Identifying and Characterizing Class Clusters to Explain Learning Performance

Ricardo Vilalta
Department of Computer Science, University of Houston
4800 Calhoun Rd., Houston, TX 77204-3010
vilalta@cs.uh.edu

Abstract

We stress that the field of machine learning would benefit significantly if more work were focused on explaining learning behavior by attending to the connection between learning strategies and the different types of example distributions. The goal of this paper is to exemplify the benefits of such kind of studies through a simple decomposition of classes into clusters. We show how the decomposition process is instrumental in explaining the behavior of Naive Bayes. We then exploit the same decomposition process to propose a new approach to learning that adapts to the characteristics of the dataset under analysis. Specifically, the idea is to fit models of different complexity over separate regions of the input space.

Introduction

Despite the long and diverse array of available learning algorithms, the reasons explaining why a particular algorithm is more successful than others on specific tasks remains elusive. A fundamental question awaits unanswered: what works well where? (Vilalta, Oblinger, & Rish 2000). Learning algorithms are commonly compared by averaging their performance (e.g., off-training set accuracy) over benchmark domains. But little work has been done to explain performance based on the characteristics of the underlying model and the example distribution (Rendell & Cho 1990).

Consider the following scenario. Learning algorithm $L_{A'}$ is said to be superior to L_A because the modification introduced in $L_{A'}$ conveys a more suitable form of bias[1]. As an example, suppose a researcher decides that a neural net, $\mathrm{NN_{bp}}$, trained via backpropagation, appears to provide the best predictive accuracy over a certain class of tasks $\mathcal{T}$. The researcher decides to modify $\mathrm{NN_{bp}}$'s mechanism by using a different sigmoid function (i.e., squashing function), eventually obtaining a significant average increase Δ_{acc} in predictive accuracy (measured via some re-sampling technique, e.g., n-fold cross-validation). The researcher then claims to have produced a better algorithm $\mathrm{NN_{bp}}'$ over $\mathcal{T}$. In this example, the reason for the success of $\mathrm{NN_{bp}}'$ is unclear (no analysis is provided explaining the reason for the success of the proposed improvement). Implicit assumptions are 1) that there is no need to scrutinize the trained model —in this case, a neural net trained via backpropagation— and 2) that there is no need to characterize the data distribution from which the training examples are drawn. The modification is gladly accepted as long as Δ_{acc} results in a positive significant difference.

The example described above failed to provide answers to two fundamental questions:

1. **Algorithm Design**. Why does the proposed change in the algorithm design produced better performance on that particular set of tasks? Using a different sigmoid function leaves most components of the basic model unaltered: linear combinations of weighted inputs still form the basis to fire each neural node; lower layer nodes serve as input to upper layer nodes; error minimization (on the training set) is achieved by conducting a hill-climbing search through a weight space, etc. What principled guidelines exist to modify a learning algorithm to generate better performance?
2. **Data Characterization**. What characteristics of a sample drawn from a fixed but unknown distribution over the input-output space are more decisive in affecting the performance of a learning algorithm? What kind of theory can be generated to understand the relation between example distributions and learning algorithms?

In this paper we discuss how the field of machine learning would benefit significantly by studying the connection between learning strategies and data distributions. To narrow our research scope, we have given higher priority to the second goal posted above. In particular, we show how even a simple characterization of different example distributions would stand as a major step towards understanding learning performance.

Data Characterization in Meta-Learning

The problem of characterizing data landscapes ia at the core of meta-learning; the cental idea is that high-quality data characteristics or meta-features provide enough information to differentiate the performance of a set of given learning

[1]The term *bias* is used here as a preference of one hypothesis over another; it is defined by the class of hypotheses or functions $\mathcal{H}$ (i.e., by the hypothesis space –of possibly infinite cardinality).

