Noname manuscript No.
(will be inserted by the editor)

Generalized Multi-scale Stacked Sequential Learning for Multi-class Classification

Eloi Puertas, Sergio Escalera, Oriol Pujol

the date of receipt and acceptance should be inserted later

Abstract In many classification problems, neighbor data labels have inherent sequential relationships in spite of their values. Sequential learning algorithms take benefit of these relationships in order to improve generalization.In this paper, we revise the Multi-Scale Sequential Learning approach (MSSL) for applying it in the multi-class case (MMSSL). We introduce the Error-Correcting Output Codes (ECOC) framework in the MSSL classifiers and propose a formulation for calculating confidence maps from the margins of the base classifiers. In addition, we propose a MMSSL compression approach which reduces the number of features in the extended data set without a loss in performance. The proposed methods are tested on several databases, showing significant performance improvement compared to classical approaches.

1 Introduction

Standard classification tasks commonly assume that samples are independently and identically drawn from a distribution (i.i.d) of samples $\mathbf{X}$ and their labels Y. However, classification problems in real world databases can break this i.i.d. assumption. For example, consider the case of object recognition in image understanding. In this case, if one pixel belongs to a certain object category, it is very likely that neighboring pixels also belong to the same object, with the exception of the borders.

In this scope, sequential learning [3] takes benefit of these relationships making easier the classification task. Here, the training data consists of sequences of pairs $(\mathbf{x}, y)$ where neighboring examples on a support lattice display some kind of coherence. Usually, sequential learning applications consider one-dimensional support lattice, i.e. when data samples belong to a sequence of text, sound or time. Some examples of applications where this kind of support lattices appears are: speech recognition, activity or gesture recognition

Address(es) of author(s) should be given

form motion data, stock market prediction, etc. This kind of relationship is also frequent in 2-D images (two-dimensional support lattice), volume images or videos (three-dimensional support lattice), and multiple sensor data (multi-dimensional support lattice).

In the literature, sequential learning has been addressed from different perspectives. From the point of view of graphical models, Hidden Markov Models (HMM) or Conditional Random Fields (CRF) [10,7,4,15,23] are used for inferring the joint or conditional probability of the sequence. Another point of view is to use graph transformer networks. In [24], a graph is used to represent segmentation hypotheses for an image representing a sequence of digits. Therefore, the input and output are considered as a graph. Then, it looks for the transformation that minimizes a loss function of the training data using a Neural Network. From the point of view of meta-learning, sequential learning has been addressed by means of sliding window techniques, recurrent sliding windows [3], or stacked sequential learning (SSL) [6]. In the case of SSL, the meta-learning scheme is as follows: first a base classifier is used over the samples to produce predictions. Then, a window among the predictions is applied and it is concatenated with the original data, building an extended dataset. Finally, a second base classifier predicts the final output from the extended dataset. In [16], it is identified that the main step of the relationship modeling is how this extended dataset is created. In consequence, a general framework for the SSL called Multi-scale Stacked Sequential Learning (MSSL) is formalized, where a multi-scale decomposition is used in the relationship modeling step, showing large improvement with respect to base SSL.

Usually, applications considered need classifiers that are able to deal with multiple classes. However, in the case of sequential learning, few of the previous approaches are able to deal with the multi-class case. One case of multi-class extension is the CRF using graph-cut with alpha-expansion [13]. Another approach is to decompose the multi-class problem into a set of binary-class problems and combine them in some way. In this sense, the Error-Correct Output Codes (ECOC) [2] framework is a well-studied methodology that is used to transform multi-class problems to an ensemble of binary classifiers. The fundamental issues here are: how this decomposition can be done in an efficient way, and how a final classification can be obtained from the different binary predictions. In the ECOC framework, these two issues are defined as coding and decoding phases in a communication problem. During the coding phase a codeword is assigned to each label in the multi-class problem. Each bit in the codeword identifies the membership of such class for a given binary classifier. The most used coding strategies are the *one-versus-all* [5], where each class is discriminated against the rest and *one-versus-one* [1], which splits each possible pairs of classes. The decoding phase of the ECOC framework is based on error-correcting principles, where distances measurements between the output code and the target codeword are the strategies most frequently applied. Among these, Hamming and Euclidean measures are the most used [14].

In this paper, we propose an efficient extension of MSSL to the multi-class case (MMSSL) based on the ECOC framework. We revise the general stacked

sequential learning scheme adapting it to both binary-class and multi-class problems. The main drawback of MMSSL is that the number of features in the extended set linearly increases with the number of classes. As a consequence, a novel feature compression approach for mitigating this problem is presented. Experiments on several databases are performed, including images, text, and sensor data, showing high classification accuracies and better performance than classical approaches. The terminology used during the rest of the paper is summarized in Table 1.

The paper is organized as follows: In section 2, we review the original MSSL for the binary-class case and the ECOC framework. In section 3, we adapt the MSSL steps to the multi-class case (MMSSL) and face the problem of feature cardinality in the extended set by means of a compression approach. Experiments and results of our methodology are shown in Section 4. Finally, Section 5 concludes the paper.

2 Background

This work holds on two frameworks, one for capturing the sequential relationship among samples and the other for facing multi-class classification problems. The former is the Multi-scale Stacked Sequential Learning [16] which is a generalization of the Stacked Sequential Learning [6]. The later is the ECOC framework, which is a general approach to reduce multi-class problems to an ensemble of binary classifiers. Each of these methodologies are explained in detail below.

2.1 Multi-scale Stacked Sequential Learning

Sequential learning assumes that samples are not independently drawn from a joint distribution of the data samples $\mathbf{X}$ and their labels Y. Thus, the training data is considered as a sequence of pairs example and its label $(\mathbf{x}, y)$, such that neighboring examples exhibit some kind of relationship [3]. In [6], an approach of sequential learning that uses a meta-learning framework [9] is presented. Basically, the Stacked Sequential Learning (SSL) scheme is as follows: first, a base classifier is trained and tested with the original data. Then, an extended data set is created which joins the original training data features with the predicted labels produced by the base classifier considering a window around the example. Finally, another classifier is trained with this new feature set. Here the relationship between pairs $(\mathbf{x}, y)$ is expressed by this new feature set. The main drawback of this approach is that the width of the window around the sample determines the maximum length of interaction among samples. Therefore, the longer the window, the further the interaction considered, but also the extended data set is increased in terms of features. This makes this approach not suitable for problems that have more than one dimension sequential relationships. In [3], the main problems of sequential learning are highlighted:

List of Symbols

$\mathbf{X}$	set of samples
Y	set of labels
$\mathbf{x}$	a sample
y	a label
$h(\mathbf{x})$	a classifier
y'	a prediction from a classifier
y''	a final prediction from a chain of classifiers
$\mathbf{x^{ext}}$	extended set
J	neighborhood relationship function
z	neighborhood model features
ρ	neighborhood
θ	neighborhood parameterization
w	number of elements in the neighborhood window
s	number of scales
c	set of different classes in a multi-class problem
$\hat{F}(\mathbf{x}, \mathbf{c})$	a prediction confidence map
N	number of classes in a multi-class problem
n	number of dichotomizers
σ	parameter of a Gaussian filter
Σ	set of scales defined by σ parameters
b	a dichotomizer
M	ECOC coding matrix
$\mathcal{Y}$	a class codeword in ECOC framework
$\mathcal{X}$	a sample prediction codeword in ECOC framework
m_x	margin for a prediction of sample $\mathbf{x}$
β	constant which governs transition in a sigmoidean function
t	number of iterations in an ADAboost classifier
δ	a soft distance
α	normalization parameter for soft distance δ
g^σ	a multidimensional isotropic gaussian filter with zero mean and σ standard deviation
$\mathcal{P}$	a set of partitions of classes
P	a partition of groups of classes
γ	a symbol in a partition codeword
Γ	a partition codeword
R	the mean ranking for each system configurations
E	the total number of experiments
k	the total number of system configuration
χ^2_F	Friedman statistic value

Table 1 Terminology used in this contribution.

(a) How to capture and exploit sequential correlations; (b) how to represent and incorporate complex loss functions; (c) how to identify long-distance interactions; and (d) how to make sequential learning computationally efficient. In [16], all these points are specifically analyzed and it is proposed a generalization of the SSL [6] by emphasizing the key role of neighborhood relationship modeling. For this aim, a block J is included in the pipeline of the basic Sequential Stacked Learning as shown in Figure 1. Therefore, the Generalized Stacked Sequential Learning process is as follows: A classifier $h_1(\mathbf{x})$ is trained with the input data set $(\mathbf{x}, y)$ and the set of predicted labels y' is obtained.

