Multi-class Multi-scale Stacked Sequential Learning

Eloi Puertas[1,2], Sergio Escalera[1,2], and Oriol Pujol[1,2]

[1] Dept. Matemàtica Aplicada i Anàlisi, Universitat de Barcelona, Gran Via 585, 08007, Barcelona, Spain
[2] Computer Vision Center, Campus UAB, Edifici O, 08193, Bellaterra, Spain
eloi@maia.ub.es, sergio@maia.ub.es, oriol@maia.ub.es

Abstract. One assumption in supervised learning is that data is independent and identically distributed. However, this assumption does not hold true in many real cases. Sequential learning is that discipline of machine learning that deals with dependent data.

In this paper, we revise the Multi-Scale Sequential Learning approach (MSSL) for applying it in the multi-class case (MMSSL). We have introduced the ECOC framework in the MSSL base classifiers and a formulation for calculating confidence maps from the margins of the base classifiers. Another important contribution of this papers is the MMSSL compression approach for reducing the number of features in the extended data set. The proposed methods are tested on 5-class and 9-class image databases.

1 Introduction

Sequential learning [3] assumes that samples are not independently drawn from a joint distribution of the data samples $\mathbf{X}$ and their labels Y. In sequential learning, training data actually consists of sequences of pairs $(\mathbf{x}, y)$, so that neighboring examples on a support lattice display some correlation. Usually sequential learning applications consider one-dimensional relationship support, but this kind of relationships appear very frequently in other domains, such as images, or video. Consider the case of object recognition in image understanding. It is clear that 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 its borders).

In literature, sequential learning has been addressed from different perspectives: from the point of view of graphical models, using Hidden Markov Models or Conditional Random Fields (CRF) [10,7,4,15] for inferring the joint or conditional probability of the sequence. 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 SSL, a first base classifier is used to produce predictions. A sliding 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 our previous work [11], we identified that the main

C. Sansone, J. Kittler, and F. Roli (Eds.): MCS 2011, LNCS 6713, pp. 197–206, 2011.

step of the relationship modeling proposed in [6] is precisely how this extended dataset is created. In consequence, we formalized a general framework for the SSL called Multi-scale Stacked Sequential Learning (MSSL), where a multi-scale decomposition is used in the relationship modeling step.

Previous approaches addres bi-class problems. Few of them have been extended to the multi-class case. However in many applications, like image segmentation, problems are inherently multi-class. In this work, our contribution is an efficient extension of MSSL to the multi-class case. We revise the general stacked sequential learning scheme for applying to multi-class as well as bi-class problems. We introduce the ECOC framework [2] in the base classifiers to extend them to the multi-class case, defining the Multi-class Multi-scale Stacked Sequential Learning (MMSSL). An important issue considered in this work is how the number of features in the extended data set increases with the number of classes. We propose a feature compression approach for mitigating this problem.

The paper is organized as follows: first, we review the original MSSL for the bi-class case. In the next section, we formulate the MSSL for the multi-class case (MMSSL), introducing the ECOC framework. Each step in MSSL is revised for the multi-class case and a compression approach for the extended dataset is explained. Then experiments and results of our methodology are shown in Section 4. Finally, Section 5 concludes the paper.

2 Related Work: Multi-scale Stacked Sequential Learning

SSL [6] is a meta-learning framework [9] consisting in two steps. In the first step, 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. At the second step, another classifier is trained with this new feature set. In [11] SSL is generalized by emphasizing the key role of neighborhood relationship modeling. The framework presented includes a new block in the pipeline of the basic SSL. Figure 1 shows the Generalized Stacked Sequential Learning process. A classifier $h_1(x)$ is trained with the input data set $(\mathbf{x}, y)$ and the set of predicted labels y' is obtained. The next block defines the policy for creating the neighborhood model of the predicted labels. It is represented by $z = J(y', \rho, \theta) : \mathcal{R} \rightarrow \mathcal{R}^w$, a function that captures the data interaction with a model parameterized by θ in a neighborhood ρ. The result of this function is a

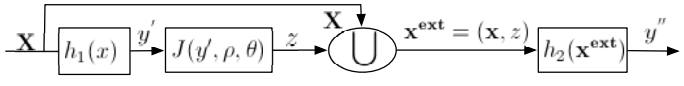

Fig. 1. Generalized stacked sequential learning

w-dimensional value, where w is the number of elements in the support lattice of the neighborhood ρ. In the case of defining the neighborhood by means of a window, w is the number of elements in the window. 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 goal of producing the final prediction y''.

