Notes on HW 1 grading

- I gave full credit as long as you gave a description, confusion matrix, and working code
- Many people's descriptions were quite short and lacking information about the experiment – I did not take off for this, but I will be more specific next time.
- I took off points for non-working code, incorrect confusion matrix, and/or incorrect definition of accuracy

Homework 3:
SVMs and Feature Selection

- Data:
  - https://archive.ics.uci.edu/ml/datasets/Spambase
  - Create a subset of the data that has equal numbers of positive and negative examples.
  - Put data into format needed for the SVM package you're using
  - Split data into ~ ½ training, ½ test (each should have equal numbers of positive and negative examples)
  - Scale training data using standardization (as in HW 2)
  - Scale test data using standardization parameters from training data
  - Randomly shuffle training data
  - Randomly shuffle test data. Use only for testing at end of cross-validation or feature selection

- **Experiment 1:** Cross-validation using linear SVM to find best "C" parameter value
  - Use SVM^{light} (http://svmlight.joachims.org/) or any SVM package
  - Use linear kernel (default in SVM^{light})
  - Use 10-fold cross-validation to test values of C parameter in
    {0, .1, .2, .3, .4, .5, .6, .7, .8, .9, 1}

    - Split training set into 10 approx equal-sized disjoint sets S_i

    - For each value of the C parameter j :
      - For i = 1 to 10

        Select S_i to be the "validation" set

        Train linear SVM using C=j and all training data except S_i.

        Test learned model on S_i to get accuracy $A_{j,i}$

      - Compute average accuracy for C=j: $A_j = \sum_{i=1}^{10} A_{j,i}$

    - Choose value C= C* that results in highest A_j

  - Train new linear SVM using all training data with C=C*
  - Test learned SVM model on test data. Report accuracy, precision, and recall (using threshold 0 to determine positive and negative classifications)
  - Use results on test data to create an ROC curve for this SVM, using about 200 evenly spaced thresholds.

- **Experiment 2:** Feature selection with linear SVM
  - Using final SVM model from Experiment 1:
    - Obtain weight vector **w**. (For SVM^{light}, see https://www.cs.cornell.edu/people/tj/svm_light/svm_light_faq.html)

Select features:

  - For m = 1 to 57
    - Select the set of m features that have highest $|w_m|$
    - Train a linear SVM, SVM_m , on all the training data, only using these m features, and using C* from Experiment 1
    - Test SVM_m on the test set to obtain accuracy.
  - Plot accuracy vs. m

- **Experiment 3:** Random feature selection (baseline)
  - Same as Experiment 2, but for each m, select m features at random from the complete set. This is to see if using SVM weights for feature selection has any advantage over random.

What to include in your report

1. **Experiment 1:**
   - Which SVM package you used
   - The C* (best value of C) you obtained
   - Accuracy, Precision, and Recall on the test data, using final learned model
   - ROC Curve

2. **Experiment 2:**
   - Plot of accuracy (on test data) vs. *m* (number of features)
   - Discussion of what the top 5 features were (see https://archive.ics.uci.edu/ml/machine-learning-databases/spambase/spambase.names for details of features)
   - Discussion of the effects of feature selection (about 1 paragraph).

- **Experiment 3:**
   - Plot of accuracy (on test data) vs. *m* (number of features)
   - Discussion of results of random feature selection vs. SVM-weighted feature selection (about 1 paragraph).

What code to turn in

- Your code / script for performing cross-validation in Experiment 1
- Your code for creating an ROC curve in Experiment 1
- Your code / script for performing SVM-weighted feature selection in Experiment 2
- Your code / script for performing random feature selection in Experiment 3

Quiz on Thursday Feb. 5

Topics: Feature selection, ensemble learning

You will need a calculator!

