International Journal of Pattern Recognition and Artificial Intelligence

HOW FAR DO WE GET USING MACHINE LEARNING BLACK-BOXES?

ANDERSON ROCHA
Inst. of Computing, Univ. of Campinas (UNICAMP)
Av. Albert Einstein, 1251 – Cidade Universitária,
13083-852, Campinas, SP, Brazil
anderson.rocha@ic.unicamp.br

JOÃO PAULO PAPA
Dept. of Computer Science, UNESP – Univ. Estadual Paulista
Av. Eng. Luiz Edmundo Carrijo Coube, 14-01
17033-360, Bauru, SP, Brazil
papa@fc.unesp.br

LUIS A. A. MEIRA
Faculty of Technology, Univ. of Campinas (UNICAMP)
R. Paschoal Marmo, 1888 – Jd. Nova Itália
13484-332, , Limeira, SP, Brazil
augustomeira@gmail.com

With several good research groups actively working in machine learning (ML) approaches, we have now the concept of self-containing machine learning solutions that oftentimes work out-of-the-box leading to the concept of ML *black-boxes*. Although it is important to have such black-boxes helping researchers to deal with several problems nowadays, it comes with an inherent problem increasingly more evident: we have observed that researchers and students are progressively relying on ML black-boxes and, usually, achieving results without knowing the machinery of the classifiers. In this regard, this paper discusses the use of machine learning black-boxes and poses the question of how far we can get using these out-of-the-box solutions instead of going deeper into the machinery of the classifiers. The paper focuses on three aspects of classifiers: (1) the way they compare examples in the feature space; (2) the impact of using features with variable dimensionality; and (3) the impact of using binary classifiers to solve a multi-class problem. We show how knowledge about the classifier's machinery can improve the results way beyond out-of-the-box machine learning solutions.

Keywords: Machine Learning Black-Boxes; Binary to Multi-class Classifiers; Support Vector Machines; Optimum-Path Forest; Visual Words; K-Nearest Neighbors.

1. Introduction

In the digital age, information reaches us at remarkable speed and the amount of data it brings is unprecedented. In the hope of understanding such flood of information, data mining and machine learning approaches are required and a common

problem we often face is data classification. To this task, there are several approaches in the literature and one of the problems they need to address in almost every situation is how to take advantage of known information for inference over unseen data examples.

Different approaches have been proposed to deal with classification problems. Solutions range from supervised learning ones which aims at developing techniques able to take advantage from labeled training samples to make decisions over unseen examples[1,2], to unsupervised ones in which label information is not used[3,4,5]. In between these solutions, there are semi-supervised approaches designed to learn from both labeled and unlabeled data[6,7].

Recently, several tools have been presented in order to perform machine learning (ML) in a more straightforward and transparent manner helping researchers to solve hard classification problems. This increasing activity is also true for supervised learning methods with several methods being proposed in the last two decades[1]. With several good research groups actively working in ML approaches, there is now the concept of self-containing machine learning solutions that oftentimes work out-of-the-box leading to the concept of ML *black-boxes*. By black-box we mean the researchers using well-known ML libraries and tools which, theoretically, come 'ready-to-use'. When using an ML black-box, researchers frequently do not need to worry about implementation details regarding the classifiers neither about some confusing parameters needed to tune the classification training process.

There are several machine learning black-boxes implemented in software such as: Weka[a], R[b], SVM Light[c], LibSVM[d], LibOPF[e], Matlab PRTools[f], among others.

Although it is important to have such black-boxes helping researchers to deal with several problems nowadays, it comes with an inherent problem increasingly more evident. After reviewing several papers across multi-disciplinary fields in which the authors reported results involving machine learning solutions we observed that the authors often misunderstand either the problem at hand or the employed ML tools to solve such a problem. From this, we have observed that researchers and students are progressively relying on ML black-boxes and, usually, achieving results without knowing the machinery of the classifiers. Such decision sometimes just mean the researchers can not explain their results but also mean they are achieving one

[a] *http://www.cs.waikato.ac.nz/ml/weka/*

[b] *http://www.r-project.org/*

[c] *http://svmlight.joachims.org/*

[d] *http://www.csie.ntu.edu.tw/∼cjlin/libsvm/*

[e] *http://www.ic.unicamp.br/∼afalcao/libopf/*

[f] *http://www.prtools.org/*

result that could be more effective if they had knowledge about some details about the classifiers.

The questions we raise in this paper are somehow in line with the ones raised by Andel and Yasinsac in their work[8]. However, in that work, the authors dealt with validation issues in a different area: simulation in mobile ad hoc networks (*manets*). According to the authors, "Simulation is a powerful tool, but it's fraught with potential pitfalls. We question this approach's validity and show how it can systematically produce misleading results".

The main contribution of this paper is to discuss and present experiments, and results related to some key choices one has to think of before using ML-based solutions. In this sense, the paper discusses the use of machine learning black-boxes and poses the question of how far researchers can get using these out-of-the-box solutions instead of going deeper into the machinery of the classifiers. We focus on three aspects of classifiers:

(1) the way they compare examples in the feature space;
(2) the impact of using features with variable dimensionality;
(3) the impact of using binary classifiers to solve a multi-class problem.

Knowledge about the classifier machinery can improve the results way beyond out-of-the-box ML solutions. For instance, with the same descriptor and the same classifier, sometimes it is possible to reduce the classification error in more than 50% by just providing the classifier with a proper comparison (dis)similarity function. In other situations, when using an intrinsic two-class classifier such as Support Vector Machines (SVMs), it might be required knowledge about the implemented combination mechanisms used to perform N-Way classification. The proper choice of the combination mechanism can not only save the researchers training computational time but also improve the classification results.

To validate our observations, we show experiments and results using a series of well known data sets, feature descriptors, and classifiers. The researchers can easily see how the classifiers behave in each scenario and how they behave when changing a few parameters regarding the policy for comparing examples, choices for N-Way classification and also the correct means for dealing with features with varying dimensionality. We believe the questions we raise in this paper may help researchers across multi-disciplinary fields when thinking about using ML approaches to solve their research problems.

The remaining of this paper is organized as follows. Section 2 presents background information with respect to machine learning and image classification. Section 3 brings the experiments that corroborate the paper hypothesis: a few choices of parameters in the classifiers' machinery can heavily influence the classification outcomes. Finally, Section 4 closes this paper and discusses additional perspectives to look at when using machine learning approaches to solve problems.

2. Background

This section presents some concepts for the general understanding of the paper.

2.1. *Machine Learning*

Machine learning techniques attempt to learn the behavior of data samples in order to fit decision boundaries that generalize over unseen data. Depending on the amount of information we have about training samples, we can classify such techniques in supervised, semi-supervised and unsupervised ones.

Supervised learning approaches are those that aim at estimating a classification function f from a *training data set*. The commonest output of the function f is a class indicator of an input object. The learning task is to predict the function outcome of any valid input object after having seen a sufficient number of training examples. In the literature, there are many different approaches for supervised learning such as Linear Discriminant Analysis (LDA), Support Vector Machines, Classification Trees, Neural Networks (NNs), Ensembles of Classifiers (Bagging and Boosting), K-Nearest Neighbors (KNN), Optimum-Path Forest (OPF), and others[1].

Unsupervised learning approaches are those in which no label information is available. Often, these methods seek to determine how the data are organized. Unsupervised learning relates to the problem of density estimation in statistics and also encompasses many other techniques designed to summarize key features. In the literature, there are many different approaches for unsupervised learning such as Self-Organizing Maps (SOM), Adaptive Resonance Theory (ART), K-Means, K-Medoids, Density-Based Partitioning, among others[9]. In between supervised and unsupervised solutions, there are semi-supervised approaches designed to learn from both labeled and unlabeled data[6,7].