The definition of $J(y', \rho, \theta)$ proposed in [11] consists of two steps: first the multi-scale decomposition that answers how to model the relationship between neighboring locations, and second, the sampling that answers how to define the support lattice to produce the final set z.

3 Multi-class Multi-scale Stacked Sequential Learning

To extend the generalized stacked sequential learning scheme to the multi-class case, it is necessary that base classifiers $h_1(x)$ and $h_2(x^{ext})$ can deal with data belonging to several classes instead of just two. This can be achieved by using as base classifier a built-in multi-class classifier. However, it is also possible to use binary base classifiers by applying an ensemble of classifiers. One of the most successful ensemble strategies is the Error-Correct Output Codes framework [2,14]. In essence, ECOC is a methodology used for reducing a multi-class problem into a sequence of two-class problems. It is composed by two phases: a coding phase, where a codeword is assigned to each class, and a decoding phase, where, given a test sample, it looks for the most similar class codeword. The codeword is a sequence of bits, where each bit identifies the membership of the class for a given binary classifier. 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. Another standard coding strategy is the *one-versus-one* [1] which considers all possible pairs of classes, with a codeword length of $\frac{N(N-1)}{2}$.

The decoding phase of the ECOC framework is based on error-correcting principles under the assumption that the learning task can be modeled as a communication problem. Decoding strategies based on distances measurements between the output code and the target codeword are the most frequently applied. Among these, Hamming measure and Euclidean measure are the most used [14].

Apart from the extension of the base classifiers, the neighborhood function J has to be also modified. Figure 2 shows the multi-class multi-scale stacked sequential learning scheme presented in this work. Now, from an input sample, the first classifier produces not only a prediction, but a measure of confidence for belonging to each class. These confidences maps are the input of the neighborhood function. This function performs a multi-class decomposition over the confidence maps. Over this decomposition, a sampling z around each input example is returned. The extended data set is built up using the original samples as well as the set of features selected in z. Additionally, in order to reduce the number of features in the extended data set, we propose a compression approach

for encoding the resulting multi-class decomposition of the confidence maps. Finally, having the extended data set $\mathbf{x^{ext}}$ as input and using again the ECOC framework, the second classifier will predict to which class belongs the input sample $\mathbf{x}$. In the next subsections we explain in detail how the generalized stacked sequential learning can be extended to the multi-class case.

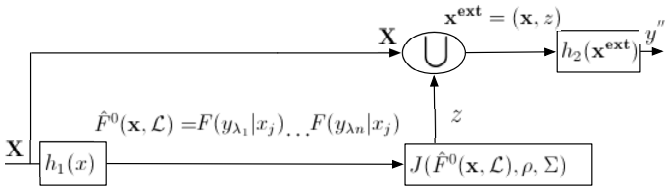

Fig. 2. Multi-class multi-scale stacked sequential learning

3.1 Extending the Base Classifiers

For the multi-class case, we need base classifiers that can handle with multiple classes. For this purpose, we use the ECOC framework explained before. Given a set of N classes, n different bipartitions (groups of classes) are formed, and n binary problems over the partitions are trained. As a result, a codeword of length n is obtained for each class, where each position (bit) of the code corresponds to a possible prediction value of a given binary classifier (generally $\{+1, -1\}$). We define a coding matrix M where each row is a codeword, where $M \in \{-1, +1\}^{N \times n}$ in the binary code case. Nonetheless, binary as well as ternary codes can be used in the coding strategy [14]. How to construct this matrix is an open problem, but in our experiments we will use the *one-versus-one* strategy cited above.

