Machine Learning and Computer Vision – Boundary Detection

Olivier Jais-Nielsen

1. Introduction

We study here the advantages of using machine learning methods for the problem of boundary detection in an image. The problem of boundary detection is not well posed because a boundary does not have a completely satisfactory mathematical definition as it is a subjective notion very much linked to the semantics of the content of the image. There are however low-level algorithms able to extract edges candidates in a very conservative way, reducing the problem of finding edges to a subset of an input image without losing good candidates but with many false positives. Additionally, there are many low-level features that can be extracted from an image and that are loosely linked with the presence of edges.

Therefore, the idea is to train a classification algorithm over these features on test images where the boundaries are known and then to test on the same kind of features extracted from other images, restricting the search on the domain provided by a conservative edge detector.

2. Experiments

The experiments were performed on a single input image from which the actual boundaries are known. The training was performed on a random subset of the features and testing was performed on another disjoint random subset. Testing was also performed on the complete feature set in order to display the resulting boundaries.

The training process (and following testing) is performed in the case the training data contains as much of both classes and in the more general case. Additionally, in the first case, the first half of the training data contains data labeled with one class and the second half with the other class.

a. Low-level feature extraction

The raw low-level edge extraction algorithm that will provide the subset of the test image on which the actual boundaries will be searched is the Canny edge detector with a very low threshold.

Figure 1 - Left: input image. Center: actual edges. Right: Canny edge detector output

The features extracted from the images at the edge candidate locations on which the classification is performed are composed of several luminance gradients, color gradients and texture gradients taken at different scales and orientations. These features are quite natural as boundaries are characterized by strong changes in the visual characteristics of an image, such as its luminance, its color or its texture.

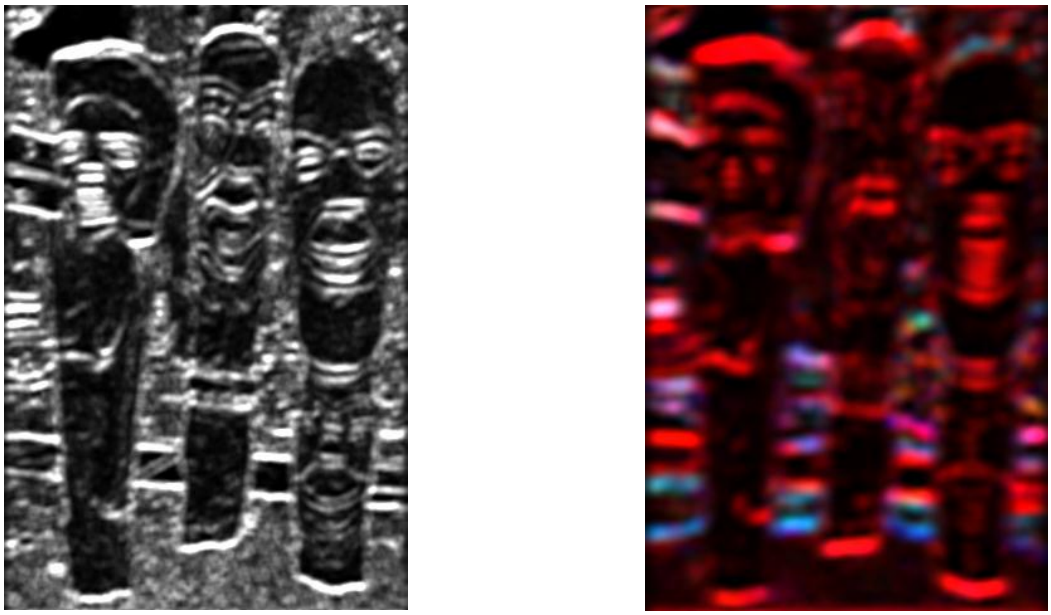

Figure 2 - Luminance and color gradient maps for some scale and orientation.

b. Logistic regression

The logistic regression is optimized using the Iterative reweighted least square algorithm. The convergence is quite fast, it takes less than 10 iterations.