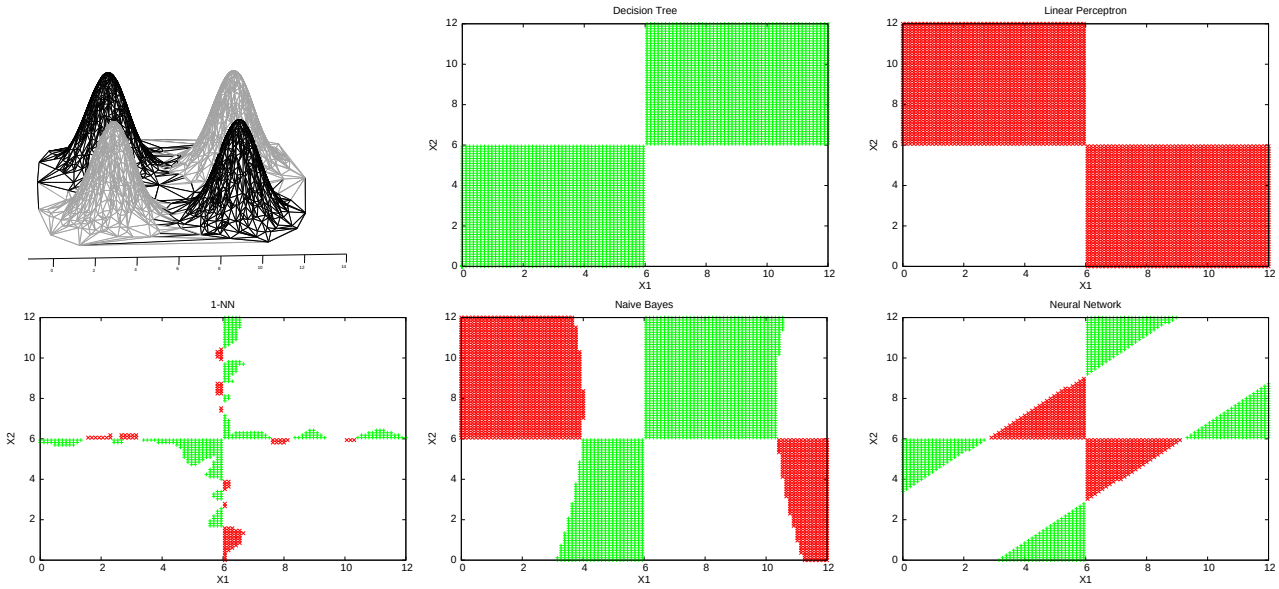

Figure 1: (top-left) A two-dimensional classification problem with 4 Gaussian models (2 for each class) distributed in an XOR manner. (top-middle to bottom-right) Errors patterns for a decision tree, a perceptron, a 1-nearest neighbor, naive Bayes, and a neural network.

algorithms (Aha 1992; Michie 1994; Gama & Brazdil 1995; Brazdil, Soares, & Pinto da Costa 2003).

Much work in data characterization has concentrated on extracting statistical and information-theoretic parameters estimated from the training set (Aha 1992; Michie 1994; Gama & Brazdil 1995; Engels *et al.* 1998; Sohn 1999). Measures include number of classes, number of features, ratio of examples to features, degree of correlation between features and target concept, average class entropy and class-conditional entropy, skewness, kurtosis, signal to noise ratio, etc. This work has produced a number of research projects with positive and tangible results (e.g., ESPRIT Statlog (1991-1994) and METAL (1998-2001)).

In addition to statistical measures, other approaches include *model-based* characterization, where the idea is to exploit properties of the induced hypothesis as a form of representing the dataset itself (Hilario & Kalousis 2000; Peng *et al.* 2002); and *landmarking*, where information obtained from the performance of a set of simple learners (i.e., learning systems with low capacity) is used to identify areas in the input space where each of the simple learners can be regarded as an expert (Pfahinger, Bensusan, & Giraud-Carrier 2000).

Previous work in data characterization exhibits strong limitations. Traditional approaches often denote a sharp distinction between the process of data characterization and the process of mapping datasets to predictive models. Meta-features are crafted without a strong justification that indicates their relevance in explaining differences in learning behavior. In contrast, our goal is to produce new forms of data characterization that can directly be used to explain the connection between example distributions and learning strategies.

A Simple Experiment

Before we present some preliminary ideas for the connection between example distributions and learning strategies, we show results from a simple illustrative experiment. To begin, let us provide some basic 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, $h : \mathcal{X} \rightarrow \mathcal{Y}$.

Now, assume a two dimensional input space where each point $\mathbf{x} \in \mathcal{R}^2$ can be assigned to one of two classes $y \in \{+, -\}$. We assume each class can be modeled as a mixture of two Gaussians. Figure 1 (top-left) shows the class-conditional probability distribution for both classes over the plane. The classes are distributed in an XOR manner, each Gaussian model being diametrically opposed to the Gaussian model that belongs to the same class. Under an optimal Bayesian classification, each class occupies two diametrically opposed squares on the plane.

