Segment and combine: a generic approach for supervised learning of invariant classifiers from topologically structured data

Pierre Geurts P.Geurts@ulg.ac.be
Raphaël Marée Raphael.Maree@ulg.ac.be
Louis Wehenkel L.Wehenkel@ulg.ac.be
Department of Electrical Engineering and Computer Science (Montefiore Institute) & Centre for Biomedical Integrative Genoproteomics (CBIG), University of Liège, Sart-Tilman B28, B-4000 Liège, Belgium

Abstract

A generic method for supervised classification of structured objects is presented. The approach induces a classifier by (i) deriving a surrogate dataset from a pre-classified dataset of structured objects, by segmenting them into pieces, (ii) learning a model relating pieces to object-classes, (iii) classifying structured objects by combining predictions made for their pieces. The segmentation allows to exploit local information and can be adapted to inject invariances into the resulting classifier. The framework is illustrated on practical sequence, time-series and image classification problems.

1. Introduction

Let $\mathcal{X}$ be a space of inputs, $\mathcal{Y}$ a space of outputs, $P_{\mathcal{X}\times\mathcal{Y}}$ a probabilistic relation among inputs and outputs, and $\ell(y, y') : \mathcal{Y} \times \mathcal{Y} \to I\!R^+$ a loss function: optimal prediction of outputs consists of computing a function $f^*(\cdot) : \mathcal{X} \to \mathcal{Y}$ minimizing the expected loss $E_{P_{\mathcal{X}\times\mathcal{Y}}}\{\ell(f(x), y)\}$. The goal of supervised learning is to exploit a sample of joint observations $(x_i, y_i)_{i=1}^N \sim P^N_{\mathcal{X}\times\mathcal{Y}}$ of i.i.d input-output pairs, in order to define automatically an optimal prediction.

Most work in supervised learning builds upon the assumption that input and output spaces can be mapped into a vector space prior to learning and prediction, either explicitly by a given set of attributes (so-called propositional learning) or implicitly by kernelization. In many problems, however, inputs and/or outputs are complex data structures composed of elementary pieces of information which are related to each other in some meaningful way. To handle such applications by supervised learning it is thus desirable to develop frameworks exploiting these structures (Dietterich, 2000).

In this paper, we consider a family of problems where the output is a discrete class label, but the inputs are complex. We furthermore suppose that inputs are topologically structured, namely that they are composed of elementary pieces of similar nature related to each other by a neighborhood relation. While not covering all kinds of structured data learning problems, this problem-class includes a large number of practically relevant applications, such as string, time-series and image classification. We present within this context a simple and generic approach for supervised learning with invariances, which consists of segmenting complex inputs into pieces, applying supervised learning to samples of pieces, and carrying out predictions for complex inputs by combining predictions from their pieces. The paper synthesizes our earlier work in the three aforementioned domains.

The paper is organized as follows: Section 2 presents the Segment and Combine framework in general and discusses some of its intrinsic biases; Section 3 focuses on learning of invariances with this approach and Section 4 on some implementation specifics. Practical applications are discussed and illustrated in Section 5, and further work is mentioned in the Conclusions.

2. Algorithm description and rationale

2.1. Segment and combine: in abstracto

Let us denote by $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ a space of input output-pairs, by $(z_1, \ldots, z_N) \in \mathcal{Z}^N$ a sample of length N, and by $\mathcal{Z}^*$ the space of all possible finite length samples. A *hard* supervised learning algorithm is an algorithm $A^x_h(\cdot) : \mathcal{Z}^* \to \mathcal{Y}^{\mathcal{X}}$ computing from a sample a predictive model, ie a function $f^x_h(\cdot) : \mathcal{X} \to \mathcal{Y}$. Let $\pi_{\mathcal{Y}}$ denote a probability distribution over $\mathcal{Y}$ and $\Pi_{\mathcal{Y}}$ the space of all such distributions. A *soft* supervised learning algorithm is an algorithm $A^x_s(\cdot) : \mathcal{Z}^* \to (\Pi_{\mathcal{Y}})^{\mathcal{X}}$ computing from a sample a conditional probability model, i.e., a function $f^x_s(\cdot) : \mathcal{X} \to \Pi_{\mathcal{Y}}$.

