Piece-Wise Model Fitting Using Local Data Patterns

Ricardo Vilalta and **Muralikrishna Achari** and **Christoph Eick** [1]

Abstract. In this paper we propose a novel classification algorithm that fits models of different complexity on separate regions of the input space. The goal is to achieve a balance between global and local learning strategies by decomposing the classification task into simpler subproblems; each task narrows the learning problem to a local region of high example density over the input space. Specifically, our proposed approach is to apply a clustering algorithm to every set of training examples that belong to the same class; each cluster becomes an intermediate concept that is learned by selecting a model with an (estimated) optimal degree of complexity. Experimental results on real-world domains show consistent good performance in predictive accuracy with our piece-wise model fitting strategy.

1 INTRODUCTION

Research in machine and statistical learning has placed at our disposal a rich set of classification algorithms, enabling us to group and understand these algorithms based on their different data-analysis strategies. A rough categorization deals with the kind of information used in computing class posterior probabilities; we can either employ all available training examples (i.e., global strategy) or give higher weight to those training examples in the neighborhood of the query example (i.e., local strategy). Both global and local learning strategies sit at two extremes of a large spectrum of possible compromises that exploit information from labelled examples. Our goal is to explore intermediate points of this spectrum based on established facts about the strengths and limitations of each strategy.

On the one hand, global learning algorithms fit a single model to the whole training data, even when the complexity of the model results inadequate on different regions of the input space. For example, a simple global model (e.g., linear combination of feature values, Naive Bayes, single logical rules, etc.) is often inadequate under regions of high example density if Bayes error is low and higher flexibility in the decision boundaries lowers the empirical risk [3]. Here the final hypothesis is drawn from a small class of approximating functions; the poor repertoire of functions produces high bias (since the best approximating function may lie far from the target function) but low variance (since there is little dependence on local irregularities in the data). The alternative is to increase the degree of complexity by drawing a hypothesis from a large class of approximating functions (e.g., neural networks with a large number of hidden units); here the hypothesis exhibits flexible decision boundaries (low bias) but becomes sensitive to small variations in the data (high variance). This is inadequate if data is sparse or a function with lower variance achieves the same empirical risk. In both cases the final hypothesis is simply averaged altogether, prone to overshoot the bias or variance component of error as example densities vary throughout the input space.

At the other extreme of the spectrum sit local learning algorithms [1, 3, 13]. These algorithms fit a different model around each query example and so pay attention to local irregularities in the data at the expense of using biased estimates of class (posterior or conditional) probabilities (e.g., k-nearest neighbor classifiers with small values for k). Reducing the degree of locality (e.g., by increasing k or the width of a kernel function) improves the probability estimates but loses sensitivity to small variations in the data. Estimating class probabilities by gathering statistics around a query example fails to capture how class clusters distribute throughout the input space.

In this study we propose a learning strategy that fits models of different complexity to separate regions of the input space. Our goal is to achieve a balance between local and global learning by gaining more control over the bias-variance tradeoff in generalization error [4]. First, locality is achieved by decomposing each class into different clusters, each cluster representing a local characterization of the class-conditional distribution [15]. Instead of modelling this distribution as a mixture of components [6], we treat each cluster as a subclass that must be learned independently of the rest. This hierarchical representation of classes where each class divides into clusters enables us to fit models of varying complexity on different regions of the input space, while retaining most available examples for training. Empirical results on real world domains show consistent good performance with this approach, particularly when each cluster contains enough examples to be modelled accurately.

This paper is organized as follows. Section 2 introduces background information on classification and discriminant functions. Section 3 compares our strategy to that of global and local learning methods. Section 4 describes our piece-wise model fitting strategy. Section 5 discusses related work. Section 6 reports on our experimental analysis. Finally, Section 7 gives a summary and discusses future work.