Quiz review

Adaboost, Part 2

Recap of Adaboost algorithm

- Given S = $\{(x_1, y_1), ..., (x_N, y_N)\}$ where $x \in X$, $y_i \in \{+1, -1\}$
- Initialize $w_1(i) = 1/N$. (Uniform distribution over data)
- For $t = 1, ..., K$:
  1. Select new training set S_t from S with replacement, according to $\mathbf{w}_t$
  2. Train L on S_t to obtain hypothesis h_t
  3. Compute the training error ε_t of h_t on $\boldsymbol{S}$:

$$\varepsilon_t = \sum_{j=1}^{N} \mathbf{w}_t(j)\, \delta(y_j \neq h_t(\mathbf{x}_j)),$$

$$\text{where } \delta(y_j \neq h_t(\mathbf{x}_j)) = \begin{cases} 1 \text{ if } y_j \neq h_t(\mathbf{x}_j) \\ 0 \text{ otherwise} \end{cases}$$

If $\varepsilon_t > 0.5$, abandon h_t and go to step 1

– Compute coefficient:

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$$

– Compute new weights on data:

For i = 1 to N

$$\mathbf{w}_{t+1}(i) = \frac{\mathbf{w}_t(i)\ \exp(-\alpha_t y_i h_t(\mathbf{x}_i))}{Z_t}$$

where Z_t is a normalization factor chosen so that $\mathbf{w}_{t+1}$ will be a probability distribution:

$$Z_t = \sum_{i=1}^{N} \mathbf{w}_t(i)\ \exp(-\alpha_t y_i h_t(\mathbf{x}_i))$$

- At the end of K iterations of this algorithm, we have $h_1, h_2, \ldots, h_K$, and $\alpha_1, \alpha_2, \ldots, \alpha_K$

- Ensemble classifier:

$$H(\mathbf{x}) = \text{sgn} \sum_{t=1}^{K} \alpha_t h_t(\mathbf{x})$$

In-class exercises

Case Study of Adaboost #1:

Viola-Jones Face Detection Algorithm

P. Viola and M. J. Jones, Robust real-time face detection. International Journal of Computer Vision, 2004.

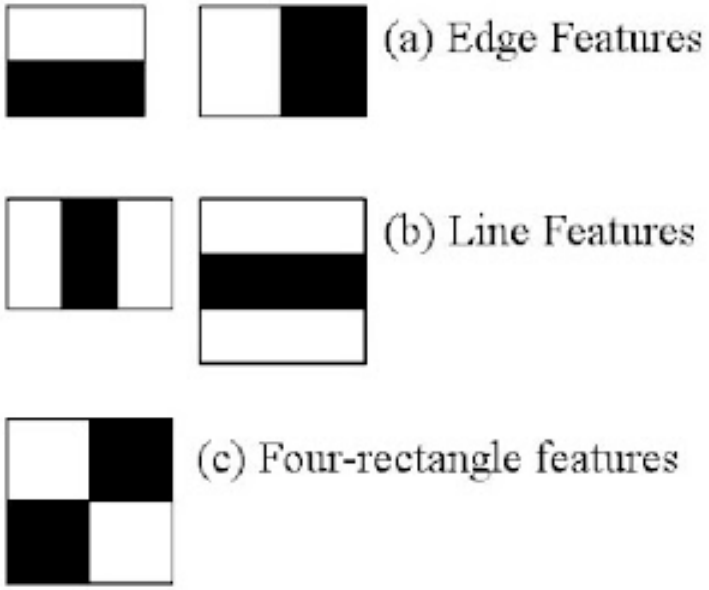
First face-detection algorithm to work well in real-time (e.g., on digital cameras)

