MDL-BASED ATTRIBUTE MODELS IN NAÏVE BAYES CLASSIFICATION

Esa Elovaara and Petri Myllymäki

Helsinki Institute for Information Technology HIIT
P.O. Box 68, Department of Computer Science
FIN-00014 University of Helsinki, FINLAND
esa.elovaara@helsinki.fi, petri.myllymaki@cs.helsinki.fi

ABSTRACT

When classifying objects with Naïve Bayes classifiers, we are faced with the problem of how to handle continuous attributes. Common solutions to this problem are discretizing, or assuming the data to be normally distributed. In this paper we take a different approach and instead model the class-specific attribute distributions of Naïve Bayes classifiers with MDL-optimal histogram density functions. We present experimental results, comparing MDL-optimal histograms to Gaussian distributions and histograms learned with other methods.

1. INTRODUCTION

Consider a botanist trying to infer the species of a flower from measurements such as length, color and shape of petals. This is the problem of classification. Every object being classified belongs to one out of a finite set of possible classes $\mathcal{C}$. Objects are represented by their attributes $(x_1, \ldots, x_n)$, such as the leaf measurements of the example. Attributes can be numeric or nominal, and numeric attributes can be discrete or continuous. Classifiers can either just give class predictions, or they can estimate the probabilities of objects belonging to certain classes. Classifiers of the latter kind are called statistical classifiers.

One of the most popular statistical classifiers is the Naïve Bayes classifier [1]. Naïve Bayes predicts the probability of an object being in a certain class. To make these predictions, Naïve Bayes needs conditional probability distributions for each of the attributes. The attribute distributions are conditioned on class values and are called class-specific distributions. Class-specific distributions are learned from training data containing object attributes paired with correct class values. Attributes of an object can be discrete or continuous. For a discrete attribute, the distributions are probability mass functions. If an attribute is continuous, probability density functions are needed instead. In this paper we concentrate on learning density functions for the purpose of classifying continuously valued data.

Perhaps the most common approach to learning density functions is to assume that the attribute data is normally distributed [2]. This can be problematic because there is no guarantee that the assumption is justified. Prediction accuracy of Naïve Bayes relies on accurate class-specific density functions, and normal distributions can not mimic e.g. multi-modal or uniform distributions accurately. Hence, a more general form of density functions is needed. Gaussian mixture models [3] and kernel methods [2] are popular choices for modeling complex densities in classifiers, but in this paper we focus on using histograms, which form a simple but versatile class of functions. A histogram is a function that divides its domain into connected intervals and maps all values in a certain interval to a single value. The connected intervals are commonly called bins, and the supremums and infimums of the intervals are called cut points.

Histograms are desirable as density functions because they can approximate all kinds of functions with unlimited precision. No assumptions about the distribution being approximated need to be made prior learning. For example, a normal distribution will not fit multi-modal data well, while a histogram with proper structure will be able to assign a high likelihood to the data. But what is meant by ”proper structure”? The number of bins and bin lengths must be selected in such a way that the ”correct shape” of the data is captured. A histogram with too few or too wide bins can not mimic real characteristics of the data, and too many or too narrow bins misinterpret random noise as significant.

Histograms learned from data offer a unified way of handling various types of attributes. Nominal and discrete attributes can be handled in the same way as continuous attributes. Nominal attributes can be given arbitrary non-overlapping discrete values, and treated like true discrete numeric attributes. The histograms learned from discrete data should have bins of equal width, centered on each unique value in the data, thus essentially reproducing the probability mass functions that would have been used if the attributes were handled as nominal or discrete.

We use the MDL principle [4] to learn histogram-shaped density functions [5] for Naïve Bayes classifiers. These MDL-optimal histograms have proper structure in the sense that they are as complex as our data allows, and their prediction error on future data is worst-case optimal. The method learns both the optimal number of bins and bin lengths.

2. THE NAÏVE BAYES CLASSIFIER

Classifiers are constructed from training data containing attributes of objects paired with correct class values. Classifiers either learn models from the training data, or they can just look for ways to discriminate between different classes. The Naïve Bayes classifier introduced below is a statistical classifier, and it uses joint probabilities to model the training data.