2 BASIC TERMS AND NOTATION

Let $(A_1, A_2, \cdots, A_n)$ be an n-component vector-valued random variable, where each A_i represents an attribute or feature; the space of all possible attribute vectors is called the input space $\mathcal{X}$. Let $\{y_1, y_2, \cdots, y_k\}$ be the possible classes or categories; the space of all possible classes is called the output space $\mathcal{Y}$. A learning algorithm receives as input a set of training examples $T = \{(\mathbf{x}, y)\}$, where $\mathbf{x} = (a_1, a_2, \cdots, a_n)$ is a vector or point of the input space and y is a point of the output space. The outcome of the learning algorithm is a function h (i.e., hypothesis) mapping the input space to the output space, $h : \mathcal{X} \rightarrow \mathcal{Y}$. Function h can then be used to predict the class of previously unseen attribute vectors.

We consider the case where a learning algorithm defines a discrim-

[1] Department of Computer Science, University of Houston. 4800 Calhoun Rd. Houston TX 77204-3010. Email: {vilalta,amkchari,ceick}@cs.uh.edu

inant function for each class $g_j(\mathbf{x})$, $j = 1, 2, \cdots, k$ and chooses the class corresponding to the discriminant function with highest value (ties are broken arbitrarily):

$$h(\mathbf{x}) = y_m \texttt{ iff } g_m(\mathbf{x}) \geq g_j(\mathbf{x}) \; \forall j \neq m \tag{1}$$

Discriminant functions can be seen as instantiations of generic models. For example, a simple discriminant function is based on a linear model:

$$g_j(\mathbf{x}) = w_0 + \sum_{i=1}^{n} w_i a_i \tag{2}$$

where each $w_i, 0 \leq i \leq n$, is a coefficient that must be estimated by the learning algorithm.

Discriminant functions can also be instantiations of probabilistic models; these models are often proportional to the class posterior probabilities:

$$g_j(\mathbf{x}) = P(y_j|\mathbf{x}) = \alpha P(y_j) P(\mathbf{x}|y_j) \tag{3}$$

where $P(y_j)$ is the a priori probability of class y_j, $P(\mathbf{x}|y_j)$ is called the likelihood or class-conditional probability, and α is a normalization factor.

3 DATA-DRIVEN ADAPTATION OF MODEL COMPLEXITY

Our approach to achieve a compromise between global and local learning in classification is based on varying the complexity of a model according to the characteristics of the data distribution. To begin we detail on some limitations pertaining to global and local learning methods. Consider first the case of a global estimation of model parameters when the learning algorithm is simple. The limited size of the space of approximating functions results in many example distributions appearing identical; the learning algorithm tends to smooth all class probabilities, often by relying on data projections on single features. As an example consider the mechanism behind Naive Bayes [11], where every discriminant function is equivalent to Equation 3, and class conditional probabilities are simple product approximations:

$$g_j(\mathbf{x}) = P(y_j) P(\mathbf{x}|y_j) = P(y_j) \prod_{i}^{n} P(a_i|y_j) \tag{4}$$

By computing class-conditional probabilities on each feature separately, Naive Bayes attempts to reconstruct the true joint input-output distribution using a product of low-order class conditional probabilities (i.e., low order components). The reconstruction is prone to miss local irregularities in the data, since among all distributions that can produce those same low-order components, Naive Bayes selects only the distribution having maximum entropy [9]. The same can be said of linear models (Equation 2) where the number of approximating functions is limited by the space of linear feature combinations. Many distributions map into the same function despite evident differences in local data patterns.

The problem of naively ignoring local irregularities in the data by collapsing multiple distributions into few functions can be tackled by simply increasing the capacity of the learning machine (i.e., the model complexity). But the advantage of this approach is soon downplayed by the additional increase in variance, as well known in statistical inference[4, 5]. As an illustration, Figure 1 shows a two dimensional input space with two classes. The distribution of examples

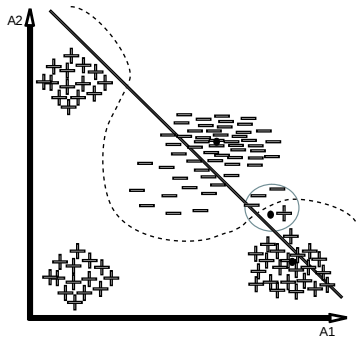

Figure 1. A high-order polynomial (dashed line) improves the classifi cation of a linear model (bold line) at the expense of increased variance. A local classifi er (circle) fails to identify cluster membership information.

