A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting

By Yoav Freund and Robert E. Schapire

(Updated from ICDM 2006 Presentation by Zhi-Hua Zhou)

Presented by Glenn Rachlin

OUTLINE:

- A little History
- On-line allocation of resources
  - Introduction
  - Problem formulation
  - Hedge(ß) and its bounds
  - Evaluation and example
- Boosting
  - Introduction
  - PAC
  - Motivation for research
  - AdaBoost and its bounds
  - Evaluation and example
  - Face recognition technology
- Summary and comparison between the two algorithms
- Questions for Final exam

BOOSTING: A BACKGROUND

- Significant advantages:
  - Solid theoretical foundation
  - Very accurate prediction
  - Very simple ("just 10 lines of code" [R. Schapire])
  - Wide and successful applications
  - … …
- R. Schapire and Y. Freund wonthe **2003 Godel Prize** (one of the most prestigious awards in theoretical computer science)
  - Prize winning paper (which introduced **AdaBoost**): "A decision theoretic generalization of on-line learning and an application to Boosting," Journal of Computer and System Sciences, 1997, 55: 119-139.

How was AdaBoost born?

- In 1988, M. Kearns and L.G. Valiant posed an interesting question:
  - Whether a "weak" learning algorithm that performs just slightly better than random guess can be "boosted" into an arbitrarily accurate "strong" learning algorithm
  - Or in other words, the key question, whether two complexity classes, "weakly learnable" and "strongly learnable" problems, were equal?

How was AdaBoost born?

- In R. Schapire's MLJ90 paper, Rob said "yes" and gave a proof to the question. The proof is a construction, which is the first Boosting algorithm
- Then, in Y. Freund's Phd thesis (1993), Yoav gave a scheme of combining weak learner by majority voting

But, these algorithms are not very practical

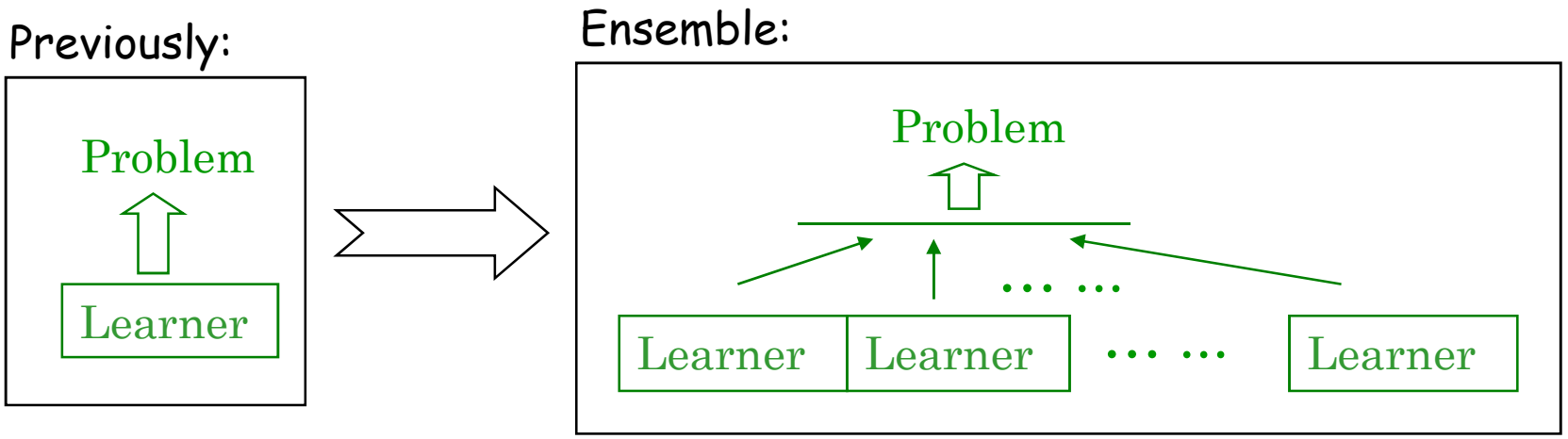
- Later, at AT&T Bell Labs, they published the 1997 paper (in fact the work was done in 1995), which proposed the AdaBoost algorithm, a practical algorithm

USEFUL DEFINITIONS:

- **Training Data**: labeled examples given to a learner
- **Test Data**: examples whose label a learner must predict
- **Final Hypothesis**: the prediction rule a learner forms based on training data to predict on new data
- **Training Error**: the prediction error of the final hypothesis on the training data
- **Generalization Error**: the true prediction error of the final hypothesis on new data.
- **Test Error**: the prediction error of the final hypothesis on the test data (an estimate of the generalization error)

ENSEMBLE LEARNING:

A machine learning paradigm where multiple learners are used to solve the problem



• The generalization ability of the ensemble is usually significantly better than that of an individual learner

• Boosting is one of the most important families of ensemble methods

On-line Allocation of Resources: Introduction

- Problem: "… dynamically apportioning resources among a set of options…"
- The Gambler example.

The Gambler:

A Gambler wants to write a program to accurately predict winners in horse races.

- He collects input data:
  - Input vector:
    - Odds
    - Dry/muddy
    - Jockey
  - Output:
    - Win or lose
- Evaluation:
  - He discovers easy to find certain "rules of thumb" for races that dictate results.
  - Hard to find one particular rule that works for multiple circumstances.

The Gambler:

- The Gambler decides to use **arcing** to find some simple solution to his combined rules.
- Arcing: "reweighting and combining"
  - reusing or selecting data in order to improve classification, a method of boosting.

Repeat T times:

1. Examine a small sample of races.
2. Derive rough rule of thumb.
   - Ex. Bet one horse with most favorable odds
3. Select new sample, derive new rule of thumb.
   - If track is muddy, bet on lightest horse, otherwise
   - Choose randomly

end

Questions to think about:

- How can we choose samples?
  - Select multiple random samples?
  - Concentrate only on errors?
- How can we combine "rules of thumb" into single rule?
  - Is it even possible?

On-line Allocation of Resources: Problem Formulation

The on-line allocation model:

- **Allocation agent:** A - the gambler
- **A certain strategy:** i - how one friend behaves
- **# of options/strategies:** {1,2,3, … ,N} - the # of friends he has
- **# of time steps:** {1,2,3, … ,T} - the # of races
- **distribution over strategies:** p^t -how much money he spends
- **the loss altogether:** l^t -all the lost money

$$L_A - min_i L_i$$

$$L_A = \sum_{t=1}^{T} p^t \cdot l^t \qquad L_i = \sum_{t=1}^{T} l_i^t$$

On-line Allocation of Resources: Hedge(b)

- Assumptions:
  - The loss suffered by any strategy be bounded
  - All weights be nonnegative
  - Initial weights sum up to 1 (optional)

Algorithm Hedge (β)

- ***Parameters:*** $\beta \in [0,1]$
- ***initial weight vector:*** $\omega^1 \in [0,1]^N$ with $\sum_{i=1}^{N} w_i^1 = 1$
- ***number of trials T***

Do for t = 1,2, …, T

1. ***Choose allocation from environment*** $p^t = \frac{w^t}{\sum_{i=1}^{N} w_i^1}$
2. ***Receive loss vector*** $l^t \in [0,1]^N$
3. ***Suffer loss*** $p^t \cdot l^t$
4. ***Set new weights vector to be*** $w_i^{t+1} = w_i^t \cdot \beta^{l_i^t}$

Goal: minimize difference between expected total loss and minimal total loss of repeating one action (ex; gambling).