Suppose that inputs have some structure which allows us to segment them in some way into pieces. Let us denote by $\mathcal{P}$ the space of all possible pieces of all possible inputs, by $p(x) = (p_i(x))_{i=1}^{k(x)} \in \mathcal{P}^{k(x)}$ the sequence of pieces obtained from some input[1]. Let $\mathcal{Q} = \mathcal{P} \times \mathcal{Y}$ and $A_s^p(\cdot) : \mathcal{Q}^* \to (\Pi_\mathcal{Y})^\mathcal{P}$ be a soft learning algorithm mapping any finite length sample $((p_1, y_1), \ldots, (p_N, y_N))$ of labelled pieces to a conditional probability model $f_s^p(\cdot) : \mathcal{P} \to \Pi_\mathcal{Y}$. Then the segment and combine idea consists of building a (hard) supervised learning algorithm $A_h^x(\cdot)$ in the following way:

1. *Segment* (exhaustively) each input-output pair (x_i, y_i) by extracting all pieces from x_i and labelling each one by y_i, yielding a sequence

$$((p_j(x_i), y_i))_{j=1}^{k(x_i)} \in \mathcal{Q}^{k(x_i)}. \quad (1)$$

   Concatenate all these latter and transform the resulting big sequence into a learning sample by:

$$ls = T\left((p_1(x_1), y_1), \ldots, (p_{k(x_N)}(x_N), y_N)\right), \quad (2)$$

   using the transformation $T(\cdot) : \mathcal{Q}^* \to \mathcal{Q}^*$.

2. Compute $A_s^p(ls)$ to get a soft model $f_s^p(\cdot)$ over $\mathcal{P}$.

3. *Combine* soft predictions from pieces by

$$f_h^x(x) = C\left(f_s^p(p_1(x)), \ldots, f_s^p(p_{k(x)}(x))\right) \quad (3)$$

   into hard predictions from inputs by using the combination operator $C(\cdot) : (\Pi_\mathcal{Y})^* \to \mathcal{Y}$.

There are thus four degrees of freedom, namely the segmentation $p(\cdot)$, the base learner $A_s^p(\cdot)$, the combination $C(\cdot)$, and the transformation $T(\cdot)$. We discuss below the first three operators in general. The transformation $T(\cdot)$ aims at explicitly introducing invariances and is discussed in Section 3.

2.2. Segmenting topologically structured data

Topologically structured data are data which are composed of elementary pieces related to each other by a neighborhood function. For example a graph (V, E) is composed of a set of vertices V which topological struture derives from the explicitly given set of edges E. Time-series data are composed of elementary samples which neighborhood is composed of the previous and next samples along the temporal axis, and similarly for strings over a finite alphabet. On the other hand, an image is composed of pixels which neighborhood derives directly from their spatial position in the image plane. Note that topological structure can also appear naturally at several levels, such as for example in text documents, composed of characters yielding strings, combined into sentences, paragraphs, etc.

Once a neighborhood function is given structuring our data, one can extract pieces from them by deriving from the (local) neighborhood function a (global) distance between elementary pieces, and for a fixed diameter extracting pieces of data within the diameter of each elementary object. For example, this allows to extract windows of fixed size from images, subsequences of fixed length from time-series or strings, etc. Depending on the relative size of the segmentation diameter and the diameter of the original data, this allows to extract pieces containing more or less global information about the original objects.

2.3. Segmentation by random sampling

If the inputs of the original dataset contain a large number of pieces, then exhaustively extracting all of them (as in Eq (1)) may lead to very large samples. For example, in image classification problems, the number of subwindows is typically of the order of the number of pixels, leading to subimage samples that may turn out to be 4 to 6 orders of magnitude larger than the original dataset of images.

To make learning tractable under such circumstances, we propose to use, instead of the exhaustive segmentation of Eq (1), a subset of specified size, by extracting from each input of the original dataset a fixed number of randomly chosen pieces. Along a similar line, we can also speed up the final prediction step (3), by using instead of all the pieces of the considered input only a randomly selected subset of specified size.

2.4. Kernel-based vs propositional base learner

The two main classes of generic supervised learning algorithms that have emerged in the literature are either propositional, in which case inputs to the learning algorithm have to be represented as a vector (of fixed length) of scalar attributes, or kernel-based, in which case one needs to define a kernel measuring the similarity of two inputs. These two types of methods have rather different algorithmic properties and their choice is thus also guided by practical constraints.

The main advantage of kernel-based methods is that by a proper choice of kernel they can be applied to a large variety of data structures without the necessity to

[1]The number of pieces may vary with x, and $p(x)$ is defined as a *sequence* (and not a *set*) to allow for multiple copies of a same piece and to preserve, at this level of generality, the order by which pieces are extracted. Thus $p(\cdot)$ is indeed a function mapping $\mathcal{X}$ to $\mathcal{P}^*$ (not to $2^\mathcal{P}$).

represent them in a finite dimensional attribute space. The advantage of propositional ones is that they can scale well with sample size, and also that they often allow to assess the importance of elementary attributes in terms of information content. Within the segment and combine approach, they can be applied to objects of variable size provided that the pieces can be represented by a vector of attributes of fixed size.

