Very Large Sets of Heuristics for Scene Interpretation (VELASH)

Charles Dubout
I&C, EPFL
IDIAP Research Institute

***Abstract*—Research in machine learning has historically relied on limited prior knowledge, usually designed by a reduced number of experts. The VELASH project does the opposite and aims at combining a very large number of heuristics (feature extractors) developed by external contributors from all horizons, to recognize as many objects as possible present in an image. Computational aspects of the learning problem are of primary concern, performance becoming critical given such a rich feature space and since we want to interact with a large number of external users. To help with the design of those heuristics, the proposed framework should provide meaningful information about the performance of an heuristic, areas in which it is useful and those in which it still needs improvement.**

***Index Terms*—Machine Learning, Object Detection, Classification, Scene Interpretation, Boosting, Feature Selection**

I. Introduction

MACHINE LEARNING in general strives to avoid the necessity for complex handcrafted prior knowledge and try to develop *universal* learning methods. The *VELASH* project, taking place in the context of the *MASH* project (http://www.mash-project.eu), under the EU's 7th Research Framework Programme (FP7), does the opposite and advocates the use of a combination of large-scale learning algorithms as a basis for high-level artificial intelligence.

The past few decades have seen a shift of ambitions in artificial intelligence research. From procedural and symbolic methods aimed at solving very ambitious tasks, the field has moved to more mathematically well-grounded techniques for classification and regression problems of modest scales. Symbolic methods allowed detailed descriptions of a priori knowledge, but had difficulties to cope with the unpredictability of real-world situations. On the other hand, today's techniques rely on statistical learning, but as those techniques aim at being *universal*, the amount of expert knowledge that they allow has typically been at a fairly low level, focusing on sophisticated signal processing and prior distributions.

Our main motivation is to get the best of both worlds and combine complex handcrafted priors with sophisticated learning-based statistical methods. If most of the other technical disciplines allow for large teams of experts to collaborate on a single complex project over a certain period of time, machine learning remains an exception. While combining learning algorithms from different groups is known to improve performance (the recently awarded *Netflix prize* is a good example), such heterogeneous models have rarely been studied, and no tool or measure exist to guide their development.

In this proposal we define an heuristic to be any feature extractor, an algorithm processing the raw signal to produce values relevant to the problem at hand. We assume that high performance can only be achieved by the combination of many of such handcrafted heuristics, and propose to develop them in an open and collaborative framework similar to the successful development process of open-source softwares and collaborative encyclopedias.

Although of general interest, we study this approach for full scene interpretation, that is to recognize as many objects as possible from a given image. Latest state-of-the-art algorithms for multiclass object classification already use multiple families of features (e.g. [1] uses 35 different feature kinds), but such techniques simply combine (most often linearly) the results obtained by classifiers trained on every family independently.

Boosting seems a good candidate as a starting block for a learning algorithm in this context, as it supports naturally training with multiple family of features, does not require all the features to be computed at the beginning, scales well with the number of features and samples, and allows various optimization techniques to be applied to speed up training.

Proposal submitted to committee: September 3th, 2010; Candidacy exam date: September 10th, 2010; Candidacy exam committee: Exam president, thesis director, co-examiner.

This research plan has been approved:

Date: _____

Doctoral candidate: _____
(name and signature)

Thesis director: _____
(name and signature)

Thesis co-director: _____
(if applicable) (name and signature)

Doct. prog. director: _____
(R. Urbanke) (signature)

Input: $\mathbf{X}, \mathbf{y}, \mathcal{H}, T$

1 **for** $i \leftarrow 1$ **to** n **do** $\mathbf{w}_i^{(1)} \leftarrow 1$

2 **for** $t \leftarrow 1$ **to** T **do**

3 $h^{(t)} \leftarrow \underset{h^{(t)} \in \mathcal{H}}{\text{argmin}}\ \epsilon^{(t)}$, where $\epsilon^{(t)} = \dfrac{\sum_{i=1, h^{(t)}(\mathbf{x}_i) \neq \mathbf{y}_i}^{n} \mathbf{w}_i^{(t)}}{\sum_{i=1}^{n} \mathbf{w}_i^{(t)}}$

4 $\alpha^{(t)} \leftarrow \frac{1}{2} log \frac{1 - \epsilon^{(t)}}{\epsilon^{(t)}}$

5 **for** $i \leftarrow 1$ **to** n **do** $\mathbf{w}_i^{(t+1)} \leftarrow \mathbf{w}_i^{(t)} e^{-\mathbf{y}_i \alpha^{(t)} h^{(t)}(\mathbf{x}_i)}$

6 **end**

7 **Output**: $f^{(T)} = \sum_{t=1}^{T} \alpha^{(t)} h^{(t)}$

Fig. 1. The pseudocode of the (discrete) *AdaBoost* algorithm. $\mathbf{X}$ is the matrix of training samples, $\mathbf{y}$ is the vector of labels, $\mathcal{H}$ is the space of possible weak learners, and T is the number of steps. $h^{(t)}$ is the optimal weak learner at step t, $\epsilon^{(t)}$ is its associated weighted error, $\alpha^{(t)}$ is its associated coefficient, and $f^{(T)}$ is the final (strong) classifier.