precludes a simple linear model attaining good performance (Figure 1 bold line). One way to increase the complexity of the model is to enlarge the original space of linear combinations to allow for more flexibility on the decision boundaries, for example by adding higher order polynomials (Figure 1, dashed line). But this comes at the expense of increased variance and possibly data overfitting.

A different solution is to follow a local learning strategy by tracing a graded circular area around the query example (Figure 1 circle). This is normally achieved by placing a kernel around the query example (e.g., a Gaussian kernel) that dictates how much weight is assigned to neighbor examples. This method pays too much attention to irregularities in the data (often resulting in high variance). In addition, it disregards the location of the query example with respect to dense areas of high posterior probabilities. Figure 1 (circle) shows a scenario where the query example is assigned class negative, despite it having a higher degree of cluster membership on class positive.

3.1 Class-cluster decomposition

Our compromise between global and local learning strategies is to increase the number of discriminant functions by assigning each an easier task, narrowing the location of its decision boundaries to a more confined region of the input space. The idea is to exploit information on the example distribution through a pre-processing step that identifies natural clusters in data [14, 15]. Each corresponding cluster represents a local characterization of the class-conditional distribution.

As an illustration, Figure 2 shows an example distribution identical to Figure 1, but with the new goal of fitting three models corresponding to the different clusters into which class positive can be decomposed. Each resulting function is in charge of distinguishing between examples within the cluster from examples outside the cluster. By increasing the number of decision boundaries per class we enable the learning algorithm to choose models of different complexity according to variations in example density. This hierarchical representation where each class divides into multiple clusters adds locality to the classification problem while retaining all examples for training.

Before providing a more detailed description of our approach (Section 4) we justify the importance behind using multiple decision boundaries within each class as a means to reduce bias while limiting the growth in variance.

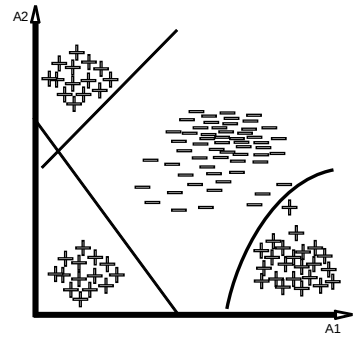

Figure 2. Learning clusters in a piece-wise manner using models of varying complexity; class positive decomposes into three clusters.

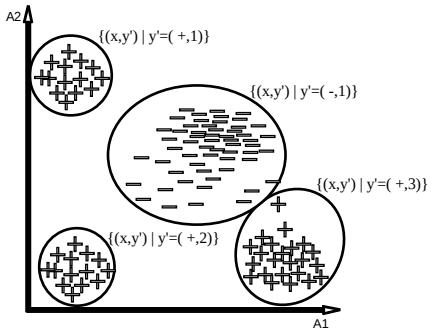

Figure 3. The class-decomposition process relabels examples to encode both class and cluster.

3.2 Capacity and the VC dimension

We make use of the VC-dimension [12] to compare the increase in complexity obtained by augmenting the number of boundaries per class (our approach) to the increase in complexity obtained by augmenting the capacity of a single global learning algorithm. First, consider a learning algorithm that has a small class of approximating functions ϕ from which to draw a hypothesis. If we wish to make class ϕ stronger we must increase its size. Recall, however, that adding too much representational power increases the variance component of error. Here we provide evidence showing that our approach increases the representational power of ϕ in small steps, enabling us to gain more control over the bias-variance tradeoff in generalization error. We start by introducing standard definitions (extracted from [5]):

Definition 1. *A set of points D is said to be shattered by a class of functions ϕ, if for every possible assignment of (binary) class labels to the points in D, there is a member function $h \in \phi$ that perfectly classifies all examples in D.*

Definition 2. *The VC dimension of a class of functions ϕ, VC(ϕ), is the size of the largest set D than can be shattered by ϕ.*

The VC dimension quantifies the representational power of ϕ in separating examples away and plays an important role in our understanding of generalization error [12]. For our purposes, we look at how VC(ϕ) varies as ϕ grows in representational power. Consider first the case where we wish to increase the complexity of a global learning algorithm by enlarging the class of functions to include high-order polynomials.