We trained different learning algorithms on a sample of size two thousand (we generated five hundred examples for each Gaussian model). We then tested each model on a sample of size five thousand drawn uniformly from the bounded plane. Figure 1 shows the error patterns (i.e., the examples misclassified) for each model on our artificial problem. Our learning algorithms include Decision Trees, Per-

ceptron, 1-Nearest-Neighbor, Naive Bayes, and Neural Network. One can conclude that different learning algorithms are characterized by different error patterns over the input space. In our example, the example distribution favors the 1-Nearest-Neighbor approach, as a local approach to classification attains almost optimal performance (provided a sufficiently large training sample). Other algorithms like Naive Bayes, Decision Tree[2], and Perceptron perform close to random guessing. What is important to understand here is the connection between the learning strategy and the particular example distribution. When is a global approach to learning expected to perform better than a local approach? How can we identify this through a summarization of the example distribution? We attempt to answer some of these questions next.

The Case of Naive Bayes

To continue our discussion, we focus on a particular study that involves the Naive Bayes classifier (Vilalta & Rish 2003). Although the behavior of Naive Bayes has been explained from different perspectives (Domingos & Pazzani 1997; Zhang & Ling 2001; Garg & D. 2001; Jaeger 2003), an understanding of the degree of match between different target distributions and the set of assumptions or bias embedded by the algorithm remains unclear. In this section we identify a kind of distributions for which the product approximation of Naive Bayes may result in multiple misclassifications; we name this problem the *class-dispersion problem*.

Specifically, Naive Bayes is a probabilistic classifier that assumes feature independence given the class. For each class y_j, Naive Bayes uses the following discriminant function:

$$g(\mathbf{x}) = P(y_j) \prod_{i=1}^{n} P(a_i|y_j) \quad (1)$$

where $P(y_j)$ is the a priori probability of class y_j and $P(a_i|y_j)$ is the class-conditional probability on a single feature value. Following our research goal we pose the question: what is the effect of the product-approximation component of Naive Bayes on generalization performance under different distributions?

To answer the question above we need to recall that Naive Bayes is a maximum entropy approximating distribution (Lewis 1959). It can be shown that the maximum entropy approximating distribution of Naive Bayes tends to flatten out class distributions over regions of examples for which the set of low order components is identical. This assumption is inappropriate under distributions where clusters of examples that belong to the same class are dispersed throughout the input space (i.e., under the class-dispersion problem). In cases when instances of the same class are scattered (Figure 1 top-left), computing marginals (i.e. single-dimensional projections) of the data may result in significant loss of information. In this case, clusters are hard to identify because a single-dimensional projection of the data loses their spatial information. This is related to the small disjunct problem in classification (Holte & Porter 1989), where the existence of many small disjuncts (i.e., class-uniform clusters covering few examples) may account for a significant amount of the total error rate. Our focus, however, is based on the distribution of clusters rather than their coverage.

[2] A close examination reveals a poor feature discretization is the cause for the low performance of the decision tree.

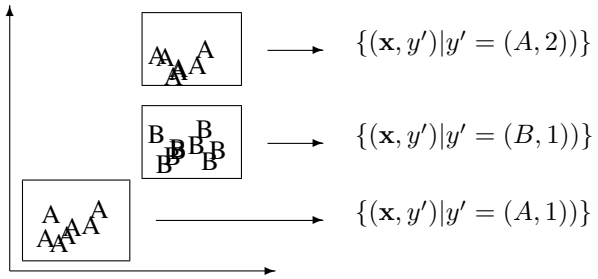

Figure 2: The mapping process relabels examples to encode both class and cluster.

This leads directly to our conjecture that Naive Bayes performs better on domains where the examples of one class are clustered together. We have shown how probability distributions having almost all the probability mass concentrated in one example are well approximated through a product distribution (see (Vilalta & Rish 2003) for theorem and (Rish, Hellerstein, & Jayram 2001) for complete proof).

Decomposing Classes Into Clusters

Our solution to the class-dispersion problem can be summarized through a two-step process:

1. Identify class-uniform clusters of examples in the training set, and
2. Relabel each cluster as a new class of examples.

The new dataset differs from the original training set in the class labeling: there is now an additional number of classes. Naive Bayes is then trained over the new dataset. During classification, performance can be assessed by simply assigning each example back to its original class. A general description of our approach follows.