II. Related Works

In this section follows the descriptions of the three papers we selected, as they have to do with Boosting using multiple families of features, accelerating Boosting through smart feature selection and feature selection per se. As the first two papers build upon the *AdaBoost* algorithm, the simplest and most popular Boosting methods, it will be presented first for the sake of completeness.

A. AdaBoost

AdaBoost, short for Adaptive Boosting, is a meta-algorithm, created by Y. Freund and R. Schapire [2]. Its[1] goal is to build a *strong* classifier $f^{(T)} : \mathbb{R}^d \rightarrow \mathbb{R}$ as a linear combination of T (discrete) weak learners $h^{(t)} \in \mathcal{H} : \mathbb{R}^d \rightarrow \{-1, 1\}$, i.e. line 7 of the pseudocode of Fig. 1. A sufficient condition for $f^{(T)}$ to be called *strong* is to have zero training error in a total number of steps T logarithmic with the total number of training samples n. As the number of training errors, also called *Hamming* loss

$$R_H(f^{(T)}) = \sum_{i=1, sign\left(f^{(T)}(\mathbf{x}_i)\right) \neq y_i}^{n} 1$$

where $y_i \in \{-1, 1\}$ is the label of training sample i, is only piecewise continuous, the algorithm prefers to minimize the exponential margin loss

$$R_e(f^{(T)}) = \sum_{i=1}^{n} e^{-y_i f^{(T)}(\mathbf{x}_i)}$$

which is an upper-bound on the *Hamming* loss and has the advantage of being continuously differentiable. It does so by performing a certain number T of Boosting steps, maintaining a distribution of weights $\mathbf{w}^{(t)}$ on the samples. At each step t *AdaBoost* trains a weak learner $h^{(t)}$ on the distribution $\mathbf{w}^{(t)}$, the only requirements on $h^{(t)}$ being that is has some discriminative power with respect to those weights, i.e. its weighted error $\epsilon^{(t)}$ as defined line 3 of Fig. 1 is strictly below $\frac{1}{2}$, or said differently, it does strictly better than random guessing. The algorithm then proceed to update the weights according to the equation of line 5.

[1]Only the discrete binary *AdaBoost* algorithm is described in this paper for simplicity reasons. But it is not difficult to generalize the algorithm to deal with continuous weak learners (e.g. *Real AdaBoost*), and/or to multiclass (e.g. *AdaBoost.MH*, *SAMME*).

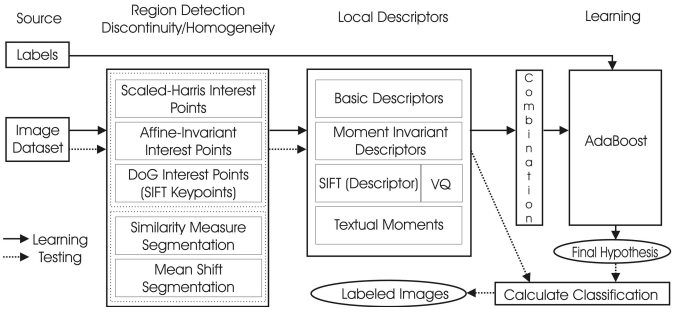

Fig. 2. The training procedure start from a set of labeled images. Regions of discontinuity and homogeneity are extracted and described forming a set of feature vectors. *AdaBoost* [2] is then used to learn a final classifier, composed of several weak learners.

B. Generic Object Recognition with Boosting

This paper by A. Opelt, A. Pinz, M. Fussenegger, and P. Auer [3] explores the limits of weakly supervised image categorization. The authors call it weakly supervised as the only information made available to the learning algorithm during training is the label of the object to categorize in the image (the algorithm thus has to determine the pose, scale, and deal with the various illumination and background conditions). To simplify the problem, the authors use only local descriptors of regions of discontinuity or homogeneity and do not take into account spatial information. They also assume that different descriptors might have different performances depending on the category of the object, and so they use Boosting as a learning method as it has no problem dealing with multiple families of features.

To learn a category the algorithm is provided with a set of training images along with a binary label indicating if a relevant object appears in the image or not. The images are first converted to gray-scale, regions of discontinuity and homogeneity are then extracted, before being encoded by local descriptors and presented to *AdaBoost* [2] (the whole process is illustrated in Fig. 2).