Next block defines the policy for creating the neighborhood model of the predicted labels. It is represented by $z = J(y', \rho, \theta) : \mathcal{R} \rightarrow \mathcal{R}^w$, where J is a function that captures the data interaction with a model parameterized by θ in a neighborhood ρ. Then, the output of $J(y', \rho, \theta)$ is joined with the original training data creating the extended training set $(\mathbf{x^{ext}}, y) = ((\mathbf{x}, z), y)$. This new set is used to train a second classifier $h_2(\mathbf{x^{ext}})$ with the aim of producing the final prediction y''.

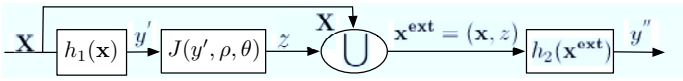


Fig. 1 Generalized Stacked Sequential Learning.

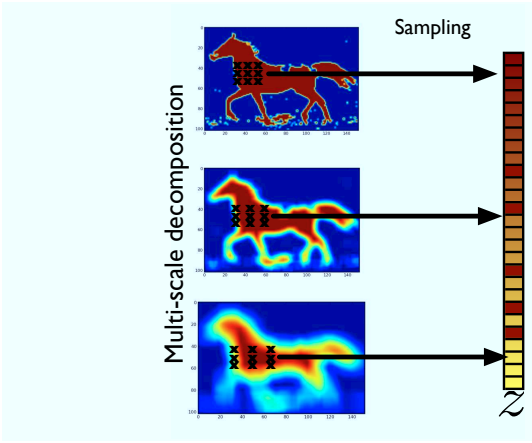


Fig. 2 Example of $J(y', \rho, \theta)$ function. Multi-scale decomposition and sampling producing the output z for a particular sample of the predicted labels image.

In [16], Multi-scale stacked sequential learning (MSSL) is presented where function $J(y', \rho, \theta)$ uses a multi-scale decomposition [11]. This function is proved to be effective in several domains (1D and 2D sequential relationships). It consists of two steps: first the multi-scale decomposition answers how to model the relationship among neighboring locations, and second, the sampling that answers how to define the support lattice to produce the final set z. Figure 2 shows an example of multi-scale decomposition of an image of predicted labels. In this case a gaussian filter is used for the multi-scale decomposition, increasing the σ parameter of the gaussian in each scale. Also, it shows the pattern sampling around an example. This pattern can be represented by a set of displacement vectors that defines the neighborhood. Each vector is also increased proportionally to the σ parameter in each scale. The vector z resulting of this function is a $w \times s$-dimensional value, where w is

the number of elements in the support lattice of the neighborhood ρ and s express the number of scales used in the multi-scale decomposition. In the case of defining the neighborhood by means of a window, w is the number of elements in the window. This approach is able to capture sequential relationships among data, as well as to capture long-range interactions in the label field.

2.2 Error-Correcting Output Codes

Error-Correcting Output Codes (ECOC) are a general framework to combine binary problems to address the mutli-class problem [2,14]. ECOC framework consists of two phases: a coding phase, where a codeword is assigned to each class of a multi-class problem, and a decoding phase, where, given a test sample, it looks for the most similar class codeword. Originaly [2], a codeword was a sequence of bits represented by $\{-1, +1\}$, where each bit identifies the membership of the class for a given binary classifier (dichotomizer). Afterwards [1], a third symbol (the zero symbol) was introduced, which means that a particular class is not considered by a given classifier. Given a set of N classes to be learned in an ECOC design, n different bipartitions (groups of classes) are formed, and n dichotomizers over the partitions are trained. As a result, a codeword $\mathcal{Y}_c, c \in [1, \ldots, N]$ of length n is obtained for each class c. Arranging the codewords as rows, a coding matrix $M \in \{-1, 0, +1\}^{N \times n}$ is defined. The most used coding strategy is the *one-versus-all* [5], where each class is discriminated against the rest, obtaining a codeword of length equal to the number of classes. Figure 3 shows an example of *one-versus-one* coding matrix, which considers all possible pairs of classes, with a codeword length of $\frac{N(N-1)}{2}$. The matrix is coded using ten dichotomizers $\{b_1, \ldots, b_{10}\}$ for a 5-class problem. The white regions are coded by 1 (considered as one class by the respective dichotomizer b_j, the dark regions by -1 (considered as the other class), and the gray regions correspond to the zero symbol (classes that are not considered by the respective dichotomizer b_j).

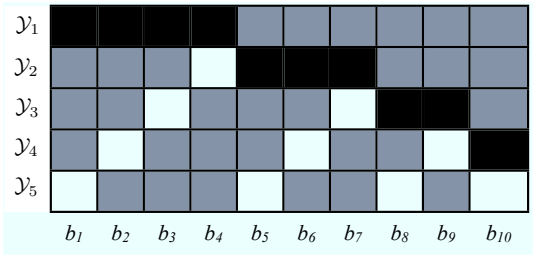


Fig. 3 ECOC *one-versus-one* coding matrix.

During the decoding process, applying the n binary classifiers, a code $\mathcal{X}$ is obtained for each data sample in the test set. This code is compared to the

base codewords ($\mathcal{Y}_c, c \in [1, \ldots, N]$) of each class defined in the matrix M. The data sample is then assigned to the class with the closest codeword. In order to find the closest codeword, the decoding strategies most frequently used are Hamming and Euclidean measures [21].

3 MMSSL: Multi-class Multi-scale Stacked Sequential Learning

In order to extend the Generalized Stacked Sequential Learning scheme to the multi-class case, it is necessary that base classifiers $h_1(\mathbf{x})$ and $h_2(x^{ext})$ can deal with data belonging to N classes instead of just two. This can be achieved using the ECOC framework explained before. Apart from the extension of the base classifiers, the neighborhood function $J(y', \rho, \theta)$ has also to be modified. Figure 4 shows the Multi-class Multi-scale Stacked Sequential Learning (MMSSL) scheme presented in this work. Given an input sample $\mathbf{x}$, the first classifier produces not only a prediction, but a measure of confidence $\hat{F}(\mathbf{x}, c)$ for belonging to each class defined in $c \in [1, \ldots, N]$. These confidence maps are the input of the neighborhood function $J(\hat{F}(\mathbf{x}, c), \rho, \Sigma)$. This function performs a multi-class decomposition over the confidence maps into s scales defined by Σ. Over this decomposition, a sampling ρ around each input example is returned, producing the z vector. The extended data set is built up using the original samples as well as the set of features in z. Finally, having the extended data set $\mathbf{x^{ext}}$ as input, the second classifier will predict to which class the input sample $\mathbf{x}$ belongs to. In the next two subsections we explain in detail this process. In the last subsection, we propose a compression approach for encoding the resulting confidence maps in order to reduce them to $\log_2 N$ without degrading the performance of the second classifier. Figure 4 shows the detail of function J once added the compression step between multi-scale decomposition and sampling.

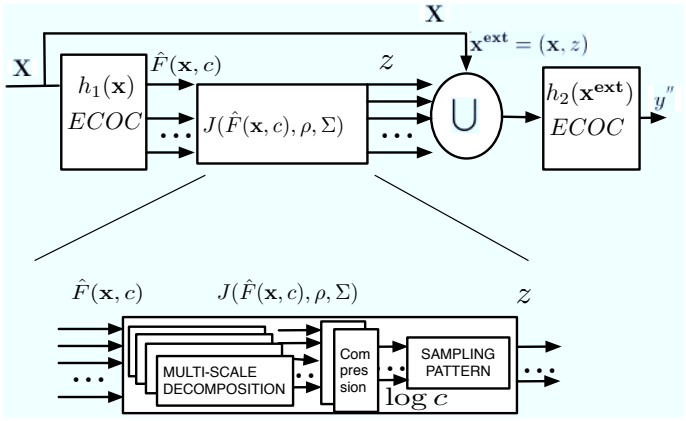


Fig. 4 Multi-class multi-scale stacked sequential learning and detail of function $J(\hat{F}(\mathbf{x}, c), \rho, \Sigma)$ with the compression step between multi-scale decomposition and sampling.

