Kernel Functions for Categorical Variables with Application to Problems in the Life Sciences

Lluís A. BELANCHE [a,1], Marco A. VILLEGAS [a]
[a] *Computer Science School - Department of Software*
Technical University of Catalonia
Jordi Girona, 1-3 08034, Barcelona, SPAIN

Abstract. We introduce a family of positive definite kernels specifically designed for problems described by categorical information. The kernels are based on the comparison of the probability mass function of the variables and have a clear interpretation in terms of similarity computations between the modalities.

We report experimental results on two different problems in the life sciences indicating that the proposed approach may markedly outperform standard kernels, so it can be used as a good alternative to other common kernel functions (at least for SVM classification) in order to obtain better accuracy.

Keywords. kernel functions, categorical variables, support vector machines

Introduction

Support Vector Machines (SVMs) are now standard kernel methods used for tasks such as classification, regression and visualization [1]. When computational considerations are put aside, perhaps the biggest limitation of these methods lies in the choice of a proper kernel for a given problem. There is undoubtedly a growing interest in the introduction of new and potentially useful kernels into the machine learning community. The kernel function is a very flexible container under which to express knowledge about the problem as well as to capture the meaningful relations in input space. Kernel methods involve the use of positive semi-definite matrices as suitable data descriptors, providing a solid framework in which to represent many types of data, as vectors in $\mathbb{R}^d$, strings, trees, graphs, and functional data, among others [2]. Surprisingly, there is a gap in the area of kernel functions specifically devoted to handle datasets with qualitative variables. A *categorical* variable can take some discrete and unordered values, which are commonly known as *modalities*. While some symbolic values are naturally ordered, in many cases no order should be defined, and the only meaningful relation is equality/non-equality.

We propose in this work a different approach grounded on the use of the probabilistic structure of the data and introduce a new family of kernels especially designed for categorical variables. We report experimental results on two different problems in the life

[1] Corresponding Author. E-mail: belanche@lsi.upc.edu

sciences that are naturally characterized by categorical predictors: the first one arises in Microbiology, and the task is to identify the source of fecal pollution in waterbodies out of a number of binary markers; the second one arises in Molecular Biology to classify DNA sequences as corresponding to promoter or non-promoters.

This paper is organized as follows. A light introduction to kernels and categorical information is provided in Section 1, followed by the definition of the categorical kernels (Section 2). We then provide experimental results in Section 3 on a synthetic benchmark as well as in two real-world classification tasks, which show that the proposed kernel improves the performance of state-of-the art kernels. The paper ends with some additional discussion and lines of current and future research.

1. Preliminaries

A kernel function implicitly defines a map $\phi : \mathscr{X} \to \mathscr{H}$ from an input space of objects $\mathscr{X}$ into some Hilbert space $\mathscr{H}$ (called the *feature space*). The "kernel trick" consists in performing the mapping and the inner product simultaneously by defining its associated kernel function [1]:

$$k(\mathbf{x},\mathbf{x}') = \left\langle \phi(\mathbf{x}), \phi(\mathbf{x}') \right\rangle_{\mathscr{H}}, \ \mathbf{x},\mathbf{x}' \in \mathscr{X}, \tag{1}$$

where $\langle \cdot,\cdot \rangle_{\mathscr{H}}$ denotes inner product in $\mathscr{H}$. This way it is possible to perform computations (like distances or inner products) in $\mathscr{H}$ without using (or even knowing) the mapping function explicitly. Probably the simplest characterization for a symmetric function $k : \mathscr{X} \times \mathscr{X} \to \mathbb{R}$ being a kernel is via the matrix it generates on finite subsets:

Definition 1.1 (Positive semi-definite function)**.** *A symmetric function k is positive semi-definite in $\mathscr{X}$ if for every $N \in \mathbb{N}$, and every choice $\boldsymbol{x}_1,\ldots,\boldsymbol{x}_N \in \mathscr{X}$, the matrix $K_{N\times N} = (k_{ij})$, where $k_{ij} = k(\boldsymbol{x}_i,\boldsymbol{x}_j)$, is positive semi-definite (PSD).*

