How Far You Can Get Using Machine Learning Black-Boxes

Anderson Rocha
Institute of Computing
Univ. of Campinas (UNICAMP)
Campinas, Brazil

João Paulo Papa
Department of Computer Science
State Univ. of São Paulo (UNESP)
Bauru, Brazil

Luis A. A. Meira
Department of Science and Technology
Federal Univ. of São Paulo (UNIFESP)
São José dos Campos, Brazil

***Abstract*—Supervised Learning (SL) is a machine learning research area which aims at developing techniques able to take advantage from labeled training samples to make decisions over unseen examples. Recently, a lot of tools have been presented in order to perform machine learning in a more straightforward and transparent manner. However, one problem that is increasingly present in most of the SL problems being solved is that, sometimes, researchers do not completely understand what supervised learning is and, more often than not, publish results using machine learning black-boxes. In this paper, we shed light over the use of machine learning black-boxes and show researchers how far they can get using these out-of-the-box solutions instead of going deeper into the machinery of the classifiers. Here, we focus on one aspect of classifiers namely the way they compare examples in the feature space and show how a simple knowledge about the classifier's machinery can lift the results way beyond out-of-the-box machine learning solutions.**

***Keywords*-Machine Learning Black-Boxes; Metrics Space; Pattern Analysis; Support Vector Machines; Optimum-Path Forest; Neural Networks; K-Nearest Neighbors.**

I. Introduction

In our 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. In this scenario, one of the problems we often face is data classification.

There are several approaches for data classification in the literature and one of the problems they need to address in almost every situation is how to take advantage of known information to infer properties to allow the 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]–[5], to unsupervised ones in which label information is not used [6]–[9]. In between these solutions, there are semi-supervised approaches designed to learn from both labeled and unlabeled data [10]–[13].

Recently, a lot of 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 15 years [1]. With several good research groups actively working in ML approaches, we now have the concept of self-containing machine learning solutions that many times work out-of-the-box leading us 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, most of the times, we do not need to worry about implementation details regarding the classifiers neither some confusing parameters needed to tune the classification training process.

There are several machine learning black-boxes out there implemented in software such as: Weka[1], R[2], SVM Light[3], LibSVM[4], LibOPF[5] among others.

Although it is quite important to have such black-boxes helping us to deal with several problems nowadays, it comes with an inherent problem increasingly more evident. Researchers and students are progressively relying on these black-boxes and, more often than not, achieving results without even knowing what is going on in the machinery of the classifiers. This lack of knowledge sometimes just mean the researchers can not explain their results but also mean they are achieving one result that could be much more effective if they had knowledge about some important details about the classifiers.

In this paper, we shed light over the use of machine learning black-boxes. We show 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 one aspect of classifiers namely the distance function used to compare examples in the feature space. We show how a simple knowledge about the classifier machinery can lift the results way beyond out-of-the-box ML solutions. For instance, we show that with the same descriptor and the same classifier, sometimes we can reduce the classification error in more than 50% by just providing the classifier with a proper comparison metric.

To validate our observations, we show 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

[1] http://www.cs.waikato.ac.nz/ml/weka/
[2] http://www.r-project.org/
[3] http://svmlight.joachims.org/
[4] http://www.csie.ntu.edu.tw/∼cjlin/libsvm/
[5] http://www.ic.unicamp.br/∼afalcao/libopf/

a few parameters regarding the policy of comparison of examples. We hope the questions we raise in this paper can help researchers when thinking about using ML approaches to solve research problems.

The remaining of this paper is organized as follows. Section II presents some subjects we consider important for the fully understanding of this paper without the need of recurring to reference books and papers. Section III conveys the experiments we perform to corroborate our position in this paper and show how just the choice of the comparison policy in the classifiers' machinery can influence the classification outcomes. Finally, Section IV closes this paper and discusses some additional perspectives we should look at when using machine learning approaches to help us solving daily-life problems.

II. Background

In this paper, we raise the question of how far researchers can get using out-of-the-box machine learning solutions. We discuss the importance of choosing appropriate policy (dis)similarity functions to compare examples in a machine learning approach. Without loss of generality, we discuss our position using image categorization problems. In a general purpose image categorization problem, we are given a set of images and we need to point out the general class of such images.