Contrarily to all approaches before them, the authors believe that using multiple kind of description methods should yield better results, as different object categories might be better represented using different kind of descriptions. They use two main kinds of information extractors: interest points (scale invariant Harris-Laplace, affine invariant, and DoG (SIFT keypoints)) and homogeneous regions (through mean-shift-segmentation and their own similarity-measure-segmentation). To describe the regions of discontinuity, they use four different descriptors: subsampled gray values, basic intensity moments,

Input: $(\mathcal{R}(I_k), \mathbf{y}_k, \mathbf{w}_k), \mathcal{R}(I_k) \equiv \{\mathbf{v}_{k,f} | f = 1, ..., F_k\}$

1

2 **forall the** $\mathbf{v}_{k,f}$ **do** $\mathbf{d}_{k,f,j} = \min_{1 \leq g \leq F_k} d(\mathbf{v}_{k,f}, \mathbf{v}_{j,g})$

3 **foreach** k, f **find** $\pi_{k,f}$ $s.t.$ $\mathbf{d}_{k,f,\pi_{k,f}(i)} \leq \mathbf{d}_{k,f,\pi_{k,f}(i+1)}$

4 $k, f = \underset{k,f}{\text{argmax}} \max_{s} \sum_{j=1}^{s} \mathbf{w}_{\pi_{k,f}(j)} \mathbf{y}_{\pi_{k,f}(j)}$

5 $\theta = \dfrac{\mathbf{d}_{k,f,\pi_{k,f}(s)} + \mathbf{d}_{k,f,\pi_{k,f}(s+1)}}{2}$

Output: $h(I_l) = sign\left(\theta - \min_{1 \leq g \leq F_l} d\left(\mathbf{v}_{k,f}, \mathcal{R}(I_l)_g\right)\right)$

Fig. 3. The pseudocode of the weak learner finder. $\mathcal{R}(I_k)$ is the set of description vectors of image k, $\mathbf{y}$ is the vector of labels, $\mathbf{w}$ is the weight vector, $h(I_l)$ is the output weak learner, and d is the distance function. *Remark:* if there is description vectors of different kinds, $d(\mathbf{v}_{k,f}, \mathbf{v}_{j,g})$ is done only between vectors of the same kind.

moments invariants, and SIFT. For homogeneous regions they use only two, namely intensity distribution and invariant moments [2].

The authors use a slightly modified version of discrete *AdaBoost* as learning algorithm, to add the possibility of putting different weights on positive and negatives images. The only modification to the pseudocode of Fig. 1 being the update of the weights of line 5, replaced by

$$\mathbf{w}_i^{(t+1)} \leftarrow \begin{cases} \mathbf{w}_i^{(t)} e^{-\mathbf{y}_i \alpha^{(t)} h^{(t)}(\mathbf{x}_i) + \log \eta} & \text{if } \mathbf{y}_i = 1 \text{ and} \\ & \mathbf{y}_i \neq h^{(t)}(\mathbf{x}_i) \\ \mathbf{w}_i^{(t)} e^{-\mathbf{y}_i \alpha^{(t)} h^{(t)}(\mathbf{x}_i)} & \text{otherwise} \end{cases}$$

To generate a ROC curve, the authors use a varying threshold th_{Ada} instead of the $sign$ function on $f^{(T)}$.

The only element missing to complete the algorithm is how to find the optimal weak learner at each Boosting step. The process is described in Fig. 3. From a labeled and weighted set of training images, the weak learner finder algorithm first build a distance matrix $\mathbf{d}$ containing for every description vector f in every image k the distance to the closest description vector in every image j (line 1 of the pseudocode). It then finds a permutation $\pi_{k,f}$ for each description vector, ordering the images from the closest to the farthest according to the distance matrix $\mathbf{d}$ (line 2). Those computations being independent of the weights $\mathbf{w}$, they are done only once at the beginning and saved for future evaluations. The algorithm then selects the best description vector $\mathbf{v}_{k,f}$, according to the criterion of line 3, and computes the threshold θ line 4. The criterion can be understood as a maximization of the quantity (relatively to the weights) of positives before the threshold on the distance from the selected description vector. That is, it looks if images close to $\mathbf{v}_{k,f}$ according to $\mathbf{d}$ are mostly positives, and thus if $\mathbf{v}_{k,f}$ is a good indicator of the presence of the relevant object.

[2]The paper gives various settings and tricks that the authors used to obtain their results, as well as all the references of all the extraction and description techniques cited here. They are not reported by lack of place and because they are not crucial to the understanding of the paper.

The authors then describe the various experiments that they did on various datasets (they even created their owns as they judged most of the available ones at the time too easy) and with various feature combinations. They use in their experiment $\eta = 1$, falling back to standard *AdaBoost*, as if setting it higher might improve the accuracy a bit, it does so at the expense of the training time. They also use only their own homogenous regions detector (similarity-measure-segmentation), as they found that it performs better than mean-shift-segmentation. Using only the homogeneous regions, the algorithm already outperforms all other state-of-the-art approaches (although some of the methods they compare with try to find explicitly the positions of all the relevant objects in an image, not only the category of the image), and obtain close to 100% accuracy on most categories of the Caltech Database and on Cars Side from the University of Illinois. They then experiment on their own database GRAZ-01 with various feature combinations, proving that different features might indeed perform better for different categories of objects, similarity-measure-segmentation being better to detect bikes and SIFT being better to detect persons for example. They also present some results of combinations of two kind of features on their database GRAZ-02, and shows that the combination of both is better than any of the two individually.

The paper describes a learning algorithm based on *AdaBoost* and an image representation based on local regions. The algorithm is very fast after its initial pre-processing and can use multiple families of feature. As different features have different performance depending on the image category, the best results (although very few are presented) are obtained when combining different feature extractors and descriptors.