Theorem 1.2 (Characterization of kernels)**.** *A symmetric function $k : \mathscr{X} \times \mathscr{X} \to \mathbb{R}$ admits the existence of a map $\phi : \mathscr{X} \to \mathscr{H}$ such that $\mathscr{H}$ is a Hilbert space and $k(\boldsymbol{x},\boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle_{\mathscr{H}}$ if and only if k is a PSD symmetric function.*

Let k be a kernel defined on the space of objects and consider a dataset $D = \{\mathbf{x}_1,\ldots,\mathbf{x}_N\}$. All the information contained in D is represented as a symmetric positive semi-definite *kernel matrix* $K_{N\times N} = (k_{ij})$, where $k_{ij} = k(\mathbf{x}_i,\mathbf{x}_j)$.

For categorical variables, it is assumed that no partial order exists among the modalities and the only possible comparison is equality. The basic similarity measure (which is a valid kernel) for these variables is the *overlap*. Let x_{ik}, x_{jk} be the modalities taken by two examples $\mathbf{x}_i, \mathbf{x}_j$, then $s(x_{ik},x_{jk}) = \mathbb{I}_{\{x_{ik}=x_{jk}\}}$, where

$$\mathbb{I}_{\{z\}} = \begin{cases} 1 & \text{if } z \text{ is true} \\ 0 & \text{if } z \text{ is false} \end{cases}$$

The similarity between two multivariate categorical vectors is then proportional to the number of variables in which they match. The (multivariate) *overlap kernel* (also

known as the *Dirac kernel*) [1] for d-dimensional vectors of categorical descriptors can be defined as:

Definition 1.3 (Overlap kernel k_0)**.**

$$k_0(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{1}{d} \sum_{i=1}^{d} \mathbb{I}_{\{x_{ik}=x_{jk}\}}$$

Another kernel that can be used is the Gaussian Radial Basis Function (RBF) kernel, known to be a safe default choice for kernel methods working on real vectors [3]:

$$k_{\text{RBF}}(\mathbf{x}, \mathbf{x}') = \exp\left(-\gamma ||\mathbf{x} - \mathbf{x}'||^2\right) , \tag{2}$$

where $\gamma > 0$ is the smoothing parameter, that has to be optimized as part of the training of the classifier (see Section 3.1). In order to use this kernel, categorical variables –say, with m modalities– are coded using a binary expansion representation (also known as a 1-out-of-m code). This coding bears an important advantage, since it allows to use distance-based kernel functions (because the coding is such that all distances between different modalities are equal, and all distances between same modalities are 0). This means, however, that all comparisons between two observations having the same number of coincident values are constant. Note finally that both the Gaussian RBF (Eq. 2) and the overlap/Dirac (Def. 1.3) kernels are known to be positive semi-definite [3].

2. Categorical Kernels

The overlap kernel in Def. 1.3 is mathematically valid but, given its simplicity, it may not capture the non-trivial relations present in the data. However, it will be our starting point to build a probabilistic version of it. In a nutshell, our idea is to express the intuition that, if the modalities being compared coincide, then their similarity is higher the less probable they are. On the other hand, if the modalities do not coincide, the similarity remains zero. Let Z be a categorical variable with Probability Mass Function (PMF) P_Z, and $P_Z(z_i)$ the probability that Z takes the value z_i. Define now the similarity function:

Definition 2.1 (Univariate kernel $k_1^{(U)}$)**.**

$$k_1^{(U)}(z_i, z_j) = \begin{cases} h_\alpha(P_Z(z_i)) & \text{if } z_i = z_j \\ 0 & \text{if } z_i \neq z_j \end{cases}$$

When the values z_i and z_j coincide, we evaluate a new function $h_\alpha(\cdot)$ on the value $P_Z(z_i)$ (which is of course equal to $P_Z(z_j)$). In fact, the function k_0 is a particular case of k_1, in which the function $h_\alpha(\cdot)$ always returns 1.

