An online framework for learning novel concepts over multiple cues

Luo Jie[1,2], Francesco Orabona[1], and Barbara Caputo[1]

[1] Idiap Research Institute, Centre du Parc, Martigny, Switzerland
[2] École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland
{jluo,forabona,bcaputo}@idiap.ch

Abstract. We propose an online learning algorithm to tackle the problem of learning under limited computational resources in a teacher-student scenario, over multiple visual cues. For each separate cue, we train an online learning algorithm that sacrifices performance in favor of bounded memory growth and fast update of the solution. We then recover back performance by using multiple cues in the online setting. To this end, we use a two-layers structure. In the first layer, we use a budget online learning algorithm for each single cue. Thus, each classifier provides confidence interpretations for target categories. On top of these classifiers, a linear online learning algorithm is added to learn the combination of these cues. As in standard online learning setups, the learning takes place in rounds. On each round, a new hypothesis is estimated as a function of the previous one. We test our algorithm on two student-teacher experimental scenarios and in both cases results show that the algorithm learns the new concepts in real time and generalizes well.

1 Introduction

There are many computer vision problems that are intrinsically *sequential.* In these problems the system starts learning from impoverished data sets and keeps updating its solution as more data is acquired. Therefore the system must be able to continuously learn new concepts, as they appear in the incoming data. This is a very frequent scenario for robots in home settings, where it is very likely to see something unknown [1] in a familiar scene. In such situations the robot cannot wait to collect enough data before building a model for the new concept, as it is expected to interact continuously with the environment. Limited space and computing power may also constrain the algorithm from being actually implemented, considering that the stream of training data can be theoretically infinite. Still, most of the used algorithms for computer vision are intrinsically *batch*, that is they produce a solution only after having seen enough training data. Moreover they are not designed to be updated often, because most of the time updating the solution is possible only through a complete re-training.

A different approach is the *online learning* framework [2]. This framework is motivated by a teacher-student scenario, that is when a new concept is presented to the machine, the machine (*student*) can ask the user (*teacher*) to provide a label. This scenario would correspond to the case of a user explaining to the robot a detected source of novelty in a scene. The algorithms developed in the online learning framework are intrinsically designed to be updated after each sample is received. Hence the computational complexity per update is extremely low, but their performance is usually lower than similar batch algorithms.

Ideally, we would like to have an online method with performance as high as batch based algorithms, with fast learning and bounded memory usage. Existing online algorithms (see for example [3, 4]) fail to satisfy all these conditions. The mainstream approaches attempt to keep the same performance of batch based methods while retaining either fast learning or bounded memory growth but not both. On the other hand, multiple-cues/sources inputs guarantee diverse and information-rich sensory data. They make it possible to achieve higher and robust performance in varied, unconstrained settings. However, when using multiple inputs, the expansion of the input space and memory requirements is linearly proportional to the number of inputs as well as the computational time, for both the training and test phase.

Some recent works in online learning applied to computer vision include: Monteleoni and Kääriäinen [5] present two active learning algorithms in the online classification setting and test it on an OCR application; Fink et. al. [6] who describe a framework for online learning and present preliminary results on office images and face recognition. Grangier and Bengio [7] propose a Passive-Aggressive algorithm [3] for Image Retrieval, which takes advantage of the efficient online learning algorithms. On the multi-cues literature, a recently proposed approach to combine cues in the batch setting is to learn the weights of the positive weighted sum of kernels [8]. Even if many attempts have been done to speed up the training process [9, and references therein], this approach is still slow and does not scale well to big datasets. Moreover these methods are intrinsically non-incremental, hence they cannot be used in a sequential setting. A theoretically motivated method for online learning over multiple cues has been proposed in [10], however they assume that all the cues live in the same space, meaning that the same kernel must be used on all the cues.

In this work we tackle the problem of learning from data using an online learning algorithm over multiple visual cues. By combining online learning with multiple cues, we manage to get the best of both worlds, i.e. high performance, bounded memory growth and fast learning time. The proposed algorithm is tested on two experimental scenarios: the first is place recognition which simulates the student-teacher scenario where the robot is shown an indoor environment composed of several rooms (this is the kitchen, this is the corridor, etc), and later it is supposed to localize and navigate to perform assigned tasks. The second is object categorization which simulates the student-teacher scenario where the autonomous agent is presented a collection of new objects. For both scenarios, results show that the algorithm learns the new concepts in real time and generalizes well to new concepts.

In the next section we describe the online learning framework and the building blocks that we will use in our online multi-cues architecture (Section 2-3).