Regardless the kind of solution used, to perform the intended tasks most of these approaches have to compute relationships between pairs of concepts. This need leads to the pursuing of a proper feature space in which such computations can be deployed. Often, these computations are performed upon feature descriptors aimed at summarizing the underlying data they represent.

In this paper, we discuss that the choice of the comparison policy between concepts has a major impact on the classification outcome. The main problem is that some researchers do not have this in mind when using machine learning black-boxes ultimately relying on their built-in comparison policies. Notwithstanding, such built-in comparison policies sometimes are not appropriate to the set of descriptors at hand.

2.2. *Feature Description*

In data classification, there is an input example to classify, the feature vector characterizing it, and a (dis)similarity function to compare this concept to other valid ones. The (dis)similarity measure is a matching function which gives the degree of (dis)similarity for a given pair of concepts as represented by their feature vectors[10].

Formally, an example herein called *stream* is a sequence of elements of an arbitrary type (e.g., bits, characters, images, etc.). A stream is a sequence whose codomain is a nonempty set[10]. Representing such stream in the image processing domain, an image stream (or simply image) $\hat{I}$ is a pair $(D_I, \vec{I})$, where:

- D_I is a finite set of coordinates or points in $\mathbb{N}^2$ ($D_I \subset \mathbb{N}^2$);
- $\vec{I} : D_I \rightarrow \mathbb{R}^n$ is a function that assigns to each coordinate p in D_I a vector $\vec{I}(p) \in \mathbb{R}^n$ (e.g., $\vec{I}(p) \in \mathbb{R}^3$ when a color in the RGB or HSV system is assigned to a coordinate).

We can model a feature vector $\vec{v}_{\hat{I}}$ of an image $\hat{I}$ as a point in an $\mathbb{R}^m$ space such that $\vec{v}_{\hat{I}} = (v_1, v_2, \ldots, v_m)$, where m is the number of dimensions (dimensionality) of the feature vector.

Proper descriptors to characterize the underlying data under analysis vary from one application to another. In the image processing domain, we can list some well-known ones such as: color histograms, multi-scale fractals, Fourier coefficients, texture descriptors, among others[11]. Essentially, an image descriptor seeks to encode the maximum image properties as possible (e.g., color, texture, shape, silhouette, etc.) in order to capture the underlying concepts they strive for summarizing[10].

We may think of an image content descriptor D as a pair (f_D, δ_D), such that

- $f_D : \hat{I} \rightarrow \mathbb{R}^m$ is a function, which extracts a feature vector $\vec{v}_{\hat{I}}$ from an image $\hat{I}$;
- $\delta_D : \mathbb{R}^m \times \mathbb{R}^m \rightarrow \mathbb{R}$ is a (dis)similarity function (e.g., based on a distance metric) that computes the similarity between two concepts as the inverse of the distance between their corresponding feature vectors.

After describing images, we can compare them using the (dis)similarity function δ_D to determine how different they are.

The whole argument of this paper is that as machine learning black-boxes are becoming increasingly available to general-purpose uses, researchers are forgetting to think of their problems under the appropriate point of view. We would like to stress that it is not difficult to find researchers that feed out-of-the-box classification solutions with such feature vectors without worrying about the underlying (dis)similarity function implemented. Sometimes, the outcome requirements are already met, for instance, when the machine learning black-box built-in (dis)similarity function represents a good comparison metric for the supplied feature vector. However, we argue that, sometimes, the researchers could have more accurate results by understanding the nature of their feature vector function extraction and its underlying appropriate (dis)similarity function.

In this paper, we have used two different types of descriptors: global- and local-based. Generally, global descriptors attempt to represent the image as a whole, resulting in a feature vector with the same number of features for any image. On the other hand, local descriptors often result a different number of features per image hardening the use of classifiers. With respect to global descriptors, in this paper we have used Moment Invariants[12,13], Beam Angle Statistics (BAS)[14], Tensor Scale

Descriptor (TSD)[15,16], Multiscale Fractal Dimension (MFD)[17], Fourier Descriptor (FD)[18] and Border-Interior Descriptor (BIC)[19]. With respect to local descriptors, we have used SIFT[20], which is one of the most robust methods under translations, scale and rotations transformations to date.

(Dis)similarity functions

To compare feature vectors, we need to use (dis)similarity functions, which attempt to tackle two distinct situations: in the first one, we have features with the same size while in the second, we may face samples represented by features with variable dimensionality.

With respect to feature vectors with the same dimensionality, we can use pointwise (dis)similarity functions, such as Manhattan, Euclidean, Canberra[21] and dLOG[19].

Basically, there are two options to deal with feature vectors with variable dimensionality. The first option is to use an appropriate (dis)similarity function which calculates the (dis)similarity between the point clouds of a pair of images considering the feature points as distributions, and calculating the differences between the two distributions. We can implement this approach using the Earth Mover's Distance (EMD)[22] or the Integrated Region Matching (IRM)[23] (dis)similarity functions. The second option is to preserve the distinctiveness power of such descriptors while increasing their generalization and ease of handling with standard machine learning classifiers. This is possible, for instance, by using the concept of visual vocabularies[24,25].

Visual Vocabularies

A visual vocabulary or dictionary-based approach consists of three stages:

- **PoI Localization**. This stage finds the points of interest (PoIs) of all the available training images;
- **Words Selection**. This stage selects (e.g., via clustering) the k most representative PoIs within all the calculated ones;
- **Quantization**. The k words represent the visual dictionary onto which all other points must be be mapped or quantized to.

To create a visual dictionary, we can perform clustering such as k-means[1] for finding representative centers for the cloud of PoIs or randomly select k PoIs of potential interest. After the quantization, each image results in a quantized feature vector (histogram) with k dimensions. At this point, any machine learning classifier or (dis)similarity function can be used given that the initial PoIs are now mapped onto a meaningful (although potentially sparse) space.

For testing, the input image is characterized and all of its PoIs are mapped onto the visual dictionary resulting in a feature vector also with k dimensions. This

feature vector is fed to the trained classifier to produce the final answer.

In this paper, we show the effects of choosing direct comparisons (such as when using EMD or IRM) with respect to the use of visual dictionaries. Each of these choices comes with a price which we shall discuss in Section 3.

2.3. *Multi-class Mechanism*

Choosing the right descriptor and the suitable comparison policy for the available examples is just the beginning. Sometimes, the use of naturally multi-class classifiers is not possible. For instance, although many statistical classification techniques do have natural multi-class extensions, some, such as the Support Vector Machines, do not. As SVMs are powerful two-class classifiers, many researchers have studied approaches for mapping multi-class problems onto a set of simpler binary classification problems (class binarization). In general, we can classify such approaches into three broad groups[26] as we describe bellow. In the following, let N_T be the number of binary classifiers used to solve a multi-class problem with N classes.

(1) **One-vs-All (OVA)**. This approach uses $N_T = N = O(N)$ binary classifiers. We train the i^{th} classifier using all patterns of class i as positive $(+1)$ examples and the remaining class patterns as negative (-1) examples[27,28].
(2) **One-vs-One (OVO)**. This approach uses $N_T = \binom{N}{2} = O(N^2)$ binary classifiers. We train the ij^{th} classifier using all patterns of class i as positive and all patterns of class j as negative examples.
(3) **Error Correcting Output Codes (ECOC)**. This approach uses a coding matrix $M \in \{-1, 0, 1\}^{N \times N_T}$ that induces a partition of the classes into two meta-classes[29,30]. Passerini et al.[31] extended this approach introducing a decoding function that combines the margins through an estimation of their class conditional probabilities assuming that all base learners are independent. Rocha and Goldenstein[32] extended Passerini el al.'s approach relaxing the base learners independence constraint and introducing the concept of high-correlated groups of binary classifiers.