A Bayes classifier is a statistical classifier that uses the Bayes theorem to calculate the probability of a data point $\mathbf{x} = (x_1, \ldots, x_n) \in \mathcal{X}^n$ belonging to some class $c \in \mathcal{C}$:

$$P(c|x_1, \ldots, x_n) = \frac{P(x_1, \ldots, x_n|c)P(c)}{P(x_1, \ldots, x_n)} \qquad (1)$$

The probabilities needed on the right hand side of the equation are estimated from training data, and the classifier's prediction is the class that maximizes (1). Because the denominator does not depend on c, the classification rule can be simplified to

$$c^* = \underset{c}{\text{argmax}}\{P(x_1, \ldots, x_n|c)P(c)\}. \qquad (2)$$

Probabilities of discrete random variables are usually estimated by maximum likelihood estimators counting frequencies of attribute assignments.

$$P(x_1, \ldots, x_n|c) = \frac{\#(\mathcal{X}_1 = x_1, \ldots, \mathcal{X}_n = x_n, \mathcal{C} = c) \in D}{\#(\mathcal{C} = c) \in D}, \qquad (3)$$

$$P(c) = \frac{\#(\mathcal{C} = c) \in D}{|D|}, \qquad (4)$$

where D is training data consisting of pairs of attribute values and class values.

In the case of continuous random variables, probability density functions learned from the data are used instead of probability mass functions. This may seem problematic, because densities are not probabilities and the probability of a specific value of a continuous random variable is zero. A density function f can be used to measure the probability of X being on a certain interval: that probability is $P(x \leq X \leq x+\Delta) = \int_x^{x+\Delta} f(x)dx$. By the definition of the derivative, $\lim_{\Delta \to 0} P(x \leq X \leq x + \Delta)/\Delta = f(x)$. Thus, for some small constant Δ, $P(X = x) = f(x) \cdot \Delta$. The factor Δ then appears in the enumerator of (2) for each class. The factors cancel out when normalization is performed, so density functions may be used in place of probability mass functions.

Although theoretically optimal, the Bayes classifier may not perform well in practice. The reason for this is the joint probability term $P(x_1, \ldots, x_n|C)$. The number of probability values needed to define the joint probability is exponential in relation to n, and an unrealistic amount of training data is needed to get estimates for all probabilities. This relation between the number of variables and the difficulty of a problem is called the *curse of dimensionality* [6]. Let us consider an example situation where X consists of sixteen binary attributes and our training data contains one thousand data points. There are $2^{16} = 65536$ values to define, but our training data only contains information on about 1.5 percent of the values. In this situation it can easily happen that every member of the training set is assigned the same probability, and the classifier just memorizes the training data. The classifier does not generalize to unseen cases similar to those in the training set.

A way to alleviate this weakness is to make assumptions about independence between variables [1]. A joint probability can be factorized using the product rule.

$$\begin{aligned} & P(X_1, \ldots, X_n) \\ = \;& P(X_1|X_2, \ldots, X_n)P(X_2, \ldots, P_n) \\ = \;& \ldots \\ = \;& \prod_{i=1}^{n} P(X_i|X_{i+1}, \ldots, X_n) \end{aligned} \qquad (5)$$

If it is assumed that all variables are independent of each other, the joint probability reduces to

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i). \qquad (6)$$

Given this independence assumption, we can factorize the class-conditional probability term $P(x_1, \ldots, x_n|c)$ of the Bayes classifier (2) and end up with the Naïve Bayes classifier

$$c^* = \underset{c}{\text{argmax}}\{P(c)\prod_{i=1}^{n} P(x_i|c)\}. \qquad (7)$$

The maximum likelihood estimator for the discrete attribute specific class-conditional probabilities is

$$P(x_j|c) = \frac{\#(X_j = x_j, \mathcal{C} = c) \in D}{\#(\mathcal{C} = c) \in D}. \qquad (8)$$

In the case of a continuous attribute, density functions for $P(X_j|c)$ must be learned from data points where $X_j = x_j$ and $\mathcal{C} = c$.