Section 4 describes our experimental findings. Finally, we conclude the paper with a summary and a discussion on possible future research.

2 Online Learning

Online learning is a process of continuous updating and exploitation of the internal knowledge. It can also be thought of as learning in a teacher-student scenario. The teacher shows an instance to the student who predicts its label. Then the teacher gives feedback to the student. An example of this would be a robot which navigates in a closed environment, learning to recognize each room from its own sensory inputs. Moreover, to gain robustness and increase the classification performance, we argue for the need of learning using multiple cues. Hence our goal is to design an online learning algorithm for learning over multiple features from the same sensor, or data from multiple sensors, which is able to take advantage of the diverse and information-rich inputs and to achieve more robust results than systems using only a single cue. In the following we will introduce the online learning framework and we will explain how to extend it to multiple cues. Due to space limitations, this is a very quick account of the online learning framework — the interested readers are referred to [2] for a comprehensive introduction.

2.1 Starting from Kernel Perceptron

In online setting, the learning takes place in rounds. The online algorithm learns the mapping $f : \mathcal{X} \rightarrow \mathbb{R}$ based on a sequence of examples $\{\mathbf{x}_t, y_t\}_{t=1}^{l}$, with instance $\mathbf{x}_t \in \mathcal{X}$ and label $y_i \in \{-1, 1\}$. We denote the hypothesis estimated after the t-th round by f_t. At each round t, the algorithm receives a new instance $\mathbf{x}_t$, then it predicts a label $\hat{y}_t$ by using the current function, $\hat{y}_t = \text{sign}(f_t(\mathbf{x}_t))$, where we could interpret $|f(\mathbf{x})|$ as the confidence in the prediction. Then, the correct label y_t is revealed. The algorithm changes its internal model everytime it makes a mistake or the confidence on the prediction is too low. Here, we denote the set of all attainable hypotheses by $\mathcal{H}$. In this paper we assume that $\mathcal{H}$ is a Reproducing Kernel Hilbert Space (RKHS) with a positive definite kernel function $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ implementing the inner product which satisfies the reproducing property, $f(\mathbf{x}) = \langle k(\mathbf{x}, \cdot), f(\cdot) \rangle$.

Perhaps the most well known online learning algorithm is Rosenblatt's Perceptron algorithm [11]. On the t-th round the instance $\mathbf{x}_t$ is given, and the algorithm makes a prediction $\hat{y}_t$. Then the true label is revealed: if there is a prediction mistake, i.e. $\hat{y}_t \neq y_t$, it updates the hypothesis, $f_t = f_{t-1} + y_t k(\mathbf{x}_t, \cdot)$, namely it stores $\mathbf{x}_t$ in the solution. Otherwise the hypothesis is left intact, $f_t = f_{t-1}$. Given the nature of the update, the hypothesis f_t can be written as a kernel expansion [12], $f_t(\mathbf{x}) = \sum_{i \in \mathcal{S}_t} \alpha_i k(\mathbf{x}_i, \mathbf{x})$. The subset of instances used to construct the function is called the *support set*. Although the Perceptron is a very simple algorithm, it has been shown to produce very good results. Several other algorithms (see Passive-Aggressive [3] and the references therein) can be seen as belonging to the Perceptron algorithm family. However, given that they update each time there is an error, if the problem is not linearly separable, they will never stop adding new instances to the support set. This will eventually lead to a memory explosion. As we aim to use the algorithm in applications where data must be acquired continuously in time, a Perceptron algorithm cannot be used as it is. Hence we will use as a basic component of our architecture the Projectron++ algorithm [13].

2.2 The Projectron++ Algorithm

The Projectron++ [13] algorithm is a Perceptron-like algorithm bounded in space and time complexity. It has a better mistake bound than Perceptron. The core idea of the algorithm comes from the work of Downs et. al. [14] on simplifying Support Vector Machine solutions. Hence, instead of updating the hypothesis every time a prediction mistake is made, or when the prediction is correct with low confidence[3], the Projectron++ first checks if the update can be expressed as a linear combination of vectors in the support set, i.e. $k(\mathbf{x}_t, \cdot) = \sum_{i=1}^{t-1} d_i k(\mathbf{x}_i, \cdot) = P_{t-1}(k(\mathbf{x}_t, \cdot))$, where $P_{t-1}(\cdot)$ is the projection operator. The concept of linear independence can be approximated and tuned by a parameter η that measures the quality of the approximation. If the instance can be approximated within an error η, it is not added to the support set but the coefficients in the old hypothesis are changed to reflect the addition of the instance. If the instance and the support set are linearly independent, the instance is added to the set, as Perceptron. We refer the reader to [13] for a detailed analysis.