On-line Allocation of Resources: Bounds of Hedge(b)

$$L_{Hedge(\beta)} \leq c\ min_i\ L_i + a\ ln\ N$$

$$\frac{L_{Hedge(\beta)}}{T} \leq min_i \frac{L_i}{T} + \frac{ln\ N}{T}$$

On-line Allocation of Resources: Evaluation

- The authors show that the Hedge(β) algorithm "yield[s] bounds that are slightly weaker in some cases, [than those produced by the algorithm proposed by Littlestone and Warmuth, 1994] but applicable to a considerably more general class of learning problems."
  - Not only binary decisions
  - Not only discrete loss

On-line Allocation of Resources: Example

"In many cases ... superior algorithms or analyses are known. Although weaker in specific cases, it should be emphasized that our results are far more general, and can be applied in settings that exhibit considerably less structure, such as the horse-racing example..."

Some more Terms:

- Boosting is a method of improving the effectiveness of predictors.
- Boosting relies on the existence of weak learners.
- A weak learner is a "rough and moderately inaccurate" predictor, but one that can predict better than chance.
- Boosting shows the strength of weak learnability.

BOOSTING: INTRODUCTION

- Aim: ".. converting a weak learning algorithm that performs just slightly better than random guessing into one with arbitrarily high accuracy."
- Example: Constructing an expert computer program
- Two problems:
  - Choosing data
  - Combining rules
- "Boosting refers to this general problem of producing a very accurate prediction rule by combining rough and moderately inaccurate rules-of-thumb."

BOOSTING: THE NOTION OF PAC

- Classification problem: set of lables Y contains 2 or more elements.
- Two givens:
  - Probability
  - Precision
- Two kinds of PAC learners:
  - Strong
  - Weak
- Remember our aim: Transform a weak learner into a strong one!

Boosting: Motivation for Research

- Previous algorithms by the same authors "work by calling a given weak learning algorithm WeakLearn multiple times, each time presenting it with a different distribution [of examples], and finally combining all the generated hypotheses into a single hypothesis."
- Problems
  - Too much has to be known in advance
  - Improvement of the overall performance depends on the weakest rules

Rules for Boosting

1. Set all weights of training examples equal.
2. Train a weak learner on the weighted examples.
3. See how well the weak learner performs on data and give it a weight based on how well it did.
4. Re-weight training examples and repeat.
5. When done, predict by voting by majority.

More formally:

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 $\alpha_t \in \mathbb{R}$.
- Update:

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

where Z_t is a normalization f
tion).

AdaBoost

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

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

the weights of incorrectly classified examples are increased so that the base learner is forced to focus on the hard examples in the training set

Output the final classifier:

$$H(x) = \text{sign} \left(\sum_{t=1}^{T} \alpha_t h_t(x) \right).$$

AdaBoost: a background

- 1990 – Boost-by-majority algorithm (Freund)
- 1995 – AdaBoost (Freund &Schapire)
- 1997 – Generalized version of AdaBoost (Schapire& Singer)
- 2001 – AdaBoost in Face Detection (Viola & Jones)

AdaBoost Terms

- Learner = Hypothesis = Classifier
- Weak Learner: < 50% error over any distribution
- Strong Classifier: thresholded linear combination of weak learner outputs

Properties of AdaBoost:

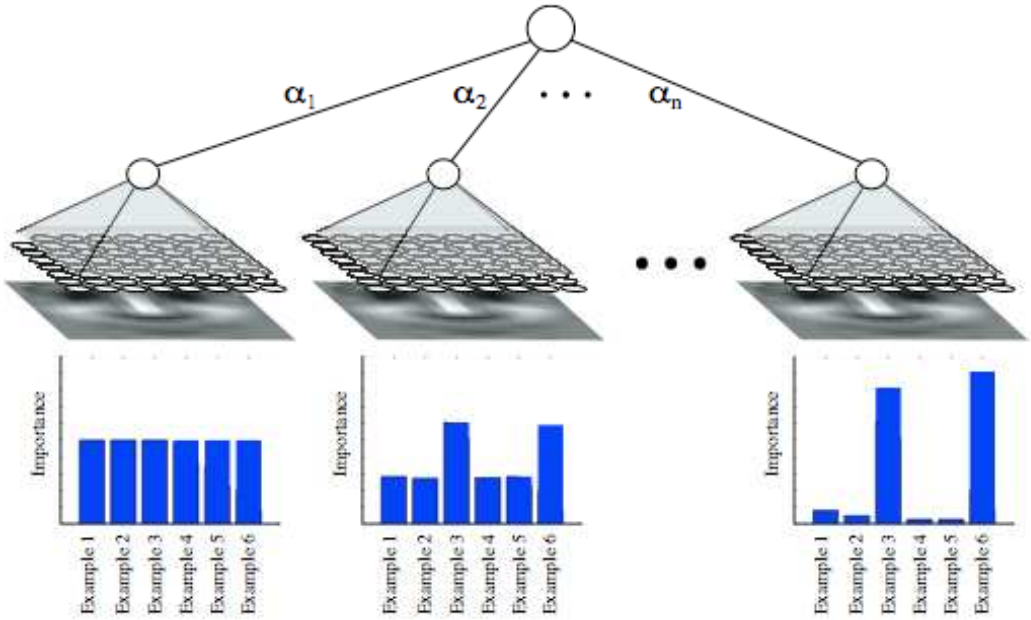
- AB is a linear classifier with all its desirable properties.
- AB output converges to the logarithm of likelihood ratio.
- AB has good generalization properties.
- AB is a feature selector with a principled strategy (minimization of upper bound on empirical error).
- AB close to sequential decision making (it produces a sequence of gradually more complex classifiers).

AdaBoost: Adaptive Boosting

- Instead of sampling, re-weight
  - Previous weak learner has only 50% accuracy over new distribution
- Can be used to learn weak classifiers
- Final classification based on weighted vote of weak classifiers

AdaBoost:

If the underlying classifiers are linear networks, then AdaBoost builds multilayer perceptrons one node at a time.