2.5. Combination by voting or averaging

We used in our applications a soft voting operator $C(\cdot)$ selecting the majority class from the average class probabilities of pieces. This is invariant with respect to permutations of pieces, and does not take into account their relative position. On the other hand, step (2) of the method targets a model $f_s^p(\cdot)$ predicting the class distribution of objects containing a certain piece, such that pieces which are found only in objects of a given class will get a high probability for this class. Therefore, (soft) voting over pieces is essentially equivalent to counting the number of pieces specific of each class, and selecting the most frequently detected one. This, admittedly, naive way of merging piecewise information yielded excellent results in our applications.

3. Imposing input-space invariances

3.1. Motivation

In many structured data problems one knows a priori that the optimal predictive model should be invariant with respect to some transformation of the inputs. For example, in many text classification problems the target relation is case insensitive and also insensitive to the insertion of white spaces etc. Similarly, in image classification, resolution and viewpoint variations are normally not supposed to change classes.

One way to define such input-space invariances consists of describing an *equivalence relation* $R \subset \mathcal{X} \times \mathcal{X}$ among inputs, and stating that the learning algorithm should return a function $f^*(\cdot)$ such that

$$\forall x, x' : (x, x') \in R \Rightarrow f^*(x) = f^*(x'). \quad (4)$$

Another way to define invariances consists of describing a *transformation operator* $O(\cdot) : \mathcal{X} \to \mathcal{X}$ over the input space, and stating that $f^*(\cdot)$ should satisfy

$$\forall x \in \mathcal{X} : f^*(O(x)) = f^*(x). \quad (5)$$

Depending on the application, it may be easier to specify invariances in one or the other way.

Invariances restrict the class of predictions $f(\cdot)$ that are compatible with prior knowledge about a learning problem, and may thus help to improve learning speed and accuracy. The stronger invariances are those that impose stronger restrictions on the hypothesis space. For example, in a classification problem the strongest possible invariance relation would be an invariance which number of equivalence classes is equal to the number of output classes. While such strong invariances are seldom available a priori, weaker ones are very often available and their exploitation can very significantly improve generalization.

3.2. Learning of invariant models

One standard way of exploiting input-space invariances is normalization. It consists essentially of normalizing the inputs in such a way that there is a one-to-one relationship between normalized inputs and invariant equivalence classes, i.e. by defining a transformation $N(\cdot) : \mathcal{X} \to \mathcal{X}'$, such that

$$\forall x, x' : (x, x') \in R \Leftrightarrow N(x) = N(x'). \quad (6)$$

This can be exploited in supervised learning by normalizing inputs prior to learning and prediction. In kernel methods, normalization can also be done implicitly by using a kernel which itself is invariant, i.e. such that

$$\forall x, x', x'' : (x, x') \in R \Rightarrow k(x, x'') = k(x', x''), \quad (7)$$

or

$$\forall x, x' : k(x, x') = k(x, O(x')). \quad (8)$$

Another generic way to inject invariances consists of adding transformed versions of the inputs to the sample. More precisely, if $\mathcal{O} = \{O_1, \ldots, O_\lambda\}$ is a set of invariant transformations, this consists of replacing each pair (x_i, y_i) of the learning sample by a collection of pairs obtained by

$$(x_i, y_i), (O_1(x_i), y_i), \ldots, (O_\lambda(x_i), y_i). \quad (9)$$

Note that if the set of invariant transformations is very large or even infinite, the enumeration can (or must) be replaced by random sampling.

3.3. Invariances of Segment and Combine

In the segment and combine approach, invariances result naturally from the choice of the piece-extraction operator $p(\cdot)$, which induces the following equivalence relation among objects

$$\forall x, x' : (x, x') \in R \Leftrightarrow p(x)p(x'), \quad (10)$$

which, indeed, is such that

$$(x, x') \in R \Rightarrow f_h^x(x) = f_h^x(x') \quad (11)$$

for any model in the form (3). This invariance can be further strengthened by assuming that the combination operator $C(\cdot)$ is itself invariant with respect to permutations of its argument (as it is the case for most aggregation or voting mechanisms). This leads to the equivalence relation (10) where equality is replaced by equality up to a permutation of components.

Furthermore, it is possible to inject invariances in an elegant way into the segment and combine approach at the learning stage, by combining segmentation together with a sample transformation operator $T(\cdot)$ to extract from an input-output pair a set of transformed pieces representative of the desired invariances (see Section 5.4, for an illustration of this idea in the context of image classification).