3 Online Multi-Cues Learning Algorithm

In this section we describe our algorithm for learning over multiple cues. We adapt the idea of *high-level* integration from the information fusion community (see [15] for a comprehensive survey), and design our algorithm with a two-layers structure. The first layer is composed of different Projectrons++, one for each cue. The second layer learns online a weighted combination of the classifiers of the first layer, hence we interpret the output of the Projectrons++ on the first layer as confidence measures of the different cues.

A lot of work has been done on how to select the best algorithm from a pool of prediction algorithms, such as the Weighted Majority algorithm [16]. However, they usually assume black-box classifiers. Here we want to learn the best combination of classifiers, not just picking the best one. Therefore we train *both* the single cue classifiers and the weighted combination with online algorithms. In the rest of this section, we will describe our algorithm in the binary setup. For multi-class problems, the algorithm is extended using the multi-class extension method presented in [17]; we omit the detailed derivation for lack of space.

Suppose we have N cues of the same data $\mathbf{X} = \{\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^N\}$. Each cue is described by a feature vector $\mathbf{x}^i \in \mathbb{R}^{m_i}$, where $\mathbf{x}^i$ could be the feature vector

[3] that is when $0 < y_t f_{t-1}(\mathbf{x}_t) < 1$

Algorithm 1 OMCL Algorithm

```
Input: Projectron++ parameter $\eta \geq 0$;    Passive-aggressive parameter $C > 0$.
Initialize: $f_0^i = \mathbf{0}, \mathcal{S}_0^i = \emptyset$, where $i = 1, 2, \ldots, N$;    $\omega_0 = \mathbf{0}$.
for $t = 1, 2, \ldots, T$ do
  Receive new instance $\mathbf{x}_t^i$, where i=1, 2, ..., N
  Predict $\hat{h}_t^i = f_{t-1}^i(\mathbf{x}_t^i)$, where i=1, 2, ..., N
  Predict $\hat{y}_t = \text{sign}(\omega_\mathbf{t} \cdot \hat{\mathbf{h}}_\mathbf{t})$
  Receive label $y_t$
  for $i = 1, 2, \ldots, N$ do
    Loss: $l1_t^i = max(0, 1 - y_t \cdot \hat{h}_t^i)$
    if $l_t^i > 0$ then
      Compute projection error $\Delta$
      if $y_t = \hat{h}_t^i$ or $\Delta \leq \eta$ then
        Projection update: $f_t^i = f_{t-1}^i + \alpha_t^i y_t P_{t-1}^i(k(\mathbf{x}_t^i, \cdot))$
      else
        Normal update: $f_t^i = f_{t-1}^i + y_t k(\mathbf{x}_t^i, \cdot)$
      end if
      Update hypothesis: $\tilde{\mathbf{h}}_t^i = f_t^i(\mathbf{x}_t^i)$
    end if
  end for
  Loss: $l2_t = max(0, 1 - y_t \omega_\mathbf{t} \cdot \tilde{\mathbf{h}}_\mathbf{t})$
  Set: $\tau_t = min(C, \frac{l2_t}{\|\tilde{\mathbf{h}}_t\|^2})$
  Update: $\omega_\mathbf{t+1} = \omega_\mathbf{t} + \tau_t y_t \tilde{\mathbf{h}}_\mathbf{t}$
end for
```

associated with one feature descriptor or one input sensor. Suppose also we are given a sequence of data $\{\mathbf{X}_t, y_t\}_{t=1}^l$, where $y_t \in \{-1, 1\}$. On round t, in the first layer, N Projectrons++ [13] learn the mapping for each views: $f_t^i : \mathbf{x}_t^i \rightarrow h_t^i$, where $h_t^i \in \mathbb{R}$, for $i = 1, 2, \ldots, N$. On top of the Projectron++ classifiers, a linear Passive-Aggressive [17] algorithm is used to learn a linear weighted combination of the confidence outputs of the classifiers: $\omega_t : \mathbf{h}_t \rightarrow \mathbb{R}$. The prediction of the algorithm is $\hat{y}_t = \text{sign}(\omega_t \cdot \mathbf{h}_t)$.