If the goal is a bit different from the one of *MASH* (image categorization versus full scene interpretation), some ideas might prove useful. Even if *MASH* does not directly support bag of features image representation (it expects a constant number of features per sub-window), a few tricks can be employed to still use such a representation. The idea is to first find good reference vectors (the $\mathbf{v}_{k,f}$ in the weak learner finder) prior to the learning, through clustering for example, and provide as features to *MASH* all the distances between the sub-window and those reference vectors. Thresholding those distances would then yield the same results as the current weak learners.

C. Fast Boosting using adversarial bandits

This paper by R. Busa-Fekete and B. Kégl [4] seeks to improve the computational complexity of the *AdaBoost* algorithm using an adversarial multi-armed bandits algorithm. It builds on a previous paper from the same authors [5] where they used stochastic bandits, itself building upon the *LazyBoost* algorithm. The algorithm was relatively successful in practice but due to its assumption that the rewards follow a stationary distribution (completely unrealistic due to the non-stochastic nature of *AdaBoost*), no proof could be made on its convergence, and thus it could not be called a Boosting algorithm. Using instead adversarial bandits, the authors can prove a weak-to-strong-learning theorem, meaning that their algorithm remains a Boosting one.

LazyBoost[3] [7] is a slightly modified *AdaBoost*: at each Boosting step the weak learner is trained using only a fixed-size random subset of the features instead of all of them. Despite its apparent naive simplicity this modification can improve the speed of *AdaBoost* by several orders of magnitude while maintaining a similar level of accuracy, at the expense of an increased number of weak learners [8].

The *Multi-armed bandits* problem is defined as follows: there is M gambling machines (i.e. the "arms" of the bandits), and at every time-step t the gambler chooses an arm j_t, pulls it, and receives a reward $r^t_{j_t} \in [0, 1]$. In the stochastic version the rewards come from a static distribution, while in the adversarial setup there is a second player that chooses a reward vector $\mathbf{r}^t \in \mathbb{R}^M$ before every step, and only the reward $r^t_{j_t}$ corresponding to the chosen arm j_t gets revealed to the first player. There is no restriction on the series of reward vectors $\mathbf{r}^t$, particularly they can depend on the previous actions of the gambler.

One can see the weights $\mathbf{w}_i^{(t)}$ of *AdaBoost* as the current exponential loss of $f^{(T)}$ on the sample i, as the weight update of line 5 of Fig. 1 simply reflects on $\mathbf{w}_i^{(t)}$ the addition of the new weak learner $h^{(t)}$ to $f^{(T)}$. The sum of the weights is thus equal to the exponential loss $R_e(f^{(T)})$. Defining the *edge* $\gamma^{(t)}$ of the weak learner $h^{(t)}$ as

$$\gamma^{(t)} = \frac{\sum_{i=1}^{n} \mathbf{w}_i^{(t)} \mathbf{y}_i h^{(t)}(\mathbf{x}_i)}{\sum_{i=1}^{n} \mathbf{w}_i^{(t)}} = 1 - 2\epsilon^{(t)}$$

it is easy to show that the exponential loss after any number of steps T is

$$R_e(f^{(T)}) = n \prod_{t=1}^{T} \sqrt{1 - \gamma^{(t)^2}} \tag{1}$$

Thus by minimizing the weighted error $\epsilon_{(t)}$ (or equivalently by maximizing the *edge* $\gamma^{(t)}$) the selected weak learner $h^{(t)}$ greedily maximize the loss reduction. One can also remark that if $\exists \rho \in \mathbb{R} > 0, s.t. \forall t, \gamma^{(t)} \geq \rho$, then the loss will decrease exponentially fast.

The main idea of the paper is to partition the weak learner space $\mathcal{H}$ into (not necessarily disjunct) subsets $\mathcal{G} = \{\mathcal{H}_1, ..., \mathcal{H}_M\}$, and to use a multi-armed bandits algorithm by Auer et al. called *Exp3.P* [6] to learn the usefulness of the subsets. Each arm represents a subset, so at each step, the algorithm selects a subset so that *AdaBoost* then only needs to find the optimal weak learner $h^{(t)}$ in that subset instead of the whole space $\mathcal{H}$. The authors call the combined version of *AdaBoost* and the *Exp3.P* algorithm, *AdaBoost.MH.Exp3.P*[4]. The authors use $r^{(t)} = \min\left(1, -\log\sqrt{1 - \gamma^{(t)^2}}\right)$ as a reward. Taking the minimum with 1 is necessary as otherwise it would be unbounded, since $\gamma^{(t)} \in \,]0, 1]$. This reward makes sense as *AdaBoost* minimizes the exponential margin loss, while the *Exp3.P* algorithm try to maximize the sum of the rewards. Setting the reward as they did thus makes the two equivalent (minus the capping by 1, which is rarely a problem in practice as it happens only when the *edge* gets close to 1) according to equation 1.

