Fast Weak Learner Based on Genetic Algorithm*

Boris Yangel
Department of Computational Mathematics and Cybernetics
Lomonosov Moscow State University, Moscow, Russia
hr0nix@acm.org

Abstract

An approach to the acceleration of parametric weak classifier boosting is proposed. Weak classifier is called parametric if it has fixed number of parameters and, therefore, can be represented as a point in multidimensional space. Genetic algorithm is used to learn parameters of such classifier. Proposed approach also takes cases when effective algorithm for learning some of the classifier parameters exists into account. Experiments confirm that such an approach can dramatically decrease classifier training time while keeping both training and test errors small, at least for some widely used pattern recognition algorithms.

Keywords: *boosting, genetic algorithm, classification, haar feature.*

1. INTRODUCTION

Boosting [1] is one of the commonly used classifier learning approaches. It is machine learning meta-algorithm that iteratively learns additive model consisting of weighed *weak* classifiers that belong to some classifier family W. In case of two-class classification problem (which we will consider in this paper) boosted classifier usually has form

$$s(x) = \operatorname{sgn}\left(\sum_{i=1}^{N} \alpha_i w_i(x)\right). \quad (1)$$

There $x \in X$ is a sample to classify, $w_i \in W$ are weak classifiers learned during boosting procedure, α_i are weak classifier weights, $w_i(x) \in \{-1, 1\}$, $s(x) \in \{-1, 1\}$. Set W is referred to as *weak* classifier family. That is because it elements should have error rate only slightly better than random guessing. It expresses the key idea of boosting: strong classifier can be built on top of many weak.

There are many boosting procedures that differ in the type of loss being optimized for the final classifier. But no matter what kind of boosting procedure is used, on each iteration it should select (learn) a weak classifier with minimal weighed loss from W family using special algorithm called *weak learner*. For some widely used families of weak classifiers that process can take a lot of time. For example, simple classifiers called *stumps* are often used in boosting. Those classifiers usually have form

$$w_i(x) = \operatorname{sgn}[g_i(\phi_i(x) - t_i)], \quad (2)$$

where $\phi_i \in \Phi$ is some object feature, t_i is a threshold and $g_i \in \{-1, 1\}$ controls the sign of the weak classifier output. When given some feature ϕ_i, optimal values for classifier parameters t_i and g_i can be calculated rather fast for most of the widely used loss functions. Problem of finding good feature ϕ_i is much harder. Exhaustive search over feature space Φ is often used. It's not a problem when object has few features, but sometimes feature space can be really huge.

*This work was supported by the Russian Fund for Fundamental Research through grant Nos. 08-07-445-a and 08-07-12081.

In pattern recognition, values of some function over various subregions of an image are often considered as the features of that image. Even small image has a lot of possible subregions, so exhaustive search over feature space becomes very expensive. For example, learning cascade of boosted stump classifiers based on haar features with *AdaBoost* and exhaustive search over feature space took several weeks in the famous work [2]. That's why it is often very important to decrease weak classifier learning time using some appropriate optimization technique.

One of the widely used approaches to the numerical optimization is genetic algorithm [3]. It is based on the biological evolution ideas. Optimization problem solution is coded as *chromosome* vector. *Initial population* of solutions is created using random number generator. *Fitness function* is then used to assign fitness value to every population member. Solutions with the largest fitness values are selected for the next step. In the next step, *genetic operators* (crossover and mutation usually) are applied to selected chromosomes to produce new solutions and to modify existing ones slightly. Those modified solutions form up a new generation. Then described process repeats. That's how evolution is modeled. It continues until global or suboptimal solution is found or time allowed for evolution is over. Genetic algorithms are often used for global extremum search in big and complicated search spaces.

2. RELATED WORK

Usage of genetic algorithm for weak learner acceleration was already proposed in several works. For example, in [4] genetic weak learner with special crossover and mutation operators was used to learn classifier based on extended haar feature set. In [5] genetic algorithm was used to select a few thousand weak classifiers with smallest error on unweighted training set before boosting process starts. Then exhaustive search over selected classifiers was performed on each boosting iteration to select the one with minimal weighed loss. In [6] boosting procedure was completely integrated with genetic algorithm. Few classifiers were selected on each boosting iteration from solution population and added to the strong classifier. Those selected classifiers were then used to produce new population members by applying genetic operators. Then, in [7] authors used special evolutionary algorithm they've called *Evolutionary Hill-Climbing* as a weak learner. Crossover operator was not used in it. Instead, 5 different mutations were applied to every population member on each algorithm iteration. Result of each mutation was rejected when it did not improve fitness function value.