After each update of the first layer, we also update the confidences on the instances before passing them to the second layer. We denote by $\tilde{h}_t^i$ the confidence of the updated hypotheses and denote by $\hat{h}_t^i$ the confidence predictions of the Projection++ classifiers before knowing the true label. Hence, instead of updating the second layer based on $\hat{h}_t^i$, the linear Passive-Aggressive algorithm on top considers the new updated confidence $\tilde{h}_t^i$. This modified updating rule prevents the errors propagating from the first layer to the second layer, and in preliminary experiments it has shown to be less likely prone to over-fitting. We call this algorithm OMCL (Online Multi-Cue Learning, see Algorithm 1).

3.1 Online to Batch Conversion

Online algorithms are meant to be constantly used in teacher-student scenarios. Hence the update process will never stop. However it is possible to transform them to batch algorithms, that is to stop the training and to test the current hypothesis on a separate test set. It is known that when an online algorithm stops the last hypothesis found can have an extremely high generalization error. This is due to the fact that the online algorithms are not converging to a fixed solution, but they are constantly trying to "track" the best solution. If the samples are Independent & Identically Distributed (IID), to obtain a good batch solution one can use the *average* of all the solutions found, instead of the last one. This choice gives also theoretical guarantees on the generalization error [18].

Our system produces two different hyperplanes at each round, one for each layer. In principle we could simply average each hyperplane separately, but this would break the IID assumption of the inputs for the second layer. So we propose an alternative method: given that the entire system is linear, it can be viewed as producing only one hyperplane at each round, that is the product of the two hyperplanes. Hence we average this unique hyperplane and in the testing phase we predict with the formula: $\text{sign}\left(\frac{1}{T}\sum_{t=1}^{T}\omega_{\mathbf{t}}\cdot\hat{\mathbf{h}}_{\mathbf{t}}\right)$.

Note that, as f_t can be written as a kernel expansion [12], the averaging does not imply any additional computational cost, but just an update of the coefficients of the expansion. We use this approach as it guarantees a theoretical bound and it was also found to perform better in practice.

4 Experiments and Results

In this section, we present an experimental evaluation of our approach on two different scenarios, corresponding to two publicly available databases. The first scenario is about place recognition for a mobile robot, and the experiments were conducted on the IDOL2 dataset [19]. The second scenario is about learning new object categories, and the experiments were conducted on the ETH80 dataset [20]. Both experiments can be considered as a teacher-student scenario, where the system is taught to recognize rooms (or objects) by a human tutor. Therefore the robot has to learn the concepts in real time, and generalize well to new instances. For all experiments, we compared the performance and the memory requirements to the standard Perceptron algorithm by replacing the Projectron++ algorithm in our framework. We also compared our algorithm to two different cues combination algorithms: the "flat" data structure and the majority voting algorithm. The "flat" structure is simply a concatenation of all the features of different cues into a long feature vector, and we trained a Projectron++ classifier for it. The majority voting algorithm predicts the label by choosing the class which receives the highest number of votes. As for a majority voting algorithm in multi-class case, the number of experts (number of cues in our experiments) required by the algorithm which guarantees a unique solution will grow exponentially with the number of classes. Although it does not happen very often in practice, we show that sometimes two or more classes receive an equal number of votes, especially when the number of cues is relatively small compared to the number of classes. We determined all of our online learning and kernel parameters via cross-validation. Our implementation of the proposed

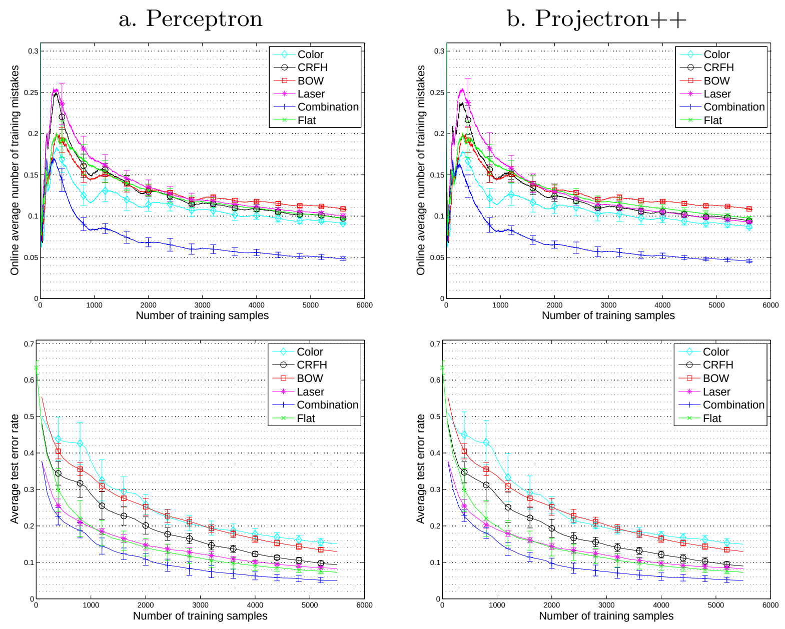

Fig. 1. Average online training error rate and recognition error rate for the test set on IDOL2 dataset as a function of the number of training samples.

algorithm uses the DOGMA package [21]; the source code is available within the same package.