To perform the categorization task, it is common to find a proper feature space in which we can characterize the images under analysis and make the comparison among them faster and more discriminative. The characterization is often made by employing image descriptors. Later on, a machine learning approach is fed with these features to perform the classification leading to the final outcome of the system.

In this context, this section presents some definitions regarding the state-of-the-art in supervised learning as well as feature image description.

A. Machine Learning

Machine learning is a scientific discipline concerned with the design and development of approaches that allow machines to evolve behaviors based on empirical data. Machine learning approaches often focus on learning to recognize complex patterns for further intelligent decision-making processes. Nowadays, we face several problems in which machine learning techniques can help us such as character recognition, speech recognition, heart attack prediction, data classification and categorization, detection of frauds in checks, credit cards, among others [1]. When solving such complex problems, we can use pre-defined classifiers or learning-based classifiers which can be supervised, semi-supervised or unsupervised.

In this context, a classifier is an inference motor that seeks to implement efficient and effective strategies to compute relationships between pairs of concepts or to compute relationships between a concept and a set of instances [1].

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 label (class indicator) of the input object under analysis. 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 (SVMs), Classification Trees, Neural Networks (NNs), Ensembles of Classifiers (Bagging and Boosting), K-Nearest Neighbors, Neural Networks and several others [1]–[4].

Unsupervised learning approaches are those in which no label information is available. Often, we 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 on the data under analysis. 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 [6], [14].

In between Supervised and Unsupervised solutions, there are semi-supervised approaches designed to learn from both labeled and unlabeled data such as [10]–[13].

No matter what kind of solution we employ, in order to perform the intended tasks, most of these approaches need to compute relationships between pairs of concepts. This need leads us to the feature space in which such computations need to 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 simple choice of the comparison policy between concepts has a major impact on the classification outcome. The problem is that, most of the times, researchers do not even have this in mind and simply use machine learning black-boxes and rely on their built-in comparison policies. Notwithstanding, such built-in comparison policies sometimes are not appropriate to the set of descriptors at hand.

B. Feature Description

Roughly speaking, in data classification, we have an input example we want to classify, the feature vector summarizing and, hopefully, characterizing this example, 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 [15].

More 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 [16].

Representing it in the domain of image processing, an image stream (or simply image) $\hat{I}$ is a pair $(D_I, \vec{I})$, where:

- D_I is a finite set of *pixels* 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 pixel 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 pixel).

In this context, 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 very well-known ones such as: color histograms, multi-scale fractals, Fourier coefficients, texture descriptors among others. Essentially, an image descriptor seeks to encode the maximum image properties as possible (e.g., color, texture, shape, silhouette, etc.) in order to encode the underlying concepts they strive for summarizing [15].

Following this line of thought, we may think of an image content descriptor D as a pair (ϵ_D, δ_D), such that

- $\epsilon_D : \hat{I} \rightarrow \mathbb{R}^m$ 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 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.

To make it easier the understanding of these concepts, suppose we want to compare two images $\hat{I}_1$ and $\hat{I}_2$. The first step consists of characterizing them using the extraction algorithm ϵ_D which yields the feature vectors $\vec{v}_{\hat{I}_1}$ and $\vec{v}_{\hat{I}_2}$. To compare the examples, we use the similarity function δ_D to determine the similarity value between $\vec{v}_{\hat{I}_1}$ and $\vec{v}_{\hat{I}_2}$ [15].

The whole argument of this paper is that as machine learning black-boxes are becoming increasingly available to general-purpose uses, people are forgetting to think of their problems under the appropriate point of view and, most of the times, feed their feature vectors to these out-of-the-box solutions 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 a much more impressive result by simply understanding the nature of their feature vector function extraction and its underlying appropriate (dis)similarity function.

In the following we present some of the feature descriptors we use in this paper as well as their appropriate (dis)similarity functions.