There were two main reasons for using genetic search instead of any other approaches in these works. Most of the classifiers used in mentioned works were some extensions of the haar classifier family originally proposed in [2]. So, huge size of the feature space and, therefore, huge size of the weak classifier family did not allow to apply exhaustive search based optimization. And complicated discrete structure of a weak classifier blocked all other optimization options.

Another important observation is the fact that all the authors of the mentioned papers were forced to implement some specialized solu-

tion for genetic weak learner. So, ability to generalize evolutionary approach to learning weak classifier is investigated in this work.

3. PROPOSED METHOD

We are interested in developing some general approach to weak classifier learning. And we are especially interested in approach that will handle classifiers of form (2) because those classifiers are widely used in pattern recognition area. This approach should work much faster than exhaustive search over classifier parameter space. In the following document sections one such approach is presented. It is based on the fact that when number of classifier parameters to optimize is fixed, weighed loss optimization problem simply turns out into multivariate function minimization problem which is well-developed area of genetic algorithm application.

3.1 Population member

Let W be some parametric family of weak classifiers. It means that every weak $w \in W$ can be described by a set of its n real-valued parameters $x_1, \ldots, x_n$. Let's also assume that for the last l parameters (l can be equal to zero) there exists some effective learning algorithm $L_E : \mathbb{R}^{n-l} \to \mathbb{R}^l$. We will refer to such parameters as to *linked*. For given values of parameters $x_1, \ldots, x_{n-l}$, called *free*, L_E finds optimal values for linked parameters that minimize loss function $E : \mathbb{R}^n \to \mathbb{R}^+$. It means that our task is to find values of free parameters that deliver the minimum to the loss function $E[x_1, \ldots, x_{n-l}, L_E(x_1, \ldots, x_{n-l})]$. So, set of parameters $x_1, \ldots, x_{n-l}$ represents solution to our optimization problem and form up a member of genetic algorithm population.

For example, for the weak classifier of form (2) parameters t_i and g_i will be linked, while parameters describing feature ϕ_i will be made free. It should be mentioned that if small changes of feature representation can have unpredictable impact on the value of the loss function, genetic optimization turns into random search. It happens, for example, when feature can be simply described by its index in the feature pool and features with close indices are not correlated. On the other hand, small changes in the image region size or position usually lead to small changes in the region characteristic, at least for some "good" stable characteristic functions like pixel intensity sum in region. For such feature sets genetic optimization is possible.

3.2 Fitness function

It is natural to assume that classifier with small error on training set should have greater probability to get to the next generation of the genetic algorithm. That allows us to introduce fitness function $F : \mathbb{R}^{n-l} \to \mathbb{R}^+$ as follows:

$$F(x_1, \ldots, x_{n-l}) = \\ = 1/E[x_1, \ldots, x_{n-l}, L_E(x_1, \ldots, x_{n-l})]. \quad (3)$$

We do not consider $E = 0$ case. Classifier can not be called weak if it has zero error value on training set. If such a classifier is presented in a weak classifier family, we can select only that classifier as a whole boosting procedure result.

3.3 Genetic representation

Every approach that allows us to encode a set of free parameters is appropriate for population member representation. In this work we have selected binary string representation which was confirmed to be effective in function optimization problems. Some alternative representations can be found, for example, in [3].

To form the binary string classifier representation, each classifier parameter should be first represented as a binary string of fixed length, using fixed-precision encoding. Then all the parameters can be simply concatenated to form the final binary string of fixed length.

Sometimes point $p \in \mathbb{R}^n$ can have no corresponding classifier. For the different families of image region classifiers it is possible, for example, when one of the free parameters representing top-left corner of a classifier window is below zero. In this case fitness function value of the population member representing that point can be forced to be zero. That is how such situations were dealt with in experiments described in section 4.. Another possible approach is to select representation and genetic operators in a way that simply does not allow such points to appear. But that approach is less general.

3.4 Genetic operators

In this work we've used the two most common genetic operators: mutation and crossover. When binary string representation is chosen, mutation and crossover are usually defined as follows:

- Crossover operator selects random position in the binary string. Then it swaps all the bits to the right of the selected position between two chromosomes. Such crossover implementation is called 1-point crossover.
- Mutation operator changes value of the random chromosome bit to the opposite.

In our case, crossover operator produces two new solutions from the two given chromosomes as following: some of the parameters (placed to the left of the selected position) are taken from the first classifier, some of the parameters (placed to the right) — from the second. And one parameter, probably, can be made from both the the first and the second classifier. Mutation operator simply produces new solution by changing value of the random classifier parameter.