Thus, the equality between two values is a fact that is more or less important depending on the probability $P_Z(z_i)$: if $P_Z(z_i)$ is relatively low, we are talking about an unusual and possibly relevant event, because it is more difficult to find another similar event z_j. Conversely, if $P_Z(z_i)$ is relatively high, we are in presence of a common event, and therefore any coincidence has a certain lack of interest, inversely proportional to its

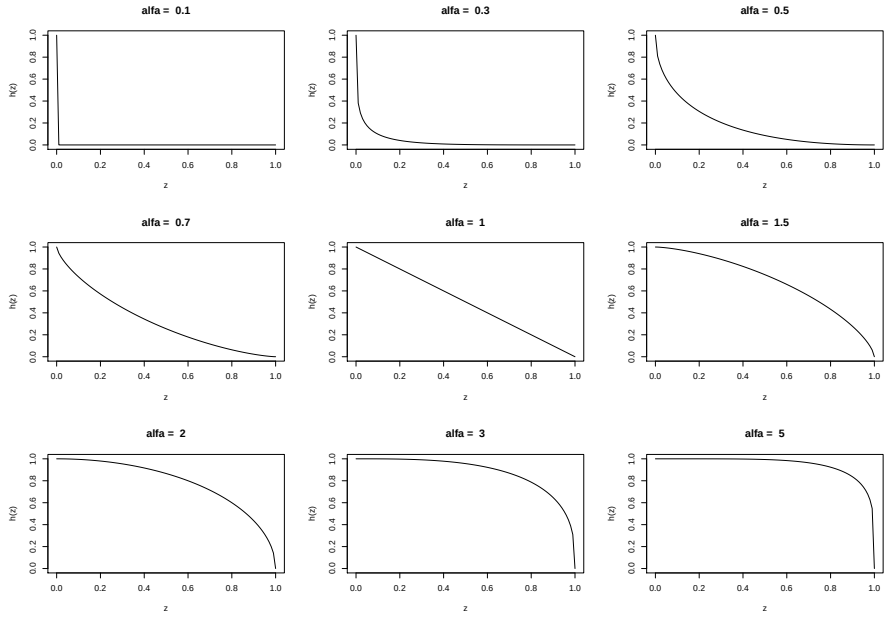

Figure 1. The family of inverting functions $h_\alpha(z)$ for different values of α.

probability. This way of thinking makes sense in many different scopes of human life. For example, people catching a bad cold is a very common situation; if two people go to the hospital with a common cold the same day, this is not a reason to raise alarms, and is not enough to conclude that they are very similar, because everybody can catch a cold (its relative incidence is high). On the other hand, if two people reach the hospital in the same day having a really odd disease, this should become a relevant fact. This coincidence tells us that these two persons *are* similar, or at least have something in common (that caused them to develop the illness). We formulate the hypothesis that, if we have a dataset in which the probabilistic structure is determinant for predicting the response (class) variable, then the new family of kernels should perform better than others not taking this information into account.

These arguments are encoded in the function h_α, which should return a high value when $P_Z(z_i)$ is small, and a low value when it is large:

Definition 2.2 (Inverting function)**.**

$$h_\alpha(z) = (1 - z^\alpha)^{1/\alpha},\ \alpha > 0$$

The function $h_\alpha(\cdot)$ depends on the parameter α which determines its non-linear behaviour[2]. It is involutive, that is, $h_\alpha(h_\alpha(z)) = z$ for all $\alpha > 0$. It is concave for $\alpha \in (0,1]$ and convex for $\alpha \in [1,\infty)$. The reference behaviour corresponds to $\alpha = 1$, in which $h_\alpha(z) = (1-z)$ –see Fig. 1. The range of the function $h_\alpha(P(z))$ is defined on the inter-

[2]This inverting family of functions is known as the Yager family of negations in fuzzy logic [4].

val $(0,1)$, since the argument $P(z) \in (0,1)$ [3] and $\alpha > 0$. These different values for the parameter α give rise to a family of kernel functions based on k_1.

