Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence

A Stratified Strategy for Efficient Kernel-based Learning

Simone Filice(†), **Danilo Croce**(‡), **Roberto Basili**(‡)
(†) Dept. of Civil Engineering and Computer Science Engineering
(‡) Dept. of Enterprise Engineering
University of Roma, Tor Vergata, Italy
{filice,croce,basili}@info.uniroma2.it

Abstract

In Kernel-based Learning the targeted phenomenon is summarized by a set of explanatory examples derived from the training set. When the model size grows with the complexity of the task, such approaches are so computationally demanding that the adoption of comprehensive models is not always viable. In this paper, a general framework aimed at minimizing this problem is proposed: multiple classifiers are stratified and dynamically invoked according to increasing levels of complexity corresponding to incrementally more expressive representation spaces. Computationally expensive inferences are thus adopted only when the classification at lower levels is too uncertain over an individual instance. The application of complex functions is thus avoided where possible, with a significant reduction of the overall costs. The proposed strategy has been integrated within two well-known algorithms: Support Vector Machines and Passive-Aggressive Online classifier. A significant cost reduction (up to 90%), with a negligible performance drop, is observed against two Natural Language Processing tasks, i.e. Question Classification and Sentiment Analysis in Twitter.

Introduction

Kernel methods, discussed in (Shawe-Taylor and Cristianini 2004) have been employed in many Machine Learning algorithms, such as Support Vector Machines (Vapnik 1998) achieving state-of-the-art performances in many tasks. One drawback of expressive but complex kernel functions, such as Sequence (Cancedda et al. 2003) or Tree kernels (Collins and Duffy 2001) is the time and space complexity required both in the learning and classification phases, that may prevent the kernel adoption in real world applications or in scenarios involving large amount of data.

In this paper, we mitigate this issue by decomposing the learning process into a sequence of incrementally complex ones: in this way, a stratified model is acquired, whereas first layers are always applied to *simple* examples, and more expressive representations are employed *just when needed*. Notice that learning algorithms usually apply the same inductive inference to each example. This may lead to unnecessarily high computational costs when simple examples are managed by a complex model or, dually, to coarse errors made by naïve approaches. It is interesting to decompose a target model into layers, according to a specific notion of complexity. Ideally, if the complexity of each example were known a priori, an $ad\text{-}hoc$ strategy could be applied to each example. While, in general, the complexity of an example is unknown, it could be estimated through the classification confidence. Let us consider margin classifiers, such as the Perceptron (Rosenblatt 1958). From a geometric perspective, the confidence can be inferred by the distance of an example from the classification hyperplane: the higher is the distance, the higher is the confidence in the response of the classifier. This intuition can be applied to pick the proper decision from a pool of classifiers, that are stratified according to their expressivity and complexity. If an example is classified according to a simpler (and more efficient) decision function with a satisfactory confidence, the analysis ends; on the contrary, the example is analyzed by the more complex classifiers until a satisfactory confidence is achieved. The proposed stratifies strategy is applied to two state-of-the-art algorithms, Support Vector Machine (SVM) and the Passive Aggressive (PA) algorithm (Crammer et al. 2006), that is one of the most popular Online Learning algorithms. Even thought SVM generally achieves better performance, PA is even more appealing as it incrementally modifies the learned models without a complete retraining.

Our framework is general and applicable to a variety of application domains and tasks. However, we focused on Natural Language Processing tasks whereas increasingly complex kernels correspond to increasingly expressive linguistic representations. Two semantic processing tasks are thus considered in the experimental evaluation, i.e. Question Classification and Sentiment Analysis in Twitter. In both cases, lower levels of the stratified model are characterized by linear (and efficient) learning algorithms based on just surface lexical information, while upper levels make use of more complex kernels able to better generalize lexical and syntactic information. Results suggest that a stratified learning approach is effective as no significant performance drop is experimented, while a significant reduction (up to 90%) of kernel computations is achieved.

In the remaining, first we discuss related works. Then, the stratified framework is defined and instantiated to SVM and PA. Finally, the experimental evaluations are reported.