3.5 Algorithm summary

Algorithm 1 Genetic weak learner

1: Generate initial population of N random binary strings;
2: **for** $i = 1, \ldots, K_{max}$ **do**
3: Add $\lceil NR_c \rceil$ members to the population by applying crossover operator to the pairs of the random population members;
4: Add $\lceil NR_m \rceil$ members to the population by applying mutation operator to the random population members;
5: Calculate value of (3) for each population member;
6: Remove all the population members except of the N best (those with largest value of (3));
7: **end for**
8: **return** weak classifier associated with point represented by best population member as a result;

Algorithm 1 uses elitism as a population member selection approach. It has 4 parameters:

- $N > 0$ — population size.
- $K_{max} > 0$ — number of generations.
- $R_c \in (0, 1]$ — crossover rate.
- $R_m \in (0, 1]$ — mutation rate.

3.6 Discussion

Advantage of the proposed method lies in the fact that computational complexity of the weak learner does not depend on the size of the weak classifier family. One can achieve balance between training time and classifier performance only by changing values of N, K_{max} and S (discussed later). Similar effect can be achieved by shrinking weak classifier family itself. But in most cases prior knowledge about weak classifier performance in boosting is simply not available.

One of the main disadvantages of the proposed weak learner is the fact that many potentially interesting weak classifiers can not be represented as a parameter vector of constant length. For example, decision trees, widely used in boosting, can have variable number of nodes. Also, as it was already mentioned, misclassification loss we are trying to optimize should be more or less stable as a function of classifier free parameters. If small perturbations of the free parameter vector lead to the unpredictable changes in the loss function value, genetic optimization does not make much sense, becoming just a random search.

4. EXPERIMENTS

4.1 Algorithms for experiments

Two boosting-based algorithms were implemented to compare proposed genetic weak learner with original learners proposed by algorithm authors. *Viola-Jones* [2] and *Face alignment via boosted ranking model* [8] were selected for that purpose because both algorithms use parametric weak classifiers applied to image regions. These algorithms are based on distinct boosting procedures (*AdaBoost* and *GentleBoost*), so loss, sample weight and classifier weight functions used in them differ a lot. Another difference between selected algorithms is a problem they solve: two-class classification in [2] and ranking in [8]. Naïve weak learner implementation is quite slow in both algorithms, so acceleration of boosting process is necessary.

Weak classifiers used in both algorithms are based on haar features and have common set of adjustable parameters. So, weak classifier in both problems can be represented as $w_i = (x_i, y_i, width_i, height_i, type_i, g_i, t_i)$. There x_i, y_i, $width_i$ and $height_i$ describe image region, $type_i$ encodes haar feature type, g_i is a haar feature sign and t_i represents weak classifier threshold. Parameters g_i and t_i are linked because both algorithms have an effective algorithm for learning them. Parameter $type_i$ was also made linked: changing feature type during genetic optimization does not make much sense because it can change fitness function value significantly after just one mutation or crossover. Separate algorithm run was performed instead for each feature type. Best result from all the runs was then selected. We've used the same 5 haar feature types as in [8] for training both classifiers.

4.2 Run patterns

Comparison of two different genetic algorithm run patterns was also performed in this work. One pattern considered was running genetic optimization once with big population size. Another pattern used was running optimization algorithm multiple times (denoted as S) with small population size and then selecting best found classifier. When population size is small, final solution depends on initial population a lot. So, considerably different results can be obtained for different algorithm runs. While this run pattern produces worse classifiers, it can be implemented on multiprocessor and multicore architectures very efficiently: each processing unit can run its own genetic simulation. That makes perfect parallel algorithm acceleration possible.

Table 1: Viola-Jones, acceleration

Run pattern			Time (sec)	Acceleration
S	N	K_{max}		
1	50	10	**2.82**	**329.38**
1	100	20	9.40	98.77
1	400	40	100.29	9.26
10	10	20	4.00	231.94
20	20	40	28.74	32.31
Brute force			928.52	1.00

Table 2: Viola-Jones, error

Run pattern			Error	
S	N	K_{max}	Learning	Test
1	50	10	0.0005	0.0356
1	100	20	0.0002	0.0380
1	400	40	0.0000	**0.0328**
10	10	20	0.0003	0.0378
20	20	40	0.0000	0.0391
Brute force			0.0000	0.0349

4.3 Training and test sets

As in work [4], [9] human faces database was used to train and test classifier for Viola-Jones algorithm. Database was divided in half to form the training and test sets. Each sample has size of 24×24 pixels.

Face images with landmarks from FG-NET aging database were used to form the database for learning face alignment ranker proposed in [8]. 600 face images were selected from database and then resized to the size of 40×40 pixels. 400 images were used to produce training set and other 200 — for testing. 10 sequential 6-step random landmark position perturbations were then applied to selected face images to produce images of misaligned faces, as described in the original paper. Training and test set samples were then made of pairs of images with increasing alignment quality.

