possible, and to explore a range of values for each factor.

For the k -nn algorithm, Euclidean distance is used throughout. Leave-one-out cross-validation (LOOCV) was used to evaluate the performance of each classifier. One advantage of using a lazy learner is that no retraining is required to carry out cross-validation; the training set and test point(s) used are simply exchanged appropriately. For each dataset, k -nn was applied exhaustively, i.e. for each possible value of k (in this case from $k=1$ up to $k=N-1$), the performance of the classifier E_{true} is calculated using LOOCV. This represents an exhaustive or brute force approach to the model selection problem for k -nn and is used to compare with the results of racing.

Hoeffding races was applied as an alternative to the above brute force approach. Complete LOOCV uses every point in the dataset for testing, while Hoeffding races sequentially through the data, attempting to eliminate candidate models. Maximum efficiency gain is obtained when as many candidate models as possible are eliminated as quickly as possible. The confidence parameter in Hoeffding races controls the likelihood that models will be eliminated. For low values of δ , models are eliminated on the basis of minimal statistical evidence. This means that more models are eliminated frequently; on the other hand, this increases the chance that a good model will be eliminated erroneously. Alternatively, high confidence values minimize the risk of eliminating the best models, at a possible cost of eliminating less models overall (and reducing the efficiency gain over brute force). In the experiments reported below, Hoeffding racing was run with $\delta = 0.95, 0.9, 0.8, 0.6, 0.5, 0.2$ and 0.05 . Racing proceeds by randomly selecting points (without replacement) to test (and thereby update E_{est}). Therefore, the result of the algorithm depends on the sequence of test points processed and is a random variable. Hence, each racing experiment was repeated 5 times. The set of models considered for racing included $k=1, \dots, 20$ for each dataset used. In addition, 10 further models were used for each dataset, selected with some consideration of the brute-force results (see next Section). Racing runs until the entire dataset has been evaluated – models remaining at the end of the run must therefore satisfy the specified confidence/accuracy.

Table 1. Dataset characteristics. The right column shows the percentage of data in each dataset that belongs to the highest-represented class.

Dataset	#Class	#Attributes	#Instances	Max. Class Distribution
Iris	3	4	150	33.33%
NT	3	5	215	69.77%
Glass	6	9	214	35.51%
Bupa	2	6	345	57.97%
SS	4	35	47	36.17%
M1	2	6	432	50.00%
M2	2	6	432	67.13%
HR	3	4	132	38.63%
HRMod	3	3	132	38.63%
BS	3	4	625	46.08%
Lenses	3	4	24	62.50%
Vehicle	4	18	846	25.37%
Vowel	11	10	990	9.09%
Wdbc	2	30	569	62.74%
Yeast	10	8	1484	31.20%

Fifteen commonly used benchmark datasets were selected from the UCI Machine Learning Repository [8] for these experiments. The characteristics of the datasets are summarized in Table 1. From Table 1, it is clear that a variety of real world classification datasets are included. Also, these datasets cover a wide range in number of instances, number of attributes and maximum class distribution. All datasets selected have either numerical or nominal numerical values. For datasets that have nominal class representation (iris, new-thyroid, soybean-small, balance-scale, vehicle, wdbc and yeast), the class representations are modified manually to appropriate numerical representations. The UCI hayes-roth dataset includes a randomly generated attribute called ‘‘hobby’’. We used this dataset,

and created, a separate dataset called “hayes-rothMod” by deleting the “hobby” attribute from hayes-roth and included in the experiments to show how this random attribute affects the racing performance.

4 Results

Table 2 summarizes the results for k -nn on all the selected datasets. In terms of the lowest LOOCV error obtained, it is evident that k -nn shows a variety of different performances on the different datasets. Interestingly, the optimal value(s) for k also varies widely across the datasets. For some datasets, $k=1$ proves to be optimal, while in other cases, vary large values of k perform best. Figs 2 and 3 show the complete error profiles for all k values, across all datasets used.