Related Work

The proposed approach consists in a combination of classifiers and differs from the majority of Ensemble Learning approaches since its main purpose is improving the efficiency in addition to enhancing classification accuracy. In Adaboost, (Freund and Schapire 1997), a "weak" classifier is iteratively applied, focusing on all the examples misclassified at previous stages. Complexity and expressiveness do not increase at each stage. In literature, some ensemble methods have also been used to reduce the computational cost. In (Xu et al. 2013) and (Kusner et al. 2014) classifiers are organized according to a tree taxonomy, called Cost-Sensitive Tree of Classifiers (CSTC), automatically built to reduce the average test-time complexity: each test example traverses the tree of classifiers along different paths, that focus on specific and reduced subsets of features, with a significant reduction in the classification time. The main limit of the CSTC is that it requires the representation space to be made explicit and it cannot be applied to richer but implicit representation spaces as those obtained through kernel functions. An efficient classification schema has been adopted in (Viola and Jones 2001) for the for Visual Object Detection task. The underlying idea is that it is often possible to rapidly determine where an object might occur in the entire image. Weak classifiers are used to detect the more promising regions, then complex processing performed by the following classifiers is only targeted to the selected regions. In (Weiss, Sapp, and Taskar 2012) a similar approach, applied to structured classification tasks like Handwriting Recognition and Part-of-Speech Tagging, were proposed. A cascade of classifiers of increasing complexity was used to support fast processing: all the classifiers were consecutively invoked, and the information provided by the previous models was used to reduce the search space of the later classification stages. The reduction of computational costs of kernel-based learning algorithms has been traditionally tackled by imposing a budget in the number of support vectors (Cesa-Bianchi and Gentile 2006; Dekel and Singer 2006; Orabona, Keshet, and Caputo 2008; Wang and Vucetic 2010; Filice et al. 2014). However, in complicated tasks, they usually need large budgets systematically triggering many kernel computations.

Kernel-based Language Learning. Kernels can directly operate on variegate forms of representation, such as feature vectors, trees, sequences or graphs (Haussler 1999). In Natural Language Learning specific kernels can be defined to capture different linguistic information. This similarity functions can be exploited by learning algorithms to obtain an expressive model for a specific linguistic phenomenon. Three kernel functions, each emphasizing a set of particular linguistic aspects, can be considered.

Bag of Word Kernel, LIN_{bow}: A basic kernel function exploits lexical information, expressed as the word overlap between texts; documents are represented as vectors whose binary dimensions suggests the presence of different terms.

Lexical Semantic Kernel, RBF_{add}: A second kernel function generalizes the lexical information, without exploiting any manually coded resource. Lexical information is obtained by a co-occurrence *Word Space* built accordingly to the methodology described in (Sahlgren 2006). A word-by-context matrix M is computed through large-scale corpus analysis. Latent Semantic Analysis (Landauer and Dumais 1997) is then applied as follows. The matrix M is decomposed through Singular Value Decomposition (SVD), every word is projected in the reduced k-dimensional Word Space, and texts are represented by *additive linear combinations* (*add*), (Mitchell and Lapata 2010). The resulting kernel is then defined as the Radial Basis Function kernel between vector pairs, (Cristianini, Shawe-Taylor, and Lodhi 2002).

Smoothed Partial Tree Kernel (SPTK_{grct}) Tree kernels exploit syntactic similarity through the idea of convolutions among substructures. Any tree kernel evaluates the number of common substructures between two trees without explicitly considering the whole fragment space (Collins and Duffy 2001). SPTK (Croce, Moschitti, and Basili 2011) main characteristic is its ability to measure the similarity between syntactic tree structures which are partially similar and whose lexical nodes (i.e. words) can differ but are semantically related. The notion of word similarity can be automatically acquired though a distributional analysis of texts, i.e. the above Word Space.

Individual kernels give rise to classifiers guided by a specific linguistic perspective. These can be combined through the (linear) combination of individual functions as this is still a valid kernel. Kernels in the combination are normalized in order to balance their contribution.

A Stratified Classification Approach

Automatic Classification is the task of identifying the category $y \in Y$ associated to an observation $x \in X$, on the basis of a training data set containing instances whose membership is known. In our setting, we will consider binary classification tasks, i.e. $y \in \{-1, 1\}$, and instances x consisting in a series of r different representations of the same object, i.e. $x = x^1, x^2, \ldots, x^r$. Each representation x^i belongs to a different space $\mathcal{X}^i$.

Let $\mathcal{U} = \{\mathcal{X}^1, \ldots, \mathcal{X}^r\}$ be the set of r different spaces, and $\mathcal{H}^1, \mathcal{H}^2, \ldots, \mathcal{H}^c$ be c different families of classification hypothesis, each one trained in a non empty subset of spaces derived from $\mathcal{U}$. This allows to consider an implicit feature space obtained combining different kernels, (Shawe-Taylor and Cristianini 2004). $\mathcal{H}^1$ could be the hypothesis space of the linear classification functions operating on BoW feature vectors, while $\mathcal{H}^2$ could be the space of the classification functions that simultaneously exploit a syntactic tree and a Latent Semantic vector in a particular multi-kernel approach. Let $f^i : \mathcal{X}^i \rightarrow \mathbb{R}$ be the classification function belonging to the $\mathcal{H}^i$ classification hypothesis space, and let $p_t^i \in \mathbb{R}$ its prediction on the t-th example x_t. In the most generic approach, we can assume to have $c - 1$ confidence indicators $e^1, \ldots, e^{c-1}$ each one producing a boolean confidence score associated to the correspondent predictions $p_t^1, \ldots, p_t^{c-1}$. As summarized in Algorithm 1, we propose a stratified learning strategy that proceeds as follows: starting from the first classifier f^1, the example x_t is consecutively evaluated by each f^i classifier until a suitable confident prediction is found, or all the c classifiers have been invoked. Then, the $l \leq c$ predictions of the invoked classifiers are ag-