1) **Moment Invariants** [17], [18]. For Moment Invariants, each object is represented by a 14-dimensional feature vector, including two sets of normalized Moment Invariants, one from the object contour and another from its solid object silhouette. Again, the Euclidean distance is usually used to measure the similarity between different shapes as represented by their Moment Invariants.
2) **Beam Angle Statistics (BAS)** [19]. The BAS descriptor is based on the beams originated from a contour pixel. A beam is defined as the set of lines connecting a contour pixel to the rest of the pixels along the contour. At each contour pixel, the angle between a pair of lines is calculated, and the shape descriptor is defined by using the third-order statistics of all the beam angles in a set of neighborhoods. The similarity between two BAS moment functions is measured by an optimal correspondent subsequence (OCS) algorithm [20]. The most appropriate metric for this descriptor, as discussed by the authors, is OCS [19] (c.f., Section II-C:5).
3) **Tensor Scale Descriptor (TSD)** [21]. TSD is a shape descriptor based on the tensor scale concept morphometric parameter yielding a unified representation of local structure thickness, orientation, and anisotropy [22]. That is, at any image point, its tensor scale is represented by the largest ellipse (2D) centered at that point and within the same homogeneous region. TSD is obtained by extracting the tensor scale parameters for the original image and then computing the ellipse orientation histogram. TSDs are compared by using a correlation-based distance function.
4) **Multiscale Fractal Dimension (MFD)** [23]. MFD is an alternative method for estimating the shape complexity of objects. Basically, it consists in estimating a curve that represents the changes in shape complexity as we change the visualization scale. Different of Fractal Dimension, which is a numeric value, this approach produces a curve which performs a more accurate shape discrimination
5) **Fourier Descriptor (FD)** [24]. Fourier transformation on shape signatures is widely used for shape analysis. The FD is essentially the Fourier transformed coefficients of the shape. These descriptors represent the shape of the object in a frequency domain. The lower frequency descriptors contain information about the general features of the shape, and the higher frequency descriptors contain information about finer details of the shape.
6) **Border-Interior Descriptor** (BIC) [25]. Stehling et al. have presented the border/interior pixel classification (BIC). The approach relies on the RGB color-space uniformly quantized in $4 \times 4 \times 4 = 64$ colors. After the quantization, the image pixels are classified as border or interior. A pixel is classified as interior if its 4-neighbors (top, bottom, left, and right) have the same quantized color. Otherwise, it is classified as border. After the image pixels are classified, two color histograms are computed: one for border pixels and another for interior pixels. The most appropriate metric for this descriptor, as discussed by the authors, is dLog [25] (c.f., Section II-C:1). In particular, for this descriptor we can generate a feature vector with a dLOG embedded space. For that, we transform each feature value using dLOG [25] and the outcome is a value between 0 and 9.

C. (Dis)similarity functions

In this section, we present some (dis)similarity functions we use in this work. Some of the descriptors we discuss in this paper, have an appropriate (dis)similarity function and we point out when that is the case.

1) **dLOG**. It is a distance function to compare two histograms [25]. It is the appropriate metric to use with the border-interior descriptor (c.f., Section II-B:6). Given two histograms q and d with M bins each, dLOG compares them according to the following definition

$$dLOG(q,d) = \sum_{i=0}^{i<M} \|f(q[i]) - f(d[i])\| \quad (1)$$

$$f(x) = \begin{cases} 0, & \text{if } x = 0 \\ 1, & \text{if } 0 < x < 1 \\ \lceil \log_2 x \rceil + 1, & \text{otherwise} \end{cases} \quad (2)$$

2) **Manhattan**. It is a distance function often used to compare two histograms or feature vectors. It is commonly known as L1 distance. Given two vectors q and d with M features each, Manhattan compares them according to the following definition

$$Manhattan(q,d) = \sum_{i=0}^{i<M} |q[i] - d[i]| \quad (3)$$

3) **Euclidean**. It is a distance function often used to compare two histograms or feature vectors. It is commonly known as L2 distance. Given two vectors q and d with M features each, the Euclidean distance compares them according to the following definition

$$Euclidean(q,d) = \sqrt{\sum_{i=0}^{i<M} (q[i] - d[i])^2} \quad (4)$$

4) **Canberra**. It is a distance function often used to compare data scattered around an origin [26]. Given two vectors q and d with M features each, the Canberra distance compares them according to the following definition

$$Canberra(q,d) = \sum_{i=0}^{i<M} \frac{|q[i] - d[i]|}{|q[i]| + |d[i]|}, \quad (5)$$

where, by definition, $\frac{0}{0} = 0$.

5) **OCS**. The OCS (*Optimal Correspondence of String Subsequences*) metric function was originally designed to compute the distance between two strings [20]. The main idea is to calculate the minimum effort to match two different alphabet sequences. For a formal definition, please refer to the work of Wang & Pavlidis [20]. It is the appropriate metric to use with the BAS descriptor (c.f., Section II-B:2).

