# An overview of Boosting

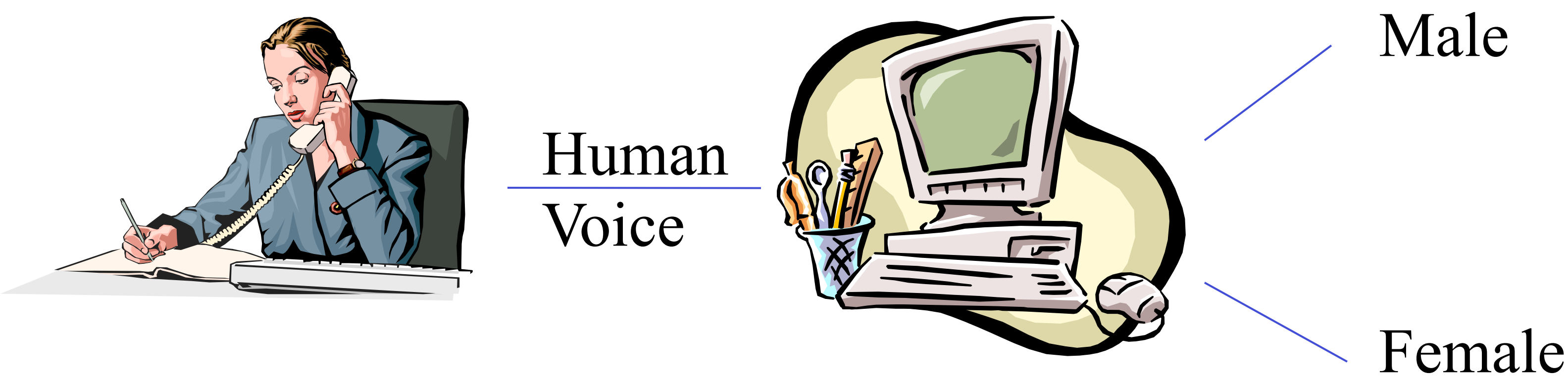
Yoav Freund
UCSD

# Plan of talk

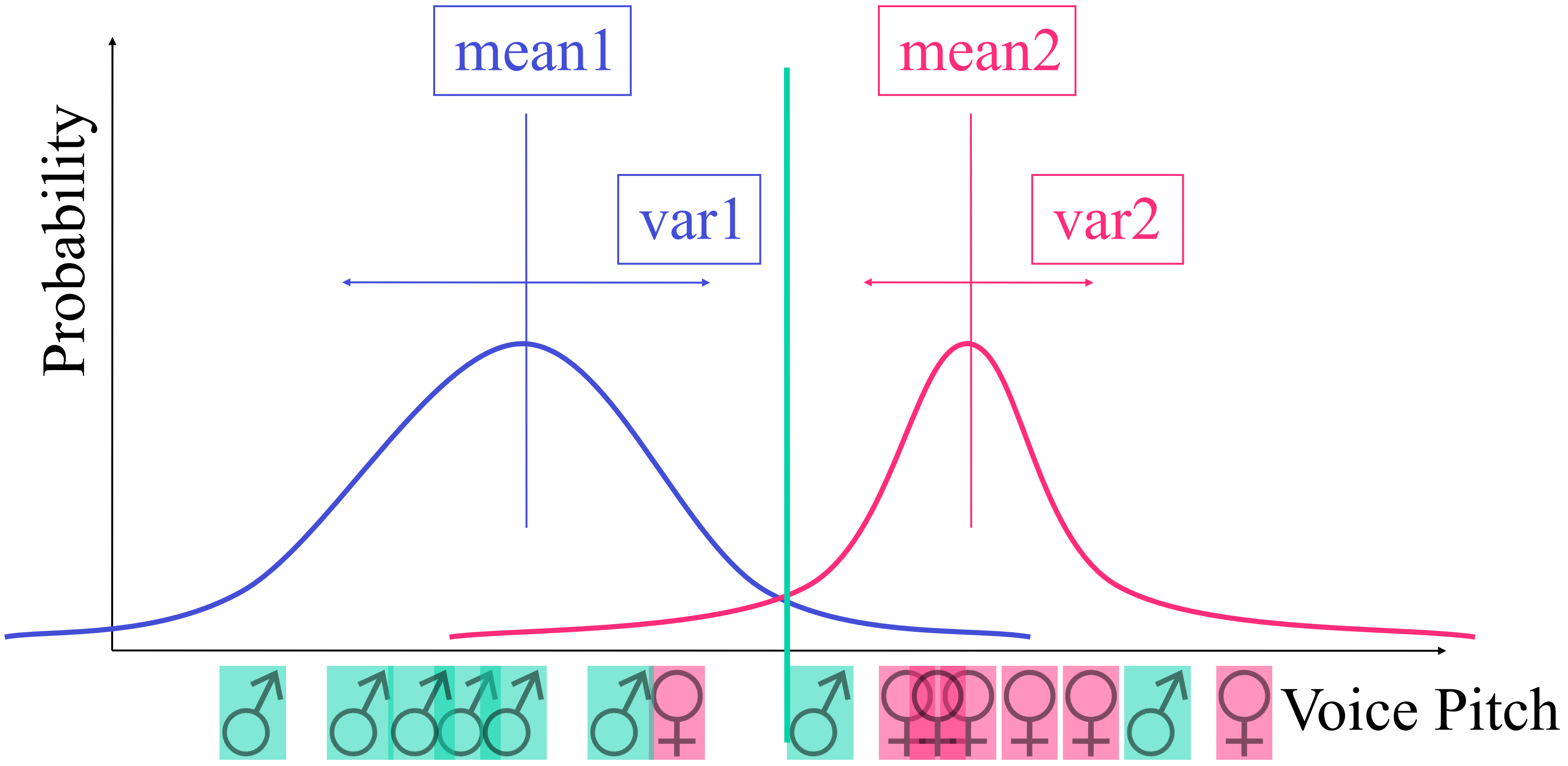
- Generative vs. non-generative modeling
- Boosting
- Alternating decision trees
- Boosting and over-fitting
- Applications

# Toy Example

- Computer receives telephone call
- Measures Pitch of voice
- Decides gender of caller

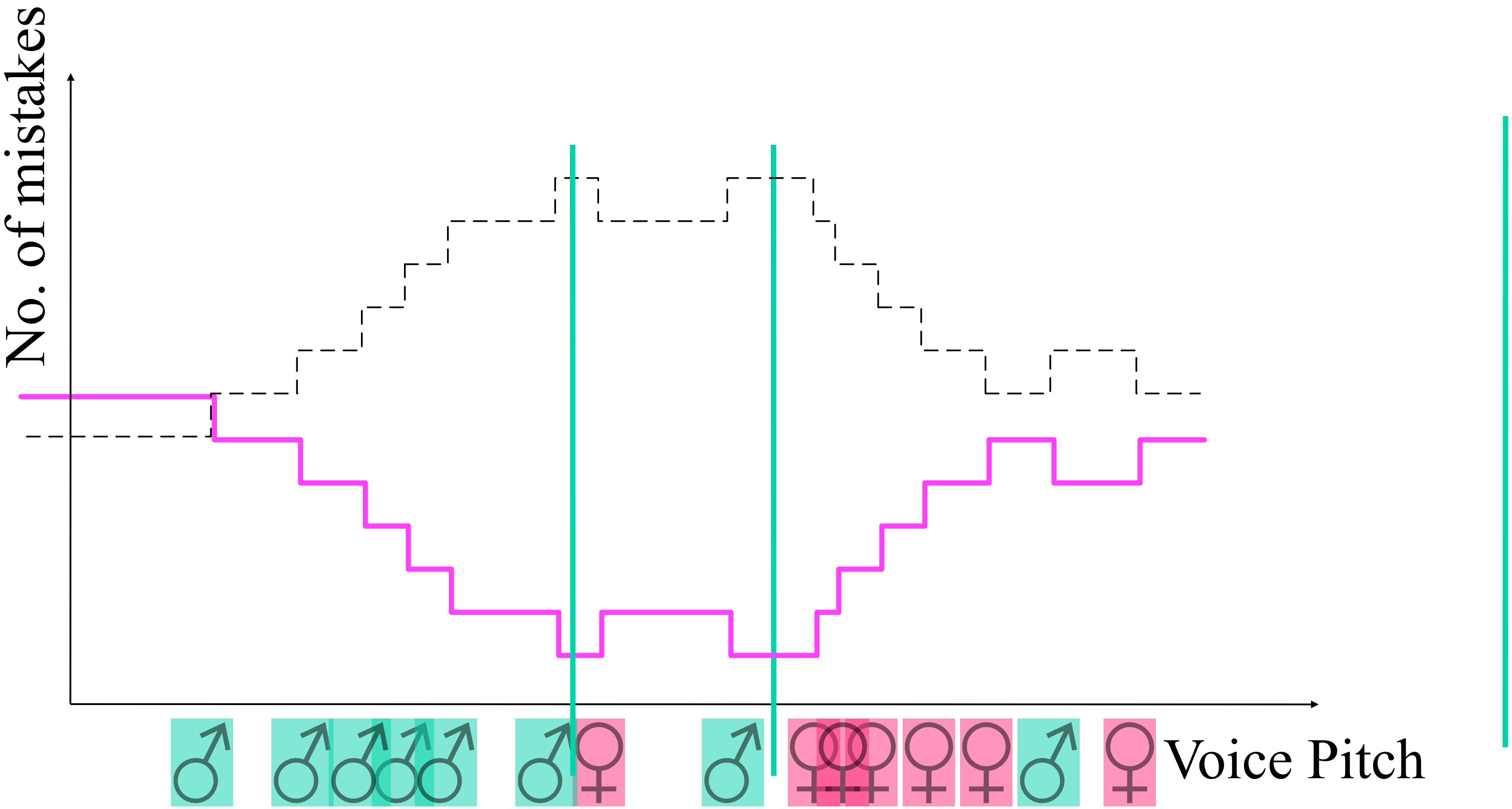


# Generative modeling

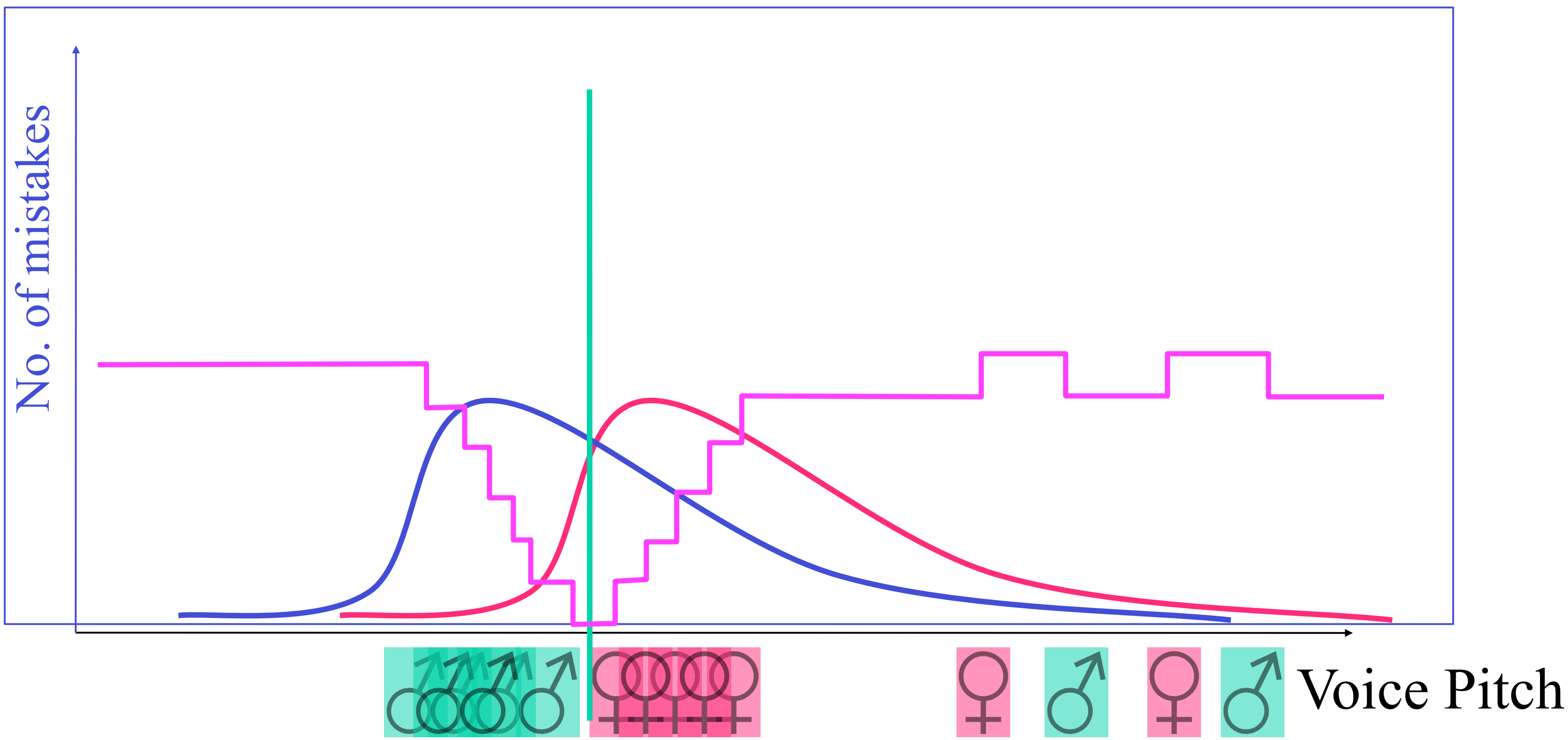


# Discriminative approach

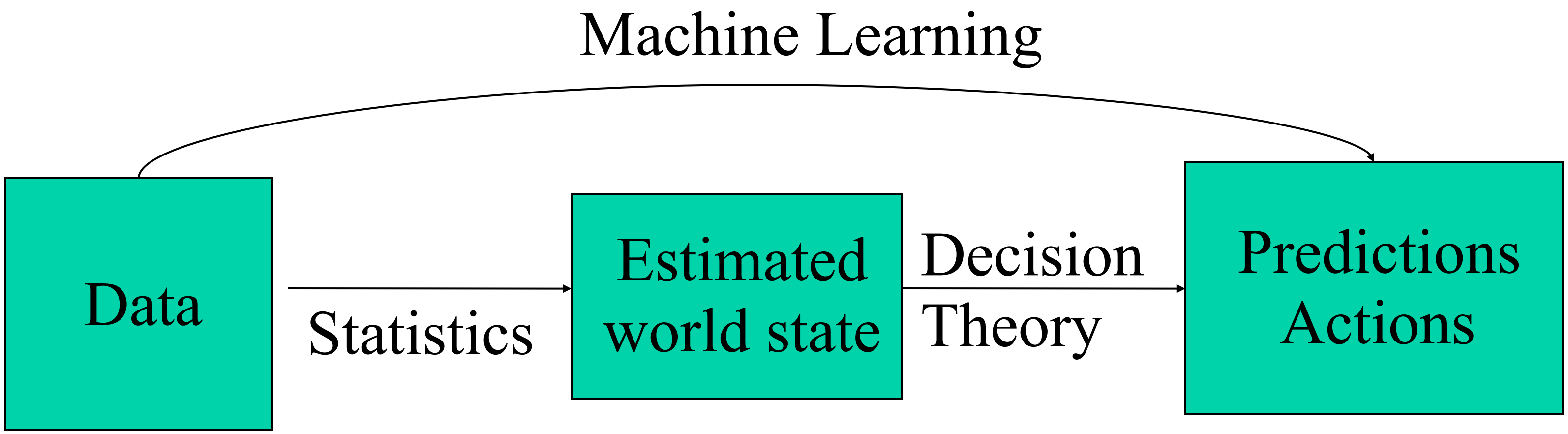
[Vapnik 85]