Most of the current available machine learning black-boxes have some class binarization approaches implemented by default and the choice of the right one is an important research problem.

3. Experiments and Discussion

This section shows the experiments we perform to validate the hypotheses in this paper. We show that the choice of the comparison policy in the classifiers' machinery influences the classification outcomes and, furthermore, must be tackled when dealing with ML problems.

We also analyze categorization problems using features with variable dimensionality. The experiments compare the use of a direct (dis)similarity function to

the mapping of the features onto a meaningful space. After mapping the features in $\mathbb{R}^k$, we are able to use a more standard and faster (dis)similarity function.

Finally, we discuss what happens if a researcher decides to solve a given N-Wayclassification problem with combinations of binary classifiers. The SVM classifier, for instance, uses combinations of binary classifiers to produce the N-Way classification results. We show the impacts on the final classification results when changing the multi-class policies regarding the choice of binary classifiers involved and their final decision-making mechanisms.

We organize the experiments in three rounds:

(1) In the first round, we assess the impacts of choosing the comparison policy in the classifiers' machinery. We evaluate a series of shape descriptors over the MPEG-7 CE Shape-1 Part-B data set (c.f., Sec. 3.1:1). We also evaluate a well known color image descriptor over the Corel color images RRSets data set (c.f., Sec. 3.1:2);
(2) In the second round, we evaluate how to solve classification problems using features with variable dimensionality. We perform experiments using the Corel color images RRSets data set (c.f., Sec. 3.1:2) and the Darmstadt ETH data set (c.f., Sec. 3.1:3);
(3) In the third and final round of experiments, we evaluate the impacts on the final classification results when using combinations of binary classifiers to solve multi-class classification problems. We evaluate SVM, the most famous classifier that uses combinations of binary classifiers to perform N-Way classification. We perform experiments using the MNist data set (c.f., Sec. 3.1:4) and the Australian Sign Language (Auslan) data set (c.f., Sec. 3.1:5).

3.1. *Data sets*

To corroborate the hypothesis we discuss in this paper, we have used a series of image descriptors and image data sets. This section presents some details about the data sets we have used and how they can be freely obtained through the Internet:

(1) **MPEG-7**. MPEG-7 CE Shape-1 Part-B data set includes 1,400 shape samples, 20 for each class encompassing 70 classes. The shape classes are very distinct, but the data set shows substantial within-class variations[g];
(2) **Corel Relevants**. This data set comprises 1,624 images from Corel Photo Gallery[19]. The collection contains 50 color image categories and is referred to as the Corel Relevant sets (RRSets)[h];
(3) **Darmstadt ETH**. This data set comprises 3,280 images of objects grouped into 8 equally-sized classes. The images portrait just one object each time and

[g] *http://www.cis.temple.edu/∼latecki/TestData/mpeg7shapeB.tar.gz*

[h] *http://webdocs.cs.ualberta.ca/∼mn/BIC/*

vary in pose and point of view[i];

(4) **MNIST**. MNist digits test data set comprises 10,000 digits divided into 10 classes. Each digit is described with 785 pre-computed features distributed along with the data set[j];

(5) **Auslan**. This data set comprises 2,565 australian sign language documents divided into 95 classes. Each document is described with 22 pre-computed features distributed along with the data set[k].

3.2. *Quality Assessment*

The accuracy *Acc* of each classifier we report in this paper is described bellow. Let Z_i, $i = 1, 2, \ldots, c$, be the number of samples in the test set Z for each class i. We define

$$e_{i,1} = \frac{FP(i)}{|Z| - Z_i} \quad \text{and} \quad e_{i,2} = \frac{FN(i)}{Z_i}, \; i = 1, \ldots, c \tag{1}$$

where $FP(i)$ and $FN(i)$ are the false positives and false negatives, respectively. That is, $FP(i)$ is the number of samples from other classes that were classified as being from the class i in Z, and $FN(i)$ is the number of samples from the class i that were incorrectly classified as being from another class in Z. The errors $e_{i,1}$ and $e_{i,2}$ are used to define

$$E(i) = e_{i,1} + e_{i,2}, \tag{2}$$

where $E(i)$ is the partial sum error of class i. Finally, the accuracy *Acc* is defined as

$$Acc = \frac{2c - \sum_{i=1}^{c} E(i)}{2c} = 1 - \frac{\sum_{i=1}^{c} E(i)}{2c}. \tag{3}$$

The accuracy *Acc* has a good behavior even for classes of very distinct size, as reported in[33].

For all the experiments we perform in this paper, we have used a 10-fold cross-validation procedure to assess how the results generalize to an independent data set[1]. All the experiments were carried out on an Intel Xeon X5670 computer with 2.93 GHz and 12Gb of RAM. Since most of ML black-boxes do some fine-tuning by default with no user warning, we have decided to use the training sets to tune SVM parameters (e.g., gamma/sigma, C etc.). We thus provide results (for the test sets) considering the best parameters found in the training sets.

[i] *http://www.mis.informatik.tu-darmstadt.de/Research/Projects/eth80-db.html or http://people.csail.mit.edu/jjl/libpmk/samples/eth.html*

[j] *http://yann.lecun.com/exdb/mnist/*

[k] *http://archive.ics.uci.edu/ml/datasets/Australian+Sign+Language+signs+(High+Quality)*

3.3. *Experiments – Round I*

3.3.1. *Part A*

The Part A of the first round of experiments explores the MPEG-7 CE Shape-1 Part-B data set to verify the hypothesis on the importance of the (dis)similarity function used in the classification. In this experiment, we have considered the classifiers SVM with a Linear, Sigmoid, and RBF kernels (from LibSVM) as well as an SVM with no kernel (LibSVM Linear). In addition, we have tested ANN and SOM classifiers as well as KNN and OPF (LibOPF) with different (dis)similarity functions to show how this choice influence the final classification outcome.

In this paper, we define the classification error as $\epsilon = 100\% - Acc$, where Acc is the classification accuracy. For instance, if a method A achieves an accuracy of $Acc_A = 80\%$, it has an error $\epsilon_A = 20\%$. In addition, if a method B, under the same scenario, has a classification error $\epsilon_B = 10\%$, we can say that B reduced the classification error in $\Delta = 100\% - \epsilon_B/\epsilon_A = 50\%$.

Table 1 shows a series of classifiers used with the BAS descriptor characterizing the chosen data set. Note how the well-known out-of-the-box SVM classifier achieves better classification accuracies as we change the way the feature vectors are compared. This is also true for the KNN and OPF classifiers which take more advantage of some (dis)similarity function than others.

The out-of-box KNN classifier (with an L2 comparison metric) has a classification error of $\epsilon = 100\% - 79.1\% = 20.9\%$. The KNN classifier adapted to compare the examples according to the OCS function, KNN-OCS, has a classification error of $\epsilon = 100\% - 85.5\% = 14.5\%$, which means a reduction of $\Delta = 1 - \frac{14.5\%}{20.9\%} = 30.6\%$ in the classification error. The reader might ask why the OCS function allows such an improvement. The reason is that OCS (dis)similarity function strives for finding the minimum effort to match two different alphabet sequences (e.g., a shape representation). OCS is also more robust to rotations in the images yielding a more reliable measure of matching between two shapes considering BAS descriptor.