D. Data sets

To corroborate the hypothesis we discuss in this paper, we have used a series of image descriptors (c.f., Section II-B) and image data sets. In this section, we present 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[6].
2) **Corel Relevants**. This data set comprises 1,624 images from Corel Photo Gallery reported in [25]. The collection contains 50 color image categories and is referred to as the Corel Relevant sets (RRSets)[7].

E. Quality Assessment

The accuracy Acc of each classifier we report in this paper is measured by taking into account the fact that the classes may have different number of elements in the testing as reported in [27]. For instance, if there are two classes with very different sizes and a classifier always assigns the label of the largest class, its accuracy will fall drastically due to the high error rate on the smallest class.

Let Z_i, $i = 1, 2, \ldots, c$, be the number of samples in the test set Z for each class i. We define

$$e_{i,1} = \frac{FP(i)}{|Z| - Z_i} \quad \text{and} \quad e_{i,2} = \frac{FN(i)}{Z_i}, \; i = 1, \ldots, c \quad (6)$$

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

$$E(i) = e_{i,1} + e_{i,2}, \quad (7)$$

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

$$Acc = \frac{2c - \sum_{i=1}^{c} E(i)}{2c} = 1 - \frac{\sum_{i=1}^{c} E(i)}{2c}. \quad (8)$$

III. Experiments and Discussion

In this section, we show the experiments we perform to validate our hypothesis 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 organize the experiments into two rounds. In the first round, we evaluate a series of shape descriptors over the MPEG-7 CE Shape-1 Part-B data set (c.f., Section II-D:1). In the second round,

[6]This data set can be found at `http://www.cis.temple.edu/~latecki/TestData/mpeg7shapeB.tar.gz`

[7]This data set can be found at `http://webdocs.cs.ualberta.ca/~mn/BIC/`

we evaluate a well known color image descriptor over the Corel color images RRSets data set (c.f., Section II-D:2).

In all experiments, we have performed a 10-fold cross-validation in order to assess how the results generalize to an independent data set [1]. More specifically, in a K-fold cross validation procedure, we partition a data set into K subsamples. Of the K subsamples, we retain a single subsample as the validation data for testing the model, and use the remaining $K-1$ subsamples as training data. We repeat the cross-validation process K times (the folds), with each of the K subsamples used exactly once as the validation data. We average the K results from the folds to produce a single estimation. Since most of ML black-boxes, by default, do some fine-tuning without warning the user, 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.

A. Experiments – Round I

In the first round of experiments, we explore the MPEG-7 CE Shape-1 Part-B data set in order to verify our 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 as well as a simple SVM with no kernel. In addition, we have tested ANN and SOM classifiers as well as KNN and OPF with different metrics to show how this choice influence the final classification outcome.

In this paper, we define the classification error as $\epsilon = 1 - \mu$, where μ is the classification accuracy. For instance, if a method A achieves an accuracy of $\mu_A = 80\%$, 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 $\Delta = 1 - \epsilon_B \ / \ \epsilon_A = 50\%$.

Table I shows a series of classifiers used with the BAS (c.f., Section II-B:2) descriptor characterizing the chosen data set. Note how the powerful 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 simple KNN and OPF classifier 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 $\epsilon = 100\% - 79.1\% \cong 20.9\%$. Compared to KNN-L2, the KNN classifier adapted to compare the examples according to the OCS metric which is more appropriate for shape representations, KNN-OCS, has a classification error of $\epsilon = 100\% - 85.5\% \cong 14.5\%$ which means a reduction of $\Delta = \frac{14.5\%}{20.9\%} \cong 31\%$ in the classification error. The reader might ask why the OCS metric allows such an improvement. The reason is simple: given that OCS (dis)similarity function strives for finding the minimum effort to match two different alphabet sequences (e.g., a shape representation) it is 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 of 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 it seems to be with a classifier such as KNN. 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 her experiments, perhaps 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 us another interesting conclusion: with the appropriate (dis)similarity function, sometimes we can obtain the classification results faster without paying the price of a smaller classification accuracy. For instance, KNN classifier achieves $\cong 85.55\%$ with the computation cost of only 0.01 seconds for testing an entire fold ($\cong 140$ elements). At the same time, the computationally-intensive SVM-RBF requires 0.47 seconds and achieves a classification accuracy of 81.25%. Sometimes, the choice of a better metric does not mean an increase in the computation time. Also, OPF and KNN classifiers with OCS metric are not necessarily computationally more intensive than their L2 versions.