Algorithm 1 c-level Stratified Classifier

```
for i:=1 to c do
  p_t^i := f^i(x_t^i);
  if e_i(p_t^i) = false ∨ i = c then
    return v_i(p_t^1, ..., p_t^i);
  end if
end for
```

gregated by an *Ensemble Combination Rule* $v_l(p_t^1, \ldots, p_t^l)$, and a single output p_t is produced. The i-th classifier will evaluate only those examples that all the previous $i-1$ classifiers predict with low confidence. It means that, if classifiers are ordered with increasing complexity levels, the most expensive evaluations are saved for all the examples that earlier classifiers unambiguously classify. Let us consider $c = 2$ stratified classifiers operating on the same d-dimensional feature vector space. The first layer is a linear classification function producing unambiguous predictions the 90% of the times, and the second classifier is a kernel machine adopting a Radial Basis Function (RBF) Kernel. The first layer has a complexity of d (i.e. a single dot product), while the second one is $d|SV|$ (i.e. a kernel operation for each support vector), thus depending on the number of support vectors $|SV|$. When $|SV|$ support vectors are selected, the non-stratified kernel machine would approximately require $d|SV|$ operations, while the proposed approach just $d(1 + 0.1|SV|)$. Notice how the stratified strategy also applies to the learning process: later (i.e. more complex) classifiers could be trained only on instances classified with poor confidence by lower level classifiers. The advantage is two-folds: only the small subset of the problematic examples is provided to the upper levels, drastically reducing the learning time; furthermore, it allows the upper levels to be specialized in treating complex examples.

Multiple predictions (one for each invoked level) are related to a single example, and must be combined in order to have a unique output. When $l \leq c$ classifiers are invoked with predictions $p_t^1, \ldots, p_t^l$, several ensemble combination rules can be defined. We explored three different policies $v_l(p_t^1, \ldots, p_t^l)$ producing the final prediction p_t: in **Maximal Complexity policy** (MCompl) the prediction associated to the highest invoked level is chosen, i.e. $p_t = p_t^l$, assuming that this level has the most complex and accurate reasoning; in **Maximal Confidence policy** (MConf) the most confident of the l predictions is selected, i.e. $p_t = p_t^i$ where $i = argmax_{j \leq l} |p_t^j|$; finally, in **Average Confidence policy** (Aver) the mean of the predictions is computed, i.e. $p_t = \frac{1}{l} \sum_{i=1}^{l} p_t^i$. Obviously, the proposed policies do not affect the classification strategy at the different levels, so they have no impact on the resulting computational cost.

Stratified Margin Classifiers

The general proposed framework does not specify which particular learning algorithm corresponds to each level of the stratified model, neither functions $e_1, \ldots, e_c$ that validate the confidence of a prediction. Maximum margin classifiers perfectly fit in the stratified framework. Their classification function $f(x)$ is a hyperplane separating the training instances. In the linear case, instances are represented as feature vectors $\mathbf{x}$ in a d-dimensional space, and the hyperplane is explicit, i.e. $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$. This linear version is extremely attractive as it is characterized by a complexity of $\mathcal{O}(d)$ (i.e., a dot product in the d-dimensional space).

Kernel methods enable to learn nonlinear classification functions and to exploit structured data, like parse trees when tree kernels are applied (Collins and Duffy 2001). Generic data representations x can be exploited using an implicit mapping $\phi(x)$ into the Reproducing Kernel Hilbert Space (RKHS) operated by the kernel function $k(\cdot, \cdot)$. The classification function in the RKHS is thus $f_t(x) = \sum_{x_i \in SV} \alpha_i k(x_i, x) + b$ where SV is the set of support vectors. The higher expressivity of kernel methods is paid in terms of computational cost, as a single classification involves the computation of $|SV|$ different kernel operations. This can prevent the adoption of kernels for complex tasks over large datasets, as a large number of support vectors is required. The suggested stratified strategy applies to such a computational issue in natural language tasks. As shallow linguistic features allow to achieve good results in several tasks, e.g. text classification (Moschitti and Basili 2004), we first exploit simple classifiers, i.e. linear algorithms. They are supposed to handle most examples and only when the result is uncertain, kernelized (i.e., more expressive) classifiers are used. It drastically reduces the overall classification cost and minimizes linguistic pre-processing costs, e.g. the full parsing, when not needed by "shallower" classifiers.