The number of probabilities needed to define a joint probability of mutually independent variables is considerably smaller. The joint probability of sixteen binary attributes is defined by $2 \cdot 16$ probabilities. Because fewer parameters are used, the functions can model fewer and less complex distributions. This is both an advantage and a disadvantage. The advantage is that the classifier is resistant to overfitting because the function can learn all of its parameters accurately from a practical amount of training data. The disadvantage is that the classifier can model only simple distributions.

Even though the Naïve Bayes classifier makes strong assumptions about variable independence, it performs better than expected in classification task even if the assumed independence does not hold. The explanation for this, given by Domingos and Pazzani [1], is that although the classifier will not necessarily estimate the class probabilities correctly, it may assign the highest probability to the correct class.

3. MDL MODEL SELECTION

Minimum description length (MDL) model selection [4] tries to find a model that enables us to describe our observations with the least amount of bits. We must code observations with the help of a model, and add a description of the model to make the whole description decodeable. For a model to be usable in MDL model selection, it must define a probability distribution over all possible observations. Because models are actually sets of distributions, we also need a way to summarize all the distributions of a model with one distribution. For this purpose we use the *normalized maximum likelihood* (NML) distribution [7].

$$\bar{P}_{NML}(\mathbf{x}^n|\mathcal{M}) = \frac{\hat{P}(\mathbf{x}^n|\mathcal{M})}{\int_{x' \in X} \hat{P}(x'|\mathcal{M})dx}, \tag{9}$$

where $\hat{P}(\mathbf{x}^n|\mathcal{M}) = \underset{m\in\mathcal{M}}{\operatorname{argmax}}\{P(\mathbf{x}^n|m)\}$.

The NML distribution's description length, or *stochastic complexity*, for model class $\mathcal{M}$ is

$$SC(\mathbf{x}^n|\mathcal{M}) = -\log \bar{P}_{NML}(\mathbf{x}^n|\mathcal{M}) =$$

$$-\log \hat{P}(\mathbf{x}^n|\mathcal{M}) + \log \int_{\mathbf{x'}^n \in X^n} \hat{P}(\mathbf{x'}^n|\mathcal{M}). \tag{10}$$

The NML distribution is special in the sense that the difference between the description length attained by it and the description length attained by the maximum likelihood distribution from the model it summarizes is minimax-optimal. The description length can be interpreted as a two-part code. The first term is the negative logarithm of the maximum likelihood assigned to the data by the model class. In statistics, this is called log-likelihood, and it is a common measure of a model's capability to fit data. The second term measures the model's complexity and is called *parametric complexity*.

4. LEARNING NAÏVE BAYES CLASSIFIERS

As we could see in the previous chapter, when learning Naive Bayes classifiers, the crucial step is how to estimate the class-conditional attribute distributions (or densities in the continuous case). In the following we first describe two methods for this, based on the MDL principle, and then in Section 4.2 three alternative methods which will be used to validate the usefulness of the MDL-based methods in the empirical part of the paper (Section 5).

4.1. MDL based methods

A brief review of the method for learning MDL-optimal histograms is given here. Refer to [5] for a detailed account. As mentioned earlier in Section 3, a model must define a probability distribution over all possible observations. The method achieves this by dividing the interval where a given histogram is defined to regular subintervals. The probability mass on a given sub-interval will be the probability of all data points located on the sub-interval.

A model $\mathcal{M}$ contains all histograms obtainable with a given set of cut points C. Point hypotheses are indexed by the parameter vector $\theta = (\theta_1, \ldots, \theta_n)$, which defines probability masses of a histogram's bins. Prediscretization of the data is used to simplify the mathematical formulation of parametric complexity because it can be defined as a sum instead of an integral. Prediscretization introduces an additional parameter ϵ, but its effect on stochastic complexity is a constant that can be ignored in the model selection process.

