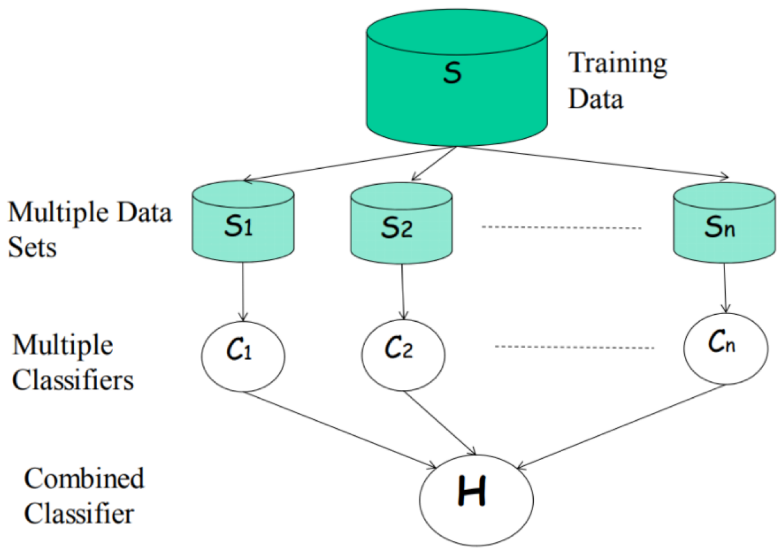
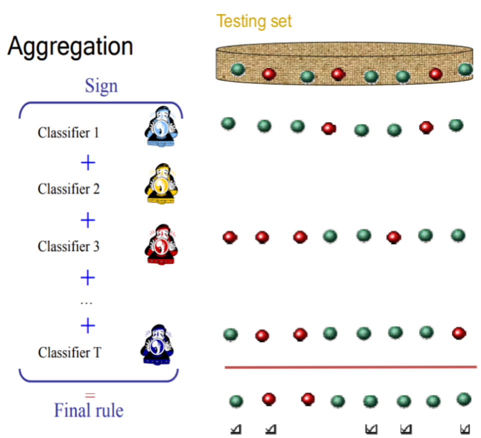
Rensselaer

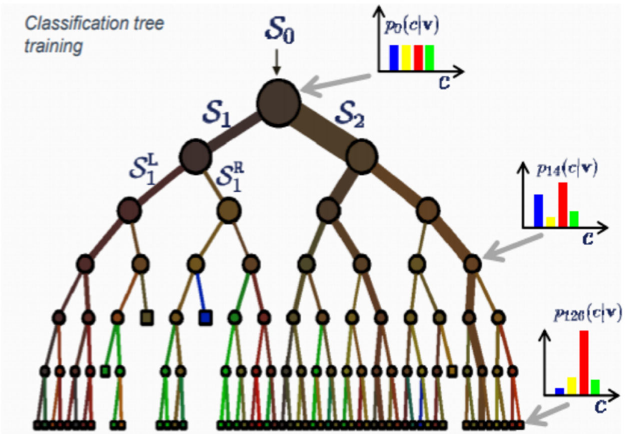
# Lecture 14: Bagging, Random Forests; Boosting (2)

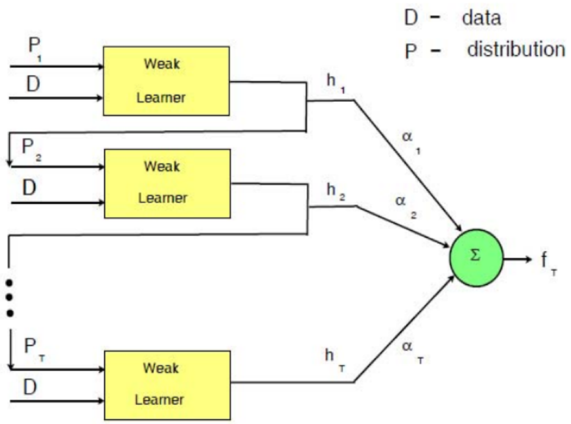
Dr. Chengjiang Long
Computer Vision Researcher at Kitware Inc.
Adjunct Professor at RPI.
Email: **longc3@rpi.edu**

# Recap Previous Lecture



# Outline

- How to Choose Weights for Boosting?
- Important Aspects of Boosting
  - Exponential loss function
  - Choice of weak learners
  - Generalization and overfitting
  - Multi-class boosting
- Applications of Boosting

# Outline

- **How to Choose Weights for Boosting?**
- Important Aspects of Boosting
  - Exponential loss function
  - Choice of weak learners
  - Generalization and overfitting
  - Multi-class boosting
- Applications of Boosting

# The AdaBoost Algorithm

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X, y_i \in Y = \{-1, +1\}$
Initialize $D_1(i) = 1/m$.
For $t = 1, \ldots, T$:

- Train base learner using distribution $D_t$.
- Get base classifier $h_t : X \to \mathbb{R}$.
- Choose $\alpha_t \in \mathbb{R}$. ← $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$
- Update:

$$D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

$$\epsilon_t = \Pr_{i \sim D_t}\left[h_t(x_i) \neq y_i\right]$$

$$\epsilon_t = \frac{1}{\sum_{i=1}^{n} D_t(i)} \sum_{i=1}^{m} D_t(i)\delta(h_t(x_i) \neq y_i)$$

# Analyzing training error

- What $\alpha_t$ to choose for hypothesis $h_t$?

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

$\varepsilon_t$ - weighted training error

- If each weak learner $h_t$ is slightly better than random guessing ($\varepsilon_t < 0.5$), then training error of AdaBoost decays exponentially fast in number of rounds T.

$$\frac{1}{m} \sum_{i=1}^{m} \delta(H(x_i) \neq y_i) \leq \exp\left(-2 \sum_{t=1}^{T} (1/2 - \epsilon_t)^2\right)$$

**Training Error**

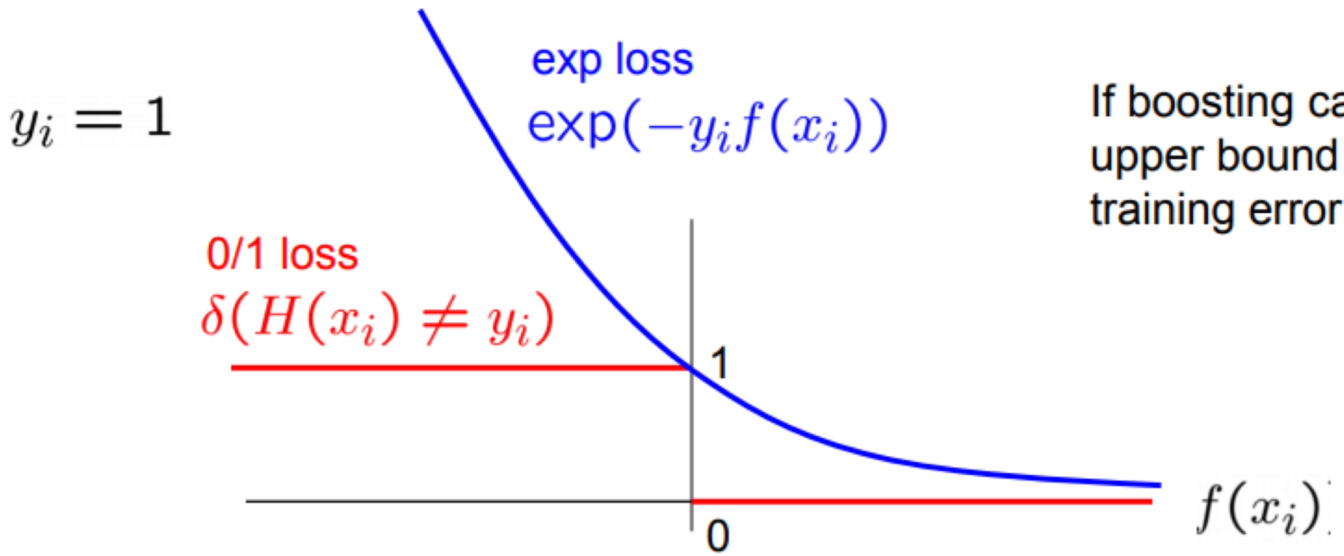
# Analyzing training error

- Training error of final classifier is bounded by:

$$\frac{1}{m}\sum_{i=1}^{m}\delta(H(x_i) \neq y_i) \leq \frac{1}{m}\sum_{i=1}^{m}\exp(-y_i f(x_i))$$

Convex upper bound

Where $f(x) = \sum_t \alpha_t h_t(x); H(x) = sign(f(x))$




If boosting can make upper bound → 0, then training error → 0

# Analyzing training error

- Training error of final classifier is bounded by:

$$\frac{1}{m}\sum_{i=1}^{m}\delta(H(x_i)\neq y_i) \leq \underline{\frac{1}{m}\sum_{i=1}^{m}\exp(-y_i f(x_i)) = \prod_t Z_t}$$

