Lecture 8

ROC curves

How to describe the performance of a binary classifier.

Boosting

Review the intuitively simple but powerful method of boosting for combining poorly performing classifiers into a single highly performing classifier. How to use this technique for feature selection. Also we will focus on the paper that is the basis for your lab assignment.

Quality of a classifier

Question:

How can we assess the quality of a learned classifier ??

Quality of a classifier

Question:

How can we assess the quality of a learned classifier ??

Answer:

On a test set count the number of times the classifier reports the wrong answer as opposed to the correct answer.

Other Issues:

How do I decide if one classifier is better than another ?

If my classifier relies on thresholding some output value then how do I choose the best threshold ?

ROC curves

First Use

During World War II for the analysis of radar signals. Following the attack on Pearl Harbor in 1941, the United States army began new research to increase the rate of correctly detected Japanese aircraft from their radar signals.

What is an ROC curve ?

A ROC curve displays the performance of a classifier by plotting its ability to discriminate between members of a class and those not belonging to the class.

Receiver Operator Characteristic

Originated from signal detection theory

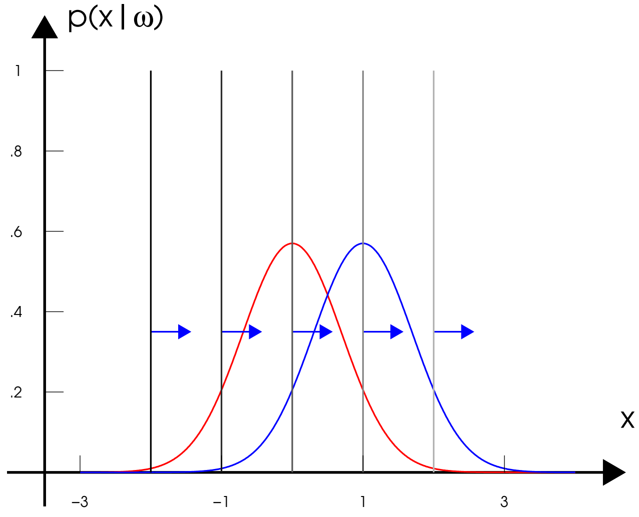
Have a binary signal corrupted by Gaussian noise

$$x_t = \begin{cases} 0 + u_t, & \text{signal absent} \\ 1 + u_t, & \text{signal present} \end{cases}$$

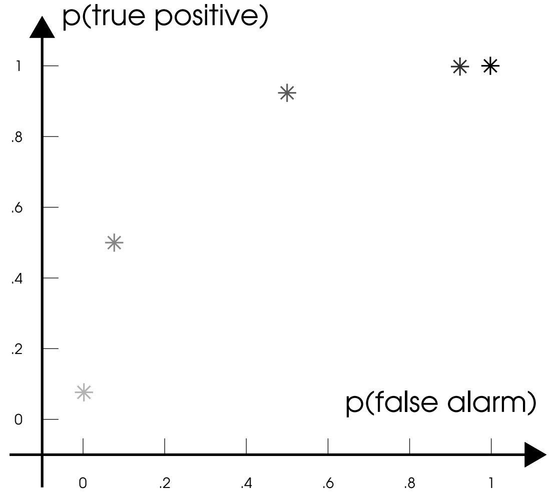
where $u_t \sim N(0, \sigma^2)$.

- How to set the **threshold** (operating point) to distinguish between **presence/absence** of signal ?
- Depends on

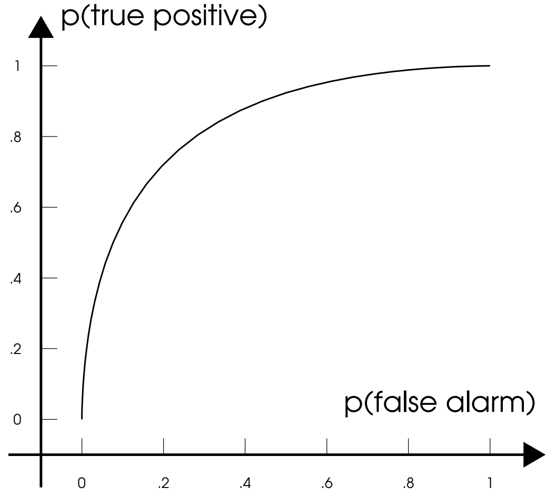
1. strength of signal,
2. noise variance, and
3. desired true positive rate (classify x_t as **signal** when **signal present**) or
4. false alarm rate (classify x_t as **signal** when **signal absent**)



Likelihoods and decision thresholds



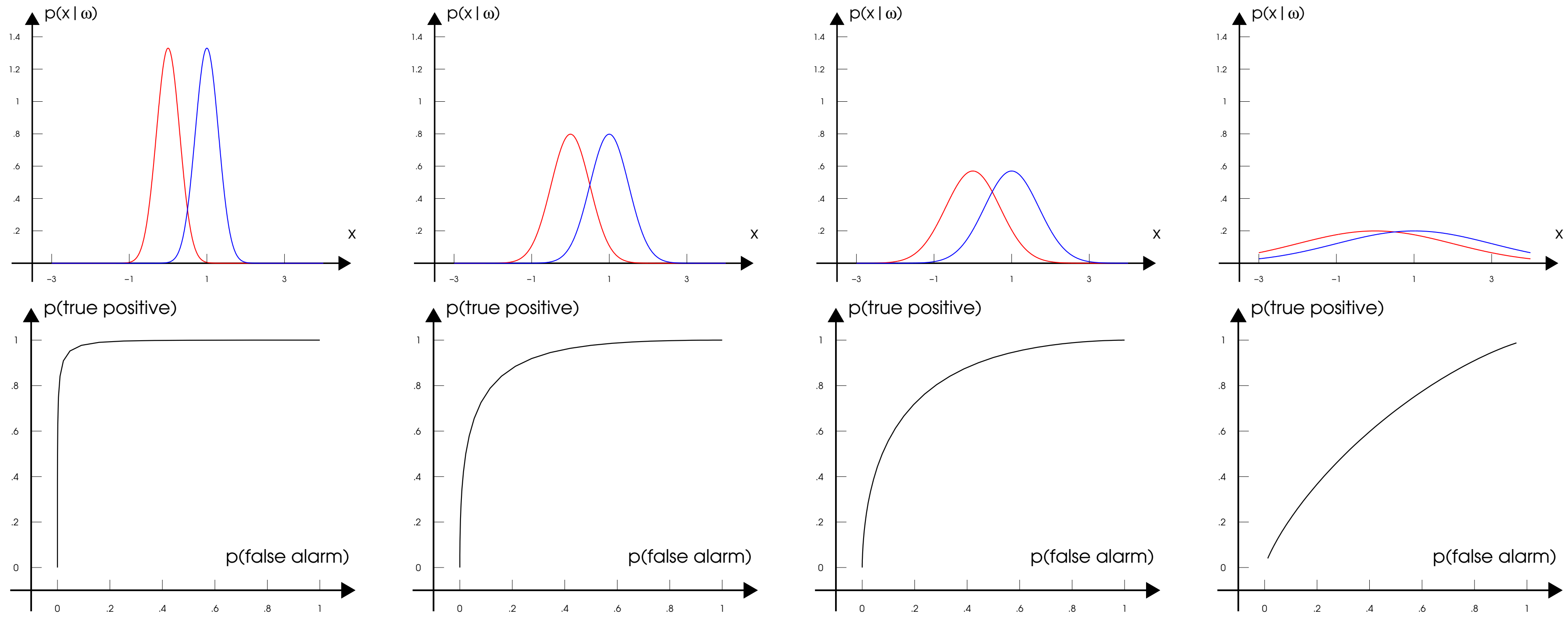
thresholds' classification rates



Full ROC curve

Blue curve signal is **present**. **Red curve** signal is **not present**.

Effect of changing σ

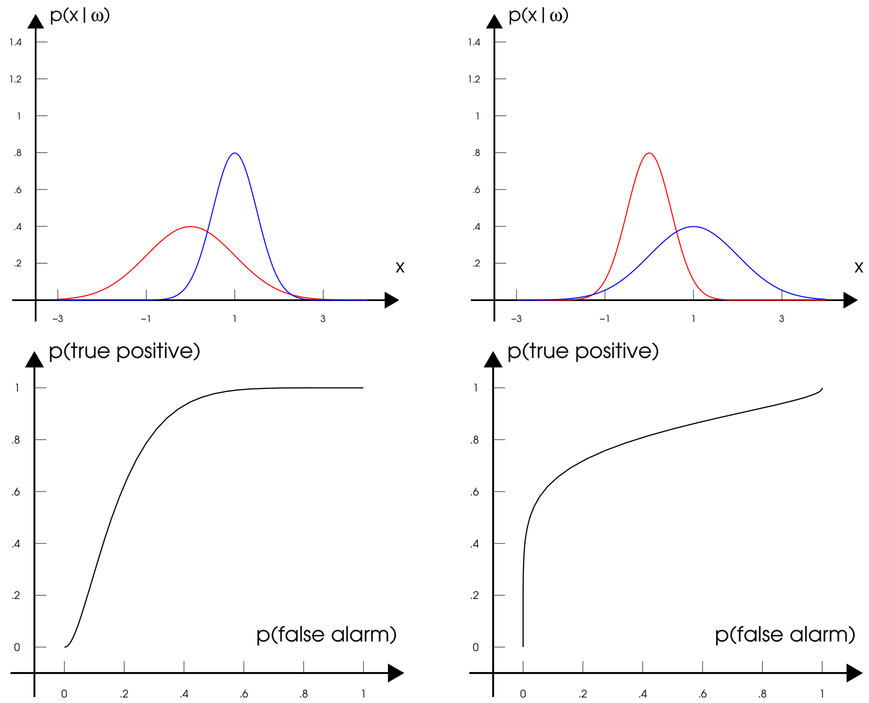


Blue curve signal is **present**. **Red curve** signal is **not present**.

Signal detection theory

Slope of ROC curve is equal to likelihood ratio: $L(x) = \frac{p(x|\text{signal})}{p(x|\text{noise})}$

if variances are equal $L(x)$ increases monotonically with x and ROC curve is convex, however, concavities occur with unequal variances



ROC analysis for classification

Based on contingency table or confusion matrix

	Predicted outcome	
Ground truth	positive	negative
positive	**True positive**	**False negative**
negative	**False positive**	**True negative**

Terminology:

- true positive = hit
- true negative = correct rejection
- false positive = false alarm (aka Type I error)
- false negative = miss (aka Type II error)
  - positive/negative refers to prediction; true/false refer to correctness

More terminology & notation

type of outcome	number
true positive	n_{tp}
false positive	n_{fp}
false negative	n_{fn}
true negative	n_{tn}
positive	$n_{\text{tp}} + n_{\text{fp}}$
negative	$n_{\text{fn}} + n_{\text{tn}}$

- **True positive rate**: fraction of positives correctly predicted

$$\text{tpr} = \frac{n_{\text{tp}}}{n_{\text{tp}} + n_{\text{fn}}}$$

- **False positive rate**: fraction of negatives incorrectly predicted

$$\text{fpr} = \frac{n_{\text{fp}}}{n_{\text{fp}} + n_{\text{tn}}}, \qquad \text{fpr} = 1 - \text{true negative rate}$$

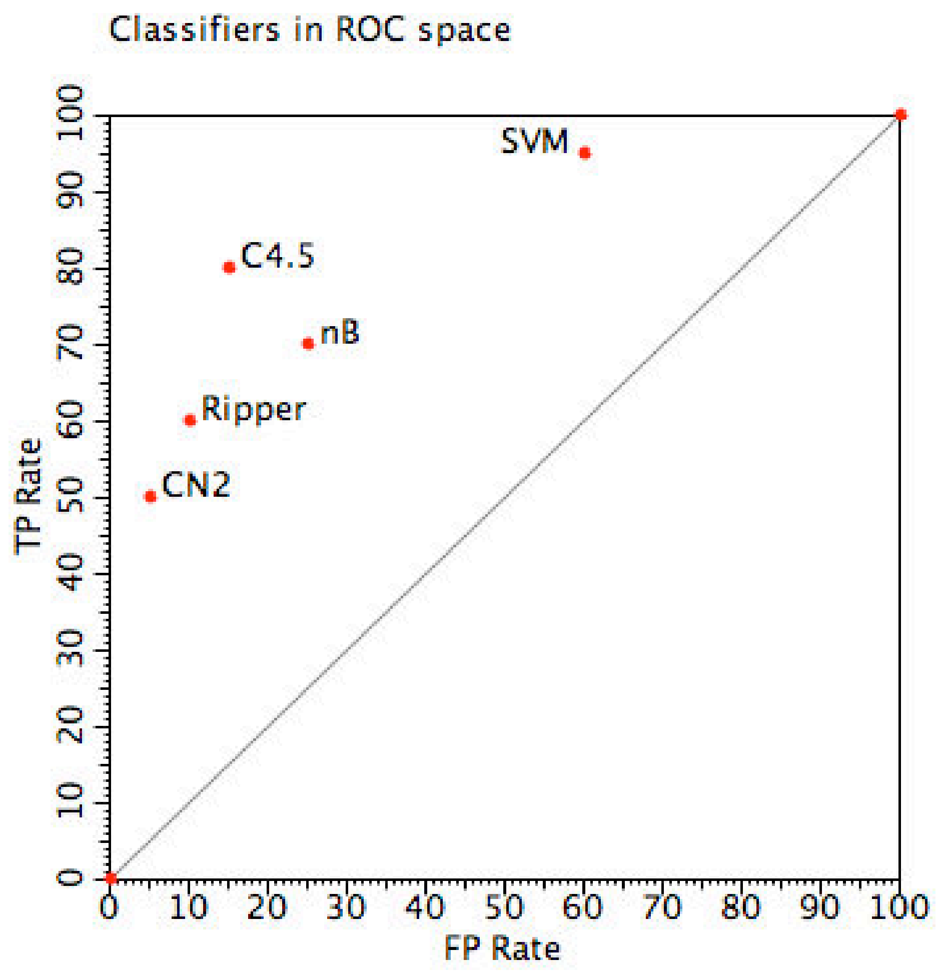
- **Accuracy**: is the proportion of correct classifications in the test set

$$\text{acc} = \frac{n_{\text{tp}} + n_{\text{tn}}}{n_{\text{tp}} + n_{\text{tn}} + n_{\text{fp}} + n_{\text{fn}}} = \text{tpr}\,\frac{n_{\text{tp}} + n_{\text{fn}}}{n} + (1 - \text{fpr})\frac{n_{\text{fp}} + n_{\text{tn}}}{n}$$

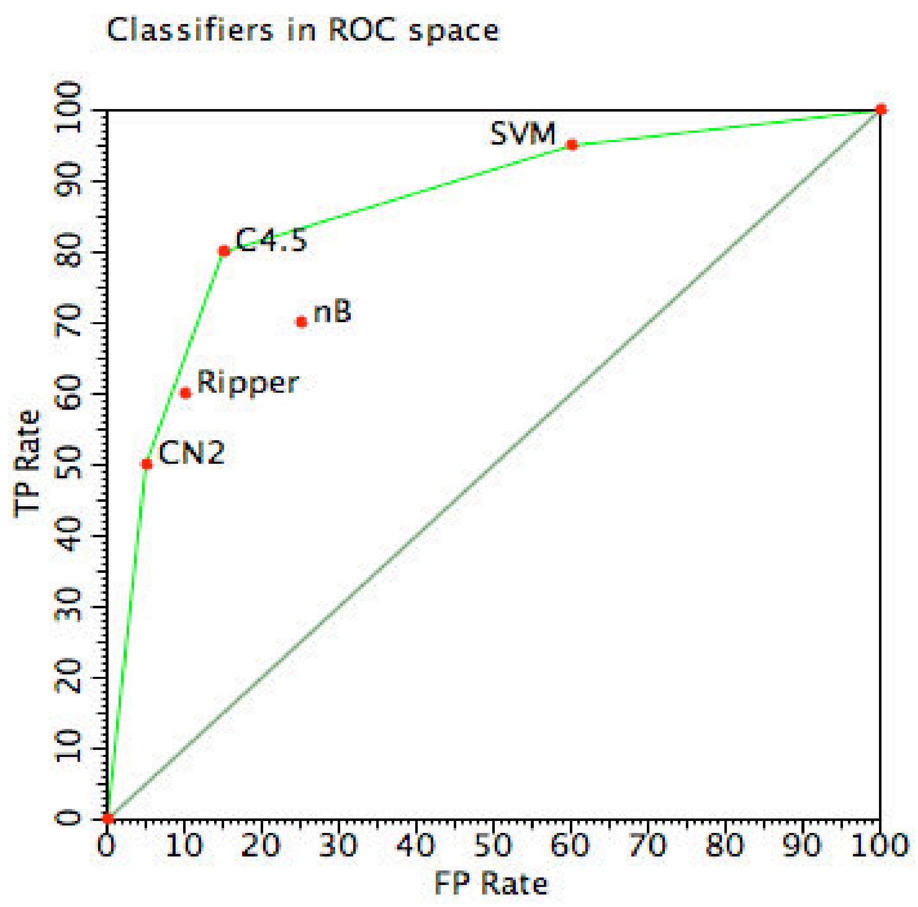
where $n = n_{\text{tp}} + n_{\text{tn}} + n_{\text{fp}} + n_{\text{fn}}$ is the total number of examples in the trial.