Observation 1. (extracted from [2]). *For any positive integer l, if ϕ_l is the class of functions of polynomials of degree l in the input space $\mathcal{R}^n$, then VC(ϕ_l) is of order $\mathcal{O}(n^l)$.*

Therefore, the VC dimension grows exponentially as we increase the degree of the polynomial. Consider now our proposed approach where the goal is to increase the number of decision boundaries per class while trying to keep the complexity of each boundary as low as possible.

Observation 2. (extracted from [8, 2]) *Let ϕ_l be the class of functions made of l intersecting hyperplanes (i.e., polygons). That is each function partitions the input space by intersecting l different hyperplanes. The VC dimension of this class of functions (over the plane $\mathcal{R}^2$) is VC(ϕ_l) $= 2l + 1$.*

Thus, in this case the VC dimension grows linearly with the number of hyperplanes (i.e., of linear boundaries). The results above suggest the complexity of a classifier, as measured by the VC dimension, grows at a slower rate when we increase the number of boundaries but limit their complexity (e.g., three boundaries on Figure 2), than when we increase the flexibility of a single global decision boundary (e.g., dashed boundary on Figure 1). This is expected to be true as long as each local classifier has complexity lower than the complexity of the global classifier under comparison. Thus in looking for a trade-off between bias and variance, our approach provides an interesting mechanism to reduce bias while limiting the growth in variance.

4 PIECE-WISE MODEL FITTING

In this section we provide a detailed description of our piece-wise model fitting approach. Our solution is a two-step process: 1) decompose classes into clusters; 2) fit a model with the right (estimated) complexity for each data cluster. We look to each step in turn.

4.1 The decomposition process

The first step pre-processes the training data by clustering examples that belong to the same class, that is $T = \{T_j\}$, where each T_j comprises all examples in T labelled with class y_j, $T_j = \{(\mathbf{x}, y) \in T | y = y_j\}$. Specifically, for each set T_j we apply a clustering algorithm C to find sets of examples (i.e., clusters) grouped together according to some distance metric over the input space[2]. Let $\{c_i^j\}$ be the set of such clusters. We map the set of examples in T_j into a new set T_j' by renaming every class label to indicate not only the class but also the cluster to which each example belongs. One simple way to do this is by making each class label a pair (a, b), where the first element represents the original class and the second element represents the cluster that the example falls into. In that case, $T_j' = \{(\mathbf{x}, y_j')\}$, where $y_j' = (y_j, i)$ whenever example $\mathbf{x}$ is assigned to cluster c_i^j.

An illustration of the decomposition process is shown in Figure 3. The transformation relabels every example to encode class and cluster label. As a result, dataset T' has now four different classes. Finally the new dataset T' is simply the union of all sets of examples of the same class relabelled according to the cluster to which each example belongs, $T' = \bigcup_{j=1}^{k} T_j'$.

4.2 The model-fitting process

The second step fits models of varying complexity to each cluster. Algorithm 1 (Figure 4) shows the logic behind this step. The input

[2] We consider a flat type of clustering (as opposed to hierarchical) and assume each object is assigned to exactly one cluster.