During the decoding process, applying the n binary classifiers, a code x is obtained for each data sample in the test set. In the SSL model, this code is obtained using the mere predicted label. However, it would be desirable that the base classifiers can provide not only a prediction but a measure of confidence for each class. The use of confidences gives a more precise information about the decisions of the first classifier than just its prediction. Given a set of possible labels $\mathcal{L} = \{\lambda_1, \ldots, \lambda_n\}$, we have n membership confidences,

$$\hat{F}^0(\mathbf{x}, \mathcal{L}) = \{F(y = \lambda_1|\mathbf{x}), \ldots, F(y = \lambda_n|\mathbf{x})\} \tag{1}$$

In order to obtain these confidence maps we need base classifiers to generate not only a class prediction, but also its confidence. 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. In the Adaboost case, we apply a sigmoid function that normalizes

Adaboost margins from the interval $[-\infty, \infty]$ to $[-1, 1]$ (the same as codeword interval) by means of the following equation,

$$f(x) = \frac{1 - e^{-\beta m_x}}{1 + e^{-\beta m_x}} \tag{2}$$

where m_x is the margin of the predicted label given by Adaboost algorithm for the example 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.

Once we have the normalized code by applying Equation 2, we use a soft distance d for decoding. The obtained code is compared to the codewords $\mathcal{C}_i, i \in [1, \ldots, N]$ of each class defined in the matrix M, as follows:

$$F(y = \lambda_1 | x_j) = e^{-\alpha d(\mathcal{C}_1, f(x_j))}, \ldots, F(y = \lambda_n | x_j) = e^{-\alpha d(\mathcal{C}_N, f(x_j))}$$

where d may be any soft distance, as the Euclidean distance for example, $\alpha = -\ln(\epsilon)/2$, and ϵ is arbitrarily small positive quantity. By applying this to the all data samples $x_j \in \mathbf{x}$ we have the confidence maps for each class as expressed in Equation 1 that will be used in the next step.

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 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: $z = J(\hat{F}^0(\mathbf{x}, \mathcal{L}), \rho, \Sigma)$.

Starting from the confidence maps $\hat{F}^0(\mathbf{x}, \mathcal{L})$, 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 field by a gaussian mask of variable σ, where σ defines the grade of the decomposition. This means that the bigger the sigma is, the longer interactions are considered. Thus, at each level of decomposition all the points have information from the rest accordingly to the sigma parameter. Given a set of $\Sigma = \{\sigma_0, ..., \sigma_n\} \in \mathbb{R}^+$ and all the predicted confidence maps $\hat{F}^0(\mathbf{x}, \mathcal{L})$, each level of the decomposition is computed as follows,

$$\hat{F}^{s_i}(\mathbf{x}, \mathcal{L}) = g^{\sigma_i}(\mathbf{x}) * \hat{F}^0(\mathbf{x}, \mathcal{L})$$

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 we have the multi-scale decomposition, we define the support lattice z. This is, the sampling performed over the multi-scale representation in order to

obtain 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 and with origin in the current prediction example. Thus, the elements covered by the window is $w = (2\rho + 1)^d$ around the origin. For the sake of simplicity, we use a fixed radius of length $\rho = 1$. Then, for each scale i considered in the previous decomposition ($\sigma_i \quad i = 1 \dots n$), the window is stretched in each direction using a displacement proportional to the scale we are analyzing. We use a displacement $\delta_i = 3\sigma_i + 0.5$. 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, that are appended to the extended data set, is equal to $(2\rho + 1)^d \times |\Sigma| \times |\mathcal{L}|$, where $|\mathcal{L}|$ is the number of classes. According to this, we can see that the extended data set increases with the number of classes the problem has. This can be a problem if the number of classes is large, since the second classifier have to deal with many features, and therefore, the learning time is increased. In the next subsection, a grouping approach for the extended features is explained.

3.3 Extended Data Set Grouping: A Compression Approach

The goal of grouping the extended data set is to compress its number of features. Using the above approach, we can see that a confidence map is obtained for each class.Then, for each map, a multi-scale decomposition is computed, and finally, a sampling around each input example is performed. To reduce the number of confidence maps, we add a compression process between the multi-scale decomposition and the sampling process. This compression is done following information theory by means of partitions. Let be $\{P^1, P^2\}$ a partition of groups of classes and $\mathcal{L} = \{\lambda_i\}$ the set of all the classes, such that $P^1 \subseteq \mathcal{L}$ and $P^2 \subseteq \mathcal{L} \mid P^1 \cup P^2 = \mathcal{L}$ and $P^1 \cap P^2 = \emptyset$, the confidence maps are encoded as