# Ill-behaved data



# Traditional Statistics vs. Machine Learning



# Comparison of methodologies

| | | |
|---|---|---|
| Model | Generative | Discriminative |
| Goal | Probability estimates | Classification rule |
| Performance measure | Likelihood | Misclassification rate |
| Mismatch problems | Outliers | Misclassifications |

# Boosting

# A weighted training set

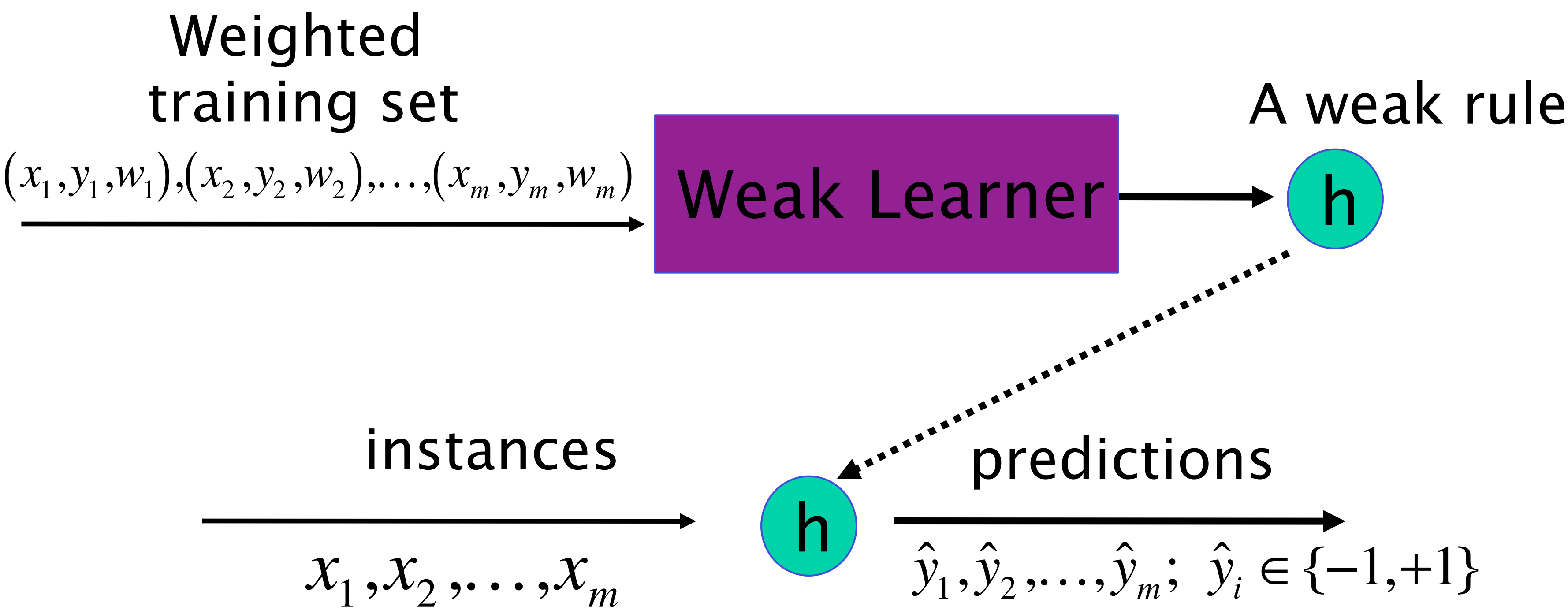
Feature vectors

Binary labels {-1,+1}

Positive weights

$$\left(x_1, y_1, w_1\right), \left(x_2, y_2, w_2\right), \ldots, \left(x_m, y_m, w_m\right)$$

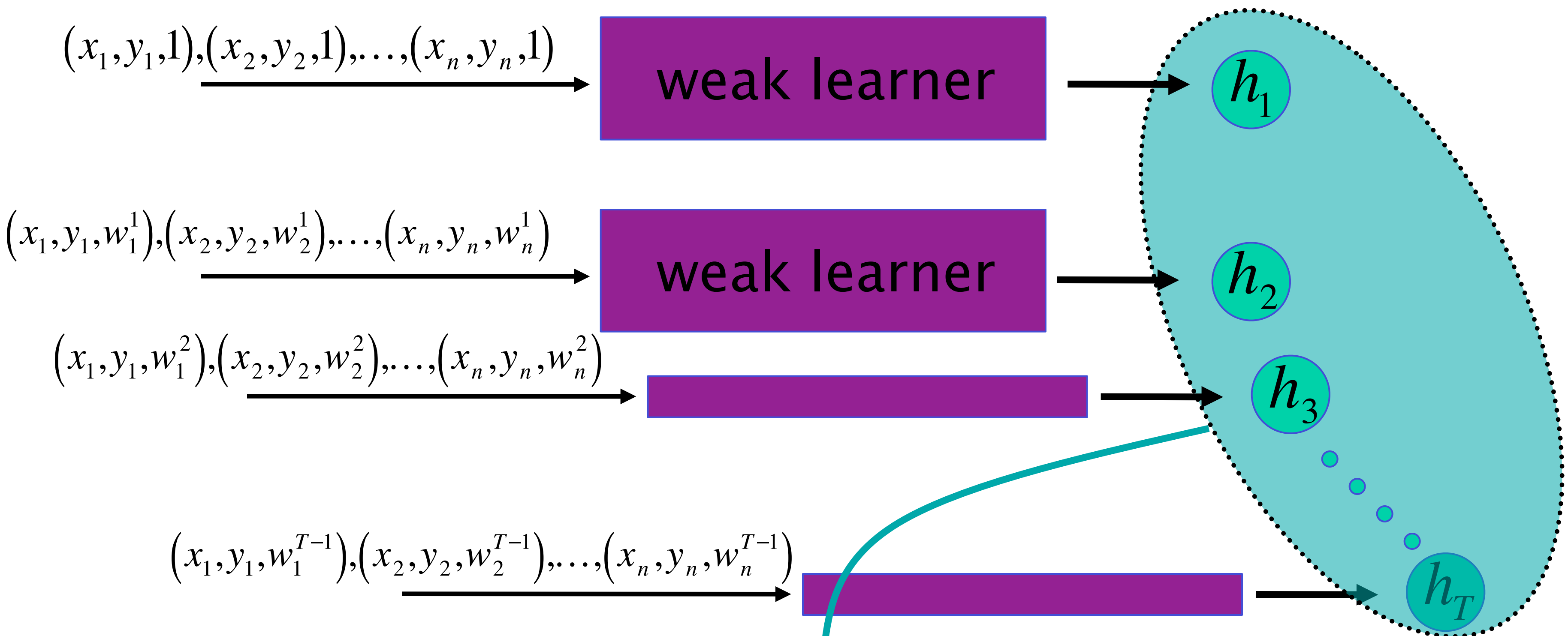
# A weak learner




The weak requirement:

$$\left| \frac{\sum_{i=1}^{m} y_i \hat{y}_i w_i}{\sum_{i=1}^{m} w_i} \right| > \gamma > 0$$

# The boosting process




$$F_T(x) = \alpha_1 h_1(x) + \alpha_2 h_2(x) + \ldots + \alpha_T h_T(x)$$

$$\text{Prediction}(x) = sign(F_T(x))$$

# A Formal Description of Boosting

[Slides from Rob Schapire]

- 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 $\epsilon_t$ on $D_t$:

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

# AdaBoost

[Slides from Rob Schapire]

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

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

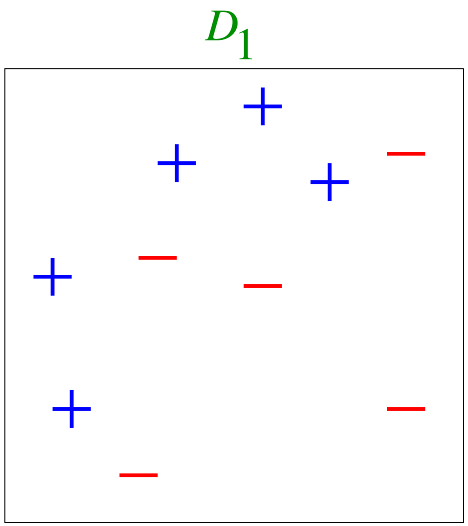
where $Z_t$ = normalization factor

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

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

# Toy Example

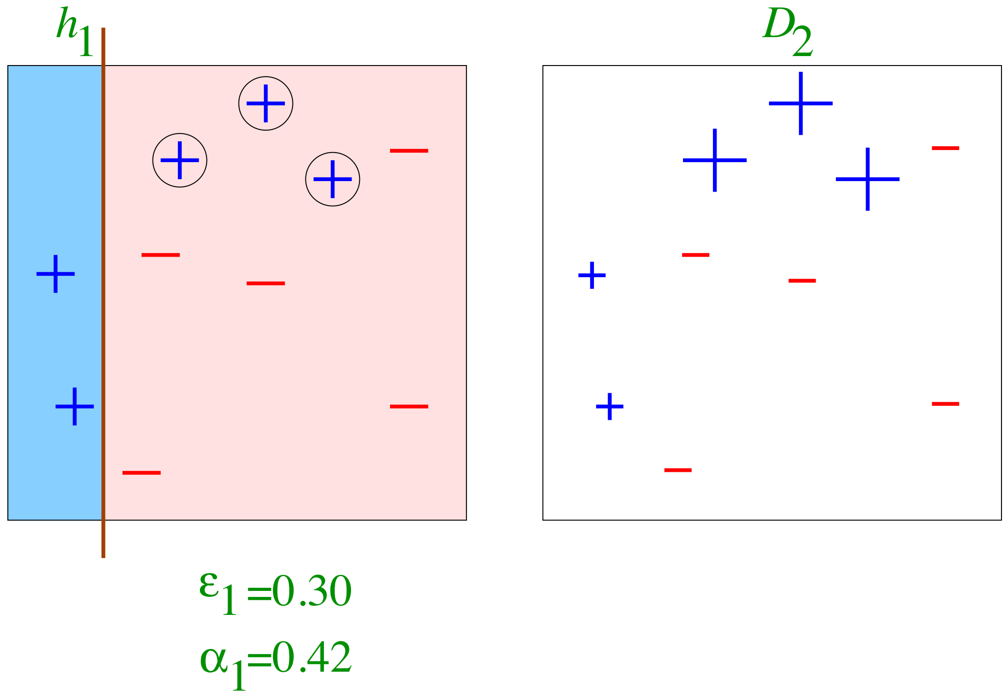
[Slides from Rob Schapire]




weak classifiers = vertical or horizontal half-planes

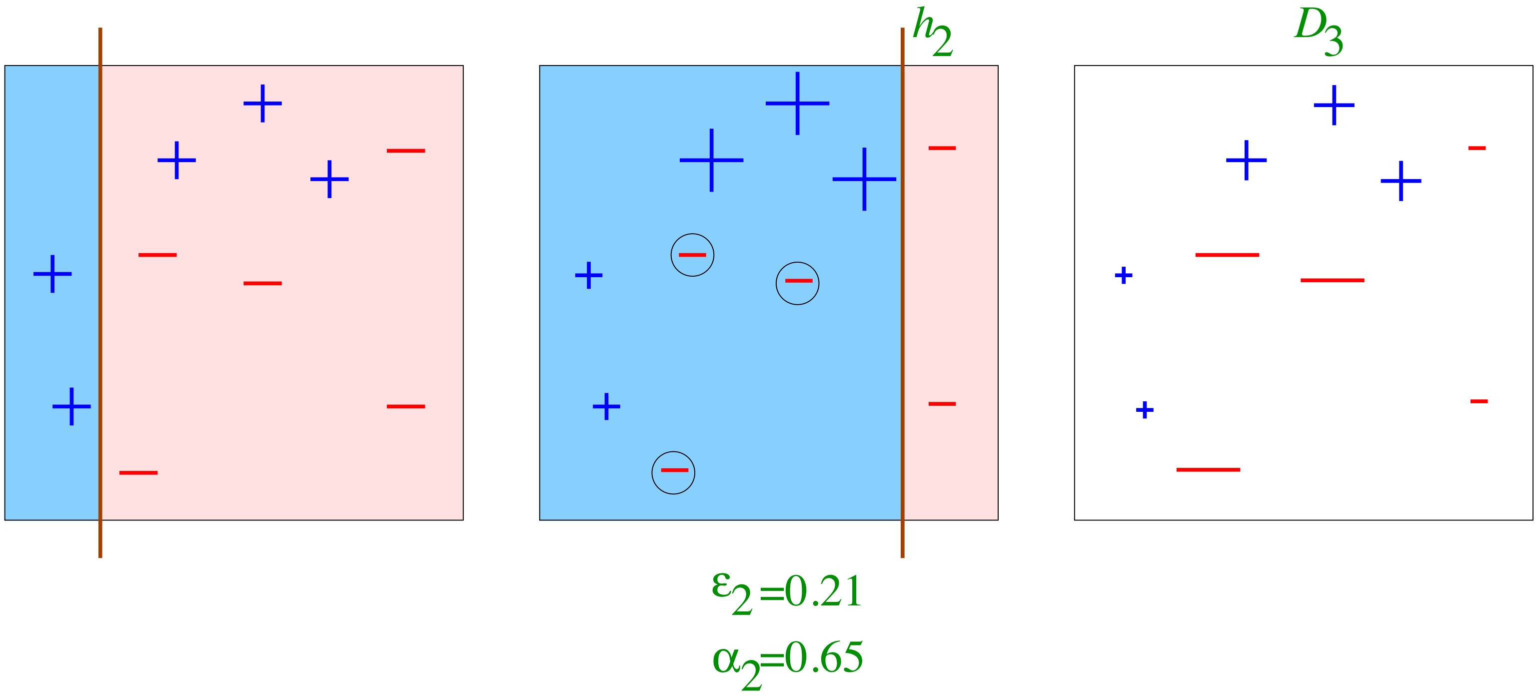
# Round 1

[Slides from Rob Schapire]