Algorithm 1: Model-Fitting Process
Input: dataset T'
Output: set of discriminant functions G
MODEL-FITTING(T')
(1) **foreach** class-uniform set T'_j in T'
(2) Relabel all examples outside T'_j as $(-)$
(3) **foreach** model M of increasing complexity
(4) Learn function g_j using model M in T_j
(5) Assess g_j on validation set
(6) Keep track of most accurate function g^*_j
(7) **end**
(8) **end**
(9) **return** $G = \{g^*_j\}$

Figure 4. The process to fi t models of varying complexity on each of the original clusters.

to the algorithm is dataset T', where examples in T have been relabelled to include class and cluster label. Hence, each new class in dataset T' corresponds to a cluster derived from the original dataset T; if at least one class decomposes into more than one cluster, the number of classes in T' is larger than the classes in T.

Our goal is to investigate a new approach to the problem of modelling class-conditional distributions. Assuming a mixture of (Gaussian) models where each cluster distributes in a similar way is often unrealistic. Instead we try to learn each cluster separately as follows. For each class in T' (i.e., for each cluster in T comprising examples of the same class) we relabel all examples outside the class as negative $(-)$ (Algo. 1, line 2). The goal at this point is to find a discriminant function that can accurately separate the two classes; this is equivalent to finding a function to classify the original cluster.

To achieve our goal we fit models of varying complexity to the relabelled data, and select the function that is most accurate on a validation set (Algo. 1, lines 3-7). Fitting models of varying complexity can be achieved in different ways. Our experiments use a support vector machine with a polynomial kernel; the degree of the polynomial is varied from one to three; the final degree is based on the best accuracy obtained on a validation set. Since we favor local models of low complexity, ties are broken by selecting the polynomial with lowest degree. The rationale behind this idea is twofold: 1) learning to classify each cluster does not necessarily assume a parametric model, and 2) the use of local models allows to increase model complexity in finer steps (Section 3.2).

The output of the model-fitting process is a set of discriminant functions $G = \{g^*_j\}$, each function g^*_j learning to classify each of the original clusters (Algo. 1, line 9). Upon arrival of a new query example, we simply select the discriminant function with highest value (Equation 1). To know the actual prediction in the original output space $\mathcal{Y}$, we predict class label y_j whenever example $\mathbf{x}$ is assigned to any of the clusters of class y_j. This is simply achieved by removing the cluster label from the predicted class-cluster pair (i.e., $(y_j, i) \rightarrow y_j$).

5 RELATED WORK

The idea of identifying clusters of examples as a pre-processing step to classification had been used before to improve the mechanism of Naive Bayes [14, 15]. Instead of applying a global classifier to the transformed space, however, our approach attempts to learn each cluster separately.

Our work bears some similarity to the idea of mixtures of local experts [6, 7], in which different classifiers compete to control different regions of the input space, and a gate network is used to weight the output of the classifiers (i.e., models the mixing parameters of a mixture distribution). Instead of searching for regions of expertise as part of the classification process, our approach differs in separating the problem into a clustering and a classification step; the clustering phase serves to pre-identify local patterns that can then be learned using models of varying complexity.

The connection between our approach and local learning can be established by looking at the difference between costs functions. In local learning [3, 13, 1], parameter estimation of the approximating function is based on minimizing a local cost:

$$C(\mathbf{x}) = \sum_i L(h(\mathbf{x}_i), t(\mathbf{x}_i)) K(||\mathbf{x}_i - \mathbf{x}||) \quad (5)$$

where the sum goes over all training examples[3], L is the loss or cost of predicting class $h(\mathbf{x}_i)$ when the true class is $t(x_i)$, and K is a kernel function that gives higher weight to those examples closer to $\mathbf{x}$. In contrast, our class decomposition process first detects the existence of local patterns through clustering, and then transforms the output space to capture those patterns. Generally speaking, we try to minimize a global cost:

$$C = \sum_i L(h'(\mathbf{x}_i), t'(\mathbf{x}_i)) \quad (6)$$

where h' and t' are now functions mapping input vectors to a new output space: $\mathcal{X} \rightarrow \mathcal{Y}'$. The mapping leaves the input space $\mathcal{X}$ intact but changes the output space $\mathcal{Y}$ into a (possibly) larger space $\mathcal{Y}'$ (i.e., $|\mathcal{Y}'| \geq |\mathcal{Y}|$, where $|\cdot|$ is the cardinality of the space), depending on the amount of cluster decomposition. Since the kernel function is eliminated, all examples bear now equal importance to classification. Local information of the data distribution is encoded in the new output space.

