Machine Learning Course Project Report: Creating Cascade of Haar feature Classifiers with Adaboost Ensemble Learning Method for Low-Resolution Pattern Classification

Anastasia Bolotnikova
University of Tartu, Estonia
June 2016

***Abstract*—The focus of this project was to develop a classifier for recognizing a soccer ball pattern by NAO humanoid robot to be used when the robot is playing in RoboCup SPL competition. The necessity for creating such algorithm came from the fact that the RoboCup SPL introduced new rules an regulations, which state that instead of using small orange (easily detectable) ball the new realistic ball will be used in the competition games. Initially the attempt was made to detect and classify a circle shape in the frame by using edge information. However, even though detecting the ball shape was possible, the method was not robust to noise caused by other objects on the field, which may under some circumstances appear as round objects in the edge information frame. The shape fitting also turned out to be too slow for a real-time application. As a result, the decision has been made to use machine learning to solve the detection task. The need for a fast and accurate algorithm lead to the decision of training the cascade of Haar feature classifiers with Adaboost Ensemble method. This idea was initially inspired by the state-of-the-art face detection algorithm - Viola-Jones, which uses the same principle for human face detection. With Adaboost algorithm it was possible to create Haar-feature classifier cascades which achieve very high recognition rate with low classification error.**

Index terms: Adaboost, Haar features, Cascade Classifier, Pattern Classification, Ball Detection, RoboCup SPL

I. Introduction

20 year ago, the RoboCup organization started the initiative of organizing international competitions, where autonomous robots, programmed and built by teams from different universities all over the globe, perform various tasks, such as simple rescue tasks, home assisting tasks and playing soccer.

There are several differen leagues in the RoboCup soccer competition. In this report the focus is on the Standard Platform League (SPL) of RoboCup, where the platform is fixed for all competitors and only the programming skills are in the game. Currently the standard platform utilized in SPL games is NAO humanoid robot [1] (Figure 1).

Fig. 1: Robotic platform used in RoboCup SPL competitions - NAO humanoid robot.

Now the thing about NAO robots, they do not have such a good processor (Intel Atom @ 1.6 GHz). And another thing about them, besides the ball detection they have to run a whole bunch of other stuff in parallel (motion, localization, game logic decisions, goal detection etc.) at a rate 30 frames per second in order to fulfill the task given to them, which is autonomously playing soccer. As a consequence, all algorithms in the system have to be super-fast, I'd even say crazy-fast. Thus, the solution for the current task of detecting the ball must satisfy the "crazy-fast" criteria.

In the search for such an algorithm, the state-of-the-art methods have been studied in this project for a possible candidate of ball detection algorithm implementation. The first method which seemed promising is the well-known Viola-Jones face detection algorithm [2]. The confusion may occur at this point - "But the human face has nothing to do with the soccer ball..." (Figure 2). Although, the objects are indeed quite different, they still have a crucial thing in common - they are patterns. Patterns that can be defined by some numerical means. And it is possible to efficiently detect both of them using exactly the same principles with generalizable enough algorithm. The Viola-Jones happens to be such an algorithm and can actually be used to detect various different pattern, however some objects cannot be efficiently detected by Viola-

(a) Hunam face. (b) Soccer ball.

Fig. 2: Picture of human face on the left, soccer ball on the right. Both objects can be detected with specifically trained boosted ensembles of Haar feature classifiers.

Jones, and it seems that the problem occur when the patterns of the same object in training set are too different from one another. Another good thing of Viola-Jones - it is a real-time algorithm. It's not new at all, it's about 15 years old, but it does satisfy the "crazy-fast" criteria which is very important in our case.

The additional hope of the Viola-Jones like algorithm to be applicable to the task of soccer ball detection was the paper in [3]. This paper basically did the same thing (Haar-features, Adaboost, cascade) as I did in this work, but they had different robotic platform and higher resolution images.

So with all that in mind, the implementation of Adaboosting Haar-feature classifiers began. The rest of the paper is organized as follows: Section II describes the database creation and labeling madness, Section III describes the Haar feature classifier concept, Section V gives the details on the Adaboost implementation, Section VI gives overview of the cascade classifier architecture. Section VII presents the results and last section concludes the work.