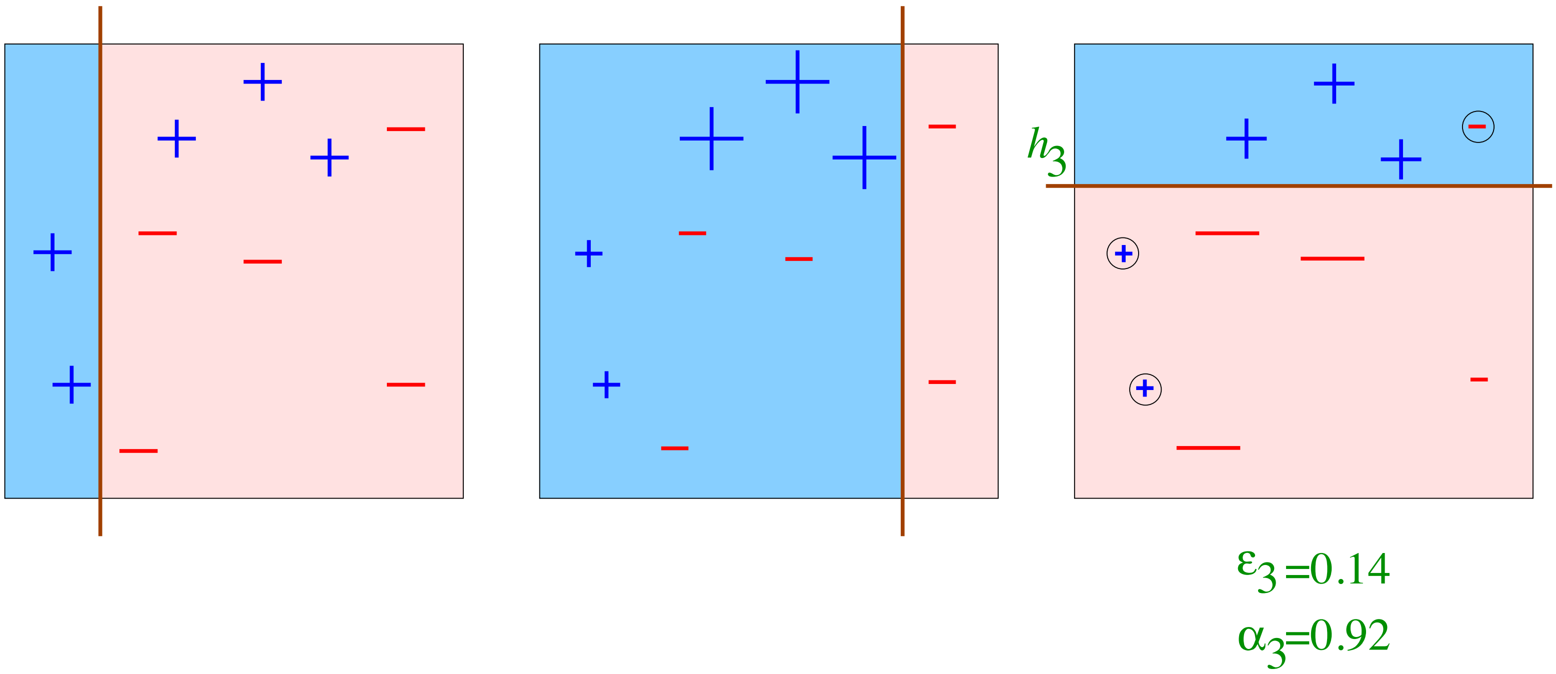
# Round 2

[Slides from Rob Schapire]



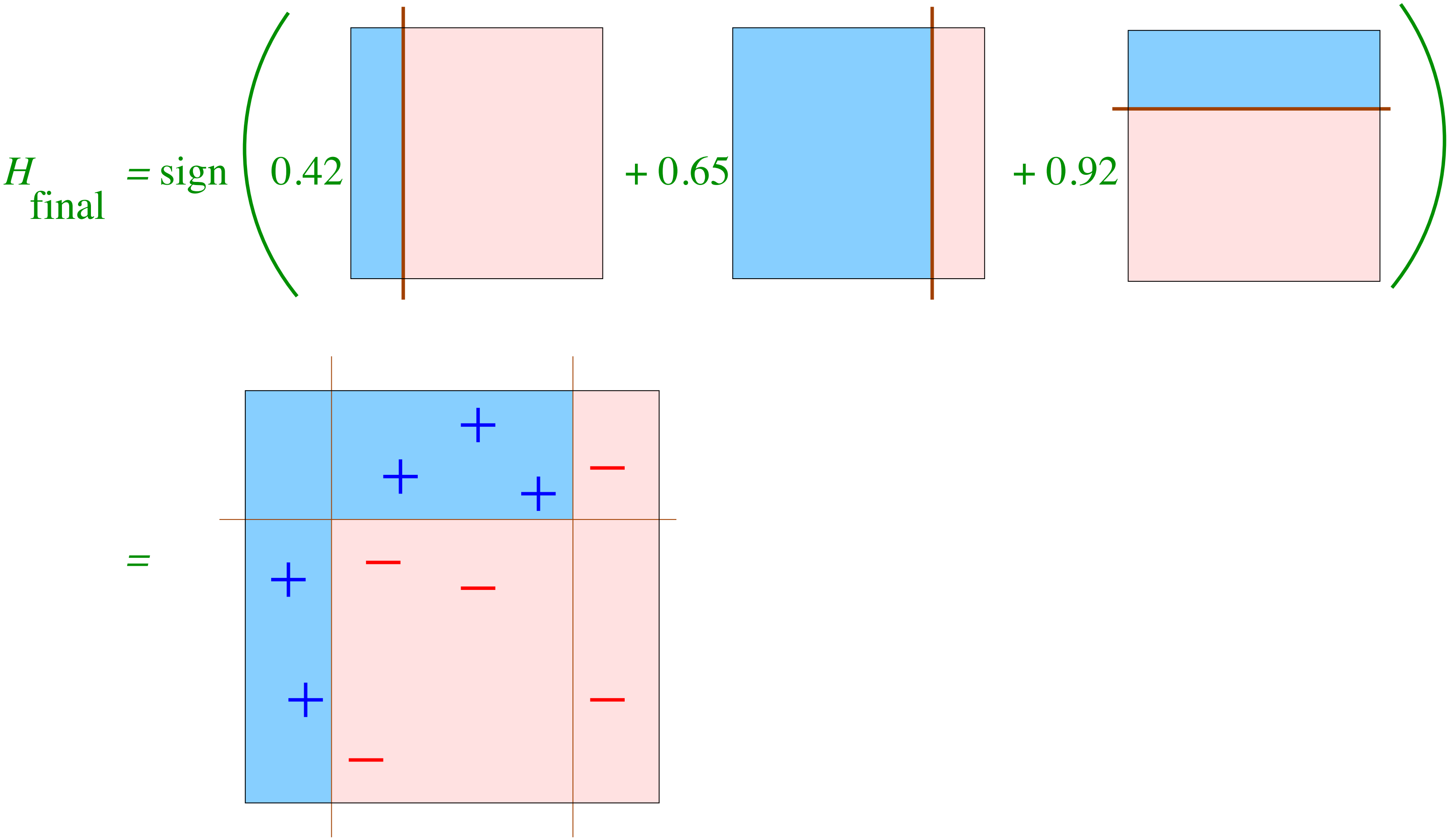
# Round 3

[Slides from Rob Schapire]



# Final Classifier

[Slides from Rob Schapire]



# Analyzing the Training Error

[Slides from Rob Schapire]

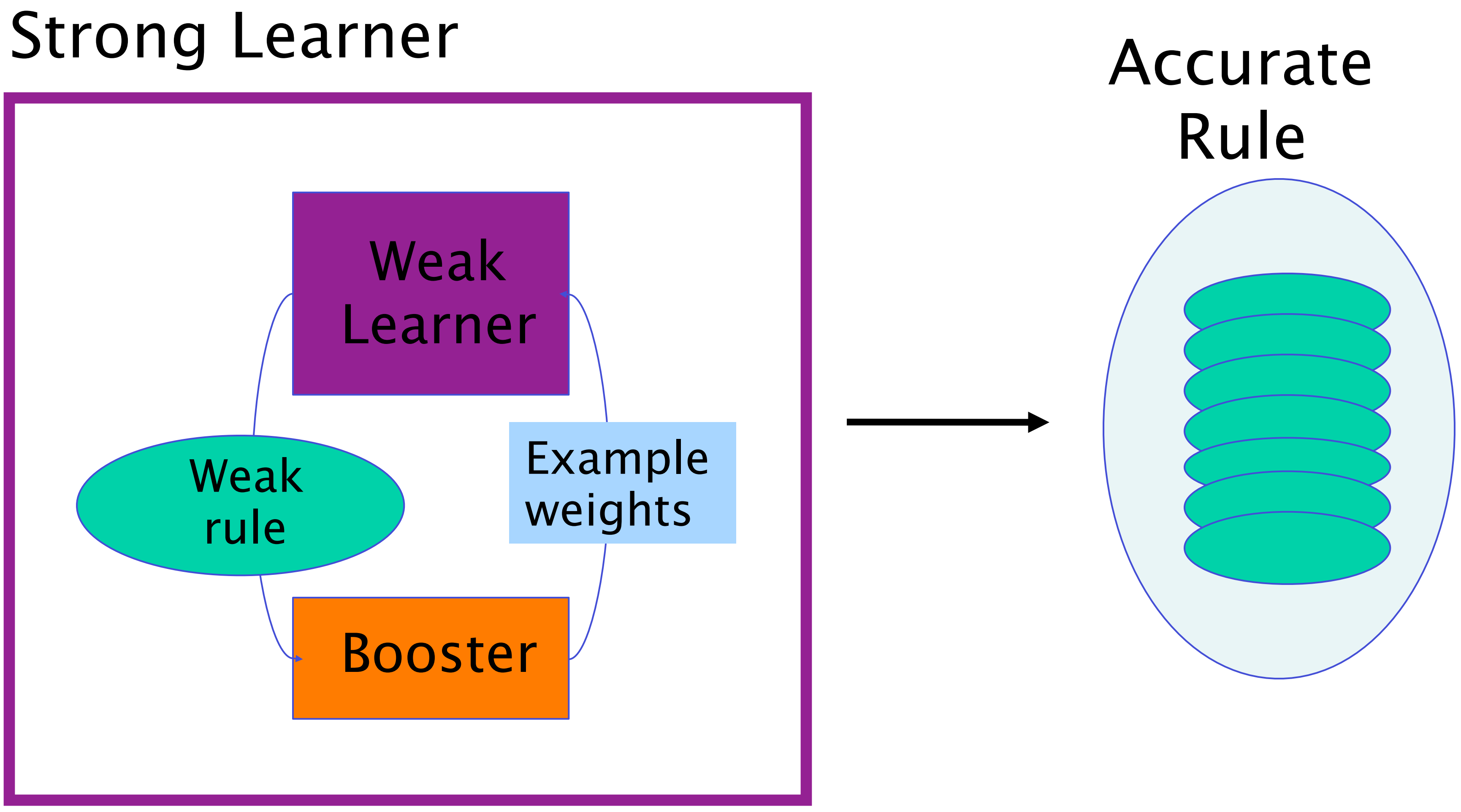
[with Freund]

- Theorem:
  - write $\epsilon_t$ as $\frac{1}{2} - \gamma_t$ [ $\gamma_t =$ "edge" ]
  - then

$$\begin{aligned}\text{training error}(H_{\text{final}}) &\le \prod_t \left[2\sqrt{\epsilon_t(1-\epsilon_t)}\right] \\ &= \prod_t \sqrt{1-4\gamma_t^2} \\ &\le \exp\left(-2\sum_t \gamma_t^2\right)\end{aligned}$$

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

# Boosting block diagram



# Boosting with specialists

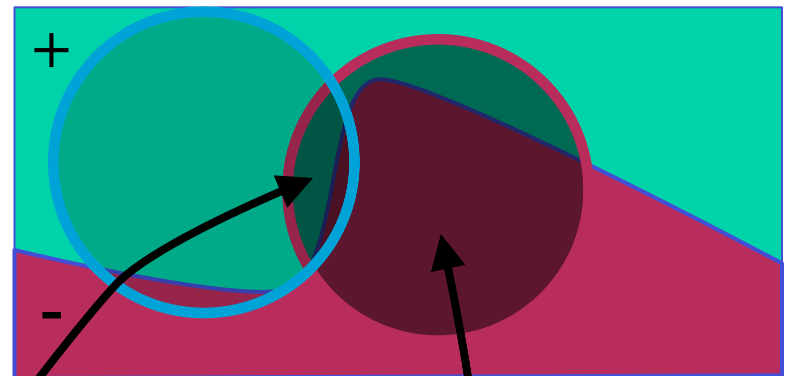
- Specialists predict only when they are confident.
- In addition to {-1,+1} specialists use 0 to indicate no-prediction.
- As boosting allows both positive and negative weights: we restrict attention to specialists that output {0,1}.