3.1 Extending the base classifiers

For training the first base classifier $h_1(\mathbf{x})$, where $\mathbf{x}$ is a sample of N possible classes, an ECOC coding strategy is defined. Based on this strategy, we obtain a codeword $\mathcal{Y}_c$, $c \in [1, 2, \ldots, N]$ of length n for each class. The symbols in the codeword $\{-1, 1, 0\}$ indicate whether this class belongs to one partition or another or if it should not be considered at all. The length of the codeword determines the number of dichotomizers (binary classifiers) that has to be trained. The matrix M defines for each dichotomizer which binary partition has to be performed on the training set. Given a test sample $\mathbf{x}$, each dichotomizer produces a prediction $[1, -1]$, forming a new codeword $\mathcal{X}$ of length n. The final predicted class is the closest codeword $\mathcal{Y}_c$ to codeword $\mathcal{X}$. A distance measure between codewords can be used for determining the closest class.

If the dichotomizers only produce binary predictions, all the predictions within $\mathcal{X}$ have the same importance. Instead, if the dichotomizers can produce a measure of confidence on its predictions, a more fine-grained distance between codewords can be obtained. Unfortunately, not all kind of classifiers can give a confidence for its predictions. However, classifiers that work with margins such as Adaboost or SVM can be used [8]. In these cases, it is necessary to convert the margins used by these classifiers to a measure of confidence with values between the codeword interval $[-1, 1]$. For example, in the Adaboost case, we apply a sigmoid function that normalizes Adaboost margins from the interval $[-\infty, \infty]$ to $[-1, 1]$, by means of the following equation,

$$f(b_i(\mathbf{x})) = \frac{1 - e^{-\beta m_{\mathbf{x}}}}{1 + e^{-\beta m_{\mathbf{x}}}},$$

where m_x is the margin of the predicted label given by one of the dichotomizers for the example $\mathbf{x}$, and a constant that governs the transition $\beta = \frac{-\ln(0.5\epsilon)}{0.25t}$, which depends on the number of iterations t that Adaboost performs, and an arbitrary small constant ϵ. We apply this equation for each dichotomizer forming a new codeword $\mathcal{X}$ of length n, where all the symbols $\in \mathbb{R}$.

Once we have a normalized codeword, we use a soft distance δ for decoding, i.e. we compare the codeword $\mathcal{X}$ with each codeword $Y_c, c \in [1, \ldots, N]$ defined in the matrix M. These distance measures can be seen as a prediction confidence measure for each class, once we normalize them to the range $[0, 1]$. Therefore, given a set of possible labels $c_i, i \in [1, \ldots, N]$, we define the membership confidences as follows:

$$\hat{F}(y = c_i|\mathbf{x}) = e^{-\alpha\delta(\mathcal{Y}_1, \mathcal{X})}, \forall i \in [1, .., N],$$

where δ is a soft distance such the Euclidean one, and α depends on δ. By applying this to the all data samples in $\mathbf{X}$ we define the confidence map for each class as expressed in Equation 1:

$$\hat{F}(\mathbf{x}, c) = \{\hat{F}(y = c_1|\mathbf{x}), \ldots, \hat{F}(y = c_N|\mathbf{x})\}, \forall \mathbf{x} \in \mathbf{X}. \quad (1)$$

3.2 Extending the neighborhood function J

We define the neighborhood function J in two stages: 1) a multi-scale decomposition over the confidence maps $\hat{F}(\mathbf{x}, c)$ and 2) a sampling performed over the multi-scale representation. This function is extended in order to deal with multiple classes. Now it is formulated as follows:

$$z = J(\hat{F}(\mathbf{x}, c), \rho, \Sigma).$$

Starting from the confidence maps, we apply a multi-scale decomposition upon them, resulting in as many decomposition sequences as labels. For the decomposition we use a multi-resolution Gaussian approach. Each level of the decomposition (scale) is generated by the convolution of the confidence map by a Gaussian mask with standard deviation σ. In this way, the bigger σ is, the longer interactions are considered. Therefore, at each level of decomposition all the points have information from the rest accordingly to the sigma parameter. Given a set of $\Sigma = \{\sigma_1, \ldots, \sigma_s\} \in \mathbb{R}^+$ and all the predicted confidence maps $\hat{F}(\mathbf{x}, c)$, each level of the decomposition $s_i, i \in [1, \ldots, s]$ is computed as follows:

$$\hat{F}^{s_i}(\mathbf{x}, c) = g^{\sigma_i}(\mathbf{x}) * \hat{F}(\mathbf{x}, c), \forall i \in [1, .., s],$$

where $g^{\sigma_i}(\mathbf{x})$ is defined as a multidimensional isotropic gaussian filter with zero mean:

$$g^{\sigma_i}(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}\sigma_i^{1/2}} e^{-\frac{1}{2}\mathbf{x}^T \sigma_i^{-1}\mathbf{x}}.$$

Once the multi-scale decomposition is performed, we define the support lattice z. This is, the sampling over the multi-scale representation which forms the extended data. Our choice is to use a scale-space sliding window over each label multi-scale decomposition. The selected window has a fixed radius with length defined by ρ in each dimension d and with origin in the current prediction example. Thus, the elements covered by the window is $w = (2\rho + 1)^d$ around the origin. Then, for each scale s_i considered in the previous decomposition the window is stretched in each direction using a displacement proportional to the scale we are analyzing. This displacement at each scale forces that each point considered around the current prediction has very small influence from previous neighbor points. In this way, the number of features of z appended to the input data set is equal to $(2\rho+1)^d \cdot s \cdot c$. According to this, we can see that the extended data set increases with the number of classes. This can produce a scalability problem, since the second classifier has to deal with large feature sets.

3.3 Extended data set grouping: a compression approach

The goal of grouping the extended data set is to compress its number of features without losing significant performance. Using our MMSSL approach, we can see that the size of the extended set depends on the number of classes,

the number of scales, and the number of samples around each example. We can choose the number of samples and scales, but the number of classes is problem dependent. Therefore, for reducing the number of confidence maps, we add a compression process between the multi-scale decomposition and the sampling process as shown in Figure 4. This compression is done following information theory by means of partitions.

Let $\mathcal{P}$ be a set $\{\{P_1^1, P_1^2\}, \ldots, \{P_\kappa^1, P_\kappa^2\}\}$ of partitions groups of classes and $c = \{c_1, \ldots, c_N\}$ the set of all the classes, so that for any j, $P_j^1 \subseteq c$, $P_j^2 \subseteq c \mid P_j^1 \cup P_j^2 = c$, and $P_j^1 \cap P_j^2 = \emptyset$. The confidence maps are grouped using the elements on $\mathcal{P}$. We have defined two different ways of combining the partitions: using *binary compression* or using *ternary compression*. Let the confidence map $\hat{F}^{s_k}$ of a certain scale s_k, $k \in [1, \ldots, s]$ be expressed as follows,

$$\hat{F}^{s_k}(\mathbf{x}, P_j) = \sum_i^N \gamma_{ij} \hat{F}^{s_k}(\mathbf{x}, c_i), \tag{2}$$

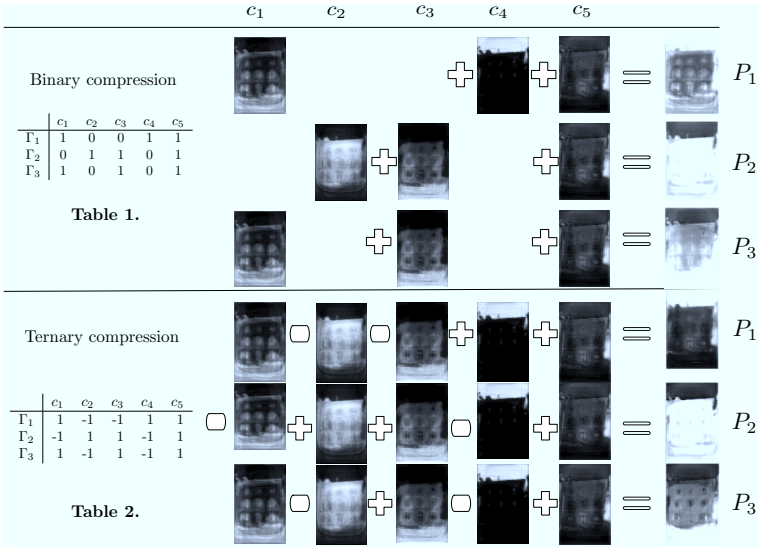
where

$$\gamma_{ij} = \begin{cases} a & \text{if } c_i \in P_j^1 \\ b & \text{if } c_i \in P_j^2 \end{cases}$$

for all the sets of partitions $P_j, j \in [1, \ldots, \kappa]$ in $\mathcal{P}$, being $a = 0$ and $b = 1$ in the case of binary compression and $a = -1$ and $b = 1$ in the case of ternary compression (we choose only $\{-1, 1\}$ values from the ternary set $\{-1, 0, 1\}$).