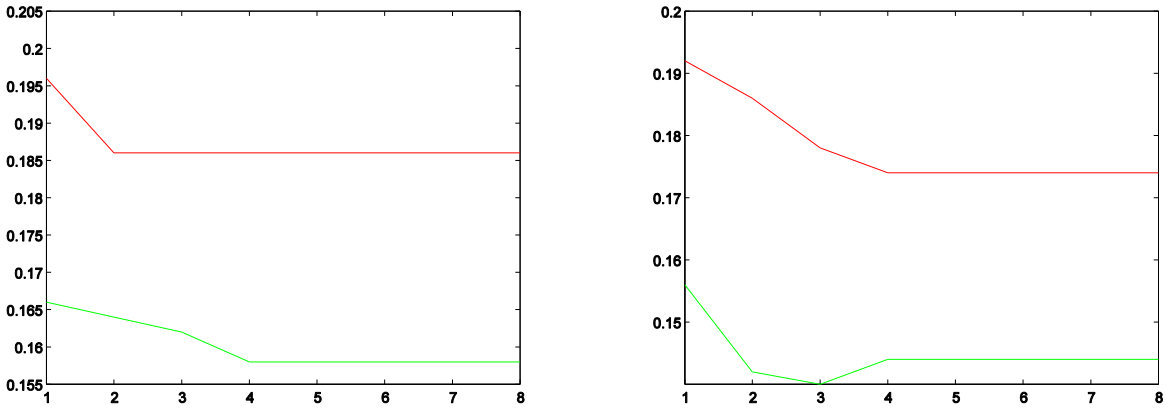

Figure 3 - Proportion of misclassified data for the logistic regression as a function of the number of IRLS iterations. In red for the test data and in green for the train data. Left: the train data contains as much of both classes. Right: no constraint on the classes in the training data.

The accuracy of the results can also be visually evaluated by displaying the conditional probability distribution of belonging to a class knowing the value of the features.

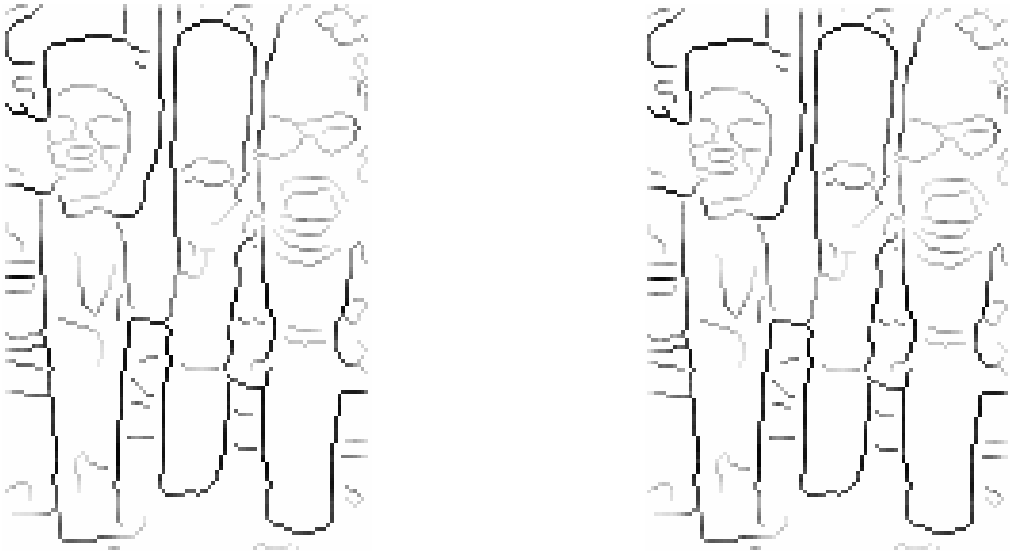

Figure 4 - Probability of belonging to an edge determined by the logistic regression (the darker the closer to 1). Left: the train data contains as much of both classes. Right: no constraint on the classes in the training data.