Classifier	μ **Acc.**	σ	$\mathbf{t_{train}}$	$\mathbf{t_{test}}$
SVM No-Kernel	71.88%	1.76%	2116.18	0.01
SVM Kernel Linear	72.03%	1.57%	88.85	0.07
SVM Kernel Sigmoid	50.00%	0.01%	77.12	0.06
SVM Kernel RBF	**81.25%**	**1.64%**	**2185.83**	**0.47**
ANN	50.43%	0.48%	1554.20	0.01
SOM	50.00%	0.01%	2376.28	0.39
KNN-L2	79.09%	1.00%	1.12	0.01
KNN-OCS	**85.55%**	**1.42%**	**1.08**	**0.01**
OPF-L1	84.56%	1.06%	0.05	0.01
OPF-L2	82.88%	1.62%	0.46	0.05
OPF-CAN	84.20%	1.20%	0.05	0.01
OPF-OCS	**87.13%**	**1.43%**	**0.04**	**0.01**

Table I
MPEG-7 CE SHAPE-1 PART-B 10-FOLD CROSS-VALIDATION AVERAGE (μ) AND STDEV (σ) CLASSIFICATION RESULTS FOR **BAS** DESCRIPTOR (C.F., SECTION II-B:2). ALSO SHOWN IN THE TABLE, ARE 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 (c.f., Section II-B:1) 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 $\epsilon = 100\% - 77.3\% \cong 22.7\%$ which means a reduction of $\Delta = \frac{22.7\%}{32.1\%} \cong 30\%$ in the classification error.

Although is dangerous to compare different classifiers with 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 us that neural network-based classifiers such as ANN and SOM do not perform well without parameter tuning and therefore are not recommended as out-of-the-box classifiers. 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 are not going do delve into more details in this direction.

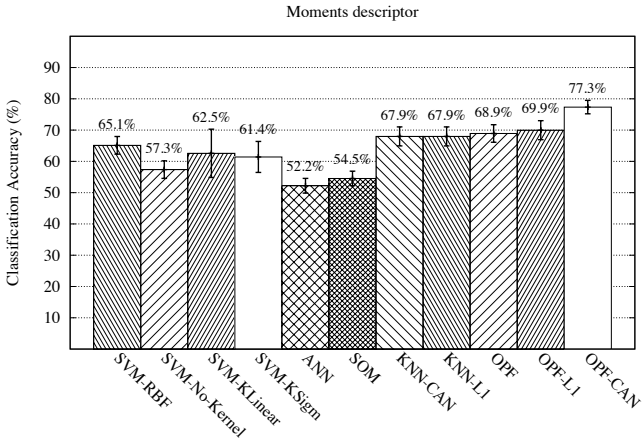

Figure 1. MPEG-7 CE Shape-1 Part-B 10-fold cross-validation average (μ) classification results and the corresponding confidence intervals with 95% confidence for **Moments** descriptor (c.f., Section II-B:1).

Sometimes, the comparison (dis)similarity measure does not have a statistically relevant variation contrary to what we have shown in the last experiments. Figure 2 shows a series of classifiers used with the Tensor Scale (c.f., Section II-B:3) descriptor characterizing the chosen data set. In this case, OPF and KNN classifiers have statistically similar classification accuracies ($\cong 77\%$). This observation also holds for SVM-RBF classifier which is comparable to OPF and KNN.