4.1 First Scenario: Place Recognition

We performed the first series of experiments on the IDOL2 database [19], which contains 24 image sequences acquired using a perspective camera mounted on two mobile robot platforms. These sequences were captured with the two robots moving in an indoor laboratory environment consisting of five different rooms (one-person office (OO), corridor (CR), two-person office (TO), kitchen (KT) and Printer Area (PA)); they were acquired under various weather and illumination conditions (sunny, cloudy, and night) and across a time span of six months. We considered the scenario where our algorithm has to incrementally update the model, so to adapt to the variations captured in the dataset.

For experiments, we used the same setup described in the original paper [19] (Section V, Part B). We considered the 12 sequences acquired by robot Dumbo, and divided them into training and testing sets, where each training sequence has a corresponding one in the test sets captured under roughly similar conditions. Similarly, each sequence was divided into five subsequences. Learning is done in chronological order, i.e. how the images were captured during acquisition of the dataset. During testing, all the sequences in the test set were taken into account. In total, we considered six different permutations of training and testing sets.

cue	OO	CR	TO	KT	PA	ALL
Color	28.4	9.5	8.1	9.9	31.4	17.4
CRFH	14.2	4.1	15.4	15.4	11.1	12.0
BOW	21.5	6.9	17.5	11.4	8.4	13.1
Laser	7.6	3.7	8.5	10.7	12.9	8.7
OLMC	7.6	2.5	1.9	6.7	13.3	6.4
Optimal	4.9	3.0	3.7	4.0	3.9	3.9
Flat	5.7	2.7	7.5	8.5	11.8	7.2
Vote	11.7 (6%)	3.3 (2%)	5.3 (3%)	5.2 (3%)	9.1 (7%)	6.9 (4%)

Table I: Place recognition error rate using different cues after the last training round. Each room is considered separately during testing, and it contributes equally to the overall results as an average. It shows that the OLMC algorithm achieves better performance than that of using each single cue. For the "vote" algorithm, the percentage of test data which have two or more classes receive equal number of votes is reported in the bracket below the error rate. Hence the algorithm can not make a definite prediction. Therefore we considered that the algorithm made a prediction error.

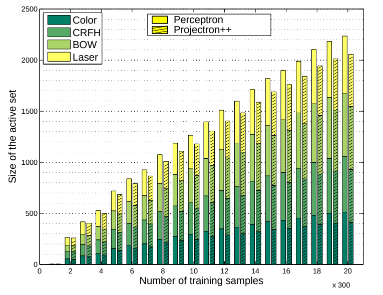

Fig. 2. Average size of support set for different algorithms on IDOL2 dataset as a function of the number of training samples.

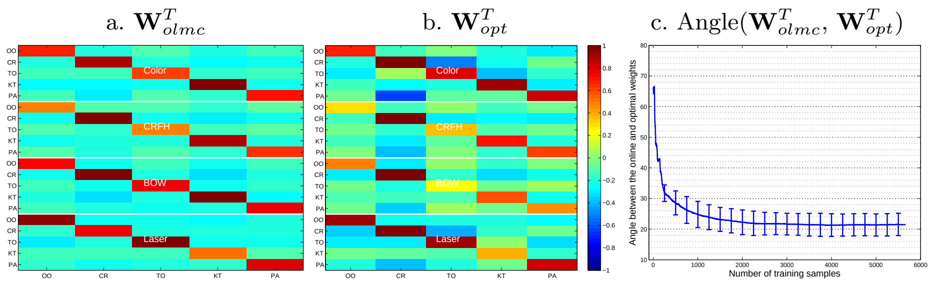

Fig. 3. Visualization of the average normalized weights for combining the confidence outputs of the Projectron++ classifiers: a. the weights obtained by our algorithm at last training round; b. the weights obtained by solving the above optimization problem. c. the angles between the two vectors $\mathbf{W}_{opt}$ and $\mathbf{W}_{olmc}$ as function of the number of training samples.