Another consequence of prediscretization is that the set of potential cut points $\mathcal{C}$ is finite:

$$\mathcal{C} = \{\mathbf{x}_{min} + \epsilon/2 + t\epsilon : t = 0, \ldots, \frac{\mathbf{x}_{max} - \mathbf{x}_{min}}{\epsilon} - 1\}, \tag{11}$$

It turns out that parametric complexity for a K-bin histogram for n data points, $\mathcal{R}^n_{h_K}$, is exactly the same as the parametric complexity of a K-valued multinomial [8]. The recursion

$$\mathcal{R}^n_{h_K} = \mathcal{R}^n_{h_{K-1}} + \frac{n}{K-2}\mathcal{R}^n_{h_k-2} \tag{12}$$

holds for $K > 2$. The case $K = 1$ is always 1, and the case $K = 2$ is a simple sum

$$\mathcal{R}^n_{h_2} = \sum_{h_1+h_2=n} \frac{n!}{h_1!h_2!}\left(\frac{h_1}{n}\right)^{h_1}\left(\frac{h_2}{n}\right)^{h_2}, \tag{13}$$

which can be computed in time $\mathcal{O}(n)$. Finally, recursion (12) is applied $K-2$ times to end up with $\mathcal{R}^n_{h_K}$. The time complexity of the whole computation is $\mathcal{O}(n+K)$.

The stochastic complexity of histogram C is

$$SC(\mathbf{x}^n|C) = -\log \frac{\prod_{k=1}^{K}\left(\frac{\epsilon \cdot h_k}{L_k \cdot n}\right)^{h_k}}{\mathcal{R}^n_{h_K}} \tag{14}$$

$$= \sum_{k=1}^{K} -h_k(\log \epsilon \cdot h_k - \log L_k \cdot n) + \mathcal{R}^n_{h_K}, \tag{15}$$

where h_k is the number of data points on the interval of bin k and L_k is the length of the same bin.

An efficient method for computing the first term of stochastic complexity is still needed. This involves finding the optimal set of cut points for the stochastic complexity criterion. A recursion formula can be used to devise a dynamic programming algorithm that achieves this, as described in [5].

The classifier employing MDL-optimal histogram density functions is referred to as *MDLh*. *MDLh* uses the class-conditional densities defined by the histograms as its "probabilities". An alternative approach would be to use probability masses of bins instead, but this was not explored here.

The *MDLg* classifier takes a different approach and discretizes attribute data globally, not separately within each class like *MDLh*. Attributes of the training data are discretized individually by the following method:

1. Learn a MDL-histogram from the attribute's data.

2. Change the value of every data point on the interval of the nth bin of the histogram to n.

A Naïve Bayes classifier for discrete attributes (8) is learned from the discretized training data. The discretizing histograms are included in the classifier and classification is done by the following method:

1. Given a data point $(x_1, \ldots, x_n)$ to classify, use the discretization histograms to derive the discretized values $(\tilde{x}_1, \ldots, \tilde{x}_n)$.

2. Classify using the probability masses $P(\tilde{x}_j|c)$.

The ϵ parameter used in the pre-discretization phase of histogram learning was estimated from data by finding the minimum distance between two non-equal data points, and assigning as value of ϵ one half of this distance.

$$\epsilon := \min \{|x_i - x_j|/2\}, \forall x_i, x_j \quad x_i \neq x_j. \tag{16}$$

No theoretically sound explanation for this method of assignment can be offered here. The method seems to give values that are small enough to capture patterns in the data, without making the search too time consuming.

4.2. Other methods

In addition to the two classifiers using MDL-optimal histograms as its context-specific density functions, three additional classifiers were used to give context to the empirical results. Two of the classifiers use histogram density functions and the third classifier uses the Gaussian density functions.

The *Shimazaki* classifier [9] uses histogram density functions that are, in a sense, optimal with regards to data. Shimazaki histograms are regular, which means that all bins are of equal width. An optimal bin width is learned from data using the following method:

1. Divide the range $[x_{min}, x_{max}]$ into N bins of width δ, and count the number of data points θ_i from all n data points that hit the ith bin.