Then plot fpr Vs tpr

An example ROC plot



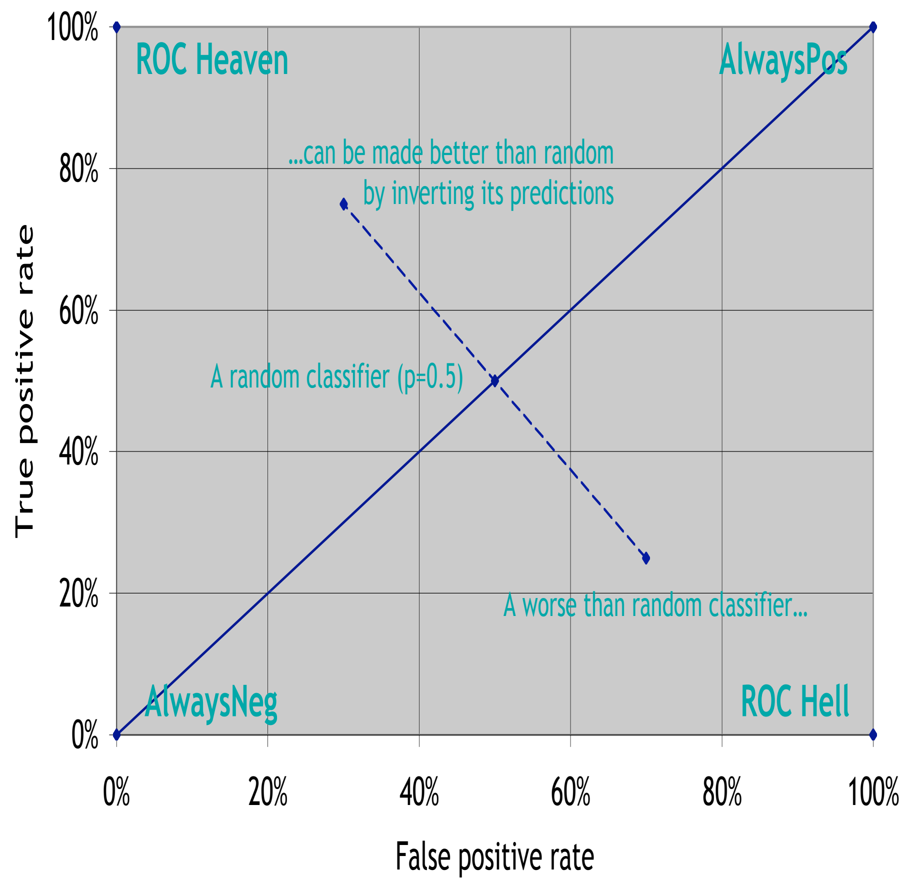
The ROC convex hull



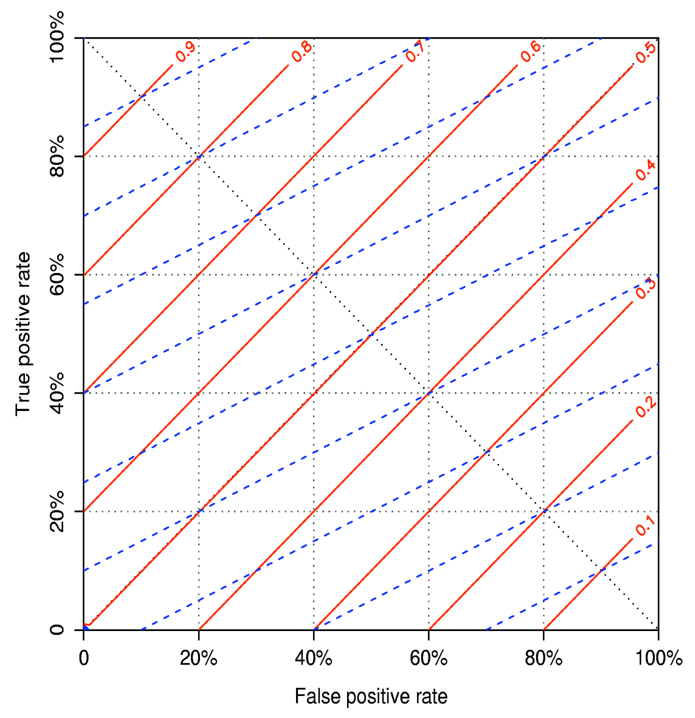
- Classifiers on the convex hull achieve the best accuracy for some ratio of positive examples to negative ones

- Classifiers below the convex hull are always sub-optimal

A closer look at ROC space



Iso-accuracy lines



- The accuracy can be written as

$$\text{acc} = \text{tpr}\,\frac{n_{\text{tp}} + n_{\text{fn}}}{n} + (1 - \text{fpr})\,\frac{n_{\text{fp}} + n_{\text{tn}}}{n}$$

- By re-arranging the above get

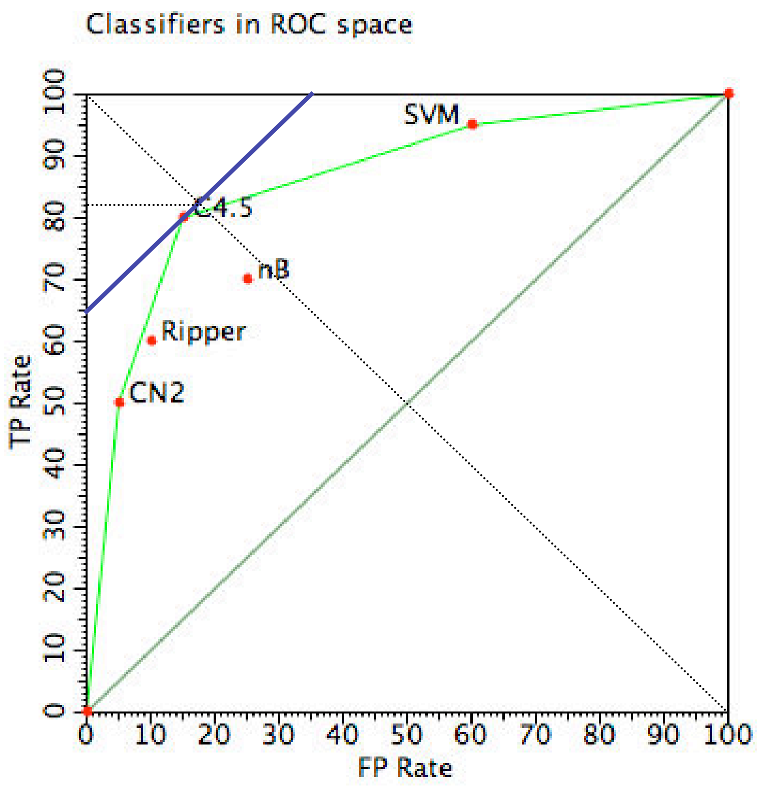
$$\text{tpr} = \text{fpr}\left(\frac{n_{\text{fp}} + n_{\text{tn}}}{n_{\text{tp}} + n_{\text{fn}}}\right) + \frac{1}{n_{\text{tp}} + n_{\text{fn}}}\left(\text{acc}\,n - (n_{\text{fp}} + n_{\text{tn}})\right)$$

- A line with slope $\frac{n_{\text{fp}}+n_{\text{tn}}}{n_{\text{tp}}+n_{\text{fn}}}$ in the (fpr, tpr) plane.
- All points on this line have accuracy acc.

Red lines (blue dashed lines) are the iso-accuracy lines for a certain ratio of negative to positive data (another ratio of negative to positive data).

Descending diagonal $\text{tpr} = -\text{fpr} + 1$, intersects the iso-accuracy line at $(\text{tpr}, \text{fpr}) = (\text{acc}, 1 - \text{acc})$.

Selecting the optimal classifier



If we suspect there are the same number of positive to negative examples

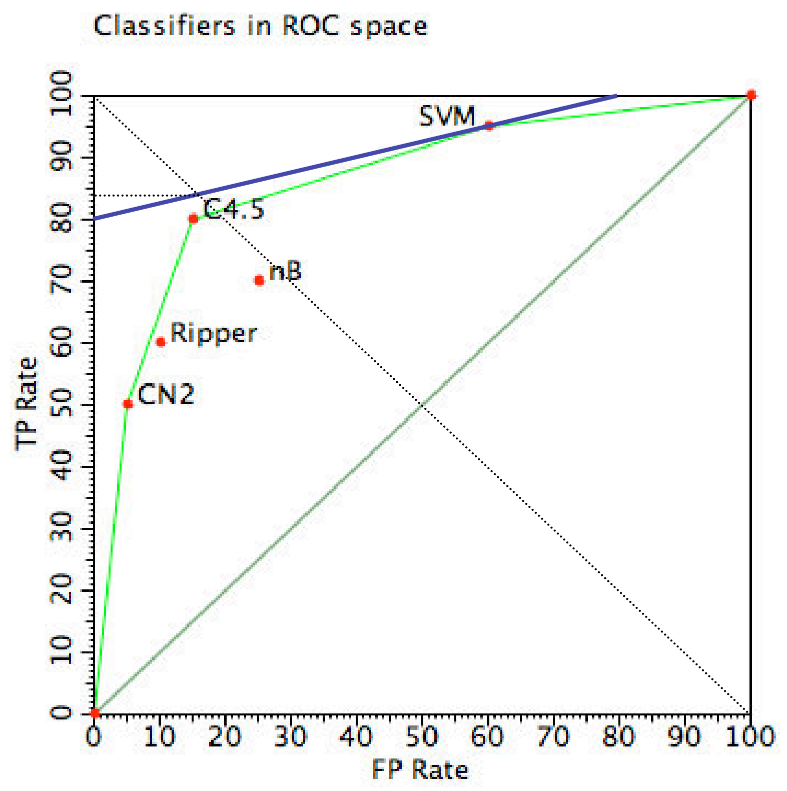
$$\implies \frac{n_{\mathsf{fp}} + n_{\mathsf{tn}}}{n_{\mathsf{tp}} + n_{\mathsf{fn}}} = 1.$$

Thus the iso-accuracy lines have slope 1 (blue line).

Where this line intersects the descending diagonal defines the accuracy of the classifier.

C4.5 is optimal and achieves about 82% accuracy

Selecting the optimal classifier



If we suspect there are 4 times of number of positive to negative examples

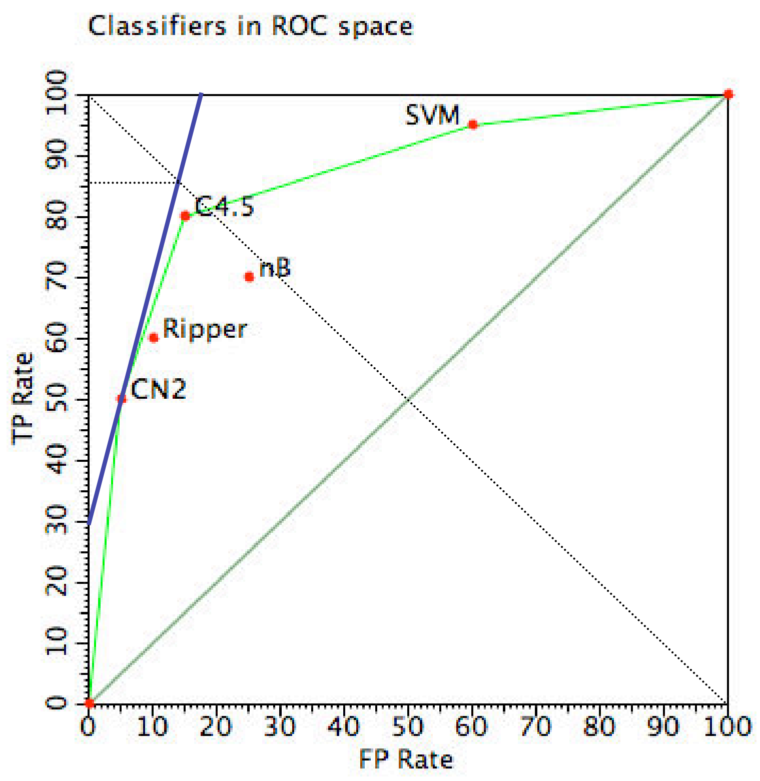
$$\implies \frac{n_{\text{fp}} + n_{\text{tn}}}{n_{\text{tp}} + n_{\text{fn}}} = \frac{1}{4}.$$

Thus the iso-accuracy lines have slope $\frac{1}{4}$ (blue line).

Where this line intersects the descending diagonal defines the accuracy of the classifier.

SVM is optimal and achieves about 84% accuracy

Selecting the optimal classifier



If we suspect there are 4 times of number of negative to positive examples

$$\implies \frac{n_{\mathsf{fp}} + n_{\mathsf{tn}}}{n_{\mathsf{tp}} + n_{\mathsf{fn}}} = 4.$$

Thus the iso-accuracy lines have slope 4 (blue line).

Where this line intersects the descending diagonal defines the accuracy of the classifier.

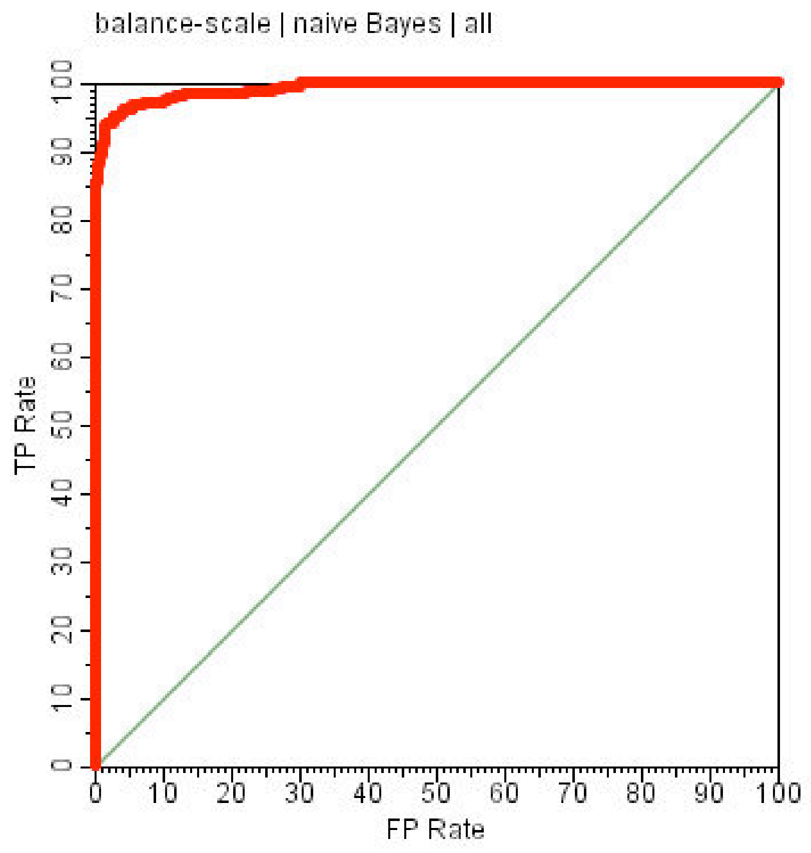
CN2 is optimal and achieves about 86% accuracy

Creating ROC curves

Assume you have a classifier that makes a decision by thresholding an output value. This type of classifier can create a ROC curve as follows:

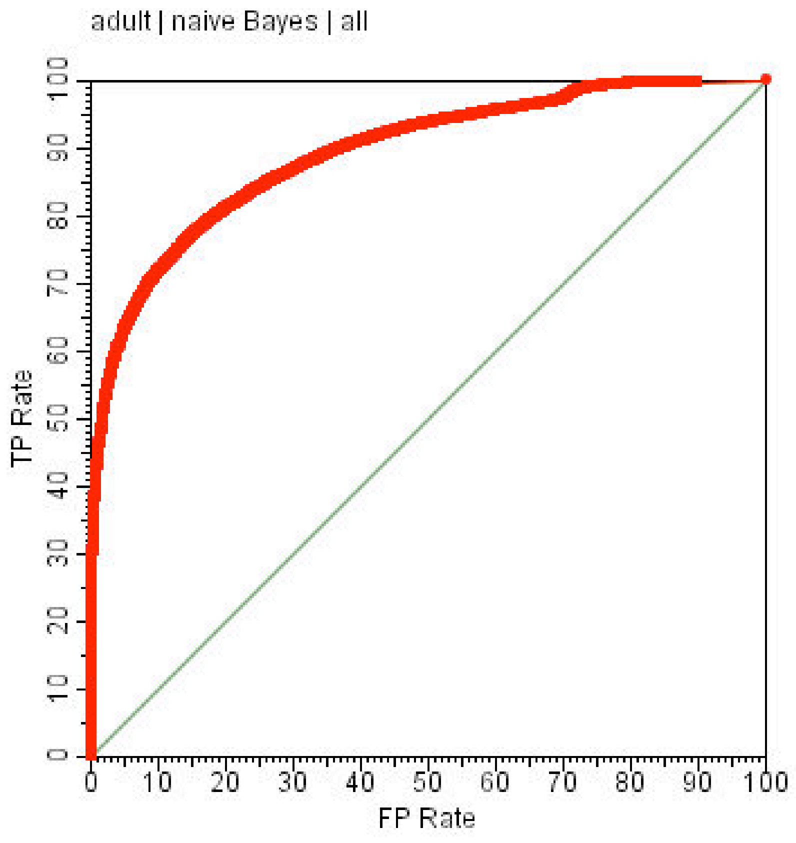
- Apply the classifier minus the final threshold function to the test data.
- Consider all possible thresholds
- Calculate the *false positive rate* and *true positive rate* at each value of the threshold.
- Plot *false positive rate* values against the *true positive rate* values.