More important than just establishing comparisons among classifiers, is to see that even with variations on the same classifier, results can be improved with a proper choice of the comparison (dis)similarity function. Sometimes, this choice is not as easy as in the KNN case. SVM classifiers often require a complex mathematical knowledge in designing kernel functions respecting a series of constraints[1]. When the researcher verifies that the comparison (dis)similarity function has a major role in his/her experiments, it might be the case of selecting a classifier in which it is straitforward to switch and evaluate different (dis)similarity functions.

This experiment gives another conclusion: with the appropriate (dis)similarity function, it is possible to obtain the classification results faster without paying the price of a worse classification accuracy. For instance, KNN classifier achieves 85.5% with the computational cost of 0.01 seconds for testing an entire fold (140 elements). In contrast, SVM-RBF requires 0.47 seconds and achieves a classification accuracy of 81.2%. The choice of a better (dis)similarity function does not mean an increase

in the computational time. Also, OPF and KNN classifiers with OCS function are not necessarily computationally more intensive than their L2 versions.

Classifier	**Acc**	**Stdev.**	$\mathbf{t_{train}}$	$\mathbf{t_{test}}$
SVM No-Kernel	71.8%	1.76%	2116.18	0.01
SVM Kernel Linear	72.0%	1.57%	88.85	0.07
SVM Kernel Sigmoid	50.0%	0.01%	77.12	0.06
SVM Kernel RBF	**81.2%**	**1.64%**	**2185.83**	**0.47**
ANN	50.4%	0.48%	1554.20	0.01
SOM	50.0%	0.01%	2376.28	0.39
KNN-L2	79.0%	1.00%	1.12	0.01
KNN-OCS	**85.5%**	**1.42%**	**1.08**	**0.01**
OPF-L1	84.5%	1.06%	0.05	0.01
OPF-L2	82.8%	1.62%	0.46	0.05
OPF-CAN	84.20%	1.20%	0.05	0.01
OPF-OCS	**87.1%**	**1.43%**	**0.04**	**0.01**

Table 1. MPEG-7 CE Shape-1 Part-B 10-fold cross-validation average (*Acc*) and stdev classification results for **BAS** descriptor. The table also shows the average training ($\mathbf{t_{train}}$) and testing ($\mathbf{t_{test}}$) times for each fold, in seconds.

Figure 1 shows a series of classifiers used with the Moments descriptor characterizing the chosen data set. Note that the out-of-the-box OPF classifier with its L2 default (dis)similarity function is not as good as its adapted version with an appropriate Canberra comparison function. The difference of the comparison policies also holds for the SVM classifier. With a more appropriate kernel, we have better classification results. The standard out-of-the-box OPF classifier (with an L2 metric) has a classification error of $\epsilon = 100\% - 68.9\% = 32.1\%$. Compared to the out-of-the-box OPF, OPF classifier enhanced with the Canberra (dis)similarity measure has a classification error of $\epsilon = 100\% - 77.3\% = 22.7\%$ which means a reduction of $\Delta = 100\% - \frac{22.7\%}{32.1\%} = 29.3\%$ in the classification error.

Although the comparison of different classifiers requires a more strict protocol, we can see that KNN classifiers have statistically similar classification errors compared to SVM-RBF classifier regardless the (dis)similarity measures. However, when this is the case, a researcher might want to choose the less time-consuming classifier (usually KNN is faster). Notwithstanding speed issues, sometimes we also have to take into consideration the available space for parameter storage. When this is the case, SVM classifier requires less space and is recommended.

This experiment also shows that neural network-based classifiers such as ANN and SOM do not perform well without parameter tuning. Furthermore, often neural networks have complex configuration parameters and to evaluate different comparison (dis)similarity function with them is not straightforward. Neural Networks

enthusiasts would say that with the proper choice of parameters, NNs are able to implicitly learn several (dis)similarity functions but we will not discuss this point any further.

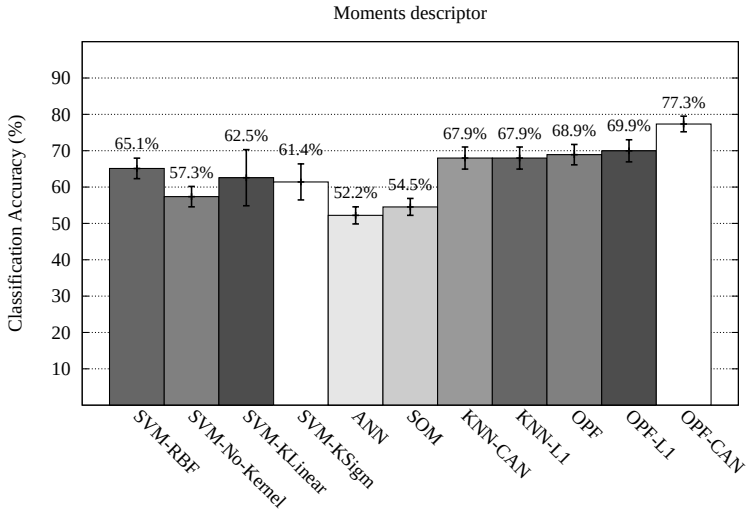


Figure 1. MPEG-7 CE Shape-1 Part-B 10-fold cross-validation average classification results and the corresponding confidence intervals with 95% confidence for **Moments** descriptor.

Sometimes, the comparison (dis)similarity measure does not have a statistically relevant variation contrary to what we have shown in the previous experiments. Figure 2 shows a series of classifiers used with the Tensor Scale descriptor characterizing the chosen data set. In this case, OPF and KNN classifiers have statistically similar classification accuracies ($\cong 77\%$) when the (dis)similarity function is changed. However, SVM presents statistically better results in this experiment when RBF comparison policy is used.

3.3.2. *Part B*

The Part B of the first round of experiments explores the Corel Relevants data set in order to verify the hypothesis on the importance of the (dis)similarity function used in the classification in the context of color image classification. In this experiment, we have considered the classifiers SVM with a Linear, Sigmoid, and RBF kernels as well as an SVM with no kernel. In addition, we have tested ANN and SOM classifiers as well as KNN and OPF with different (dis)similarity functions to show how this choice influences the final classification outcome. For this particular experiment, we have chosen the well-established color image descriptor BIC.

Table 2 shows a series of classifiers used with BIC descriptor characterizing the

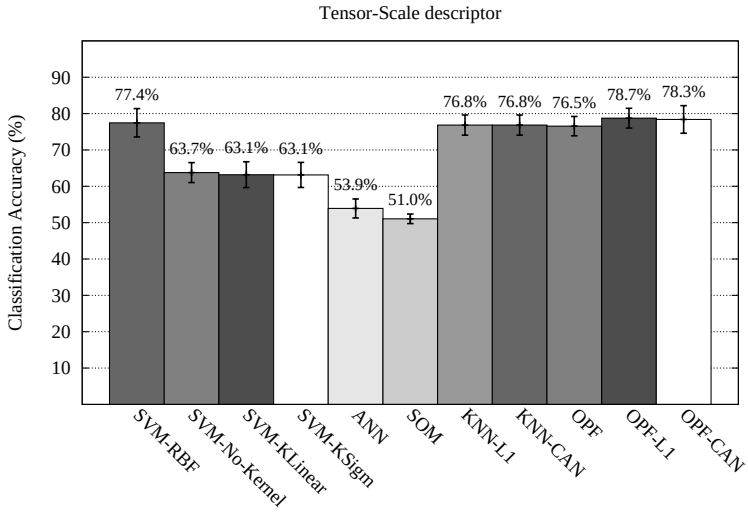


Figure 2. MPEG-7 CE Shape-1 Part-B 10-fold cross-validation average classification results and the corresponding confidence intervals with 95% confidence for **Tensor Scale** descriptor.

chosen data set. BIC descriptor allows the generation of a feature vector with an embedded dLOG space. When that is the case, we refer to the classifier used in the experiment followed by the embedded space and the comparison policy used. For instance, `SVM-dLOG Kernel Linear` uses the SVM classifier with a linear kernel fed with a feature vector in an embedded dLOG space. On the other hand, if we have the label `KNN-L2`, it means we have used KNN classifier fed with a feature vector without dLOG embedding and L2 is the chosen comparison policy.