We use a partition strategy for $\mathcal{P}$ which produces a minimum set of partitions $\mathcal{P} = \{\{P_1^1, P_1^2\} \ldots, \{P_\kappa^1, P_\kappa^2\}\}$, where $\kappa = \lceil \log_2 |c| \rceil$, being $\lceil x \rceil = \min\{n \in \mathbb{Z} \mid n \geq x\}$. Our strategy builds the partitions assigning an unique binary code of length equals to number of partitions in $\mathcal{P}$ for each class. For example in Figure 5 a 5-class problem $c = \{c_1, c_2, c_3, c_4, c_5\}$ is illustrated. We can reduce the problem to a set of three partitions $\mathcal{P} = \{\{P_1^1 = \{c2, c3\}, P_1^2 = \{c1, c4, c5\}\}, \{\{P_2^1 = \{c1, c4\}, P_2^2 = \{c2, c3, c5\}\}, \{\{P_3^1 = \{c2, c4\}, P_3^2 = \{c1, c3, c5\}\}$. Therefore, in the binary case, the assigned codes for each partition are $\Gamma_1 = \{1, 0, 0, 1, 1\}$, $\Gamma_2 = \{0, 1, 1, 0, 1\}$, and $\Gamma_3 = \{1, 0, 1, 0, 1\}$, and in the ternary case, the assigned codes are $\Gamma_1 = \{1, -1, -1, 1, 1\}$, $\Gamma_2 = \{-1, 1, 1, -1, 1\}$, and $\Gamma_3 = \{1, -1, 1, -1, 1\}$. Thus, applying Equation 2, we obtain the likelihood maps for each partition, P_1, P_2, P_3. As it is shown in Figure 5, in the case of binary compression, the classes in P_i^1 for any partition i are not considered, while in the case of ternary compression, the classes in P_i^1 and P_i^2 for any partition i are combined.

Following this compression approach, now the support lattice z is defined over $\hat{F}^{\Sigma}(\mathbf{x}, \mathcal{P})$. This is, applying Equation 2 over all the scales defined by $\Sigma = \{\sigma_1, \ldots, \sigma_s\}$. Therefore, the number of features in z is reduced from $(2\rho + 1)^d \cdot s \cdot c$ to $(2\rho + 1)^d \cdot s \cdot \lceil \log_2 c \rceil$.



	c_1	c_2	c_3	c_4	c_5
Γ_1	1	0	0	1	1
Γ_2	0	1	1	0	1
Γ_3	1	0	1	0	1

Table 1.

	c_1	c_2	c_3	c_4	c_5
Γ_1	1	-1	-1	1	1
Γ_2	-1	1	1	-1	1
Γ_3	1	-1	1	-1	1

Table 2.

Fig. 5 5-class likelihood maps compressed to three, using partitions. Binary approach is represented by Table 1. The symbols used are 0 and 1. Ternary approach is represented by Table 2. The symbols used are -1 and 1. Applying Equation. 2 we obtain the aggregated likelihood maps $P_1, P_2, P_3 \in \mathcal{P}$. In the case of binary compression, any class marked with zero in a codeword Γ is not considered, while in the case of ternary compression, all classes are aggregated according to each of the codewords Γ.

4 Experiments and Results

Before presenting the results, data, methods and validation protocol for each experiment are discussed. The results are presented in two aspects, a) statistical results, where different measures are computed and significance tests are performed on different datasets, and b) qualitative results, where concrete results are particularly analyzed for a more intuitive understanding of the behavior of each method.

4.1 Experimental Settings

– **Data**: we test our multi-class methodology performing 9 different experiments out from four databases:
  1. Sensor Motion data database: The sensor motion database [20] is a data set of accelerometer sensor runs from 15 different people performing certain activities. Each accelerometer sample is labeled as one of 5 different activities, namely walking, climbing stairs, standing idle, interacting and working. The spatial relationship in label space is 1D. There are two different scenarios. Sequential scenario is where all the people is doing the activities in the same order (*motion sequential scenario*). Random scenario is where all the people is performing the activities in

random order (*motion random scenario*)). We also performed a third experiment for benchmark purposes in which there are only activities from one person (*motion one person*).

2. FAQ database [6,4]: The FAQ database is a set of frequented asked questions pages from Usenet. There are 48 annotated pages from several topics. Each line in a page is labeled as (0) header, (1) question, (2) answer, or (3) tailing. There are 24 boolean features characterizing each line. The spatial relationship in label space is 1D.
3. IVUS image database [17]: It contains images from Intravascular Ultrasound (IVUS). They are a set of IVUS frames manually labelled. 8 classes are considered: (1) blood, (2) plaque, (3) media, (4) media adventitia, (5) guide-wire, (6) shadowing, (7) external tissue, and (8) calcium. The spatial relationship in label space is 2D. There are 29 textural features in total extracted from IVUS data.
4. e-trims database [12]: The e-trims database is comprised of two image datasets, *e-trims 4-class* with four annotated object classes and *e-trims 8-class* with eight annotated object classes. There are 60 annotated images in each of the dataset. The object classes considered in 4-class dataset are: (1) building, (2) pavement/road, (3) sky, and (4) vegetation. In 8-class dataset the object classes considered are: (1) building, (2) car, (3) door, (4) pavement, (5) road, (6) sky, (7) vegetation, and (8) window. Additionally, for each database we have a background class (0) for any other object. All images are resized proportionally to 150 pixels height. Train images are stratified sampled, taking 3000 pixels. We have performed experiments with two different set of features: **RGB** representation of each pixel, and RGB plus **HOG** (Histogram of oriented gradient [18]) with 9 bins, ending up with 12 features for sample. The spatial relationship in label space is 2D.

– **Methods**: We test all the databases with four different configurations of our MMSSL methodology. Also, we test with Real Adaboost [22] and CRF Multi-label optimization through Graph Cut α-expansion [13] as baseline experiments. The settings for all the MMSSL configurations are the same, the only difference is the way the extended data set is generated. We have used as base classifier a Real Adaboost ensemble of 100 decision stumps. The coding strategy for the ECOC framework in each classifier is *one-versus-one* and the decoding measure is Euclidean distance. The neighborhood function performs a Gaussian multi-resolution decomposition in 4 scales, using $\Sigma = \{1, 2, 4, 8\}$, except in IVUS database where we used 6 scales $\Sigma = \{1, 2, 4, 8, 16, 32\}$ due to the images dimensions. In 1D databases, we used $w = 7$ elements in both directions of the neighborhood, while in 2D databases we used just the surrounding points, i.e. $w = 1$. Summarizing, the different experiments we have performed are:

1. MMSSL using labels. It uses the MMSSL framework using only the predicted labels from the first classifier as input for neighborhood function.

2. MMSSL using confidences. It uses the MMSSL framework using the confidence maps for all the classes as input for neighborhood function.
3. MMSSL using compression approach with binary matrix: It uses the MMSSL framework using a compression over the confidence map. The compression matrix uses binary values $\{0, 1\}$.
4. MMSSL using compression approach with ternary matrix. It uses the MMSSL framework using a compression over the confidence map. The compression matrix uses ternary values $\{-1, 1\}$.
5. Adaboost. Uses only one Adaboost classifier, without taking into account the neighborhood relationship. Used as baseline experiment.
6. Multi-label optimization. It uses multi-label optimization via α-expansion. We have applied the α-expansion optimization, using the confidence maps for each class obtained from the first classifier. For the neighborhood term, we use the intensity between the point and its neighbors for each direction defined in the database.

– **Validation**: For sensor motion and FAQ databases we use one-leave-out for final prediction, whereas for IVUS and E-trims databases we use 5-fold cross-validation. For each fold, the base classifier $h_1(x)$ uses ten-fold cross-validation for predicting the labels of the training set, which produces the confidence maps used later for the second classifier $h_2(x)$. We measure the results in terms of the accuracy, and the mean of overlapping, recall, and precision from a $N \times N$ confusion matrix , computed as follow: $accuracy = \frac{\sum_i^N TP_i}{\sum_i^N (TP+FP+FN)_i}$, $overlapping_i = \frac{TP_i}{(TP+FN+FP)_i}$, $recall_i = \frac{TP_i}{(TP+FN)_i}$, and $precision_i = \frac{TP_i}{(FP+TP)_i}$, where TP_i means the predictions correctly classified in the class i, FP_i means the predictions misclassified as class i and FN_i means the actual class i predictions misclassified as any other class. For comparing the results obtained from the different experiments we have used statistic tests: the Friedman test for checking the non-randomness of the results and the Nemenyi test for checking if one of the configurations can be statistically singled out [19].