4. Specifics of Segment and Combine

4.1. Ensemble-based base learners

Ensemble methods aggegate predictions of an ensemble of models induced from a learning sample by injecting some variance, e.g., by randomizing the algorithm and/or by changing the learning sample from one model to the other. Typically, the larger the number of models the more accurate the final prediction.

In the context of segment and combine, the use of ensemble methods thus leads to a double voting scheme: one over models, and the other over pieces. Also, since the segment and combine may introduce randomization by subsampling of pieces, one can imagine various ways of interlacing the sampling of models and the sampling of pieces, and thereby further improve the efficiency/accuracy tradeoff of the method.

4.2. Piece extraction and diameter tuning

At the prediction step, it is possible to extract the most relevant (class-specific) pieces from an object by sorting them by increasing order of the entropy of the predicted probability distribution. This allows one to gain further insight into the way a particular object is classified (see §§5.3-5.4 for illustrations).

For the approach to work well the diameter of the pieces generally needs to be adjusted to the problem at hand. There are two alternative ways to make sure that pieces of appropriate size are used: one consists of using a cross-validation wrapper to identify an optimal diameter from the data (see §§5.2-5.3 for examples); the other way consists of randomizing the diameter of pieces so as to inject scale invariance into the resulting classifier (see §5.4).

5. Illustrations

We illustrate the proposed framework on three classes of topologically structured data, namely strings, time-series, and images. Our presentation is mostly borrowed from (Geurts et al., 2005), (Geurts & Wehenkel, 2005), and (Marée et al., 2005b) respectively. The interested reader may refer to these publications for more details. These applications share some commonalities that we summarize first.

5.1. Commonalities

First, we used a piece extraction mechanism that randomly choses a fixed number of pieces from each object, as mentioned in section 2.3. Note that these pieces were described in terms of raw input attributes without any feature selection or extraction.

Second, piece-classifiers were learned using propositional base learners in the form of ensemble of decision tree models, such as Tree-Boosting, Tree-Bagging and Extra-Trees (see Geurts et al., 2006).

Third, we used a simple combination mechanism which merely averages the soft predictions derived from the elementary pieces. In particular, to make a prediction for a new object with an ensemble of trees grown from pieces, each piece is simply propagated into each tree of the ensemble. Each tree outputs conditional class probability estimates for each piece. Each piece thus receives M vectors of class probability estimates, where M denotes the number of trees in the ensemble. All the predictions are then averaged and the class corresponding to the largest aggregated probability estimate is assigned to the structured object.

5.2. String classification

Symbolic sequence (string) classification is a rather generic problem which appears for instance in text categorization, computer-user modeling, automatic classification of alarm logs, and intrusion detection. One domain where discrete sequences are especially frequent is biology. In the context of bioinformatics, automatic sequence classification can be applied both to genomic data (DNA or RNA strings) as well as to proteomic data (strings of amino-acids), and there is a multitude of applications ranging from fast database search to the identification of patterns of some specific physical properties.

The most frequent approach to handle symbolic sequences is to derive from them a (potentially very large) number of candidate atributes and then use these as inputs for standard supervised learning meth-

ods (possibly in combination with attribute selection techniques). Different sets of attributes have been proposed for biological sequences (Hu et al., 2000; Simonis et al., 2004; Saeys et al., 2004). Another approach is to define a similarity measure between sequences and then exploit this similarity in k-nearest neighbor or other kernel-based methods. Again, to be effective, this similarity measure has to be adapted to the problem at hand (see e.g., Lodhi et al., 2002, for texts, Wang et al., 1999; Vert et al., 2004, for biological sequences).

In this section, we illustrate the segment and combine approach on two datasets of DNA sequences considered in (Geurts et al., 2005):

- Splice (Blake & Merz, 1998): the goal of this problem is to recognize DNA coding regions. The dataset contains 3190 sequences of 60 nucleotides classified into three classes: non coding sequences, sequences centered at the junctions between coding and non coding regions, and sequences centered at the junctions between non coding and coding regions.
- MSN2 (Simonis et al., 2004): the goal of this problem is to recognise genes that are targeted by the same transcription factor. The dataset contains 112 sequences of length 800, half of them targeted by the MSN2 transcription factor.