However, the underlying classifier can be anything, decision trees, neural networks, etc…

THE ADABOOST ALGORITHM:

Initialize the weight vector $w_i^1 = D(i)\ for\ i = 1, 2, ..., N$

Do for t = 1,2, ..., T

1. Set $p^t = \frac{w^t}{\sum_{i=1}^{N} w_i^1}$

2. Call Weak Learn, providing it with the distribution p^t; get back a hypothesis $h_t : X \to [0, 1]$

3. Calculate the error of $\epsilon_t = \sum_{i=1}^{N} p_i^t \mid h_t(x_i) - y_i \mid$

4. Set $\beta_t = \frac{\epsilon_t}{1 - \epsilon_t}$

5. Set new weights vector to be $w_i^{t+1} = w_i^t \beta_t^{1-|h_t(x_i)-y_i|}$

Output the hypothesis

$$h_f(x) = \left\{ \begin{array}{c} 1\ if\ \sum_{t=1}^{T} (\log \frac{1}{\beta_t})\ h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \log \frac{1}{\beta_t} \\ 0\ otherwise \end{array} \right\}$$

THEORETICAL PROPERTIES:

- Y. Freund and R. Schapire[JCSS97] have proved that the training error of AdaBoost is bounded by:

$$\prod_t Z_t = \prod_t \left[2\sqrt{\epsilon_t(1-\epsilon_t)}\right] = \prod_t \sqrt{1-4\gamma_t^2} \leq \exp\left(-2\sum_t \gamma_t^2\right)$$

where $\gamma_t = 1/2 - \epsilon_t$

Thus, if each base classifier is slightly better than random so that $\gamma_t \geq \gamma$ for some $\gamma > 0$ then the training error drops exponentially fast in T since the above bound is at most $e^{-2T\gamma^2}$

THEORETICAL PROPERTIES (CON'T…):

- Y. Freund and R. Schapire[JCSS97] have tried to bound the generalization error as:

  where $\hat{\Pr}[\cdot]$ denotes empirical probability on training sample, s is the sample size, learner

$$\hat{\Pr}[H(x) \neq y] + \tilde{O}\left(\sqrt{\frac{Td}{m}}\right)$$

The above bounds suggest that Boosting will overfit if T is large. However, empirical studies show that Boosting oftendoes not overfit

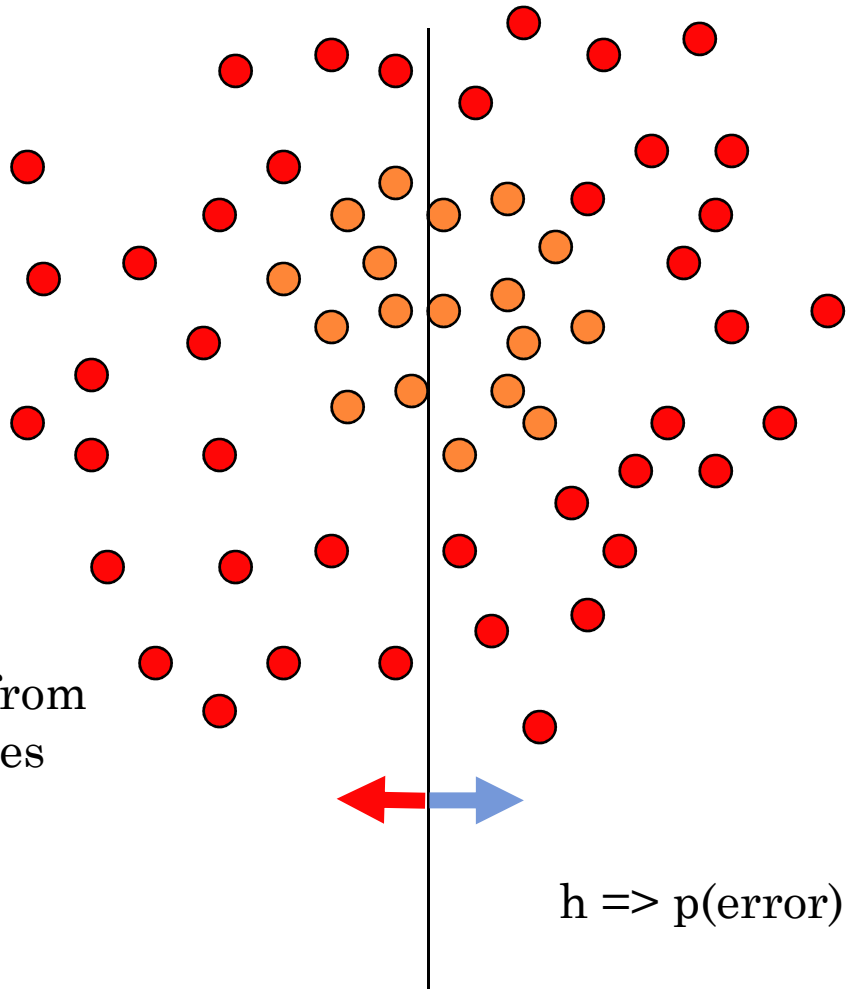
- R. Schapire et al. [AnnStat98] gave a margin-based bound:

  $$\hat{\Pr}[\text{margin}_f(x,y) \leq \theta] + \tilde{O}\left(\sqrt{\frac{d}{m\theta^2}}\right)$$ for any $\theta > 0$ with high probability

  where

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

TOY EXAMPLE – TAKEN FROM ANTONIO TORRALBA @MIT



Each data point has

a class label:

$$y_t = \begin{cases} +1 & (\bullet) \\ -1 & (\circ) \end{cases}$$

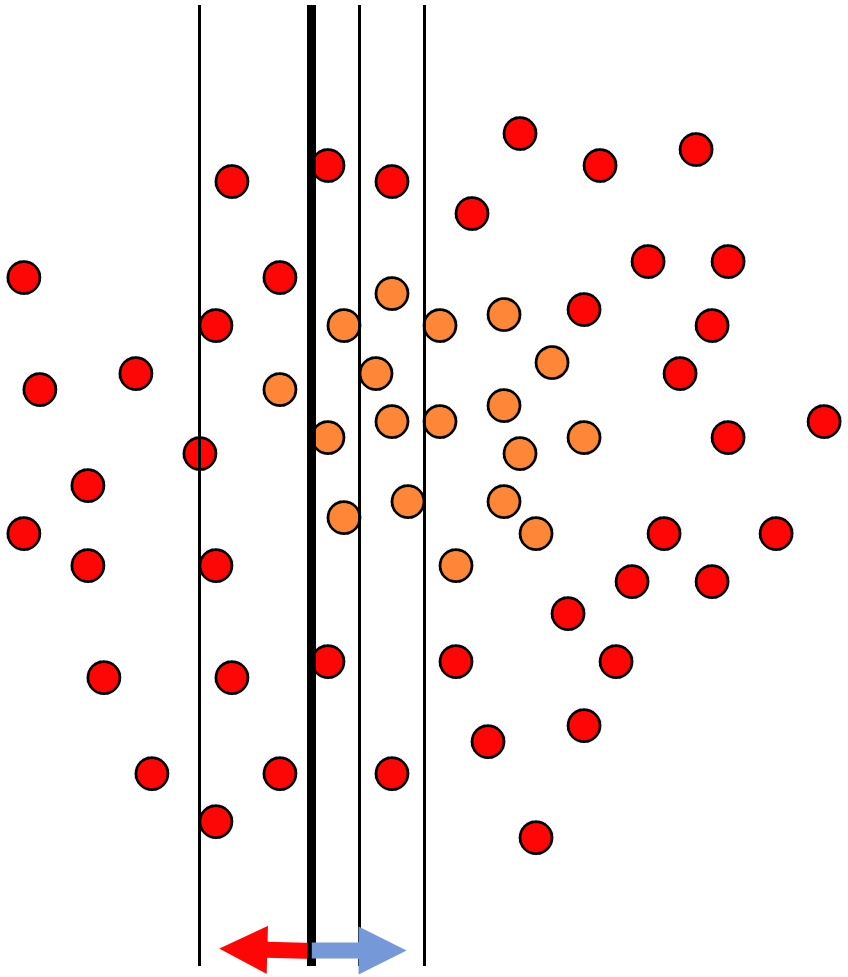
and a weight:

$$w_t = 1$$

Weak learners from the family of lines

h => p(error) = 0.5 it is at chance

TOY EXAMPLE



Each data point has

a class label:

$$y_t = \begin{cases} +1 & (\bullet) \\ -1 & (\circ) \end{cases}$$

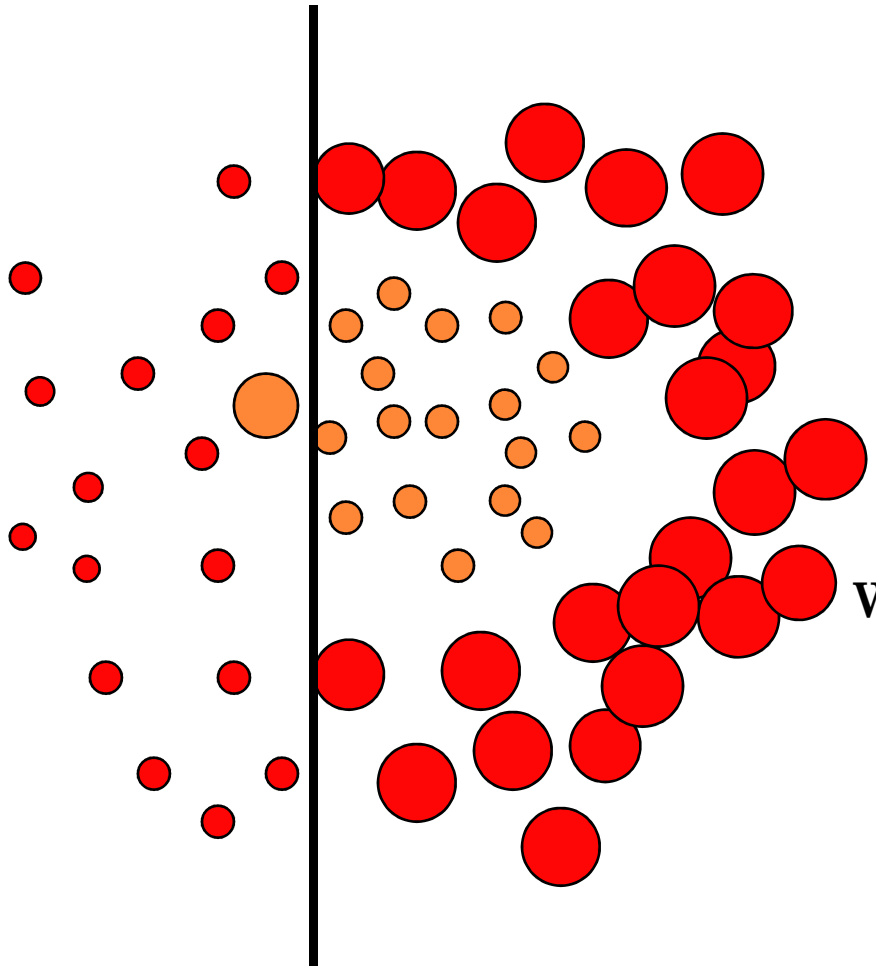
and a weight:

$$w_t = 1$$

This one seems to be the best

This is a '**weak classifier**': It performs slightly better than chance.

TOY EXAMPLE



Each data point has

a class label:

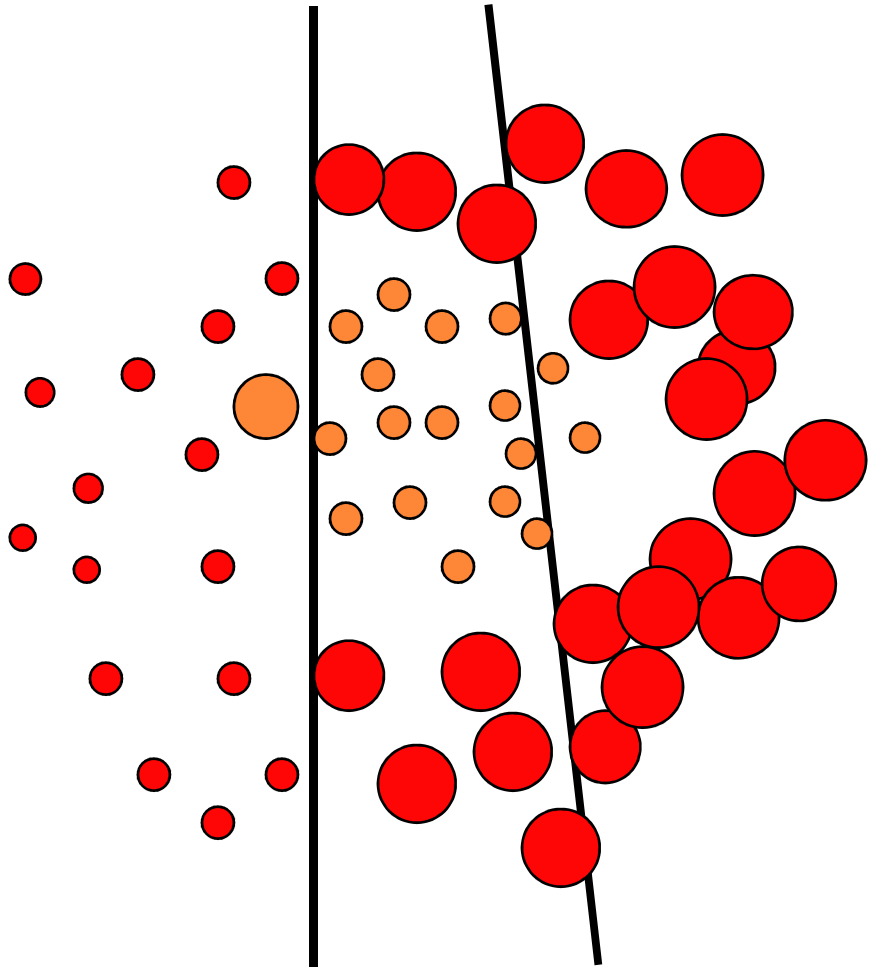
$$y_t = \begin{cases} +1 & (\bullet) \\ -1 & (\circ) \end{cases}$$