Here are the actual results of the classification (i. e. a hard threshold on the previous probability distribution).

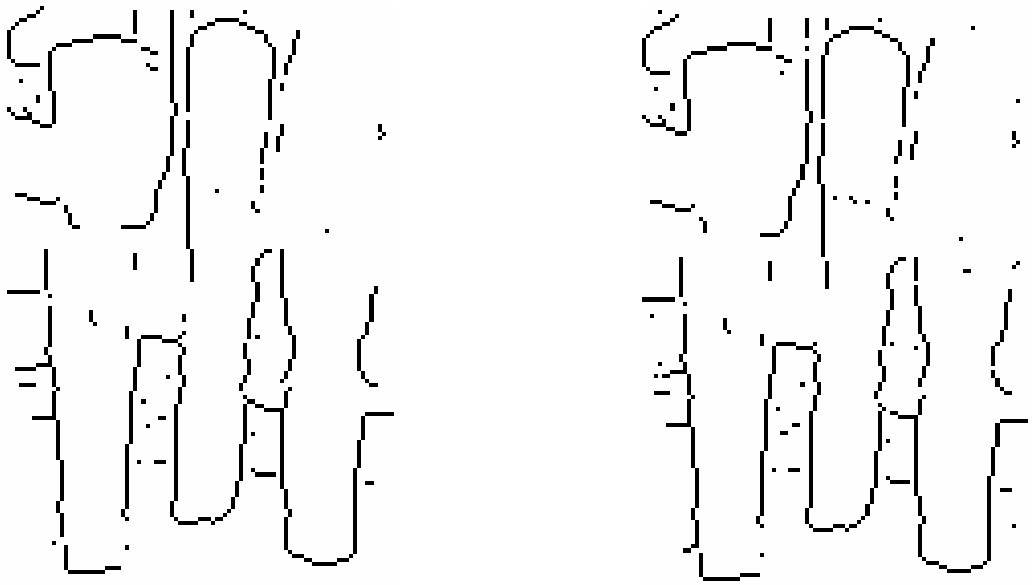

Figure 5 - Final class labels on all the data (black represents the boundary class). Left: the train data contains as much of both classes. Right: no constraint on the classes in the training data.

We can see from the misclassification rates that the classification is better when the training data is chosen completely randomly.

c. Adaboost

The convergence of the adaboost training process is not strict and seems to be longer than 500 rounds when considering the misclassification rate for the training data. However, the misclassification rate for the testing data seems to remain roughly stable after 300 rounds.