II. Training and testing samples

One of the reasons, one may not be so excited about the machine learning solution to the problem, is the mess with the training set creation. Because in most of the cases, first, you need to collect the data (the more the better + the better the better), then, you need to properly manually label it and then if everything has been done well so far, you might have a good data for training the model. And if you mess up big at this point - no machine learning magic will save you later. This section describes how the data in our case was collected and labeled.

The frames in the NAO soccer software used by University of Tartu RoboCup team are represented at the low-level as foveas [5]. Those are 80x60 images (like Figure 3), which give low-resolution representation of the frame, if necessary one can zoom in to some parts of the fovea to see what is in there at the higher resolution (up to 8 time more than lowest fovea 80x60). The maximum resolution of NAO camera is 1280 by 960, but no one really has time to process all that, as I said before - algorithm has to be crazy-fast. So in this work we focus on the fovea classification, so we work with the 80x60 (for bottom camera) and 160x120 (for top camera) images. NAO has two cameras (Figure 4) and the detection algorithm first runs on the bottom one and then on the top one if no ball was found at the bottom. The top camera has 2 time higher maximum resolution than the bottom camera.

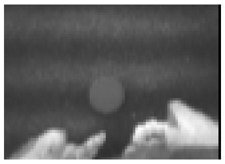


Fig. 3: 80x60 fovea is about this big. Robot is looking down at its feet and an old RoboCup SPL ball. "Hot spots" due to bad lighting can also be seen of this sample.

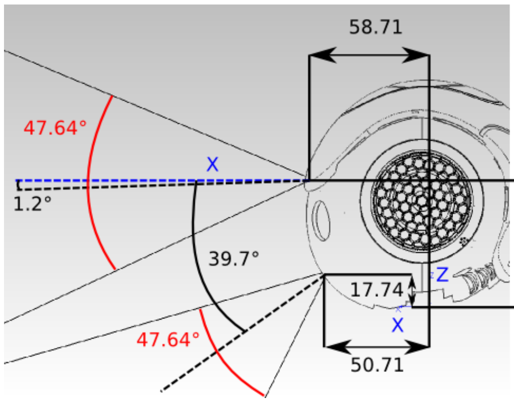


Fig. 4: NAO robot cameras.

Anyway, the first problem in data collection part of the project was the bad light. The typical lights in the laboratory caused so called "hot spots" in the foveas. Figure 5.a illustrates the effect of the "hot spots" - the light is not equal in all the rows of the fovea, some rows have more light while others don't. In the official RoboCup competitions, however, the light is typically equally distributed.

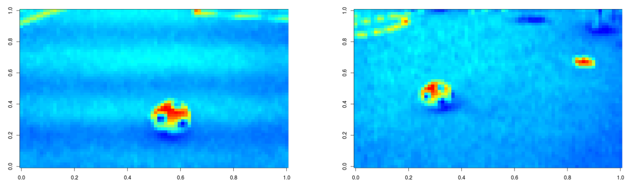

(a) Periodic horizontal hotspots (b) No hotspots

Fig. 5: Frames taken with different light conditions.

So first things first - we had to get bunch of projectors to make a light in the lab nice and equal. Figure 6 illustrates how happy were NAOs when everything was finally set up.

Fig. 6: NAO robots happy about the new light in the lab.

Now that the lights were good, several foveas were taken from NAO camera. After the foveas (both gray and edge information) were collected, the labeling process began. With the help of Python script it was possible to label all the samples relatively painlessly: it came down to clicking on bunch of foveas (around 600-700) in the top left and bottom right corner of the ball location. As a result of all this mess, we finally got the samples for training. Positive samples were the balls and negative were everything else in the fovea except the ball. Figures 7 and 8 shows samples from the training (and testing) set.

III. Haar feature and weak classifier

The Cascade of Viola-Jones classifiers consists of many weak classifiers, each one of them only has to be better than a random guess, error should be less than 0.5. The weak classifier is the Haar-feature, it's location in the subwindow, size of the Haar feature and the threshold which says what is ball and what is not a ball. Figure 9 shows what Haar features were used in this project.

The score of the feature for each sample is computed by subtracting pixel values under the black region of the Haar feature from the sum of the pixels values under the white region. This is done by using integral image, which allows computing the sum of a block with up to 4 array references.