Some example ROC curves



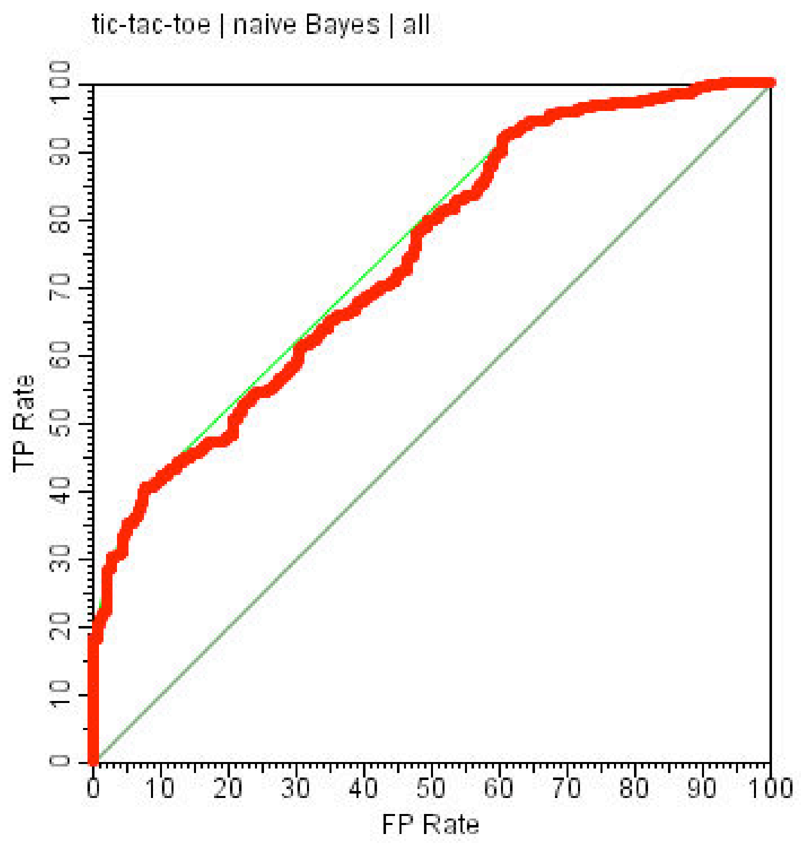
Good separation between classes, convex curve

Some example ROC curves



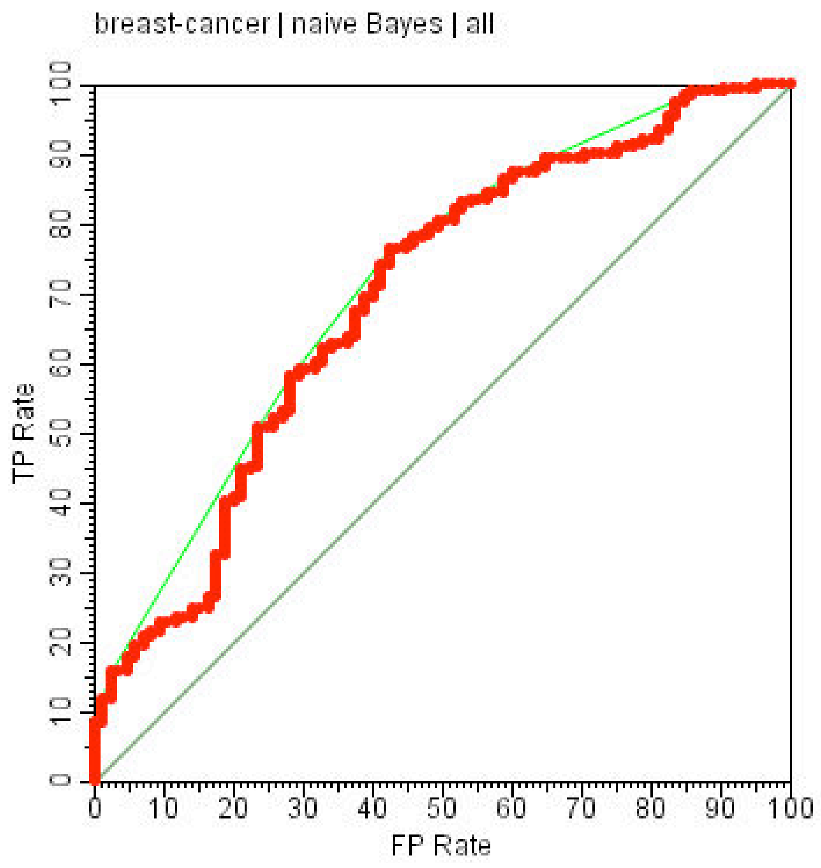
Reasonable separation, mostly convex

Some example ROC curves



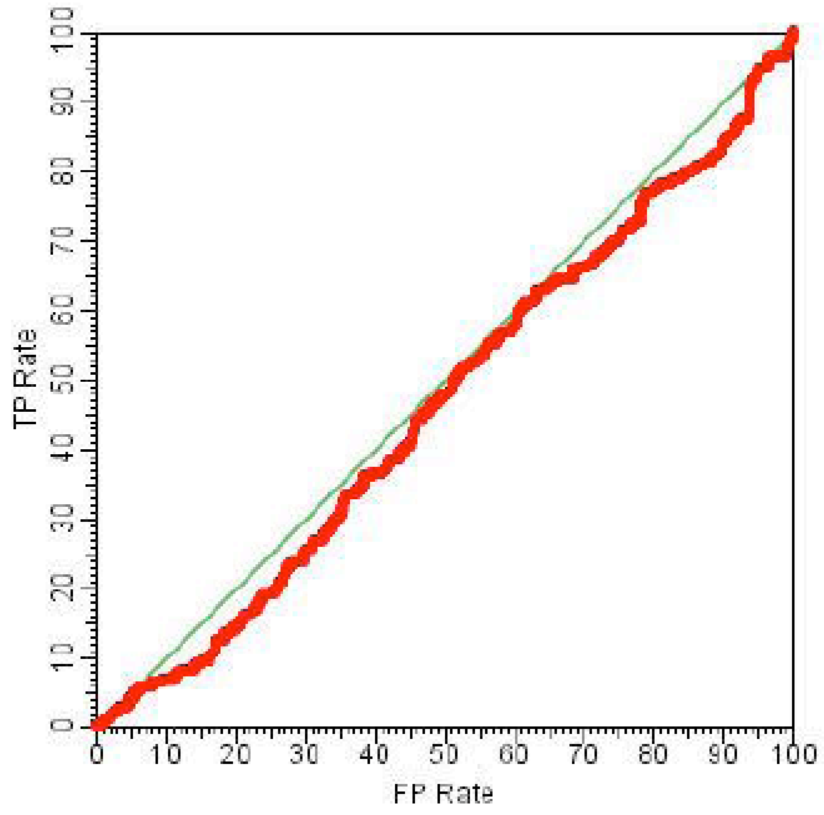
Fairly poor separation, mostly convex

Some example ROC curves



Poor separation, large and small concavities

Some example ROC curves



Random performance

ROC curves

- The curve visualises the quality of the classifier or probabilistic model on a test set without committing to a classification threshold

- The slope of the curve indicates class distribution in that segment of the classification

  diagonal segment $\Longrightarrow$ locally random behaviour

- Concavities indicate locally worse than random behaviour

The AUC metric

- The Area Under ROC Curve (AUC) assesses the classification in terms of separation of the classes.
  - all the positives before the negatives: $\mathbf{AUC = 1}$
  - random ordering: $\mathbf{AUC = 0.5}$
  - all the negatives before the positives: $\mathbf{AUC = 0}$

The AUC is equal to the **probability** that a classifier will **rank** a randomly chosen **positive instance higher** than a randomly chosen **negative one**.

Precision-recall curves

- **Precision**: fraction of positive predictions correct

$$\text{prec} = \frac{n_{\text{tp}}}{n_{\text{tp}} + n_{\text{fp}}}$$

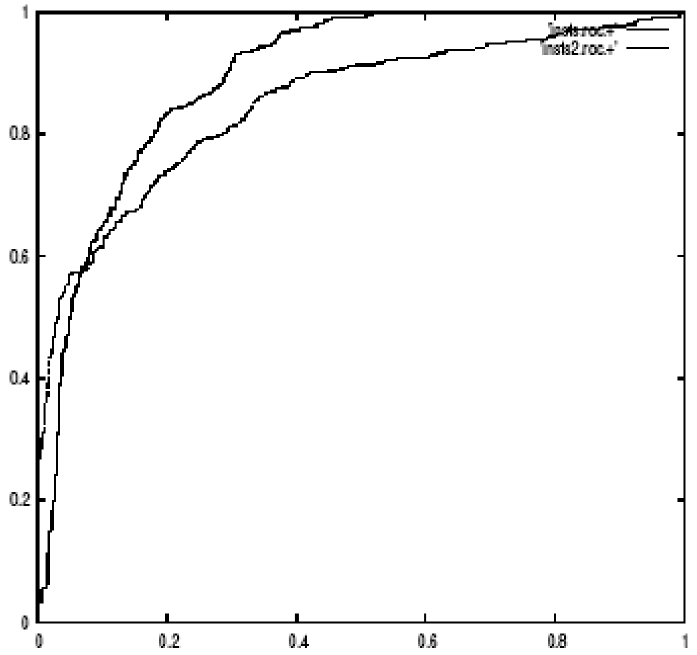
- **Recall**: fraction of positives correctly predicted

$$\text{rec} = \text{tpr} = \frac{n_{\text{tp}}}{n_{\text{tp}} + n_{\text{fn}}}$$

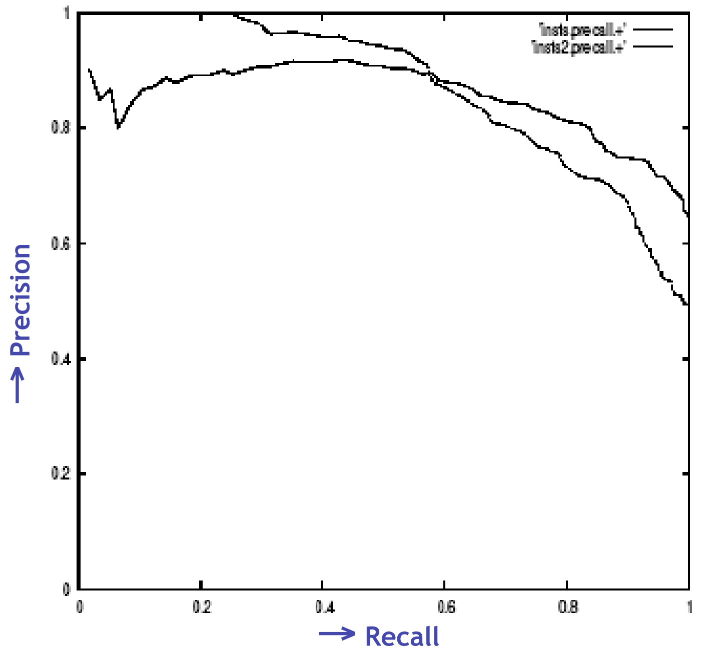
- **Note**: neither depends on true negatives

  makes sense in information retrieval, where true negatives tend to dominate $\implies$ low fpr easy

PR curves vs. ROC curves



Two ROC curves



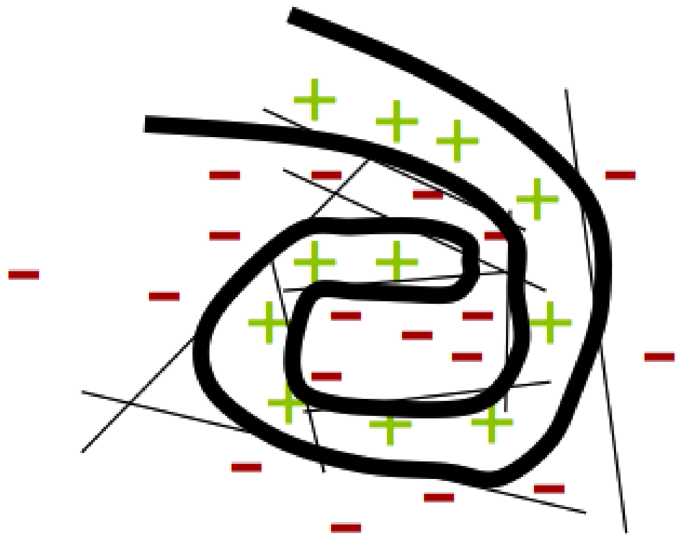
Corresponding PR curves

Ensemble Learning - Combining classifiers

Ensemble Learning

Want to

- train **and** combine weak classifiers



to construct a **strong classifier** more powerful than any of the individual ones.

When can we take this approach?

- First need the introduction of some terminology.
- The **Bias** of a classifier at x is

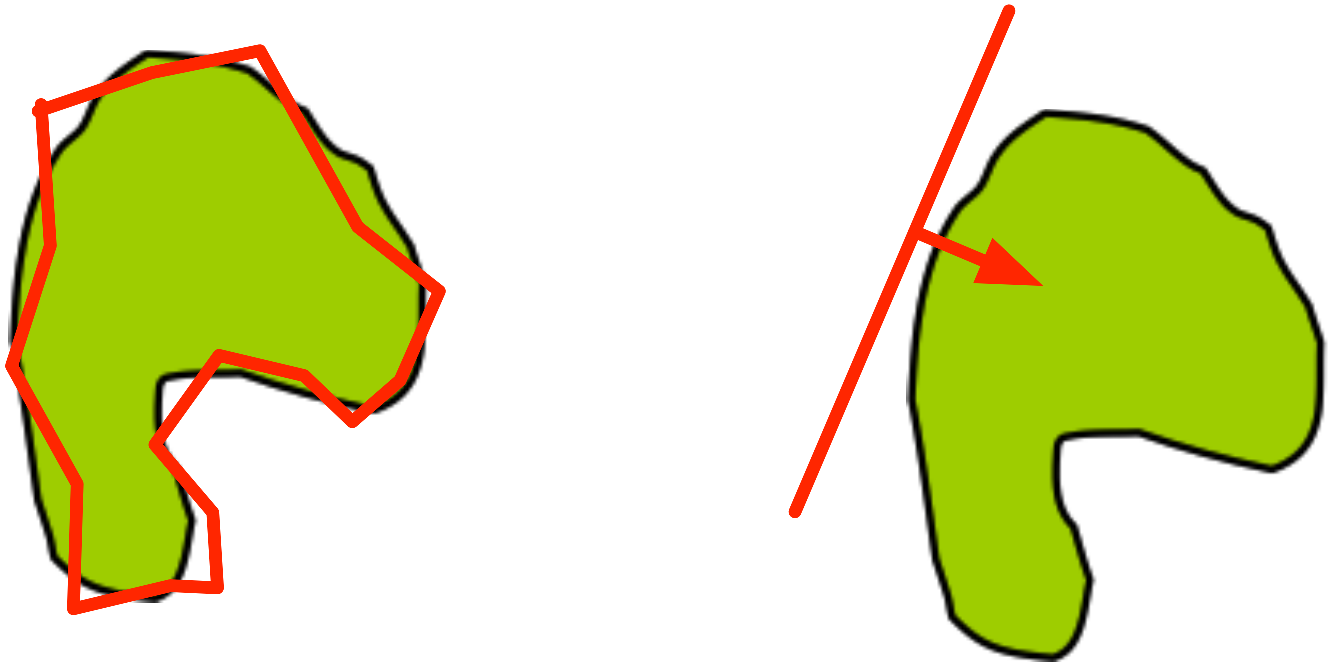
$$\mathrm{Bias}(x) = \mathrm{E}[\hat{f}(x)] - \mathrm{E}[f(x)]$$

  that is

- the discrepancy between the classifier's estimate at x averaged over different training sets and the true expected value of $f(x)$.

Characterization of a classifier: Bias

Green region is the true boundary.



Low-bias classifier **High-bias classifier**

High model complexity (large # of d.o.f.) $\Longrightarrow$ Low-bias

Low model complexity (small # of d.o.f.) $\Longrightarrow$ High-bias

Ensemble Prediction: Voting

A **diverse** and **complementary** set of high-bias classifiers, with performance better than chance, combined by **voting**

$$f_V(\mathbf{x}) = \operatorname{sign}\left(\sum_{t=1}^{T} h_t(\mathbf{x})\right)$$

can produce a classifier with a low-bias.

Example: Voting of oriented hyper-planes can define convex regions.

Ensemble Learning & Prediction

But how can we

- define a set of **diverse** and **complementary** high-bias classifiers, with non-random performance ?
- combine this set of high-biased classifiers to produce a low-bias classifier able to model a complex boundary (superior to voting)?

Ensemble Learning & Prediction

Exploit Labelled Training data

- Train different classifiers which focus on different subsets of the training data.
- Use a weighted sum of these $diversely$ trained classifiers.

This approach allows simple high-bias classifiers to be combined to model very complex boundaries.

Boosting

Boosting basics

Goal Want to build a (*strong*) classifier for a two class problem from a set of labeled training data and a set of poorly performing but computationally efficient classifiers. This final classifier should be easy to train, have high recognition rates and good generalization properties.

Training Input A set of n i.i.d. training examples and their labels:

$$\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}, \quad \text{with each } y_j \in \{-1, 1\}$$

A base class, $\mathcal{H}$, of simple classifiers. Where for each $h_t \in \mathcal{H}$ $h_t(\mathbf{x}) \in \{-1, +1\}$.

Boosting Output A classifier of the form

$$\phi(\mathbf{x}) = \operatorname{sgn}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right), \quad \text{where } \operatorname{sgn}(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases}$$

Boosting basics: Weak classifier

Data Weights Have training data $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$. Define an n dimensional weight vector $\mathbf{D} = (D_1, \ldots, D_n)$ with each $D_i \geq 0$ and $\sum_i D_i = 1$. These weights represent the cost of misclassifying a training example.

Error Measure Define the error of the classifier wrt $\mathbf{D}$ as

$$\epsilon(\mathbf{D}, h_t) = \sum_{i=1}^{n} D_i \,\mathrm{Ind}(h_t(\mathbf{x}_i) \neq y_i), \quad \mathrm{Ind}(x) = \begin{cases} 1 & \text{if } x \text{ is true} \\ 0 & \text{if } x \text{ is false} \end{cases}$$

Weak Classifier For every $\mathbf{D}$ a weak classifier must satisfy

$$\epsilon(\mathbf{D}, h_t) \leq \frac{1}{2} - \gamma, \quad \gamma > 0$$

An example weak classifier

$$h_t(\mathbf{x}) = \begin{cases} 1 & \text{if } f_t(\mathbf{x}) > \theta_t \\ -1 & \text{otherwise} \end{cases}$$

where for example

- $f_t(\mathbf{x}) = x_t$ the tth component of the vector $\mathbf{x}$ or
- $f_t(\mathbf{x}) = \mathbf{w}_t^T \mathbf{x} \implies h_t(\cdot)$ is a linear classifier.
- $\cdots$

Boosting basics: The idea

Idea outline Train a sequence of weak classifiers on modified data weights, and form a weighted average.