$$\mathcal{F}^{s_j}(\{P^1, P^2\}) = \sum_i \hat{F}^{s_j}(\mathbf{x}, \lambda_i \in P^1) - \sum_i \hat{F}^{s_j}(\mathbf{x}, \lambda_i \in P^2)$$

for all the scales $s_j \in \Sigma$. Then, using a coding strategy, a minimum set of partitions $\mathcal{P} = \{P^1, P^2\}_1, \cdots, \{P^1, P^2\}_\alpha$ is defined, where $\alpha = \lceil \log_2 |\mathcal{L}| \rceil$ is the minimum number of partitions for compressing all the classes.

Following this compression approach, now the support lattice z is defined over $\mathcal{F}^{\Sigma}(\mathcal{P})$, this is, over all the scales in Σ and all the partitions in $\mathcal{P}$. Therefore, the number of features in z is reduced to $(2\rho + 1)^d \times |\Sigma| \times \lceil \log_2 |\mathcal{L}| \rceil$.

4 Experiments and Results

Before presenting the results, we discuss the data, methods and validation protocol of the experiment.

- **Data:** We test our multi-class methodology on the e-trims database [12]. The e-trims database is comprised of two datasets, 4-class with four annotated object classes and 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. Examples of this database and ground-truth for 4-class and 8-class are shown in Figure 5.
- **Methods:** We test our method using all the confidences maps, named *MMSSL Standard* and using the compression approach, named *MMSSL Compressed*. The settings for both experiments are the same, the only difference is the number of generated features in the extended dataset. We have used as base classifier a Real Adaboost ensemble of 100 decision stumps. For each image in the training set we perform a stratified sampling of 3000 pixels per image. For each example (pixel) the feature vector is only composed of RGB attributes. This data is used for training the first classifier using leave-one-image-out to produce confidence maps. 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 $\Sigma = \{1, 2, 4, 8\}$. Observe that in order to compute the neighbors of each pixel the whole left-out image is classified. Finally, both classifiers are trained using the same feature samples without and with the extended set, respectively.

  Furthermore, we have performed an experiment using multi-label optimization via α-expansion [13]. Using the confidence maps for each class obtained from the first classifier, we have applied the α-expansion optimization. For the neighborhood term, we have took into account the intensity of the 4-connected pixels.
- **Validation:** We have performed each experiment using six disjoint folds, where 50 images are used as train set and 10 images used as test.

Tables 3 and 4 show accuracy, overlapping, sensitivity and specifity averaged for all the classes. The accuracy of both MMSSL approaches in both databases are very similar. This fact reinforce that our idea of grouping features without losing performance is correct. The main advantage for using the compression approach is that reducing the number of features in the extended dataset, the time of the learning phase for the second classifier is reduced as well. In this way if the number of classes increases the approach of MMSSL is still feasible.

Figures 3 and 4 show the overlapping for each class in 4-class and 8-class database respectively. Large classes as building or sky have more overlapping than the rest. Observe that class road/pavement in 4-class or road in 8-class have a significantly larger overlapping than the Adaboost approach, this means that using only the RGB features, this class is difficult to distinguish, but the

Table 1. Method results for 4-class database

Method	Accuracy	Overlapping	Sensivity	Specifity
MMSSL Standard	0.8380	0, 6425	0, 7398	0, 795
MMSSL Compression	0.8358	0, 6390	0, 7342	0, 8001
Optimization α-expansion	0.5503	0, 3940	0, 6121	0, 5471
Adaboost	0.6200	0, 4366	0, 5634	0, 6278

Table 2. Method results for 8-class database

Method	Accuracy	Overlapping	Sensivity	Specifity
MMSSL Standard	0.7015	0, 4248	0, 5390	0, 5941
MMSSL Compression	0.7070	0, 4528	0, 5685	0, 6443
Optimization α-expansion	0.5138	0, 2570	0, 3927	0, 3529
Adaboost	0.6200	0, 2915	0, 4031	0, 4347