The images were described using three image descriptors, namely, CRFH [22] (Gaussian derivative along x & y direction), Bag-of-Words [23] (SIFT, 300 visual words) and RGB color histogram. In addition, we also used a simple geometric feature from the Laser Scan sensor [24].

Fig. 1 reports the average online training and recognition error rate on the test set, where the average online training error rate is the number of prediction mistakes the algorithm makes on a given input sequence normalized by the length of the sequence. Fig. 2 shows the size of the support sets as a function of the number of training samples. Here, the size of the support set of Projectron++ is close to that of Perceptron. This is because the support set of Perceptron is already very compact. Since the online training error rate is low, both algorithms do not update very frequently. In Table I we summarize the results using each cue after finishing the last training round. We see that our algorithm outperforms both the "flat" data structure and the majority vote algorithm. The majority vote algorithm could not make a definite prediction on approximately 4% of the test data, because there are two or more classes which received an equal number of votes.

Moreover, we would like to see what is the difference in performance between the learned linear weights for combining confidence outputs of the Projectron++ classifiers and the optimal linear weighted solution. In another words, what is the best performance a linear weighted combination rule can achieve? We obtained an optimal combination weights, denoted as $\mathbf{W}_{opt}$, by solving a convex optimization program (see Appendix A) on the confidence outputs of the Projectron++ classifiers on the test set. We reported the result in Table I, which shows that our algorithms achieve performance similar to that of the optimal solution. We also visualized the average normalized weights for both the optimal solution and the weights obtained by our algorithm at the last learning round in Fig. 3a&b. From these figures, we can see that the weights on the diagonal of the matrix, which corresponds to the multi-class classifiers' confidence interpretations on the same target category, have highest values. Fig. 3c reports the average angle between the two vectors $\mathbf{W}_{opt}$ and $\mathbf{W}_{olmc}$, which is the weights obtained by our algorithms during the online learning process. We can see that the angle between these two vectors gradually converges to a low value.

4.2 Second Scenario: Object Categorization

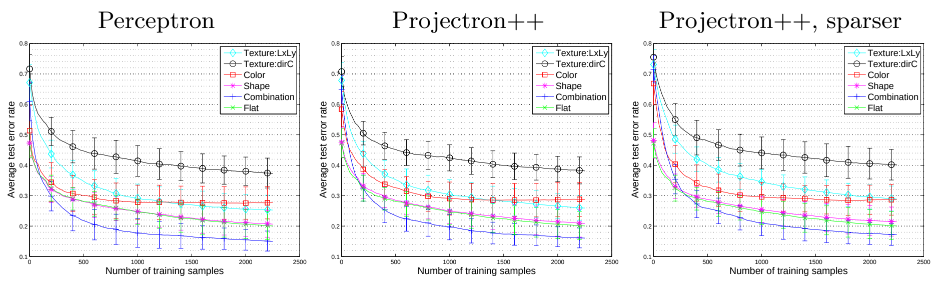

Fig. 4. Average online recognition error rate for the categorization of never seen objects on ETH-80 dataset as a function of the number of training samples. We see that all algorithms achieve roughly similar performance (Projectron++ is slightly better), the Perceptron converges earlier than the Projectron++ algorithms.

We tested the algorithm on the ETH-80 objects dataset [20]. The ETH-80 dataset consists of 80 objects from eight different categories (apple, tomato, pear, toy-cows, toy-horses, toy-dogs, toy-cars and cups). Each category contains 10 objects with 41 views per object, spaced equally over the viewing hemisphere, for a total of 3280 images. We use four image descriptors: one color feature (RGB color histogram), two texture descriptors (Composed Receptive Field Histogram

Cues	apple	car	cow	cup	dog	horse	pear	tomato	all
L_xL_y	26.3	4.1	52.4	29.3	29.2	49.4	7.0	9.6	25.9
$DirC$	25.0	21.9	60.7	43.3	65.9	55.7	29.8	3.2	38.2
Color	38.9	18.8	15.5	16.2	49.3	57.7	33.0	1.2	28.8
Shape	30.4	1.3	28.6	2.5	33.3	28.9	3.4	37.8	20.8
OLMC	11.1	1.5	17.4	13.7	34.1	44.2	5.0	1.1	16.0
Flat	29.3	1.5	28.2	2.0	32.1	28.0	3.3	35.1	20.0
Vote	24.2 (19%)	1.3 (1%)	30.1 (15%)	11.3 (9%)	34.0 (18%)	41.2 (20%)	3.6 (3%)	6.1 (5%)	19.0 (11%)

Table II: Categorization error rate for different objects using different cues after finishing the last training round. We could see that our algorithm outperforms the "Flat" structure, the "Vote" algorithm and the case when using each cue alone. It also shows that some cues are very descriptive of certain objects, but not of the others. For example, the color feature achieves almost perfect performance on tomato, but its performance on other objects is low. It also supports our motivation on designing multi-cues algorithms. For the "vote" algorithm, the percentage of test data which have tow or more classes receives equal number of votes is reported in the bracket.