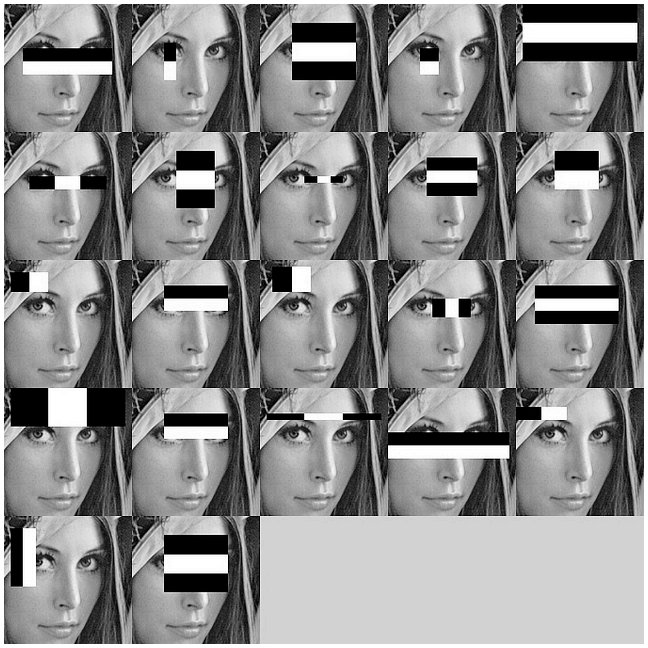
Training Data

- **Positive:** Faces scaled and aligned to a base resolution of 24 by 24 pixels.
- **Negative:** Much larger number of non-faces.

Figure 8. Example of frontal upright face images used for training.

Features





From ***http://makematics.com/research/viola-jones/***

Use rectangle features at multiple sizes and location in an image subwindow (candidate face).

For each feature f_j :

$$f_j = \sum_{b \in \text{black pixels}} \text{intensity(pixel } b\text{)} \; - \sum_{w \in \text{white pixels}} \text{intensity(pixel } w\text{)}$$

***Possible number of features per 24 x 24 pixel subwindow* > 180,000.**

Need feature selection!

Detecting faces

Given a new image:

- Scan image using subwindows at all locations and at different scales

- For each subwindow, compute features and send them to an ensemble classifier (learned via boosting). If classifier is positive ("face"), then detect a face at this location and scale.

Preprocessing: Viola & Jones use a clever pre-processing step that allows the rectangular features to be computed very quickly. (See their paper for description.)

They use a variant of AdaBoost to both select a small set of features and train the classifier.

Base ("weak") classifiers

For each feature f_j ,

$$h_j = \left\{ \begin{array}{c} 1 \text{ if } p_j f_j(x) < p_j \theta_j \\ -1 \text{ otherwise} \end{array} \right\}$$

where x is a 24 x 24-pixel subwindow of an image, θ_j is the threshold that best separates the data using feature f_j , and p_j is either -1 or 1.

Such features are called "decision stumps".

Boosting algorithm

- Given example images $(x_1, y_1), \ldots, (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negatives and positives respectively.

- For $t = 1, \ldots, T$:

  1. Normalize the weights,

$$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$

     so that w_t is a probability distribution.

  2. For each feature, j, train a classifier h_j which is restricted to using a single feature. The error is evaluated with respect to w_t, $\epsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right|$.