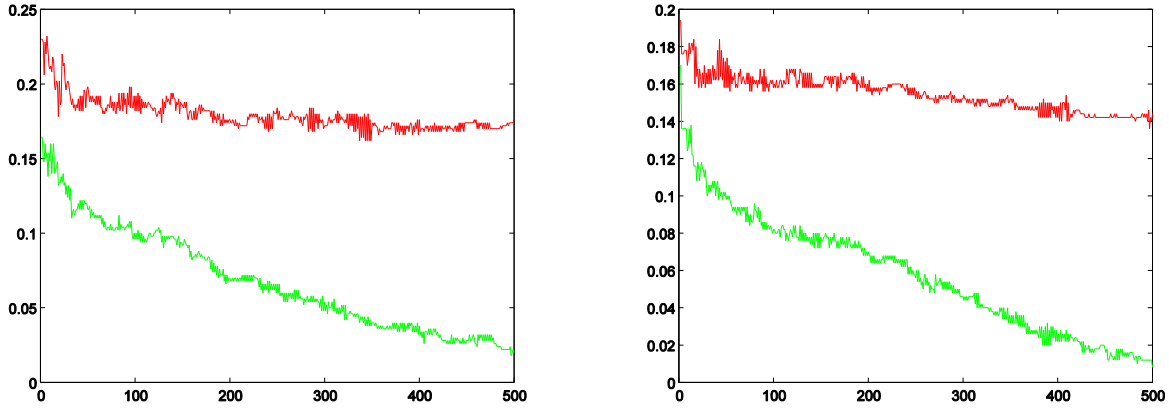

Figure 6 - Proportion of misclassified data for adaboost as a function of the number of rounds. In red for the test data and in green for the train data. Left: the train data contains as much of both classes. Right: no constraint on the classes in the training data.

For adaboost, the resulting decision function has, for a feature vector x and a label $y \in \{-1, 1\}$, the form $y = \mathrm{sgn}\big(f(x)\big)$. However, we have the following relationship:

$$f(x) = \frac{\mathbb{P}(y = 1|x)}{\mathbb{P}(y = -1|x)}$$

Therefore

$$\mathbb{P}(y = 1|x) = \frac{1}{1 + \exp(-f(x))}$$

We can then, as for the logistic regression plot the posterior probability of a boundary.

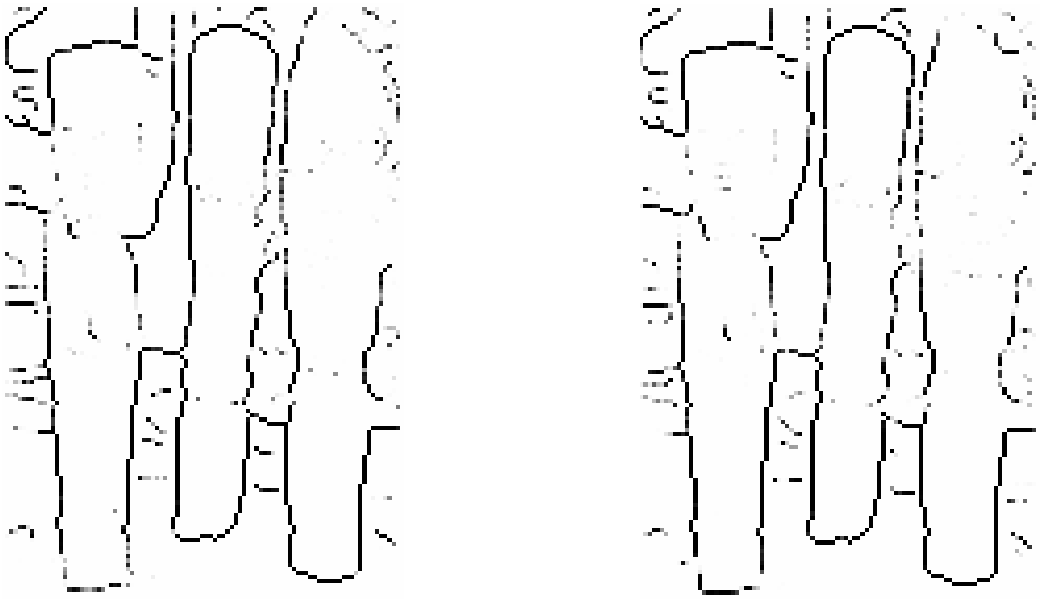

Figure 7 - Probability of belonging to an edge determined by adaboost (the darker the closer to 1). Left: the train data contains as much of both classes. Right: no constraint on the classes in the training data.