# Adaboost as variational optimization

Weak Rule: $h_t : X \to \{0,1\}$

Label: $y \in \{-1,+1\}$

Training set: $\{(x_1, y_1, 1), (x_2, y_2, 1), \ldots, (x_n, y_n$




$$F_0(x) \equiv 0$$

$$\text{for } t = 1..T$$

$$w_i^t = \exp\left(-y_i F_{t-1}(x_i)\right)$$

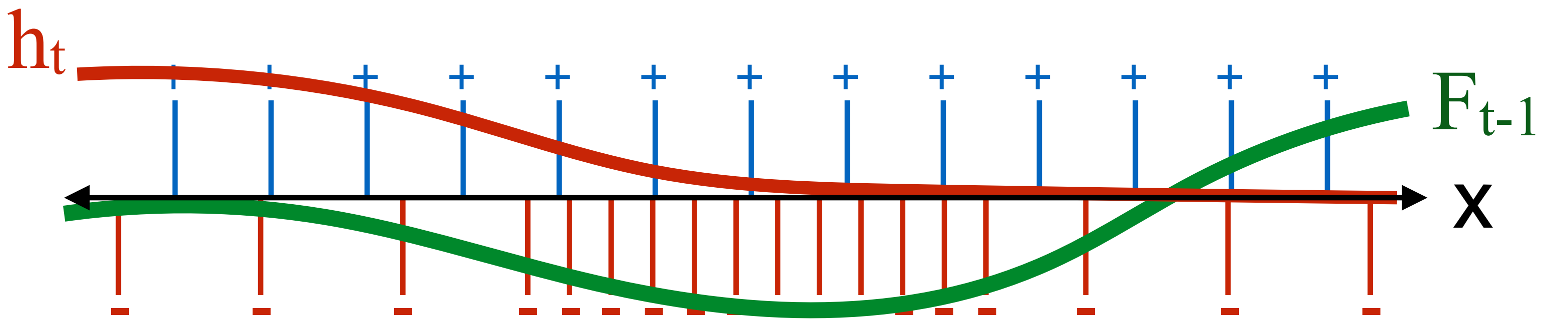
$$\text{Get } h_t \text{ from } \textit{weak} - \textit{learner}$$

$$\alpha_t = \frac{1}{2} \ln\left( \epsilon + \sum_{i:h_t(x_i)=1, y_i=1} w_i^t \right) \Bigg/ \left( \epsilon + \sum_{i:h_t(x_i)=1, y_i=-1} w_i^t \right)$$

$$F_t = F_{t-1} + \alpha_t h_t$$

# AdaBoost as variational optimization.

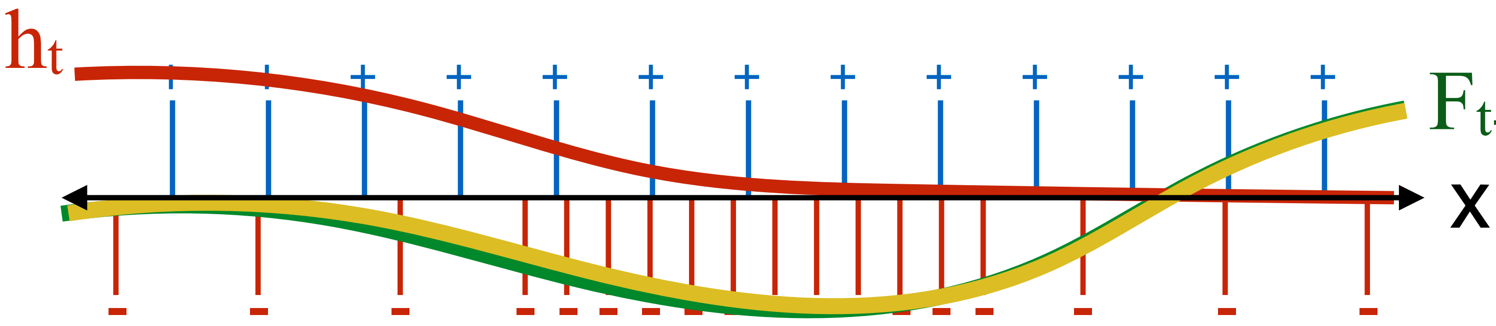
- $x$ = input, a scalar,
- $y$ = output, -1 or +1
- $(x_1,y_1),(x_2,y_2), \ldots ,(x_n,y_n)$ = training set
- $F_{t-1}$ = Strong rule after t-1 boosting iterations.
- $h_t$ = Weak rule produced at iteration t



# AdaBoost as variational optimization.

- $F_{t-1}$ = Strong rule after t-1 boosting iterations.
- $h_t$ = Weak rule produced at iteration t

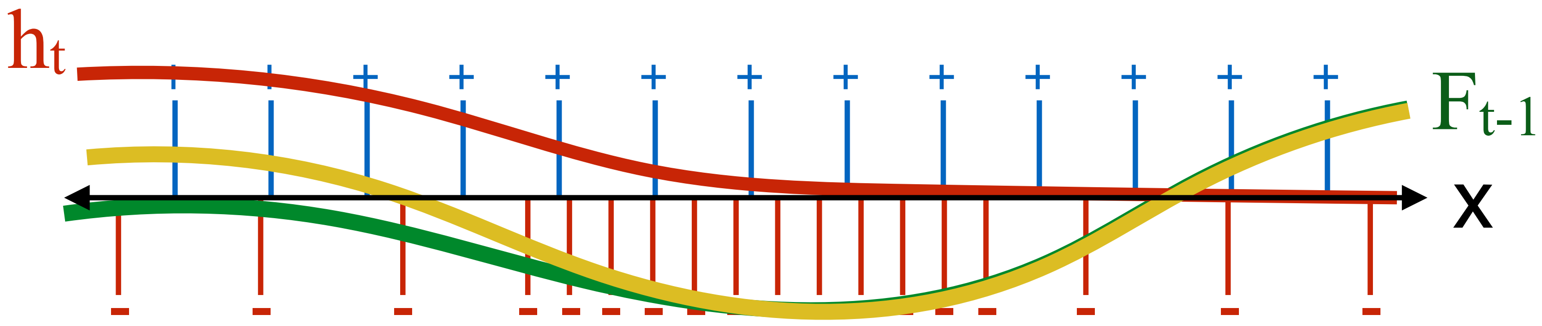
$$F_{t-1}(x) + 0h_t(x)$$



# AdaBoost as variational optimization.

- $F_{t-1}$ = Strong rule after t-1 boosting iterations.
- $h_t$ = Weak rule produced at iteration t

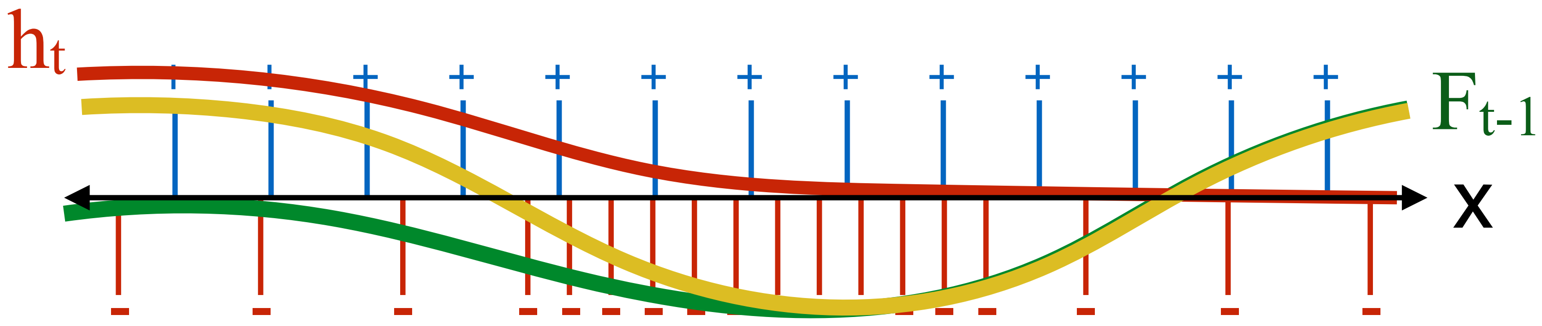
$$F_{t-1}(x)+0.4h_t(x)$$



# AdaBoost as variational optimization.

- $F_{t-1}$ = Strong rule after t-1 boosting iterations.
- $h_t$ = Weak rule produced at iteration t

$$F_t(x) = F_{t-1}(x) + 0.8 h_t(x)$$



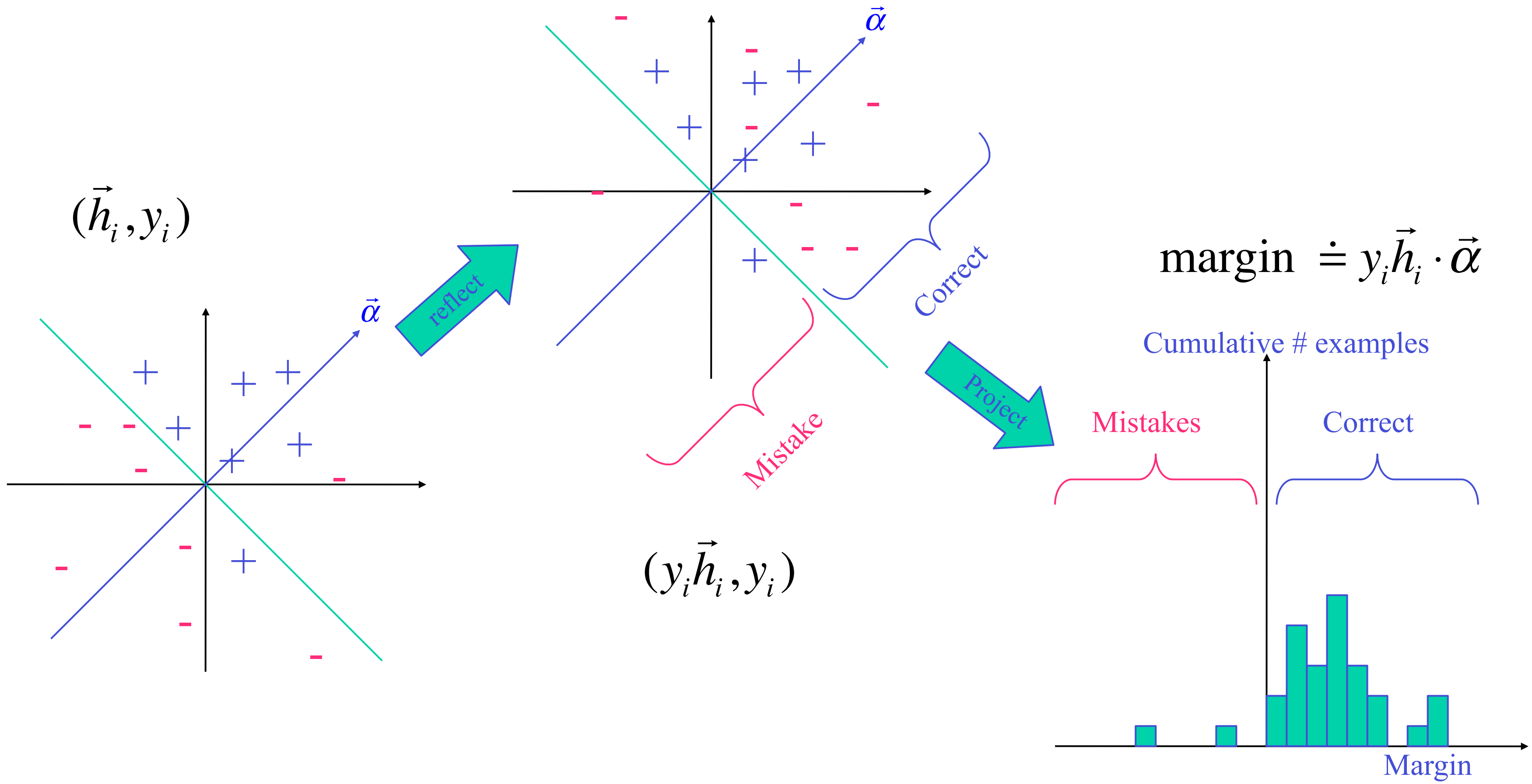
# Margins

Fix a set of weak rules: $\vec{h}(x) = (h_1(x), h_2(x), \ldots, h_T(x))$

Represent the i'th example $x_i$ with outputs of the weak rules: $\vec{h}_i$

Labels: $y \in \{-1, +1\}$ Training set: $(\vec{h}_1, y_1), (\vec{h}_2, y_2), \ldots, (\vec{h}_n, y_n)$

