Recognition quality enhancement

In a situation where for a given training set it is difficult to build a single classifier of sufficiently good quality, you can try to design a larger number of weak classifiers and combine their results.

We hope that the differences between individual classifiers, put together with an appropriate criterion for making the final decisions will result in a better outcome (i.e. recognition ratio or other objective function).

- Bagging (*Bootstrap AGGregatING*)
- AdaBoost (*ADAptive BOOSTing*)
- Using AdaBoost in real-time object detection

Ludmila I. Kuncheva, *Combining Pattern Classifiers. Methods and Algorithms*, Wiley Interscience, 2004

Materiały przygotowano dzięki wsparciu udzielonemu z funduszy norweskich i funduszy EOG, pochodzących z Islandii, Liechtensteinu i Norwegii oraz środków krajowych

Bagging

Ideally, the classifiers comprising the committee should be built on different data (training) sets from the target population.

Because in practice our training set is limited, we draw samples from the training set (with replacement) to create a training set for a particular classifier (*bootstrap sample*).

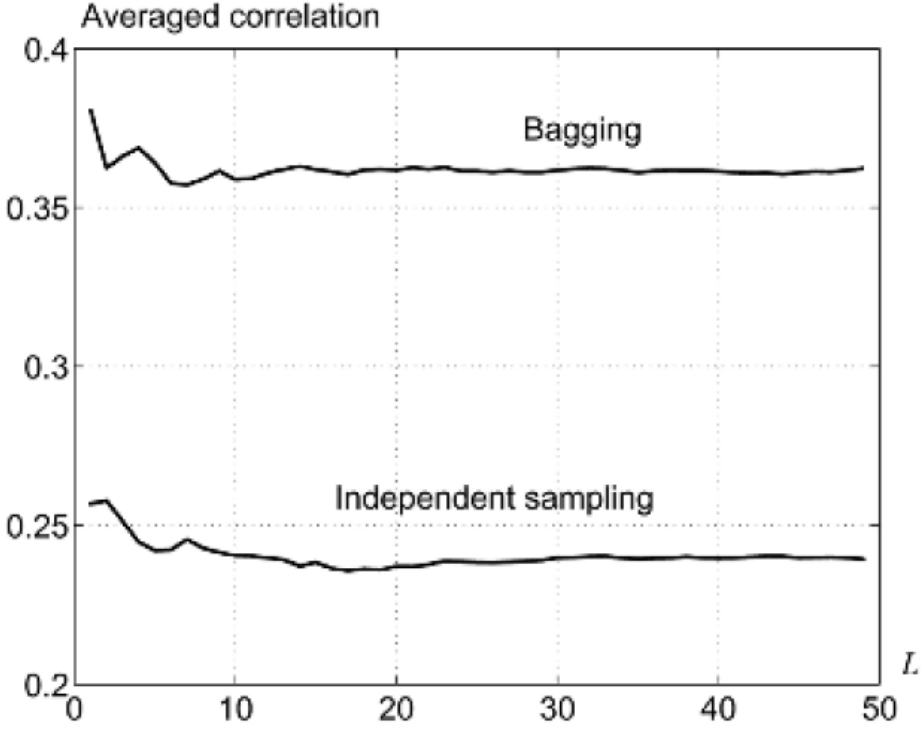
After building L classifiers (definitely better are in this case unstable classifiers, decision trees for instance) we are ready for classification.