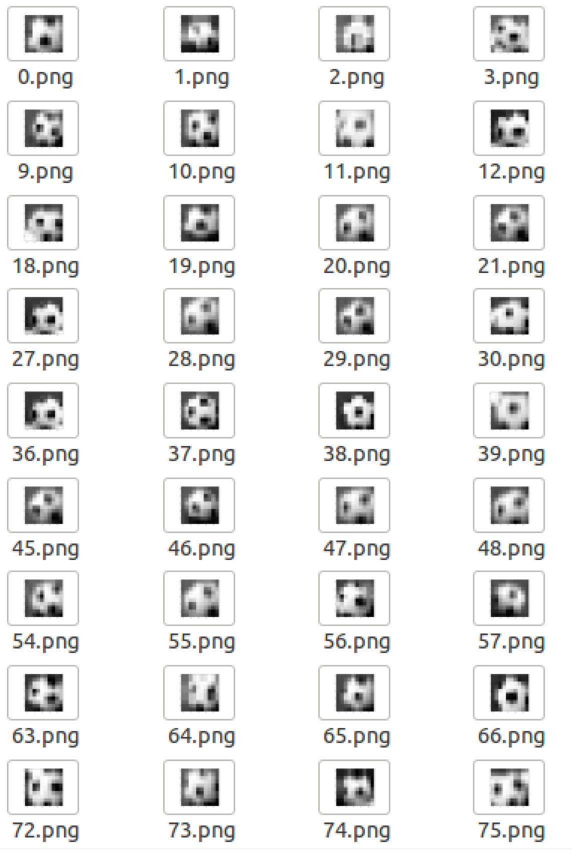


Fig. 7: Positive samples in the training (and testing set.)

With a good "weak" classifier, it turned out that the data can be separated quite well already (Figure 10). However, we see from the figure that there are quite a lot of false negatives and also some false positives. In order to improve this situation we combine many (50-100) weak classifiers and perform a weighted majority voting by all of them and then we decide what is ball and what is not a ball.

Now in order for us to properly combine all those weak classifiers and, more importantly, in order to select the strongest of all the possible weak classifiers, we use Adaboost which is described in the following section.

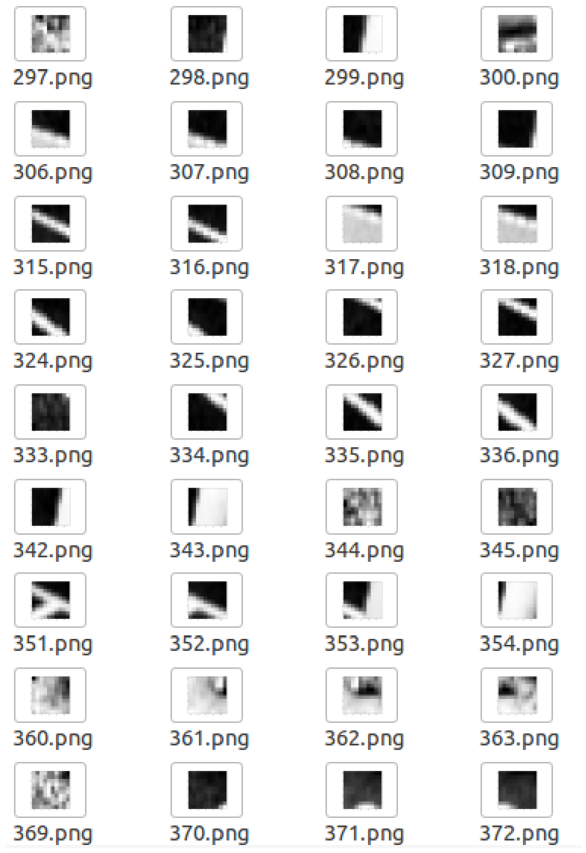


Fig. 8: Negative samples in the training (and testing set.)

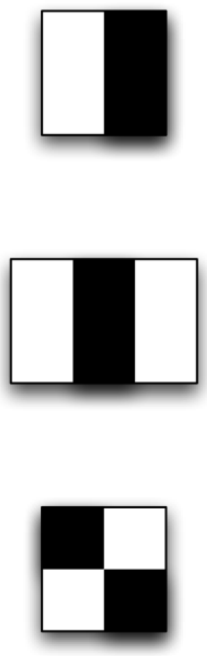
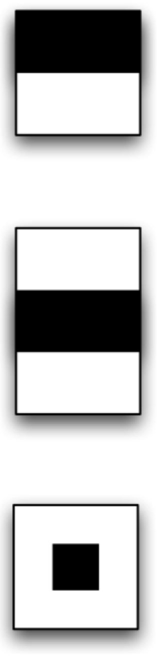



Fig. 9: Haar features used for weak classifiers.