Algorithm Overview

- Each weak learner attempts to learn patterns which were difficult for previous classifier.
- This takes place by increasing the weight of difficult patterns.
- Final classifier formed by weighting weak learners according to their performance.

The Algorithm (AdaBoost)

1. Initialize: $\mathbf{D}^{(1)} = \{\frac{1}{n}, \cdots, \frac{1}{n}\}$, $t = 1$.

2. **while** $t \leq T$

   - Construct the weak classifier: Given the misclassification costs $\mathbf{D}$ and $\mathcal{H}$, find the best $h_t(\cdot)$ with respect to minimizing error ϵ_t.

   - Compute error and strong classifier weights

$$\epsilon_t = \sum_{i=1}^{n} D_i^{(t)} \operatorname{Ind}(y_i \neq h_t(\mathbf{x}_i)), \qquad \alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$

   - Update data weights

$$D_i^{(t+1)} = \frac{D_i^{(t)} \exp\left(-\alpha_t y_i h_t(\mathbf{x}_i)\right)}{Z_t}, \quad Z_t = \sum_{i=1}^{n} D_i^{(t)} \exp\left(-\alpha_t y_i h_t(\mathbf{x}_i)\right)$$

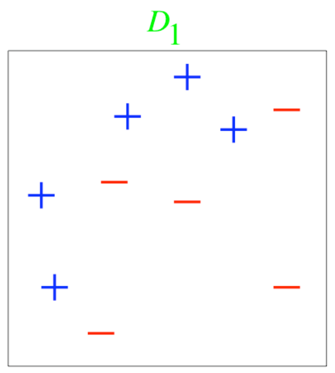
- If $\epsilon_t < 1/2$ set $t \leftarrow t + 1$ and go to 2; Else exit

3. Final classifier

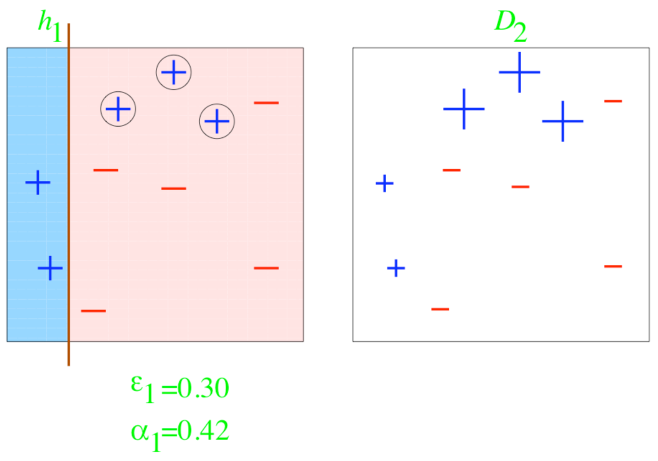
$$\phi(\mathbf{x}) = \operatorname{sgn}\left(\frac{\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})}{\sum_{t=1}^{T} \alpha_t}\right)$$

AdaBoost in action

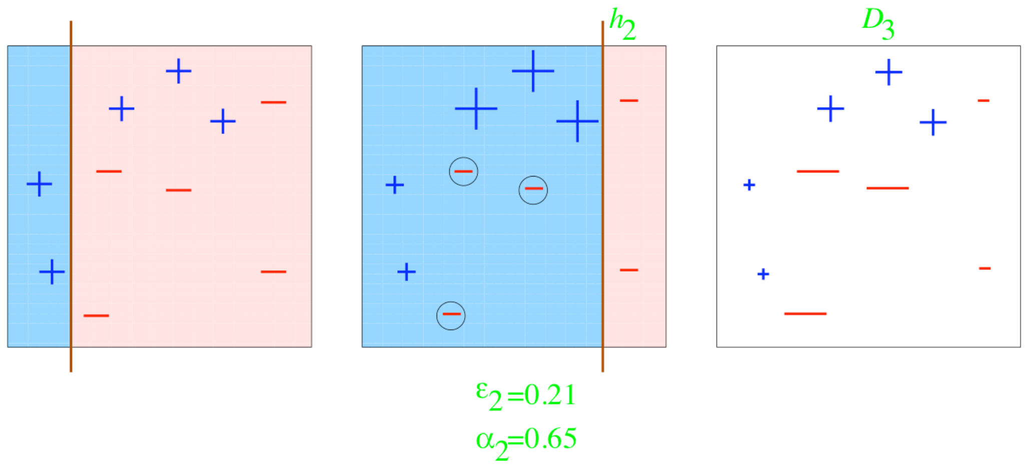
Toy Problem Build a strong classifier from the two-class training data in the figure below. Our weak classifiers are vertical and horizontal half-planes.



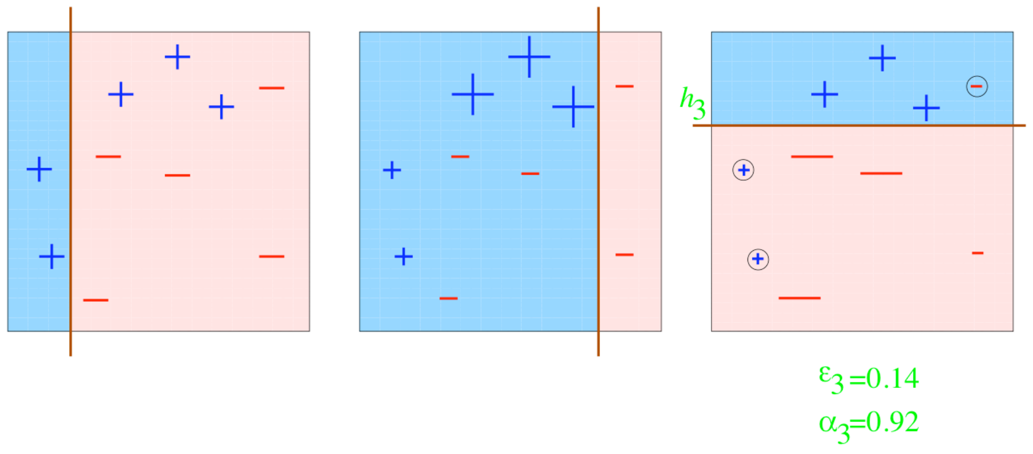
Round 1



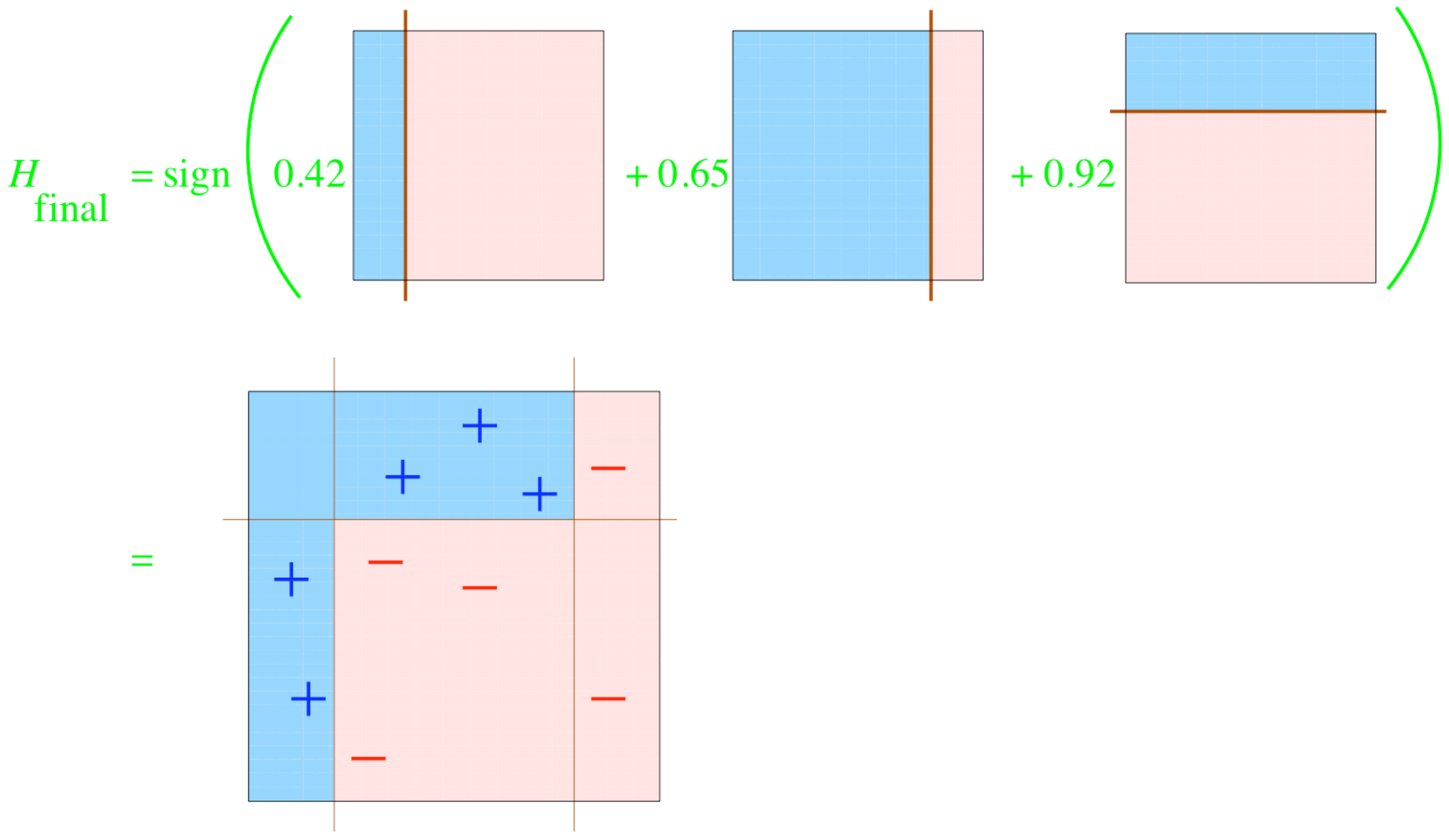
Round 2



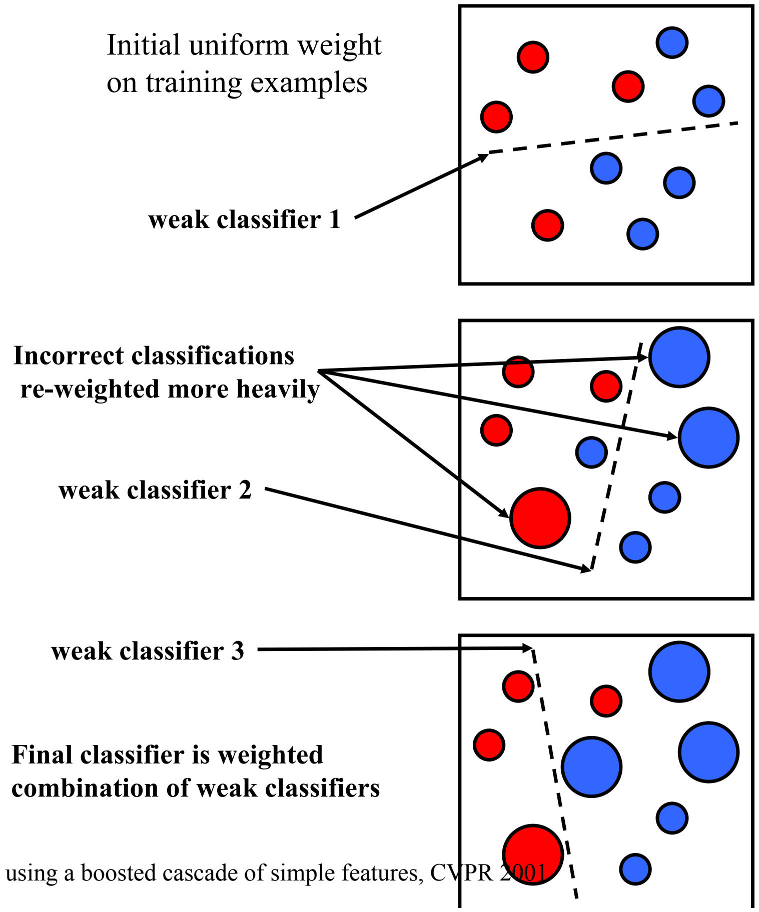
Round 3



Final Classifier

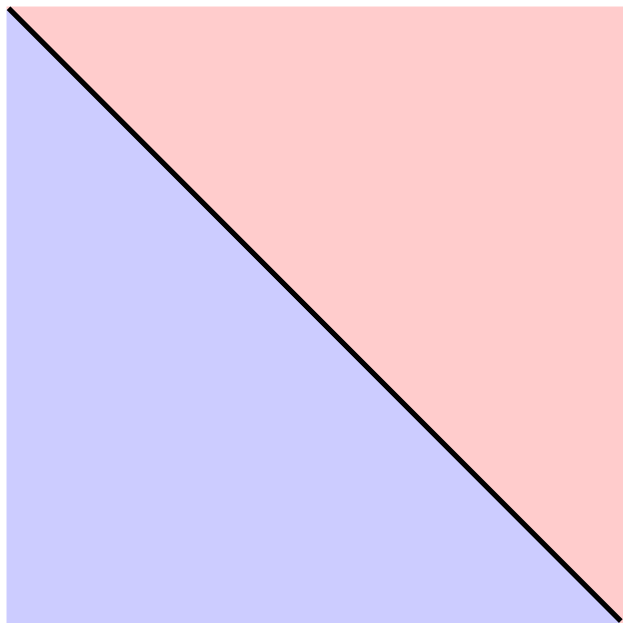


What's going on

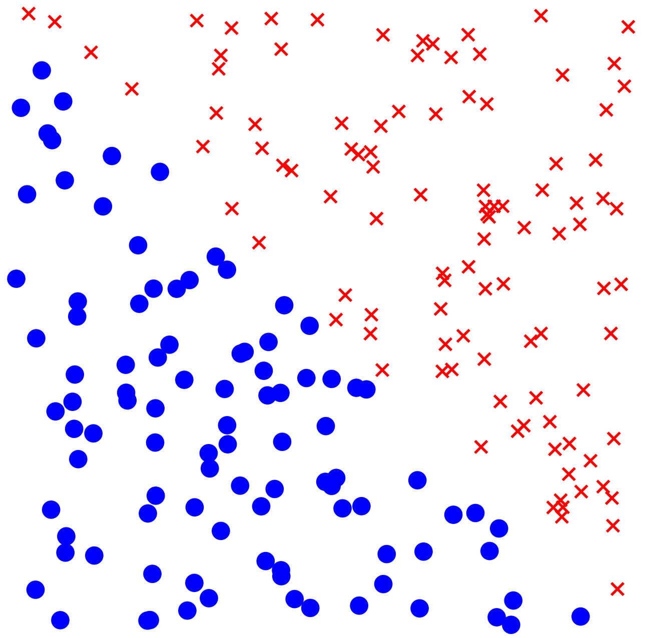


using a boosted cascade of simple features, CVPR 2001

Binary classification example



True decision boundary

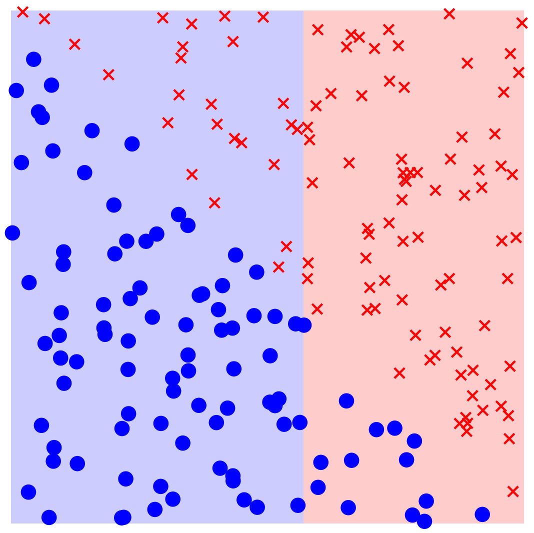


Training data

$\mathcal{H}$ is the set of all possible oriented vertical and horizontal lines.