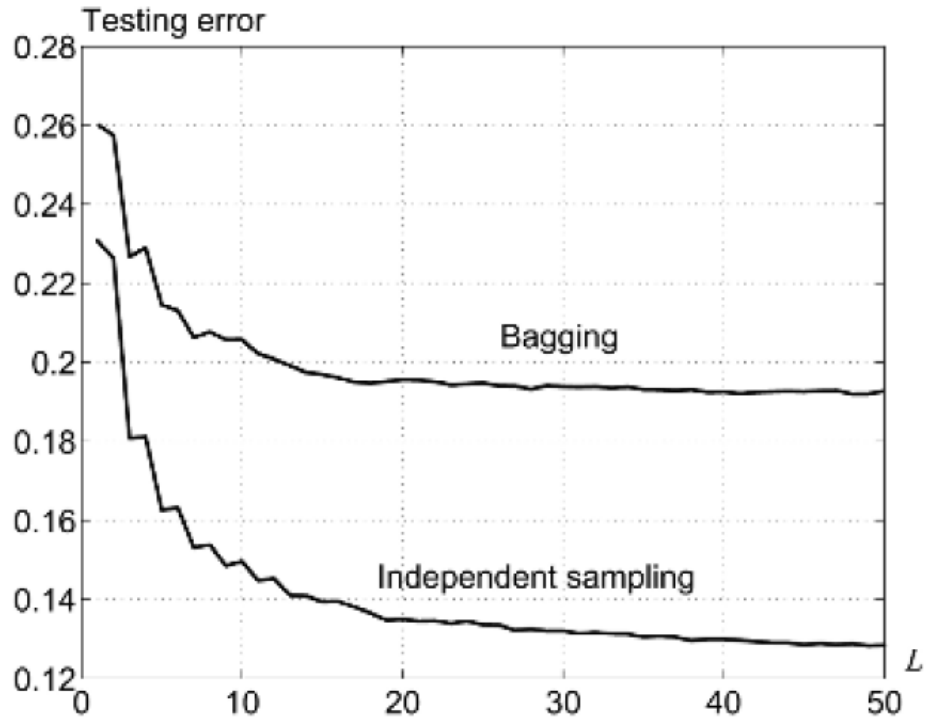
Each classifier votes for just one class of the sample x under classification. The final decision is determined by ordinary majority voting.

Materiały przygotowano dzięki wsparciu udzielonemu z funduszy norweskich i funduszy EOG, pochodzących z Islandii, Liechtensteinu i Norwegii oraz środków krajowych

Bagging – why does it work?

The key here is the independence of the classifiers - if they are independent indeed, the qualified majority voting guarantees better classification result than any of single classifiers.





These charts show an average classifiers' decision correlation and quality classification on the test set (for artificial data points arranged in a checkerboard pattern in the unit square). Conclusion: it's better to have more data than to use bagging.

Materiały przygotowano dzięki wsparciu udzielonemu z funduszy norweskich i funduszy EOG, pochodzących z Islandii, Liechtensteinu i Norwegii oraz środków krajowych

Random Forests

This is the classic implementation of bagging inevitably using as a base classifier a decision tree. A single tree can be created based on selected features, selected samples of training data or with changed parameters used during tree construction.

One of the best heuristics is to use two samplings:

1. We select the bootstrap sample from the entire data set as a training set for a specific tree.
2. In each node of the tree under construction we randomly select a subset of features, among which we will seek the best feature for the separation of samples in this node.

Materiały przygotowano dzięki wsparciu udzielonemu z funduszy norweskich i funduszy EOG, pochodzących z Islandii, Liechtensteinu i Norwegii oraz środków krajowych

Pasting Small Votes

This approach is associated with the emergence of large data sets, which as a whole don't allow sensible analysis. Single classifiers are trained on relatively small samples from the whole set called bites. There are two methods used to select bites: random and based on sample's importance (let's name them Rvotes or Ivotes).

Random bites are basically the classic implementation of bagging, although the sample size for a single classifier is a lot smaller than the training set.

The basic decision that the designer has to take is the number of elementary classifiers L. A final decision is a result of majority voting.

Because the training data set is huge we can afford for validation set which can be used to determine the appropriate value of L.

Materiały przygotowano dzięki wsparciu udzielonemu z funduszy norweskich i funduszy EOG, pochodzących z Islandii, Liechtensteinu i Norwegii oraz środków krajowych

Importance sampled small votes

In each step we assess generalization properties of the committee built so far. The idea is that the training set for the next classifier should consist in half of “difficult” samples and in half of the "easy" samples. The procedure to select single sample from the whole training set Z is following:

1. We draw randomly sample z_j form Z.
2. We create a list of classifiers that haven’t „seen” sample z_j (*out-of-bag classifiers*). If the list is empty we reject sample z_j.
3. We perform majority voting of the classifiers in the list.
4. If z_j is misclassified we add it to the training set. If z_j is classified properly we add it to the training set with probability depending on generalization error (< 0.5).

Materiały przygotowano dzięki wsparciu udzielonemu z funduszy norweskich i funduszy EOG, pochodzących z Islandii, Liechtensteinu i Norwegii oraz środków krajowych

AdaBoost (ADAptive BOOSTing)

Set Z is sampled first with uniform distribution, which varies in the course of learning by increasing chances of selecting samples that are "difficult". Distribution of samples Z is updated after each pass (or creating a component classifier): misclassified samples' weights are increased, while properly classified samples have their weights decreased.