B. Experiments – Round II

In the second round of experiments, we explore the Corel Relevants data set in order to verify our hypothesis on the importance of the (dis)similarity function used in the

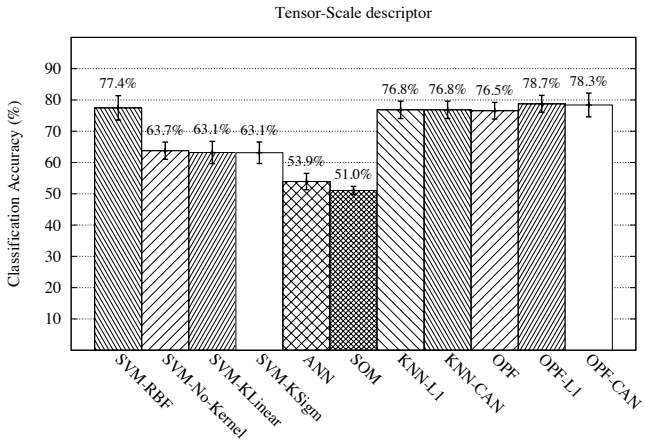

Figure 2. MPEG-7 CE Shape-1 Part-B 10-fold cross-validation average (μ) classification results and the corresponding confidence intervals with 95% confidence for **Tensor Scale** descriptor (c.f., Section II-B:3).

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 a simple SVM with no kernel. In addition, we have tested ANN and SOM classifiers as well as KNN and OPF with different metrics to show how this choice influences the final classification outcome. For this particular experiment, we have chosen the well-established color image descriptor BIC (c.f., Section II-B:6)). BIC descriptor codifies each input image with two histograms of border and interior pixels [25].

Table II shows a series of classifiers used with BIC descriptor characterizing the chosen data set. Recall that 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 have used KNN classifier fed with a feature vector with no dLOG embedded and L2 is the chosen comparison policy.

This experiment shows us that the powerful 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. This behavior allows SVM black-boxes to generalize gracefully. 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. Note that in

this experiment the simple classifiers KNN and OPF once more take advantage of simple tweaks in the comparison policies and produce very good results with no real increase in the classification computational time. KNN with a simple L1 metric over the embedded dLOG space reduces $\cong 31\%$ of the classification error when compared to SVM with a linear kernel fed with the same feature vector.

We also can observe that OPF classifier with an embedded dLOG space achieves $\cong 90\%$ of classification accuracy ($\cong 10\%$ classification error) while achieving $\cong 88\%$ classification accuracy using a simple 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. [27] 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 our hypothesis in this paper: it is very important to think about the classification choices taking place and how to compensate them. A bad decision can give us a terrible result (e.g., $\cong 55\%$ 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 at hands can achieve very good results (e.g., $\cong 90\%$ classification accuracy when using OPF and an embedded dLOG metric space).

C. Additional Experiments

Even though we do not show explicitly here, we have conducted several additional experiments with descriptors, classifiers and data sets to reinforce the hypothesis we draw in this paper. For instance, considering the BAS descriptor and the *ShapeMatcher*[8] data set, the out-of-the-box KNN classifier (which uses an L2 metric by default) has a classification error of $\epsilon \cong 26.9\% \pm 1.9\%$. However, if we change the comparison function to OCS, KNN has a classification error of only $\epsilon \cong 14.2\% \pm 1.2\%$, reducing the error in $\Delta \cong 47.2\%$.

In the same line, the out-of-the-box OPF classifier (which uses L2 metric by default) has a classification error of $\epsilon \cong 28.1\% \pm 1.83\%$. However, if we change the comparison function to L1, OPF reduces this error in $\Delta \cong 62\%$ achieving an error of $\epsilon \cong 11.4\% \pm 0.6\%$. Even more radical, when OPF classifier is adjusted to use OCS function as the comparison policy, it reduces the classification error in $\Delta \cong 77\%$ compared to its standard version with L2 metric.

[8]This data set comprises 2,688 shape images with 21 classes with 128 samples for each one and can be found at `http://www.cs.toronto.edu/~dmac/ShapeMatcher/`