The $k_1^{(U)}$ functions are *univariate* kernels, returning a value concerning a single categorical dimension. In consequence, to perform a multivariate comparison between the d-dimensional data points $\mathbf{x}_i$ and $\mathbf{x}_j$, we need to aggregate the univariate comparisons:

Definition 2.3 (Multivariate kernel k_1)**.**

$$k_1(\boldsymbol{x}_i,\boldsymbol{x}_j) = \exp\left(\frac{\gamma}{d}\sum_{i=1}^{d} k_1^{(U)}(x_{ik},x_{jk})\right),\ \gamma > 0$$

Theorem 2.4. *The kernel matrix generated by the multivariate kernel k_1 in Def. (2.3) is positive semi-definite (PSD).*

PROOF. We shall prove a more general result. Let V a categorical variable and $\mathscr{V} = \{v_1,\ldots,v_m\}$ the set of modalities of this variable. Let $h : \mathscr{V} \to (0,1)$ any function; then $k_h(x,x') = h(x)\mathbb{I}_{\{x=x'\}},\ x,x' \in \mathscr{V}$ is a kernel in $\mathscr{V}$.

Indeed, for any $v \in \mathscr{V}$, define $\phi : \mathscr{V} \longrightarrow [0,1)^m$ as $v \to \phi(v) = (0,\ldots,\sqrt{h(v)},\ldots,0)$ of length m, with the non-zero value at position i when $v = v_i$. Then

$$k_h(x,x') = \sum_{i=1}^{m} \phi_i(x)\phi_i(x')$$

and therefore k_h is a kernel in $\mathscr{V}$. The proof is completed by noting that the sum of a number of kernels is a kernel, multiplication by a positive factor $\frac{\gamma}{d}$ keeps the PSD property, and the exponential of a kernel is also a kernel. ■

Note that there is a close analogy between this categorical kernel and the Gaussian RBF kernel, both being the exponential of the sum of partial (univariate) kernels. An important difference between the Gaussian RBF and the categorical kernel lies in the fact that the latter has a compact support, which means that it can be actually equal to 0, resulting in important computational gains.

3. Experiments

We now turn to the experimental section. We analyse first a synthetic problem (oneMonks) to show that the three kernels considered in the comparison (k_0, k_1 and the RBF kernel with coded categories) behave properly for categorical data from the learning point of view. We then tackle two real problems in the life sciences. The first one arises in Microbiology, where it is known as the *microbial source tracking* problem. The task is to identify the source of fecal pollution in waterbodies using a number of binary markers. The second problem arises in Molecular Biology and consists in building a predictive model to classify DNA sequences as corresponding to promoter or non-promoters. Both

[3] Without loss of generality, we assume that $P(z) \in (0,1)$ excluding the values $\{0,1\}$, because $P(z) = 0$ would represent an impossible event, and $P(z) = 1$ a systematic one (with no randomness).

datasets are characterized by a very low number of samples, which makes overfitting avoidance a delicate undertaking, and probably demands the most of the learning algorithms and their ability to capture the non-trivial and true relations present in the training data. These two problems are naturally formulated as supervised classification problems that can be solved by SVMs.

3.1. Experimental Setup

All the experiments were developed using the freely available R programming language [5]. We selected the `kernlab` package because it offers additional flexibility to accept user-defined kernel functions and works directly with kernel matrices. We have developed efficient code for the computation of the kernel matrices introduced in this work.

For the oneMonks problem, we setup different configurations for training, testing and validation, described below. For the two real problems, the datasets are randomly split taking 2/3 of the data for training and the remaining for testing (these partitions are taken preserving the class proportions). For each experiment, all kernel parameters and the cost (soft-margin) parameter of the SVM were optimized over a grid of possible choices on the training set only, to maximize the mean accuracy over an internal ten-fold cross-validation (10cv). After that, we refit the model on the whole training set but using the optimized parameters. The results shown correspond to the performance of these SVMs on the heldout test sets. This training/test splitting process was repeated 40 times. It should be noted that, with the standard kernel functions, the resulting kernel matrix does not depend on the training/testing data partitions, because the kernel definition $k(x,y)$ only involves the values x and y. However, the calculation of the new kernels k_1 require the knowledge of the PMF for each variable in the dataset. This PMF is estimated using only the training data, because this knowledge influences the resulting kernel matrix and the final performance. Actually, the PMFs are estimated anew for each 10cv fold to make it a honest resampling experiment.