The authors then go on proving a weak-to-strong learning theorem for their *AdaBoost.MH.Exp3.P* algorithm. The proof of that theorem rely on an upper bound on the weak regret of the *Exp3.P* algorithm (Theorem 6.3 of [6]). The weak regret is defined as follows

$$\max_j \sum_{t=1}^{T} \mathbf{r}_j^{(t)} - \sum_{t=1}^{T} \mathbf{r}^{(t)}$$

where $\mathbf{r}_j^{(t)}$ is the reward of arm j at step t. The weak regret thus compare the reward obtained by the gambler to the best fixed arm retrospectively. As the proof rely on the weak regret, the authors require stronger assumptions on the *edge* of weak learners than *AdaBoost*. As the weak regret compares the obtained reward to the one of the best arm, the proof requires the assumption that there exists a subset of $\mathcal{G}$, denoted by $\mathcal{H}_\dagger$, and a constant $\rho_\dagger \in \mathbb{R} > 0$ such that under any distribution of the weights $\mathbf{w}$, there exists a weak learner in $\mathcal{H}_\dagger$ with an *edge* greater or equal to $\rho_\dagger$. If the assumption holds, then the upper bound on the weak regret of the *Exp3.P* algorithm can be used to upper bound the number of steps T necessary for their algorithm to reach zero training error, and that number is logarithmic with n, thus proving that *AdaBoost.MH.Exp3.P* is indeed a Boosting algorithm.

In their experiments, the authors use *stumps* as weak learner (as well as two other weak learner: decision trees and product of stumps, but model them as respectively a recursive or an iterative sequence of *stumps*). Since a *stump* is simply a threshold on a particular feature, the authors partition the space of weak learners by assigning a subset to every feature (partitioning further would not decrease the computational time). The authors first carried out two synthetic experiments, where they could control the number of useful features. Their algorithm scores halfway between full (standard) *AdaBoost* and random sampling of the features (*LazyBoost*) in the first experiment and very close to full *AdaBoost* in the second. The authors then proceed to benchmark their algorithm on five well known datasets (MNIST, USPS, UCI pendigit, UCI isolet, UCI letter). It is hard to draw any conclusion from the test errors on those datasets, *AdaBoost.MH.Exp3.P* and *LazyBoost* seem a bit better than plain *AdaBoost* when using stumps (maybe through the regularization effect of randomization), and a bit worse using decision trees or product of stumps (maybe because of the assumption than trees and products can be modeled as a sequence of stumps). For sure the results of *AdaBoost.MH.Exp3.P* are not significantly different from those of *LazyBoost*. The criteria on which *AdaBoost.MH.Exp3.P* really makes a difference is computational efficiency, as it require one to two orders of magnitude less time than standard *AdaBoost* to reach a similar level of accuracy, provided than that level is not too high. It also seems to clearly outperform *LazyBoost* on that criteria, although since the authors does not

[3]Despite its name, *LazyBoost* cannot really be called a Boosting algorithm, as no proof can be made on it's convergence speed.

[4]*MH* as they use a multiclass version of *AdaBoost*, *AdaBoost.MH*. Despite that difference, the following explanations remain valid as *AdaBoost* is very similar to its generalized variant.

provide any information on the size of the subset of random features selected at each step, no clear conclusion can be given on that point.

In conclusion the authors present a new Boosting algorithm, called *AdaBoost.MH.Exp3.P*, that uses a multi-armed bandits algorithm to accelerate *AdaBoost*. The algorithm seem able to compete in accuracy with *AdaBoost*, while taking one or two orders of magnitude less time. A few remarks must still be made about the paper, to prove a weak-to-strong-learning bound, the authors had to make an additional assumption over standard *AdaBoost* ($\exists \mathcal{H}_\dagger \subset \mathcal{H}, \exists \rho_\dagger > 0 \ s.t. \ \forall \mathbf{w}, \exists h_\dagger \in \mathcal{H}_\dagger \ s.t. \ \gamma_{h_\dagger} \geq \rho_\dagger$). It is also unclear how to partition the space of weak learner other than stumps, and how the algorithm really compare against *LazyBoost*, as the authors did not detail the parameters that they used.

All in all, as the algorithm is very fast it might be very useful to the *MASH* project. A very simple partitioning of the features would be to have one subset per feature family. The bandits algorithm would then directly learn the usefulness of the families. It would then make a very sensible baseline for future algorithms to compare with.

D. Feature Selection using Multiple Streams

This paper by P. Dhillon, D. Foster, and L. Ungar [9] extends a previous paper ("Streamwise Feature Selection" [10]) by J. Zhou and the two last authors of the current paper to handle multiple family of features.

Streamwise feature selection is a very efficient and successful technique to do feature selection. Contrarily to most other common methods, such as *stepwise* feature selection or regularization, *streamwise* feature selection considers that the feature come from a stream, and can thus handle dynamically generated or even infinite feature sets, while most other methods are batch methods, and require the entire feature set to be known in advance. Also due to its online nature, *streamwise* feature selection only need to look at every feature once, making it the method of choice when the feature set is very large.

Feature selection algorithm in general and *streamwise* feature selection in particular assume that all feature come from a single equivalence class. But often the feature space has some structure, which if taken into account might help the feature selection process as generally good features are spread unevenly across classes. Even when this is not the case, it is always possible to create new classes of features by using projections such as principal components analysis (*PCA*), transformation such as log or square root, interactions (products of features), or clustering in the feature space.