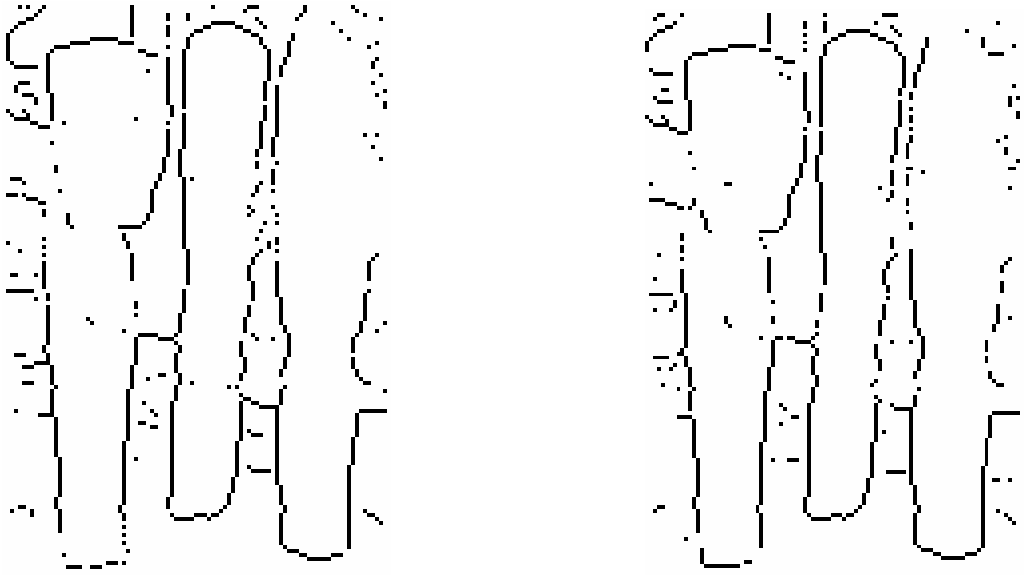

Figure 8 - Final class labels on all the data (black represents the boundary class). Left: the train data contains as much of both classes. Right: no constraint on the classes in the training data.

d. Comparison

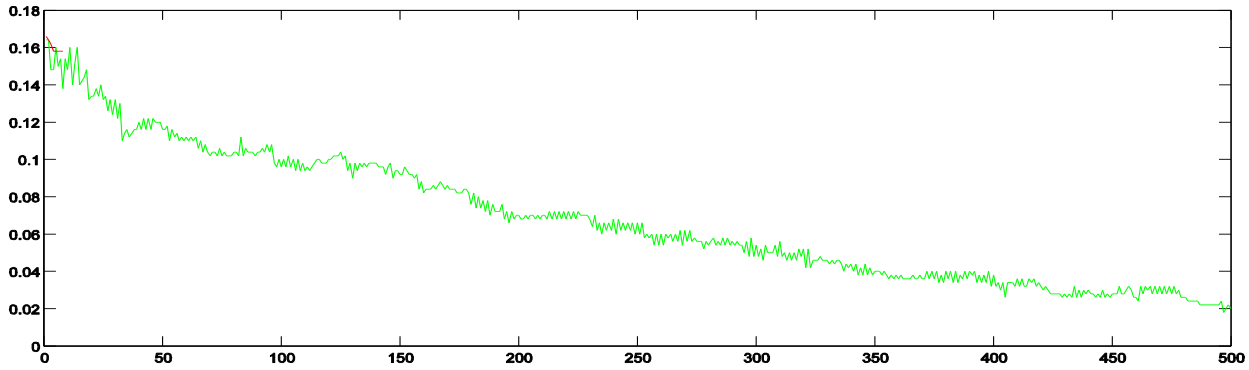

Figure 9 - Misclassification rate on the training data for the logistic regression (red) as a function of the number optimization iterations and for adaboost (green) as a function of the number of rounds. No constraint on the classes in the training data.