  3. Choose the classifier, h_t, with the lowest error ϵ_t.

4. Update the weights:

$$w_{t+1,i} = w_{t,i}\beta_t^{1-e_i}$$

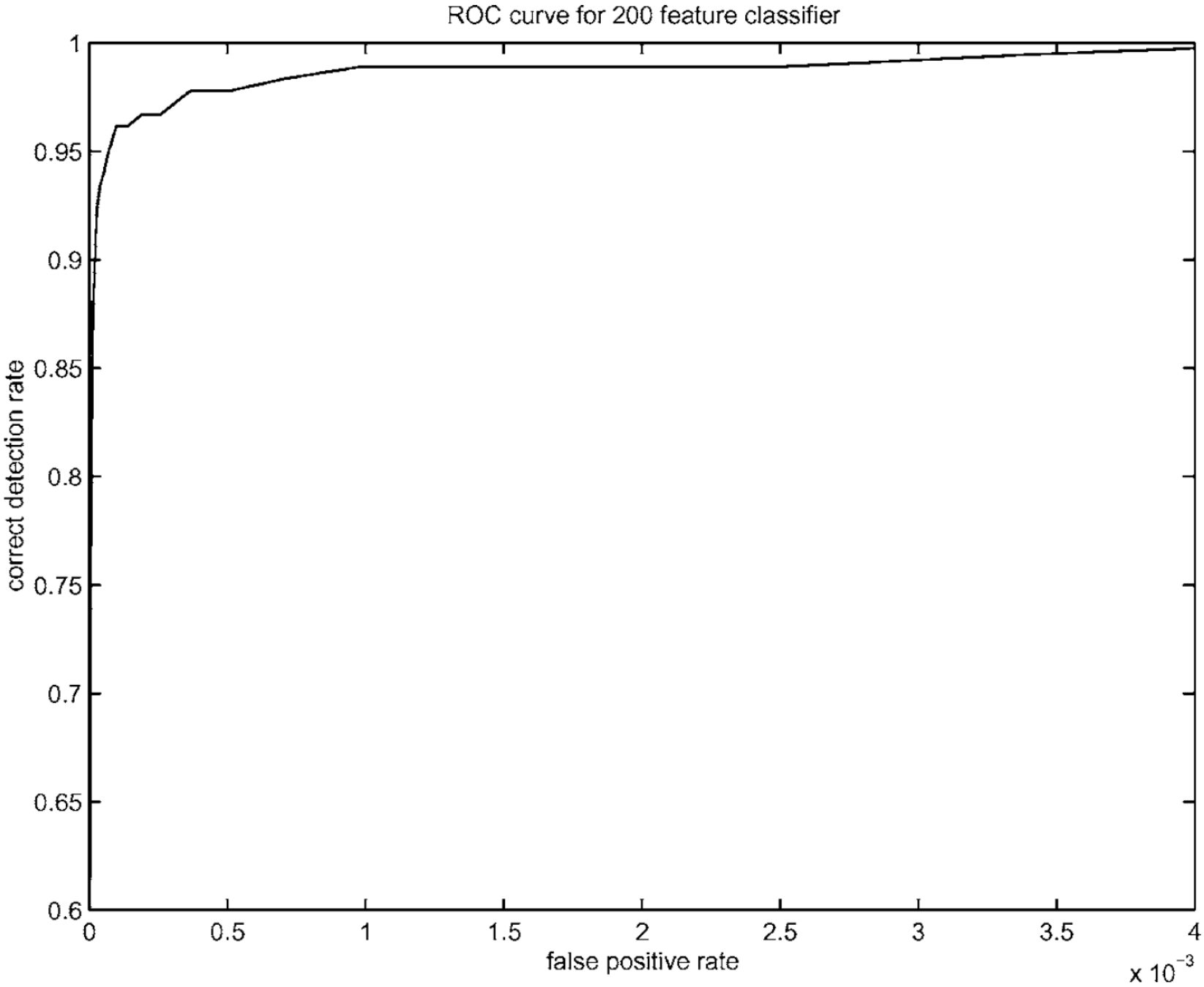
where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.

- The final strong classifier is:

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

where $\alpha_t = \ln \frac{1}{\beta_t}$

Note that only the top T features are used.


ROC curve for 200 feature classifier
correct detection rate
false positive rate
x 10^-3
1
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0.6
0
0.5
1
1.5
2
2.5
3
3.5
4