This experiment shows that the SVM is better with a dLOG embedded space regardless the comparison policy. This happens because the dLOG mapping transforms the feature vector in a well-conditioned feature vector where each bin ranges from 0 to 9^{19}. This behavior allows SVM black-boxes to generalize well. However, when we use the normal feature vector where each bin does not have an upper bound, the comparison policy between elements presents additional normalization problems to SVM, hardening the localization of meaningful hyperplanes to separate the elements.

ANN and SOM classifiers do not take advantage of dLOG embedding and produce poor results. In this experiment, the classifiers KNN and OPF once more take advantage of simple changes in the comparison policies and produce good results with no significant increase in the classification computational time. KNN with an L1 metric over the embedded dLOG space (`KNN-dLOG-L1`) reduces 31.9% of the classification error when compared to SVM with a linear kernel (`SVM-dLOG Kernel Linear`) fed with the same feature vector.

We also can observe that OPF classifier with an embedded dLOG space

achieves 89.7% classification accuracy (10.3% classification error) while achieving 87.8% classification accuracy using the L1 comparison policy and no embedded space. This is because OPF is more robust for comparing feature vectors with no upper bounds. The OPF classifier does not only use the distance between samples to classify them. Papa et al.[33] have showed that the path-cost function applied by OPF to partition the graph into optimum-path trees encodes a power of connectivity between the samples, which makes OPF more robust to handle overlapped classes. On the other hand, KNN is more sensitive to L2 metric than the other classifiers. We believe this happens because, with an unbounded feature vector, some ill-conditioned bins (in the feature vector) dominates the others when squared.

Finally, this experiment reinforces this paper's hypothesis: it is important to think about the classification choices taking place and how to compensate them. A bad decision can lead unsatisfactory results such as 54.6% classification accuracy when using ANN and an embedded dLOG metric space. On the other hand, a better planning and knowledge about the classifier and the problem under consideration can lead to more effective results (e.g., 89.7% classification accuracy when using OPF and an embedded dLOG metric space).

3.4. *Experiments – Round II*

The second round of experiments evaluates how to solve classification problems when dealing with descriptors with features with variable dimensionality. For instance, in the image categorization case, this often happens when using local feature descriptors such as Scale and Invariant Feature Transform (SIFT)[20] or Speeded Up Robust Features (SURF)[34].

Consider a categorization problem in which we have to deal with descriptors producing features with variable dimensionality. In this case, there are several approaches to tackle the variable dimensionality, but two of them are often considered. The first one consists of choosing a direct and often expensive (dis)similarity function that takes the varying dimensionality into consideration. The second approach consists in embedding features with variable dimensionality onto the space $\mathbb{R}^k$ in which standard and cheaper (dis)similarity functions can be used.

The problem with the first solution is the time consumed to perform the (dis)similarity computations. In addition, the use of specific and more complex (dis)similarity functions ties the approach to classification algorithms that supports the use of such (dis)similarity computations (e.g., K-Nearest Neighbors).

The problem with the second solution is that the pre-computation to embed features in $\mathbb{R}^k$ can lead to results that are not as good as the direct computations. To complicate, choosing the appropriate embedding is not always straightforward. However, after the embedding, this approach has the advantage of allowing the direct use of standard machine learning classifiers such as Liner Discriminant Analysis and Support Vector Machines[1].

Figure 3 depicts the experiment results for direct (dis)similarity computations

Classifier	**Acc.**	**Stdev.**	$\mathbf{t_{train}}$	$\mathbf{t_{test}}$
SVM-dLOG No-Kernel	79.1%	2.89%	3.72	0.01
SVM-dLOG Kernel Linear	**83.7%**	**2.85%**	**2670.83**	**0.06**
SVM-dLOG Kernel Sigmoid	83.5%	3.01%	55.20	0.06
SVM-dLOG Kernel RBF	83.2%	3.09%	68.40	0.06
SVM No-Kernel	72.3%	2.56%	20.90	0.01
SVM Kernel Linear	**76.4%**	**3.50%**	**56.74**	**0.05**
SVM Kernel Sigmoid	50.0%	0.01%	52.91	0.06
SVM RBF	50.2%	0.32%	69.20	0.06
ANN-dLOG	54.6%	1.45%	141.29	0.01
ANN	55.7%	1.57%	138.51	0.01
SOM-dLOG	63.4%	3.39%	272.19	0.35
SOM	58.4%	3.16%	196.09	0.32
KNN-dLOG-L1	**88.9%**	**1.50%**	**9.01**	**0.06**
KNN-dLOG-L2	71.2%	2.74%	9.08	0.08
KNN-dLOG-CAN	**88.9%**	**1.50%**	**8.96**	**0.07**
KNN-L1	84.8%	1.43%	8.33	0.06
KNN-L2	63.4%	2.02%	9.07	0.08
KNN-CAN	**84.8%**	**1.43%**	**9.04**	**0.07**
OPF-dLOG-L1	**89.7%**	**1.56%**	**0.06**	**0.01**
OPF-dLOG-L2	81.2%	2.94%	0.03	0.01
OPF-dLOG-CAN	85.0%	2.78%	0.06	0.01
OPF-L1	**87.8%**	**2.07%**	**0.06**	**0.01**
OPF-L2	73.4%	2.92%	0.03	0.07
OPF-CAN	87.2%	2.40%	0.07	0.01

Table 2. Corel Relevants 10-fold cross-validation average (*Acc*) and standard deviation (Stdev.). classification results for **BIC** descriptor (c.f., Sec. 2.2). The table also shows the average training ($\mathbf{t_{train}}$) and testing ($\mathbf{t_{test}}$) times for each fold, in seconds.

compared to the $\mathbb{R}^k$ embedding. Direct computation approaches here use IRM or EMD (dis)similarity functions in conjunction with KNN and OPF classifiers. On the other hand, embedding-based approaches use some form of explicit mapping onto a specific space. In this paper, we use the visual vocabularies concept to embed features into an $\mathbb{R}^k$ space.

Figure 3(a) shows that direct (dis)similarity computation with EMD is more effective than with IRM regardless the classifier used (KNN or OPF). For instance, OPF with EMD (`OPF-EMD`) achieves 87.4% classification accuracy. Using the $\mathbb{R}^k$ embedding, the result is 61.7% classification accuracy with OPF classifier. This embedding is performed using a visual dictionary with the 100 most representative words calculated with K-Means, setting $K = k = 100$. Each word represents a dimension, so this approach embedded the original features in $\mathbb{R}^{100}$.