3.2. A Synthetic Problem: the oneMonks

This problem is classic in benchmark classification problems [6]. In its original version, three independent problems are applied to 6 symbolic variables (taking values as "1", "2", etc). Here we have grouped the three problems into a single one, as follows:

- $P_1 : (x_1 = x_2) \vee x_5 = 1$
- P_2 : two or more $x_i = 1$ in $x_1, \ldots, x_6$
- $P_3 : (x_5 = 3 \wedge x_4 = 1) \vee (x_5 \neq 3 \wedge x_2 \neq 2)$

In our oneMonks problem, the boolean condition $P_2 \wedge \neg(P_1 \wedge P_3)$ is then checked. If it is satisfied the class of the observation is 1; otherwise, it is 0. We define four *configurations* by modifying a set of parameters in order to obtain the corresponding dataset –see Table 1 for a summary. The first three columns define the sample sizes used for training, validation and test. The next two columns concern the parameters of the kernels. The last column details the values tested for the cost (soft-margin) C parameter of the SVMs.

Fig. 2 is a summary of the results grouped by the different configurations (`A1, B1, C1, D1`) and kernel type (k_0, k_1 and RBF). The plots show the full distribution of classification results for the different kernel variants, in the form of a boxplot, as well as a sum-

Table 1. Configurations for the oneMonks experiments; $(a : b)$ denotes all integers c such that $a \leq c \leq b$.

	Train	Val.	Test	γ	α	C
A1	20	10	500	$2^{(-3:2)}$	$\{0.1, 0.2, 0.3, 0.5, 0.7, 0.9, 1, 1.5\}$	$10^{(-1:2)}$
B1	40	20	500			
C1	80	40	800			
D1	240	120	800			

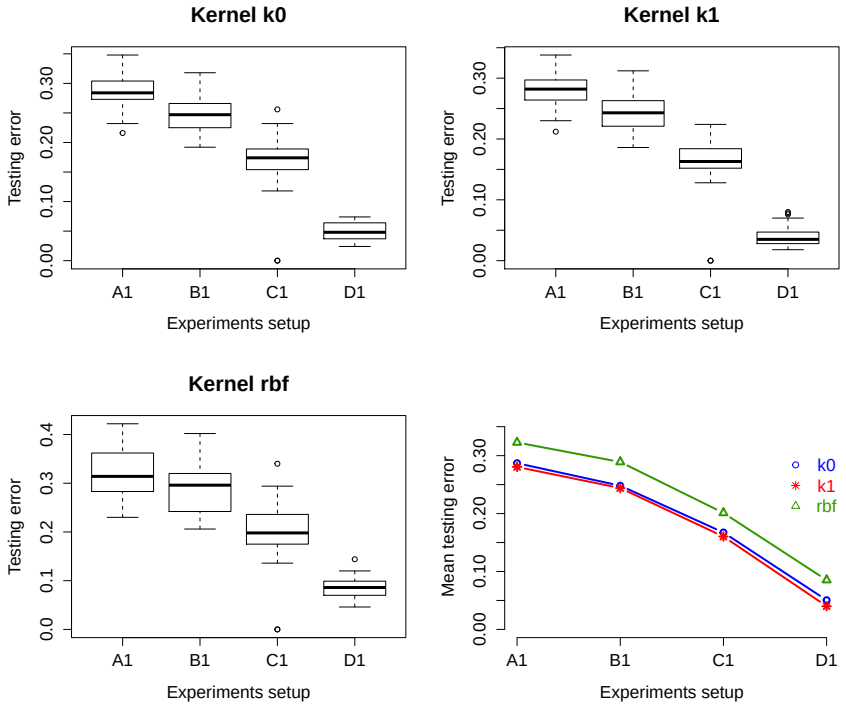

Figure 2. Test error distributions on the oneMonks problems. Bottom right plot is a summary of mean results across configurations.

mary plot of the mean accuracy across the configurations. We can observe how, for all kernels, the testing error is decreased and is also less dispersed as the training set size is increased, as one could reasonably expect. The results also express how the new kernels slightly outperform the standard RBF in mean accuracy across all configurations. Considering the mean error, in experiment `A1` k_0 obtains 0.287, k_1 0.281 and the RBF 0.323. In all configurations, these relative positions are kept, widening a little bit in the more informed experiment `D1`, where k_0 obtains 0.0505, k_1 0.0401 and the RBF 0.0852. If we consider the standard deviation of the errors, both k_0 and k_1 show less dispersion (thus more stability) than the RBF kernel (0.0157 for k_0 and 0.0169 for k_1, against 0.0220 for the RBF kernel in configuration `D1`). Similar results are found in the other configurations.