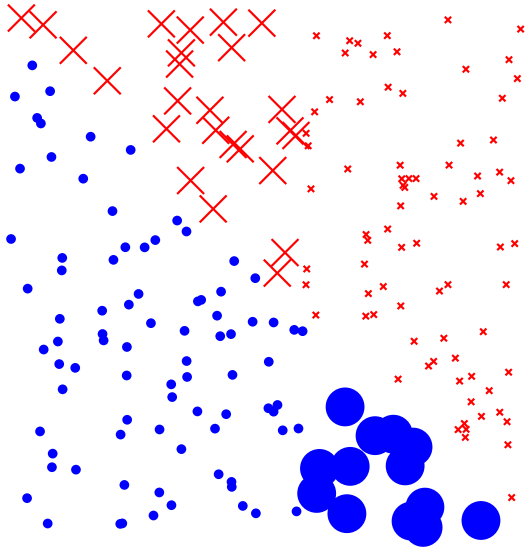
Example

Round 1



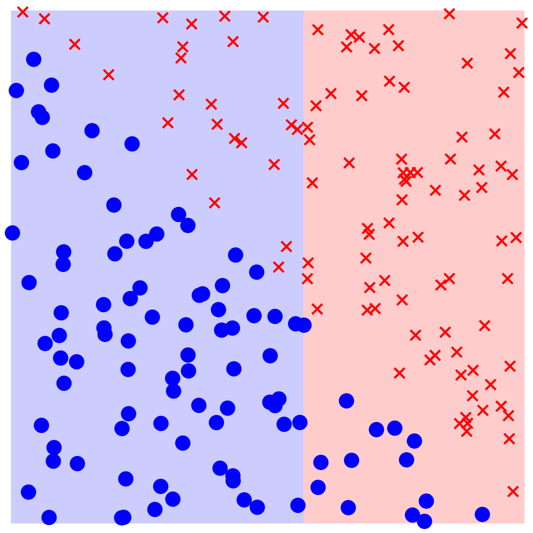
Chosen weak classifier

$\epsilon_1 = 0.19,\ \alpha_1 = 1.45$



Re-weight training points

$w_i^{(2)}$**'s**

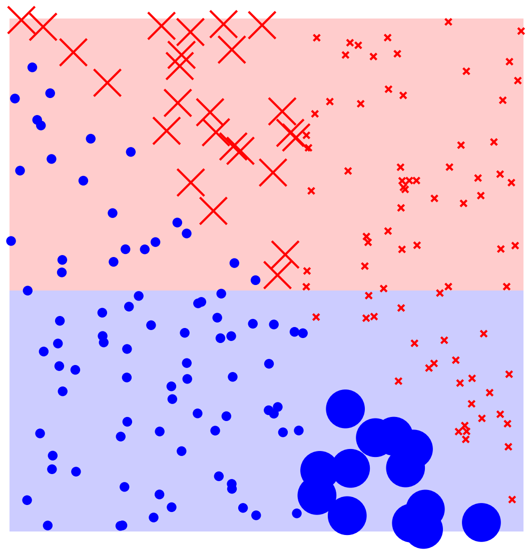


Current strong classifier

$f_2(\mathbf{x})$

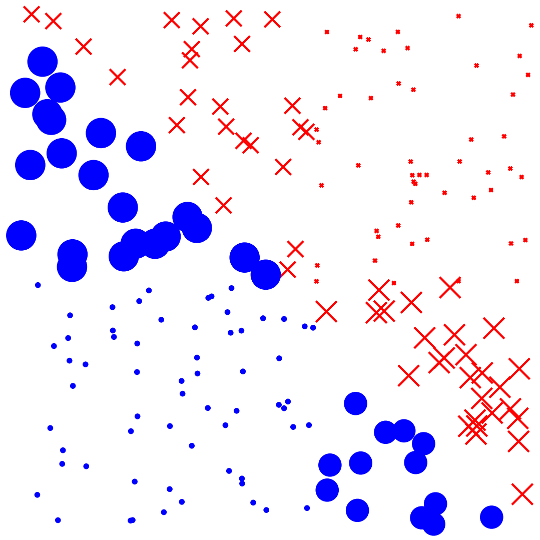
Example

Round 2



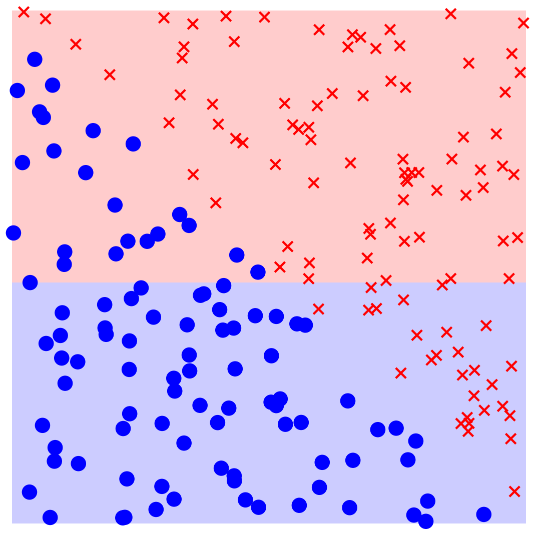
Chosen weak classifier

$\epsilon_2 = 0.1512, \ \alpha_2 = 1.725$



Re-weight training points

$w_i^{(3)}$**'s**

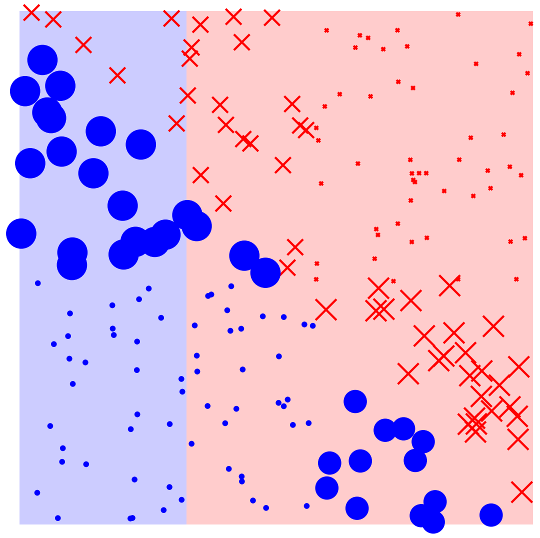


Current strong classifier

$f_2(\mathbf{x})$

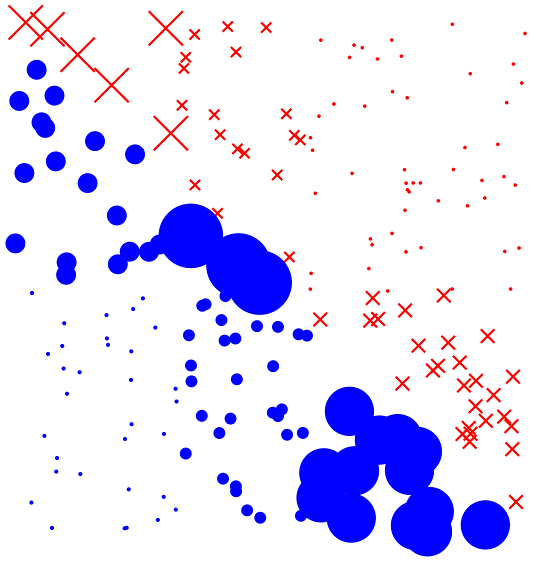
Example

Round 3



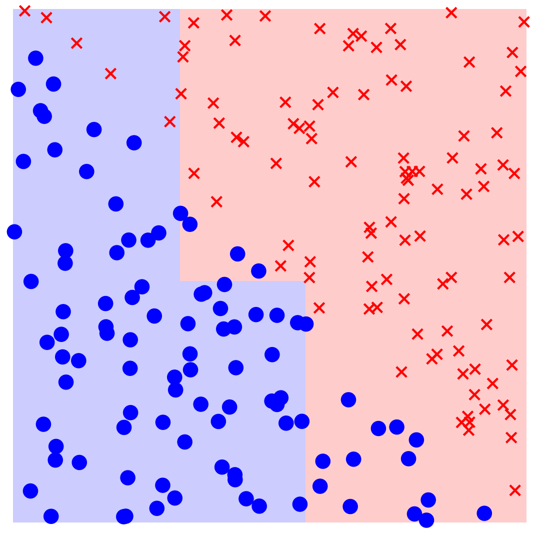
Chosen weak classifier

$\epsilon_3 = 0.2324, \; \alpha_3 = 1.1946$



Re-weight training points

$w_i^{(4)}$**'s**

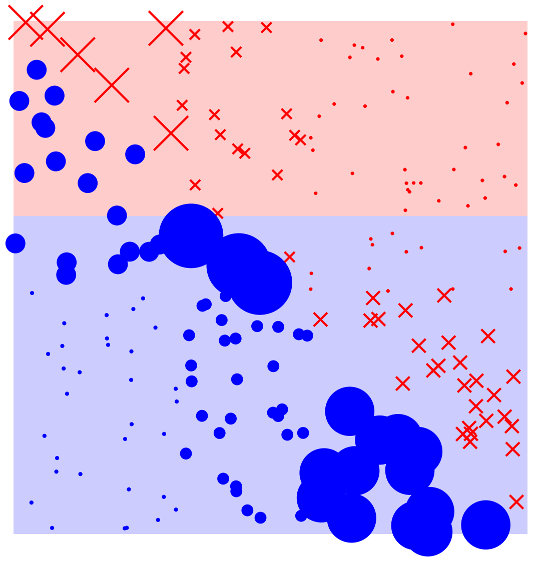


Current strong classifier

$f_3(\mathbf{x})$

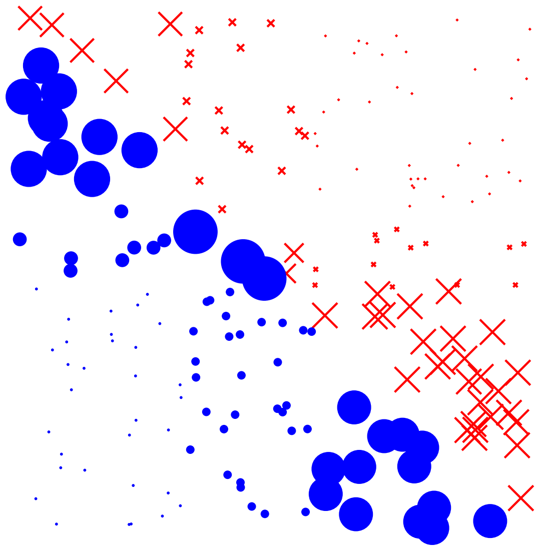
Example

Round 4



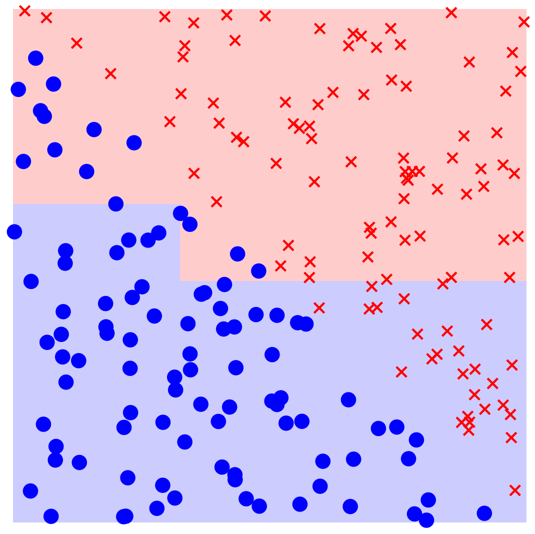
Chosen weak classifier

$\epsilon_4 = 0.2714, \; \alpha_4 = 0.9874$



Re-weight training points

$w_i^{(5)}$**'s**

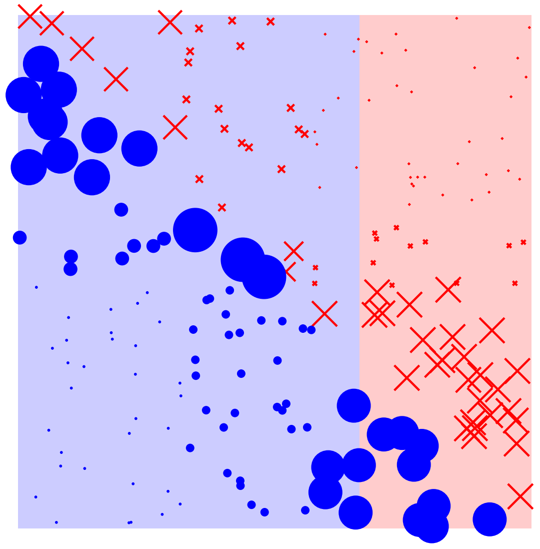


Current strong classifier

$f_4(\mathbf{x})$

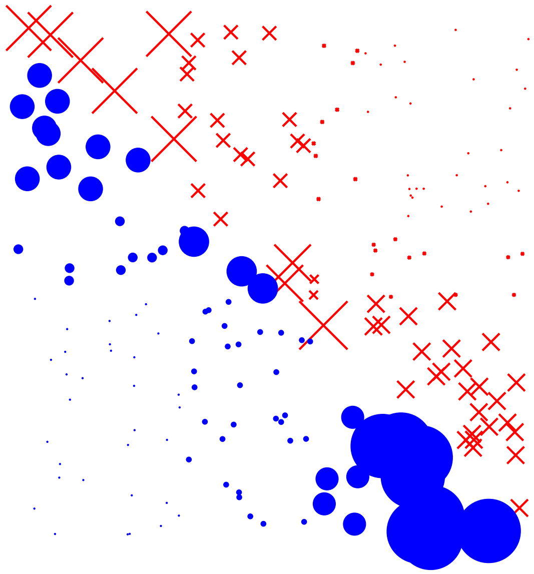
Example

Round 5



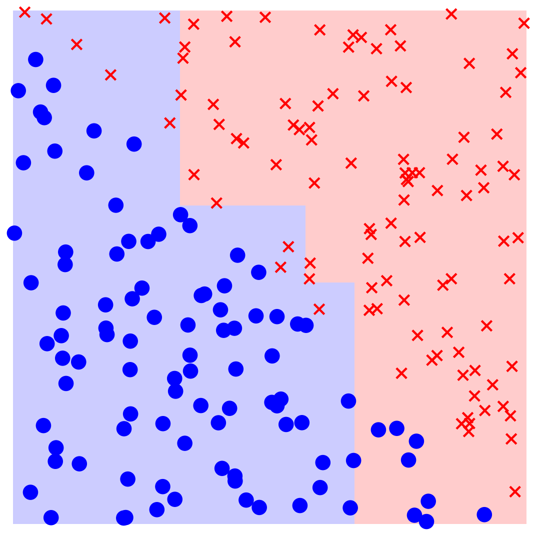
Chosen weak classifier

$\epsilon_5 = 0.2616, \ \alpha_5 = 1.0375$



Re-weight training points

$w_i^{(6)}$**'s**

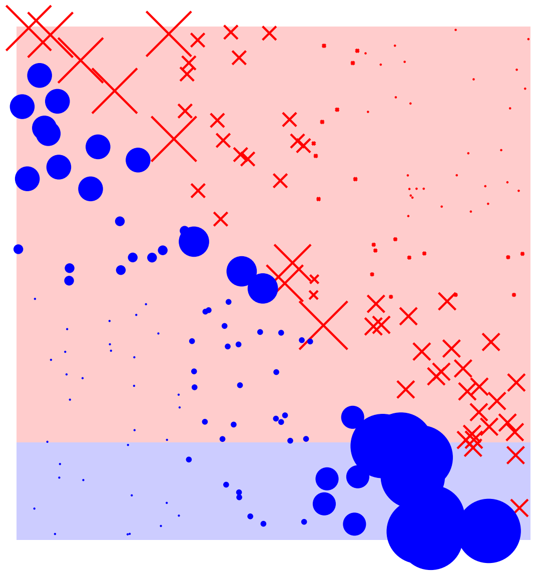


Current strong classifier

$f_5(\mathbf{x})$

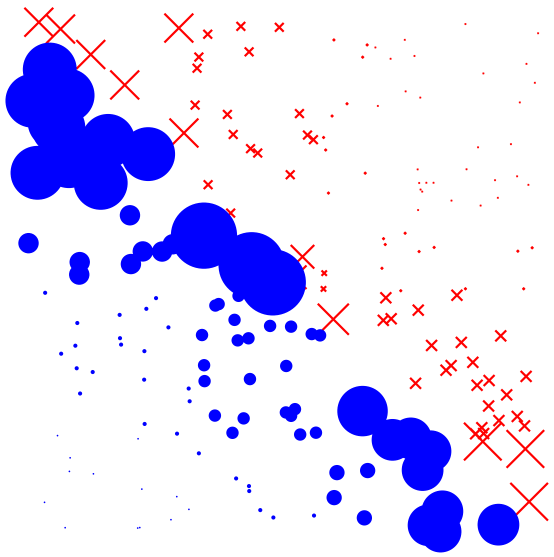
Example

Round 6



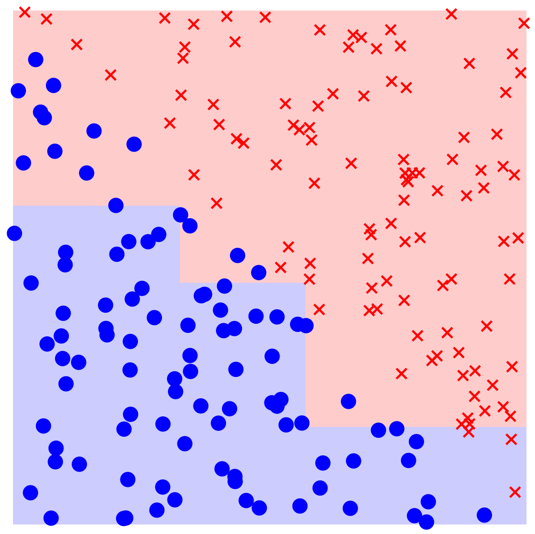
Chosen weak classifier

$\epsilon_6 = 0.2262, \ \alpha_6 = 1.2298$



Re-weight training points

$w_i^{(7)}$**'s**

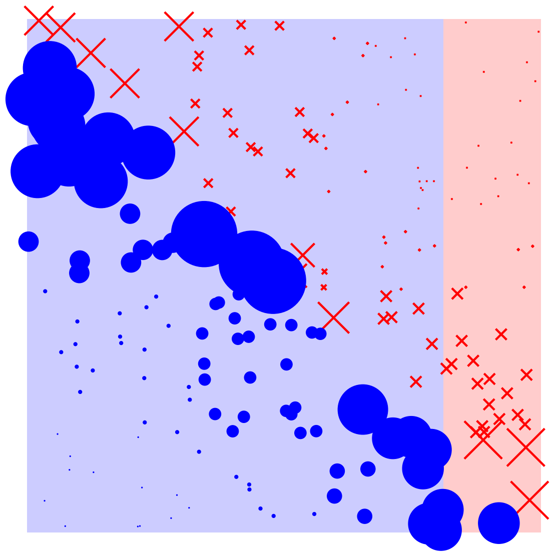


Current strong classifier

$f_6(\mathbf{x})$

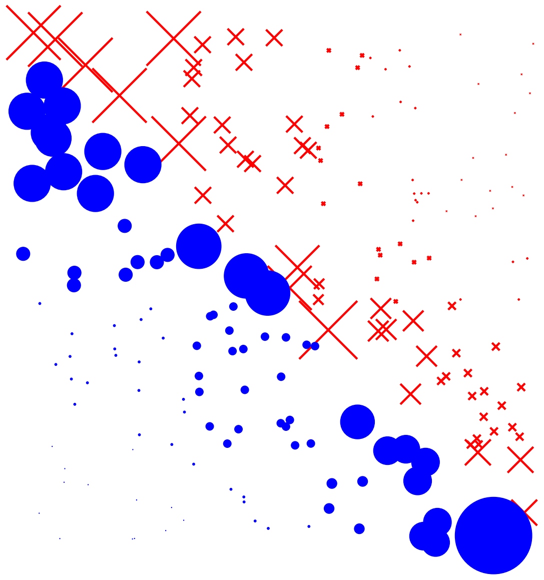
Example

Round 7



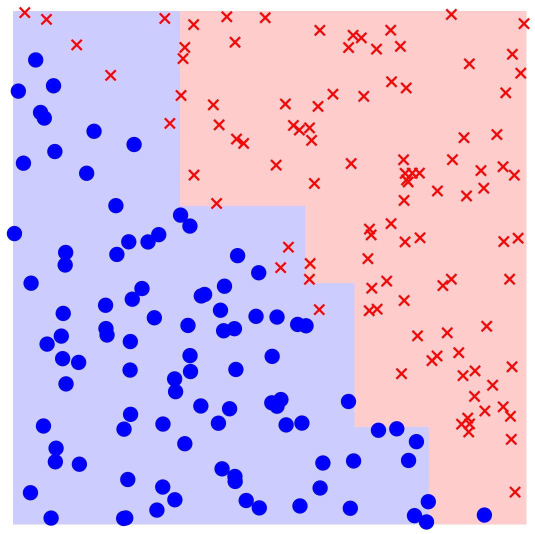
Chosen weak classifier

$\epsilon_7 = 0.2680, \; \alpha_7 = 1.0049$



Re-weight training points

$w_i^{(8)}$**'s**

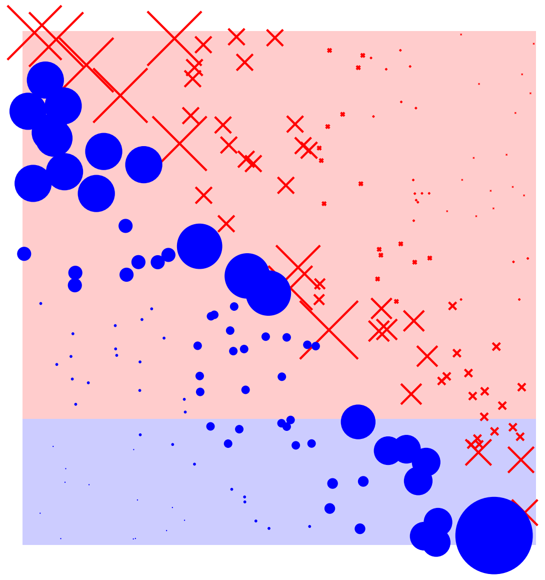