The experiment in Figure 3(b) confirms the experiment results of Figure 3(a) using another data set. In this case, the $\mathbb{R}^k$ embedding yields a classification result closer to the one obtained with direct (dis)similarity computations.

We believe that the broader the image context in which the local descriptors are used for characterization, the closer will be the classification results either using direct (dis)similarity computations or explicit embedding. As local descriptors are very distinctive, they play a key role in describing objects for direct matching purposes as we show in the experiment of Figure 3(a). On the other hand, for a broader categorization task, as we show in the experiment of Figure 3(b), the visual vocabulary-based mapping is comparable to the direct (dis)similarity computations.

We now turn to the training and classification times when using KNN classifier for direct EMD (dis)similarity computation versus the explicit embedding using visual words. The computational time we report is for the average time over the 10-fold cross-validation procedure executions. Tables 3(a) and 3(b) show the training and classification time for both approaches.

Table 3. Training and classification time for KNN classifier. The training time in (b) already includes the time for K-Means clustering to find the projecting space as well as the mapping onto such space. The testing time in (b) already includes the time for mapping the testing image to the found dictionary.

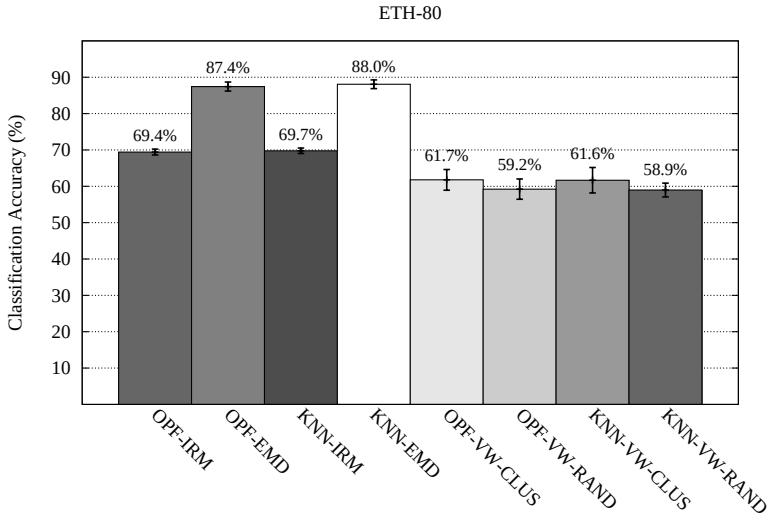
(a) Considering the direct EMD (dis)similarity approach.

	Training time	Testing time for one example
ETH-80 + EMD	3.8 hour(s)	4.7 sec(s).
Relevants + EMD	17.7 hour(s)	43.0 sec(s).

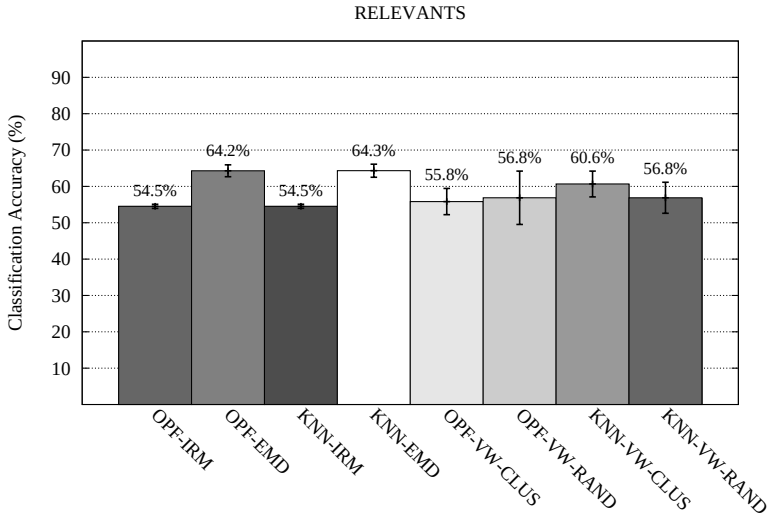
(b) Considering the L2 Euclidean (dis)similarity approach on the space.

	Training time	Testing time for one example
ETH-80 with 100 Visual Words	0.30 hour(s)	0.5 sec(s).
Relevants with 100 Visual Words	0.03 hour(s)	0.1 sec(s).

For the experiments results, the direct computation of EMD approach is one or two orders of magnitude more expensive than the mapping onto the space using 100 visual words. To conclude, two aspects must be highlighted: (1) the number of representative words used in the $\mathbb{R}^k$ embedding greatly influences the final classification outcome. The 100-visual words used herein are just an example; and (2) although $\mathbb{R}^k$ embedding led to worse results than the direct (dis)similarity computation, embedding-based solutions tend to be faster than the approaches based on direct (dis)similarity calculations. IRM (dis)similarity has time complexity $O(m^2)$ to calculate the distance between two features of dimension m while EMD's time complexity is $O(m^3)$. Conversely, the (dis)similarity computations in $\mathbb{R}^k$ is, most of the times, linear in the space dimension.



(a) Direct (dis)similarity computation (e.g., IRM and EMD) *vs.* Explicit Mapping (e.g., Visual Vocabularies) – ETH-80 Data set.



(b) Direct (dis)similarity computation (e.g., IRM and EMD) *vs.* Explicit Mapping (e.g., Visual Vocabularies) – Relevants Data set.

Figure 3. The effects of using descriptors producing features with variable dimensionality..

3.5. *Experiments – Round III*

The third and final round of experiments evaluates the impacts on the final classification results when using combinations of binary classifiers to solve multi-class

classification problems.

It is not unusual to find researchers discussing about the speed (or lack of) of Support Vector Machines when dealing with N-Way classification problems. Before discussing speed issues, however, we need to understand what is going on in the machinery of an SVM-based classifier.

For N-Way classification problems, SVMs use some form of combination of several binary SVMs. For instance, if SVM solves the multi-class from binary problem using OVA, it will use less binary classifiers but will have imbalanced training sets. On the other hand, if it solves the multi-class from binary problem using OVO, it will have balanced and small training sets but will have many binary classifiers. In the experiments, we show the impacts of such choices.

We do not defend whether multi-class from binary classifiers such as SVMs are better or worse than any other naturally multi-class classifier such as K-Nearest Neighbors[1] or Optimum-Path Forest[33]. We discuss what happens if one has decided to solve a given N-Way classification problem with a given multi-class from binary classifier such as SVMs.

We compare five different approaches to combine binary classifiers to solve multi-class problems. We present results for three data sets using the *One-vs-All*, *One-vs-One*, *Error Correcting Output Codes*, and two optimizations proposed by Passerini et al.[31] and Rocha and Goldenstein[32] over Error Correcting Output Codes.

Figures 4 and 5 depict results for five different approaches that combine binary classifiers for N-Way classification. The experiments comprise the MNist, and Auslan data sets considering combinations of SVM binary classifiers.

For Figure 4, if we use all the $N_c = \binom{10}{2}$, we need 45 binary SVM classifiers and achieve a classification accuracy of 93.0%. Any other number of binary classifiers does not yield a good result as the blue curve shows. OVA approach only requires 10 binary classifiers achieving 91.0% classification accuracy. However, the OVA approach gives rise to highly imbalanced training sets having one positive and nine negative classes for each binary classifier. ECOC-based solutions present good results achieving the same 91.0% classification accuracy but using more balanced training sets.