3.3. Application to the Microbial Source Tracking Problem

The study of fecal source pollution in waterbodies is a major problem in ensuring the welfare of human populations, given its incidence in a variety of diseases, specially in under-developed countries. Microbial source tracking methods attempt to identify the source of contamination, allowing for improved risk analysis and better water management [7]. The available data includes a number of chemical, microbial, and eukaryotic markers of fecal pollution in water. All variables (except the class) are binary, i.e., they signal the presence or absence of a particular marker. The data set includes 9 binary variables, 138 observations and four classes, with 212 missing entries out of $1,242$ (approximately 17%). All variables have percentages of missing entries greater than 15%.

There are many proposals for modelling with missing values (see e.g. [8]). The interest in the present work lies in the probabilistic information that their distribution may bear. Therefore, in order to cope with the missing values a simple yet interesting approach is to treat them as a further modality. If the presence of a missing value says something about the class of the observation (because there is a pattern of missingness), then there is a possibility that models show increased predictive performance. We therefore model the predictors as categorical variables with *three modalities*: positive, negative and missing (hence there are no true missing values in the data). At the very worst, the use of the full dataset (with the information supplied by the 138 observations), will enable to draw more significant results due to the increased sample size. It is worth to recall that the data set is quite small, taking into account that we deal with a four-class problem.

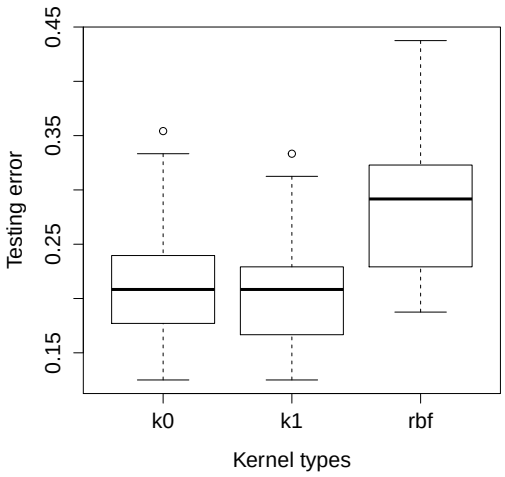

Figure 3. Test error distributions on the microbial source tracking problem.

The results are summarized in Fig. 3. The kernels k_0 and k_1 are quite similar in accuracy (the median values are actually equal) and both are better than the RBF kernel. However, the k_1 kernel is preferable to k_0. This is apparent in the first quartile of the distribution of test errors (which is lower for k_1, thus showing a better downwards ten-

dency); moreover, the distribution of test errors for k_1 is more stable (standard deviation of 0.0517 against 0.0579 for k_1 and 0.0747 for the RBF kernel).

A recent study using this same data set investigated the development of predictive models using all the available data and categorical predictors (exactly as in this paper) and also using the complete data only (78 observations out of 138, thus loosing 43% of the dataset) [9]. The best result was achieved by a Naïve Bayes classifier, yielding a leave-one-out (and probably optimistic) prediction error of 22.1% with the categorical variables. Our results are 21.7% for the SVM with the k_0 kernel and 20.7% for the SVM with the k_1 kernel. Using the complete data only, the best result was achieved by a LDA classifier, yielding a leave-one-out prediction error of 20.5%.

3.4. Application to the Promoter Gene Problem

The *PromoterGene* dataset contains DNA sequences of promoter and non-promoters, which it is available from the UCI repository [10]. Transcription factors are the proteins that mediate transcriptional regulation and bind to specific short DNA sequences called binding sites (BS). These BSs are mainly located in a region upstream to the regulated gene, called a *promoter*. Promoters have a region where a protein (RNA polymerase) must make contact and their helical DNA sequence must have a valid conformation so that the two pieces of the contact region spatially align. The dataset consists of 106 observations and 57 categorical variables describing the DNA sequence, represented as the nucleotide at each position: [A] adenine, [C] cytosine, [G] guanine, [T] thymine. The response is a binary class: "+" for a promoter gene and "−" for a non-promoter gene.