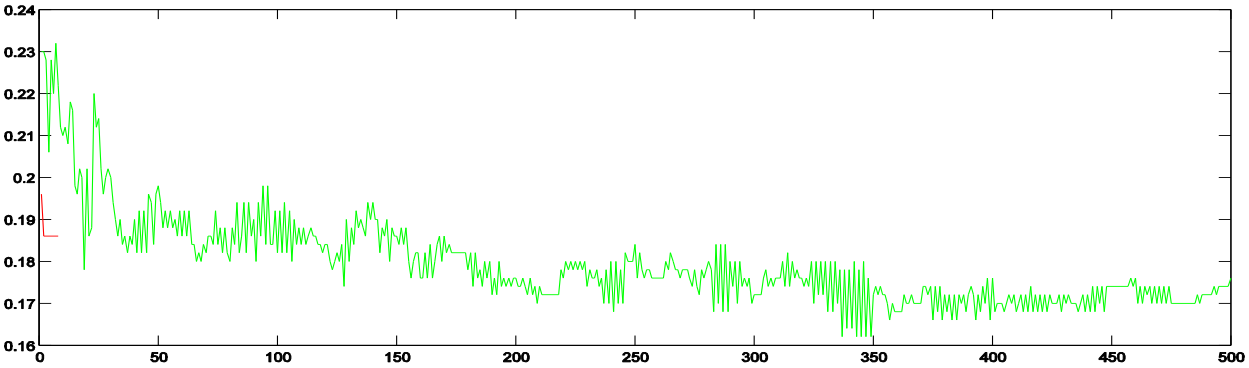

Figure 10 - Misclassification rate on the test data for the logistic regression (red) as a function of the number optimization iterations and for adaboost (green) as a function of the number of rounds. No constraint on the classes in the training data.

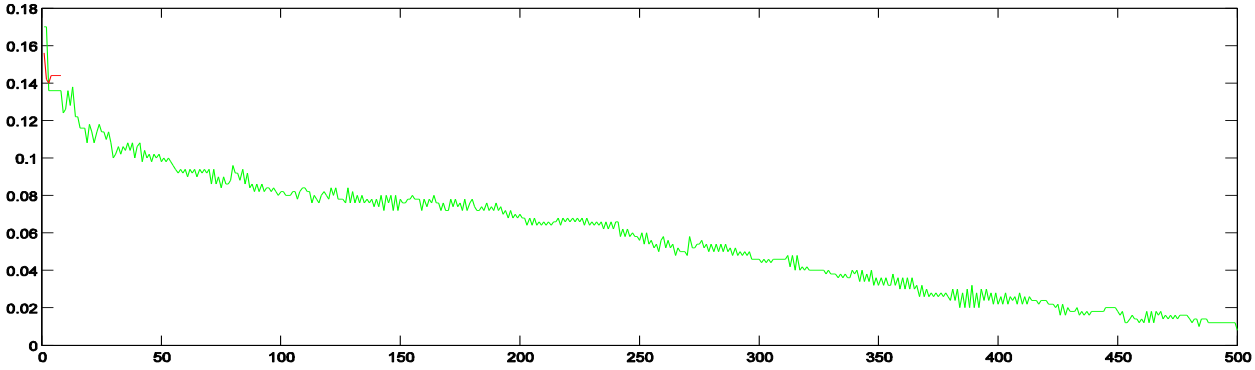

Figure 11 - Misclassification rate on the training data for the logistic regression (red) as a function of the number optimization iterations and for adaboost (green) as a function of the number of rounds. The train data contains as much of both classes.