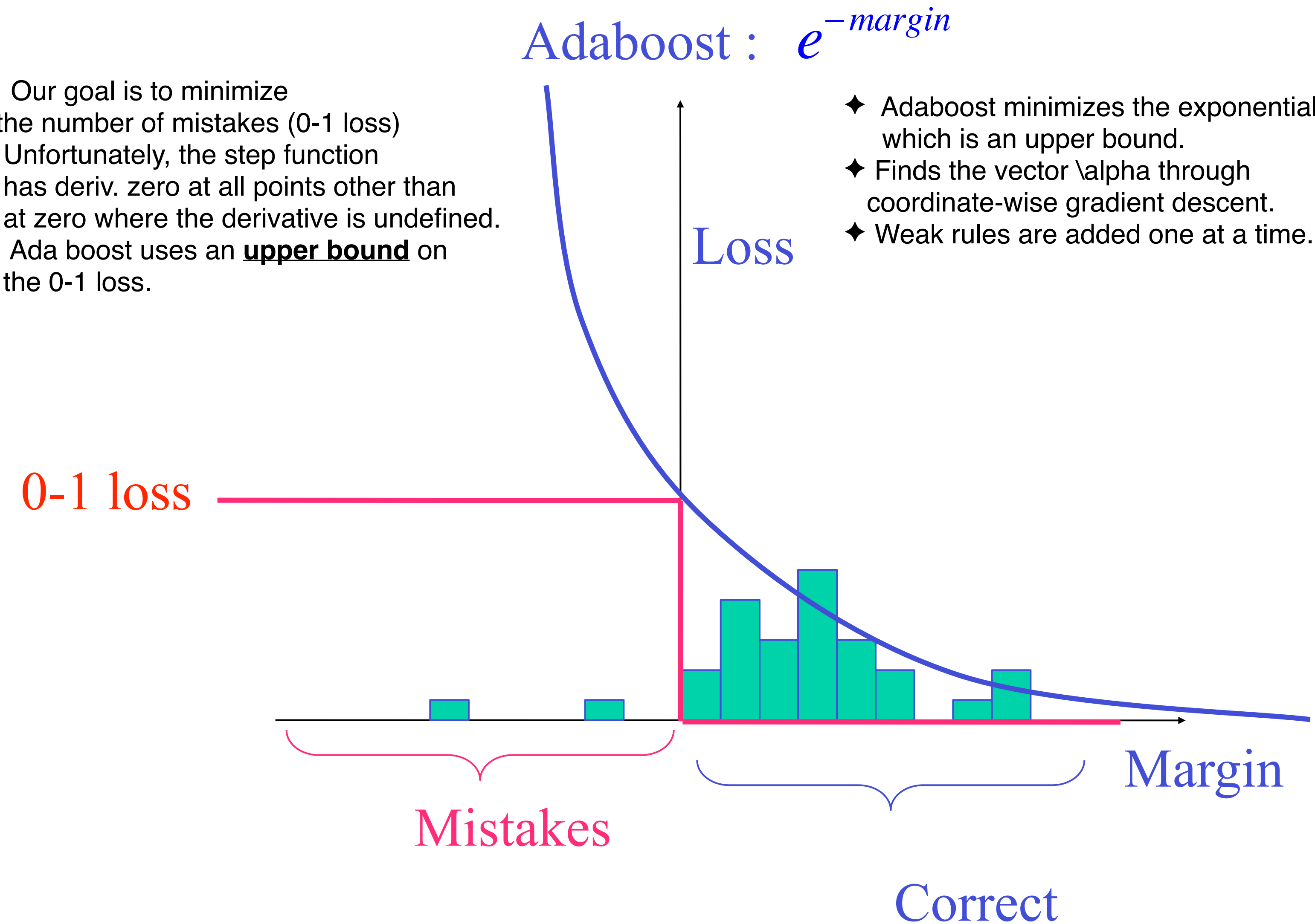
*Goal*: Find a weight vector $\vec{\alpha} = (\alpha_1, \ldots, \alpha_T)$ that minimizes number of training mistakes



# Boosting as gradient descent

- Our goal is to minimize the number of mistakes (0-1 loss)
- Unfortunately, the step function has deriv. zero at all points other than at zero where the derivative is undefined.
- Ada boost uses an **upper bound** on the 0-1 loss.

- Adaboost minimizes the exponential loss which is an upper bound.
- Finds the vector \alpha through coordinate-wise gradient descent.
- Weak rules are added one at a time.



# Adaboost as gradient descent

- ✦ Weak rules are added one at a time.
- ✦ Fixing the alphas for 1..t-1, find alpha for the new rule ($h_t$)
- ✦ Weak rule defines how each example would move = increase or decrease the margin.

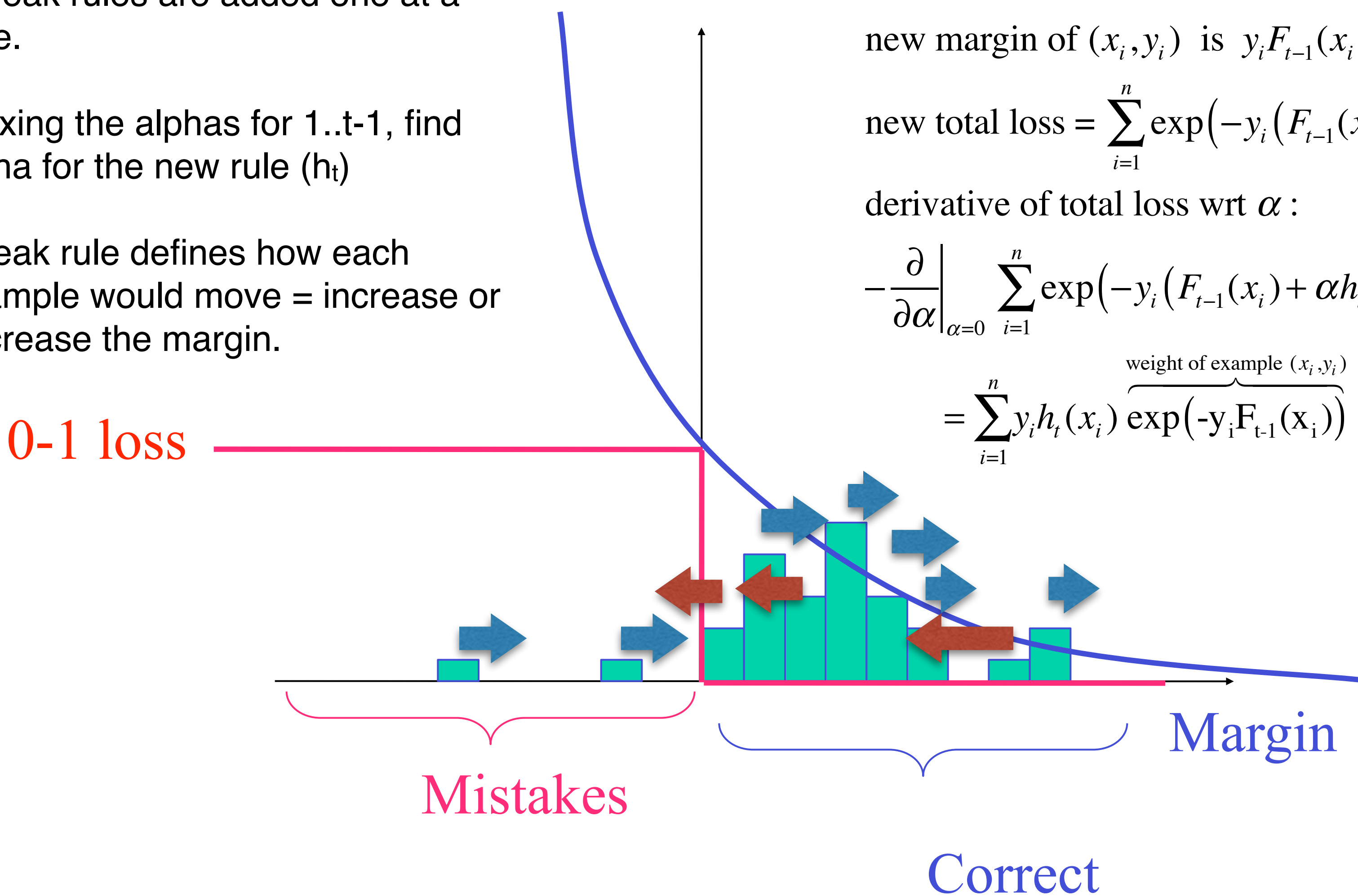
new margin of $(x_i, y_i)$ is $y_i F_{t-1}(x_i) + \alpha y_i h_t(x_i)$

new total loss = $\sum_{i=1}^{n} \exp\left(-y_i \left(F_{t-1}(x_i) + \alpha h_t(x_i)\right)\right)$

derivative of total loss wrt $\alpha$ :

$$-\frac{\partial}{\partial \alpha}\bigg|_{\alpha=0} \sum_{i=1}^{n} \exp\left(-y_i \left(F_{t-1}(x_i) + \alpha h_t(x_i)\right)\right)$$

$$= \sum_{i=1}^{n} y_i h_t(x_i) \overbrace{\exp\left(-y_i F_{t-1}(x_i)\right)}^{\text{weight of example } (x_i, y_i)}$$



# Logitboost as gradient descent

Also called Gentle-Boost and Logit Boost, Hastie, Freedman & Tibshirani

margin=$m$

Instead of the loss $e^{-m}$

define loss to be $\log\left(1+e^{-m}\right)$

When margin $\gg 0$ : $\log\left(1+e^{-m}\right) \approx e^{-m}$

When margin $\ll 0$ : $\log\left(1+e^{-m}\right) \approx -m$

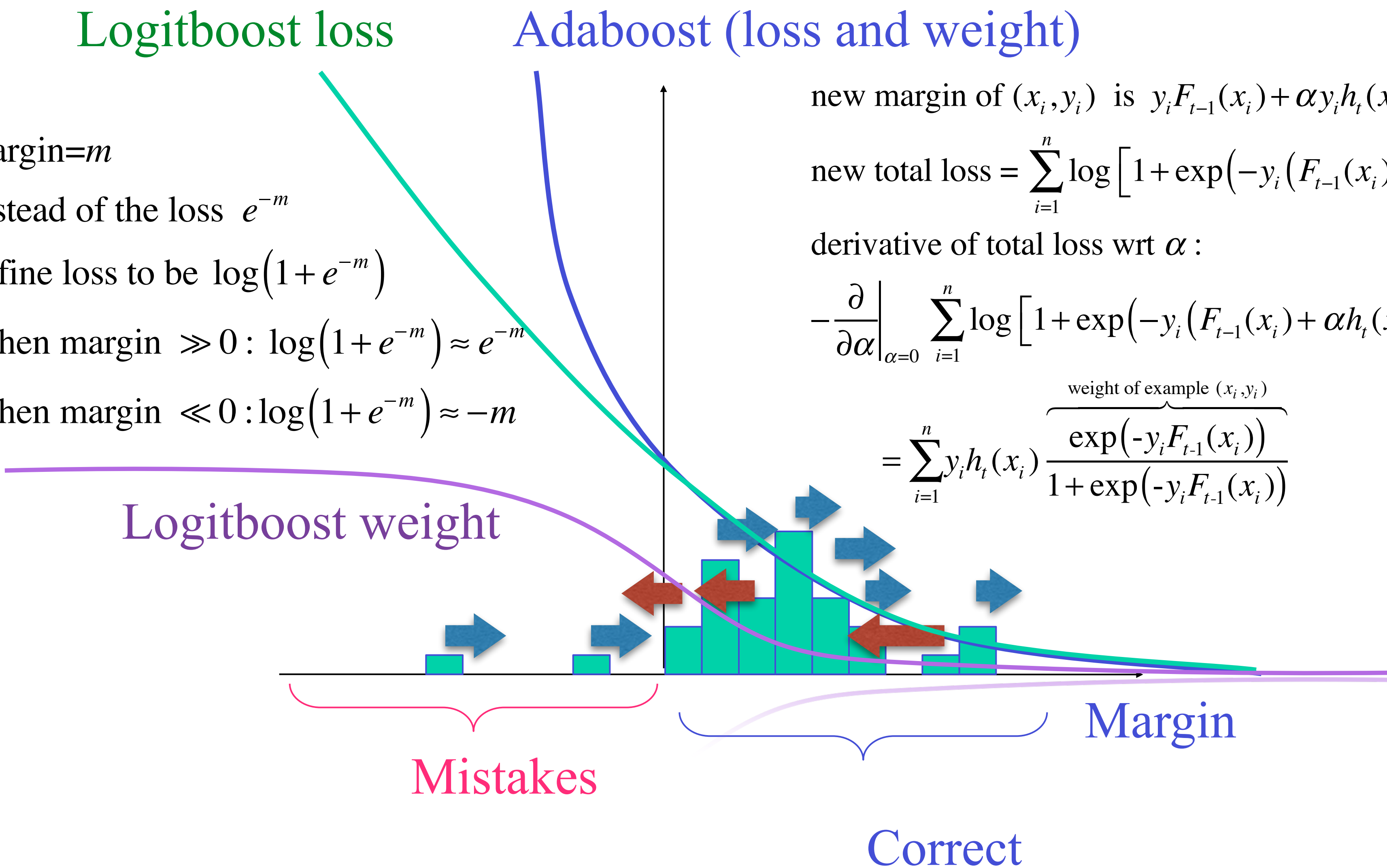
new margin of $(x_i, y_i)$ is $y_i F_{t-1}(x_i) + \alpha y_i h_t(x_i)$

new total loss = $\sum_{i=1}^{n} \log\left[1+\exp\left(-y_i\left(F_{t-1}(x_i)+\alpha h_t(x\right.\right.\right.$

derivative of total loss wrt $\alpha$ :

$$-\frac{\partial}{\partial \alpha}\bigg|_{\alpha=0} \sum_{i=1}^{n} \log\left[1+\exp\left(-y_i\left(F_{t-1}(x_i)+\alpha h_t(x_i)\right)\right)\right]$$

$$= \sum_{i=1}^{n} y_i h_t(x_i) \overbrace{\frac{\exp\left(-y_i F_{t-1}(x_i)\right)}{1+\exp\left(-y_i F_{t-1}(x_i)\right)}}^{\text{weight of example } (x_i, y_i)}$$



# Noise resistance

- Adaboost:
  - perform well when a achievable error rate is close to zero (almost consistent case).
  - Errors = examples with negative margins, get very large weights, can overfit.
- Logitboost:
  - Inferior to adaboost when achievable error rate is close to zero.
  - Often better than Adaboost when achievable error rate is high.
  - Weight on any example never larger than 1.