Our work can be compared to the mechanism of radial basis functions [5], where the final hypothesis is represented as a linear combination of models. Commonly the shape of each model follows a standard Gaussian density (through a prototype and scale parameters). In practice, however, the Gaussian assumption comes unwarranted; our approach instead is to learn local distributions by fitting local models to each cluster.

6 EXPERIMENTS

In our experiments, the clustering algorithm follows the Expectation Maximization (EM) technique [10]; it groups examples into clusters by modelling each cluster through a probability density function. Each example in the dataset has a probability of class membership and is assigned to the cluster with highest posterior probability. The number of clusters is estimated using cross-validation. All our implementations use the WEKA machine-learning class library [16], set with default values.

On each run we use 80% of the examples for training, 10% for validation, and 10% for testing. Reports on accuracy are the average of ten runs (over the testing set). Tests of significance are reported at the $p = 0.05$ level using a two-tailed t-student distribution. Our datasets come from the University of California at Irvine (UCI) Repository. To prevent the situation where the clustering algorithm lacks enough evidence, we considered only datasets with at least 200 examples.

[3] The summation is replaced by an integral if the input space is continuous.

Table 1 displays our results. The second and third columns report the mean accuracy[4] of a local learning algorithm (Nearest Neighbor with $k = 1$) and a global learning algorithm (Neural Network) respectively. The fourth column corresponds to our approach where each cluster is learned using a support vector machine with a polynomial kernel of varying degree (Section 4.2). The fifth column shows if there is a significant difference between our approach and Nearest Neighbor ($*$), and/or if there is a significant difference between our approach and Neural Network ($+$). The sixth column shows the average degree of the polynomial kernel for each class-unform cluster.

Compared to Nearest Neighbor our approach is significantly better in ten datasets, and only worse in two datasets. Compared to Neural Network, our approach is similar, with two significant wins and two significant losses. Two datasets, Letter and Vowel, appear difficult to learn under our approach. Both datasets decompose into the highest number of clusters. The average number of clusters obtained from the class decomposition step over all datasets in Table 1 is 16.4. The Letter and Vowel datasets decompose into 76 and 51 clusters respectively. Results can be explained by observing that decomposing classes into many clusters increases the likelihood of finding clusters comprising few examples. When learning each cluster this results in a class imbalance conflict; the size of each cluster is much lower than the size of the complement dataset.

An interesting observation is that our approach is never significantly worse than both other algorithms (except for dataset Letter, explained above), which points to an interesting balance between local and global learning strategies. In addition, the average degree for the polynomial kernel on each dataset shows most clusters can be learned with simple linear or quadratic models (last column, Table 1), which facilitates their interpretation.

Table 1. Predictive accuracy on real-world domains.

Domain	Nearest Neighbor	Neural Network	Local Fitting	Test Sig.	Avg. Degree
Autos	74.73	77.30	72.10		1.40
Balance-Scale	86.19	92.38	91.26	$*$	1.40
Breast-Cancer	69.64	68.92	70.3		1.39
Breast-W	96.28	96.28	96.14		1.28
Colic	80.55	79.44	81.38	$*$	1.49
Credit-a	82.70	82.17	85.78	$*+$	1.40
Credit-g	70.80	70.60	72.40		1.40
Diabetes	67.40	72.80	75.58	$*+$	1.80
Heart-c	74.60	78.90	81.33	$*$	1.20
Heart-Statlog	76.29	78.88	84.81	$*$	1.50
Hypothyroid	63.48	71.66	71.96	$*$	1.60
Chess	90.60	97.49	96.95	$*$	1.21
Letter	73.26	72.49	67.21	$*+$	2.00
Mushroom	99.77	99.55	99.55		1.00
Tumor	41.47	41.45	45.29		1.30
Sick	87.30	90.6	92.17	$*$	1.70
Soybean	90.94	94.56	94.20		1.09
Vehicle	68.92	82.38	79.16	$*+$	2.00
Vote	93.72	93.95	95.81		1.20
Vowel	98.48	92.62	93.53	$*$	2.10

7 SUMMARY AND FUTURE WORK

In this paper we exploit information obtained by clustering sets of examples to divide the classification problem into simpler subproblems. Every cluster represents a subconcept that can be learned independently of the rest. Locality is achieved through a piece-wise characterization of the class-conditional distribution. Since we retain all examples for analysis, class probability estimations do not suffer from biased local samples.