The output provided by margin classifiers is the distance from the classification hyperplane, that heuristically reflects a confidence measure. This idea, already exploited in Active Learning (Settles 2010), can be used to model a simple confidence indicator $e(p)$ for margin classifiers, such that $e(p) = true$ iff $|p| < m$, where m is a proper margin defining the ambiguity region. More generally, at each level i, an ambiguity margin m_i and the corresponding confidence indicator e_i could be defined. In the following, the stratified strategy will be applied to two popular learning schemas, i.e. batch learners and online learners.

Stratified Batch Learning Algorithms. In the batch learning paradigm, the complete training dataset is supposed to be available during the learning phase. All the examples are simultaneously exploited in order to generate a model to be used in the classification phase, when predictions on new instances must be provided. Usually, the learning process aims at minimizing a cost function, and a global optimization is performed on the whole training set D. A largely popular batch learning method is the Support Vector Machine (SVM) algorithm (Vapnik 1998) that achieves state-of-art performances in many tasks. Embedding batch algorithms, like SVM, into the proposed stratified framework is quite straightforward: the first level classifier learns a classification function exploiting the complete training set D. The generated classification function f^1 is then used to classify all examples and only the uncertain ones (falling in the region defined by the ambiguity margin m_1) are selected as a training set D^2 for the second level. The procedure is recursively applied for training the increasingly complex levels,

Algorithm 2 c-level Stratified SVM learning

$D^1 := D$;
for i:=1 **to** c **do**
  $f^i := SVM_{learn}(D^i)$;
  $D^{i+1} := \{\}$;
  for all $x_j \in D^i$ **do**
    $p^j = f^i(x_j)$
    if $p^j < m_i$ **then**
      $D^{i+1} = D^{i+1} \cup \{x_j\}$;
    end if
  end for
end for

Algorithm 3 c-level Stratified PA Classifier

for all $x_t \in D$ **do**
  for i:=1 **to** c **do**
    $p_t^i := f_t^i(x_t^i)$;
    if $max(0, 1 - p_t^i y_t) > 0$ **then**
      if $i - th$ classifier is linear **then**
        $\alpha_t := y_t \cdot \min\left\{C_i(y_t), \frac{1-p_t^i y_t}{\|\mathbf{x}_t^i\|^2}\right\}$;
        $\mathbf{w}_{t+1}^i := \mathbf{w}_t^i + \alpha_t \mathbf{x}_t^i$;
      else
        $\alpha_t := y_t \cdot \min\left\{C_i(y_t), \frac{1-p_t^i y_t}{\|x_t^i\|_{\mathcal{H}^i}^2}\right\}$;
        $f_{t+1}^i(x) := f_t^i(x) + \alpha_t k(x_t^i, x)$
      end if
    end if
    if $p_t^i \geq m_i \vee i = c$ **then**
      return $v_i(p_t^1, \ldots, p_t^i)$;
    end if
  end for
end for

as reported in Algorithm 2.

Stratified Online Learning Algorithms. Online Learning (OL) differs from batch learning because each individual example is exploited as soon as it is available. A common OL procedure exploits each training example through a sequence of two steps: a classification stage, followed by a possible model correction stage. This paradigm is very appealing as its inherent updating capability allows to scale without complete re-training and to capture shifting concepts. The Passive Aggressive (PA) algorithm (Crammer et al. 2006) is one of the most popular OL approaches. The underlying idea is quite simple: when an example is misclassified, the model is updated selecting the most similar hypothesis to the current one, among the set of classification hypotheses that correctly classify the example.

The resulting stratified approach that embeds the PA algorithm is described in Algorithm 3. At each level i the prediction $p_t^i := f_t^i(x_t^i)$ is evaluated and the model is updated when a non zero hinge loss $H(p, y) = \max(0, 1 - py)$ occurs. If a linear PA formulation is applicable, the explicit hyperplane is directly updated, otherwise the novel example is added as a support vector. The weight α_t is computed accordingly to the PA-I formulation (Crammer et al. 2006). Finally, if a high-confidence response is observed ($p_t^i \geq m_i$) or the current level is the last ($i = c$) the ultimate prediction is aggregated by the specific Ensemble Combination Rule $v_i(p_t^1, \ldots, p_t^i)$.

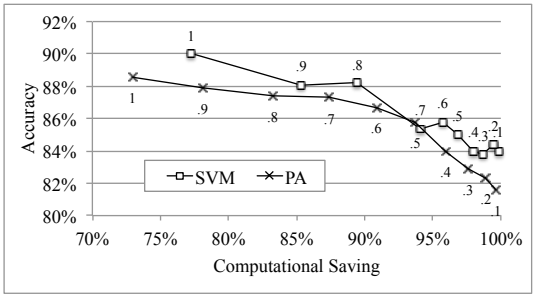

Figure 1: QC results w.r.t. Savings at a given margin (reported next to each point) with the *MCompl policy*

Experimental Evaluations

In this section the stratified strategy is evaluated w.r.t. to the Question Classification and Sentiment Analysis tasks.