4.4 Hardware

All the experiments were performed on PC equipped with 2.33 GHz Intel Core 2 Quad processor and 2 GB of DDR2 RAM.

4.5 Results

Tables 1 and 3 show average duration of 1 boosting iteration together with comparison to exhaustive search. Tables 2 and 4 show error rate of the final classifiers on the training and test sets. We have not trained any classifier using exhaustive search for boosted ranking model because it would take about a year to finish the process on our training set.

Experiments with Viola-Jones object detector have shown that classifier trained using genetic weak learner performs only slightly worse than classifier trained using exhaustive search over feature space. For $N = 400$ final classifier even shows better performance. Classifier trained with $S = 1$, $N = 50$ and $K_{max} = 10$ accelerates boosting nearly 300 times compared to exhaustive search while still performing good on the test set. Classifiers trained with small values of N and big values of S (using second run pattern) perform worse than any other. But, as it was mentioned before, such classifiers can be trained on multiprocessor or multicore systems very efficiently.

Experiments with face alignment via boosted ranking model have shown how exactly classifier performance depends on values of S,

Table 3: Face alignment via BRM, acceleration

Run pattern			Time (sec)	Acceleration
S	N	K_{max}		
1	25	10	**68.15**	**5195.88**
1	50	10	173.33	2043.09
2	75	15	909.55	389.34
4	100	20	3582.37	98.85

Table 4: Face alignment via BRM, error

Run mode			Error	
S	N	K_{max}	Learning	Test
1	25	10	0.0278	0.0317
1	50	10	0.0246	0.0297
2	75	15	0.0199	0.0268
4	100	20	**0.0173**	**0.0259**

N and K_{max}. Increasing value of the each parameter results in increased training time, but also in increased classifier performance. Nevertheless, difference in training time is more significant compared to the difference in prediction error. Classifier with $S = 1$, $N = 25$ and $K_{max} = 10$ was trained 50 times faster than the best obtained classifier for BRM, but its error on the test set is only 1.2 times worse. It makes such a classifier a perfect candidate for preliminary experiments that usually take place before training of the final classifier starts.

5. CONCLUSION

An approach to boosting procedure acceleration was proposed in this work. Approach is based on usage of genetic weak learner for learning weak classifier of special parametric form on each boosting iteration. Genetic weak learner uses genetic algorithm with binary chromosomes. That genetic algorithm is designed to solve an optimization problem of selecting weak classifier with the smallest weighed loss from some parametric classifier family. Proposed method was generalized for the case when there exists an effective algorithm for learning some of the parameters of a weak classifier. Experiments have shown that such approach allows us to accelerate training process dramatically for practical tasks while keeping good generalization properties.

Genetic weak learner proposed in this work can't be used to boost any tree-based classifiers. That fact limits its usage in many scenarios because stump weak classifiers can not represent any deep interactions between different object features. So, in the future work we plan to generalize our approach for accelerating tree-based boosting.

Another option for future research is performing additional experiments with classifiers not related to haar features in any way. That will confirm the profit of the proposed algorithm in computer vision problems that are not biased towards haar feature usage. In fact, it would be nice to determine different parametric classifier families that can be efficiently boosted using proposed weak learner.

6. REFERENCES

[1] Robert E. Schapire, "The boosting approach to machine learning an overview," in *MSRI Workshop on Nonlinear Estimation and Classification*, 2002.

[2] Paul Viola and Michael Jones, "Robust real-time object detection," in *International Journal of Computer Vision*, 2001.

[3] David E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Professional, January 1989.

[4] Andre Treptow and Andreas Zell, "Combining adaboost learning and evolutionary search to select features for real-time object detection," 2004.

[5] Geovany A. Ramirez, "Face and street detection with asymmetric haar features," 2007.

[6] K. Masada, Qian Chen, Haiyuan Wu, and T. Wada, "Ga based feature generation for training cascade object detector," in *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, 2008, pp. 1–4.

[7] Y. Abramson, F. Moutarde, B. Stanciulescu, and B. Steux, "Combining adaboost with a hill-climbing evolutionary feature search for efficient training of performant visual object detectors," in *FLINS06*, March 2006.

[8] Hao Wu, Xiaoming Liu, and Gianfranco Doretto, "Face alignment via boosted ranking model," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, 2008, pp. 1–8.

[9] P. Carbonetto, "Viola-jones training data," `http://www.cs.ubc.ca/~pcarbo/viola-traindata.tar.gz`, 2002.