Classifier	μ Acc.	σ	$\mathbf{t_{train}}$	$\mathbf{t_{test}}$
SVM-dLOG No-Kernel	79.13%	2.89%	3.72	0.01
SVM-dLOG Kernel Linear	**83.75%**	**2.85%**	**2670.83**	**0.06**
SVM-dLOG Kernel Sigmoid	83.54%	3.01%	55.20	0.06
SVM-dLOG Kernel RBF	83.16%	3.09%	68.40	0.06
SVM No-Kernel	72.33%	2.56%	20.90	0.01
SVM Kernel Linear	**76.41%**	**3.50%**	**56.74**	**0.05**
SVM Kernel Sigmoid	50.00%	0.01%	52.91	0.06
SVM RBF	50.17%	0.32%	69.20	0.06
ANN-dLOG	54.62%	1.45%	141.29	0.01
ANN	55.68%	1.57%	138.51	0.01
SOM-dLOG	63.42%	3.39%	272.19	0.35
SOM	58.38%	3.16%	196.09	0.32
KNN-dLOG-L1	**88.93%**	**1.50%**	**9.01**	**0.06**
KNN-dLOG-L2	71.24%	2.74%	9.08	0.08
KNN-dLOG-CAN	**88.93%**	**1.50%**	**8.96**	**0.07**
KNN-L1	84.82%	1.43%	8.33	0.06
KNN-L2	63.39%	2.02%	9.07	0.08
KNN-CAN	**84.82%**	**1.43%**	**9.04**	**0.07**
OPF-dLOG-L1	**89.75%**	**1.56%**	**0.06**	**0.01**
OPF-dLOG-L2	81.24%	2.94%	0.03	0.01
OPF-dLOG-CAN	85.00%	2.78%	0.06	0.01
OPF-L1	**87.83%**	**2.07%**	**0.06**	**0.01**
OPF-L2	73.39%	2.92%	0.03	0.07
OPF-CAN	87.20%	2.40%	0.07	0.01

Table II
COREL RELEVANTS 10-FOLD CROSS-VALIDATION AVERAGE (μ) AND STDEV (σ) CLASSIFICATION RESULTS FOR **BIC** DESCRIPTOR (C.F., SECTION II-B:6). ALSO SHOWN IN THE TABLE, ARE THE AVERAGE TRAINING ($\mathbf{t_{train}}$) AND TESTING ($\mathbf{t_{test}}$) TIMES FOR EACH FOLD, IN SECONDS.

IV. CONCLUSIONS AND FINAL THOUGHTS

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

We have focused on one aspect of the classifiers namely the (dis)similarity function used to compare examples in the feature space. We have shown how the knowledge about the classifier machinery lifts the results way beyond out-of-the-box ML solutions. For instance, using KNN classifier, BAS descriptor with the appropriate OCS metric, we are able to reduce the classification error in the MPEG-7 CE Shape-1 Part-B data set in $\epsilon \cong 23\%$ when compared to the out-of-the-box SVM classifier with a standard (non-tweaked) RBF kernel (c.f., Section III-A).

It is important to state that we are not advocating in favor of any classifier. On the contrary, we believe that each situation requires a proper problem understanding for the deployment of a good solution. The experiments we have performed reinforces our hypothesis in this paper: it is very important to think about the classification choices taking place and how to compensate them. A bad decision can give us a terrible result.

We should not select one classifier just because it is an hype and everyone is using it. Perhaps, it is interesting to use the so called out-of-the-boxes classifiers (with no

difficult parameter configuration) as a starting point towards the solution of the problem. After selecting a few classifiers, we need to think about what is going on within them in order to compensate some decisions. For instance, if we choose to represent the data at hands with an unbounded descriptor, perhaps we should choose a classifier which is not highly sensitive to this kind of data (c.f., Section III-B).

We also note that sometimes the feature descriptors have similar discrimination power. In these cases, more than on any other, the success of a classification task will depend on the proper choice of the (dis)similarity function.

Although in this paper we have focused in the choice of (dis)similarity function as comparison policies, we believe there are several other important choices we need to think of when solving a machine learning problem. We have just touched the tip of the iceberg. Therefore, our future work include the investigation of: (1) the impact of using a binary classifier to solve a multi-class classifier throughout binary combinations; (2) the impact of using features of with different dimensionality; (3) the parameter choice for modeling the kernels in kernel-based classifiers (e.g., SVMs); (4) the use of categorical variables with numerical ones; and finally (5) 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) Grant 2009/16206-1, University of Campinas PAPDIC Grant 519.292-340/10 and the Brazilian National Research Council (CNPq) for the financial support. We also thank the reviewers for the invaluable comments.

REFERENCES

[1] C. M. Bishop, *Pattern Recognition and Machine Learning*, 1st ed. Springer, 2006.

[2] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*. 2: Wiley-Interscience, 2000.

[3] C. Cortes and V. Vapnik, "Support-vector networks," *Springer Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[4] K.-R. Muller, S. Mika, G. Ratsch, K. Tsuda, and B. Scholkopf, "An introduction to kernel-based learning algorithms," *IEEE TNN*, vol. 12, no. 2, pp. 181–2002, Dec. 2001.