Question Classification

Question classification (QC) deals with the mapping of a question into one of k predefined classes of questions, thus posing constraints on the search space of possible answers. We used the UIUC dataset (Li and Roth 2006). It is composed by a training set of 5,452 questions and a test set of 500 questions, organized in 6 coarse-grained classes.

The previously discussed kernels are evaluated to investigate the contribution of different information in our stratified model. The deepest analysis, which considers also syntax, is given by the Smoothed Partial Tree Kernel over Grammatical Relation Centered Trees ($SPTK_{grct}$), as experimented in (Croce, Moschitti, and Basili 2011). Lexical generalization within RBF_{add} and $SPTK_{grct}$ is derived through the distributional analysis of UkWaC (Baroni et al. 2009) corpus, which is a large-scale document collection made by 2 billion tokens. A co-occurrence Word-Space with a window of size ± 3 is acquired. Co-occurrencies are weighted by estimating the Point-wise Mutual Information between words. The SVD reduction is then applied with a dimensionality cut of $d = 250$. Classifier parameters are tuned with a Repeated Random Sub-sampling Validation, consisting in a 10-fold validation strategy. To emphasize the contribution of less frequent classes, the cost factor within SVM and the aggressiveness parameter within PA have been modified according to (Morik, Brockhausen, and Joachims 1999) and (Filice et al. 2014), respectively. Results are reported in terms of accuracy i.e. the percentage of correctly classified questions.

Table 1 reports results of the online learning (PA) and the batch learning (SVM) algorithms organized according to different kernels. Moreover, for each stratified analysis, different Ensemble Combination Rules are considered in the three columns *MCompl*, *MConf* and *Aver*. In each row a specific kernel is investigated and results of the monolithic (i.e. non stratified) algorithm are compared with the 2-level stratified counterpart with ambiguity margin $m_1 = 1$. The monolithic version always applies the most sophisticated and computationally expensive processing of the kernelized approach, so representing a sort of upper-bound. On the contrary, the stratified approach handles most of the instances in the first layer by a linear (i.e. fast and non-kernelized) algorithm operating on a BoW representation ($*LIN_{bow}$), while

Table 1: Results w.r.t. QC task. The * indicates linear algorithms. The - indicates a monolithic approach, where the second stage is missing.

FirstLevel	SecondLevel	Online				Batch			
		MCompl	MConf	Aver	Sav.	MCompl	MConf	Aver	Sav.
*LIN$_{bow}$	-		81.4%				84.2%		
*LIN$_{bow}$	*LIN$_{bow}$	81.9%	81.8%	82.0%	-	84.0%	84.4%	85.4%	-
RBF$_{add}$	-		85.9%		75%		91.4%		83%
*LIN$_{bow}$	RBF$_{add}$	86.7%	86.5%	87.3%		90.2%	88.8%	90.4%	
SPTK$_{grct}$	-		87.7%		74%		92.0%		77%
*LIN$_{bow}$	SPTK$_{grct}$	87.5%	87.7%	86.4%		90.2%	90.2%	91.4%	
LIN$_{bow}$+RBF$_{add}$	-		88.3%		73%		88.2%		76%
*LIN$_{bow}$	LIN$_{bow}$+RBF$_{add}$	87.7%	87.1%	86.1%		87.2%	85.8%	88.0%	
LIN$_{bow}$+SPTK$_{grct}$	-		87.8%		73%		90.4%		77%
*LIN$_{bow}$	LIN$_{bow}$+SPTK$_{grct}$	87.4%	86.6%	86.4%		89.0%	87.0%	89.2%	
LIN$_{bow}$+RBF$_{add}$+SPTK$_{grct}$	-		89.3%		73%		90.8%		77%
*LIN$_{bow}$	LIN$_{bow}$+RBF$_{add}$+SPTK$_{grct}$	88.6%	88.0%	86.6%		90.0%	88.0%	89.0%	

Table 2: Results w.r.t. the QC task. The * indicates linear algorithms.

FirstLevel	SecondLevel	Third Level	Online			Batch		
			MCompl	MConf	Aver	MCompl	MConf	Aver
SPTK$_{grct}$	-	-		87.0%			92.0%	
*LIN$_{bow}$	SPTK$_{grct}$	-	87.5%	87.7%	86.4%	90.2%	90.2%	91.4%
*LIN$_{bow}$	RBF$_{add}$	SPTK$_{grct}$	87.9%	88.2%	88.6%	88.0%	90.0%	91.6%

the second level is a kernelized classifier. The * symbol indicates that a linear algorithm is applied. As an example, the third row shows the comparison between a monolithic setting that operates with a SPTK$_{grct}$ kernel and the stratified counterpart. The monolithic *LIN_{bow}, that employs the simplest representation and the most efficient linear algorithm, is to be considered a lower-bound. As expected, the introduction of lexical generalization in the RBF$_{add}$ kernel is beneficial, while results are further improved introducing syntax through the SPTK$_{grct}$.