Let $T = \{(\mathbf{x}, y)\}$ be the input dataset. Our first step is to map T into another dataset T' through a class-decomposition process. 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).

The second step is to train Naive Bayes on T' to obtain hypothesis h'. The hypothesis acts over the transformed output space $h' : \mathcal{X} \rightarrow \mathcal{Y}'$. The classification of a new input vector $\mathbf{x}$ is obtained by applying a function g over $h'(x)$ that will essentially bring the class label back to the original output space, $g : \mathcal{Y}' \rightarrow \mathcal{Y}$.

An illustration of the transformation above is shown in Figure 2. We assume a two-dimensional input space where examples belong to either class A or B. Let's suppose a clustering algorithm separates class A into two clusters, while class B is grouped into one single cluster. The transformation relabels every example to encode class and cluster label. As a result, dataset T' has now three different classes.

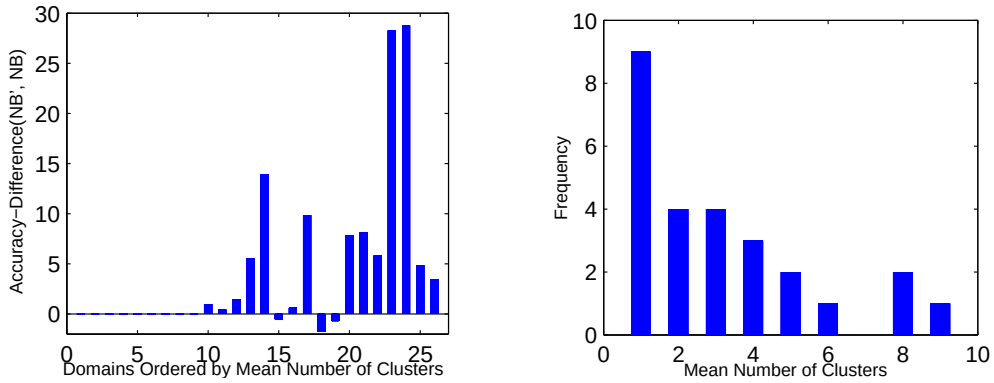

Figure 3: (left) Accuracy difference between Naive Bayes with the transformation and Naive Bayes standard. (right) A histogram of domains based on the mean number of clusters per class.

The decomposition process aims at eliminating the cases where a class spreads out into multiple regions. As each cluster is transformed into a class of its own, the class-dispersion problem vanishes. The result is a new input-output space where each class sits in a tight region. By reducing the class-dispersion problem, the conditional probabilities estimated by Naive Bayes better conform with the assumption of a product distribution (i.e., of a maximum-entropy distribution).

Experiments using the Class-Decomposition

Experiments on real-world domains show how the transformation process improves the accuracy of Naive Bayes, particularly on those domains with many clusters per class. Our datasets (26 domains) can be obtained from the University of California at Irvine repository. Predictive accuracy on each dataset was obtained using stratified 10-fold cross-validation, averaged over 5 repetitions. The clustering algorithm follows the Expectation Maximization (EM) technique (McLachlan & Krishnan 1997); it groups examples into clusters by modeling 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.

Our results show accuracy improvement in most of the datasets used for our experiments. Where no improvement is observed the difference is not statistically significant; in the extreme case where each class is grouped into one single cluster, the performance of our proposed approach is identical to Naive Bayes. In some other domains, the improvement is evident. Figure 3 (left) shows the difference between our approach (NB$'$) and Naive Bayes (NB) (y-axis) where domains are ordered according to the mean number of clusters per class (x-axis). Most significant differences correspond to domains with many clusters per class (we note the increase is not monotonic).

In addition, our results shed some light on the competitiveness of Naive Bayes in real-world domains. Figure 3 (right) shows a histogram of the mean number of clusters per class for each dataset. Most datasets exhibit a distribution characterized by few clusters per class, a situation that favors the assumption behind a product distribution. Few datasets exhibit many clusters per class, which explains why Naive Bayes often appears at the same level of performance as other (more sophisticated) algorithms.

In summary, our study identified a class of input-output distributions where the product-approximation component of Naive Bayes classifier exerts a negative impact on performance. Our data characterization captures the number of clusters per class as an approximation to the degree of class dispersion.