As the number of classes increases (Fig. 5), ECOC-based approaches lead to better results with less binary classifiers. This result shows that it is possible to combine a few binary classifiers to produce high classification rates taking advantage of binary classifiers such as SVMs. Figure 5 shows a problem with 96 classes. In this case, the OVO approach requires all the $N_c = \binom{96}{2} = 4,560$ binary SVM classifiers achieving a classification accuracy of 90.0%.

The imbalance problem mentioned earlier when using an OVA approach is more pronounced in this experiment. OVA approach requires 96 binary classifiers achieving 47.0% classification accuracy. This is worse than the OVO-based results. ECOC-based solutions such as the ones showed in the red, green and black curves yield good classification results using a small number of binary classifiers. For instance, the original ECOC approach achieves 87.0% classification accuracy with 100 bi-

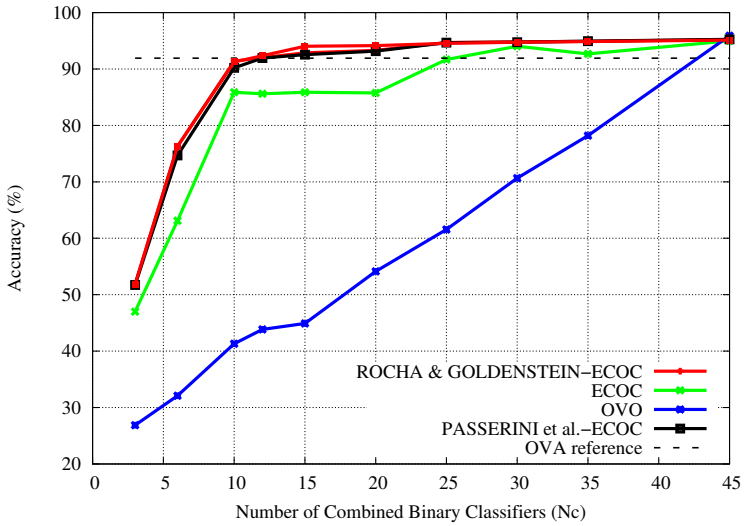


Figure 4. Different approaches for combining binary classifiers for N-Way classification. OVA *vs.* ECOC *vs.* OVO *vs.* Passerini et al.-ECOC *vs.* Rocha and Goldenstein-ECOC for MNist data set considering SVM binary classifiers.

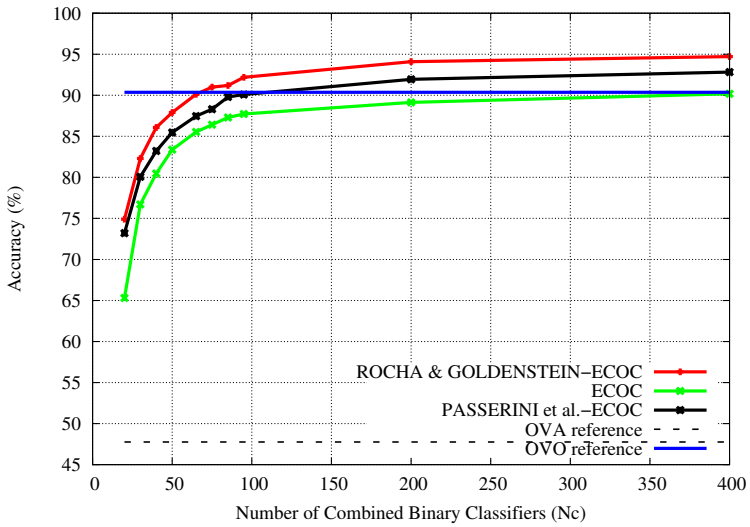


Figure 5. Different approaches for combining binary classifiers for N-Way classification. OVA *vs.* ECOC *vs.* OVO *vs.* Passerini et al.-ECOC *vs.* Rocha and Goldenstein-ECOC for Auslan data set considering SVM binary classifiers.

nary classifiers. This result uses about the same number of binary classifiers as OVA approach and it is 100% more accurate.

The experiments in this section lead to two conclusions: (1) binary classifiers can be used to solve N-Way classification problems; and (2) the existing combination approaches for this intent can lead to different results, some of them with serious imbalance training problems.

As a recommendation, researchers can perform a small experiment with a smaller version of the problem under consideration using some of the standard combination approaches in order to determine the most appropriate combination policy.

4. Conclusions

In this paper we have discussed the problems with respect to the use of machine learning black-boxes which shall be taken into account when solving classification problems. Despite the initial good results obtained with such out-of-the-box classifiers, it is important to go deeper into the classifier machinery and, with the proper knowledge, change a few parameters to obtain a more discriminative classifier.

For this discussion, we have focused on three aspects of the classifiers: (1) the way they compare examples in the feature space; (2) the impact of using features with different dimensionality; and (3) the impact of using binary classifiers to solve multi-class problems.

We have shown how the knowledge about the classifier machinery improves the results beyond out-of-the-box ML solutions. For instance, using KNN classifier and BAS descriptor with the appropriate OCS metric, it is possible to reduce the classification error in the MPEG-7 CE Shape-1 Part-B data set in $\epsilon = 30.9\%$ when compared to the out-of-the-box KNN classifier using the standard (non-tweaked) L2 metric (c.f., Sec. 3.3).

As another example, we discussed two approaches to deal with a classification problem considering features with variable dimensionality. It is possible either to use an appropriate (dis)similarity function that takes this difference into account such as the Earth Mover's Distance or to embed the features in $\mathbb{R}^k$ using the concept of visual dictionaries. In the experiments results, embedding-based solutions tend to be faster than the approaches based on direct (dis)similarity calculations while not loosing much in classification accuracy.

We also have showed that some well known binary classifiers such as Support Vector Machines are widely used for solving multi-class problems. As SVM is inherently a binary classifier, we discussed how some out-of-the-box machine learning black-boxes extend it to the multi-class operational scenario. We also discussed and presented experiments showing that the choice of the class binarization policy can improve the results beyond the ones obtained by default while reducing the computational time in some cases.

It is important to state that we are not defending any particular classifier.

We believe that each situation requires a proper problem understanding for the deployment of a good solution. The experiments we have performed reinforces this paper's hypothesis: it is important to think about the classification choices taking place and how to compensate them.

In face of the results we show in this paper and based on our experience, we are not in favor of the use of machine learning black-boxes. Independent of our position, however, we know some researchers and practitioners still will use such black-boxes either due to time constraints or due to unawareness on how to do otherwise. For them, we believe the paper will be useful as a guide to perform a good choice of which black-box to use and how to deal with its implemented methods. If one, otherwise, uses the classifiers directly (e.g., own author's implementation of a classifier), this paper will also provide him/her with a basis for some of the key choices to think about when solving a given problem.

Sometimes, these so called black-boxes are restrictive and do not offer basic options of changing important parameters of the implemented methods. As an example, take the case of SVM. An important SVM companion is the option to perform grid search for choosing its parameters. This option might be missing in an ML black-box. Another example, also related to SVM, refers to the behavior of such classifier when dealing with multi-class problems. Normally, this is implemented in the black-boxes using either OVA or OVO approaches. As we showed in this paper, the former suffers from imbalance on the training data while the latter requires the training of N^2 binary base learners.

Finally, there are several other important choices we need to think of when solving a machine learning problem either directly or using an ML black-box. We have discussed three of them. Other aspects to consider include (1) the parameter choice for modeling the kernels in kernel-based classifiers (e.g., SVMs); (2) the use of categorical variables in conjunction with numerical ones; and finally (3) the behavior of other important classifiers in the literature such as Maximum Likelihood[1] with respect to all of these questions.