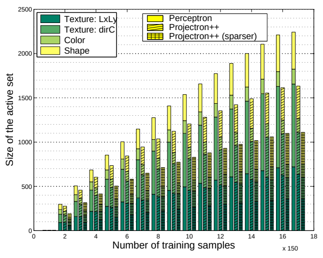

Fig. 5. Average size of support set for different algorithms on ETH80 dataset as a function of the number of training samples.

(CRFH) [22] with two different kinds of filters: Gaussian derivative L_xL_y and gradient direction ($DirC$) and a global shape feature (centered masks). We randomly selected 7 out of the 10 objects for each category as training set, and the rest of them as test set. All the experiments were performed over 20 different permutations of the training set.

We first show the behavior of the algorithms over time. In Fig. 4, we show the average recognition error on never seen objects as a function of the number of learned samples. In the experiments, we used two different setting of η parameters, labeled as Projectron++ and Projectron++, sparser. The growth of the support set as a function of the number of samples is depicted in Fig. 5. We see that the Projectron++ algorithm obtain similar performance as the Perceptron algorithm with less than 3/4 (Projectron++) and 1/2 (Projectron++, sparser) of the size of the support set. Finally, in Table II we summarize the error rate using different cues for each category after finishing the last training round (Projectron++).

5 Discussion and Conclusions

We presented an online method for learning from multi-cues/sources inputs. Through experiments on two image datasets, representative of two student-teacher scenarios for autonomous systems, we showed that our algorithm is able to learn a linear weighted combination of the marginal output of classifiers on each sources, and that this method outperforms the case when we use each cue alone. Moreover, it achieves performance comparable to the batch performance [19, 20] with a much lower memory and computational cost. We also showed that the budget Projectron++ algorithm had the advantage of reducing the support set without removing or scaling instances in the set. This keeps performance high, while reducing the problem of the expansion of the input space and memory requirement when using multiple inputs. Thanks to the robustness gained by using multiple cues, the algorithm could reduce more the support set (e.g. Projectron++, sparser, see Fig. 4c & Fig. 5) without any significant loss in performance. This trade-off would be a potentially useful function for applications working in a highly dynamic environment and with limited memory resources, particularly for systems equipped with multiple sensors. Thanks to the efficiency of the learning algorithms, both learning and predicting could be done in real time with our Matlab implementation on most computer hardwares which run Matlab software.

In the future, we would like to explore theoretical properties of our algorithm. It is natural to extend our algorithms to the active learning setup [5] to reduce the effort of data labeling. Meanwhile, it would be interesting to explore the properties of co-training the classifier using the messages passed from some other classifiers with high confidence on predicting certain cues.

Acknowledgments We thank Andrzej Pronobis for sharing the laser sensor features. This work was sponsored by the EU project DIRAC IST-027787.

References

1. Weinshall, D., Hermansky, H., Zweig, A., Jie, L., Jimison, H., Ohl, F., Pavel, M.: Beyond novelty detection: Incongruent events, when general and specific classifiers disagree. In: Proc. NIPS'08
2. Cesa-Bianchi, N., Lugosi, G.: Prediction, learning, and games. Cambridge University Press (2006)
3. Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., Singer, Y.: Online passive-aggressive algorithms. Journal of Machine Learning Research **7** (2006) 551–585
4. Dekel, O., Shalev-Shwartz, S., Singer, Y.: The Forgetron: A kernel-based Perceptron on a budget. SIAM Journal on Computing **37** (2007) 1342–1372
5. Monteleoni, C., Kääriäinen, M.: Practical online active learning for classication. In: Proc. CVPR'07, Online Learning for Classification Workshop
6. Fink, M., Shalev-Shwartz, S. Singer, Y., Ullman, S.: Online multiclass learning by interclass hypothesis sharing. In: Proc. ICML'06
7. Grangier, D., Bengio, S.: A discriminative kernel-based approach to rank images from text queries. IEEE Transactions on Pattern Analysis and Machine Intelligence **30**(8) (2008) 1371–1384
8. Lanckriet, G.R.G., Cristianini, N., Bartlett, P., El Ghaoui, L., Jordan, M.I.: Learning the kernel matrix with semidenite programming. Journal of Machine Learning Research **5** (2004) 27–72
9. Rakotomamonjy, A., Bach, F., Grandvalet, Y., Canu, S.: SimpleMKL. Journal of Machine Learning Research **9** (2008) 2491–2521
10. Cavallanti, G., Cesa-Bianchi, N., Gentile, C.: Linear algorithms for online multitask classification. In: Proc. COLT'09
11. Rosenblatt, F.: The Perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review **65** (1958) 386–407
12. Schölkopf, B., Herbrich, R., Smola, A., Williamson, R.: A generalized representer theorem. In: Proc. COLT'00
13. Orabona, F., Keshet, J., Caputo, B.: The Projectron: a bounded kernel-based Perceptron. In: Proc. ICML'08