Where $f(x)=\sum_t \alpha_t h_t(x); H(x)=sign(f(x))$

**Proof:** **Using Weight Update Rule**

$$D_1(i)=1/m$$

$$D_2(i)=\frac{1}{m}\frac{e^{-\alpha_1 y_i h_1(x_i)}}{Z_1}$$

$$D_3(i)=\frac{1}{m}\frac{e^{-\alpha_1 y_i h_1(x_i)}e^{-\alpha_2 y_i h_2(x_i)}}{Z_1 Z_2}$$

...

$$D_{T+1}(i)=\frac{1}{m}\frac{\exp(-y_i f(x_i))}{\prod_t Z_t}$$

**Wts of all pts add to 1**

$$\sum_{i=1}^{m} D_{T+1}(i)=1$$

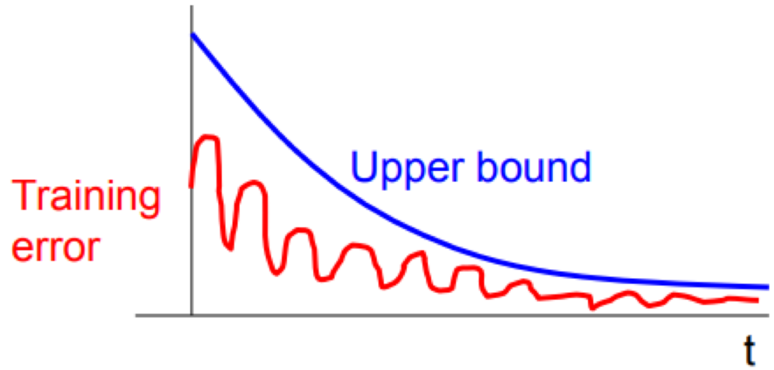
# Analyzing training error

- Training error of final classifier is bounded by:

$$\frac{1}{m}\sum_{i=1}^{m}\delta(H(x_i)\neq y_i) \leq \frac{1}{m}\sum_{i=1}^{m}\exp(-y_i f(x_i)) = \prod_t Z_t$$

Where $f(x) = \sum_t \alpha_t h_t(x); H(x) = sign(f(x))$

- If Zt < 1, training error decreases exponentially (even though weak learners may not be good εt ~0.5)



# What $\alpha_t$ to choose for hypothesis $h_t$?

- Training error of final classifier is bounded by:

$$\frac{1}{m}\sum_{i=1}^{m}\delta(H(x_i)\neq y_i) \leq \frac{1}{m}\sum_{i=1}^{m}\exp(-y_i f(x_i)) = \prod_t Z_t$$

Where $f(x) = \sum_t \alpha_t h_t(x); H(x) = sign(f(x))$

- If we minimize $\prod_t Z_t$ , we minimize our training error, We can tighten this bound greedily, by choosing $\alpha_t$ and $h_t$ on each iteration to minimize $Z_t$.

$$Z_t = \sum_{i=1}^{m} D_t(i)\exp(-\alpha_t y_i h_t(x_i))$$

# What $\alpha_t$ to choose for hypothesis $h_t$?

- We can tighten this bound greedily, by choosing αt and ht on each iteration to minimize Zt.

$$Z_t = \sum_{i=1}^{m} D_t(i) \exp(-\alpha_t y_i h_t(x_i))$$

- For boolean target function, this is accomplished by [Freund & Schapire '97]:

$$\boxed{\alpha_t = \tfrac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right)}$$

**Proof:**

$$\begin{aligned} Z_t &= \sum_{i:y_i \neq h_t(x_i)} D_t(i) e^{\alpha_t} + \sum_{i:y_i = h_t(x_i)} D_t(i) e^{-\alpha_t} \\ &= \epsilon_t e^{\alpha_t} + (1-\epsilon_t) e^{-\alpha_t} \end{aligned}$$

$$\frac{\partial Z_t}{\alpha_t} = \epsilon_t e^{\alpha_t} - (1-\epsilon_t) e^{-\alpha_t} = 0 \qquad \Rightarrow e^{2\alpha_t} = \frac{1-\epsilon_t}{\epsilon_t}$$

# What $\alpha_t$ to choose for hypothesis $h_t$?

- We can tighten this bound greedily, by choosing αt and ht on each iteration to minimize Zt.

$$Z_t = \sum_{i=1}^{m} D_t(i) \exp(-\alpha_t y_i h_t(x_i))$$

- For boolean target function, this is accomplished by [Freund & Schapire ’97]:

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

**Proof:**

$$\begin{aligned} Z_t &= \sum_{i:y_i \neq h_t(x_i)} D_t(i) e^{\alpha_t} + \sum_{i:y_i = h_t(x_i)} D_t(i) e^{-\alpha_t} \\ &= \epsilon_t e^{\alpha_t} + (1-\epsilon_t) e^{-\alpha_t} \\ &= 2\sqrt{\epsilon_t(1-\epsilon_t)} = \sqrt{1-(1-2\epsilon_t)^2} \end{aligned}$$

# Dumb classifiers made Smart

- Training error of final classifier is bounded by:

$$\frac{1}{m}\sum_{i=1}^{m}\delta(H(x_i)\neq y_i)\leq\prod_t Z_t=\prod_t\sqrt{1-(1-2\epsilon_t)^2}$$

$$\leq\exp\left(-2\sum_{t=1}^{T}(1/2-\epsilon_t)^2\right)$$

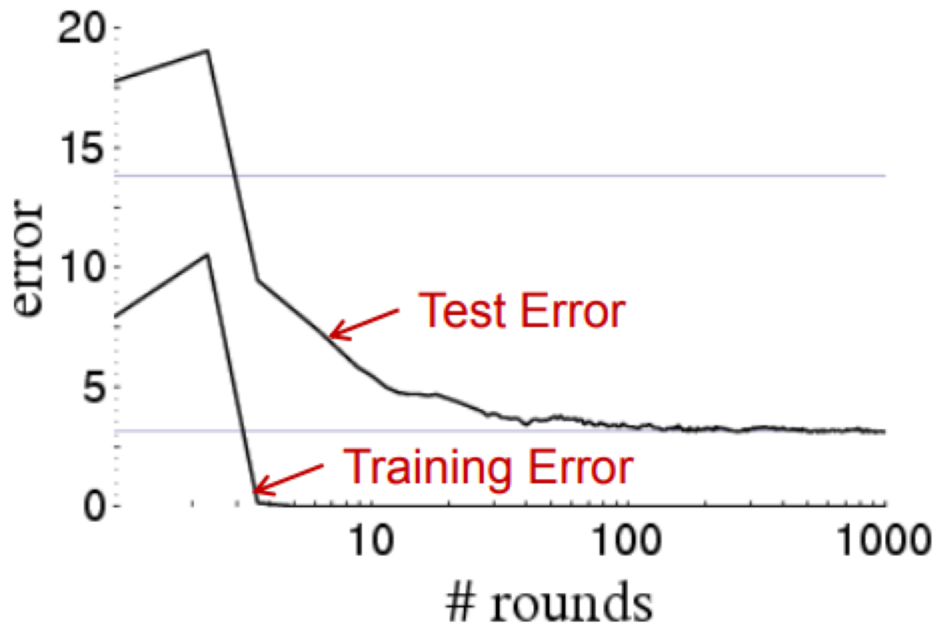
grows as $\varepsilon_t$ moves away from 1/2

**If each classifier is (at least slightly) better than random $\varepsilon_t < 0.5$**

**AdaBoost will achieve zero *training error* exponentially fast (in number of rounds T) !!**

**What about test error?**

# Boosting results – Digit recognition




[Schapire, 1989]