4.2 Numerical Results

Tables 2 to 8 show accuracy, overlapping, recall, and precision averaged for each experiment. Best results are marked in bold. The tables show similar tendency of the different classifiers results for different databases. Non sequential methods such Adaboost give the poorest accuracies. Multi-label optimization using Graph cut achieves better results, specially in 2D databases. Finally, all methods based in MMSSL give the best results. Usually, using just predictions it leads to worse results than using confidence maps. It is also remarkable that by using compression techniques (binary and ternary coding) the global accuracy is not significantly degraded. In order to compare the performances provided for each of theses strategies, Table 9 shown in the mean rank of each strategy considering the accuracy terms of the 9 different experiments. The

rankings are obtained estimating each particular ranking r_i^j for each data sequence i and each system configuration j, and computing the mean ranking R for each configuration as $R_j = \frac{1}{E}\sum_i r_i^j$, where E is the total number of experiments.

	Accuracy	Overlapping	Recall	Precision
ADABoost	0.5771	0.3142	0.4419	0.4504
GraphCut	0.5766	0.3129	0.4404	0.4489
Labels	0.6403	0.4766	0.6079	0.6516
Standard	0.7069	0.5905	0.7048	**0.8098**
SublinealBinary	**0.7361**	**0.6021**	**0.7427**	0.7914
SublinealTernary	0.7026	0.5648	0.6843	0.7638

Table 2 Result figures for database motion sequential scenario.

	Accuracy	Overlapping	Recall	Precision
ADABoost	0.5771	0.3142	0.4419	0.4504
GraphCut	0.5766	0.3129	0.4404	0.4489
Labels	0.5951	0.3833	0.5292	0.5375
Standard	0.7109	0.4365	0.552	0.5867
SublinealBinay	**0.7305**	**0.4677**	**0.5912**	**0.6266**
SublinealTernary	0.6937	0.4392	0.5748	0.6159

Table 3 Result figures for database motion random scenario.

	Accuracy	Overlapping	Recall	Precision
ADABoost	0.7607	0.553	0.6805	0.715
GraphCut	0.7888	0.5865	0.7043	0.7654
Labels	0.879	0.7489	0.8372	0.8736
Standard	**0.902**	**0.793**	**0.8792**	**0.8824**
SublinealBinary	0.8571	0.7133	0.8125	0.8458
SublinealTernary	0.8796	0.7477	0.8395	0.8652

Table 4 Result figures for database motion one person.

In order to reject the null hypothesis that the measured ranks differ from the mean rank, and that the ranks are affected by randomness in the results, we use the Friedman test. The Friedman statistic value is computed as follows:

$$\chi_F^2 = \frac{12E}{k(k+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right].$$

In our case, with $k = 6$ system configurations to compare, $\chi_F^2 = 35.79$. Since this value is undesirable conservative, Iman and Davenport [19] proposed a

	Accuracy	Overlapping	Recall	Precision
ADABoost	0.8552	0.2392	0.2781	0.3906
GraphCut	0.858	0.2355	0.2718	0.4427
Labels	0.8906	0.4346	0.4961	0.6675
Standard	0.8866	0.5125	0.5627	**0.8122**
SublinealBinary	0.8786	0.4809	0.5275	0.7649
SublinealTernary	**0.8998**	**0.5628**	**0.6277**	0.8067

Table 5 Result figures for database FAQ.

	Accuracy	Overlapping	Recall	Precision
ADABoost	0.6605	0.3127	0.422	0.4978
GraphCuts	0.6748	0.3102	0.4175	0.4654
Labels	0.6789	0.3359	0.4435	0.5098
Standard	**0.7199**	**0.3764**	**0.4842**	**0.5555**
SublinealBinary	0.684	0.3379	0.4457	0.5205
SublinealTernary	0.7006	0.3544	0.4618	0.5345

Table 6 Result figures for database IVUS, using 6 scales.

		Accuracy	Overlapping	Recall	Precision
RGB	ADABoost	0.7274	0.3612	0.4351	0.5334
	GraphCuts	0.7283	0.3435	0.4113	0.4688
	Labels	0.7612	0.4232	0.5004	0.6716
	Standard	0.8074	**0.5189**	**0.6137**	0.6922
	SublinealBinary	0.7987	0.4957	0.5806	0.6924
	SublinealTernary	**0.8078**	0.5172	0.6085	**0.7028**
HOG	ADABoost	0.8067	0.5115	0.608	0.6648
	GraphCuts	0.8317	0.53	0.6108	0.6962
	Labels	0.8305	0.5447	0.6385	0.6878
	Standard	**0.8686**	**0.599**	**0.6912**	**0.7373**
	SublinealBinary	0.8514	0.5767	0.678	0.7151
	SublinealTernary	0.8599	0.5852	0.6752	0.7333

Table 7 Result figures for database ETRIMS 4 classes RGB and HOG.

corrected statistic:

$$F_F = \frac{(N-1)\chi_F^2}{E(k-1)-\chi_F^2}.$$

Applying this correction we obtain $F_F = 31.11$. With 6 methods and 9 experiments, F_F is distributed according to the F distribution with 5 and 40 degrees of freedom. The critical value of $F(5, 40)$ for 0.05 is 2.44. As the value of $F_F = 31.11$ is higher than 2.44 we can reject the null hypothesis. Once we have checked for the non-randomness of the results, we can perform an a post-hoc test to check if one of the configurations can be statistically singled out. For this purpose we use the Nemenyi test. The Nemenyi statistic is obtained as follows:

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6E}}.$$

		Accuracy	Overlapping	Recall	Precision
RGB	ADABoost	0.606	0.1991	0.2591	0.3003
	GraphCuts	0.6039	0.1859	0.2405	0.2719
	Labels	0.6549	0.2526	0.3193	0.4297
	Standard	**0.703**	**0.3133**	**0.3891**	**0.4752**
	SublinealBinary	0.6616	0.267	0.3389	0.4439
	SublinealTernary	0.6742	0.2768	0.346	0.4361
HOG	ADABoost	0.6723	0.2868	0.3618	0.4623
	GraphCuts	0.6812	0.2618	0.3255	0.3678
	Labels	0.6885	0.3031	0.3797	0.4706
	Standard	**0.7312**	**0.3479**	**0.4338**	**0.5103**
	SublinealBinary	0.6895	0.3038	0.3837	0.4765
	SublinealTernary	0.7164	0.3348	0.4222	0.4986

Table 8 Result figures for database ETRIMS 8 classes RGB and HOG.

Method	ADAboost	GraphCut	Labels	Standard	Sub.Binary	Sub.Ternary
Rank	5.7	5.1	3.9	1.7	2.8	1.9

Table 9 Mean rank of each strategy considering the accuracy terms of the different experiments.

In our case with $k = 6$ system configurations to compare and $E = 9$ experiments (data configurations) the critical value for a 90% of confidence is $CD = 1.27$. In Figure 6 we can see a graphical representation of this post-hoc test. As the ranking of the MMSSL Standard method intersects with both sublineal approaches ranks for that value of the CD, we can state that MMSSL using confidences outperforms the rest of the methods in the presented experiments. Moreover, it reveals that among compressed and non compressed MMSSL strategies statistically significant differences do not exist. This fact reinforce our idea of grouping features without losing performance is feasible. The main advantage for using the compression approach is that by reducing the number of features in the extended dataset, the time of the learning phase for the second classifier is reduced. Therefore, the MMSSL framework scales sublinearly in feature space with the number of classes without a loss in generalization.

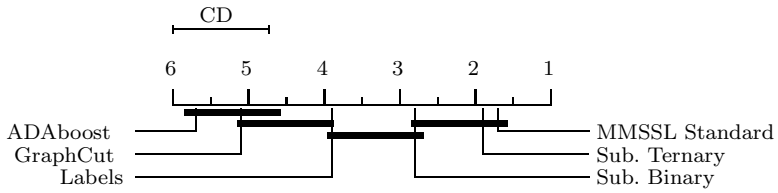


Fig. 6 Comparison of all methods against each other with the Nemenyi test. Groups of classifiers that are not significantly different are connected.

4.3 Qualitative Results

In this section we highlight general observations comparing ADAboost, multi-label optimization gaph cut and our MMSSL approach. Figure 7 shows results in 1D motion database. The rest of figures shows results in 2D databases, Figure 8 and Figure 9 show results in e-trims database 4 and 8 classes, respectively, and Figure 10 shows results in IVUS database,.