# Gradient-Boost / AnyBoost

- A general recipe for learning by incremental optimization
- Applies to any (differentiable) loss function.

labeled example is $(x, y)$ (can be anything).

prediction is of the form: $F_t(x) = \sum_{i=1}^{t} \alpha_i h_i(x)$

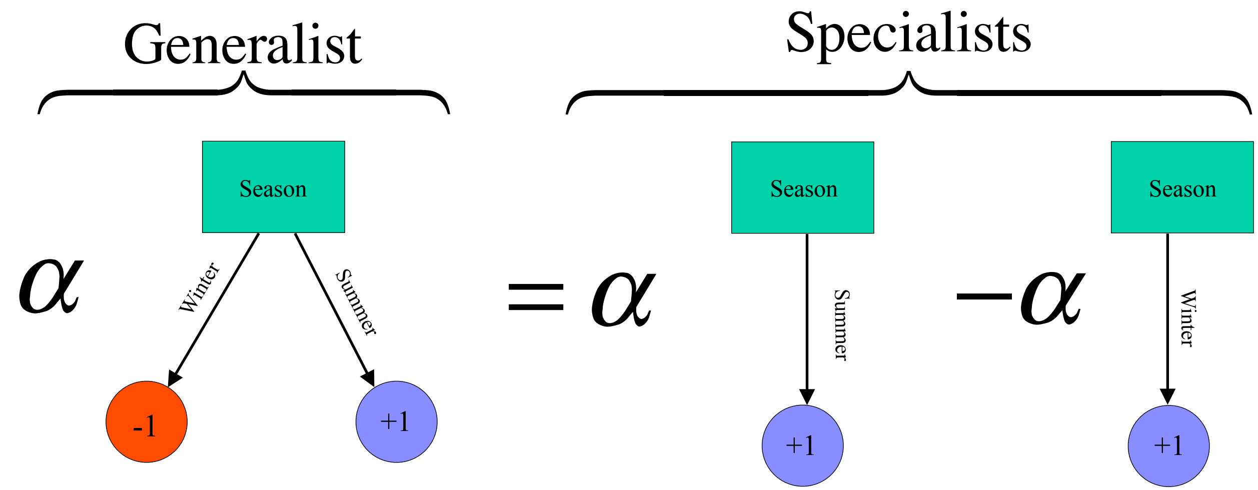
Loss: $L\left(F_t(x), y\right)$ Total Loss so far is: $\sum_{j=1}^{n} L\left(F_{t-1}(x_j), y_j\right)$

Weight of example $(x, y)$: $\frac{\partial}{\partial z} L\left(F_{t-1}(x) + z, y\right)$

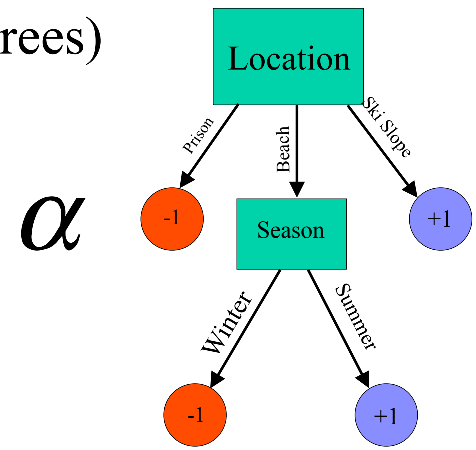
- At each iteration:
  - calc example weights
  - call weak learner with weighted example
  - Add generated weak rule to create new rule: $F_t = F_{t-1} + \alpha_t h_t$

# Styles of weak learners

- Simple (stumps)




- Complex (fully grown trees)

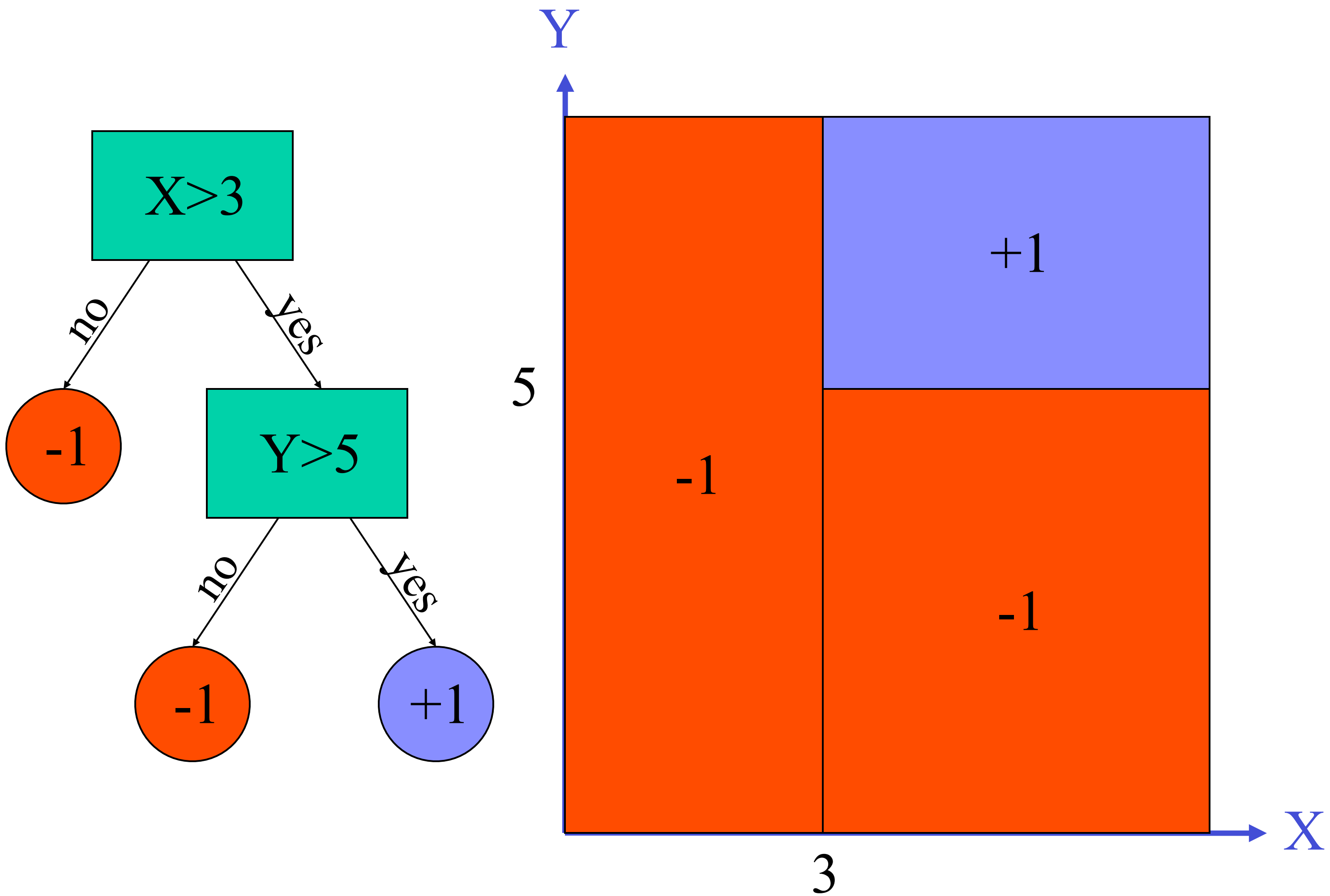



- Everything in between (neural networks, Nearest neighbors, Naive Bayes,…..)