The effectiveness of the stratification is measured in terms of *Computational Saving* (see column *Sav.* in table 1) as the percentage of kernel operations avoided in the second layer w.r.t. the corresponding monolithic algorithm during the classification of the test set. For example, the monolithic PA operating with a SPTK$_{grct}$ kernel and the stratified counterpart with ambiguity margin $m_1 = 1$ achieve the same accuracy (87.7%), although the stratified PA guarantees a drastic reduction in the computational complexity: the stratified PA performs only 26% of the kernel operations w.r.t. the monolithic PA, with a computational saving of 74%. It is worth noticing that the ambiguity margin represents a model parameter that has a deep impact on the trade-off between accuracy and computational cost, as shown in Figure 1. The curves are obtained varying the ambiguity margin from 0.1 to 1 using the best configuration of both SVM (i.e. SPTK$_{grct}$) and PA (i.e. LIN$_{bow}$+RBF$_{add}$+SPTK$_{grct}$). The larger is the ambiguity margin, the higher is the computational cost. Values of m larger than 1 were not beneficial as just higher computational costs are obtained, both in training and testing. The achieved results are straightforward: all the considered kernels confirm the robustness of the stratified approach with accuracies close to the upper-bounds, even with lower ambiguity margins. At the same time, the computational saving is impressive: for example, with an ambiguity margin of 0.6 for the PA and 0.8 for the SVM, no significant accuracy drop is observed, but the computational saving is approximately 90%, i.e. the total computational cost is reduced of an order of magnitude. The number of times the kernel machine is invoked drastically decreases both in testing and training; this leads to a reduction of the number of training examples needed at the second level, that produces a more compact model, i.e. a classifier with fewer SVs. Regarding the ensemble combination policies, in the OL setting better results are achieved by using the *Maximal Complexity policy* (*MComp*), so neglecting the predictions of the weaker classifiers. It is different in the batch setting where the *Maximal Confidence policy* (*Mconf* setting) and *Average Confidence policy* (*Aver*) achieve better results: the SVM algorithm performs a global optimization and provides a robust solution even exploiting the simple LIN$_{bow}$.

Beside the efficiency of linear algorithms with respect to complex kernel-based machines, one has to consider that the complexity of kernel functions can deeply vary. For example the RBF$_{add}$ kernel requires a single operation on vector pairs, while the same operation is needed to be executed in the SPTK$_{grct}$ a number of times that tends to be quadratic w.r.t. the number of lexical nodes in a tree, as in (Croce, Moschitti, and Basili 2011). The overall cost of stratifying several complex kernels would be thus reduced by applying first the less expensive operations. We thus evaluated a 3-level stratified approach, in which *LIN$_{bow}$, RBF$_{add}$ and SPTK$_{grct}$ are consecutively exploited. The first two levels operate as a double filtering for minimizing the number of times the most complex classifier is invoked. Table 2 reports the accuracy results obtained with this 3-level approach with ambiguity margins $m_1 = m_2 = 1$. The results achieved by this stratified schema are very promising in both Online and Batch settings. The 3-level PA is able to overcome the monolithic SPTK$_{grct}$, avoiding 89,4% of the tree kernel computa-

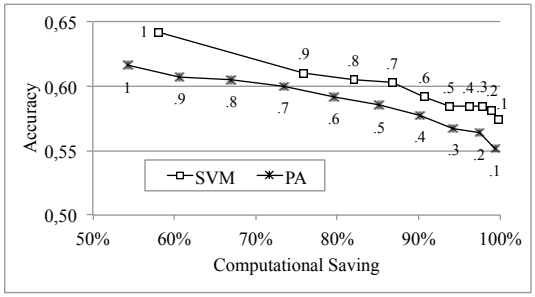

Figure 2: SA results w.r.t. the Saving at a given margin (reported next to each point) with the *MCompl policy*

tions, and the 2-level SPTK$_{grct}$, preventing 59.3% of the tree kernel computations. The three-level SVM achieves results very close to the monolithic and the 2-level counterparts, with a computational saving of 91.8% and 65.2%.