Again, the fundamental decision is to determine the number of component classifiers L (and, of course, the number of samples in the training set for a single classifier).

In the classification phase we weighted classifiers' votes (the weights are determined based on the classification quality during learning).

Materiały przygotowano dzięki wsparciu udzielonemu z funduszy norweskich i funduszy EOG, pochodzących z Islandii, Liechtensteinu i Norwegii oraz środków krajowych

AdaBoost.M1 – training phase

Weights initialization: $w_j^1 = 1/N$

For k=1, …, L

1. Select from Z training set S_k according to distribution w^k
2. Train classifier D_k on the training set S_k
3. Asses (weighted) classification quality of D_k: $\varepsilon_k = \sum_{j=1}^{N} w_j^k l_k^j$

   (l=1 when sample is misclassified; 0 in other cases)
4. If $\varepsilon_k = 0$ or $\varepsilon_k \geq 0.5$ reject classifier, reinitialize weights and return to point 1.
5. Assess D_k quality: $\beta_k = \frac{\varepsilon_k}{1-\varepsilon_k}$ ($\varepsilon_k \in (0, 0.5)$)
6. Update weights: $w_j^{k+1} = \frac{w_j^k \beta_k^{\left(1-l_k^j\right)}}{\sum_{i=1}^{N} w_i^k \beta_k^{\left(1-l_k^i\right)}}$

Materiały przygotowano dzięki wsparciu udzielonemu z funduszy norweskich i funduszy EOG, pochodzących z Islandii, Liechtensteinu i Norwegii oraz środków krajowych

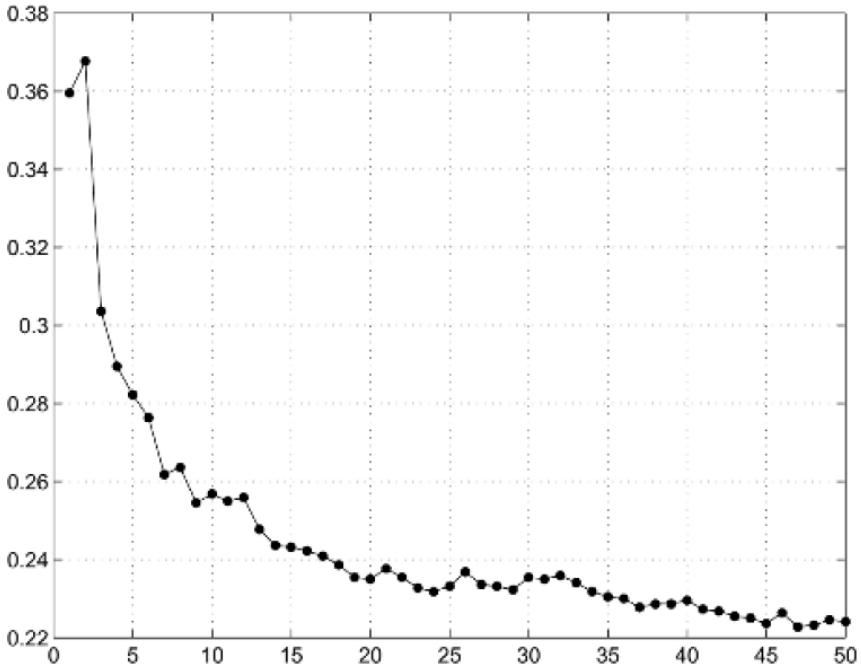
AdaBoost.M1 – classification phase

1. Sum the weighted votes cast on each class ω_t :

$$\mu_t(\mathbf{x}) = \sum_{D_k(\mathbf{x})=\omega_t} \ln\left(\frac{1}{\beta_k}\right)$$

2. Select class label for which μ_t has the max value:

$$\arg\max\left(\mu_t(\mathbf{x})\right)$$



Error rate on the testing set vs. number of component classifiers (data set: *forensic glass* from UCI)

Materiały przygotowano dzięki wsparciu udzielonemu z funduszy norweskich i funduszy EOG, pochodzących z Islandii, Liechtensteinu i Norwegii oraz środków krajowych