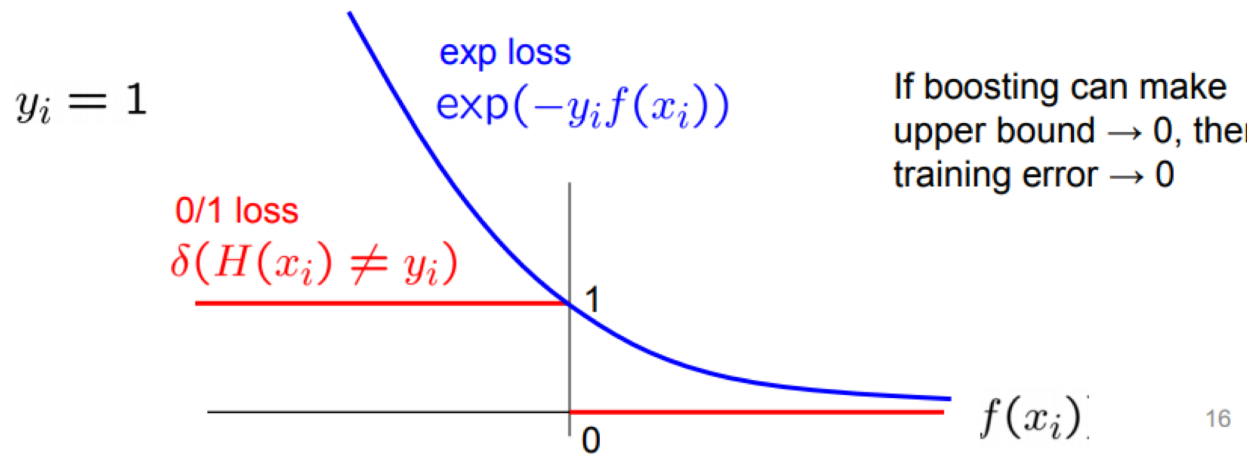
- Boosting often, **but not always**
  – Robust to overfitting
  – Test set error decreases even after training error is zero

# Outline

- How to Choose Weights for Boosting?
- Important Aspects of Boosting
  - **Exponential loss function**
  - Choice of weak learners
  - Generalization and overfitting
  - Multi-class boosting
- Applications of Boosting

# Exponential Loss Function

- The exponential loss function is an upper bound of the 0-1 loss function (classification error)
- AdaBoost provably minimizes exponential loss
- Therefore, it also minimizes the upper bound of classification error



# Exponential Loss Function

- AdaBoost attempts to minimize:

$$\prod_t Z_t = \frac{1}{m} \sum_i \exp(-y_i f(x_i))$$

- Really a coordinate descent procedure
  – At each round add at $h_t$ to sum to minimize the above objective function

- Why this loss function?
  – upper bound on training (classification) error
  – easy to work with
  – connection to logistic regression

# Coordinate Descent Explanation

- $\{g_1, \ldots, g_N\}$ = space of all weak classifiers
- want to find $\lambda_1, \ldots, \lambda_N$ to minimize

$$L(\lambda_1, \ldots, \lambda_N) = \sum_i \exp\left(-y_i \sum_j \lambda_j g_j(x_i)\right)$$

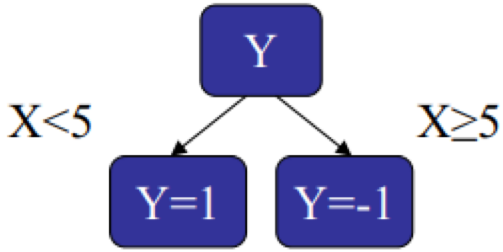
- AdaBoost is actually doing coordinate descent on this optimization problem:
  - initially, all $\lambda_j = 0$
  - each round: choose one coordinate $\lambda_j$ (corresponding to $h_t$) and update (increment by $\alpha_t$)
  - choose update causing biggest decrease in loss
- powerful technique for minimizing over huge space of functions

# Outline

- How to Choose Weights for Boosting?
- Important Aspects of Boosting
  - Exponential loss function
  - **Choice of weak learners**
  - Generalization and overfitting
  - Multi-class boosting
- Applications of Boosting

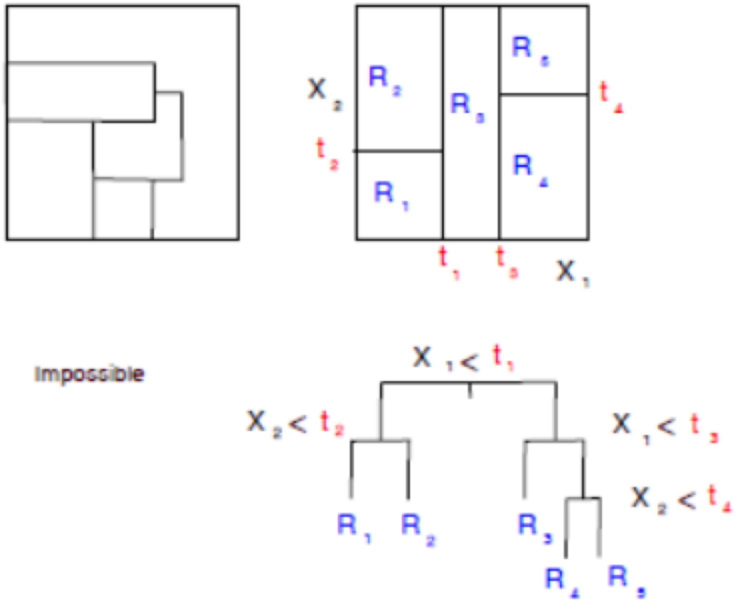
# Weak Learners

- Stumps:
  – Single-axis parallel partition of space
- Decision trees:
  – Hierarchical partition of space
- Multi-layer perceptrons:
  – General nonlinear function approximators

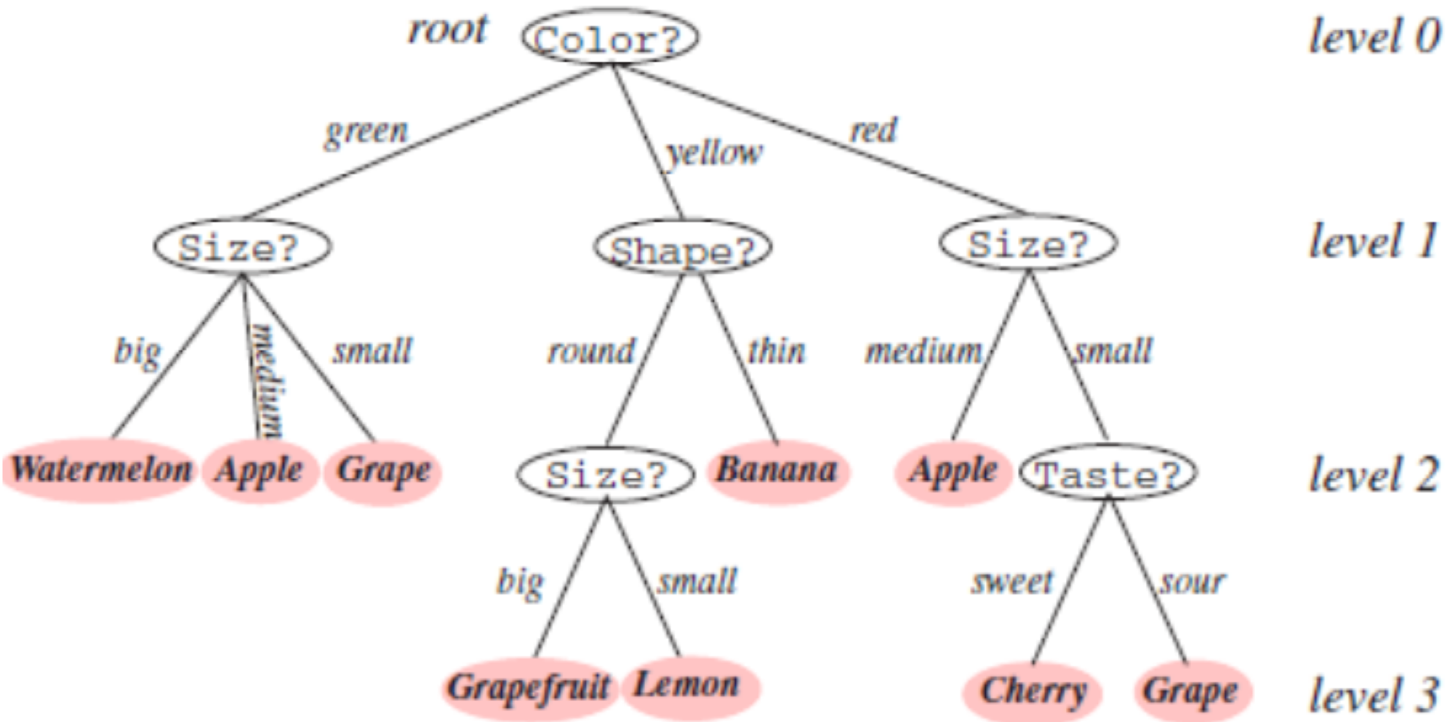


# Decision Trees

- Hierarchical and recursive partitioning of the feature space
- A simple model (e.g. constant) is fit in each region
- Often, splits are parallel to axes



# Decision Trees – Nominal Features



# Outline

- How to Choose Weights for Boosting?
- Important Aspects of Boosting
  - Exponential loss function
  - Choice of weak learners
  - **Generalization and overfitting**
  - Multi-class boosting
- Applications of Boosting

# Generalization Error

$$error_{true}(H) \leq error_{train}(H) + \tilde{O}\left(\sqrt{\frac{Td}{m}}\right)$$