Current strong classifier

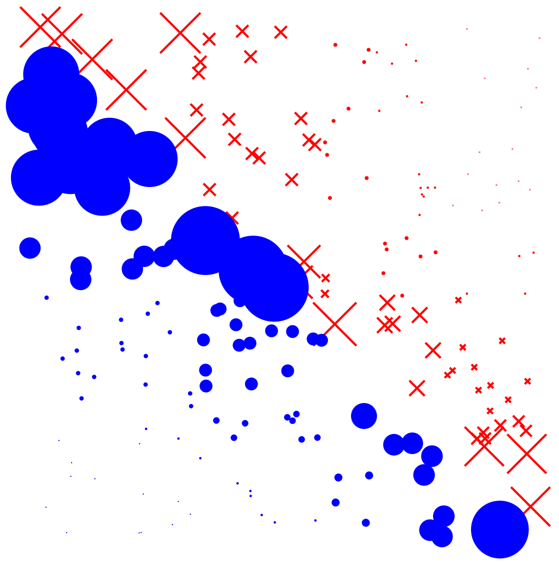
$f_7(\mathbf{x})$

Example

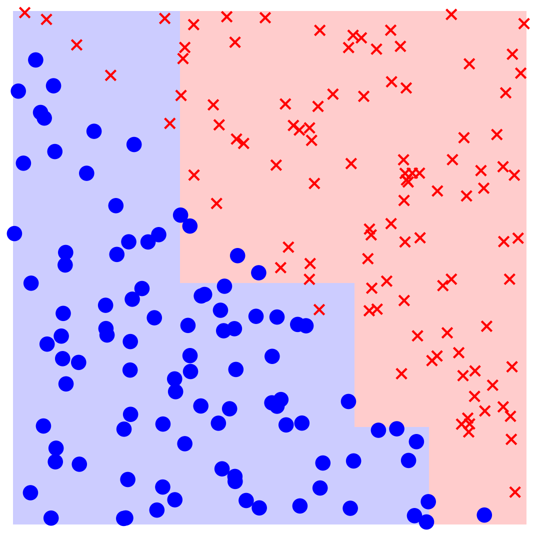
Round 8


Chosen weak classifier

$\epsilon_8 = 0.3282, \ \alpha_8 = 0.7165$


Re-weight training points

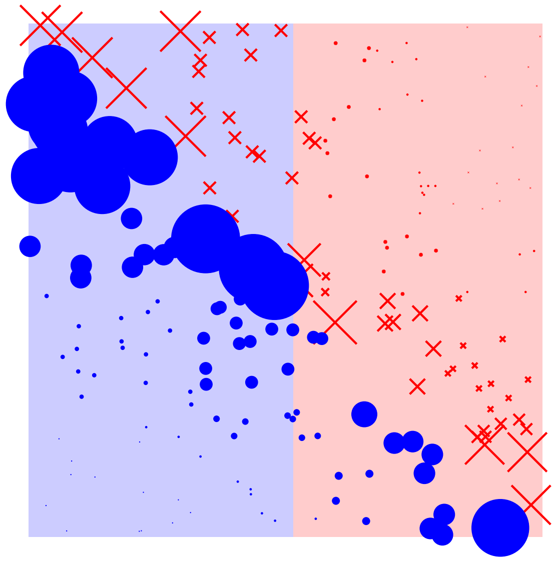
$w_i^{(9)}$**'s**


Current strong classifier

$f_8(\mathbf{x})$

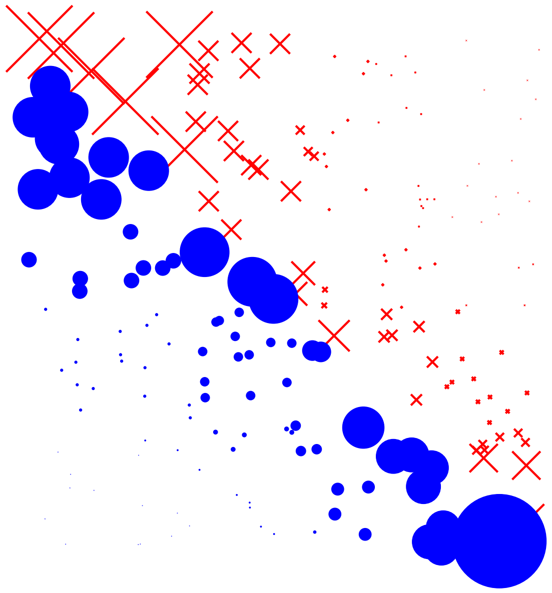
Example

Round 9



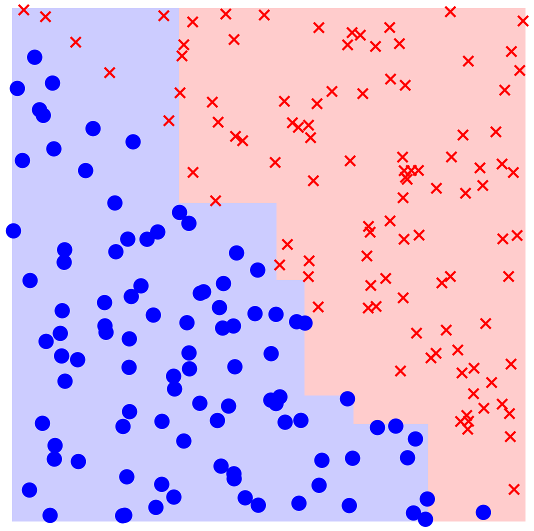
Chosen weak classifier

$\epsilon_9 = 0.3048$, $\alpha_9 = 0.8246$



Re-weight training points

$w_i^{(10)}$'s

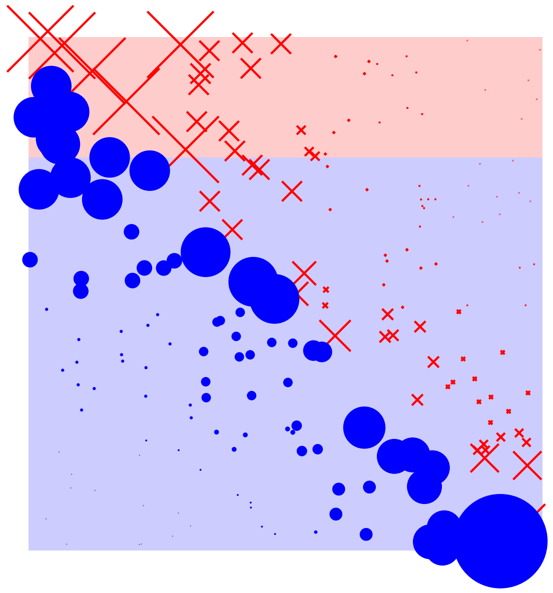


Current strong classifier

$f_9(\mathbf{x})$

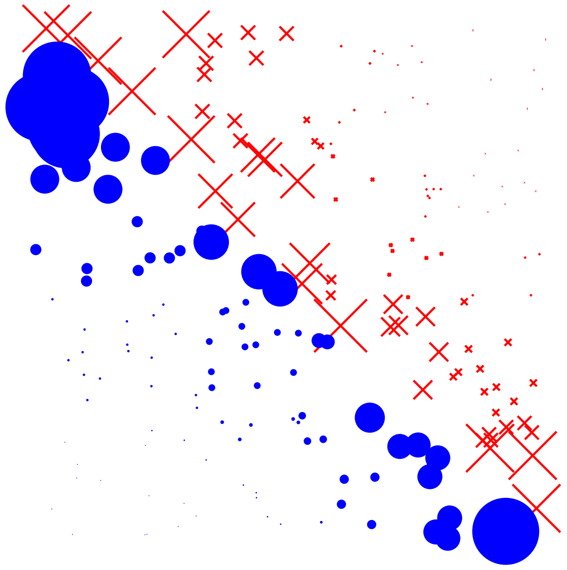
Example

Round 10



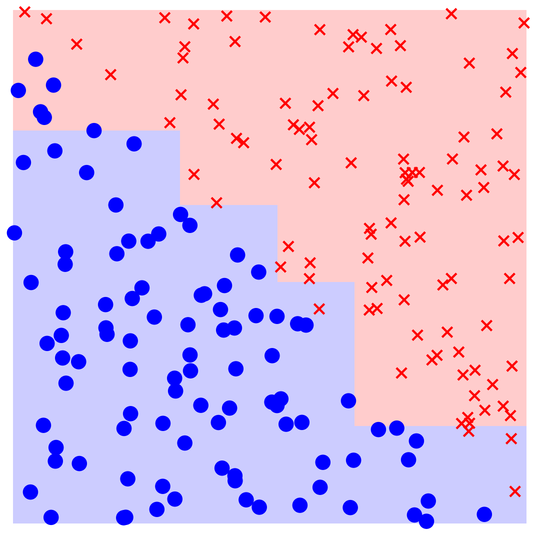
Chosen weak classifier

$\epsilon_{10} = 0.2943,\ \alpha_{10} = 0.8744$



Re-weight training points

$w_i^{(11)}$**'s**

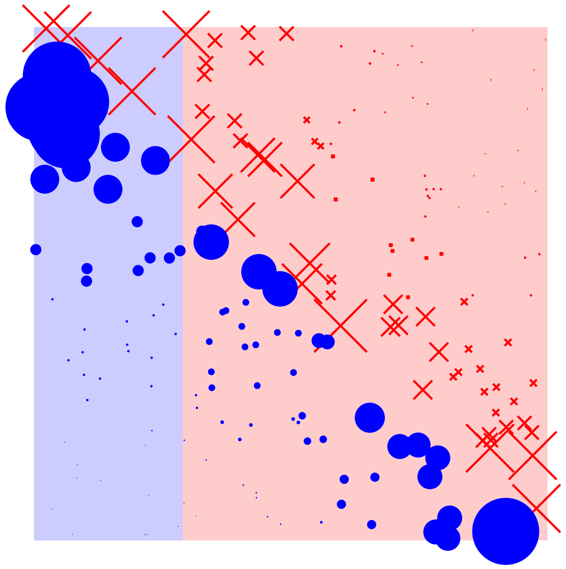


Current strong classifier

$f_{10}(\mathbf{x})$

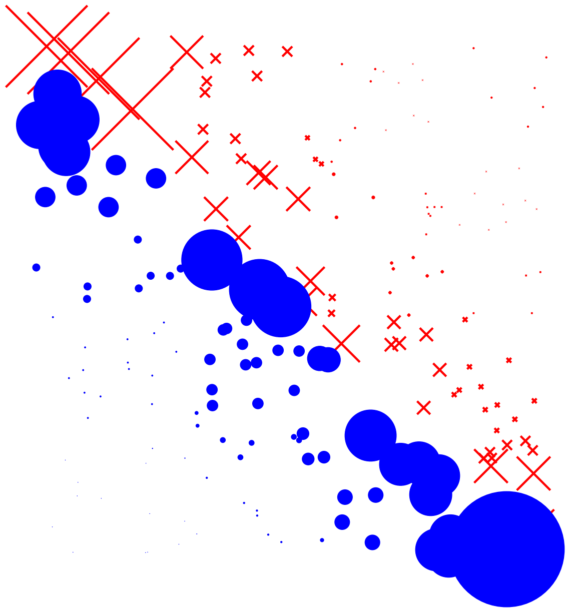
Example

Round 11



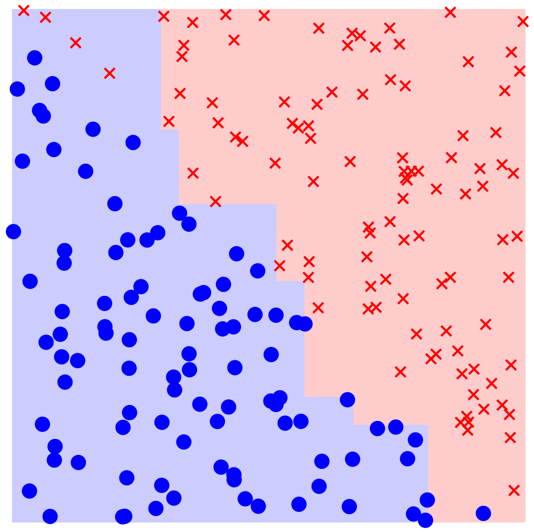
Chosen weak classifier

$\epsilon_{11} = 0.2876$, $\alpha_{11} = 0.9071$



Re-weight training points

$w_i^{(12)}$**'s**



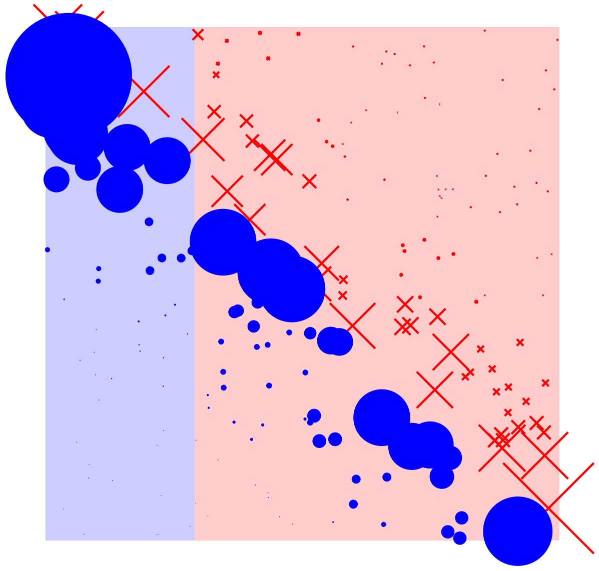
Current strong classifier

$f_{11}(\mathbf{x})$

Example

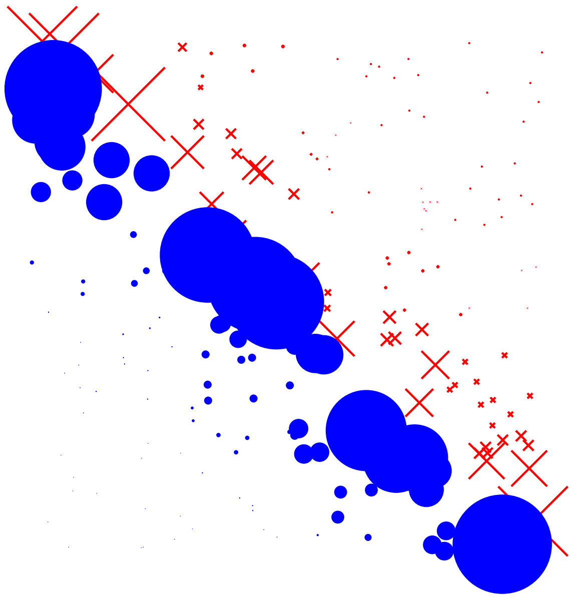
Example

Round 21



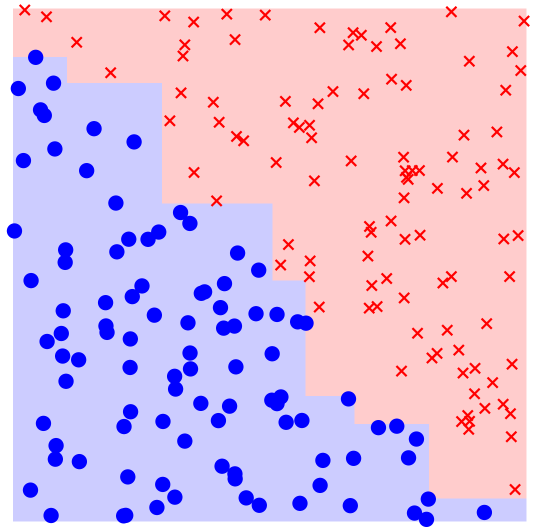
Chosen weak classifier

$\epsilon_{21} = 0.3491, \; \alpha_{21} = 0.6232$



Re-weight training points

$w_i^{(22)}$**'s**



Current strong classifier

$f_{21}(\mathbf{x})$

Interpretation of Ada-Boost

Ada-Boost returns a strong classifier of the form:

$$\phi_m(\mathbf{x}) = \sum_{t=1}^{m} \alpha_t \, h_t(\mathbf{x})$$

and is equivalent to forward stagewise additive modeling. At each iteration find the weight α_m and weak classifier h_m by solving this minimization problem

$$(\alpha_m, h_m) = \arg\min_{\alpha, h} \sum_{i=1}^{n} L(y_i, \, \phi_{m-1}(\mathbf{x}_i) + \alpha \, h(\mathbf{x}_i))$$

where the exponential loss function

$$L(y, f(\mathbf{x})) = \exp(-y \, f(\mathbf{x}))$$

is used.