2. Construct the mean and variance of the number of bin hits, $\bar{\theta} := \frac{1}{N}\sum_i \theta_i$, $v := \frac{1}{N}\sum_i (\theta_i - \bar{\theta})^2$.

3. Compute the cost function, $C_n(\delta) = \frac{2\bar{\theta} - v}{(n\delta)^2}$.

4. Repeat the previous steps while changing the bin size δ to search for $\delta*$ that minimizes $C_n(\delta)$

The *EWB-10* classifier uses ten-bin regular histogram density functions. The range $[x_{min}, x_{max}]$ is divided into ten intervals of equal length.

The probability masses of bins $\theta = (\theta_1, \ldots, \theta_n)$ are estimated using maximum likelihood estimators in all classifiers that employ histogram density functions. The maximum likelihood estimator for the mass of bin j is

$$\theta_j := \frac{n_j}{n}, \tag{17}$$

where n_j is the number of data points inside the bin/interval j and n is the total number of data points. In addition, the distributions are smoothed using the Laplace method, where the number of data points inside every bin were incremented by one.

The *Gaussian* classifier uses Gaussian distributions as its class-specific distributions. Density functions of Gaussian distributions are of the form

$$P(x_j|c) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x_j - \mu_c^j)^2}{2(\sigma_c^j)^2}}. \tag{18}$$

Parameters were learned using maximum likelihood estimators

$$\mu_c^j = \frac{1}{N}\sum_i^N x_i, \tag{19}$$

$$\sigma_c^j = \frac{1}{N-1}\sum_i (x_i - \mu_c^j)^2, \tag{20}$$

where x_i belongs to the set V_j of values of attribute j on data points of class c and $|V_j| = N$.

5. EMPIRICAL RESULTS

5.1. Test setup

Classification tests were run with multiple data sets available from the UCI machine learning repository [10]. Most of these data sets contained a mix of continuous, discrete and nominal attributes. The nominal attributes were given arbitrary non-overlapping numerical values and handled identically with the true numerical attributes. Thirty tenfold cross-validation runs were done with each data set.

Cross-validation is a method for training and testing classifiers in such a way that available data can be used efficiently. When doing n-fold cross-validation the data is randomly divided into n partitions, and the following procedure is repeated for each partition p:

1. Learn a classifier by using the other partitions as a training set.

2. Test the accuracy of the learned classifier with p.

Report the mean accuracy value of the n tests.

Classifier accuracy was measured by percentage of correct classifications and log-sum values. Log-sum is the sum of logarithms of probabilities given to the correct classes. The log-sum measure is useful for selecting classifiers for decision theoretic tasks. For such tasks it is not only important to predict the correct class, but also to estimate class probabilities accurately.

5.2. Test results

The classification percentage results are documented in Figure 1. Standard deviation values were very small for all classifiers and are not documented here. The sets on the x-axis of the figure are ordered in such a way that values for the *EWB-10* classifier increase. Differences between classifiers seem to be higher on sets where *EWB-10* performs poorly. The *MDLg* classifier performs clearly the best. It achieves highest accuracy on most data sets, and it is usually second best when it does not win. The Gaussian classifier performs well on some sets, such as Iris, where the attributes truly are normally distributed. The three histogram density function based classifiers do not perform very well, and differences between them are not large. The *MDLh* classifier dominates the group on sets where *EWB-10* performs poorly. Differences in accuracy diminish on sets where *EWB-10* starts to perform stronger, and mutual ranking between the three classifiers seems to fluctuate randomly.

The log-sum results are documented in Figure 2. Standard deviation values were very small for all classifiers and are not documented here. The sets on the x-axis of the figure are ordered in such a way that values for the *EWB-10* classifier decrease. *MDLg* is clearly the winner again. The ranking of classifiers on set by set basis is close to the ranking on the classification percentage measure. Only the Gaussian classifier has consistently dropped in the rankings. Again, the differences between the classifiers diminish as *EWB-10* starts obtaining better results.