- T – number of boosting rounds
- d – VC dimension of weak learner, measures complexity of classifier
  – The Vapnik-Chervonenkis (VC) dimension is a standard measure of the "complexity" of a space of binary functions
- m – number of training examples

# Overfitting

- This bound suggests that boosting will overfit if run for too many rounds
- Several authors observed empirically that boosting often does not overfit, even when run for thousands of rounds
  – Moreover, it was observed that AdaBoost would sometimes continue to drive down the generalization error long after the training error had reached zero, clearly contradicting the bound above

# Analysis of Margins

- An alternative analysis can be made in terms of the **margins** of the training examples. The margin of example (x,y) is:

$$\text{margin}_f(x, y) = \frac{yf(x)}{\sum_t |\alpha_t|} = \frac{y \sum_t \alpha_t h_t(x)}{\sum_t |\alpha_t|}$$

- It is a number in [-1, 1] and it is positive when the example is correctly classified
- Larger margins on the training set translate into a superior upper bound on the generalization error

# Analysis of Margins

- It can be shown that the generalization error is at most:

$$\hat{\Pr}\left[\text{margin}_f(x,y) \leq \theta\right] + \tilde{O}\left(\sqrt{\frac{d}{m\theta^2}}\right)$$

  – Independent of T

- Boosting is particularly aggressive at increasing the margin since it concentrates on the examples with the smallest margins
  – positive or negative

# Margin Analysis

- Margin theory gives a qualitative explanation of the effectiveness of boosting
- Quantitatively, the bounds are rather weak
- One classifier can have a margin distribution that is better than that of another classifier, and yet be inferior in test accuracy
- Margin theory points to a strong connection between boosting and the support vector machines

# Advantages of Boosting

- Simple and easy to implement
- Flexible – can be combined with any learning algorithm
- No requirement on data being in metric space

  – data features don't need to be normalized, like in kNN and SVMs (this has been a central problem in machine learning)
- Feature selection and fusion are naturally combined with the same goal for minimizing an objective error function

# Advantages of Boosting (cont.)

- Can show that if a gap exists between positive and negative points, generalization error converges to zero
- No parameters to tune (maybe T)
- No prior knowledge needed about weak learner
- Provably effective
- Versatile – can be applied on a wide variety of problems
- Non-parametric

# Disadvantages of Boosting

- Performance of AdaBoost depends on data and weak learner
- Consistent with theory, AdaBoost can fail if
  - weak classifier too complex – overfitting
  - weak classifier too weak - underfitting
- Empirically, AdaBoost seems especially susceptible to uniform noise
- Decision boundaries are often rugged

# Outline

- How to Choose Weights for Boosting?
- Important Aspects of Boosting
  - Exponential loss function
  - Choice of weak learners
  - Generalization and overfitting
  - **Multi-class boosting**
- Applications of Boosting

# Multi-class AdaBoost

- Assume y϶{1,…,k}
- Direct approach (AdaBoost.M1):

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} \beta_t & \text{if } h_t(x_i) = y_i \\ 1 & \text{otherwise} \end{cases}$$

$$\beta_t = \epsilon_t / (1 - \epsilon_t)$$

$$h_{fin}(x) = \arg\max_{y \in Y} \sum_{t: h_t(x) = y} \log \frac{1}{\beta_t}$$

- can prove same bound on error if $\varepsilon t < 0.5$
- else: abort

# Reducing to Binary Problems

- Say possible labels are {a,b,c,d,e}
- Each training example replaced by five {-1,+1} labeled examples

$$x \quad , \quad c \rightarrow \begin{cases} (x, a) & , \quad -1 \\ (x, b) & , \quad -1 \\ (x, c) & , \quad +1 \\ (x, d) & , \quad -1 \\ (x, e) & , \quad -1 \end{cases}$$

# Limitation of AdaBoost.M1

- Achieving $\varepsilon_t < 0.5$, may be hard if k (number of classes) is large
- [Mukherjee and Schapire, 2010]: weak learners that perform slightly better than random chance can be used in multi-class boosting framework

# Outline

- How to Choose Weights for Boosting?
- Important Aspects of Boosting
  - Exponential loss function
  - Choice of weak learners
  - Generalization and overfitting
  - Multi-class boosting
- **Applications of Boosting**

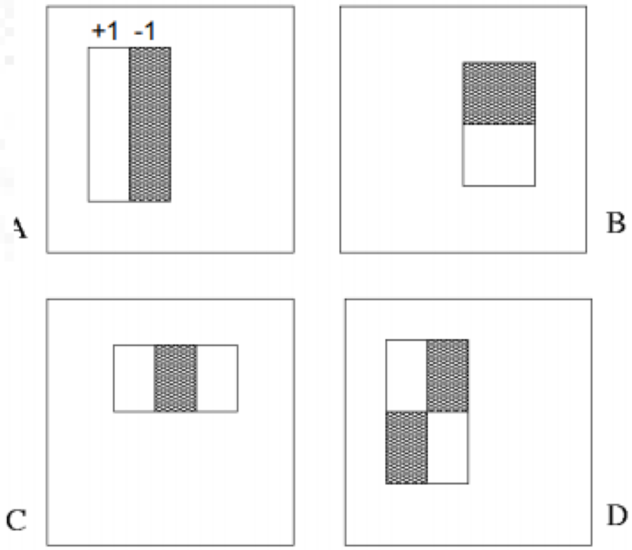
# Face detection and recognition

# Face Detection


- Where are the faces?
- What kind of features?
- What kind of classifiers?

# Image Features

"Rectangle filters"



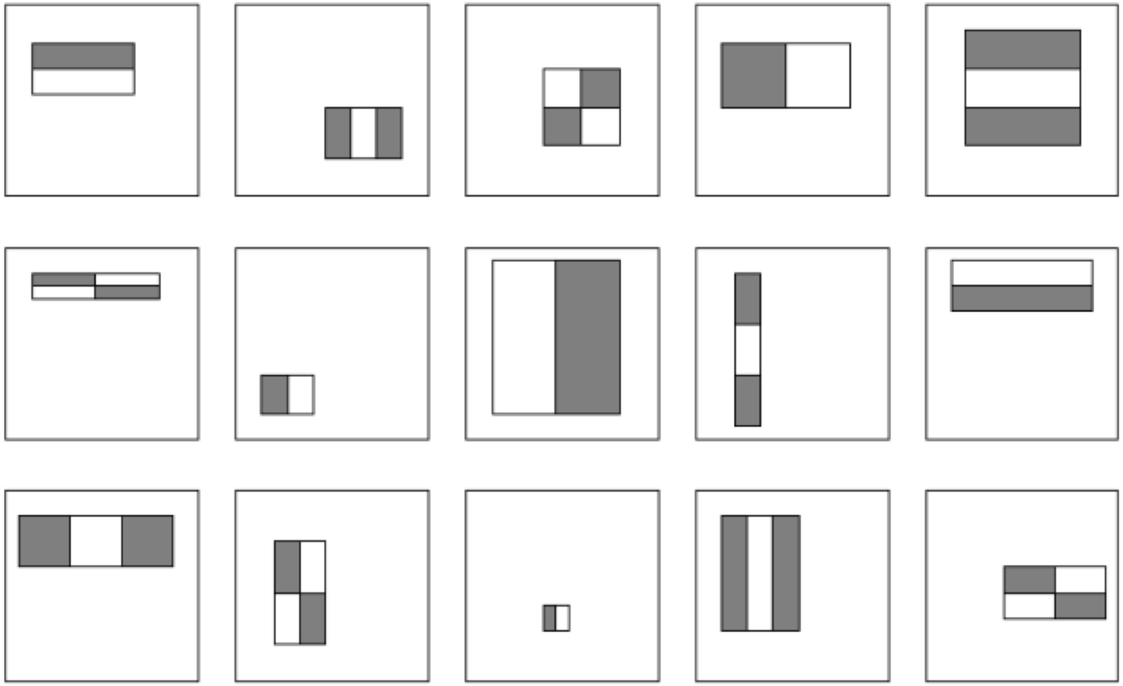

People call them Haar-like features, since similar to 2D Haar wavelets.

*Value =*

*∑ (pixels in white area) –*
*∑ (pixels in black area)*

# Large library of filters

[Viola & Jones, CVPR 2001]



Considering all possible filter parameters: position, scale, and type:

160,000+ possible features associated with each 24 x 24 window

Use AdaBoost both to select the informative features and to form the classifier

# Feature selection

- For a 24x24 detection region, the number of possible rectangle features is ~160,000!