Streamwise feature selection is a greedy algorithm which works by examining features one by one, selecting it if it significantly improve the accuracy of the model (details below) and otherwise discarding it for good. Because it enables one to decide which feature to test at every moment, it is particularly well suited to handle multiple feature classes, by examining first features from class which have produced more beneficial features in the past (the original paper [10], section 3 actually already contains this idea, but the current paper examines it in much more details).

Input: k $streams$, $\mathbf{y}$, w_0, α_Δ
1 **for** $j \leftarrow 1$ **to** k **do**
2 $\quad w_j \leftarrow \frac{w_0}{k}$
3 $\quad i_j \leftarrow 1$
4 **end**
5 $model = \emptyset$
6 **while** $\exists stream \neq \emptyset$ **do**
7 $\quad j \leftarrow \text{argmax}_j \frac{w_j}{i_j}$
8 $\quad x \leftarrow stream_j$
9 $\quad \alpha \leftarrow \frac{w_j}{2i_j}$
10 $\quad$ **if** *p-value*$(x, \mathbf{y}, model) \leq \alpha$ **then**
11 $\quad\quad model \leftarrow model \cup x$
12 $\quad\quad w_j \leftarrow w_j + \alpha_\Delta - \alpha$
13 $\quad$ **else**
14 $\quad\quad w_j \leftarrow w_j - \alpha$
15 $\quad$ **end**
16 $\quad i_j \leftarrow i_j + 1$
17 **end**
Output: $model$

Fig. 4. The pseudocode of the multiple *streamwise* feature selection (MSFS) algorithm. The algorithm takes as input k streams of features x, a vector of target values $\mathbf{y}$, the initial wealth w_0, and α_Δ is a parameter controlling with w_0 the false discovery rate. One can remark that the algorithm falls back to the original α-investing algorithm if $k = 1$.

Streamwise feature selection considers feature sequentially for addition to a linear model

$$y = \beta_0 + \sum_j \beta_j x_j$$

by comparing the reduction in training error against an adaptively adjusted threshold. Their is two flavors of *streamwise* feature selection, depending on how the threshold is computed. The alpha-investing variant, as used in the current paper, and the information-investing variant. According to the original paper [10], both yield virtually identical results. The original alpha-investing algorithm is given in Fig. 4, provided that $k = 1$. The threshold α computed line 9 of the algorithm, corresponds to the allowed probability of adding a spurious feature to the model (one which would decrease the accuracy on a hypothetical test set). It is adjusted using the wealth w, representing the currently acceptable number of future false positive. Wealth is increased when a feature is added to the model (line 11 and 12), as the feature is assumed correct, and therefore allow more future false positives without deteriorating the overall false discovery rate. If the feature is deemed spurious, wealth is decreased (line 14) in order to keep the guarantee of not adding more than a target fraction of spurious feature. The authors also give a formula to compute the p-value of x (that is, the probability that the coefficient of x would be judged to be non-zero when it is in reality zero) in the case of regression:

$$\text{p-value} = e^{-\frac{\text{RMSE}_{(model \cup x)} - \text{RMSE}_{(model)}}{\sigma^2}}$$

where RMSE means the root mean squared error of the model on the training set, and the variance σ^2 is estimated as $\frac{\text{RMSE}_{(model)}}{n}$ where n is the number of training samples.

Streamwise feature selection provide guarantee bounds on the ratio of the number of expected spurious feature included in the model $E(N)$ and $E(M)$, the expected number of beneficial features

$$E(N) < \frac{\alpha_\Delta E(M) + w_0}{1 - \alpha_\Delta}$$

In all their experiment the authors use as parameter $\alpha_\Delta = w_0 = \frac{1}{2}$, thus approximately guaranteeing $E(N) < E(M)$.

The authors modified the original *streamwise* feature selection algorithm to handle multiple feature classes by giving to each class its own stream, splitting uniformly the initial wealth among the streams, and keeping track of the individual wealth and of how many features from that stream have already been tested. As the wealth represents how successful the stream has been in producing useful features so far, the problem of which stream to select at every step is easily solved, the one with the highest probability of yielding a useful feature is always selected.

Giving each feature class its own stream is advantageous when the distribution of good features is not uniform across feature classes. In that case the streams of those classes with more good features will get higher wealth, and will thus be favored by the algorithm. In the extreme case where a single class contains all the good feature, the algorithm will quickly favor its stream and very few features from the other streams will need to be considered.

The authors then proceed to benchmark their multiple *streamwise* feature selection (MSFS) algorithm against multiple other feature selection schemes, including but not limited to: standard *streamwise* feature selection (SFS), *stepwise* feature selection, Lasso, or multiple kernel learning (MKL), on various datasets, adding feature classes when not already present (PCA and squared). MSFS obtain very competitive results, even the best when also including interaction terms (product of already selected features with themselves and other features). It is also the second fastest method, beaten only by regular *streamwise* feature selection, but that it seems to outperform in term of test error, although it depends of course on the structure of the particular dataset at hand.