Table 2. Summary of results for exhaustive k -nn. Shown are the k values that obtained the best performance in terms of E_{true} (multiple values indicate multiple optimal values of k), the value of E_{true} for the best value(s) of k , and the error difference between the best and worst performing values of k .

Dataset	Best k value	Lowest Average Error	Error Difference
Iris	19 - 21	0.020	0.980
NT	1	0.051	0.251
Glass	1	0.266	0.664
Bupa	9, 32	0.304	0.116
SS	3	0.000	0.638
M1	5, 6, 7, 9	0.000	1.000
M2	1	0.278	0.120
HR	1, 2	0.424	0.576
HRMod	66 - 70	0.341	0.659
BS	10	0.094	0.906
Lenses	4	0.625	0.375
Vehicle	3	0.337	0.663
Vowel	1	0.097	0.236
Wdbc	38 - 569	0.627	0.053
Yeast	21, 29	0.402	0.079

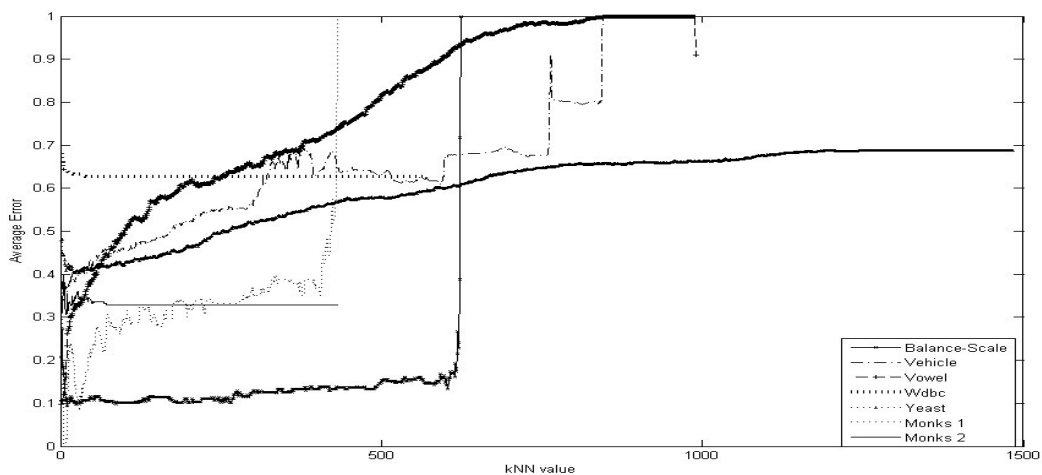


Fig. 2. LOOCV error with respect to different k values in k -nn, for datasets larger than 400 instances.

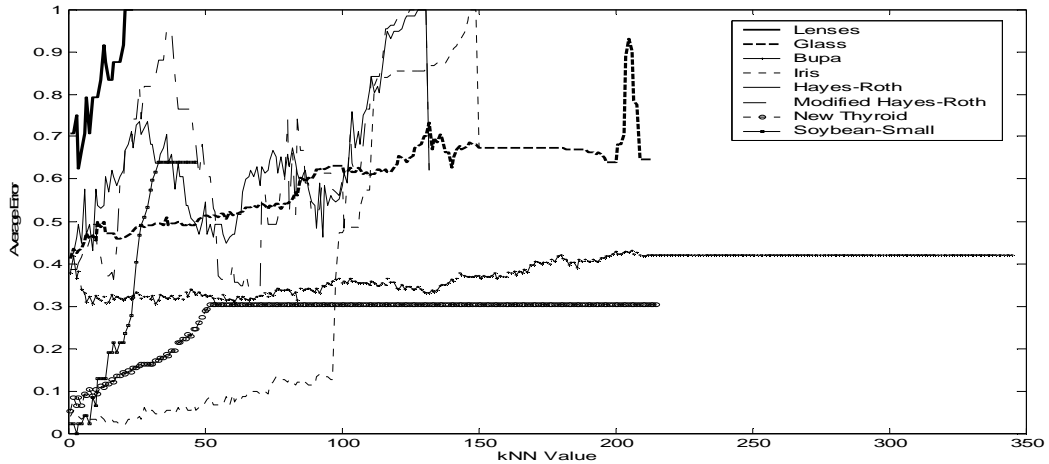


Fig. 3. LOOCV error with respect to different k values in k -nn, for datasets smaller than 400 instances.