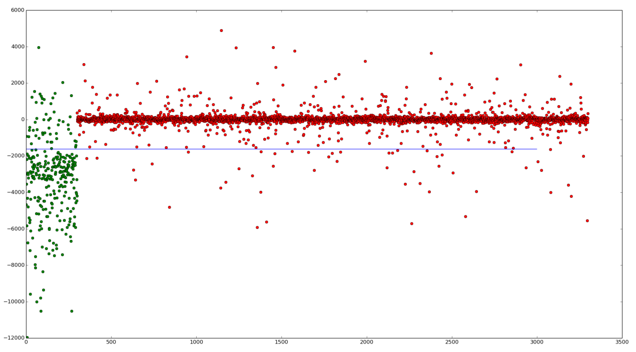
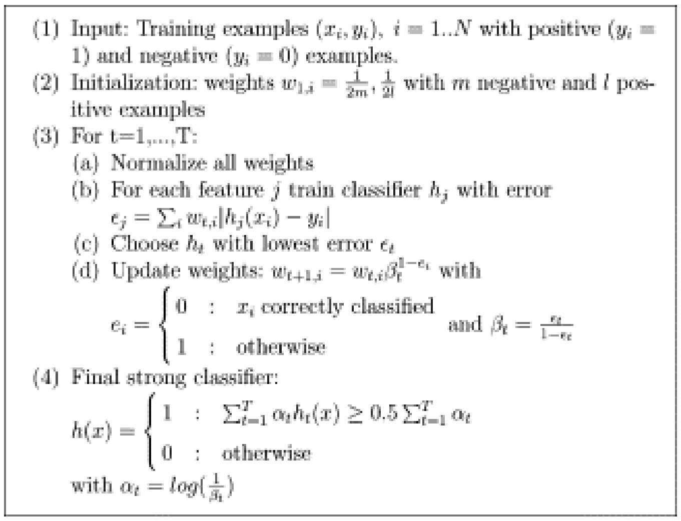


Fig. 10: Data separation with one Haar-feature: green - positive samples; red - negative samples; blue line - feature threshold.

IV. Adaboost for selecting the strongest of the weakest and combining them

The algorithm for Adaboost is shown in the Figure 11.



```
(1) Input: Training examples $(x_i, y_i)$, $i = 1..N$ with positive ($y_i = 1$) and negative ($y_i = 0$) examples.
(2) Initialization: weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ with $m$ negative and $l$ positive examples
(3) For t=1,...,T:
    (a) Normalize all weights
    (b) For each feature $j$ train classifier $h_j$ with error
        $e_j = \sum_i w_{t,i}|h_j(x_i) - y_i|$
    (c) Choose $h_t$ with lowest error $e_t$
    (d) Update weights: $w_{t+1,i} = w_{t,i}\beta_t^{1-e_i}$ with
        $e_i = \begin{cases} 0 & : \; x_i \text{ correctly classified} \\ 1 & : \; \text{otherwise} \end{cases}$ and $\beta_t = \frac{e_t}{1-e_t}$
(4) Final strong classifier:
    $h(x) = \begin{cases} 1 & : \; \sum_{t=1}^{T} \alpha_t h_t(x) \geq 0.5 \sum_{t=1}^{T} \alpha_t \\ 0 & : \; \text{otherwise} \end{cases}$
    with $\alpha_t = log(\frac{1}{\beta_t})$
```

Fig. 11: Adaboost algorithm for Haar-feature classifier cascade training [3].

Algorithm implemented and used in this project is almost exactly the same as the figure above describes, with the exception that the formula used for the final strong classifier decision is following:

$$h(x) = sign(\sum_j a_j \cdot h_j(x))$$

This is due to the fact that the labels in our framework are “1” for positive samples and “-1” for negative samples, and not 1-0.

The source code of the algorithm can be found in the public GitHub repository in [4].

V. Cascade of weak classifiers

The result of the Adaboost, the strongest of weak classifiers, is used in the final algorithm in the form of a cascade. That means that after applying few strongest weak classifiers as a first stage of the cascade, many negative windows can be eliminated already and there is no need to apply more weak classifiers (stages two, three etc.) to those windows. This trick speeds up the process of detection.