The classifiers based on MDL-optimal histograms suffer from long learning times. It took over twenty four hours to run a cross-validation run with the *MDLh* classifier on the larger data sets. The *MDLg* classifier was even worse, because it had to generate a histogram from a whole attribute, whereas *MDLh* could divide an attribute to separate histograms by class value.

6. CONCLUSIONS

A comparison of several approaches to Naïve Bayes classification of data containing continuous attributes was presented. The purpose of the comparison was to assess the value of using MDL-optimal histograms in Naïve Bayes classifiers.

The straightforward approach of using MDL-optimal histograms as density functions (*MDLh*) was not particularly successful. *MDLh* method attained only marginally better results when compared with a classifier using regular ten-bin histograms. On the other hand, a method based on discretizing data with the help of MDL-optimal histograms (*MDLg*) achieved quite good results. On data sets where the attributes were normally distributed, *MDLg* reached accuracies comparable with a classifier using a model based on normal distributions, and in other cases it performed better. Unfortunately, the computational requirements of learning MDL-optimal histograms are quite high, and could pose a problem for some applications.

7. ACKNOWLEDGMENTS

This work was supported in part by the Academy of Finland under the project ModeST, and by the IST Programme of the European Community under the PASCAL Network of Excellence.

8. REFERENCES

[1] Pedro Domingos and Michael Pazzani, "Beyond independence: Conditions for the optimality of the simple bayesian classifier," in *Machine Learning*. 1996, pp. 105–112, Morgan Kaufmann.

[2] George John and Pat Langley, "Estimating continuous distributions in bayesian classifiers," in *In Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*. 1995, pp. 338–345, Morgan Kaufmann.

[3] Nir Friedman, Moises Goldszmidt, and Thomas J. Lee, "Bayesian network classification with continuous attributes: Getting the best of both discretization and parametric fitting," in *In Proceedings of the International Conference on Machine Learning (ICML)*. 1998, pp. 179–187, Morgan Kaufmann.

[4] Jorma Rissanen, *Information and Complexity in Statistical Modeling*, Springer, New York, NY, USA, 2007.

[5] Petri Kontkanen and Petri Myllymäki, "MDL histogram density estimation," in *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistic*, 2007.

[6] Paul M. Baggenstoss, "The class-specific classifier: Avoiding the curse of dimensionality," *IEEE Aerosp. Electron. Syst. Mag*, vol. 19, pp. 1–2, 2002.

[7] Yu M. Shtarkov, "Universal sequential coding of single messages," *Problems of Information Transmissio*, vol. 23, pp. 3–17, 1987.

[8] P. Kontkanen and P. Myllymäki, "A linear-time algorithm for computing the multinomial stochastic complexity," *Information Processing Letters*, vol. 103, no. 6, pp. 227–233, 2007.

[9] Hideaki Shimazaki and Shigeru Shinomoto, "A method for selecting the bin size of a time histogram," in *Neural Computation*, 2007.

[10] A. Asuncion and D.J. Newman, "UCI machine learning repository," 2007.