The complexities of these profiles shows that the performance of k -nn can be highly sensitive to the value of k . In many cases, the curve does not flatten out as k increases (up to the total size of the training set). Our expectation, based on intuition about how k -nn works, was that curves would typically increase (and to an extent it is clear that low k values tend to work better in general) and flatten out to some asymptotic value as k approached the size of the training set. Evidently, using a local search or trial-and-error approach to perform model selection in k -nn (a 1-D model selection problem!) is likely to yield suboptimal results. As mentioned above, in addition to racing models from $k=1, \dots, 20$, we selected 10 further values of k by selecting values which were at a variety of different error values, to try and provide variation in the models used in racing.

The results of the racing experiments are shown in Figs. 4 and 5, in terms of the number of models eliminated by the end of the racing run and the percentage saving in computation compared to running 30 models exhaustively (i.e. for the entire LOOCV calculation over the dataset). Note that, as model are eliminated dynamically at various points during the race, the percentage saving and the total number of boxes eliminated at the end of the trails do not describe the same information. All values are averaged over the 5 repeated trials of each racing run. As the confidence value decreases, the number of models eliminated (and hence the % saving) generally tends to increase. This is in line with the aim of the confidence parameter; nevertheless, due to the random selection of datapoints in racing, occasionally this saving decreases. In addition, lower confidence values generally results in models being eliminated earlier in the racing run, which results in a bigger saving in computation time.

The performance of racing in terms of the computational savings gained was variable and overall perhaps lower than expected. For 90% confidence, racing only produced a saving of greater than 50% on two of the datasets. On the other hand, racing proved to be very conservative in terms of which models were eliminated. We examined which models were eliminated over the racing runs and compared this with the optimal values of k found from the exhaustive k experiments. It was not until the confidence was set to 5% that any racing run eliminated any of the best performing k values for any datasets, and then this happened only occasionally (twice for “Glass” and “Bupa” and once for “HayesRoth” and “Yeast”). The distribution-free assumption behind the Hoeffding bound results in a conservative confidence estimate, and our results suggest that this is true even with lower confidence values that have been previously considered.

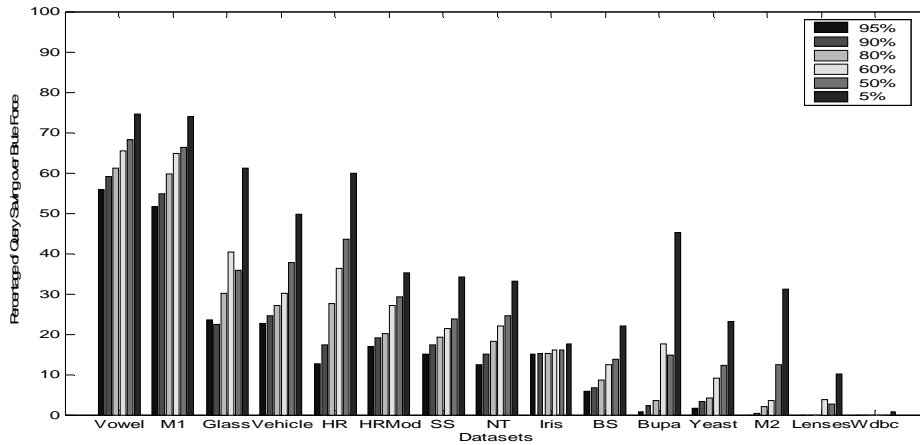


Figure 4. Racing results for each dataset over different confidence levels: average percentage of computation time saving compared to exhaustive evaluation of 30 models.