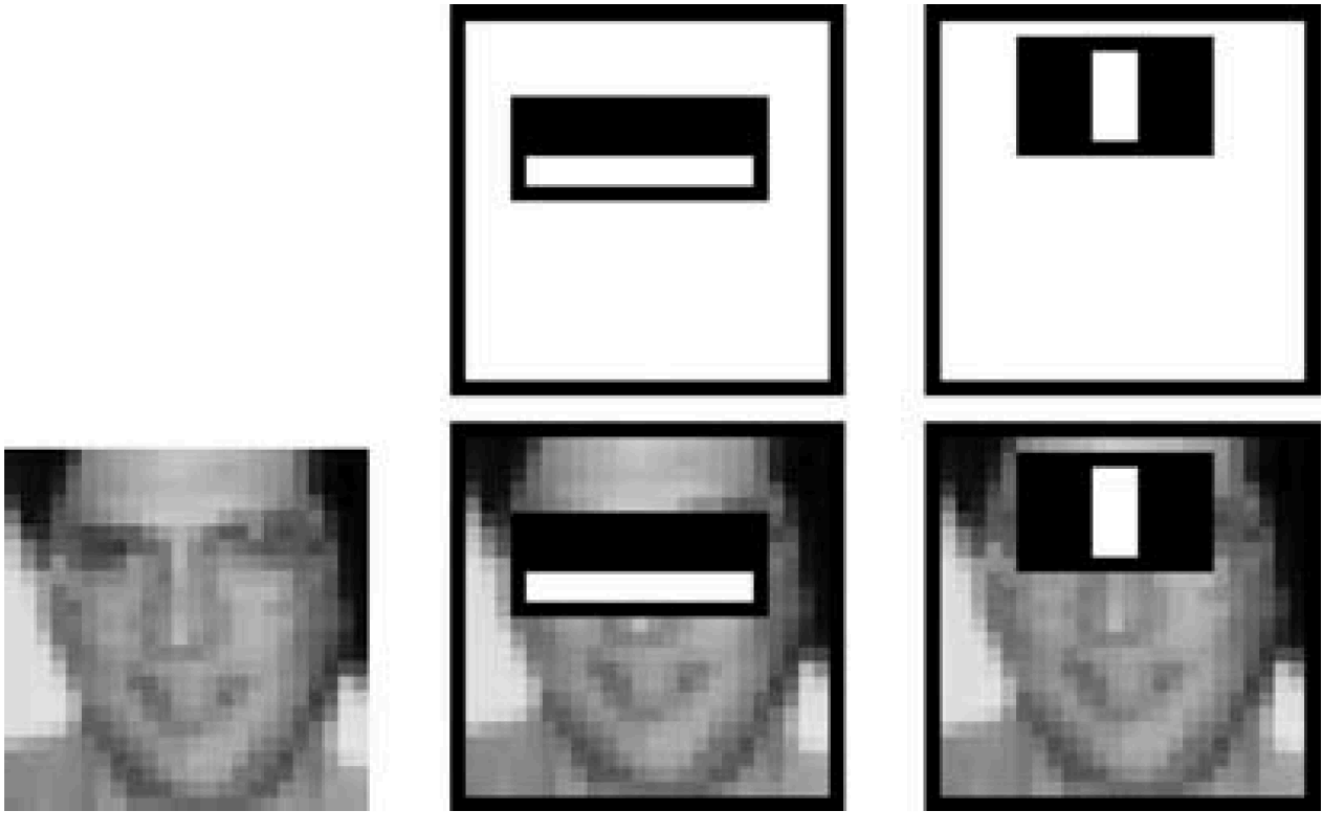
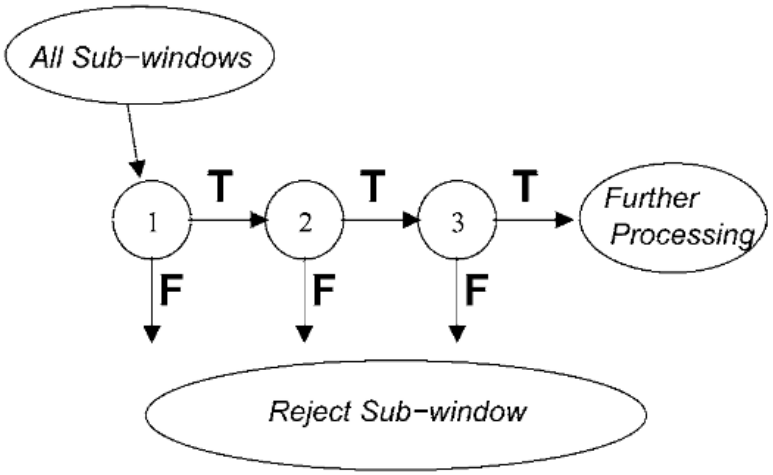


Figure 5. The first and second features selected by AdaBoost. The two features are shown in the top row and then overlayed on a typical training face in the bottom row. The first feature measures the difference in intensity between the region of the eyes and a region across the upper cheeks. The feature capitalizes on the observation that the eye region is often darker than the cheeks. The second feature compares the intensities in the eye regions to the intensity across the bridge of the nose.