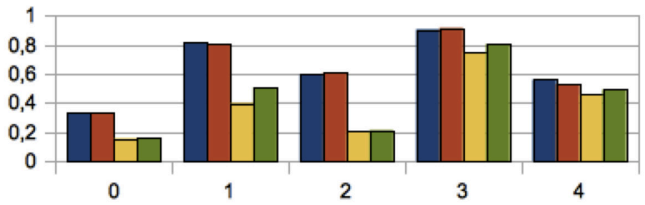

Fig. 3. Method Overlapping for 4-class

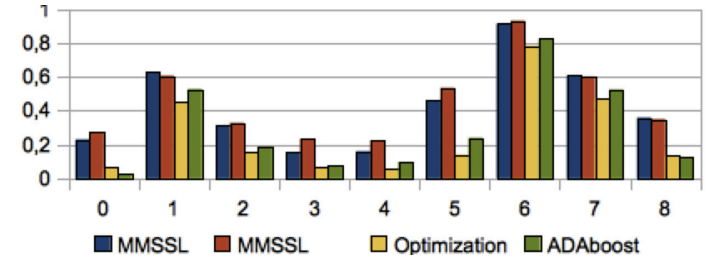

Fig. 4. Method Overlapping for 8-class

relationship among the rest classes makes it easier (road/pavement class is at bottom of the building).

Figure 5 shows results for some images in the 4-class and 8-class dataset. Observe that MMSSL methods grasp the relationship between adjacent classes and ADAboost can not. In addition, we have observed that using the optimization α-expansion, the results are poor. This means that the classification performed by the first classifier is not good enough to generalize from the surrounding pixels. On contrary, our MMSSL method can extract longer interrelations between samples and, thus, obtain better results.