Real-time object detection

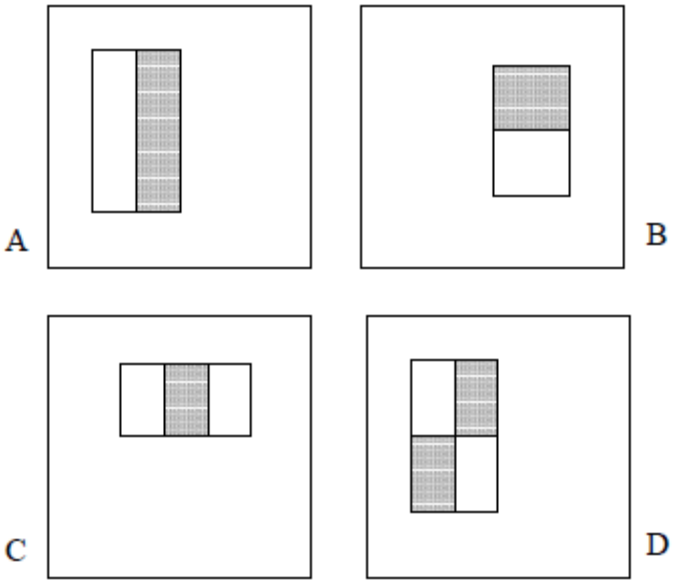
An interesting exapmle of AdaBoost practical application is face detection on images/photographs. Authors showed here very simple features with efficient method for their computation and how to use such simple features to train very good classifier.
This aglorithm is widely used in practice – in particular OpenCV library uses it in face and human sylwetka detection (here the optimized version dubbed *CascadeClassifier* is used).

Paul Viola, Michael Jones, *Robust Real-time Object Detection*, Second International Workshop on Statistical and Computational Theories of Vision – Modeling, Learning, Computing, and Sampling, Vancouver, Canada, July 13, 2001

Materiały przygotowano dzięki wsparciu udzielonemu z funduszy norweskich i funduszy EOG, pochodzących z Islandii, Liechtensteinu i Norwegii oraz środków krajowych

Object detection - features

The simple features used are reminiscent of Haar basis functions. Two-rectangle features are shown in (A) and (B). Figure (C) shows a three-rectangle feature, and (D) a four-rectangle feature.



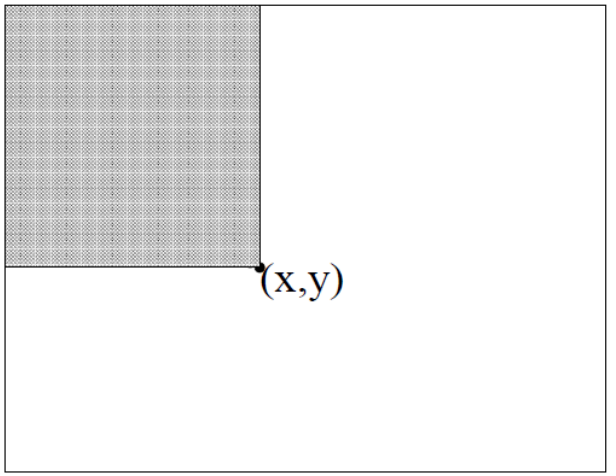
The sum of the pixels which lie within the white rectangles are subtracted from the sum of pixels in the grey rectangles.

Given that the base resolution of the detector (i.e. sliding window) is 24x24 (the exhaustive set of rectangle features is quite large: 45396). Using fixed detector size means that images must be analyzed at different scales. Note that with such a feature definition you can scale **coordinates** of rectangles instead of whole images.

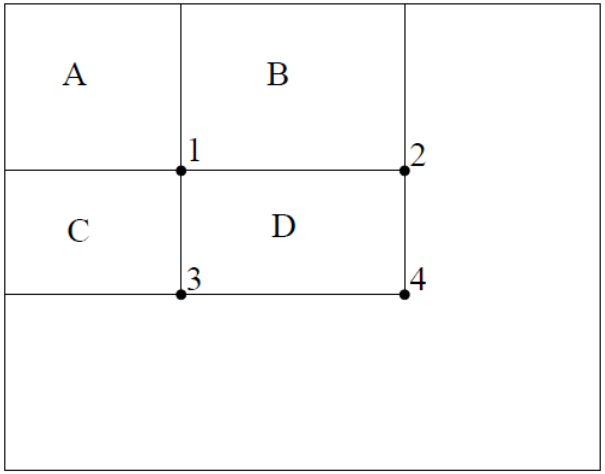
Materiały przygotowano dzięki wsparciu udzielonemu z funduszy norweskich i funduszy EOG, pochodzących z Islandii, Liechtensteinu i Norwegii oraz środków krajowych

Efficient feature computation

Rectangle features can be computed very rapidly using an intermediate representation for the image called the integral image. The integral image at location (x, y) contains the sum of the pixels above and to the left of (x, y) inclusive.