Viola-Jones "attentional cascade" algorithm

- Increases performance while reducing computation time

- **Main idea:**
  - Vast majority of sub-windows are negative.
  - Simple classifiers used to reject majority of sub-windows before more complex classifiers are applied
  - See their paper for details



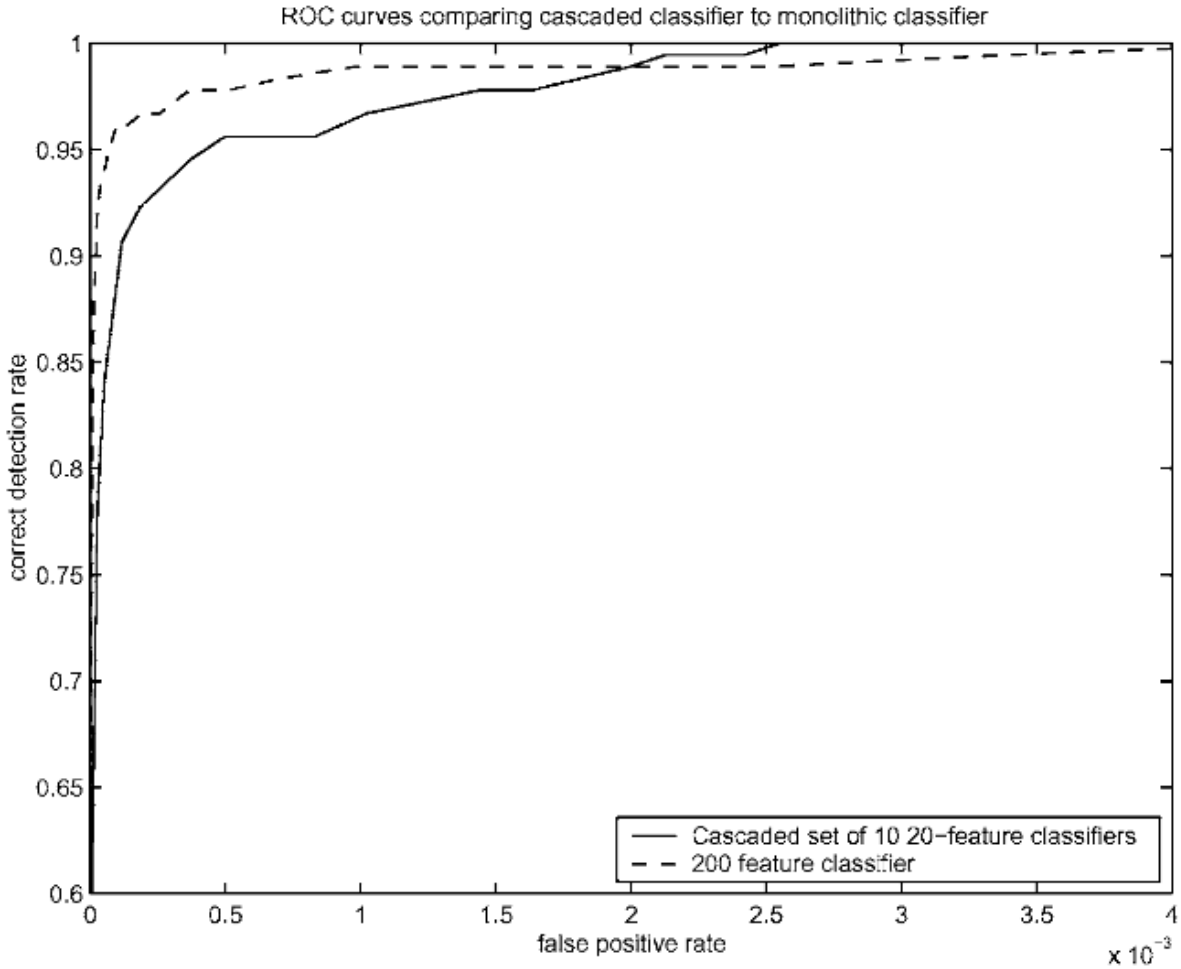
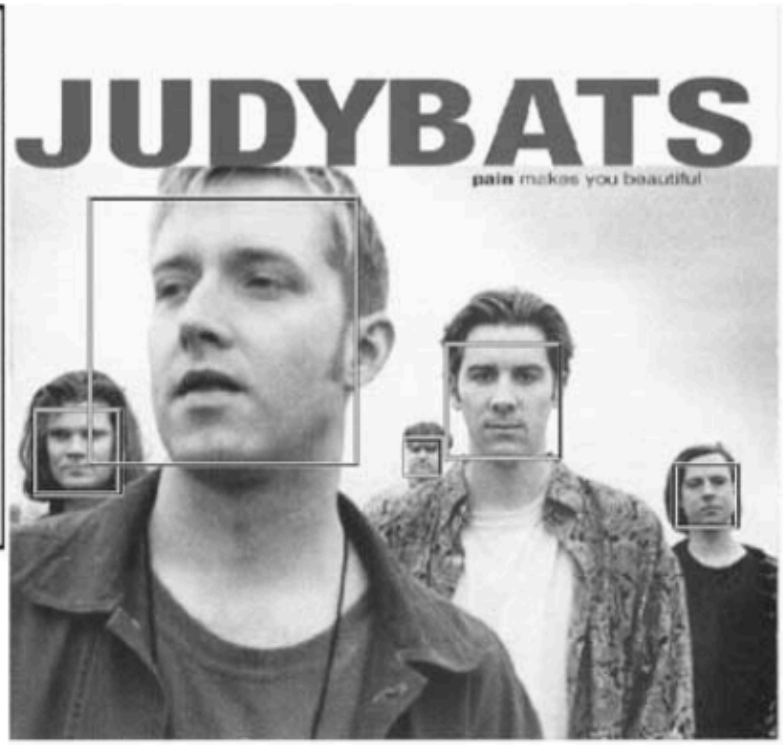
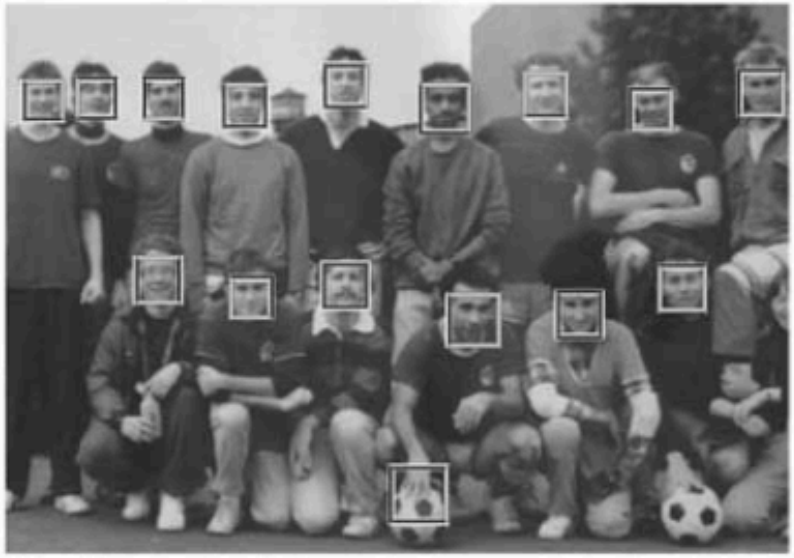