We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t H_t\}$$

We set a new problem for which the previous weak classifier performs at chance again

TOY EXAMPLE



Each data point has

a class label:

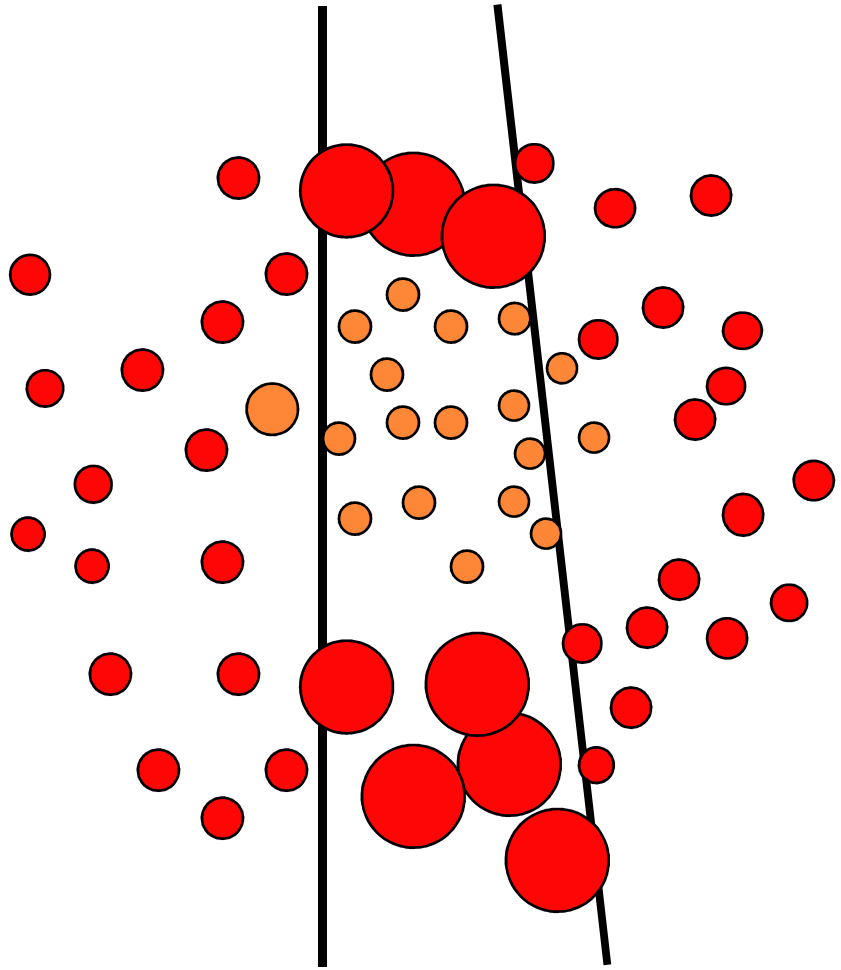
$$y_t = \begin{cases} +1 & (\bullet) \\ -1 & (\circ) \end{cases}$$

We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t H_t\}$$

We set a new problem for which the previous weak classifier performs at chance agai

TOY EXAMPLE



Each data point has

a class label:

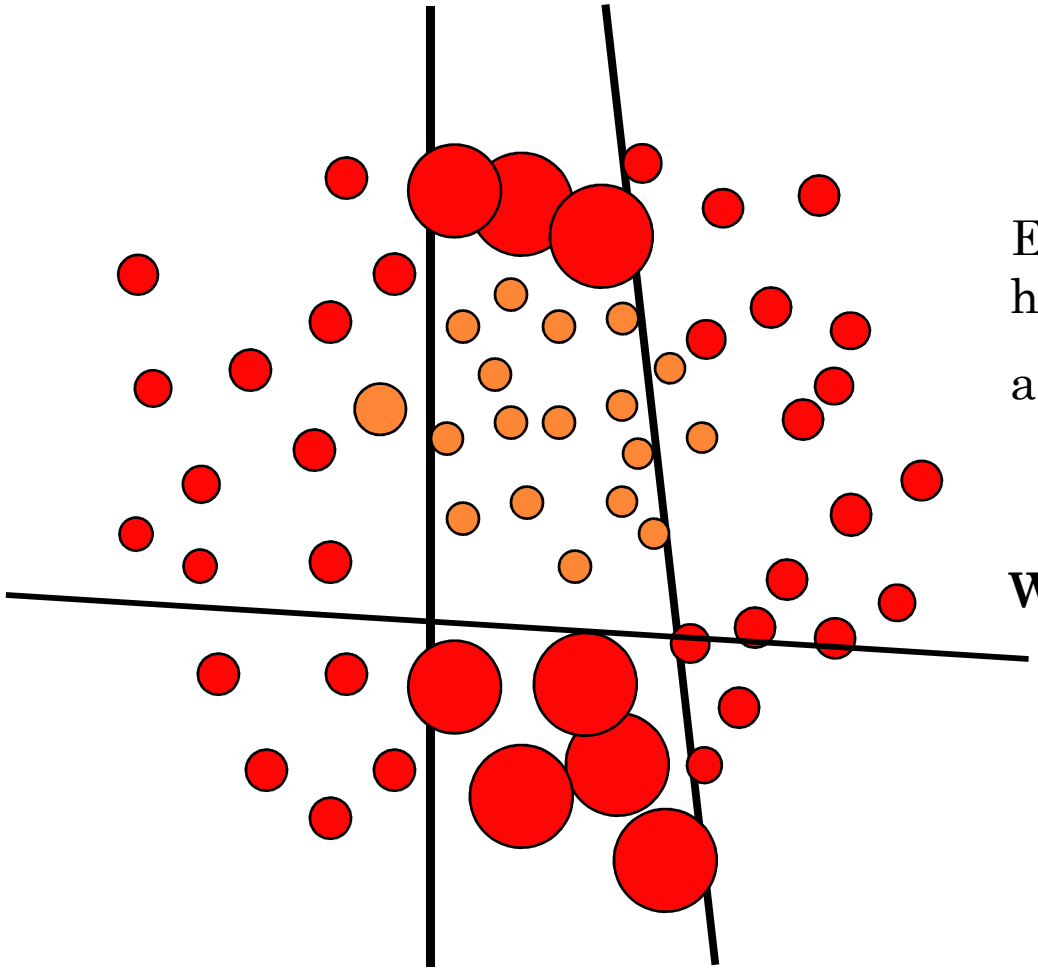
$$y_t = \begin{cases} +1 & (\bullet) \\ -1 & (\circ) \end{cases}$$