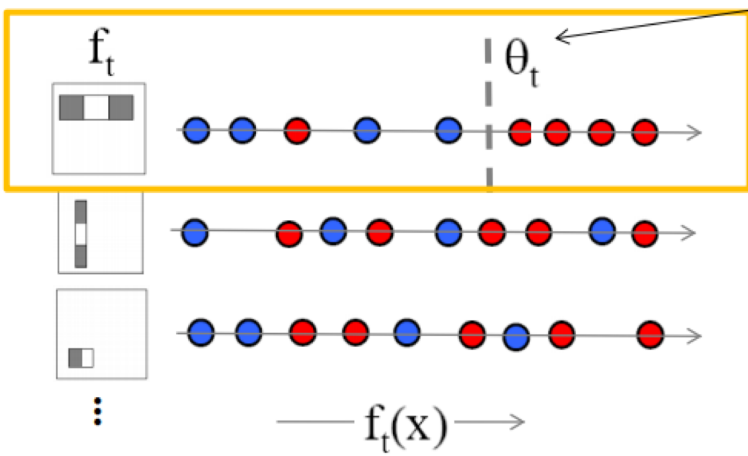
- At test time, it is impractical to evaluate the entire feature set

- **Can we create a good classifier using just a small subset of all possible features?**

- How to select such a subset?

# AdaBoost for feature+classifier selection

- Want to select the single rectangle feature and threshold that best separates positive (faces) and negative (non-faces) training examples, in terms of weighted error.




Outputs of a possible rectangle feature on faces and non-faces.

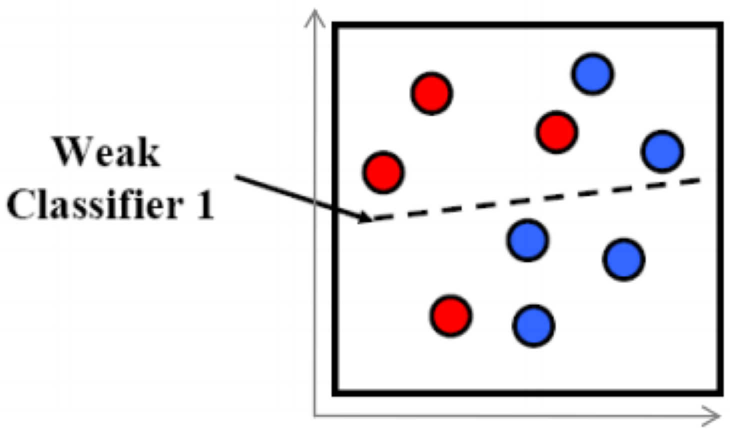
$\theta_t$ is a threshold for classifier $h_t$

Resulting weak classifier:

$$h_t(x) = \begin{cases} +1 & \text{if } f_t(x) > \theta_t \\ 0 & \text{otherwise} \end{cases}$$

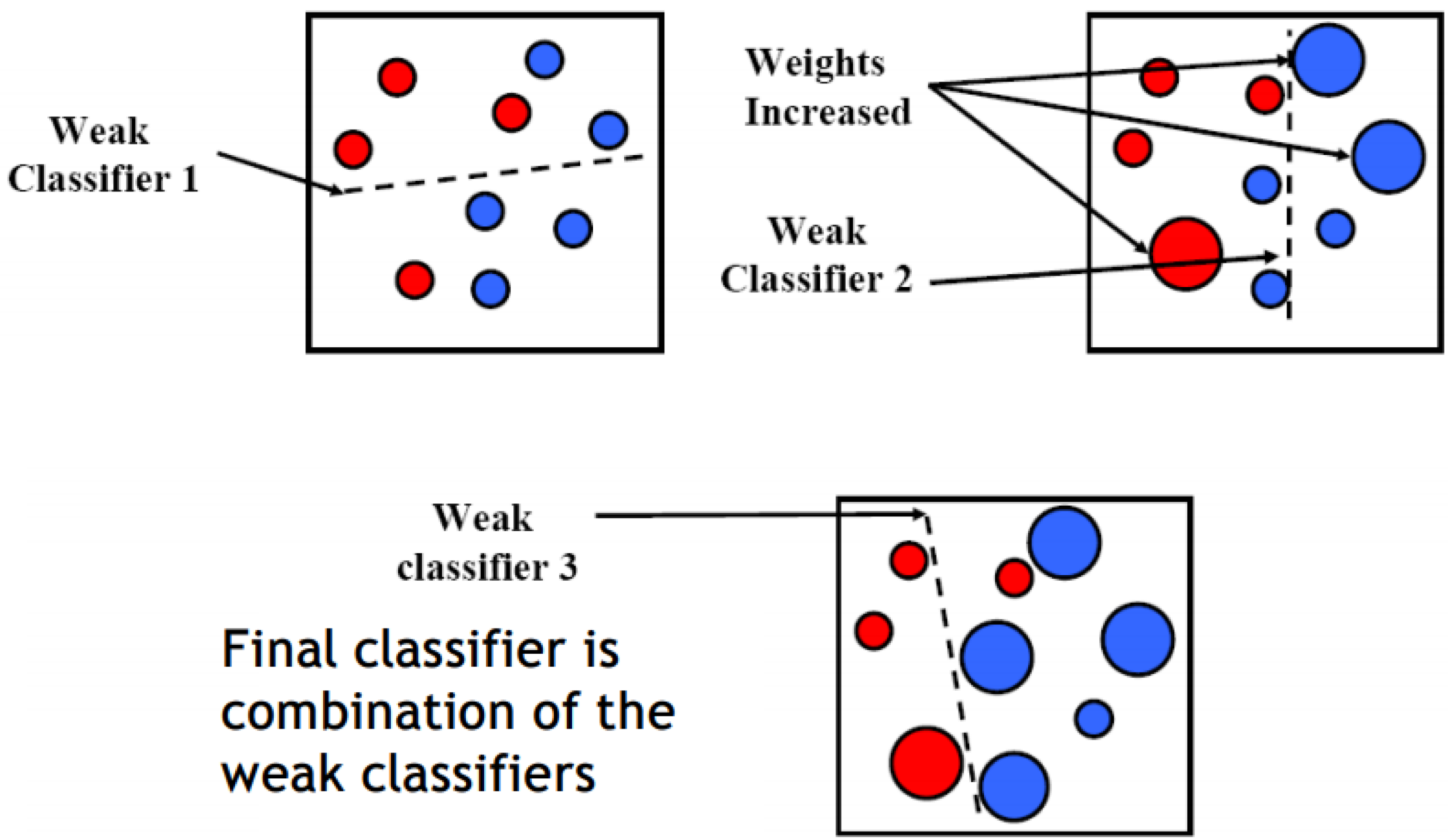
For next round, reweight the examples according to errors, choose another filter/threshold combo.

# AdaBoost: Intuition




- Consider a 2-d feature space with positive and negative examples.
- Each weak classifier splits the training examples with at least 50% accuracy.
- Examples misclassified by a previous weak learner are given more emphasis at future rounds.

# AdaBoost: Intuition

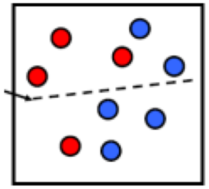



Final classifier is combination of the weak classifiers

# AdaBoost Algorithm modified by Viola Jones

- 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 $w_t$, $\epsilon_j = \sum_i w_i |h_j(x_i) - y_i|$. sum over training samples
  3. Choose the classifier, $h_t$, with the lowest error $\epsilon_t$.
  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 $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.

NOTE: Our code uses equal weights for all samples



$\{x_1, \ldots x_n\}$

For T rounds: meaning we will construct T weak classifiers

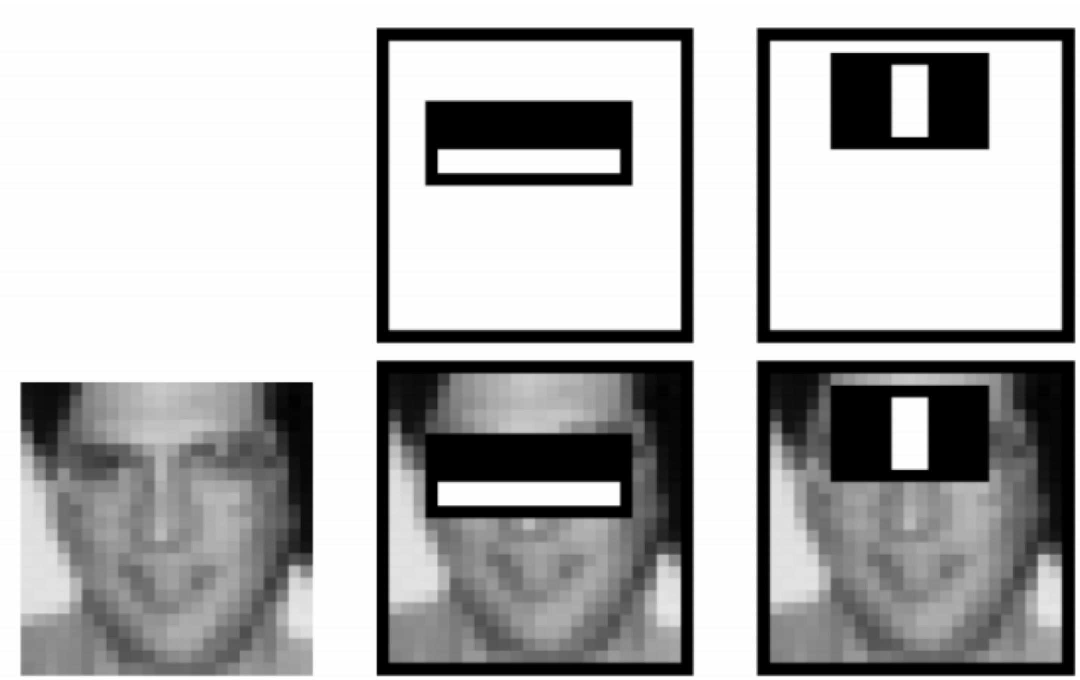
Normalize weights

Find the best threshold and polarity for each feature, and return error.