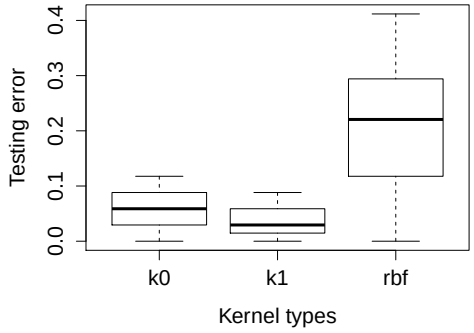

Figure 4. Test error distributions on the PromoterGene problem.

The results are summarized in Fig. 4. The performance of k_0 and k_1 clearly outperform the one obtained by the RBF kernel. In fact, the plot presents strong evidence that k_1 (with a mean test error of 0.0382) solves this problem markedly better than both the RBF kernel (0.2125) or k_0 (0.0618). Additionally, the kernel k_1 also shows a lower dispersion than all other kernels, having a standard deviation of 0.0299, while the RBF has 0.1119 and k_0 has 0.0364.

This is a problem for which there is a probalistic structure linked to motifs. The protein structure of the binding region of a transcription factor dictates the length and the base composition of the sequences it can bind. The BS sequence pattern is called a *motif*. A common way of representing a motif is by aligning the BS sequences and then building a profile matrix, which holds the frequencies of each nucleotide at each position.

4. Conclusions

This paper has dealt with the development of new kernels for categorical variables. The results confirm the relevance of probabilistic information for the problem of learning with categorical predictors, that motivated this work. Specifically, we note that on the Promoter Gene problem (the one for which the assumption of a probabilistic structure seems more tenable), the categorical kernel outperforms the others by a wider margin, confirming the competitiveness of our method. This means, first, that there is an information that is not used by any of k_0 or the RBF kernel and second, that there are ways of profiting this information. In particular, where the results (mean errors and their variance) in standard kernels are already good, k_1 improves a little with respect to k_0, and both of them outperform the RBF kernel. The poor performance exhibited by the latter kernel may be explained in that it easily overfits the small-sample data in the high-dimensional representation where it works. Where the results have still margin of improvement (as in the PromoterGene dataset), then the benefit of using k_1 is higher.

Our aim is to develop kernel functions for categorical variables that admit an interpretation in terms of a similarity, that are provably valid kernels, capture some intuitive semantics and extract non-trivial information from the data (in the sense that are not reduced to equality comparisons). Future work will consider problems characterized by *mixtures* of continuous and categorical variables, even with different sets and numbers of modalities (given our construction of the kernel, this is quite straightforward), and devise truly multivariate kernels (not build by simply aggregating the univariate comparisons) that are able to capture higher-order relations.

References

[1] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge Univ. Press, 2004.

[2] B. Schölkopf, K. Tsuda, and J.P. Vert (Eds.) *Kernel Methods in Computational Biology*. MIT Press, 2004.

[3] B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, 2002.

[4] G. J. Klir and B. Yuan: *Fuzzy Sets and Fuzzy Logic. Theory and Applications*. Prentice Hall, 1995.

[5] R Core Team (2013). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org/.

[6] S. B. Thrun et al (24 authors). *The monks problems: a performance comparison of different learning algorithms*. CMU-CS-91-197 Tech. Rep., 1991.

[7] T.M. Scott, J.B. Rose, T.M. Jenkins, S.R. Farrah and J. Lukasik. Microbial source tracking: Current methodology and future directions. *Applied and Environmental Microbiology*, 68(12):5796–5803, 2002.

[8] R. Little and D. Rubin. *Statistical Analysis with Missing Data*. Wiley-Interscience, 2009.

[9] E. Ballesté, X. Bonjoch, Ll. Belanche and A. R. Blanch. Molecular indicators used in the development of predictive models for microbial source tracking. *Applied and Environmental Microbiology*, 76(6):1789–1795, 2010.

[10] A. Frank and A. Asuncion. UCI machine learning repository, 2010. http://archive.ics.uci.edu/ml/.