The Python code for the Adaboost used in this project parses the output of the trained cascade into the C++ code, which I can copy directly, add to the soccer software and test on the robot. There are some additional features in the code, such as visualizing how the data is separated by every individual weak classifier, or visualizing how the cascade evolves (separates data better and better) as the new weak classifier is added to the cascade. It is also possible to save the samples from the testing set to the folders TP, TN, FP and FN to analyze what samples were classified correctly or misclassified, to allow user to improve the quality of the training data.

VI. Results

Figures 12 - 14 illustrate the performance of the Adaboost cascades on the samples from the testing set (unseen data). Images containing the ball are represented by green dots, while non-ball images are red dots. Each dot is assigned a score by the trained cascade of how well test image matches soccer ball features (Y axis). X axis represents test sample index.

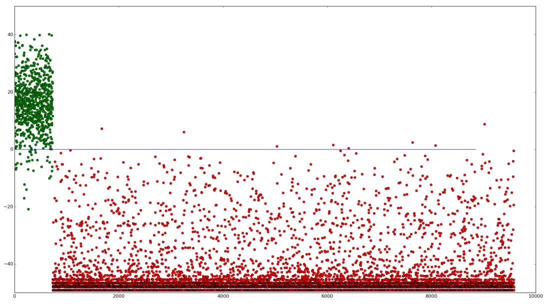


Fig. 12: Data separation with a cascade trained on the 60% of data for 158 minutes.

The computation of 103 features on the half of all possible subwindows of the frame takes around 5000 nanoseconds, which seems to be acceptable speed of processing in our framework. However, the detection of an old orange RoboCup SPL ball was done in just 300 nanoseconds on average.

VII. Conclusion and future work

The issue of RoboCup SPL ball detection challenge and the solution have been introduced in this report. The Haar-feature classifier has been explained. Adaboost algorithm has been presented with the reference to the source code and the classifier cascade architecture has been introduced. The results of the experiments on the data show good performance in separating positive and negative samples. However, I was still struggling in getting it to fully work on the NAO robot due to the (still) relatively high number of false positives - some lines and other objects are still detected as balls. This can maybe occur due to the fact that not all possible objects under all possible scales, rotations and sizes are present in the training set. And getting all that in the training set is quite challenging task itself. And also maybe the problem is something different than that, like model built on the data collected by one robot is not well-applicable to be used on another robot.

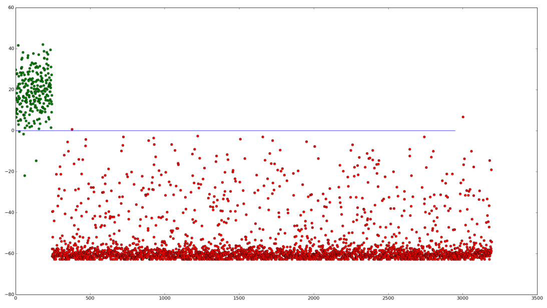


Fig. 13: Data separation with cascade trained on 80% of data, subwindow size 10x10.

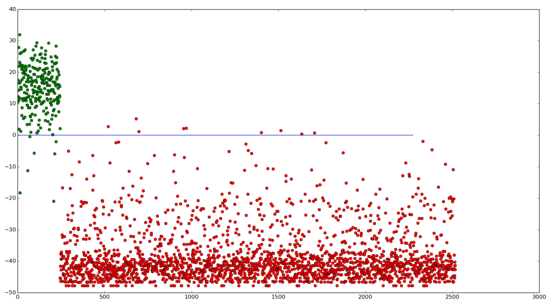


Fig. 14: Data separation with cascade trained on 80% of data, subwindow size 14x14.

References

[1] NAO humanoid robot. "https://www.softbankrobotics.com/emea/en/nao"

[2] "Rapid object detection using a boosted cascade of simple features" P. Viola and M. Jones

[3] "Real-Time Object Tracking for Soccer-Robots without Color Information" A. Treptow, A. Masselli and A. Zell

[4] Repository with the source code of the project https://github.com/anastasiabolotnikova/adaboost

[5] "A Foveated Vision System for Robotic Soccer" http://cgi.cse.unsw.edu.au/~robocup/2014ChampionTeamPaperReports/20110825-Carl.Chatfield-VisionFoveated.pdf