We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t H_t\}$$

We set a new problem for which the previous weak classifier performs at chance agai

TOY EXAMPLE



Each data point has

a class label:

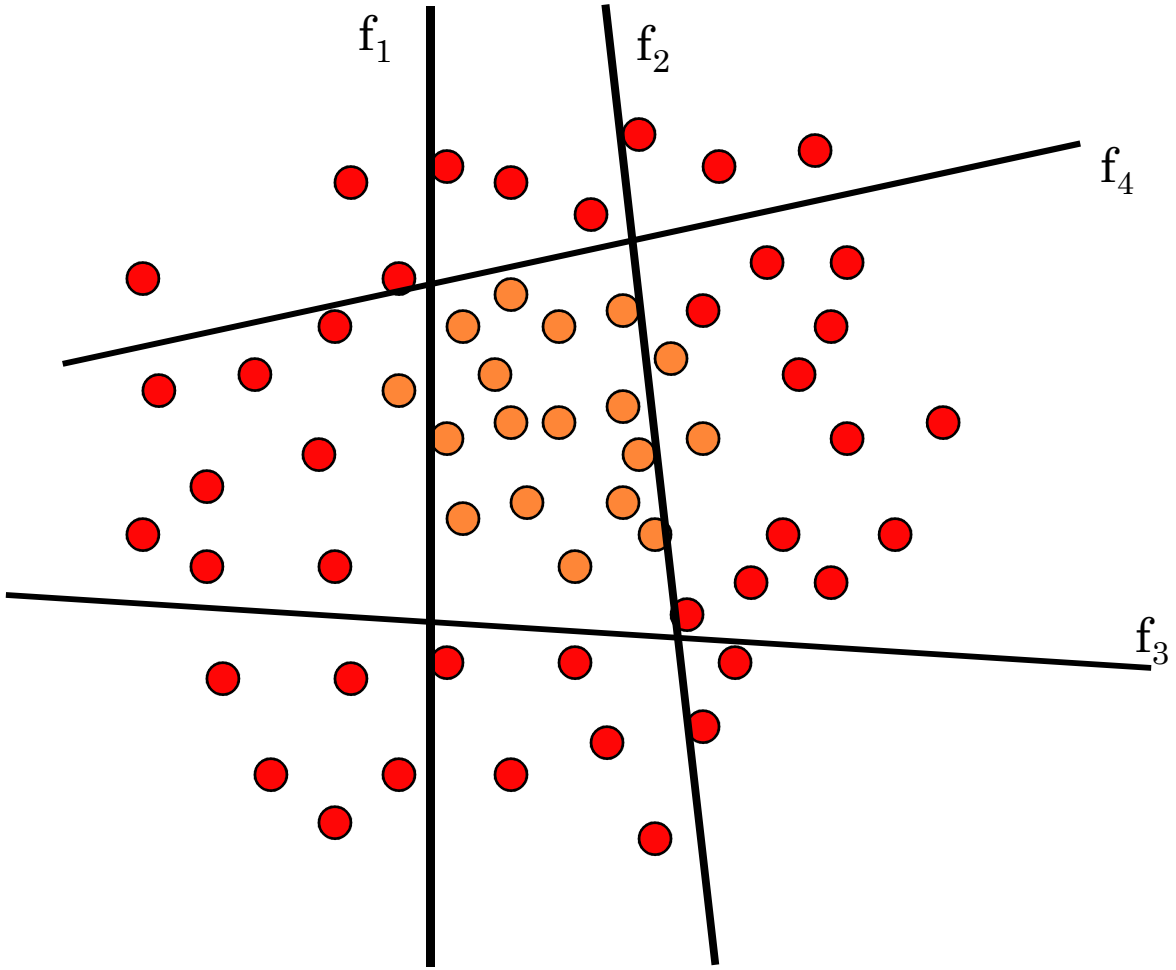
$$y_t = \begin{cases} +1 & (\bullet) \\ -1 & (\circ) \end{cases}$$

We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t H_t\}$$

We set a new problem for which the previous weak classifier performs at chance agai

TOY EXAMPLE



The strong (non- linear) classifier is built as the combination of all the weak (linear) classifiers.

Formal Procedure of AdaBoost

- given training set $(x_1, y_1), \ldots, (x_m, y_m)$
- $y_i \in \{-1, +1\}$ correct label of instance $x_i \in X$
- for $t = 1, \ldots, T$:
  - construct distribution D_t on $\{1, \ldots, m\}$
  - find weak classifier ("rule of thumb")

    $$h_t : X \to \{-1, +1\}$$

    with small error ϵ_t on D_t:

    $$\epsilon_t = \Pr_{D_t}[h_t(x_i) \neq y_i]$$
- output final classifier H_{final}

Procedure of Adaboost:

- constructing D_t:
  - $D_1(i) = 1/m$
  - given D_t and h_t:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } y_i = h_t(x_i) \\ e^{\alpha_t} & \text{if } y_i \neq h_t(x_i) \end{cases}$$

$$= \frac{D_t(i)}{Z_t} \exp(-\alpha_t \, y_i \, h_t(x_i))$$

where Z_t = normalization constant

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right) > 0$$

- final classifier:
  - $H_{\text{final}}(x) = \text{sign}\left(\sum_t \alpha_t h_t(x) \right)$

Error on Training Set:

- Theorem:
  - write ϵ_t as $1/2 - \gamma_t$
  - then

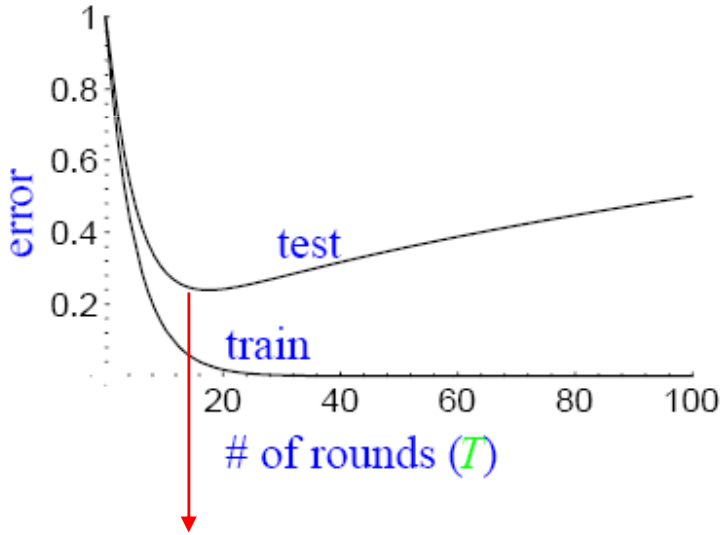
$$\text{training error}(H_{\text{final}}) \leq \exp\left(-2 \sum_t \gamma_t^2\right)$$

- so: if $\forall t : \ \gamma_t \geq \gamma > 0$
  then $\text{training error}(H_{\text{final}}) \leq e^{-2\gamma^2 T}$
- AdaBoost is adaptive:
  - does not need to know γ or T a priori
  - can exploit $\gamma_t \gg \gamma$

Proof later on black board if anyone interested and time permits

But we are NOT interested in Training set

- Will Adaboost screw up with a fat complex classifier finally?