Sentiment Analysis in Twitter

Twitter represents an intriguing source of information as it is used to share opinions about brands, products, or situations (Jansen et al. 2009). The growing interest in the automatic interpretation of opinion statements makes Sentiment Analysis on Twitter a hot topic, which is challenging as tweets are short, informal and characterized by their own particular language. This task is also interesting as an online learning schema is supposed to better adapt to a very dynamic scenario like micro-blogging. The evaluation of the stratified approach is carried out on the SemEval-2013 Task 2 corpus, (Wilson et al. 2013). In particular, we focus on the *Message Polarity Classification*: the entire tweet is classified w.r.t. three classes *positive*, *negative* and *neutral*. The training dataset is composed of $10,205$ annotated tweets, while the test dataset is made of $3,813$ tweets. The same classifier parametrization of QC has been carried out. Again, different kernels are evaluated: a linear kernel operating on a BoW representation ($*LIN_{bow}$) and the radial basis function kernel over an additive linear combination of terms from a Semantic Word Space (RBF_{add}). The SPTK is not used here, as the low quality of parse tree obtained from tweets, a problem discussed in (Foster et al. 2011), would compromise the overall kernel contribution. The Word Space is acquired through the analysis of a generic corpus made of 3 million of tweets; the same setting of the previous section has been adopted. A pre-processing stage is applied: fully capitalized words are converted in lowercase; reply marks, hyperlinks, hashtags and emoticons are replaced by special tokens.

In Table 3 we evaluated a 2-level stratified approach with ambiguity margin $m_1 = 1$. Results have the same setting of Table 1, but they are numerically reported in terms of average F1-Measure between the positive and negative classes, i.e. the metric adopted in the SemEval challenge. Again, a linear (i.e. non kernelized) algorithm operating on a BoW representation (*LIN$_{bow}$) is less expressive and achieves lower results. Word generalization is confirmed to be beneficial, as demonstrated by the RBF$_{add}$ kernel adoption.

Figure 2 shows the trade-off between F1 mean and the *Computational Saving*. The empirical findings of the previous section are still valid: the robustness of the stratified approach is confirmed with performances close to the upperbounds, even with low ambiguity margins. The computational saving is still relevant: in the PA with an ambiguity margin of 0.7, no significant F1 drop is observed, but the computational saving is approximately 70%. The task is more complex than the previous one (i.e. QC) and this impacts on the Computational Saving, that is generally lower with respect to the previous section, due to the fact that many lower confidence predictions occur so that complex kernels are triggered several times. Regarding the ensemble combination rules, experimental results suggest again that the *MConf* is the most effective in the online schema, while the *MConf* or *Aver* are the most suitable in batch learning. The 2-stratified PA would have ranked 8^{th} (with a $F1 = 0.616$) over 36 systems, w.r.t. systems trained without using external annotated tweets. The 2-stratified SVM would have ranked 5^{th} (with a $F1 = 0.642$). It is straightforward, considering that top systems in the challenge used hand-made lexicons (Wilson et al. 2013).

Conclusion and Future work

In this paper we proposed a Stratified Learning framework where multiple classifiers are combined and dynamically invoked according to the incremental complexity of the corresponding representation space. We applied the strategy to SVM and PA learning algorithms by studying its applicability to state-of-the-art learning algorithms as well as to online learning contexts. In the experimental evaluations, no significant performance drop is experimented, while a dramatic reduction of the processing cost is achieved. Overall, the proposed strategy represents an easily applicable yet effective strategy for any classification tasks, in batch as well as in online settings. It facilitates the adoption of expressive kernels, mitigating their inherent computational complexity. The design of the kernel cascades evaluated here is heuristically biased by the expressiveness of the involved kernel functions and their corresponding computational cost. Future work will focus on how the proper cascade and the corresponding ambiguity margins can be analytically determined, in order to achieve a mathematical proof of its optimal trade-off between the inductive inference accuracy and overall computational cost.

References

Baroni, M.; Bernardini, S.; Ferraresi, A.; and Zanchetta, E. 2009. The wacky wide web: a collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation* 43(3):209–226.

Cancedda, N.; Gaussier, É.; Goutte, C.; and Renders, J.-M. 2003. Word-sequence kernels. *Journal of Machine Learning Research* 3:1059–1082.

Cesa-Bianchi, N., and Gentile, C. 2006. Tracking the best hyperplane with a simple budget perceptron. In *In proc. of the nineteenth annual conference on Computational Learning Theory*, 483–498. Springer-Verlag.

Collins, M., and Duffy, N. 2001. Convolution kernels for natural language. In *Proceedings of Neural Information Processing Systems (NIPS'2001)*, 625–632.

Table 3: Results w.r.t. SA task. The * indicates linear algorithms. The - indicates a monolithic approach, where the second stage is missing.