Our piece-wise model fitting approach is justified by the difference in the rate of complexity obtained by augmenting the number of boundaries per class (our approach) to the increase in complexity obtained by augmenting the capacity of a single global learning algorithm (Section 3.2). The former enables us to increase the model complexity in finer steps.

Experiments in real-world domains show consistent good performance in predictive accuracy with our piece-wise model fitting strategy; our approach exhibits a good balance between local and global learning strategies. Our experiments also show some datasets decompose into an excessive number of clusters (e.g., dataset Letter decomposes into 76 clusters); in this case learning to discriminate clusters often results in a class imbalance conflict. Future work will explore the use of different sampling techniques in an attempt to model each cluster accurately.

Finally, we plan to explore how to couple the clustering and classification steps more tightly. For example, some parameters in the clustering algorithm (e.g., number of clusters) can be estimated by searching for an optimum in classification accuracy.

[4] For space reasons we omit reports on standard deviations; these can be obtained directly from the authors.

ACKNOWLEDGEMENTS

We thank the reviewers for their valuable comments. This work was partially supported by a grant from the University of Houston.

REFERENCES

[1] C. G. Atkeson, A. W. Moore, and S. Schaal, 'Locally Weighted Learning', *Artificial Intelligence Review*, **11**, 11–73, (1997).
[2] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth, 'Learnability and the Vapnik-Chervonenkis Dimension', *Journal of Association for Computing Machinery*, **36**(4), 929–965, (1989).
[3] L. Bottou and V. Vapnik, 'Local Learning Algorithms', *Neural Computation*, **4(6)**, 888–900, (1992).
[4] S. Geman, E. Bienenstock, and R. Doursat, 'Neural Networks and the Bias/Variance Dilemma', *Neural Computation*, 1–58, (1992).
[5] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning; Data Mining, Inference, and Prediction*, Springer-Verlag, 2001.
[6] R. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, 'Adaptive Mixtures of Local Experts', *Neural Computation*, **3**, 79–87, (1991).
[7] M. I. Jordan and R. Jacobs, 'Hierarchical Mixturs of Experts and the EM Algorithm', *Neural Computation*, **6**, 181–214, (1994).
[8] M. Kearns and U. Vazirani, *An Introduction to Computational Learning Theory*, MIT Press, 1994.
[9] P. M. Lewis, 'Approximating Probability Distributions to Reduce Storage Requirements', *Information and Control*, **2**, 214–225, (1959).
[10] G. McLachlan and T. Krishnan, *The EM Algorithm and Extensions*, John Wiley and Sons, 1997.
[11] T. Mitchell, *Machine Learning*, Springler-Verlag, 1997.
[12] V. Vapnik, *The Nature of Statistical Learning Theory*, Springler-Verlag, 1999.
[13] V. Vapnik and L. Bottou, 'Local Algorithms for Pattern Recognition and Dependencies Estimation', *Neural Computation*, **5**(6), 893–909, (1993).
[14] R. Vilalta, M. Achari, and C. Eick, 'Class Decomposition Via Clustering: A New Framework For Low-Variance Classifi ers', *IEEE Third International Conference on Data Mining*, 673–676, (2003).
[15] R. Vilalta and I. Rish, 'A Decomposition Of Classes Via Clustering To Explain And Improve Naive Bayes', *European Conference on Machine Learning*, 444–455, (2003).
[16] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*, Academic Press, London U.K, 2000.