14. Downs, T., Gates, K., Masters, A.: Exact simplification of support vectors solutions. Journal of Machine Learning Research **2** (2001) 293–297
15. Sanderson, C., Paliwal, K.K.: Identity verification using speech and face information. Digital Signal Processing **14**(5) (2004) 449–480
16. Littlestone, N., Warmuth, M.: Weighted majority algorithm. In: IEEE Symposium on Foundations of Computer Science. (1989)
17. Crammer, K., Singer, Y.: Ultraconservative online algorithms for multiclass problems. Journal of Machine Learning Research **3** (2003) 951–991
18. Cesa-Bianchi, N., Conconi, A., Gentile, C.: On the generalization ability of on-line learning algorithms. IEEE Trans. on Information Theory **50**(9) (2004) 2050–2057
19. Luo, J., Pronobis, A., Caputo, B., Jensfelt, P.: Incremental learning for place recognition in dynamic environments. In: Proc. IROS'07
20. Leibe, B., Schiele, B.: Analyzing appearance and contour based methods for object categorization. In: Proc. CVPR'03
21. Orabona, F.: DOGMA: a MATLAB toolbox for Online Learning. (2009) Software available at `http://dogma.sourceforge.net`.
22. Linde, O., Lindeberg, T.: Object recognition using composed receptive field histograms of higher dimensionality. In: Proc. ICPR'04
23. Sivic, J., Zisserman, A.: Video Google: A text retrieval approach to object matching in videos. In: Proc. ICCV'03
24. Mozos, O.M., Stachniss, C., Burgard, W.: Supervised learning of places from range data using adaboost. In: Proc. ICRA'05

Appendix A

Let $\{\hat{\mathbf{h}}_i, y_i\}_{i=1}^{l}$ be the confidence outputs of the Projectron++ classifiers on the test set of l instances, where each sample $\hat{\mathbf{h}}_i$ is drawn from a domain $\mathbb{R}^m$ and each label y_i is an integer from the set $\mathcal{Y} = \{1, 2, \ldots, k\}$. In the multi-class setup, $m = n \times k$, where n is the number of cues (i.e. number of the Projectron++ classifiers) and k is the number of classes. Therefore, we obtained the optimal linear solution by solving the following convex optimization problem:

$$\min_{\mathbf{W},\xi} \quad \frac{1}{2}||\mathbf{W}||^2 + C \sum_{i=1,\ldots,l,\ j\in y_i^C} \xi_{i,j}$$

$$\underset{\forall i,\ j\in y_i^C}{subject\ to}\ \overline{W}_{y_i} \cdot \hat{\mathbf{h}}_\mathbf{i} + \xi_{i,j} > \overline{W}_j \cdot \hat{\mathbf{h}}_\mathbf{i}$$

$$\xi_{i,j} \geq 0$$

where $\mathbf{W}$ is the multi-class linear weighted combination matrix of size $k \times m$, and $\overline{W}_r$ is the r-th row of $\mathbf{W}$. We denote the complement set of y_i as $y_i^C = \mathcal{Y} \backslash y_i$. This setting is a generalization of linear binary classifiers. Therefore, the value $\overline{W}_r \cdot \hat{\mathbf{h}}_\mathbf{i}$ denotes the confidence for the r class and the classifier predicts the label using the function:

$$\hat{y} = \arg\max_{r=1,\ldots,k} \{\overline{W}_r \cdot \hat{\mathbf{h}}\}$$

The regularizer $||\mathbf{W}||^2$ is introduced to prevent slack variables $\xi_{i,j}$ producing solutions close to 0^+. The cost C value is decided through cross validation.