Beauty of AdaBoost

Training Error

Training error $\rightarrow 0$ exponentially.

Good Generalization Properties I

Naive Expectation: Would expect over-fitting but....

Training Error: Converges to zero.

Test Error: Asymptotes - no over-fitting observed. It continues to decrease after training error vanishes.

Good Generalization Properties II

The thesis is that large margins on training set yield smaller generalization error.

$$\phi(\mathbf{x}) = \operatorname{sgn}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right) = \operatorname{sgn}\left(f(\mathbf{x})\right)$$

Classification correct: $\operatorname{sgn}\left(f(\mathbf{x})\right) = y.$

Margin: $\text{margin } f(\mathbf{x}, y) \equiv y f(\mathbf{x})$

Large Margins: *Robust* correct classification.

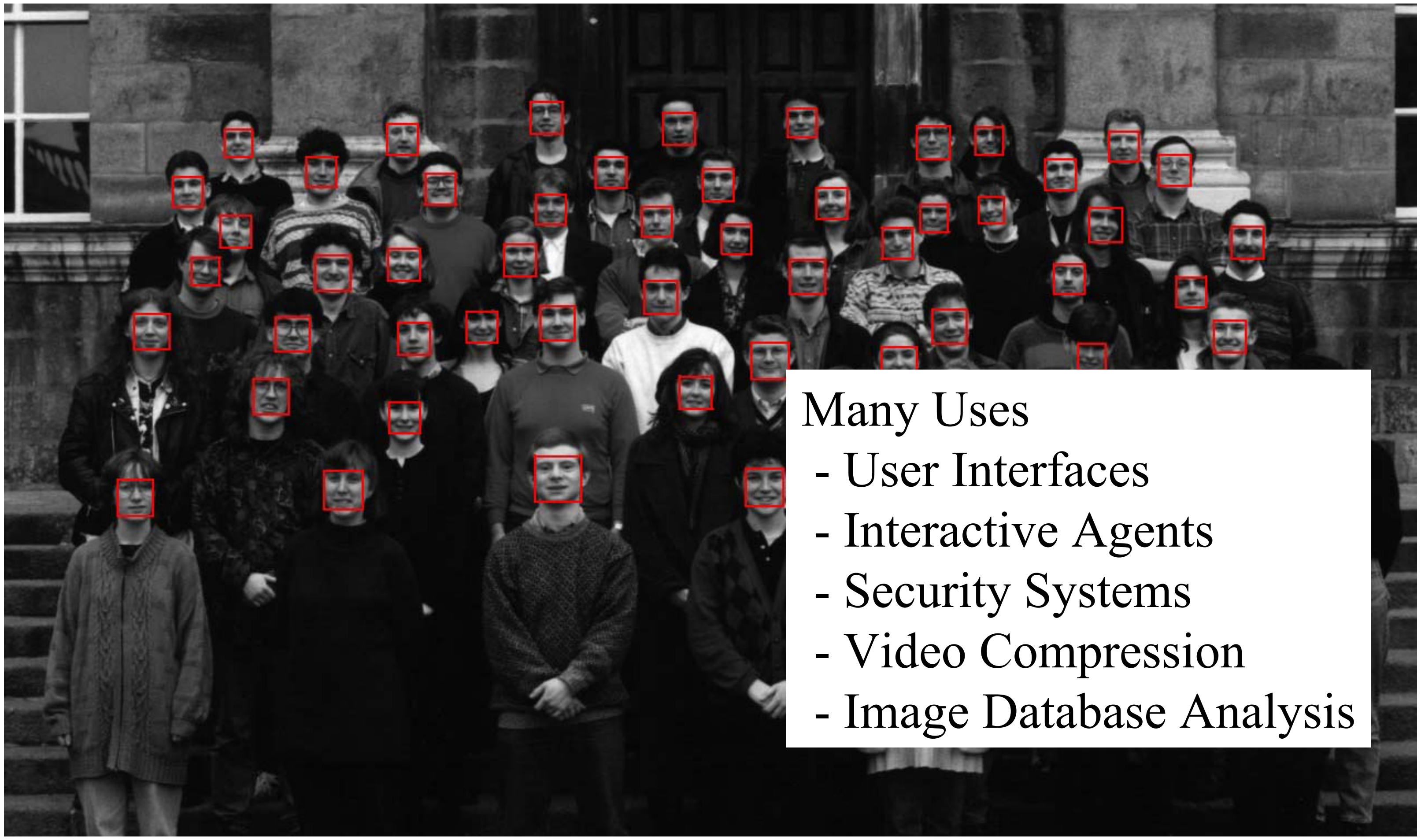
Paper review

The rest of the lecture will describe the content of this paper.

Rapid Object Detection Using a Boosted Cascade of Simple Features
by **Paul Viola** and **Michael J. Jones**, CVPR 2001

Ada-boost applied to a computer vision problem - face/object finding.

Face detection example



Many Uses
- User Interfaces
- Interactive Agents
- Security Systems
- Video Compression
- Image Database Analysis

Classifier learned from labeled data

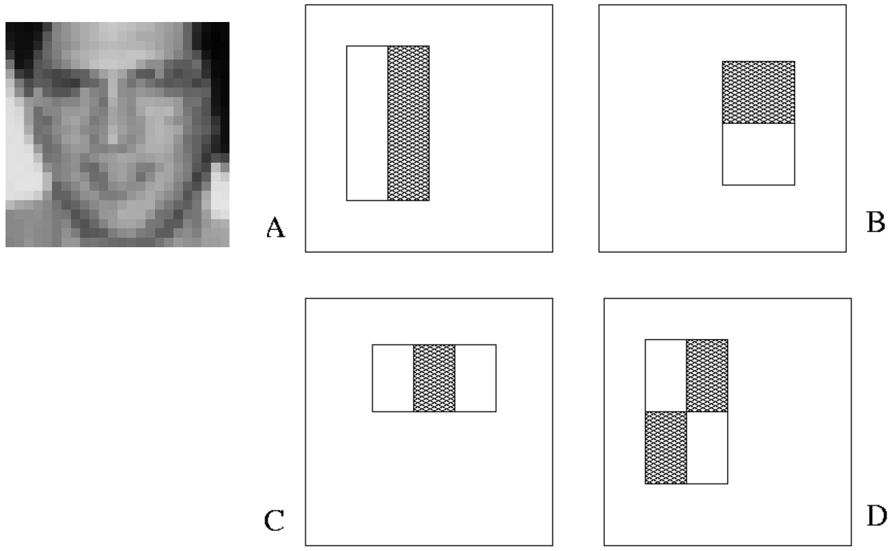
- Training Data
  - 5000 faces, all frontal
  - 10^8 non-faces
  - Faces are normalized
    scale, translation
- Many variations
  - across individuals
  - illumination
  - Pose (rotation both in plane and out)

Novelty of this approach

- Set of features - huge about 16,000,000 features.
- Efficient feature selection using Ada-Boost.
- New image representation: Integral Image.
- Cascaded Classifier for rapid detection
  - Hierarchy of Attentional Filters

Combining these ideas yields the fastest known face detector for gray scale images.

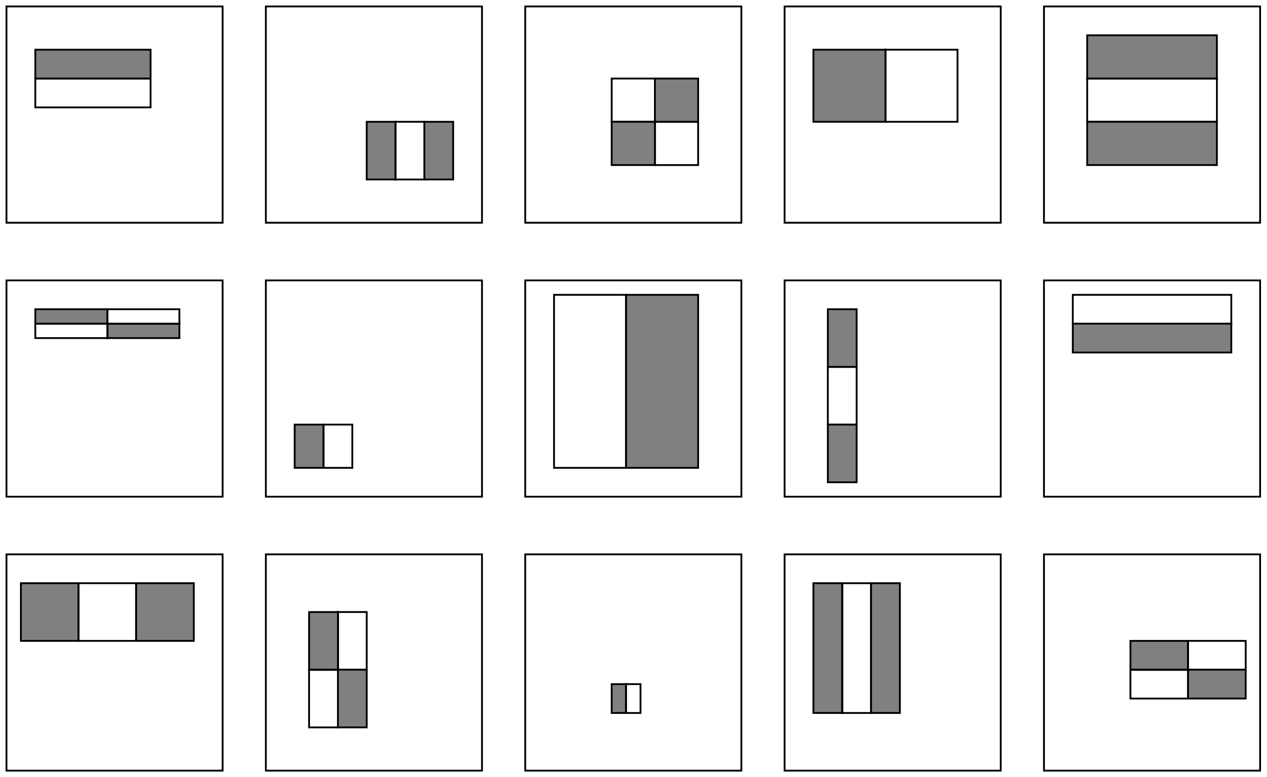
Image features



Rectangle Filters which are similar to Haar wavelets

Compute differences between sums of pixels in adjacent rectangles

Huge *library* of filters



Obtain a huge feature vector $\mathbf{x} = (x_1, \ldots, x_t, \ldots, x_d)$ where $d =$ 160,000×100=16,000,000.

How to compute them quickly: Integral image

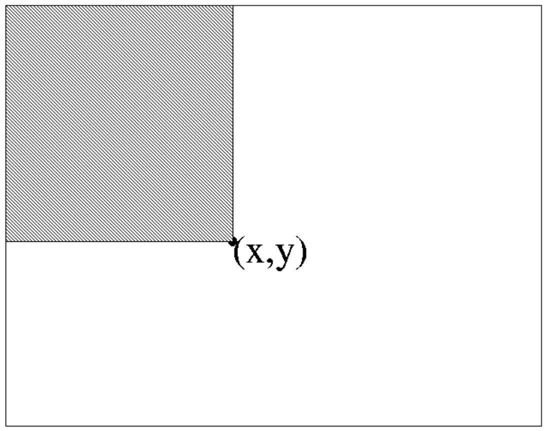
- Define the Integral Image

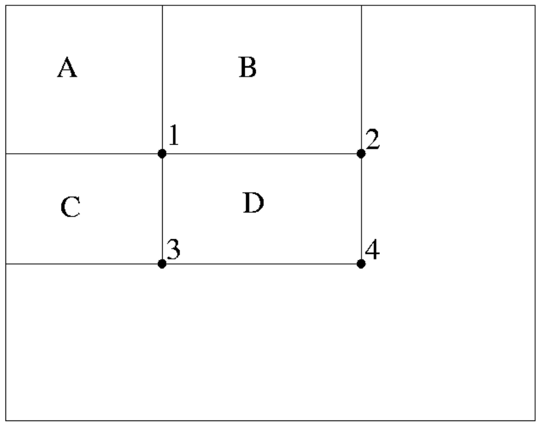
$$I'(x,y) = \sum_{x' \le x, y' \le y} I(x',y')$$

- Any rectangular sum can be computed in constant time:

$$\begin{aligned} D &= 1 + 4 - (2+3) \\ &= A + (A+B+C+D) \\ &- (A+C+A+B) = D \end{aligned}$$

- The rectangle filters can be computed as differences between rectangles.





Classifier construction

Define weak classifiers as

$$\boxed{h_t(\mathbf{x}) = \begin{cases} 1 & \text{if } x_t > \theta_t \\ -1 & \text{otherwise} \end{cases}}$$

Then use

Use AdaBoost to efficiently choose best features and to combine the weak classifiers

AdaBoost for efficient feature selection

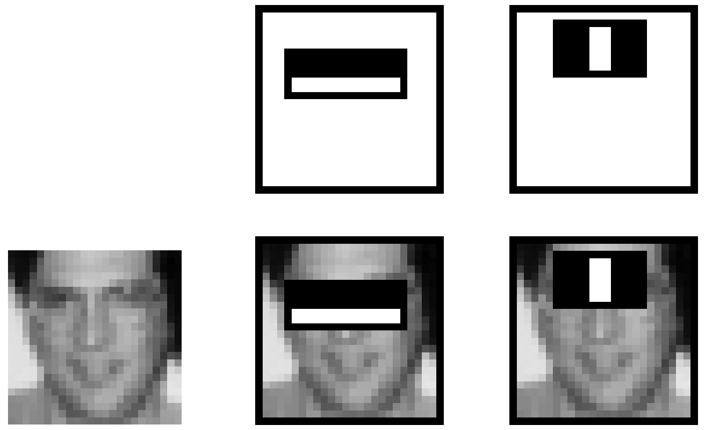
- Threshold on the response of individual Feature = Weak Classifier
- For each round of boosting:
  - Evaluate each rectangle filter on each example
  - Sort examples by filter values
  - Select best threshold for each filter (min error)
    - Sorted list can be quickly scanned for the optimal threshold
  - Select best filter/threshold combination
  - Weight on this feature is a simple function of error rate
  - Re-weight examples
  - (There are many tricks to make this more efficient.)

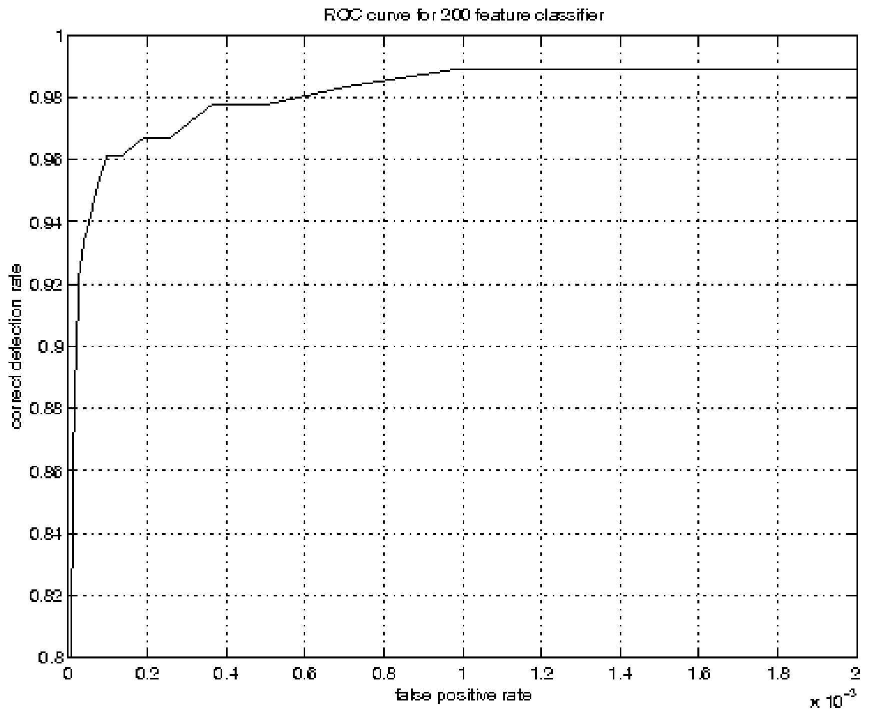
Example classifier for face detection

A classifier with 200 rectangle features was learned using AdaBoost

95% correct detection on test set with 1 in 14084 false positives.

Not quite competitive......