The MSFS algorithm similarly to the one of the previous section is also very fast and could also be added to the *MASH* project as a baseline or in conjunction with another classification technique (although the simple regression model should probably be adapted), the class of the features being already given. The ability to dynamically add features might also prove useful to find new beneficial features as interactions in-between families. The MSFS algorithm being a simple extension to multiple feature classes of an already popular technique, there is nothing to reproach to it without formulating the same reproach to SFS.

III. Learning in Very High Dimensional Feature Space

This concluding section describes the current state of our researches, as well as a plan of a possible course of actions. It gives references to current algorithms in scene interpretation, or more generally, object detection and classification, and describes how we intend to build upon and advance the state-of-the-art.

A. State of the research in the field

Many real-world problems have already been solved thanks to statistical learning (e.g. face detection [8], robot control [11], handwritten character recognition [12], etc.). The deep learning architecture [13] even taking that idea to an extreme by trying to build from the ground up higher and higher levels of abstraction directly from the raw data. Despite these efforts, performance remains far from human-level.

The need for complex priors to encode the expert knowledge and reduce the requirement of a large training set has appeared in many application fields for which data is scarce. In such cases, ad-hoc prior knowledge has to be encoded, either by putting prior distributions on the parameters, constraining the solution space of the learning algorithm, or explicitly decomposing the problem into simpler ones as in the DARPA grand challenge [14]. But very few research projects today study models with very complex priors.

Detection and recognition of a large class of objects is a hot topic now in computer vision. Several databases and challenges are publicly available (Caltech-256, PASCAL VOC, etc.) and relatively good results have been attained for image categorization (e.g. [1]).

B. State of our researches

Eight months after the beginning of the project, the initial development phase of *MASH* is now drawing to an end, and the project is already ready to accept external contributions through its website (http://www.mash-project.eu). The core of the *MASH* framework is now running on dedicated servers and can run classification (detection should follow soon) experiments through the web. A few baseline classifiers have been developed (AdaBoost, SVM, k-NN, etc.), as well as a few initial heuristics (HoG, Haar and Fourier transforms, color histograms, etc.). Extensive tests particularly on the SVRT database (more details below) have also been carried out.
The list of classifiers currently running:

- AdaBoost: as presented here, can use any classifier of the framework as weak learner, but usually stumps or trees
- LogitBoost: variant of AdaBoost using the logistic (rather than exponential) loss function
- LPBoost: Boosting algorithm directly maximizing a margin between training samples of different classes
- C4.5: decision tree algorithm developed by Dr. R. Quinlan, successor of ID3, using a greedy entropy heuristic in order to find splits and a pruning strategies based on confidence limits
- Perceptron: batch and stochastic version of the classical Perceptron algorithm (as well as a kernelized version)
- SVM: Support Vector Machine classifier, constructs an hyperplane in a high or infinite dimensional space separating the two classes as much as possible (maximizing the margin and thus reducing the generalization error)

- linear SVM: linear Support Vector Machine classifier, much faster than the standard SVM when using the linear kernel
- k-NN: Nearest Neighbor classifier, classifies a given sample according to the mode of the labels of its k nearest neighbor (in the L-k norm sense)
- NB: Naive Bayes' classifier, assumes that the features are independent and can be well approximated by gaussians
- One-vs-One and One-vs-All: combine binary classifiers (such as a SVM) in one versus one or one versus all schemes to tackle multiclass problems

As well as the list of different filters that can be applied on the data prior to training:

- Min-max: normalizes all the features to lie in the range [0, 1]
- Statistical: normalizes all the features to zero mean, unit standard deviation
- Energy: normalizes all the samples to unit norm (in the L-k norm sense)
- PCA: project the data on its principal components

The list of databases currently set up:

- COIL-100: Columbia University Image Library, a (very) simple database of objects centered on a dark background
- Caltech-256: collection of all 30607 images of everyday objects distributed in 256 categories
- MNIST: subset of the NIST dataset, a vast database of handwritten digits (http://yann.lecun.com/exdb/mnist/)
- SVRT: Synthetic Visual Reasoning Test, a group of 23 abstract visual problems, which goal is to compare the performance of humans and ML algorithms (http://www.idiap.ch/~fleuret/svrt/challenge.html)

The framework was already used to provide the baseline of the SVRT project. *AdaBoost* (with stumps) and a Gaussian SVM were trained and tested on every of the 23 problems using different combinations of heuristics. Preliminary analyses reveal that if the SVM is superior when using simple heuristics (e.g. the raw pixel data), the opposite becomes true when using complex heuristics (e.g. the Fourier transform), the SVM performance remaining approximately the same while the one of *AdaBoost* can increase a lot, most often becoming superior.