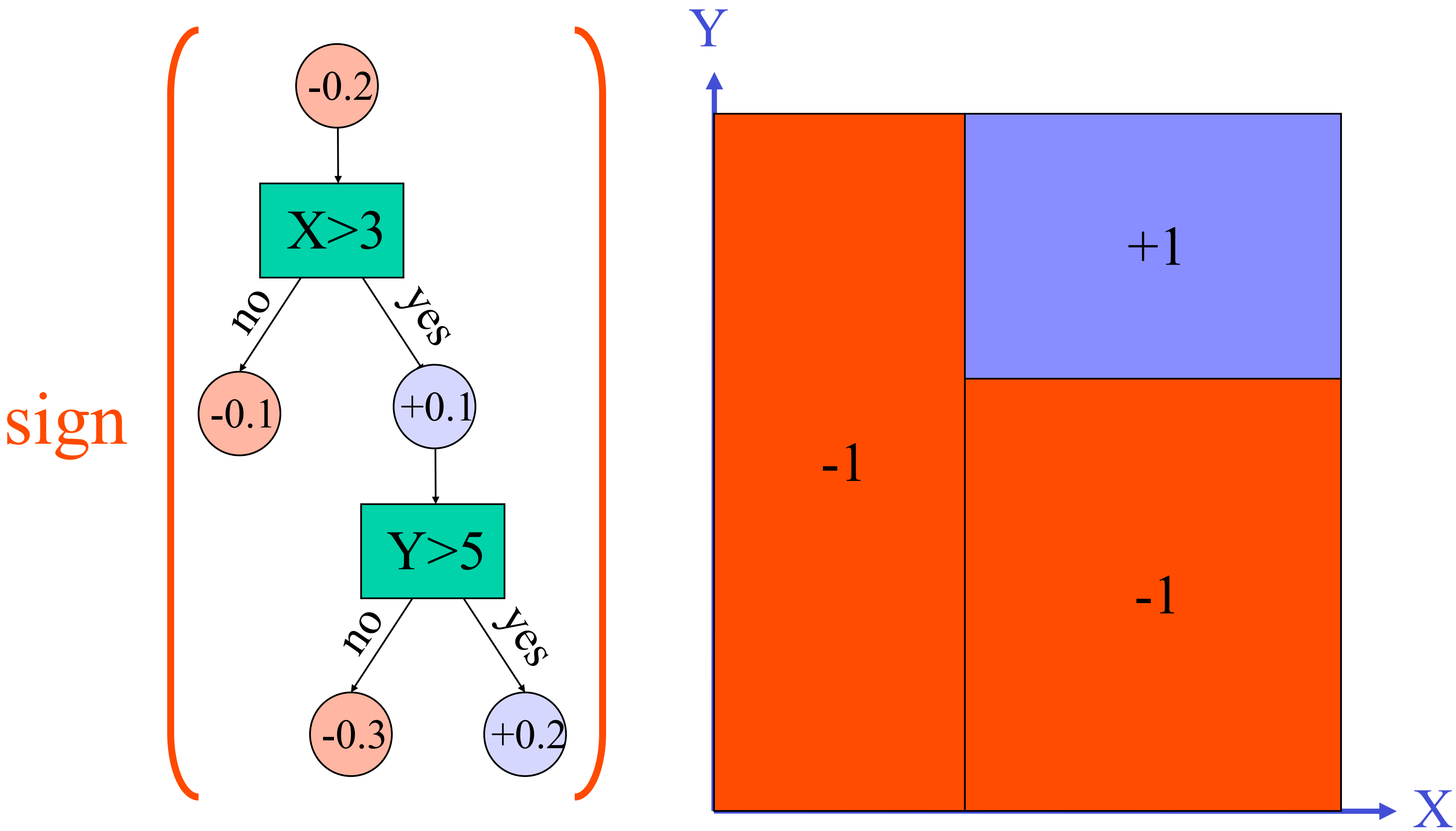
# Alternating Decision Trees

Joint work with Llew Mason

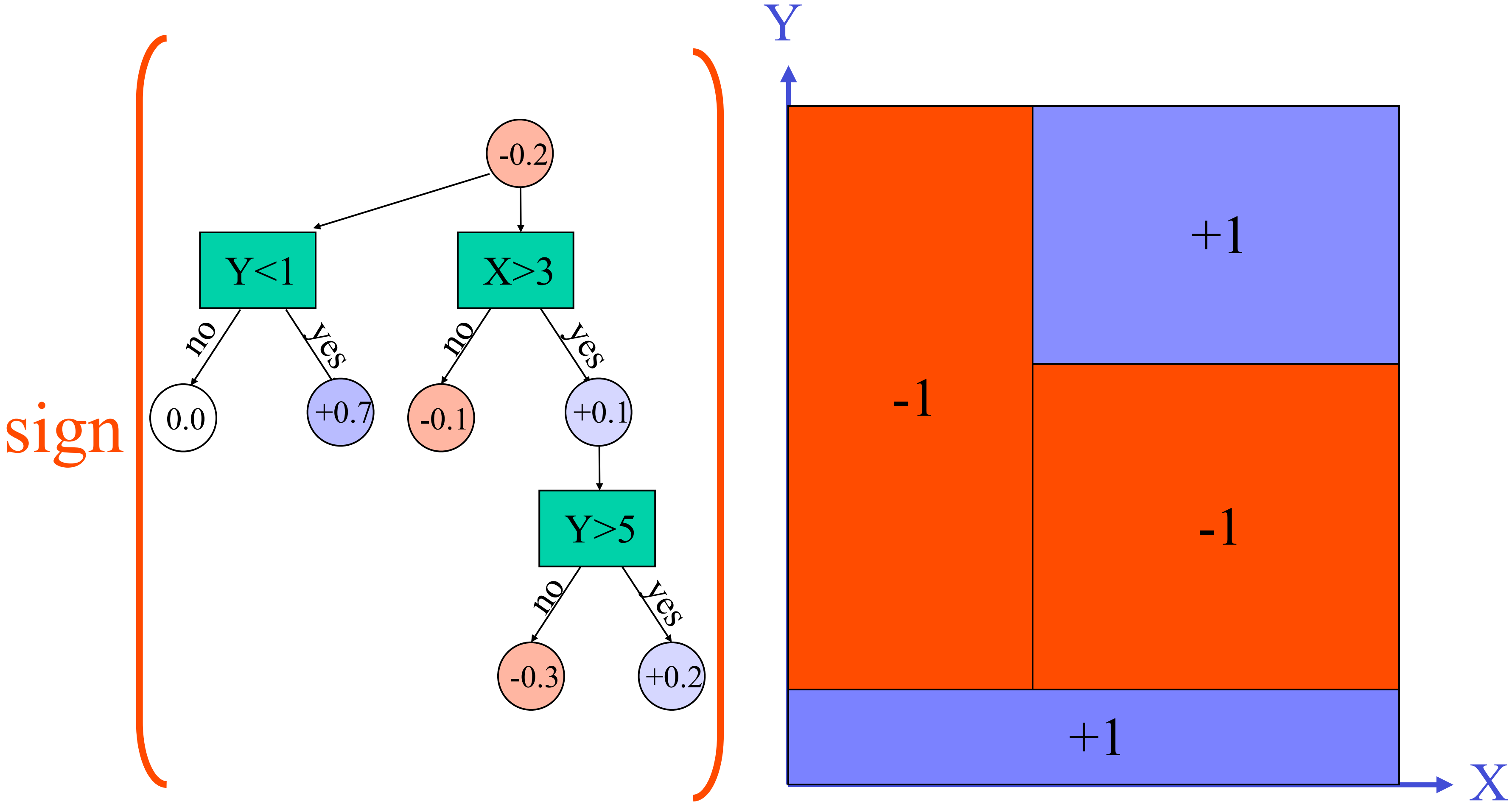
# Decision Trees



# Decision tree as a sum



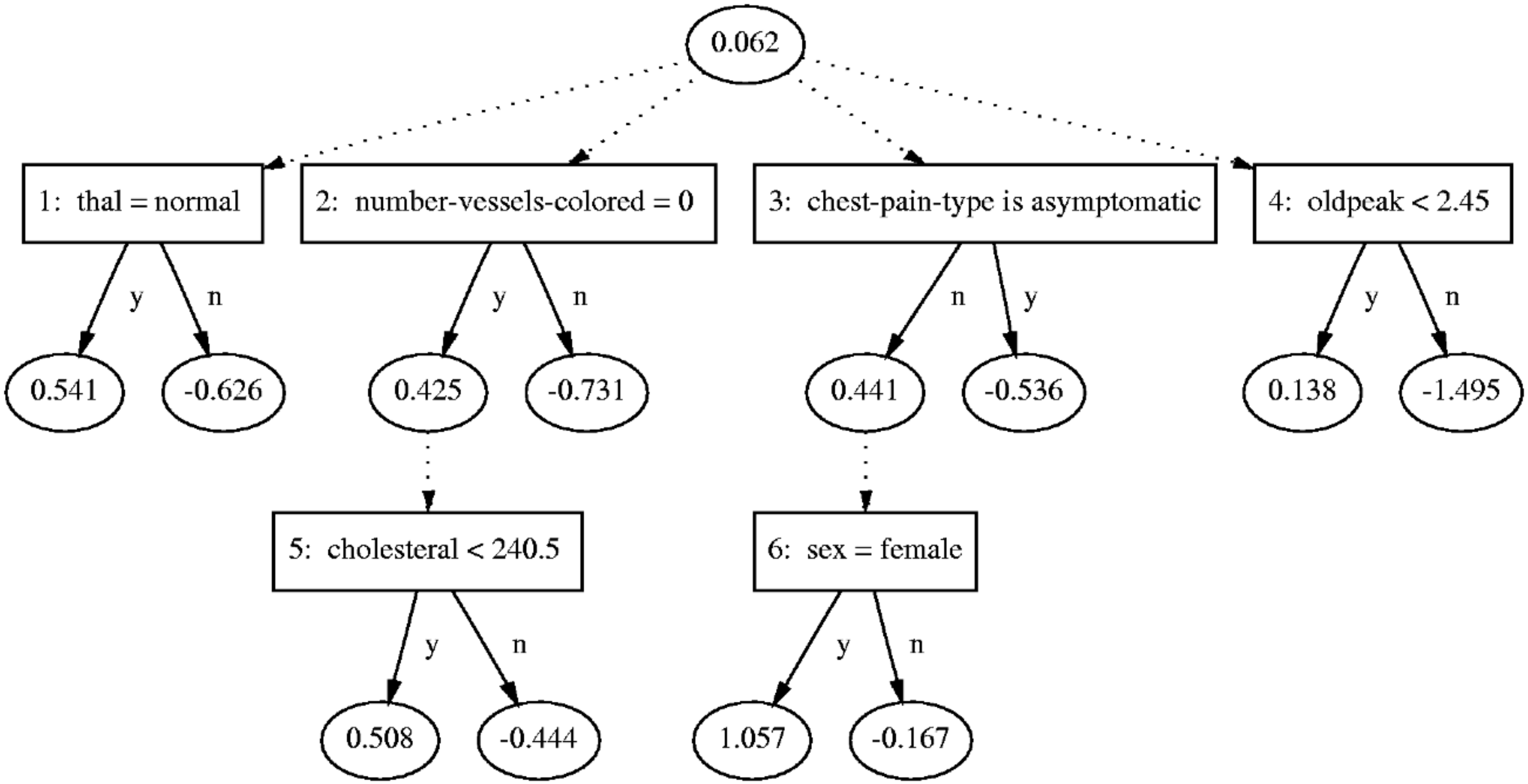
# An alternating decision tree



# Example: Medical Diagnostics

- `Cleve` dataset from UC Irvine database.
- Heart disease diagnostics (+1=healthy,-1=sick)
- 13 features from tests (real valued and discrete).
- 303 instances.

# Adtree for Cleveland heart-disease diagnostics problem



# Cross-validated accuracy

| Learning algorithm | Number of splits | Average test error | Test error variance |
|---|---|---|---|
| ADtree | 6 | 17.0% | 0.6% |
| C5.0 | 27 | 27.2% | 0.5% |
| C5.0 + boosting | 446 | 20.2% | 0.5% |
| Boost Stumps | 16 | 16.5% | 0.8% |

# Applications of Boosting

# Applications of Boosting

- Academic research
- Applied research
- Commercial deployment

# Academic research

## % test error rates

| Database | Other | Boosting | Error reduction |
|---|---|---|---|
| Cleveland | 27.2 (DT) | 16.5 | 39% |
| Promoters | 22.0 (DT) | 11.8 | 46% |
| Letter | 13.8 (DT) | 3.5 | 74% |
| Reuters 4 | 5.8, 6.0, 9.8 | 2.95 | ~60% |
| Reuters 8 | 11.3, 12.1, 13.4 | 7.4 | ~40% |

Schapire, Singer, Gorin 98

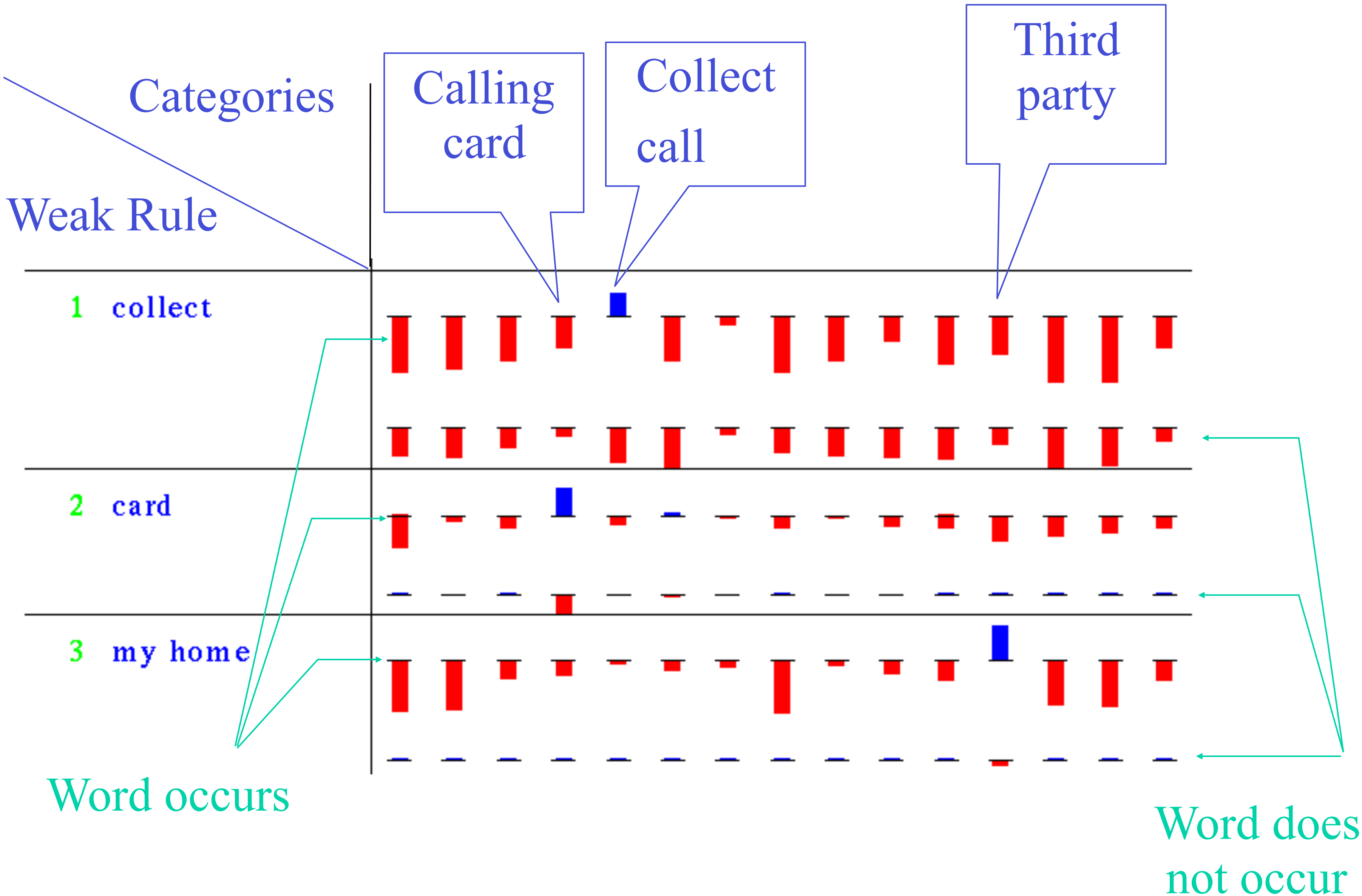
# Applied research

- *"AT&T, How may I help you?"*
  - Classify voice requests
  - Voice -> text -> category
  - Fourteen categories
    - Area code,
    - AT&T service,
    - billing credit,
    - calling card,
    - collect,
    - competitor
    - competitor,
    - dial assistance,
    - directory,
    - how to dial,
    - person to person,
    - rate,
    - third party,
    - time charge

# Example transcribed sentences

- Yes I'd like to place a collect call long distance please
  - collect
- Operator I need to make a call but I need to bill it to my office
  - third party
- Yes I'd like to place a call on my master card please
  - calling card
- I just called a number in Sioux city and I musta rang the wrong number because I got the wrong party and I would like to have that taken off my bill
  - billing credit

# Weak rules generated by “boostexter”



# Results

- 7844 training examples
  - hand transcribed
- 1000 test examples
  - hand / machine transcribed
- Accuracy with 20% rejected
  - Machine transcribed: 75%
- Hand transcribed: 90%

Freund, Mason, Rogers, Pregibon, Cortes 2000

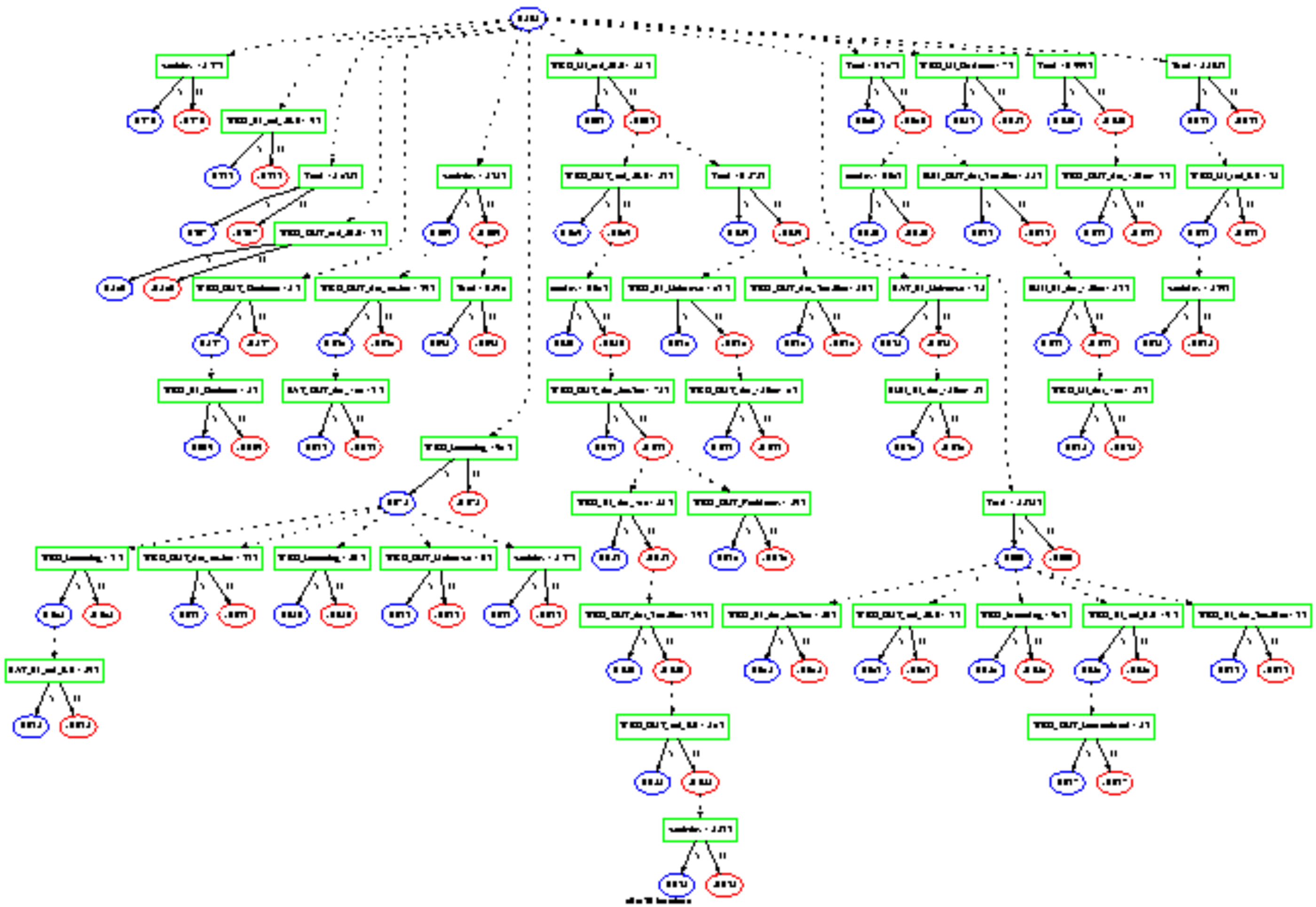
# Commercial deployment

- Distinguish business/residence customers
- Using statistics from call-detail records
- Alternating decision trees
  - Combines very simple rules
  - Can over-fit, cross validation used to stop