Global and Local Learning

We now try to answer the following question: how can we generalize the results obtained from the study of Naive Bayes to a more general set of classifiers? Giving answer to such question is important to 1) understand poor performance on certain tasks, and 2) propose new approaches to learning that combine the strengths of current learning strategies.

To begin, let us provide a rough categorization of learning algorithms. A criterion to differentiate learning strategies can be related to 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 classified examples.

We focus our study on the differences between global and local learning. 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 flex-

ibility in the decision boundaries lowers the empirical risk (Bottou & Vapnik 1992). 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 (Bottou & Vapnik 1992; Vapnik & Bottou). 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.

Learning Strategies that Adapt to the Example Distribution

We claim one reason explaining poor learning behavior lies on the bias toward strategies that are purely global or purely local. In a pure global strategy the complexity of the model may turn inadequate on certain regions of the input space. In a pure local strategy, too much emphasis is commonly paid to local irregularities in the data.

To test our claim, we have proposed a learning strategy that fits models of different complexity over separate regions of the input space (Vilalta, Achari, & Eick 2004). The idea borrows from the class decomposition process described above. First, locality is achieved by decomposing each class into different clusters, each cluster representing a local characterization of the class-conditional distribution. Instead of modeling this distribution as a mixture of components (Jacobs *et al.* 1991), 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.

Specifically, 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 the class-decomposition process explained before to identify natural clusters in data. Each corresponding cluster represents a local characterization of the class-conditional distribution.

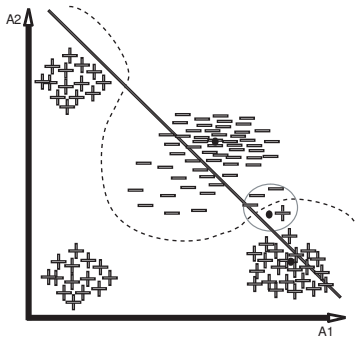

Figure 4: A high-order polynomial (dashed line) improves the classification of a linear model (bold line) at the expense of increased variance. A local classifier (circle) fails to identify cluster membership information.

As an illustration, Figure 4 shows a two dimensional input space with two classes. The distribution of examples precludes a simple linear model attaining good performance (Figure 4 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 4, dashed line). But this comes at the expense of increased variance and possibly data over-fitting.

A different solution is to follow a local learning strategy by tracing a graded circular area around the query example (Figure 4 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 4 (circle) shows a scenario where the query example is assigned class negative, despite it having a higher degree of cluster membership on class positive.

Figure 5 shows an example distribution identical to Figure 4, 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.

Our algorithm (Figure 6) fits models of varying complexity to the relabeled data (i.e., the data obtained after clustering examples of same class and relabeling each cluster as a new class), and selects the function that is most accurate

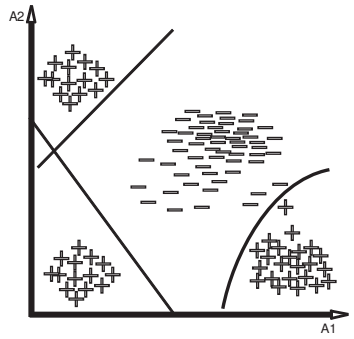

Figure 5: Learning clusters in a piece-wise manner using models of varying complexity; class positive decomposes into three clusters.

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 6: The process to fit models of varying complexity on each of the original clusters.

on a validation set. 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.

Experiments on twenty real-world domains (see (Vilalta, Achari, & Eick 2004) for details) show our proposed strategy outperforming a local learning algorithm (Nearest Neighbor) and performing similarly to a global learning algorithm (Neural Network). Results are consistently above or close to the best of the global or local learning algorithms, which points to an interesting balance between both 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 which facilitates their interpretation.

Summary and Conclusions

We have shown the importance of extracting relevant characteristics about example distributions to understand learning behavior. Our research has simply focused on capturing the number of clusters per class as an approximation to the degree of class dispersion. This was instrumental in explaining the behavior of Naive Bayes. Essentially, the performance of Naive Bayes is expected to degrade as the number of clusters per class increases. We then exploited the class decomposition process to propose a new learning algorithm that fits models of varying complexity to the data.