The images resulting from *ADAboost* classification show how this method does not capture sequential relationship among labels. For example, in 1D database results shown in Figure 7, we can see how contiguous points inside a long class interval are classified as belonging to another class. In 2D database results show spurious classified pixels appearing inside big objects. For example, in the first row of Figure 8 in the upper side of the building few pixels appear labelled as tree. In the second row of the same image clouds in the middle of sky are marked as building and in the third row of the same figure a wire crossing the sky is misclassified. In the last two rows shadows on the top of the buildings are classified as road. In Figure 9 as many other classes exist, the effect of spurious artifacts on Adaboost results are more notorious, for example in the last row, dark clouds are misclassified as belonging to the building. In Figure 10 we can see that Adaboost fails, producing results far from the real classification, like in the first and second row. All artifacts observed appear due to specific pixel values which lead the classifier to a misclassification.

On the contrary, the multi-label optimization technique by means of *Graph Cut* captures sequential relationships between labels, erasing such interclass artifacts. In 1D database results shown in Figure 7, we can see how the number of bad classified contiguous points decreases with respect to ADAboost, but it still fails in classify correctly short intervals of contiguous points of certain classes. In 2D databases, the drawbacks of this method are a) the tendency to crop the contours of the objects producing sharp shapes resembling blobs, as is reflected in the first, third and fourth rows of Figure 8 where trees lose all its shape, even the building in the third row is rounded, and b) the elimination of entire overlapped objects, as is shown in the three first rows of Figure 9, where trees, windows and doors are completely removed, only prevailing the building class. Even though, long objects are still misclassified, as the shadows in the top part of the buildings in the last two rows in Figure 8, or worst, the dark clouds in the last row of Figure 9 are completely joined with the building forming a huge building. In Figure 10 we can see a fairly improvement with respect to Adaboost, but it still fails in the classification of the three first rows. This method fails mainly because it is not considering the relationship among objects at different scales.

The last method considered is our approach, *MMSSL* using confidences without compression.The results of this method are qualitatively better than the rest. The results are a trade-off between spacial coherence and shape preservation. This is because the relationship among classes is considered at different scales. In 1D database results shown in Figure 7, we can see how the MMSSL is the only method that achieves good performance as well in

long sequences as in short sequences of points of the same class and does not matter whether the activities are carried on in the same order as trained or not. In 2D databases, we can see in Figure 8 how MMSSL is able to keep the shape of buildings and trees in all the images and how it removes interclass artifacts that previous method were not able to, for example the shadows on the top of the building in the last two rows. In Figure 9 we can see in all the images that windows, trees and doors are fairly kept, even the dark clouds in the last row are practically removed, appearing only spurious pixels in the border of the image. Moreover, in Figure 10, we can see how MMSSL is able to close big areas of the same class like in the three firsts rows, where the rest of methods fail. Also is remarkable in the fourth and fifth row how narrower classes between wider classes are preserved. The points where our method fails the most are the junctions between not clearly distinct classes, for example in the second row in Figure 9 where cars are classified altogether as one, mixing with the grass and the road.

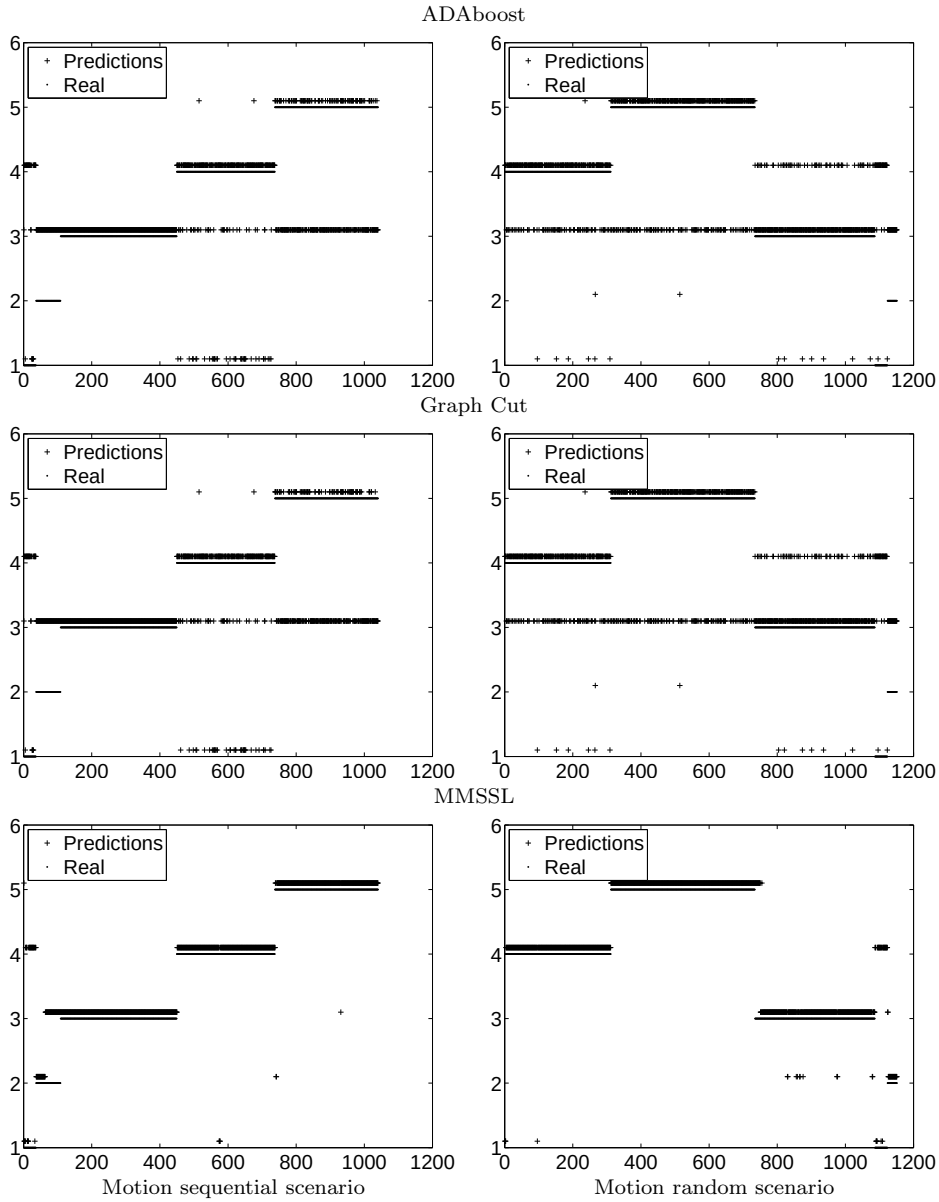


Fig. 7 Figures of final classification in motion sequential scenario and motion random scenario for ADAboost, multi-label optimization Graph cut, and our proposal MMSSL. Y-axe shows the labels for each class and X-axe is the time interval. Predictions values are marked with + and real values are marked just below with dots.

Fig. 8 Figures of final classification in ETRIMS 4 Classes HOG database. (a) Shows the original image, (b) the groundtruth image, and (c),(d), and (e) show ADAboost, GraphCut, and MMSSL without compression, respectively.