We applied on these two problems the instance of the segment and combine framework described above. Pieces are defined in this case by all subsequences of a given length of an input sequence. We randomly extracted 20,000 of them from learning sequences and we classified them with ensembles of 50 Extra-Trees (Geurts et al., 2006). To make a prediction, we extracted all possible pieces from the tested sequences and averaged class probability estimates for these pieces. Error rates were estimated by holdout of 1190 sequences on Splice and by 10-fold cross-validation on MSN2. Figure 1 shows the evolution of the error with the diameter of the pieces ranging from 5 to the full length of the sequence in both cases. These two curves illustrate the importance of the tuning of the diameter parameter. On Splice, the optimal value corresponds to the maximal sequence length for this problem while on the MSN2 problem, the optimal value corresponds to the smallest piece length. Hence, by the adaptation of the diameter, the method can handle as well problems with highly centered data as problems characterized by local position independent patterns. In both cases, the segment and combine approach proves very effective with respect to a standard method using

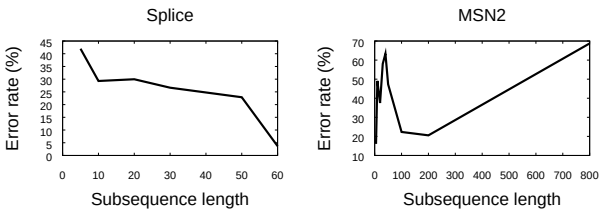

Figure 1. Evolution of the error with the subsequence length, left on Splice, right on MSN2

as input variables the number of occurences of all possible n-grams of a given length (Geurts et al., 2005). On MSN2, we also obtain good results with respect to a feature extraction method taking into account biological aspects (Simonis et al., 2004).

Another interesting characteristic of the segment and combine approach for this application is that it allows one to retrieve discriminant patterns from sequences from the piece classifier. In (Geurts et al., 2005), we provide a procedure exploiting the subsequence classifiers derived by supervised learning in order to identify the subsequences which are highly specific of a given class, and at the same time occur frequently in sequences of this class.

5.3. Time-series classification

Time-series classification is at the same time an important problem, from the viewpoint of its multitudinous applications, and a difficult one from the viewpoint of machine learning methodology. Specific applications of time-series classification concern the non intrusive monitoring and diagnosis of processes and biological systems, for example to decide whether the system is in a healthy operating condition on the basis of measurements of various signals. Other relevant applications concern speech recognition, behavior analysis, in particular biometrics and fraud detection, which can all be settled in the form of time-series classification problems.

Like for discrete sequences, the most common approach to solve this problem is to define a (possibly very large) collection of temporal predicates (or patterns) which can be applied to each time-series in order to compute features (logical or numerical scalar attributes) which are then used as input representation to a base learner (Kadous & Sammut, 2005; Kudo et al., 1999; Mierswa & Morik, 2005; Olszewski, 2001; Saito, 1994; Zhang et al., 2004). A related approach is to incorporate directly the temporal feature extraction step into the learning algorithm (Alonso González & Rodríguez Diez, 2004; Geurts, 2001; Yamada et al.,

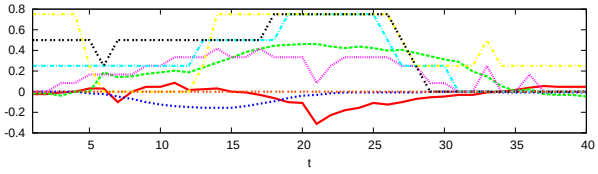

Figure 2. An instance of the Auslan-s problem

2003). Another approach is to define a distance measure between time-series that takes into account temporal specific peculiarities (e.g. invariance with respect to time or amplitude rescaling) and then to use this distance measure in combination with nearest neighbors or other kernel-based methods (Ratanamahatana & Keogh, 2004; Shimodaira et al., 2001). A potential advantage of these approaches is the possibility to bias the representation by exploiting prior problem specific knowledge, by defining a dictionary of application specific patterns, or by using an ad hoc similarity measure. At the same time, this problem specific modeling step makes the application of machine learning non autonomous.

In (Geurts & Wehenkel, 2005), we have applied the segment and combine method on 10 benchmark time-series problems and obtained results competitive with state-of-the-art algorithms from the literature without any specific problem adaptation. As an illustration, one of these datasets, Auslan-s (Kadous, 1999), collects 200 recordings of the Australian sign language using instrumented gloves. Each time-series measures the temporal evolution of 8 variables on variable length intervals and corresponds to one out of 10 words (see Figure 2 for one example). We applied to this problem the segment and combine method with ensembles of 100 Extra-Trees and a random sample of 10000 pieces of fixed length. The length of the pieces was optimized in this case by 10-fold cross-validation. This yields an error rate of 1.0% (estimated by another external run of 10-fold cross-validation) which turns out to be slightly better than the best previously published error rate on this problem (1.5% in Kadous, 1999). As a comparison, the use of Extra-Trees with a simple normalization technique transforming a time-series into a vector of fixed dimensionality yields a much higher error of 4.5%.