Acknowledgments

We thank the São Paulo Research Foundation (FAPESP) Grants 2009/16206-1 and 2010/05647-4, University of Campinas PAPDIC Grant 519.292-340/10, the Brazilian National Research Council (CNPq), and Microsoft Research for the financial support.

Bibliography

1. C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 1 edition, 2006.
2. A. Rocha, D. Hauagge, J. Wainer, and S. Goldenstein. Automatic fruit and vegetable classification from images. *Elsevier Computers and Electronics in Agriculture*, 70(1):96–104, Jan. 2010.
3. R. Baeza-Yates. *Clustering and Information Retrieval*. Kluwer Academic Publishers, 1 edition, 2003.

4. G. Heidemann. Unsupervised image categorization. *Elsevier Image and Vision Computing*, 23(10):861–876, Oct. 2004.
5. M. Weber. *Unsupervised learning of models for object recognition*. Phd thesis, California Institute of Technology, May 2000.
6. O. Chapelle, B. Scholkopf, and A. Zien. *Semi-supervised learning*. MIT Press, 2006.
7. M. G. Quiles, L. Zhao, F. A. Breve, and A. Rocha. Label propagation through neuronal synchrony. In *IEEE Intl. Joint Conf. on Neural Networks*, 2010.
8. T. R. Andel and A. Yasinsac. On the credibility of Manet simulations. *Computer*, 39(7):48–54, 2006.
9. P. Berkhin. *Grouping Multidimensional Data*, chapter A Survey of Clustering Data Mining Techniques. Springer, 2006.
10. R. Torres and A. Falcão. Content-based image retrieval: Theory and applications. *Journal of Theoretical and Applied Computing*, 13(2):161–185, 2006.
11. Rafael Gonzalez and Richard Woods. *Digital Image Processing*. Prentice-Hall, 3 edition, 2007.
12. M. K. Hu. Visual pattern recognition by moment invariants. *IEEE Trans. on Information Theory*, 8(2):179–187, 1962.
13. S. A. Dudani, K. J. Breeding, and R. B. McGhee. Aircraft identification by moment invariants. *IEEE Trans. on Computers*, C-26(1):39–46, 1977.
14. N. Arica and F. T. Y. Vural. BAS: a perceptual shape descriptor based on the beam angle statistics. *Elsevier Pattern Recognition Letters*, 24(9-10):1627–1639, 2003.
15. P. K. Saha. Tensor scale: A local morphometric parameter with applications to computer vision and image processing. *Elsevier Computer Vision and Image Understanding*, 99(3):384–413, 2005.
16. F. A. Andaló, P. A. V. Miranda, R. Torres, and A. Falcão. Shape feature extraction and description based on tensor scale. *Elsevier Pattern Recognition*, 43(1):26–36, 2010.
17. R. Torres, A. X. Falcão, and L. F. Costa. A graph-based approach for multiscale shape analysis. *Elsevier Pattern Recognition*, 37(6):1163–1174, 2004.
18. C. T. Zahn and R. Z. Roskies. Fourier descriptors for plane closed curves. *IEEE Trans. on Computer*, c-21(3):269–281, 1972.
19. R. Stehling, M. Nascimento, and A. Falcão. A compact and efficient image retrieval approach based on border/interior pixel classification. In *ACM Intl. Conf. on Information and Knowledge Management*, pages 102–109, 2002.
20. D. Lowe. Distinctive image features from scale-invariant keypoints. *Intl. Journal of Computer Vision*, 60(2):91–110, February 2004.
21. G. N. Lance and W. T. Williams. Computer programs for hierarchical polythetic classification ("similarity analysis'). *Oxford Computer Journal*, 9:60–64, 1966.
22. Y. Rubner, C. Tomasi, and L. J. Guibas. The earth mover's distance as a metric for image retrieval. *IJCV*, 40(2):99–121, 2000.
23. J. Li, J. Z. Wang, and G. Wiederhold. IRM: Integrated region matching for image retrieval. In *ACM Intl. Conf. on Multimedia*, pages 147–156, 2000.
24. E. Valle. *Local-descriptor matching for image identification systems*. Phd thesis, Universit de Cergy-Pontoise, Cergy-Pontoise, France, 2008.
25. J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *IEEE Intl. Conf. on Computer Vision*, pages 1470–1477, 2003.
26. O. Pujol, P. Radeva, and J. Vitria. Discriminant ECOC: A heuristic method for application dependent design of ECOC. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 28(6):1007–1012, Jun 2006.
27. P. Clark and R. Boswell. Rule induction with CN2: Some improvements. In *European Working Session on Learning on Machine Learning*, pages 151–163, 1991.

28. R. Anand, K. Mehrotra, C. Mohan, and S. Ranka. Efficient classification for multiclass problems using modular neural networks. *IEEE Trans. on Neural Networks*, 6(1):117–124, Jan 1995.
29. T. Dietterich and G. Bakiri. Solving multi-class problems via ECOC. *Journal of Artificial Intelligence Research*, 2(1):263–286, Jan 1996.
30. E. Allwein, R. Shapire, and Y. Singer. Reducing multi-class to binary: A unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1(1):113–141, Jan 2000.
31. A. Passerini, M. Pontil, and P. Frasconi. New results on error correcting output codes of kernel machines. *IEEE Trans. on Neural Networks*, 15(1):45–54, Jan 2004.
32. A. Rocha and S. Goldenstein. From binary to multi-class: divide to conquer. In *Intl. Conf. on Computer Vision Theory and Applications*, pages 323–330, February 2009.
33. J.P. Papa, A.X. Falcão, and C.T.N. Suzuki. Supervised pattern classification based on optimum-path forest. *Wiley Intl. Journal of Imaging Systems and Technology*, 19(2):120–131, 2009.
34. Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European Conf. on Computer Vision*, pages 1–14, 2006.

Authors' Biographies

Anderson de Rezende Rocha received his B.Sc (Computer Science) degree from Federal University of Lavras (UFLA), Brazil in 2003. He received his M.Sc. and Ph.D. (Computer Science) from University of Campinas (Unicamp), Brazil, in 2006 and 2009, respectively. Currently, he is an assistant professor in the Institute of Computing, Unicamp, Brazil. His main interests include digital image and video forensics, pattern analysis, machine learning, and general computer vision. He has actively worked as a program committee member in several important Computer Vision, Pattern Recognition, and Digital Forensics events. In 2011, he has been elected an affiliate member of the *Brazilian Academy of Sciences* (ABC). He is an elected member of the IEEE Information Forensics and Security Technical Committee (IFS-TC) and co-general chair of the 2011 IEEE Intl. Workshop on Information Forensics and Security (WIFS). In 2011, he was also named a *Microsoft Research Faculty Fellow*.

João Paulo Papa João Paulo Papa received his B.Sc. in Information Systems from the São Paulo State University, SP, Brazil. In 2005, he received his M.Sc. in Computer Science from the Federal University of São Carlos, SP, Brazil. In 2008, he received his Ph.D. in Computer Science from the University of Campinas, SP, Brazil. During 2008-2009, he had worked as post-doctorate researcher at the same institute. He has been Professor at the Computer Science Department, São Paulo State University, since 2009, and his research interests include machine learning, pattern recognition and image processing.

Luis A. A. Meira Luis A. A. Meira received his Engineering (Computer Science) and Ph.D. degrees from University of Campinas (Unicamp), Brazil. Currently, he is an Assistant Professor in the Faculty of Technology, Unicamp, Brazil. His main interests include Computer Theory, Approximation Algorithms, Combinatorics & Optimization and Machine Learning Theory.