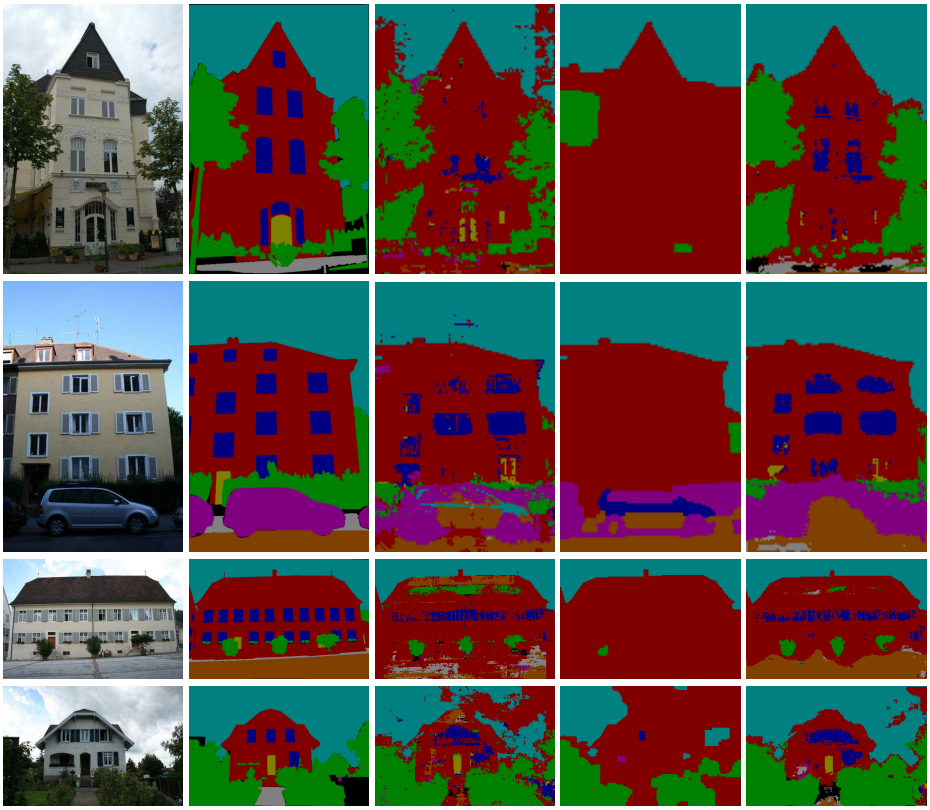


Fig. 9 Figures of final classification in ETRIMS 8 Classes HOG database. (a) Shows the original image, (b) the groundtruth image, and (c),(d), and (e) show ADAboost, GraphCut, and MMSSL without compression, respectively.

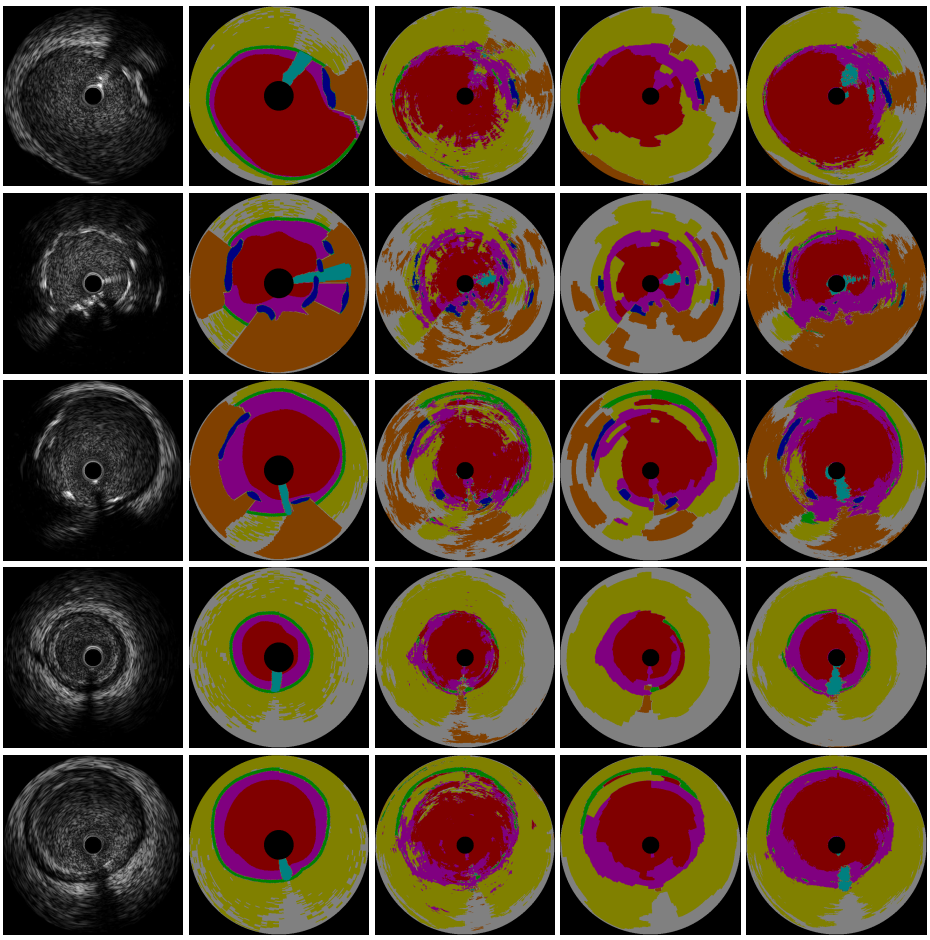


Fig. 10 Figures of final classification in IVUS using 6 scales. (a) Shows the original image, (b) the groundtruth image, and (c),(d), and (e) show ADAboost, GraphCut, and MMSSL without compression, respectively.

4.4 Comparing among proposed techniques

Finally, Figures 11 and 12 show the difference among different four MMSSL configurations: MMSSL using only label predictions, MMSSL using confidences without compression, MMSSL using binary compression, and MMSSL using ternary compression. MMSSL using labels is prone to fail in some long areas of contiguous pixels. For example in the first three rows of Figure 11 some areas between road and vegetation are misclassified as building. In the first two rows of Figure 12 also appear strange misclassified areas on the top of the building. Although, in this situations, the rest of methods that use confidences do not fail. This is because using confidences instead of the most probable label in the extended data set, the second classifier can learn relationships between labels considering not only the final prediction class, but the probabilities of being of each class. In this way, it is easier to break ties of equiprobable predictions in favor of the most coherent class. The second row of Figure 12 shows the learning capacity of the likelihood maps. In column (d), MMSSL using confidences without compression, we can see a boundary marked as unknown object (class 0, black label) in front of the building. Inside this region, it is marked as car label. In column (c), MMSSL only using labels predictions, and in the groundtruth image, column (b), these elements are omitted, but in the original image, column (a), it is appreciable a woman riding a bicycle in that area. Therefore MMSSL using likelihood maps is capable of detecting them as an element different to road or building, and assigning the inner region to car label, given its visual appearance and position.

Differences between compressed and non compressed methods are not so straightforward to see, but while in Figure 11 there are few differences in Figure 12 we can see how non compressed methods lead to smoother results than compressed methods. Compressed methods tend to fail in closing some classes, appearing spurious pixels inside them. For example, in second and third row in Figure 11 using binary compression some few pixels labeled as vegetation appear in the middle of building, while this does not happen in the non compressed approach. In Figure 12 we can see in the first and second rows of binary approach pixels in the sky labeled as building and in the first row of ternary approach pixels in the top of the building labeled as car. All in all, these misclassified samples with respect to non compressed MMSSL are very few, taking into consideration the number of features that are compressed. In fact, in some images like the fourth row in Figure 11 and the last row in Figure 12 the results of the three MMSL methods using likelihoods are almost the same. Even in situations like the building roof in the last row of 11 ternary compression can reduce areas of misclassified pixels that standard method could not resolve.

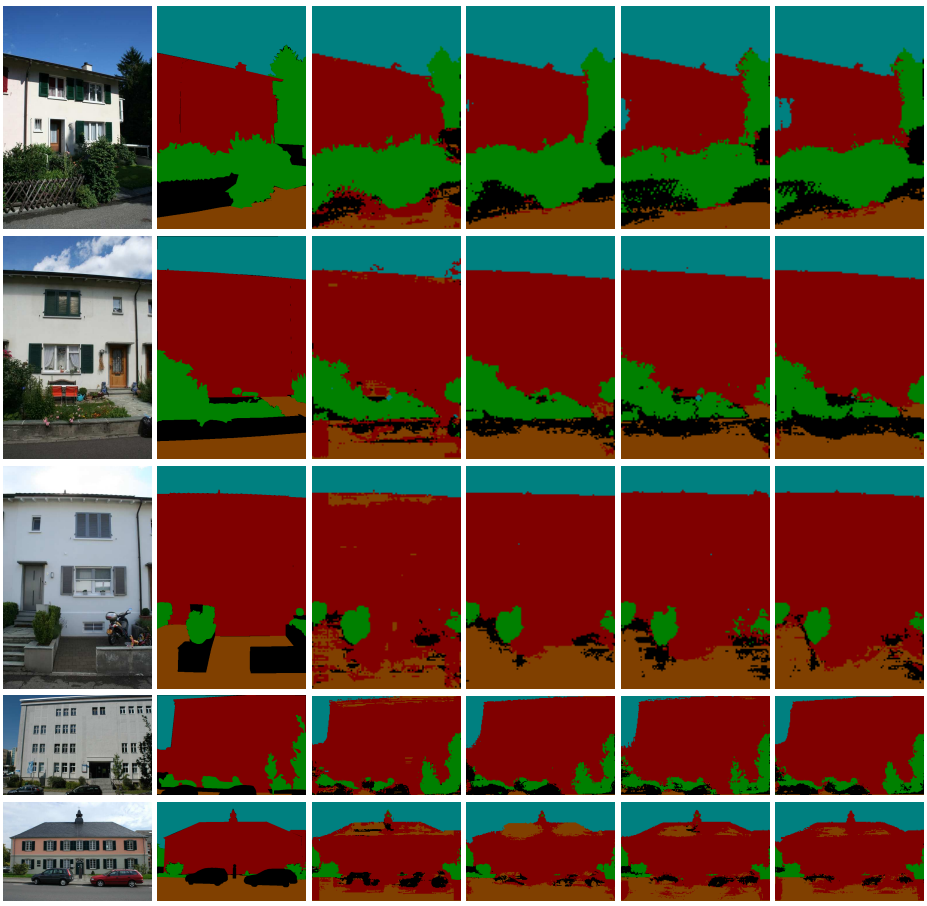


Fig. 11 Comparative between multi-class multi-scale stacked sequential learning approaches in ETRIMS 4 Classes HOG database. (a) Shows the original image, (b) the groundtruth image, and (c), (d), (e), and (f) shows the different MMSSL schemes: (c) MMSSL using label predictions, (d) MMSSL using confidences, (e) MMSSL using binary compression, and (f) MMSSL using ternary compression