As another illustration of the interpretability of the segment and combine method, Figure 3 shows, in the top part, two time-series from an artificial problem with two classes, and, in the bottom part, the evolution of the probabilities of the two classes as predicted for subseries (of 50 time points) as they move progressively from left to right on the time axis. Class 1 differs from class 2 only by the occurence of a sharp peak (around $t = 75$ in these examples). From the probability plots, we see that, most of the time, the two classes are equally likely, but at the time where the peak appears ($t \in [60-70]$) the probability of class 1 increases for the left series (where a peak appears) and decreases for the right series (where no peak appears). Notice that the voting scheme used to classify the whole time-series from its subseries amounts to integrating these curves along the time axis and deciding on the most likely class once all subseries have been incorporated. This suggests that, once a subseries classifier has been trained, the segment combine approach can be used in real-time in order to classify signals through time.

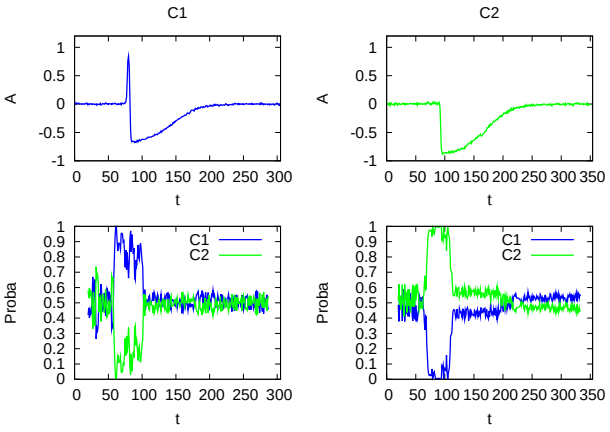

Figure 3. Illustration of intepretability on a time-series problem

Note that, although all time-series considered in (Geurts & Wehenkel, 2005) were numerical, the segment and combine method can also handle time-series mixing numerical and symbolic data provided that the used base learner can handle them.

5.4. Image classification

Given a set of training images labelled into a finite number of classes, the goal of an automatic image classification method is to build a classifier that will be able to predict accurately the class of new, unseen images. This problem is very challenging as there exist a lot of very different images classes, as illustrated in Figure 4, and real-world images could be taken under various viewing conditions due to scale, viewpoint, rotation, and illumination changes, as well as partial occlusions and cluttered backgrounds.

In the computer vision literature, a number of approaches extract from images some local patches using region or interest point detectors (Mikolajczyk &

Figure 4. Examples of types of images tackled with the segment and combine framework

Schmid, 2005). These patches are then described by an attribute vector after some dimensionality reduction (Mikolajczyk et al., 2005), and often injected into nearest neighbor or kernel-based algorithms. The instance of the segment and combine framework that we evaluated comes within the scope of these approaches. It was successfully applied to the recognition of images representing different kinds of objects, handwritten digits, faces, buildings, photographs, x-rays, ...

More precisely, the method presented in (Marée et al., 2005b) extracts a large number of possibly overlapping, square subwindows of random sizes and at random positions from images. Each subwindow size is randomly chosen between 1×1 pixels and the minimum horizontal or vertical size of the current image. The position is then randomly chosen so that each subwindow is fully contained in the image. Robustness to various viewing conditions was handled as follows. To be robust to scale changes, subwindows are simply transformed to a fixed size. Note that this transformation also allows to use propositional base learners that work with fixed-size feature vectors, even if images are of different sizes (which is often the case). The resized subwindows are then transformed to a HSV color space that tends to limit the effects of practically occurring illumination changes. Regarding rotation changes, a variant of the method was also introduced where subwindows are randomly rotated before resizing them. Indeed, it encourages the learning algorithm to produce models which will classify subwindows extracted from rotated versions of an image in the same way. Other transformations could be considered, for example to deal with perspective distortion due to large viewpoint changes. Notice that randomizing the size and the orientation of the extracted subwindows is essentially equivalent to extracting windows of fixed size and orientation from scaled and/or rotated images, and so imposes classifier invariance with respect to these transformations. Note also that ensuring scale invariance by window-size randomization results in the fact that there is no need here to tune the window size to problem specifics.

Like in other applications, we have observed that accuracy of the approach is a monotonically increasing function of the number of pieces used in the learning and prediction stages. Using surrogate datasets of about one hundred thousand subwindows for learning was however sufficient to produce state-of-the-art results on most image datasets, while only one hundred random subwindows needed to be extracted from each test image.