The sum of the pixels within rectangle D can be computed with four array references. The value of the integral image at location 1 is the sum of the pixels in rectangle A. The value at location 2 is A + B, at location 3 is A + C, and at location 4 is A + B + C + D. The sum within D can be computed as 4 + 1 - (2 + 3).



Materiały przygotowano dzięki wsparciu udzielonemu z funduszy norweskich i funduszy EOG, pochodzących z Islandii, Liechtensteinu i Norwegii oraz środków krajowych

Base classifiers

In the 24x24 window we have 43396 features, of which just a very small number can be combined to form an effective classifier. The main challenge is to find these features.
The solution is set of very simple (and weak) classification functions each of which depend on a single feature. The weak learning algorithm is designed to select the single rectangle feature which best separates the positive and negative examples.

The classifier $h_j(\mathbf{x})$ is defined bye: feature f_j, threshold value θ_j and polarity p_j (its value is 1 or -1 – beautiful!!!) and operates as follows:

$$h_j(\mathbf{x}) = \begin{cases} 1 & jesli\ p_j f_j(\mathbf{x}) < p_j \theta_j \\ 0 & wpp. \end{cases}$$

Materiały przygotowano dzięki wsparciu udzielonemu z funduszy norweskich i funduszy EOG, pochodzących z Islandii, Liechtensteinu i Norwegii oraz środków krajowych

Classifiers' selection – AdaBoost

We have the training set of images $(x_1, y_1), \ldots, (x_n, y_n)$ where $y_i = 0, 1$ for negative (no face) and positive (face) examples respectively.
The important thing to remember is the assymetry in the size of positive (small) and negative (large) example subsets.
It's quite natural, because image fragments (of the size of sliding window) mostly don't contain faces.

Weights are initialized with $w_{1,i} = \frac{1}{2m}$ for $y_i = 0$, where *m* is the number of negative examples and with $w_{1,i} = \frac{1}{2l}$ for $y_i = 1$, where *l* is the number of positive examples.
After fixing (guessing?) the number of classifiers T we can start the main AdaBoost procedure.

Materiały przygotowano dzięki wsparciu udzielonemu z funduszy norweskich i funduszy EOG, pochodzących z Islandii, Liechtensteinu i Norwegii oraz środków krajowych

Classifiers' (feature) selection – AdaBoost

For $t = 1 \ldots T$

1. Normalize the weights $w_{t,i} = \frac{w_{t,i}}{\sum_{j=1}^{N} w_{t,j}}$
2. For each feature *j* prepare a classifier h_j and compute it's classification error: $\varepsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right|$
3. Choose the classifier h_t with the lowest error ε_t
4. Compute the classifier's weight: $\beta_t = \frac{\varepsilon_t}{1 - \varepsilon_t}$
5. Update the weights $w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$ where $e_i = 0$ if exapmle is classified correctly, and $e_i = 1$ otherwise.

Materiały przygotowano dzięki wsparciu udzielonemu z funduszy norweskich i funduszy EOG, pochodzących z Islandii, Liechtensteinu i Norwegii oraz środków krajowych