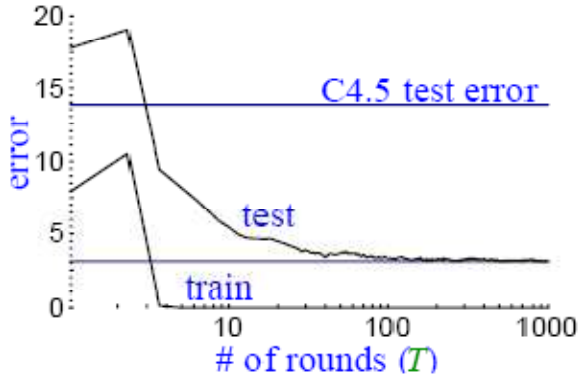


Occam's razor – simple is the best

Over fitting

Shall we stop before over fitting? If only over fitting happens.

Actual Typical Run



(boosting C4.5 on "letter" dataset)

- test error does not increase, even after 1000 rounds
  - (total size > 2,000,000 nodes)
- test error continues to drop even after training error is zero!

	# rounds		
	5	100	1000
train error	0.0	0.0	0.0
test error	8.4	3.3	3.1

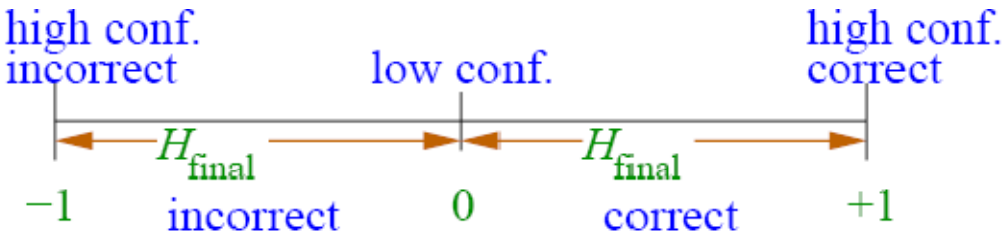
AN EXPLANATION BY MARGIN

- This margin is not the margin in SVM

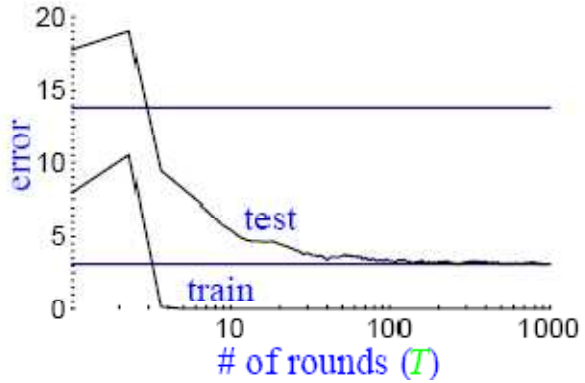
  - key idea:
    - training error only measures whether classifications are right or wrong
    - should also consider confidence of classifications
  - can write: $H_{\text{final}}(x) = \text{sign}(f(x))$

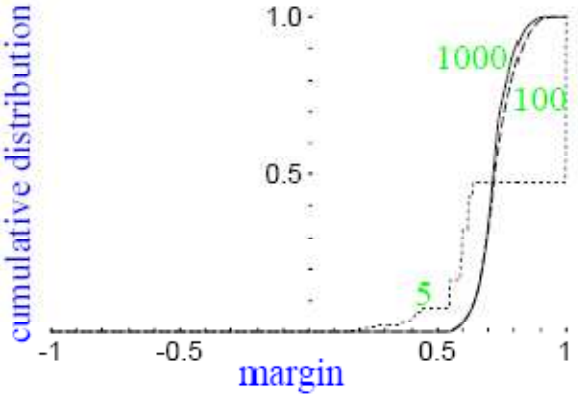
    where $f(x) = \frac{\sum_t \alpha_t h_t(x)}{\sum_t \alpha_t} \in [-1, +1]$
  - define margin of example (x, y) to be $y\ f(x)$

    = measure of confidence of classifications



Margin Distribution





	# rounds		
	5	100	1000
train error	0.0	0.0	0.0
test error	8.4	3.3	3.1
% margins ≤ 0.5	7.7	0.0	0.0
minimum margin	0.14	0.52	0.55

Although final classifier is getting larger, margins are still increasing

Final classifier is actually getting to simpler classifer

Practical Advantages of AdaBoost:

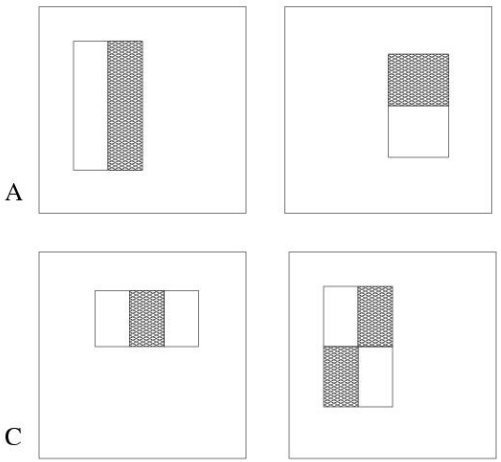
- Simple and easy to program.
- No parameters to tune (except T).
- Effective, provided it can consistently find rough rules of thumb.
  - Goal is to find hypothesis barely better than guessing.
- Can combine with any (or many) classifiers to find weak hypotheses: neural networks, decision trees, simple rules of thumb, nearest-neighbor classifiers, …

APPLICATION:

AdaBoost and its variants have been applied to diverse domains with great success. Here I only show one example