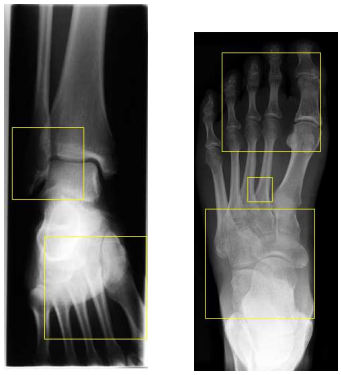

Figure 5. Subwindows with the highest number of correct votes for two images (from classes ankle joint, and foot)

Here also, one can use the method to identify relevant pieces of a test image. Indeed, among the subwindows, those which contribute to the correct classification of one image are those which receive a high probability of the majority class. For example, in medical applications, this functionality could be very helpful if the goal of the image classication task is to detect and classify diseased regions. The most relevant regions for a given class could then be shown to experts for further analysis. To illustrate this, we show in Figure 5, taken from (Marée et al., 2005a), some examples of subwindows receiving a high number of correct votes for two images taken from the IRMA database[2] containing 10000 images pre-classified into 57 classes corresponding to various imaging modalities and directions, body parts, and biological systems examined.

5.5. Computational complexity considerations

In order to let the reader appreciate whether this approach is applicable to large scale datasets, we discuss briefly its computational complexity.

The complexity of the preprocessing, piece extraction step obviously depends on the particular transformation T that is applied. In the case of our applications,

[2]http://www.irma-project.org/, courtesy of TM Lehmann, Dept. of Medical Informatics, RWTH Aachen, Germany.

it is linear in the product of the number of original training objects, the number of pieces extracted per object, and the size of the pieces, both in terms of computing time and in terms of memory requirements.

The computational complexity of the training step strongly depends on the type of base learner that is used. In the case where Extra-Trees are used, as in the above applications, the computational complexity is typically (assuming the trees are not too unbalanced) on the order of $nMN_p \log N_p$, where n denotes the number of attributes used to represent pieces, M is the number of ensemble terms, and N_p is the total number of pieces used in the training set. Note that this means that the algorithm is highly scalable, especially considering the fact that multiple trees may be built in parallel, allowing a theoretical speed up possibility by a factor of M.

The computational complexity of the prediction step is also base-learner dependent. With the Extra-Trees method it is on the order of $n_p Md$, where n_p denotes the number of pieces extracted from an object, M the number of trees, and d the average tree depth (which is on the order of $\log N_p$ for not too unbalanced trees). Here the possible speed by parallel computations is even much better, since individual trees and individual windows may be classified in parallel, which would lead to a theoretical speed-up by a factor of $n_p M$.

To fix ideas, let us provide some CPU times for the large scale image classification problem discussed in (Marée et al., 2005a). In this application, the original training set contained 9000 512×512 grey-level images classified into 57 classes. From these, we extracted a total of 800,000 subwindows of random size renormalized to 16×16 and represented by an attribute vector of size 256. Training an ensemble of 25 Extra-Trees on this huge dataset took about 18 hours of CPU time on a standard 2.4 GHz Pentium PC with 2GB RAM (implentation in C, running under LINUX). For testing, it took 1125 seconds to predict classes of all the 1000 test images with these trees, while using $n_p = 500$ (i.e., about 1 s per image).

6. Conclusions

In this paper we have presented a generic Segment and Combine framework for supervised learning of classifiers of complex, topologically structured data. While presently limited to classification problems which can be solved by detecting and counting the occurrences of class-specific local patterns, the method has already been extensively studied and validated on a large number of string, time-series, and image classification problems, where excellent results with respect to the state-of-the-art have been obtained. The contribution of this paper, with respect to our previous work, was to present and discuss this method in general and highlight its intrinsic biases and strengths. In particular, the fact that it allows to inject invariances into the learned models appears as an important feature.

Among the many possible directions for further work, we first mention the combination of segment and combine with boosting, which could lead to more efficient and more accurate classifiers. On the other hand, the extension to other types of problems, such as time-series forecasting and image segmentation would be quite useful. More generally, the exploitation of other types of loss-functions, in particular kernel-based similarities, could help to generalize the method to learning with structured outputs. We also believe that further work aiming at the definition of more sophisticated or alternative combination operators, with different semantics could allow to further extend the range of applications covered by the approach.

Acknowledgements

Pierre Geurts is a scientific research worker of the F.N.R.S. (Belgium); at the time of writing this paper he was on leave at the University of Evry (IBISC, Prof. F.d'Alché-Buc) with support from the C.N.R.S. (France). Raphaël Marée is supported by the GIGA interdisciplinary cluster of genoproteomics of the University of Liège.

References

Alonso González, J., & Rodríguez Diez, J. J. (2004). Boosting interval-based literals: Variable length and early classification. In M. Last, A. Kandel and H. Bunke (Eds.), *Data mining in time series databases*. World Scientific.

Blake, C., & Merz, C. (1998). UCI repository of machine learning databases. http://www.ics.uci.edu/∼mlearn/MLRepository.html.