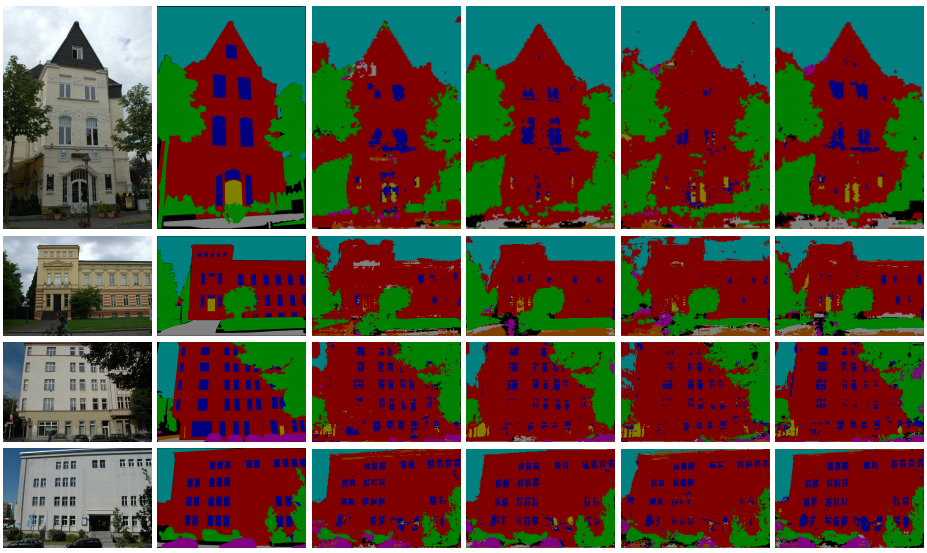


Fig. 12 Comparative between multi-class multi-scale stacked sequential learning approaches in ETRIMS 8 Classes HOG database. (a) Shows the original image, (b) the groundtruth image, and (c), (d), (e), and (f) shows the different MMSSL schemes: (c) MMSSL using label predictions, (d) MMSSL using confidences, (e) MMSSL using binary compression, and (f) MMSSL using ternary compression.

5 Conclusion

In this paper we adapt the multi-scale sequential learning (MSSL) to the multi-class case (MMSSL). First, we introduce the ECOC framework in the MSSL classifiers. Next, we show how to compute the confidence maps using the normalized margins obtained from the ECOC base classifiers. Finally we define a compression approach for reducing the number of features in the extended data set. The results show that, on the one hand, MMSSL achieves accurate classification performance in multi-class classification problems taking benefit of sequential learning. On the other hand, the compression process is feasible, since in terms of accuracy the loss of information is negligible. As future work, we will study how to extend the compression process not only to the set of confidence labels, but to the whole extended set. By reducing the amount of features used in neighbor sampling, we can improve the speed of the method and deal with databases having larger number of classes.

References

1. E.Allwein, R. Schapire, and Y.Singer, *Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers*, J. Machine Learning Research, vol. 1, pp. 113–141, 2002.
2. T.G Dietterich and G.Bakiri, *Solving Multiclass Learning Problems via Error-Correcting Output Codes*, J.Artificial Intelligence Research, vol. 2, pp. 263–286, 1995.
3. T. G. Dietterich, *Machine Learning for Sequential Data: A Review*, Proc. on Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition, Lecture Notes in Computer Science, vol. 2396, pp. 15–30, 2002.
4. T. G. Dietterich, A. Ashenfelter, and Y. Bulatov, *Training conditional random fields via gradient tree boosting*, In Proc. of the 21th ICML, pp. 217–224, 2004.
5. N.J.Nilsson, *Learning Machines*. McGraw-Hill, 1965.
6. W. W. Cohen and V. R. de Carvalho, *Stacked sequential learning*, Proc. of IJCAI 2005, pp. 671–676, 2005.
7. A. McCallum, D. Freitag, and F. Pereira, *Maximum entropy markov models for information extraction and segmentation*, Proc. of ICML 2000, pp. 591–598, 2000.
8. J. Friedman, T. Hastie, and R. Tibshirani, *Additive logistic regression: a statistical view of boosting*, Annals of Statistics, vol. 28, num. 2, 1998.
9. D. H. Wolpert, *Stacked generalization*, Neural Networks, vol. 5, n. 2, pp. 241–259, 1992.
10. J. D. Lafferty, A. McCallum, and F. Pereira, *Conditional random fields: Probabilistic models for segmenting and labeling sequence data*, Proc. of ICML 2001, pp. 282–289, 2001.
11. P.. Burt and E. Adelson, *The laplacian pyramid as a compact image code*, IEEE Transactions on Communications, vol. 31,n. 4, pp. 532–540, 1983.
12. F. Korč. and W. Förstner, *eTRIMS Image Database for Interpreting Images of Man-Made Scenes*, TR-IGG-P-2009-01, University of Bonn, 2009.
13. Y. Boykov and G. Funka-Lea, *Graph Cuts and Efficient N-D Image Segmentation*, In I. Journal of Computer Vision, vol. 70, no. 2, pp. 109-131, 2006.
14. S. Escalera, D. Tax, O. Pujol, P. Radeva, and R. Duin,*Subclass Problem-dependent Design of Error-Correcting Output Codes*, IEEE T. in Pattern Analysis and Machine Intelligence, vol. 30, issue 6, pp. 1041–1054, 2008.
15. V. Mottl, S. Dvoenko, and A. Kopylov, *Pattern recognition in interrelated data: The problem, fundamental assumptions, recognition algorithms*, Proc. of the 17th ICPR, Cambridge UK, Vol. 1, pp. 188–191, 2004.
16. C. Gatta, E. Puertas, and O. Pujol, *Multi-scale stacked sequential learning*, Pattern Recognition, vol. 44, no. 10-11, pp. 2414–2426, 2011.

17. Ciompi F, et al. *A holistic approach for the detection of media-adventitia border in IVUS*, Med Image Comput Comput Assist Interv. MICCAI'11, vol. 14, no. 3, pp. 411–419. 2011.
18. N. Dalal, and B.Triggs, *Histograms of Oriented Gradients for Human Detection*, Proc of. 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), vol. 1, pp. 886–893. 2011.
19. J. Demšar, *Statistical Comparisons of Classifiers over Multiple Data Sets*, Journal of Machine Learning Research, vol. 7, pp. 1–30, 2006.
20. P. Casale, O. Pujol, and P. Radeva, *Personalization and user verification in wearable systems using biometric walking patterns* , in Personal and Ubiquitous Computing, pp. 1–18, 2011.
21. S. Escalera, O. Pujol, and P. Radeva, *On the Decoding Process in Ternary Error-Correcting Output Codes*, in Transactions in Pattern Analysis and Machine Intelligence, vol. 32, issue 1, pp. 120–134, 2010.
22. Yoav Freund and Robert E. Schapire, *A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting*, in Journal of Computer and System Sciences, vol 55, pp. 119–139, 1995.
23. Y. Boykov and V. Kolmogorov, *Computing geodesics and minimal surfaces via graph cuts*, in Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on, pp.26-33 vol.1, 13-16 Oct. 2003
24. L. Bottou and Y. LeCun, *Graph Transformer Networks for Image Recognition*, Bulletin of the International Statistical Institute (ISI)