Re-weight the examples:
Incorrectly classified -> more weight
Correctly classified -> less weight

# Boosting for face detection

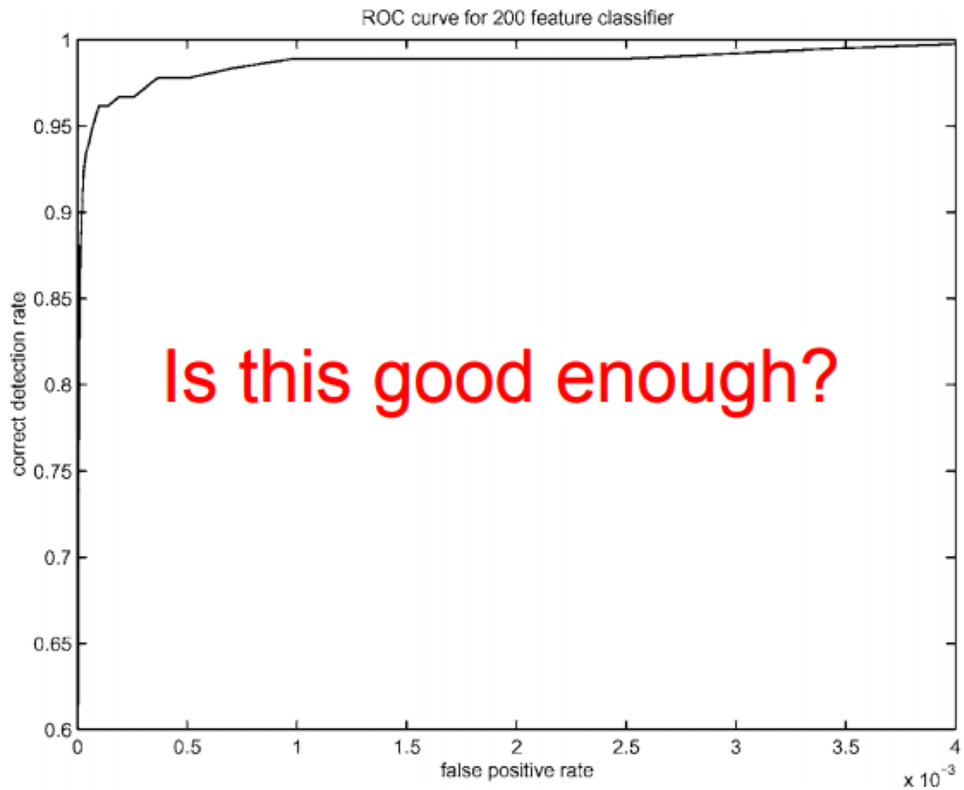
- First two features selected by boosting:



This feature combination can yield 100% detection rate and 50% false positive rate

# Boosting for face detection

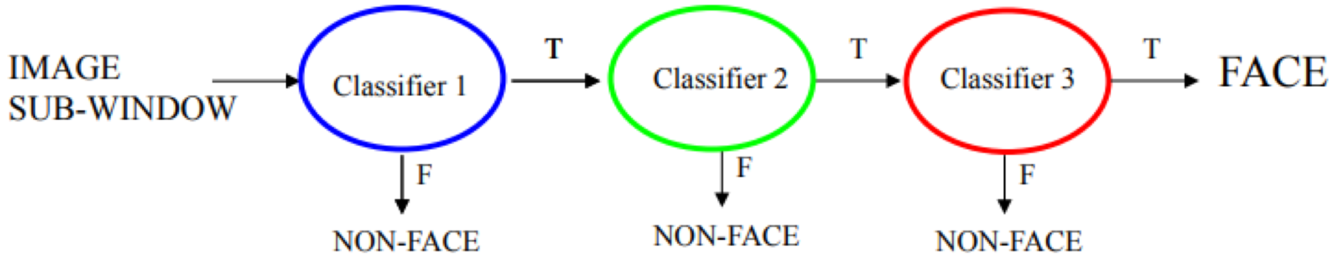
- A 200-feature classifier can yield 95% detection rate and a false positive rate of 1 in 14084




Receiver operating characteristic (ROC) curve

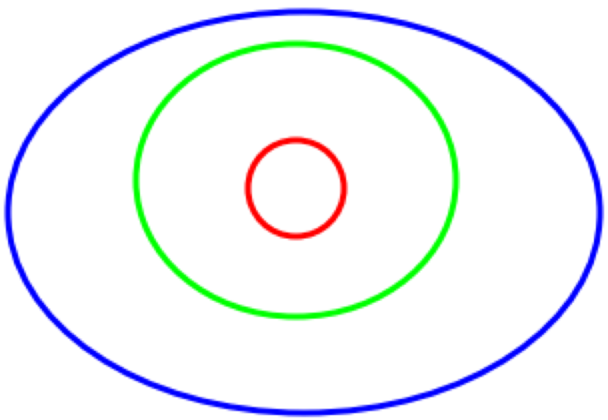
# Attentional cascade (from Viola-Jones)

- We start with simple classifiers which reject many of the negative sub-windows while detecting almost all positive sub-windows
- Positive response from the first classifier triggers the evaluation of a second (more complex) classifier, and so on
- A negative outcome at any point leads to the immediate rejection of the sub-window

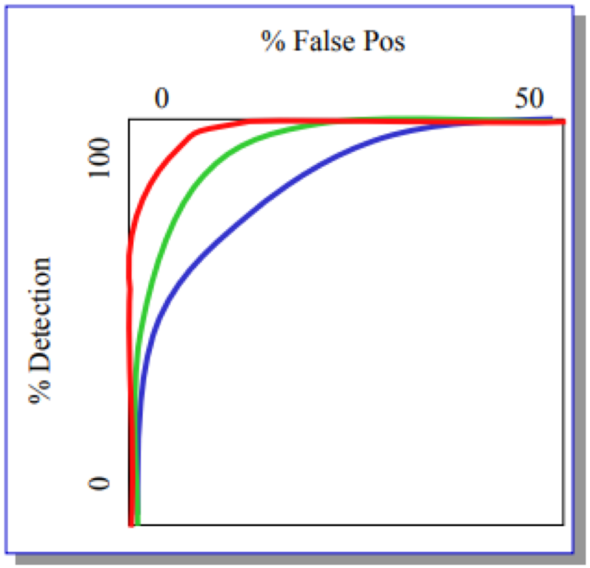


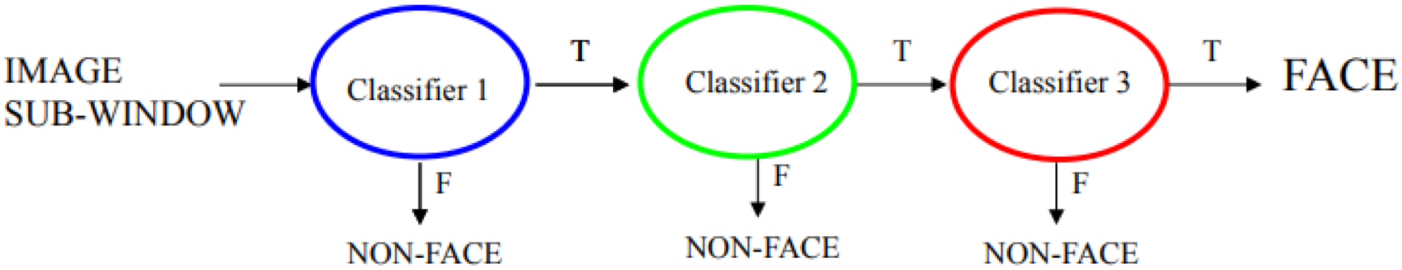
# Attentional cascade

- Chain of classifiers that are progressively more complex and have lower false positive rates:



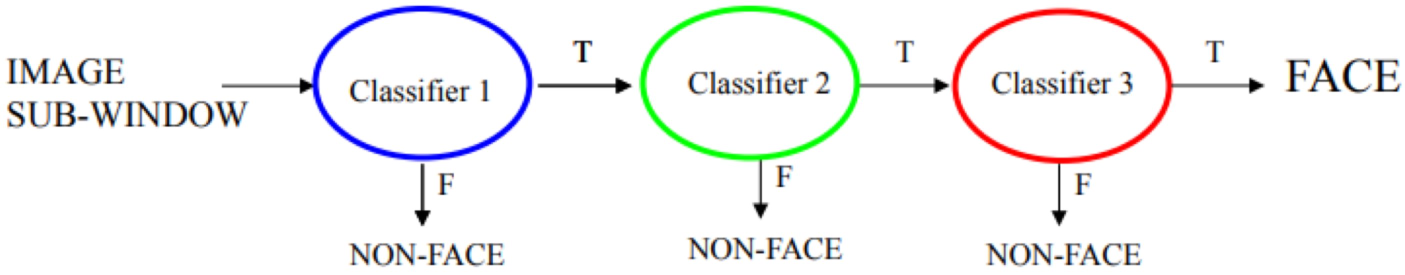
Receiver operating characteristic



# Attentional cascade

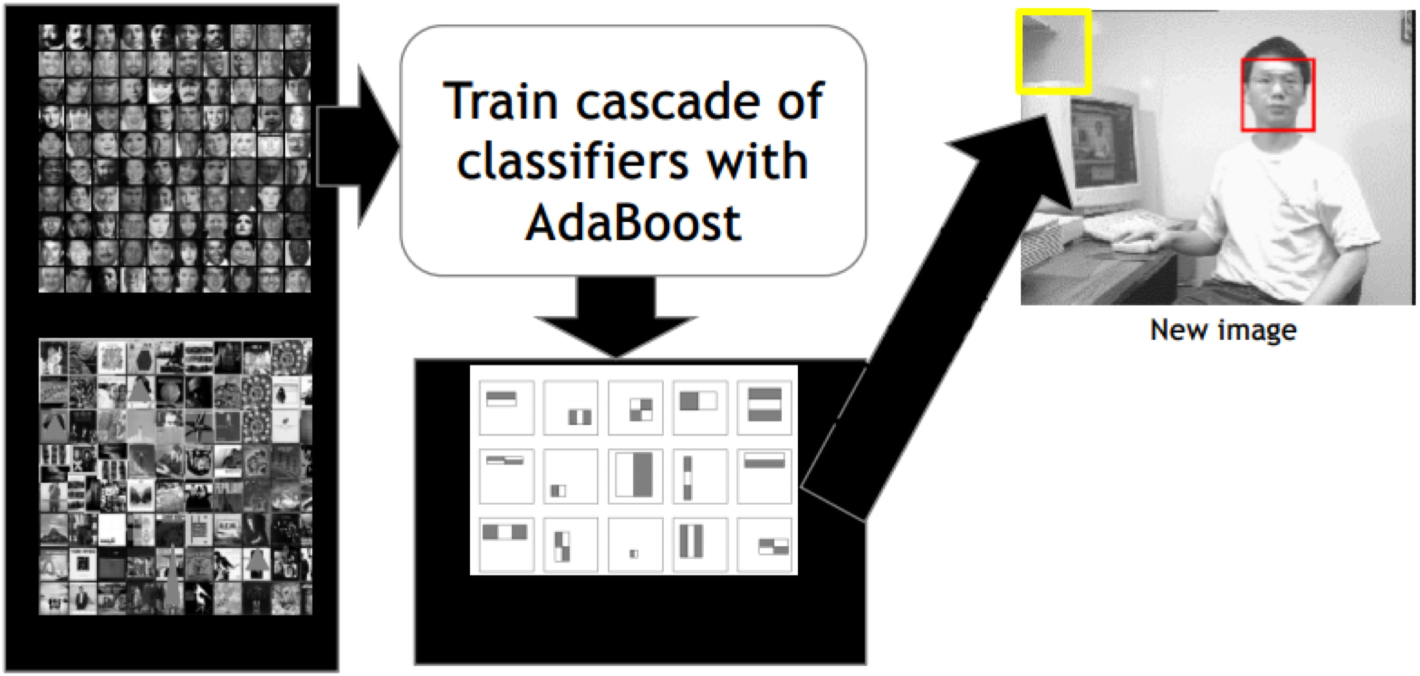
- The detection rate and the false positive rate of the cascade are found by multiplying the respective rates of the individual stages
- A detection rate of 0.9 and a false positive rate on the order of 10^-6 can be achieved by a 10-stage cascade if each stage has a detection rate of 0.99 (0.9910 ≈ 0.9) and a false positive rate of about 0.30.



# Training the cascade

- Set target detection and false positive rates for each stage
- Keep adding features to the current stage until its target rates have been met
  - - Need to lower AdaBoost threshold to maximize detection (as opposed to minimizing total classification error)
  - - Test on a validation set
- If the overall false positive rate is not low enough, then add another stage
- Use false positives from current stage as the negative training examples for the next stage

# Viola-Jones Face Detector: Summary




- Train with 5K positives, 350M negatives
- Real-time detector using 38 layer cascade
- 6061 features in final layer
- [Implementation available in OpenCV: http://www.intel.com/technology/computing/opencv/]