Figure 7. ROC curves comparing a 200-feature classifier with a cascaded classifier containing ten 20-feature classifiers. Accuracy is not significantly different, but the speed of the cascaded classifier is almost 10 times faster.


JUDYBATS
pain makes you beautiful



From Wikipedia

Advantages of Viola–Jones algorithm [edit]

- Extremely fast feature computation
- Efficient feature selection
- Scale and location invariant detector
- Instead of scaling the image itself (e.g. pyramid-filters), we scale the features.
- Such a generic detection scheme can be trained for detection of other types of objects (e.g. cars, hands)

Disadvantages of Viola–Jones algorithm [edit]

- Detector is most effective only on frontal images of faces
- It can hardly cope with 45° face rotation both around the vertical and horizontal axis.
- Sensitive to lighting conditions
- We might get multiple detections of the same face, due to overlapping sub-windows.

Paul Viola

2nd

VP Science, Prime Air at Amazon

Greater Seattle Area | Computer Software

Previous Highspot, Inc., Microsoft, Mitsubishi Electric Research Labs (MERL)

Education Massachusetts Institute of Technology

Connect | Send Paul InMail

500+ connections

Experience

VP Science, Prime Air

Amazon

July 2014 – Present (1 year 8 months) | Seattle, WA

Delivering packages in 30 mins.

Problems with Boosting

- Sensitive to noise, outliers: Boosting focuses on those examples that are hard to classify

- But this can be a way to identify outliers or noise in a dataset