C. Research plan

One of the main issue is how to handle learning in such a large (up to millions of features) and heterogeneous (hundreds of different heuristics) feature space. Problems that have to be addressed are

- overfitting: the number of features is likely to be much larger than the number of samples
- computational cost: as the computing resources are limited, the computational cost of every heuristic should be taken into account in order to stay on (time) budget
- equity: if the heuristics were always evaluated in the same order, heuristics tested first would be unjustly favored, also if an heuristic is marginally better than another one, it might be always selected, thus some kind of principled randomization must be adopted

Creating sensible heuristics for the problem at hand relies heavily on the capacity to identify the weaknesses of the current system. Hence, an important task of the project is to find how to provide useful feedbacks to a contributor about a heuristic, its performance, how it compares with the other heuristics, areas where it excels and areas requiring improvements, etc. Another important feedback to provide to the contributors is an analysis of the overall performance of the current system, for example its worst mistakes.

A possible time line could be

First year (already done):

- Review of the literature about machine learning in general and scene interpretation/object detection/image classification in particular
- Development of some baseline general purpose learning algorithms
- Set up of some standard datasets and evaluation of the baselines

Second year:

- Development of more sensible baselines tailored to learning with a large number of features, such as the algorithms discussed in this proposal, as well as other state-of-the-art algorithms such as MKL [1]
- Study of methods to speed up the learning (e.g. taking the computational cost of an heuristic into account while doing feature selection, or Dynamic Heuristic Sampling (see §D below))
- Study the fairness on the selection and evaluation of features

Third and fourth years:

- Online computation ordering (reorder the evaluation of weak-learners in order to speed-up testing)
- Help with the design of new heuristics, possible ideas include: clustering mistakes, i.e. how to summarize the areas requiring improvement to a designer; clustering heuristics, find way to explore the heuristic space.
- Thesis writing.

D. Dynamic Heuristic Sampling

Dynamic Heuristic Sampling is an approach that we are currently investigating. In the context of Boosting with decision stumps, at each step we have a weight distribution on the samples and we need to select an optimal feature. We would like that feature to maximally decrease the current Boosting loss. The idea is to have a way to compute the distribution of the loss reductions associated to the features of every heuristic depending on the current sample weights. This could be achieved by storing the responses of a limited number of features (for example 100) on all the samples from every heuristic at the start, and then compute the loss for each at every step. We could then sample features from the heuristic with the highest expected loss reduction. We expect the algorithm to have the following behavior, to first exhaust the most informative heuristic before switching to another one (possibly already exploited). One core issue to address is to prioritize the estimation of the expected loss reduction, maybe with bandit-style strategies. Some optimizations also comes

to mind; it is probably unnecessary to re-estimate the loss reduction of heuristics previously judged bad if the weights did not change much (as happen from a step to the next). Also if we have a fixed time budget at each step, the computational cost of the heuristics can also be taken into account, e.g. do we expect a higher loss reduction from trying 1000 features from that fast heuristic, or 10 from that slow one?

REFERENCES

[1] P. Gehler and S. Nowozin, "On Feature Combination for Multiclass Object Classification", *Proceedings of the Twelfth IEEE International Conference on Computer Vision (ICCV 2009)*, 2009.

[2] Y. Freund and R. Schapire, "A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting", *Lecture Notes In Computer Science; Vol. 904*,1995.

[3] A. Opelt, A. Pinz, M. Fussenegger, and P. Auer, "Generic Object Recognition with Boosting", *Pattern Analysis and Machine Intelligence (PAMI), IEEE Transactions on*, 28:416-431, 2006.

[4] R. Busa-Fekete and B. Kégl, "Fast boosting using adversarial bandits", *Proceedings of International Conference on Machine Learning (ICML)*, 2010.

[5] R. Busa-Fekete and B. Kégl, "Accelerating AdaBoost using UCB", *JMLR W&CP 7:111-122*, 2009.

[6] P. Auer, N. Cesa-Bianchi, N. Freund, and Y. Schapire, "The non-stochastic multi-armed bandit problem", *SIAM J. on Computing*, 32(1):48-77, 2002.

[7] G. Escudero, L. Màrquez, and G. Rigau, "Boosting Applied to Word Sense Disambiguation", European Conference on Machine Learning, 2000.

[8] P. Viola and M. Jones, "Robust real-time face detection", *International Journal of Computer Vision, 57:137-154*, 2004.

[9] P. Dhillon, D. Foster, and L. Ungar, "Feature Selection using Multiple Streams", *International Conference on Artificial Intelligence and Statistics*, 2010.

[10] J. Zhou, D. Foster, and L. Ungar, "Streamwise Feature Selection", *Journal of Machine Learning Research 7: 1861-1885*, 2006.

[11] Y. LeCun, U. Muller, J. Ben, E. Cosatoo, and B. Flepp, "Off-road obstacle avoidance through end-to-end learning", *Neural Information Processing Systems, Vol. 17*, 2005.

[12] R. Plamondon and S. Srihari, "On-line and off-line handwriting recognition: A comprehensive survey", *IEEE Transactions on Pattern Analysis and Machine Intelligence 22(1), 6384*, 2000.

[13] Y. Bengio and Y. LeCun, "Scaling Learning Algorithms towards AI", *Large-Scale Kernel Machines, MIT Press*, 2007.

[14] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, j. Diebel, P. Fong, J. Gale, H. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney, "Winning the darpa grand challenge", *Journal of Field Robotics*, 2006.