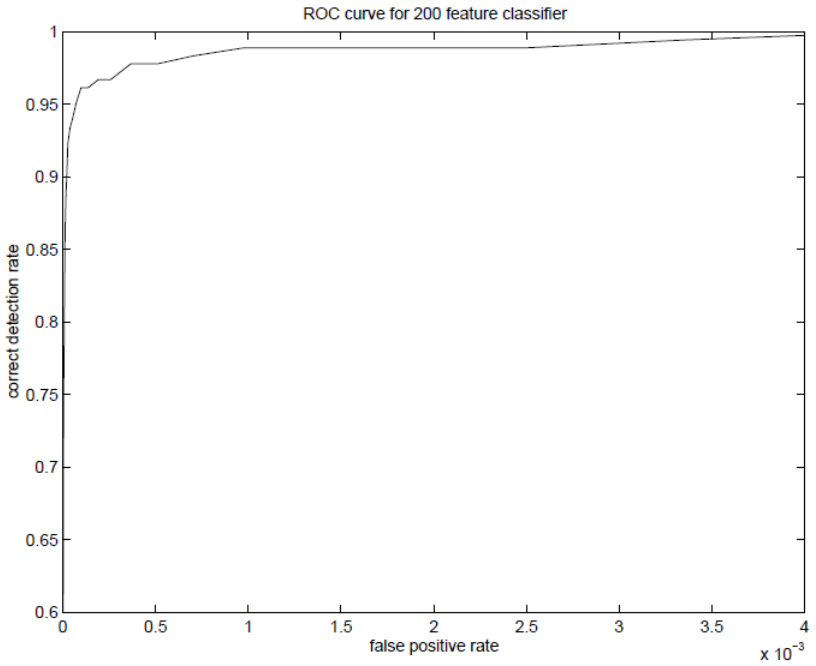
AdaBoost – classification

The final strong classifier is:

$$h(\mathbf{x}) = \begin{cases} 1 & if \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & otherwise \end{cases}$$

, where $\alpha_t = \log \frac{1}{\beta_t}$

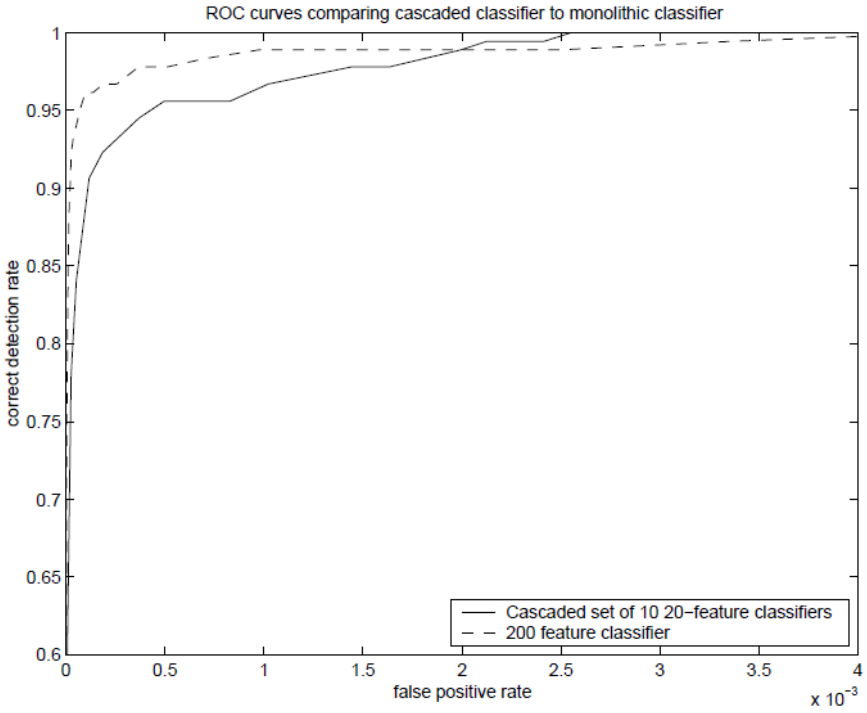
Trained in this way *monolithic* classifier constructed from 200 features works very well: with the detection rate of 95% the classifier yielded a false positive rate of 1 in 14084 on a testing dataset.
The problem is that this classifier is still far from real-time operation.



Materiały przygotowano dzięki wsparciu udzielonemu z funduszy norweskich i funduszy EOG, pochodzących z Islandii, Liechtensteinu i Norwegii oraz środków krajowych

Classifiers Cascade

The solution is to construct a cascade of classifiers which achieves increased detection performance while radically reducing computation time. The key insight is that smaller (and more efficient) boosted classifiers can be constructed which reject many of the negative sub-windows while detecting almost all positive instances.



Materiały przygotowano dzięki wsparciu udzielonemu z funduszy norweskich i funduszy EOG, pochodzących z Islandii, Liechtensteinu i Norwegii oraz środków krajowych