P. Viola & M. Jones [CVPR'01] combined AdaBoost with a cascade process for face detection

They regarded rectangular features as weak classifiers



$$h(x, f, p, \theta) = \begin{cases} 1 & \text{if } pf(x) < p\theta \\ 0 & otherwise \end{cases}$$

f = feature

p = polarity {+1,-1}

θ = threshold

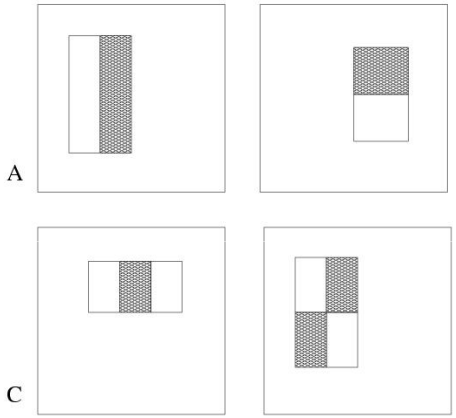
APPLICATION:

By using AdaBoost to weight the weak classifiers, they got two very intuitive features for face detection

In order to get high accuracy as well as high efficiency, they used a cascade process (which is beyond the scope of this talk)

Finally, a very strong face detector: On a 466MHz SUN machine, 384×288 image costed only 0.067 seconds!(in average, only 8 features needed to be evaluated per image)

Application:



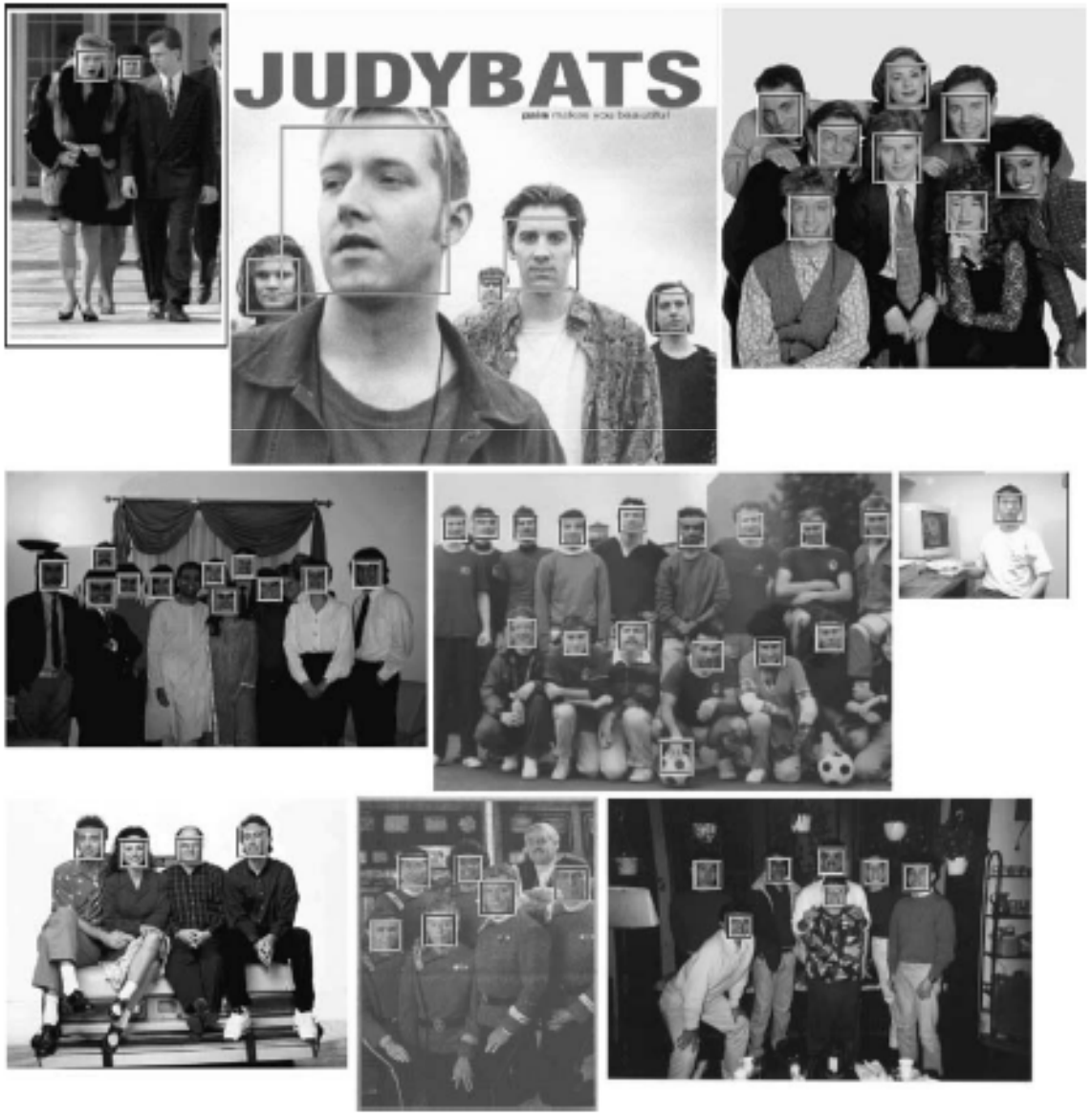
Rectangular feature types:

- *two-rectangle feature* (horizontal/vertical)
- *three-rectangle feature*
- *four-rectangle feature*

Using a 24x24 pixel base detection window, with all possible combinations of orientation, location and scale of these feature types the full set of features has 49,396 features.

The motivation behind using rectangular features, as opposed to more expressive steerable filters is their extreme computational efficiency.

APPLICATION:



Where are we going?

- New learning algorithms?
- Better image descriptors!
- Probably they need to be learned.
- Probably they need to be hierarchical.
- We need (to use) more data.
- Computer vision is in the state of machine learning 10 years ago (at least for object classification).
- Benchmark datasets start to become available, e.g. PASCAL VOC.

APPLICATION:

Comparable accuracy, but **15 times faster** than state-of-the-art of face detectors (at that time)

The Viola-Jones detector has been recognized as one of the most exciting breakthrough in computer vision (in particular, face detection) during the past ten years.

It is the most popularly used face-detector by far

"Boosting" has become a buzzword in computer vision

Exam Questions:

1. Why does Boosting often not overfit?
2. What is Boosting? (main goal/method)
3. Will Adaboost always maximize the margin?

Exam Question I: Why boosting does not overfit?

- Many researchers have studied this and several theoretical explanations have been given, but no one has convinced others
- The **margin theory** of Boosting (see pp.9) is particularly interesting
  - If it succeeds, a strong connection of Boosting and SVM can be found.
  - But
    - L. Breiman [NCJ99] indicated that larger margin does not necessarily mean better generalization

This almost sentenced the margin theory of Boosting to death

Exam Question II:

- What is the main goal/ method of boosting?
  - convert a weak learning algorithm that performs just slightly better than a guess into one with high accuracy, a strong one.

Exam Question III:

- Will adaboost always maximize the margin?

AdaBoost may converge to a margin that is significantly below maximum. (R, Daubechies, Schapire 04)

Thanks!

Applause goes to R. Schapire& Y. Freund