Future research in meta-learning needs more complex forms of data characterization that can be used to explain learning performance. Previous work has shown how characterizing example distributions is a difficult task in machine learning (Michie 1994; Gama & Brazdil 1995). Most existing measures adopt a general view of the example distribution under analysis; meta-features are obtained by averaging results over the entire training set, implicitly smoothing the actual example distribution (e.g., class-conditional entropy is estimated by projecting all training examples over a single feature dimension). We claim a need exists to attain more specialized descriptors of the example distribution in a form that can be related to learning performance. Our data characterization based on quantifying the number of clusters (or groups of clusters) per class is still a crude approximation to the degree of class dispersion. We need data structures able to characterize properties of the class-conditional data landscape concisely and accurately, in a way that relates to learning behavior.

Acknowledgments

References

Aha, D. W. 1992. Generalizing from case studies: A case study. In *Proceedings of the Ninth International Workshop on Machine Learning*. 1–10.

Bottou, L., and Vapnik, V. 1992. Local Learning Algorithms. *Neural Computation* 4(6):888–900.

Brazdil, P.; Soares, C.; and Pinto da Costa, J. 2003. Ranking learning algorithms: Using ibl and meta-learning on accuracy and time results. *Machine Learning Journal* 50(3):251–277.

Domingos, P., and Pazzani, M. 1997. On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning Journal* 29:103–130.

Engels, R.; Theusinger, C.; Gama, J.; and Brazdil, P. 1998. Using a data metric for offering preprocessing advice in data-mining applications. In *Proceedings of the Thirteenth European Conference on Artificial Intelligence*.

Gama, J., and Brazdil, P. 1995. Characterization of classification algorithms. In *7th Portuguese Conference on Artifical Intelligence, EPIA*, 189–200.

Garg, A., and D., R. 2001. Understanding probabilistic classifiers. In *European Conference on Machine Learning*, 179–191.

Hilario, M., and Kalousis, A. 2000. Building algorithm profiles for prior model selection in knowledge discovery systems. *Engineering Intelligent Systems* 8(2).

Holte, R. C., and Porter, B. W. 1989. Concept learning

and the problem of small disjuncts. In *International Joint Conference on Artificial Intelligence*, 813–824.

Jacobs, R.; Jordan, M. I.; Nowlan, S. J.; and Hinton, G. E. 1991. Adaptive Mixtures of Local Experts. *Neural Computation* 3:79–87.

Jaeger, M. 2003. Probabilistic classifiers and the concepts they recognize. In *Proceedings of the Twentieth International Conference on Machine Learning*.

Lewis, P. 1959. Approximating probability distributions to reduce storage requirements. *Information and Control* 214–225.

McLachlan, G., and Krishnan, T. 1997. *The EM Algorithm and Extensions*. John Wiley and Sons Ellis Horwood.

Michie, D. 1994. *Machine Learning, Neural and Statistical Classification*. Ellis Horwood.

Peng, Y.; Flach, P.; Brazdil, P.; and Soares, C. 2002. Decision tree-based characterization for meta-learning. In *ECML/PKDD'02 Workshop on Integration and Collaboration Aspects of Data Mining, Decision Support and Meta-Learning*, 111–122.

Pfahinger, B.; Bensusan, H.; and Giraud-Carrier, C. 2000. Meta-learning by landmarking various learning algorithms. In *Proceedings of the Seventeenth International Conference on Machine Learning*.

Rendell, L. A., and Cho, H. 1990. Empirical learning as a function of concept character. *Machine Learning Journal* 5(3):267–298.

Rish, I.; Hellerstein, J.; and Jayram, T. 2001. An analysis of naive bayes on low-entropy distributions. In *IBM T.J. Watson Research Center, RC91994*.

Sohn, S. Y. 1999. Meta analysis of classification algorithms for pattern recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21(11):1137–1144.

Vapnik, V., and Bottou, L. Local algorithms for pattern recognition and dependencies estimation. *Neural Computation* 5(6):893–909.

Vilalta, R., and Rish, I. 2003. A decomposition of classes via clustering to explain and improve naive bayes. In *European Conference on Machine Learning*, 444–455.

Vilalta, R.; Achari, M.; and Eick, C. 2004. Piece-wise model fitting using local data patterns. In *Sixteenth European Conference on Artificial Intelligence*.

Vilalta, R.; Oblinger, D.; and Rish, I. 2000. What works well where in inductive learning? In *Workshop: What Works Well Where? as part of the 2000 International Conference on Machine Learning*.

Zhang, H., and Ling, C. 2001. Geometric properties of naive bayes in nominal domains. In *European Conference on Machine Learning*, 588–599.