Dietterich, T. G. (2000). The divide-and-conquer manifesto. *Algorithmic Learning Theory 11th International Conference* (pp. 13–26).

Geurts, P. (2001). Pattern extraction for time-series classification. *Proceedings of PKDD 2001, 5th European Conference on Principles of Data Mining and Knowledge Discovery* (pp. 115–127). Freiburg: Springer-Verlag.

Geurts, P., Blanco Cuesta, A., & Wehenkel, L. (2005). Segment and combine approach for biological sequence classification. *Proc. IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology* (pp. 194–201).

Geurts, P., Ernst, D., & Wehenkel, L. (2006). Extremely randomized trees. *Machine Learning*, Advance access: DOI 10.1007/s10994-006-6226-1, 40 pages.

Geurts, P., & Wehenkel, L. (2005). Segment and combine approach for non-parametric time-series classification. *Proc. of the 9th European Conference on Principles and Practice of Knowledge Discovery in Databases*.

Hu, Y., Sandmeyer, S., McLaughlin, C., & Kibler, D. (2000). Combinatorial motif analysis and hypothesis generation on a genomic scale. *Bioinformatics*, *16*, 222–232.

Kadous, M. W. (1999). Learning comprehensible descriptions of multivariate time series. *Proceedings of the Sixteenth International Conference on Machine Learning, ICML'99* (pp. 454–463). Bled, Slovenia.

Kadous, M. W., & Sammut, C. (2005). Classification of multivariate time series and structured data using constructive induction. *Machine learning*, *58*, 179–216.

Kudo, M., Toyama, J., & Shimbo, M. (1999). Multidimensional curve classification using passing-through regions. *Pattern Recognition Letters*, *20*, 1103–1111.

Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., & Watkins, C. (2002). Text classification using string kernels. *Journal of Machine Learning Research*, *2*, 419–444.

Marée, R., Geurts, P., Piater, J., & Wehenkel, L. (2005a). Biomedical image classification with random subwindows and decision trees. *Proc. ICCV workshop on Computer Vision for Biomedical Image Applications* (pp. 220–229). Springer-Verlag.

Marée, R., Geurts, P., Piater, J., & Wehenkel, L. (2005b). Random subwindows for robust image classification. *Proc. of the IEEE International Conference on Computer Vision and Pattern Recognition* (pp. 34–40).

Mierswa, I., & Morik, K. (2005). Automatic feature extraction for classifying audio data. *Machine Learning*, *58*, 127–149.

Mikolajczyk, K., & Schmid, C. (2005). A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, *27*, 1615–1630.

Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., Kadir, T., & Gool, L. V. (2005). A comparison of affine region detectors. *International Journal of Computer Vision*, *65*, 43–72.

Olszewski, R. T. (2001). *Generalized feature extraction for structural pattern recognition in time-series data.* Doctoral dissertation, Carnegie Mellon University, Pittsburgh, PA.

Ratanamahatana, C., & Keogh, E. (2004). Making time-series classification more accurate using learned constraints. *Proceedings of SIAM*.

Saeys, Y., Degroeve, S., Aeyels, D., Rouzé, P., & Van de Peer, Y. (2004). Feature selection for splice site prediction: a new method using eda-based feature ranking. *BMC Bioinformatics*, *5*, 64.

Saito, N. (1994). *Local feature extraction and its application using a library of bases.* Doctoral dissertation, Department of Mathematics, Yale University.

Shimodaira, H., Noma, K., Nakai, M., & Sagayama, S. (2001). Dynamic time-alignment kernel in support vector machine. *Advances in Neural Information Processing Systems 14, NIPS2001* (pp. 921–928).

Simonis, N., Wodak, S., Cohen, G., & van Helden, J. (2004). Combining pattern discovery and discriminant analysis to predict gene co-regulation. *Bioinformatics*, *20*, 2370–2379.

Vert, J.-P., Saigo, H., & Akutsu (2004). *Kernel methods in computational biology*, chapter Local alignment kernels for biological sequences. MIT Press.

Wang, J. T.-L., Rozen, S., Shapiro, B. A., Shasha, D., Wang, Z., & Yin, M. (1999). New techniques for DNA sequence classification. *Journal of Computational Biology*, *6*, 209–218.

Yamada, Y., Suzuki, E., Yokoi, H., & Takabayashi, K. (2003). Decision-tree induction from time-series data based on standard-example split test. *Proceedings of the 20th International Conference on Machine Learning (ICML-2003)*.

Zhang, H., Ho, T., & Lin, M. (2004). A non-parametric wavelet feature extractor for time series classification. *Proceedings of the 8th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)* (pp. 595–603).