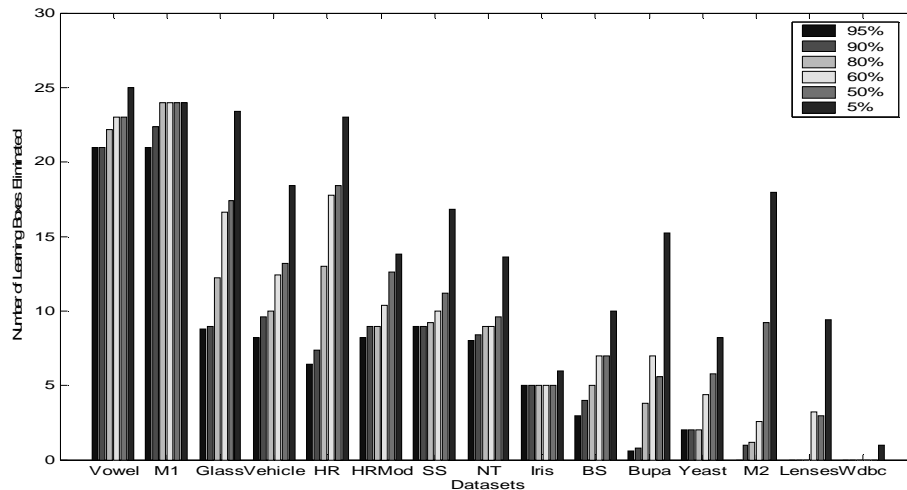


Figure 5. Racing results for each dataset over different confidence levels: average number of learning models eliminated over each racing run.

5 Conclusion

This paper has presented an empirical study of the performance of the Hoeffding racing algorithm for model selection in the simple k-nn classifier, on a suite of UCI datasets. The results reveal surprising variability in the performance of k-nn with respect to the value of k, and highlight the danger in relying on trial-and-error approaches to parameter tuning. The results of applying Hoeffding races to the k-nn model selection problem show that this racing algorithm is sometimes able to provide significant computation benefits compared with an exhaustive evaluation; however performance depends strongly on the dataset in question. In addition, on these datasets the racing algorithm was more likely to provide less computational saving than to incorrectly eliminate the best models from consideration. Although it is not possible to directly compare our results with those in [1,2] (where different datasets,

including regression problems were considered), the results in this paper indicate that racing is sometimes less effective than might have been previously thought.

We have only considered Hoeffding racing in this paper. Alternative racing criteria (e.g. Bayesian races [2]) provide tighter bounds as a result of making stronger assumptions about the distribution over the evaluation error. It would be interesting to conduct experiments comparing a variety of racing criteria to examine their sensitivity to different datasets.

References

1. Maron, O. and Moore, A.W. *Hoeffding Races: Accelerating Model Selection Search for Classification and Function Approximation*. Advances in Neural Information Processing System, ed. J.D. Cowan, Tesauro, G. Alspector, J. Vol. 6. 1994: Morgan Kaufmann.
2. Maron, O. and Moore, A. W. *The Racing Algorithm: Model Selection for Lazy Learners*. Artificial Intelligence Review, 1997. **11**: p. 193-225.
3. Birattari, M., Stutzle, T., Paquete, L. and Varrentrapp, K. *A Racing Algorithm for Configuring Metaheuristics*. in *Genetic and Evolutionary Computation Conference 2002*. 2002. New York, USA: Morgan Kaufmann Publishers.
4. Yuan, B. and Gallagher, M. *Statistical Racing Techniques for Improved Empirical Evaluation of Evolutionary Algorithms*. in *Parallel Problem Solving from Nature VIII*. 2004. Birmingham, UK: Springer.
5. Hastie, T., Tibshirani, R. and Friedman, J. *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer series in statistics. 2001, New York, USA: Springer.
6. Atkeson, C. G., Moore, A. W. and Schaal, S. *Locally Weighted Learning*. Artificial Intelligence Review, 1997. **11**:11-73.
7. Mitchell, T.M., *Machine Learning*. 1997, New York, USA: The McGraw-Hill Companies, Inc.
8. Blake, C. L. and Merz, C.J. *UCI Repository of Machine Learning Databases*. 1998: California, USA. <http://www.ics.uci.edu/~mllearn/MLRepository.html>