FirstLevel	SecondLevel	Online				Batch			
		MCompl	MConf	Aver	Sav.	MCompl	MConf	Aver	Sav.
*LIN_{bow}	-		0.548		-		0.570		-
*LIN_{bow}	*LIN_{bow}	0.552	0.553	0.554		0.572	0.575	0.578	
RBF_{add}	-		0.584		55%		0.617		62%
*LIN_{bow}	RBF_{add}	0.611	0.615	0.608		0.609	0.620	0.642	
LIN_{bow}+RBF_{add}	-		0.631		54%		0.654		58%
*LIN_{bow}	LIN_{bow}+RBF_{add}	0.616	0.606	0.597		0.642	0.643	0.636	

Crammer, K.; Dekel, O.; Keshet, J.; Shalev-Shwartz, S.; and Singer, Y. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research* 7:551–585.

Cristianini, N.; Shawe-Taylor, J.; and Lodhi, H. 2002. Latent semantic kernels. *J. Intell. Inf. Syst.* 18(2-3):127–152.

Croce, D.; Moschitti, A.; and Basili, R. 2011. Structured lexical similarity via convolution kernels on dependency trees. In *Proceedings of EMNLP*.

Dekel, O., and Singer, Y. 2006. Support vector machines on a budget. In Schlkopf, B.; Platt, J.; and Hoffman, T., eds., *NIPS*, 345–352. MIT Press.

Filice, S.; Castellucci, G.; Croce, D.; and Basili, R. 2014. Effective kernelized online learning in language processing tasks. In *ECIR*, 347–358.

Foster, J.; Çetinoglu, Ö.; Wagner, J.; Roux, J. L.; Hogan, S.; Nivre, J.; Hogan, D.; and van Genabith, J. 2011. #hardtoparse: Pos tagging and parsing the twitterverse. In *Analyzing Microtext*.

Freund, Y., and Schapire, R. E. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.* 55(1):119–139.

Haussler, D. 1999. Convolution kernels on discrete structures. In *Technical Report UCS-CRL-99-10*.

Jansen, B. J.; Zhang, M.; Sobel, K.; and Chowdury, A. 2009. Twitter power: Tweets as electronic word of mouth. *J. Am. Soc. Inf. Sci. Technol.* 60(11):2169–2188.

Kusner, M. J.; Chen, W.; Zhou, Q.; Xu, Z. E.; Weinberger, K. Q.; and Chen, Y. 2014. Feature-cost sensitive learning with submodular trees of classifiers. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada.*, 1939–1945.

Landauer, T., and Dumais, S. 1997. A solution to plato's problem: The latent semantic analysis theory of acquisition, induction and representation of knowledge. *Psychological Review* 104.

Li, X., and Roth, D. 2006. Learning question classifiers: the role of semantic information. *Natural Language Engineering* 12(3):229–249.

Mitchell, J., and Lapata, M. 2010. Composition in distributional models of semantics. *Cognitive Science* 34(8):1388–1429.

Morik, K.; Brockhausen, P.; and Joachims, T. 1999. Combining statistical learning with a knowledge-based approach – a case study in intensive care monitoring. In *International Conference on Machine Learning (ICML)*, 268–277.

Moschitti, A., and Basili, R. 2004. Complex linguistic features for text classification: A comprehensive study. *Advances in Information Retrieval* 181–196.

Orabona, F.; Keshet, J.; and Caputo, B. 2008. The projectron: a bounded kernel-based perceptron. In *Proceedings of ICML '08*, 720–727. USA: ACM.

Rosenblatt, F. 1958. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review* 65(6):386–408.

Sahlgren, M. 2006. *The Word-Space Model*. Ph.D. Dissertation, Stockholm University.

Settles, B. 2010. Active Learning Literature Survey. Technical Report 1648, University of Wisconsin–Madison.

Shawe-Taylor, J., and Cristianini, N. 2004. *Kernel Methods for Pattern Analysis*. New York, NY, USA: Cambridge University Press.

Vapnik, V. N. 1998. *Statistical Learning Theory*. Wiley-Interscience.

Viola, P. A., and Jones, M. J. 2001. Rapid object detection using a boosted cascade of simple features. In *CVPR (1)*, 511–518. IEEE Computer Society.

Wang, Z., and Vucetic, S. 2010. Online passive-aggressive algorithms on a budget. *Journal of Machine Learning Research - Proceedings Track* 9:908–915.

Weiss, D. J.; Sapp, B.; and Taskar, B. 2012. Structured prediction cascades. *CoRR* abs/1208.3279.

Wilson, T.; Kozareva, Z.; Nakov, P.; Ritter, A.; Rosenthal, S.; and Stoyonov, V. 2013. Semeval-2013 task 2: Sentiment analysis in twitter. In *Proceedings of the 7th International Workshop on Semantic Evaluation*. Association for Computational Linguistics.

Xu, Z. E.; Kusner, M. J.; Weinberger, K. Q.; and Chen, M. 2013. Cost-sensitive tree of classifiers. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, volume 28 of *JMLR Proceedings*, 133–141. JMLR.org.