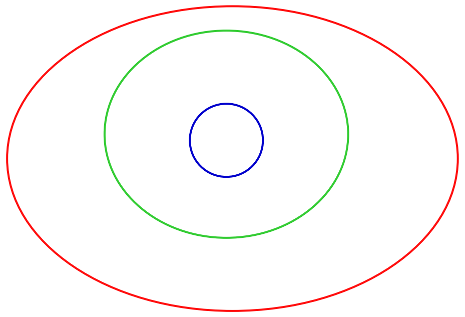


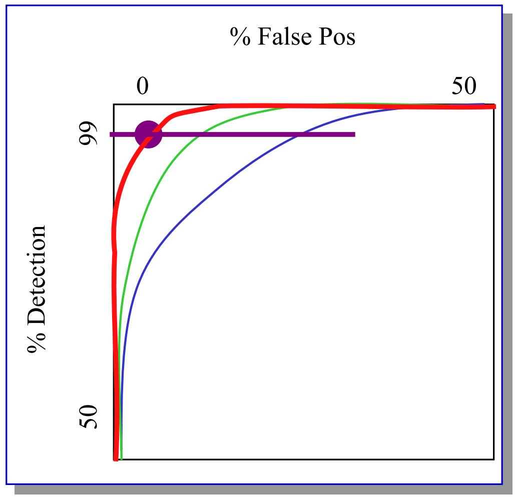


ROC curve for 200 feature classifier

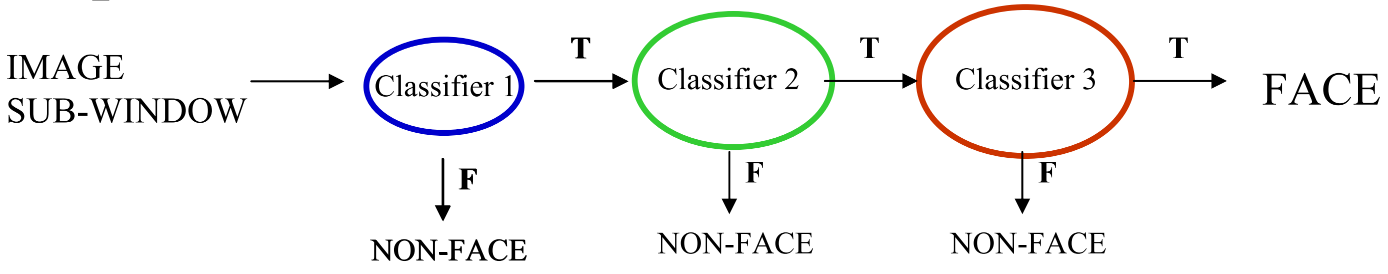
Trading speed for accuracy

Given a nested set of classifier hypothesis classes:

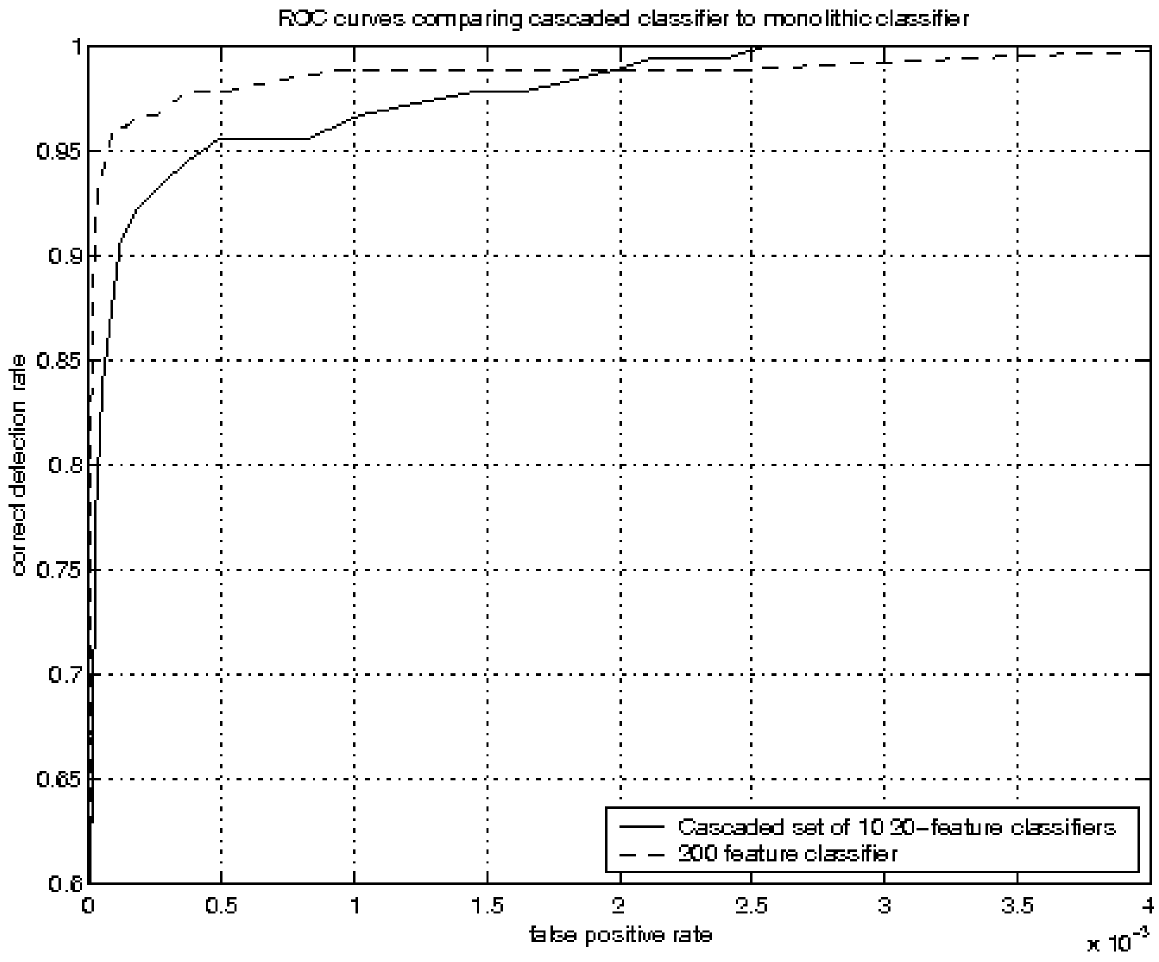




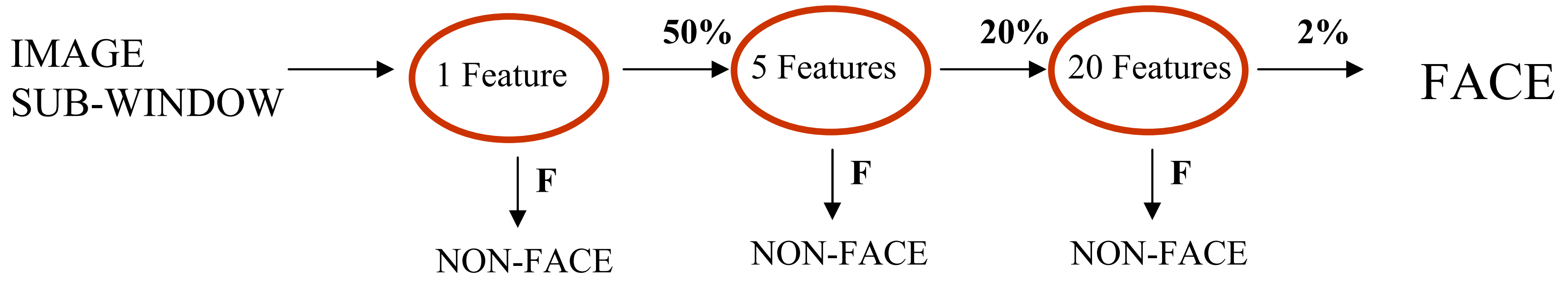
Computational Risk Minimization



Experiment: Simple cascaded classifier



Cascaded classifier



- A 1 feature classifier achieves 100% detection rate and about 50% false positive rate.
- A 5 feature classifier achieves 100% detection rate and 40% false positive rate (20% cumulative) - using data from previous stage.
- A 20 feature classifier achieves 100% detection rate with 10% false positive rate (2% cumulative)

A real-time face detection system

Training faces: 4916 face images (24×24 pixels) plus vertical flips for a total of 9832 faces.

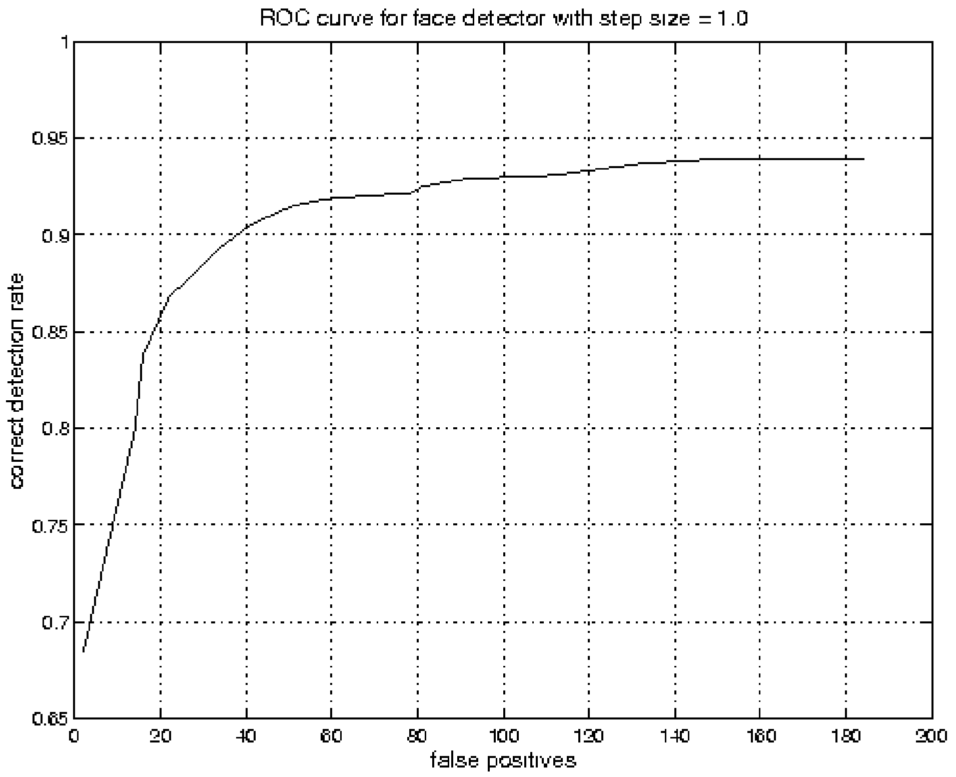
Training non-faces: 350 million sub-windows from 9500 non-face images.

Final detector: 38 layer cascaded classifier The number of features per layer was 1, 10, 25, 25, 50, 50, 50, 75, 100, . . ., 200, . . .

Final classifier contains 6061 features.

Accuracy of face detector

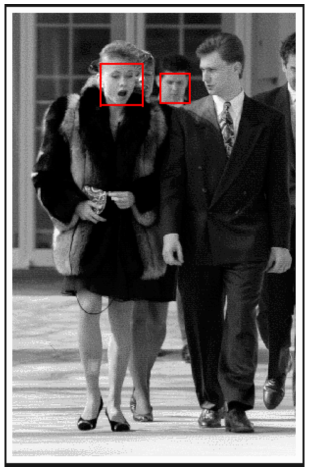
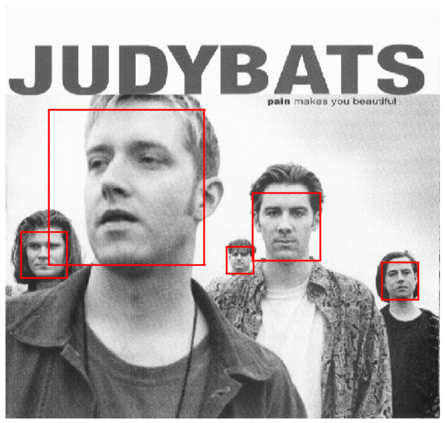
Performance on MIT+CMU test set containing 130 images with 507 faces and about 75 million sub-windows.

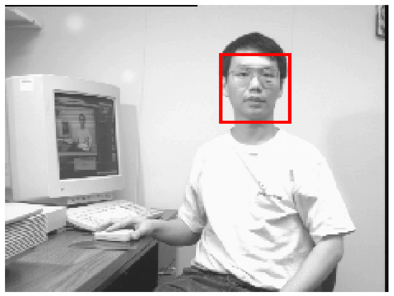
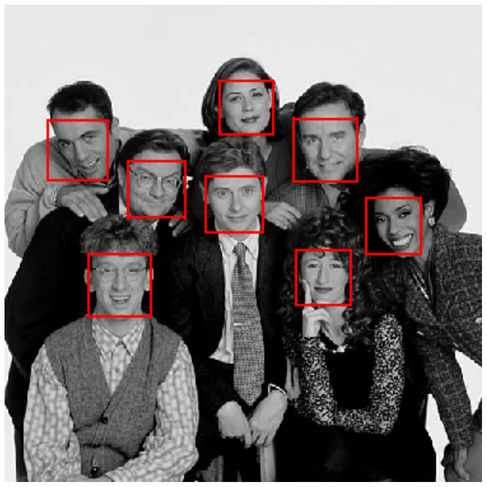
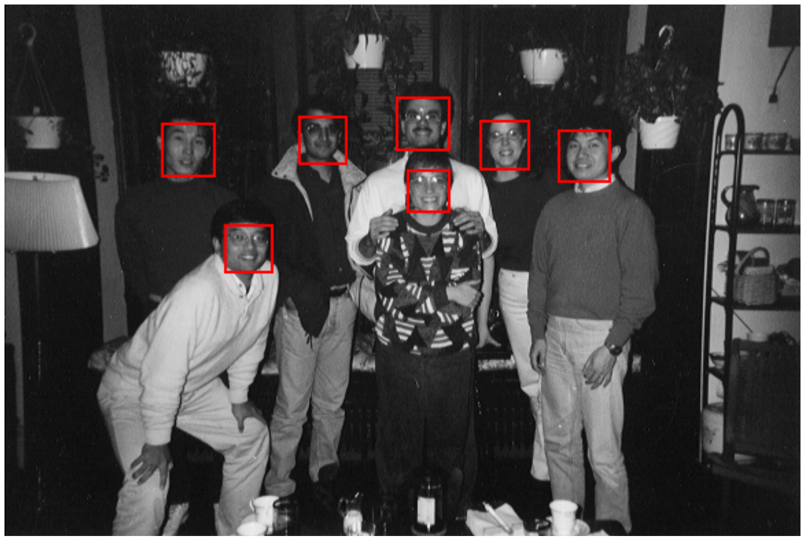


Speed of face detector

- Speed is proportional to the average number of features computed per sub-window.
- On the MIT+CMU test set, an average of 9 features out of a total of 6061 are computed per sub-window.
- On a 700 MhzPentium III, a 384$\times$288 pixel image takes about 0.067 seconds to process (15 fps).

Output of face detector on test images

Implementation notes

Training a cascade-based face detector using boosting and the *rectangular* features is **computationally expensive**.

Bottleneck is at training and selecting *rectangular* features for a **single weak classifier**.

For most published algorithms it takes at least $O(nT)$ where n is the number of examples and T is the number of features.

Paper review

Fast training and selection of Haar features using statistics in boosting-based face detection
by **Minh-Tri Pham** and **Tat-Jen Cham**, ICCV 2007

Speed up the training of the weak classifiers.

Their idea

- Have n training images and have the corresponding integral image $\mathbf{h}_i$ for each one.

- Assume that the integral images for the face and the non-face class can be described by multi-variate Gaussian distributions.

$$N(\boldsymbol{\mu}_F, \Sigma_F) \quad \text{and} \quad N(\boldsymbol{\mu}_{NF}, \Sigma_{NF})$$

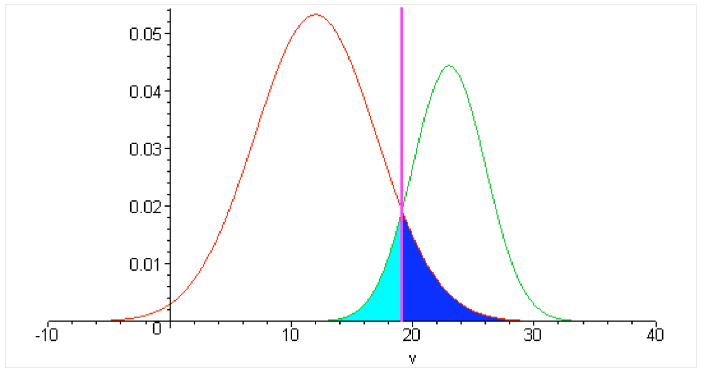
  (These means and covariances are computed using the weights associated with each image within the boosting framework.)

- Then the distribution of a rectangular feature for each class can be described by a 1D Gaussian:

$$N(\mathbf{g}^T\boldsymbol{\mu}_F, \mathbf{g}^T\Sigma_F\mathbf{g}) \quad \text{and} \quad N(\mathbf{g}^T\boldsymbol{\mu}_{NF}, \mathbf{g}^T\Sigma_{NF}\,\mathbf{g})$$

where $\mathbf{g}$ is a sparse vector that results in $\mathbf{g}^T\mathbf{h}_i$ equally the value of applying a rectangular feature to the original image.

- This results in two classes each following a 1D Gaussian distribution:



- In this case there is a closed form solution for finding the threshold to discriminate between the two classes.

- Training time now takes $O(nd^2 + T)$ where d is the number of pixels in the image.

- Take home message: learning *less than optimal* weak classifiers has allowed a much faster training time which in turn has allowed the exploration of a much larger potential set of features from which to build the final classifier.

  The latter more than compensates for the sub-optimal performance of the weak classifier in the final classifier.

Pen & Paper assignment

- Details available on the course website.
- Your assignment is a small pen & paper exercise based on the ada-boost algorithm.
- Mail me about any errors you spot in the Exercise notes.
- I will notify the class about errors spotted and corrections via the course website and mailing list.