# The implemented system

- Training Data
  - 5000 faces

  – All frontal, rescaled to 24x24 pixels
  - 300 million non-faces

  – 9500 non-face images
  - Faces are normalized

  – Scale, translation

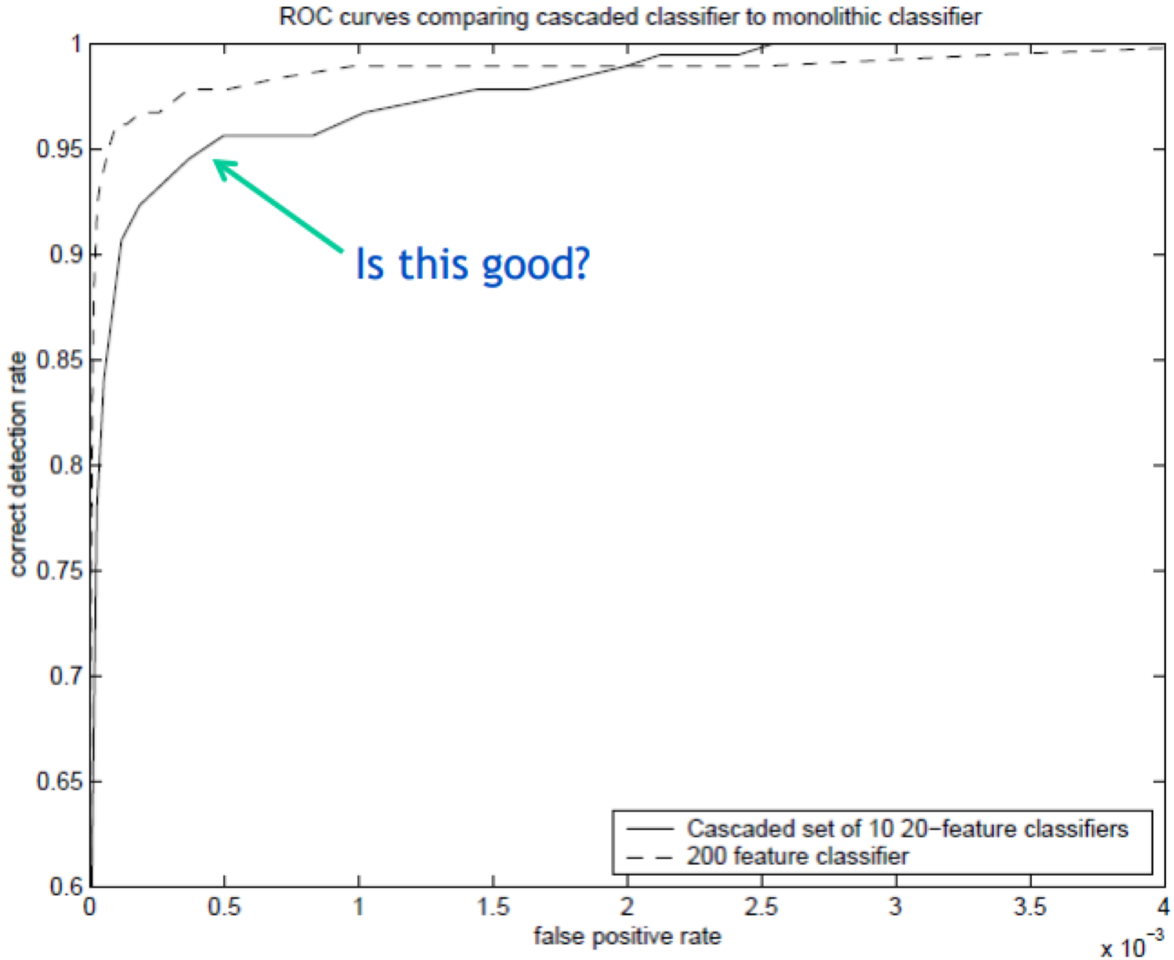
- Many variations

  -Across individuals

  -Illumination

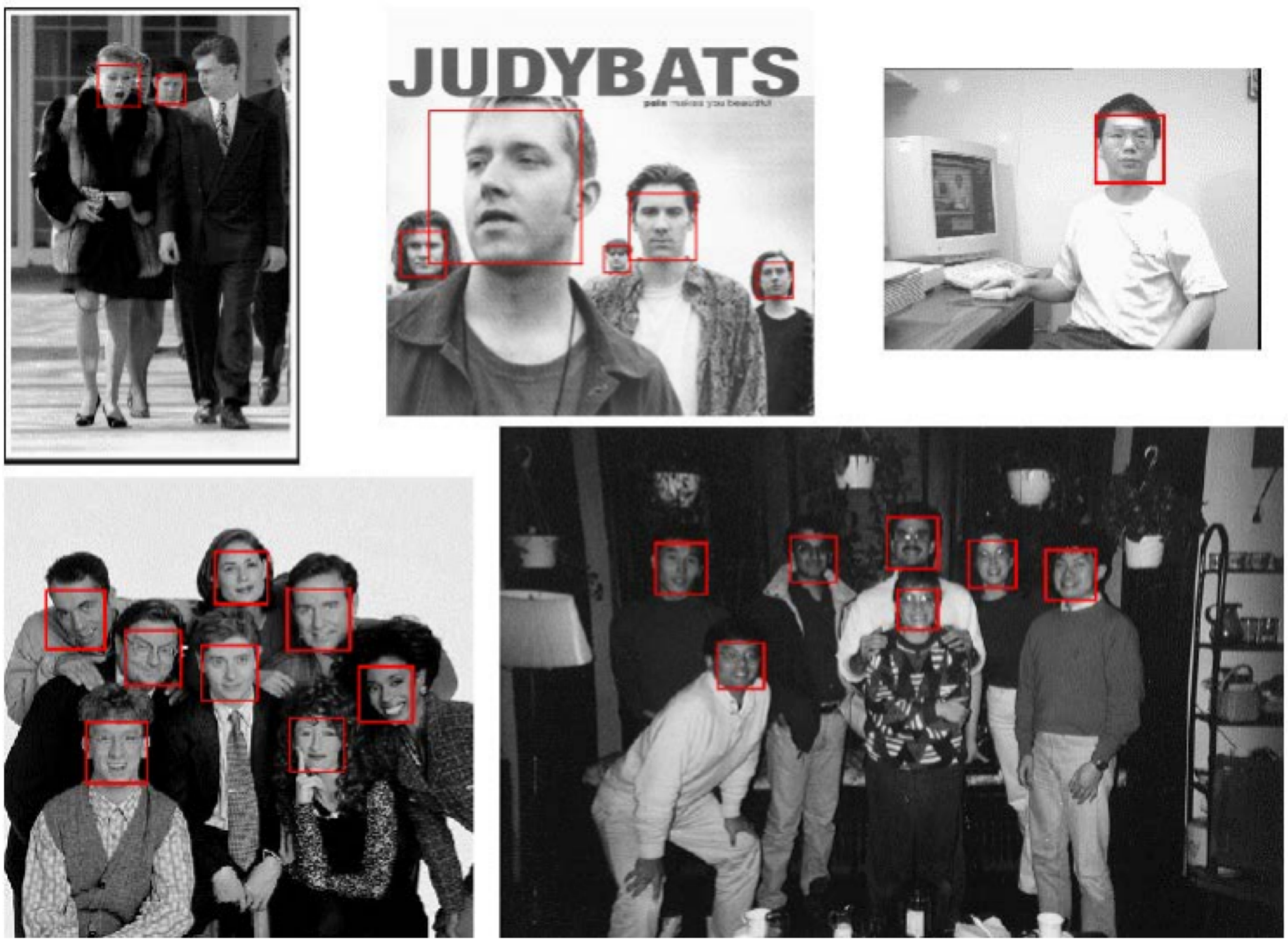
  -Pose

# Performance

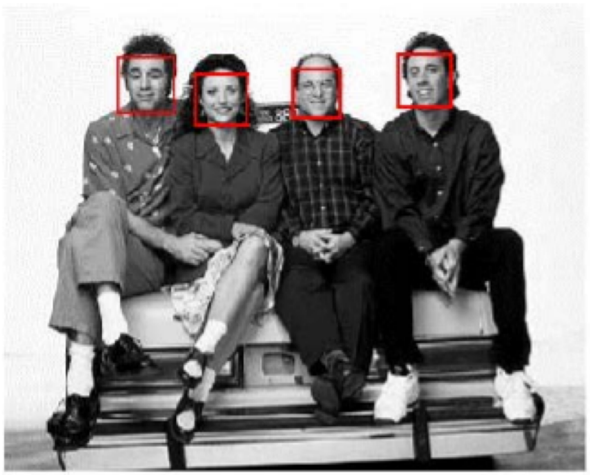


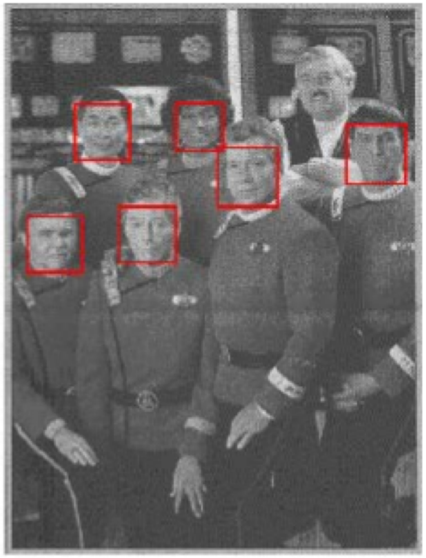

Similar accuracy, but 10x faster

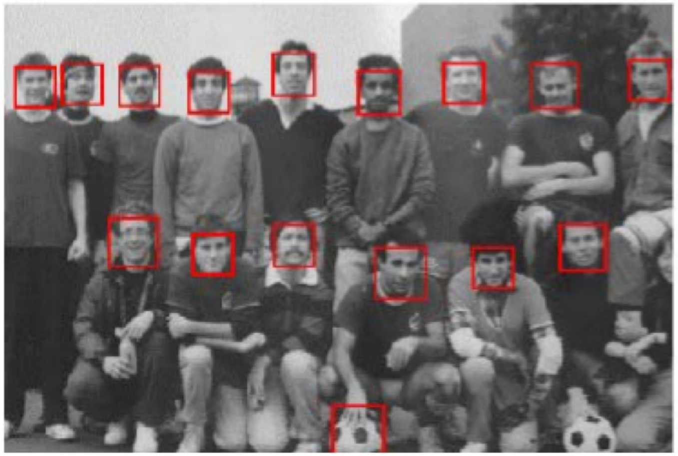
# Viola-Jones Face Detector: Results

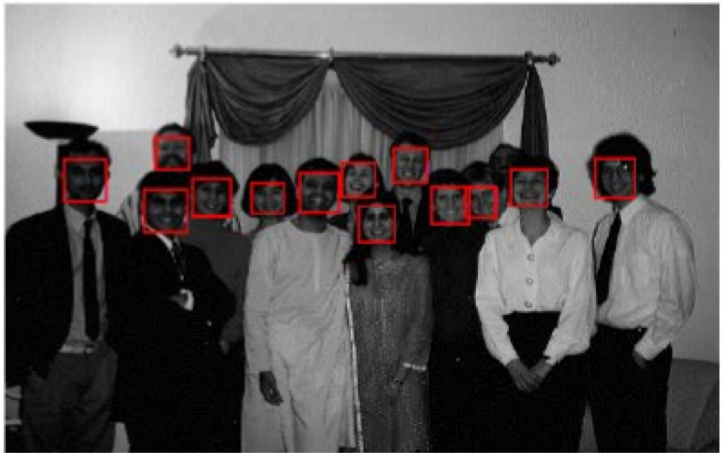


# Viola-Jones Face Detector: Results









# *Q & A*