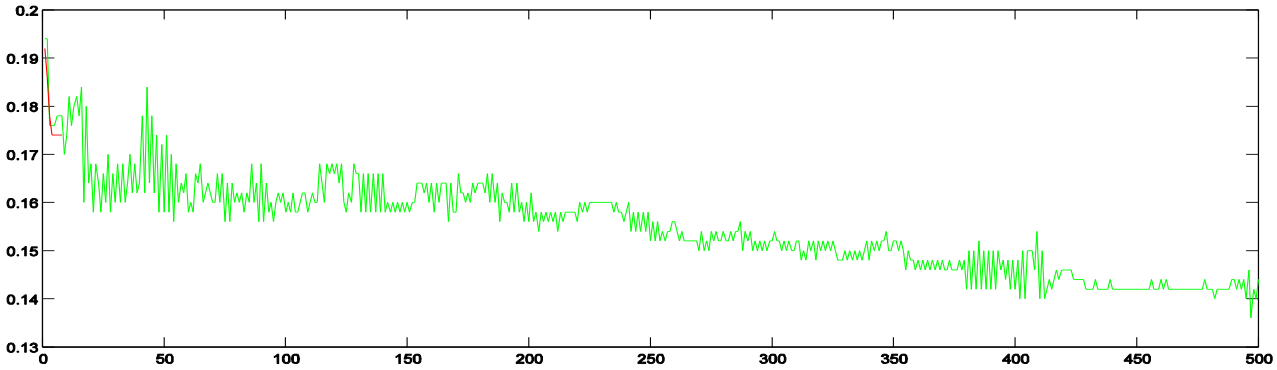

Figure 12 - Misclassification rate on the test data for the logistic regression (red) as a function of the number optimization iterations and for adaboost (green) as a function of the number of rounds. The train data contains as much of both classes.

It appears that although the logistic regression converges faster, the adaboost classifier performs much better.

3. Conclusion

Adaboost performs much better than the logistic regression on this task. Indeed, as show in Figure 13, both classes show a strong overlap in the 8-dimensional feature-space and therefore are hard to separate by a hyperplan in contrast with a very adaptive decision function produced by adaboost.

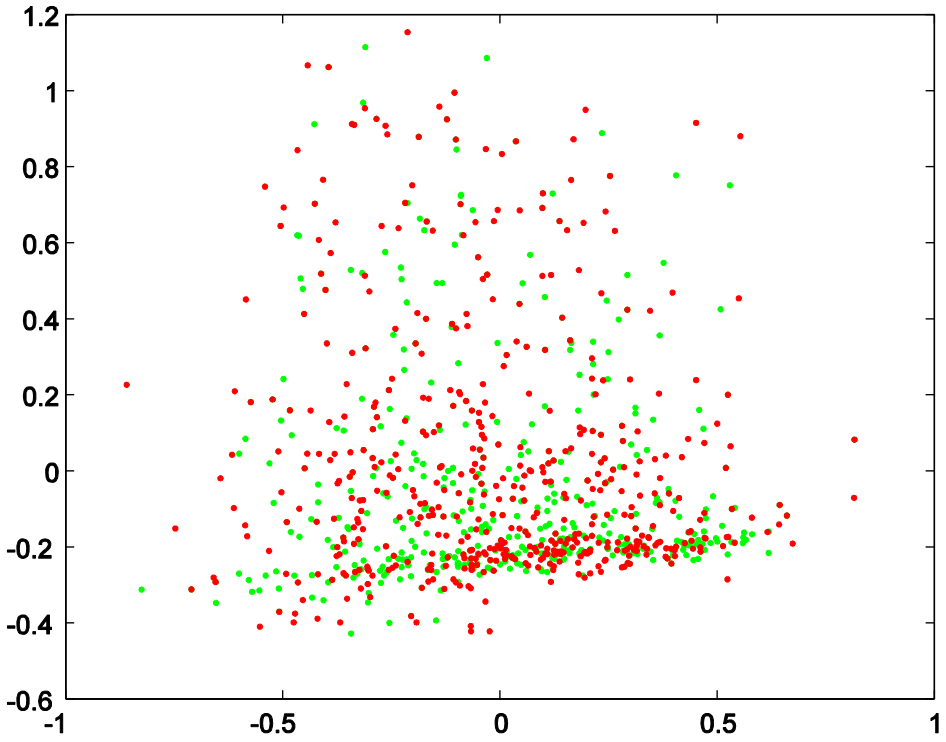

Figure 13 - Training features (green) and testing features (red) projections on the first principal component plane.

Finally, imposing the same amount of training features for each classes decreases the efficiency of both classifiers which is quite logical since it makes the training data less representative of the actual class distribution.