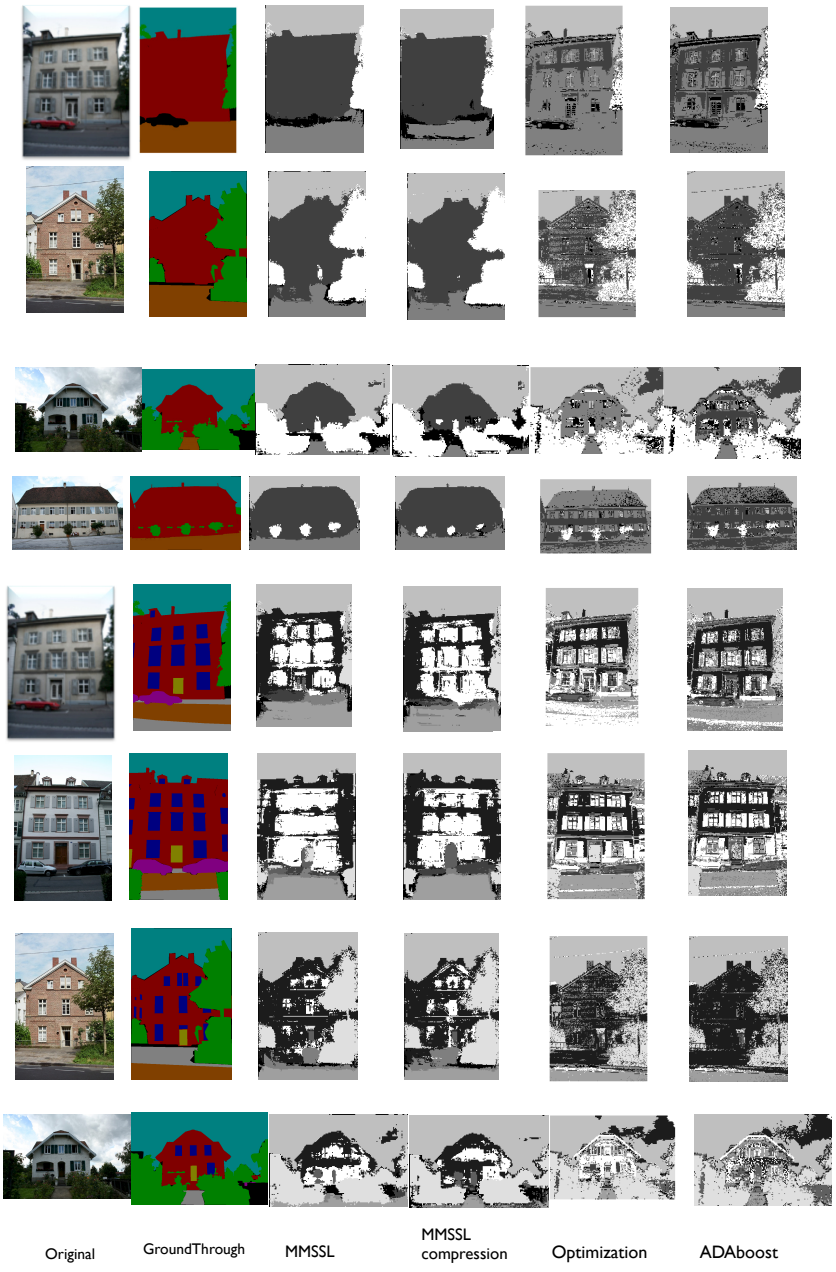

Fig. 5. Etrims 4-class (four top rows) and 8-class (four bottom rows) results

5 Conclusions

In this paper we adapted multi-scale sequential learning (MSSL) to the multi-class case (MMSSL). First, we have introduced the ECOC Framework in the base classifiers. Next, we show how to compute the confidence maps using the normalized margins obtained from the base classifiers. Another important contribution is the compression approach we have used for reducing the number of features in the extended data set. Taking into account that the experiments was done using only RGB features, the effect of sequential features from MMSSL is highly remarkable in our results. Moreover, they also shows that in terms of accuracy the loss of information during de compression process is negligible, but the amount of reduced features is considerable.

References

1. Allwein, E., Schapire, R., Singer, Y.: Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers. J. Machine Learning Research 1, 113–141 (2002)
2. Dietterich, T.G., Bakiri, G.: Solving Multiclass Learning Problems via Error-Correcting Output Codes. J. Artificial Intelligence Research 2, 263–286 (1995)
3. Dietterich, T.G.: Machine Learning for Sequential Data: A Review. In: Caelli, T.M., Amin, A., Duin, R.P.W., Kamel, M.S., de Ridder, D. (eds.) SPR 2002 and SSPR 2002. LNCS, vol. 2396, pp. 15–30. Springer, Heidelberg (2002)
4. Dietterich, T.G., Ashenfelter, A., Bulatov, Y.: Training conditional random fields via gradient tree boosting. In: Proc. of the 21th ICML (2004)
5. Nilsson, N.J.: Learning Machines. McGraw-Hill, New York (1965)
6. Cohen, W.W., de Carvalho, V.R.: Stacked sequential learning. In: IJCAI 2005, pp. 671–676 (2005)
7. McCallum, A., Freitag, D., Pereira, F.: Maximum entropy markov models for information extraction and segmentation. In: Proc. of ICML 2000, pp. 591–598 (2000)
8. Friedman, J., Hastie, T., Tibshirani, R.: Additive logistic regression: a statistical view of boosting. Annals of Statistics 28 (2000)
9. Wolpert, D.H.: Stacked generalization. Neural Networks 5(2), 241–259 (1992)
10. Lafferty, J.D., McCallum, A., Pereira, F.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Proc. of ICML 2001, pp. 282–289 (2001)
11. Pujol, O., Puertas, E., Gatta, C.: Multi-scale stacked sequential learning. In: Benediktsson, J.A., Kittler, J., Roli, F. (eds.) MCS 2009. LNCS, vol. 5519, pp. 262–271. Springer, Heidelberg (2009)
12. Korč, F., Förstner, W.: eTRIMS Image Database for Interpreting Images of Man-Made Scenes, TR-IGG-P-2009-01, University of Bonn (2009)
13. Boykov, Y., Funka-Lea, G.: Graph Cuts and Efficient N-D Image Segmentation. I. Journal of Computer Vision 70(2), 109–131 (2006)
14. Escalera, S., Tax, D., Pujol, O., Radeva, P., Duin, R.: Subclass Problem-dependent Design of Error-Correcting Output Codes. IEEE T. in Pattern Analysis and Machine Intelligence 30(6), 1041–1054 (2008)
15. Mottl, V., Dvoenko, S., Kopylov, A.: Pattern recognition in interrelated data: The problem, fundamental assumptions, recognition algorithms. In: Proc. of the 17th ICPR, Cambridge, UK vol. 1, pp. 188–191 (2004)