[5] A. Rocha, D. Hauagge, J. Wainer, and S. Goldenstein, "Automatic fruit and vegetable classification from images," *Elsevier COMPAG*, vol. 70, no. 1, pp. 96–104, Jan. 2010.

[6] A. Jain and R. C. Dubes, *Algorithms for Clustering Data*, 1st ed. Prentice-Hall, 1988.

[7] R. Baeza-Yates, *Clustering and Information Retrieval*, 1st ed. Kluwer Academic Publishers, 2003.

[8] G. Heidemann, "Unsupervised image categorization," *Elsevier IVC*, vol. 23, no. 10, pp. 861–876, Oct. 2004.

[9] M. Weber, "Unsupervised learning of models for object recognition," PhD Thesis, California Institute of Technology, Pasadena, United States, May 2000.

[10] X. Zhu, "Semi-supervised learning with graphs," Ph.D. dissertation, School of Computer Science, Carnegie Mellon, 2005.

[11] O. Chapelle, B. Scholkopf, and A. Zien, *Semi-supervised learning*. The MIT Press, 2006.

[12] S. Abney, *Semisupervised Learning for Computational Linguistics*. CRC Press, 2008.

[13] M. G. Quiles, L. Zhao, F. A. Breve, and A. Rocha, "Label propagation through neuronal synchrony," in *IEEE IJCNN*, 2010.

[14] P. Berkhin, *Grouping Multidimensional Data*. Springer, 2006, ch. A Survey of Clustering Data Mining Techniques.

[15] R. Torres and A. Falcão, "Content-based image retrieval: Theory and applications," *RITA*, vol. 13, no. 2, pp. 161–185, 2006.

[16] M. A. Gonçalves, E. A. Fox, and N. A. Kipp, "Streams, structures, spaces, scenarios, societies (5s): A formal model for digital libraries," *ACM TOIS*, vol. 22, no. 2, pp. 1–43, 2004.

[17] M. K. Hu, "Visual pattern recognition by moment invariants," *IEEE Trans. on Info. Theory*, vol. 8, no. 2, pp. 179–187, 1962.

[18] S. A. Dudani, K. J. Breeding, and R. B. McGhee, "Aircraft identification by moment invariants," *IEEE Trans. on Computers*, vol. C-26, no. 1, pp. 39–46, 1977.

[19] N. Arica and F. T. Y. Vural, "BAS: a perceptual shape descriptor based on the beam angle statistics," *Elsevier PRL*, vol. 24, no. 9-10, pp. 1627–1639, 2003.

[20] Y. P. Wang and T. Pavlidis, "Optimal correspondence of string subsequences," *IEEE TPAMI*, vol. 12, no. 11, pp. 1080–1087, 1990.

[21] P. K. Saha, "Tensor scale: A local morphometric parameter with applications to computer vision and image processing," *Elsevier CVIU*, vol. 99, no. 3, pp. 384–413, 2005.

[22] F. A. Andaló, P. A. V. Miranda, R. da S., Torres, and a. A. X. Falc "Shape feature extraction and description based on tensor scale," *Elsevier Pattern Recognition*, vol. 43, no. 1, pp. 26–36, 2010.

[23] R. S. Torres, A. X. Falcão, and L. F. Costa, "A graph-based approach for multiscale shape analysis," *Elsevier Pattern Recognition*, vol. 37, no. 6, pp. 1163–1174, 2004.

[24] C. T. Zahn and R. Z. Roskies, "Fourier descriptors for plane closed curves," *IEEE Trans. on Computer*, vol. c-21, no. 3, pp. 269–281, 1972.

[25] R. Stehling, M. Nascimento, and A. Falcão, "A compact and efficient image retrieval approach based on border/interior pixel classification," in *ACM CIKM*, 2002, pp. 102–109.

[26] G. Lance and W. Williams, "Computer programs for hierarchical polythetic classification ("similarity analysis')," *Oxford Computer Journal*, vol. 9, pp. 60–64, 1966.

[27] J. Papa, A. Falcão, and C. Suzuki, "Supervised pattern classification based on optimum-path forest," *Wiley IJIST*, vol. 19, no. 2, pp. 120–131, 2009.