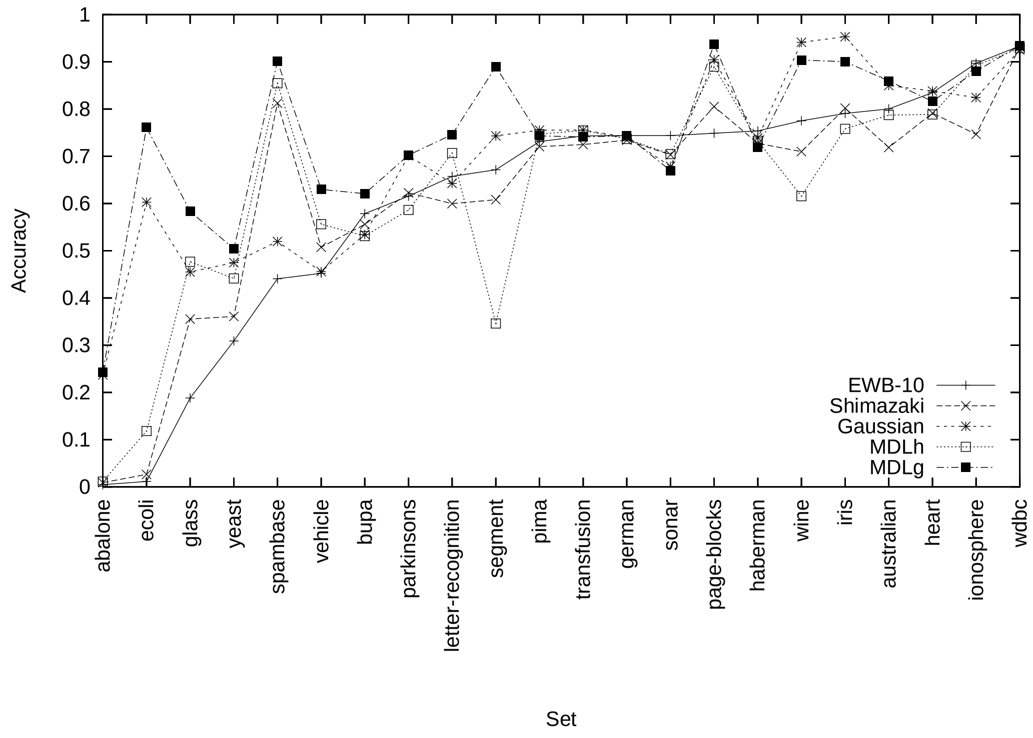

Figure 1. Average classification accuracy of thirty cross-validation runs.

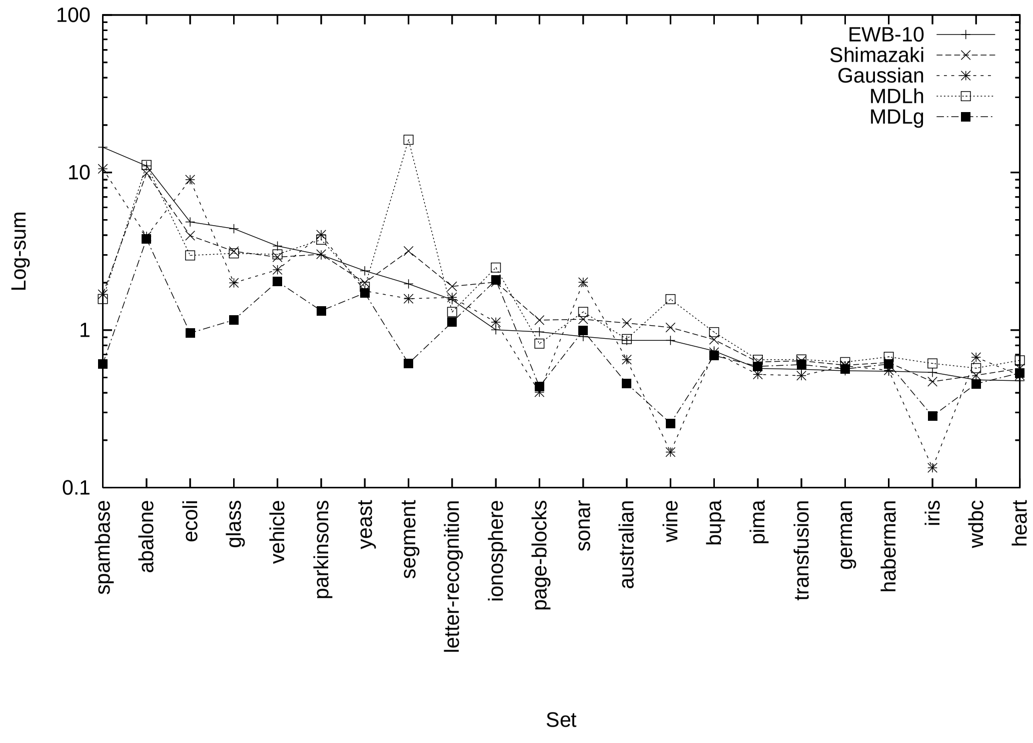

Figure 2. Average log-sum of thirty cross-validation runs.