# Massive datasets (for 1997)

- 260M calls / day
- 230M telephone numbers
  - Label unknown for ~30%
- Hancock: software for computing statistical signatures.
- 100K randomly selected training examples,
- ~10K is enough
  - Training takes about 2 hours.
  - Generated classifier has to be both accurate and efficient

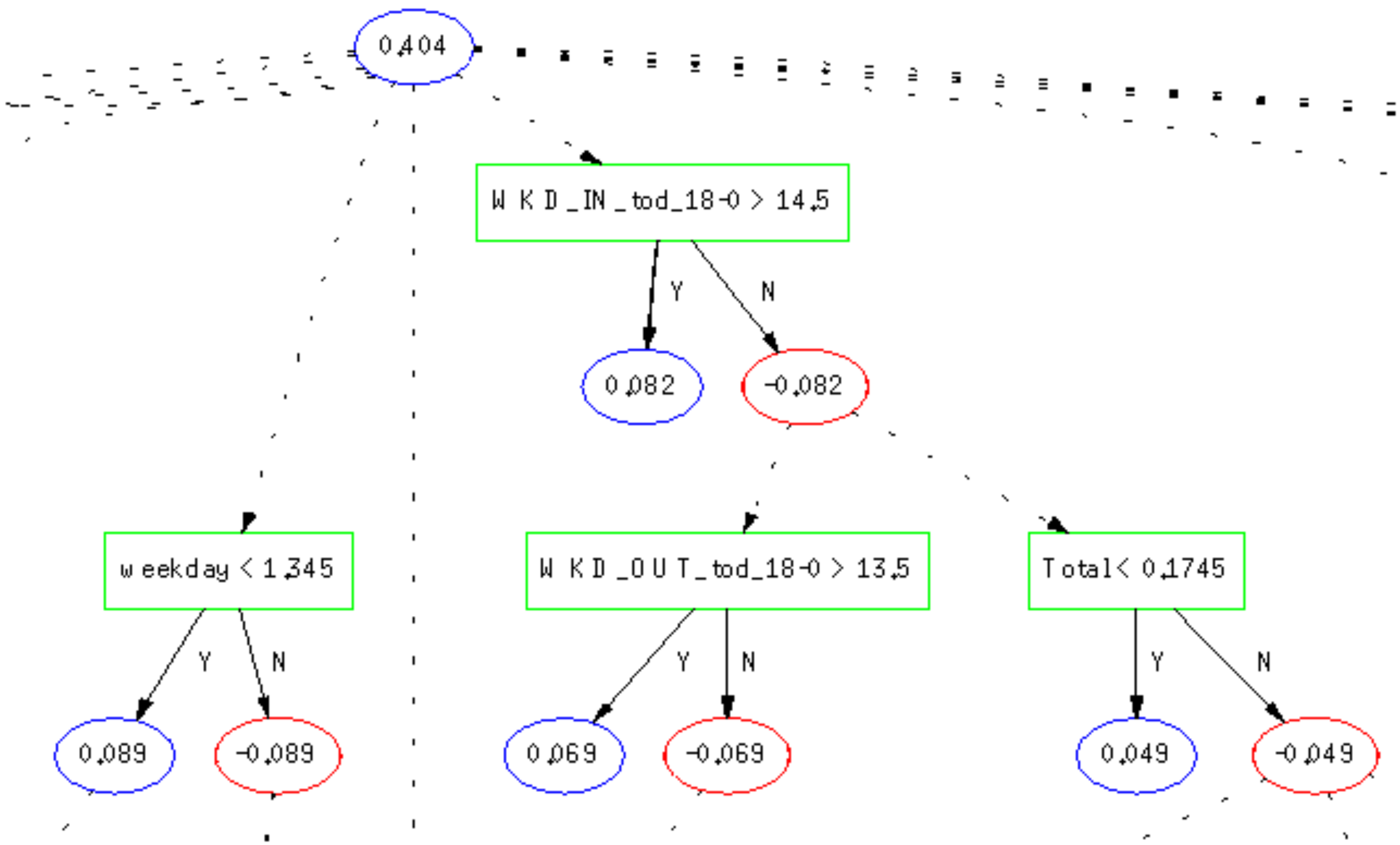
# Alternating tree for “buizocity”



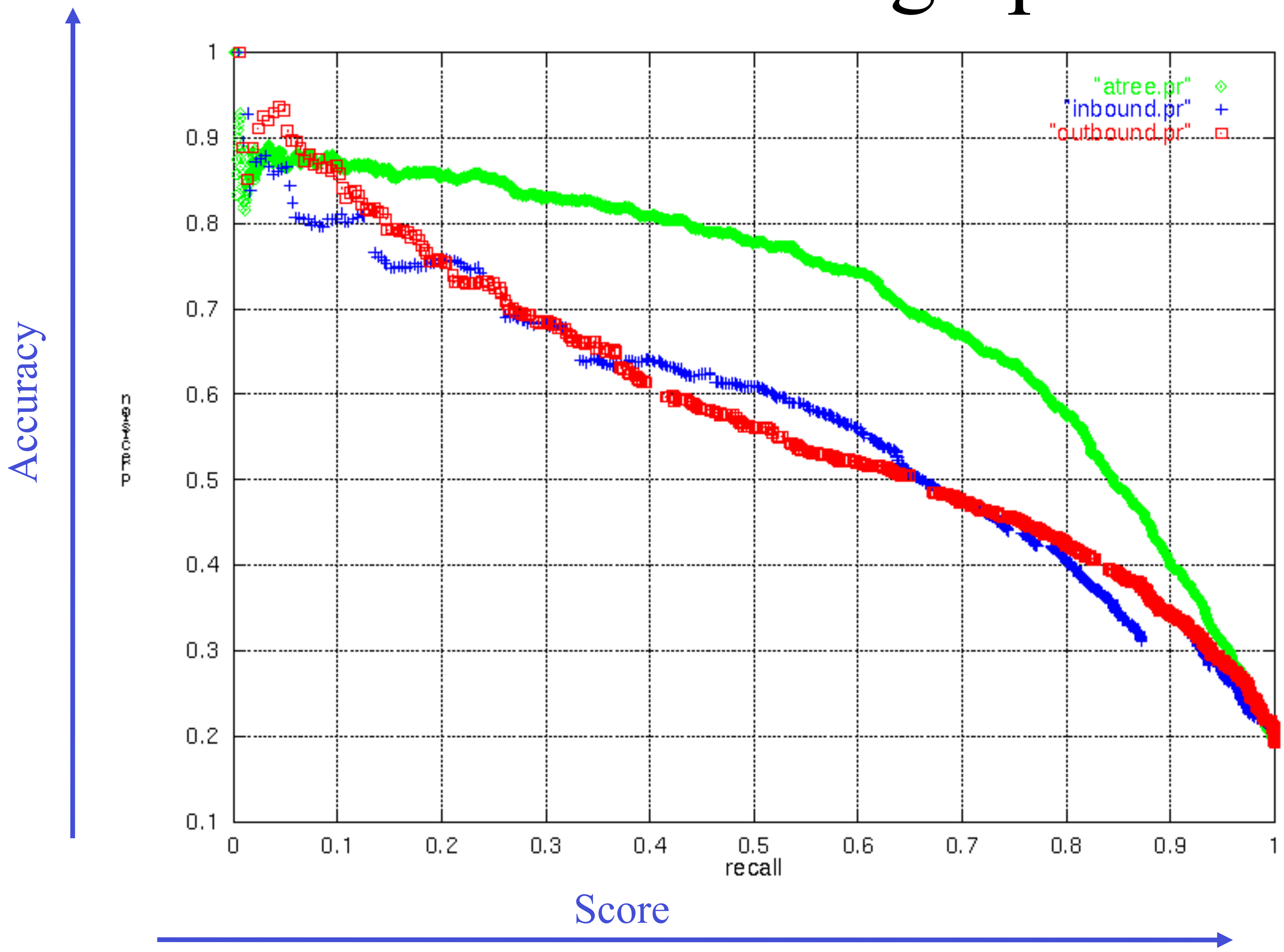
# Alternating Tree (Detail)

Positive predictions ⇔ Residences

Negative predictions ⇔ Businesses



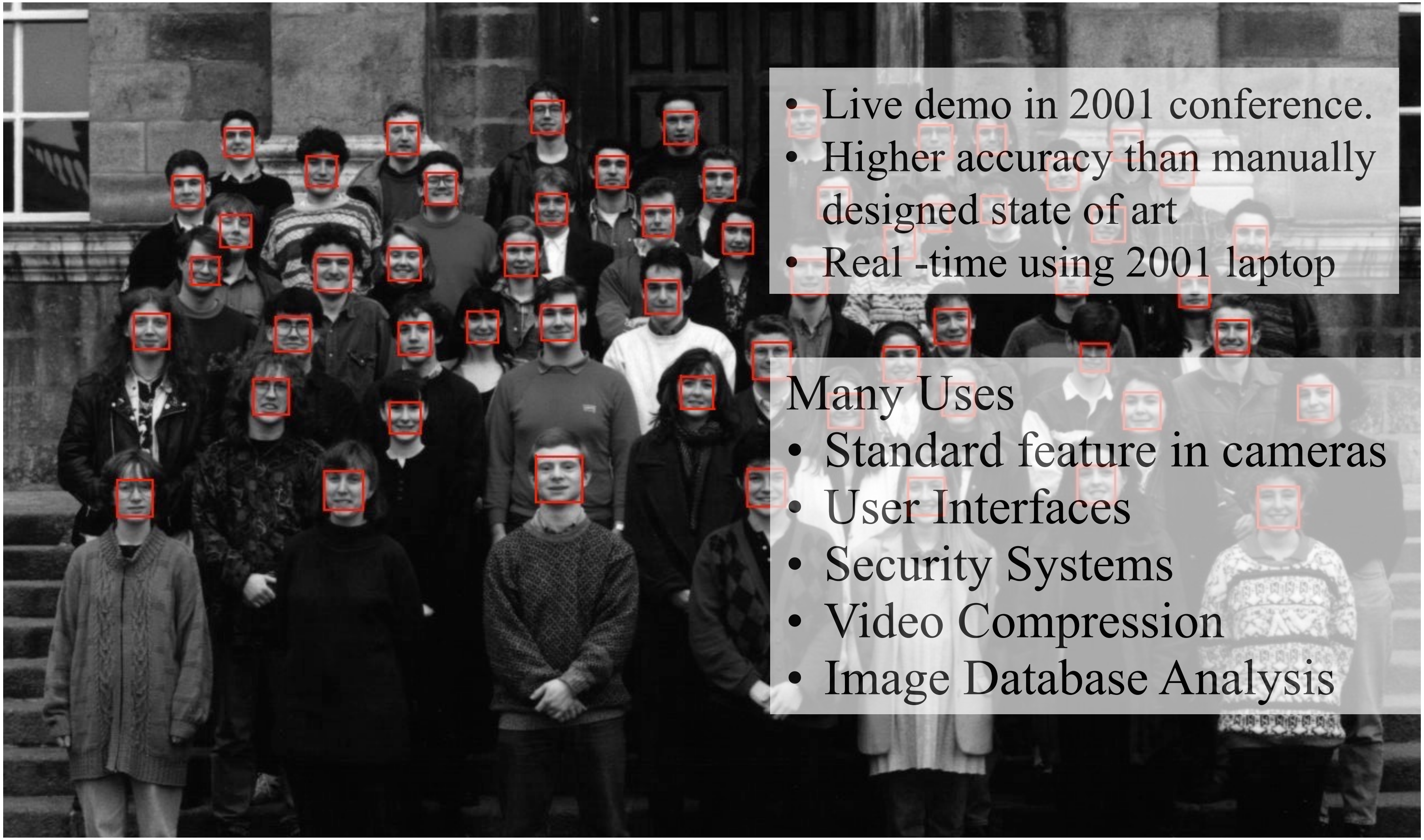
# Precision/recall graphs



# Face Detection

Viola and Jones / 2001

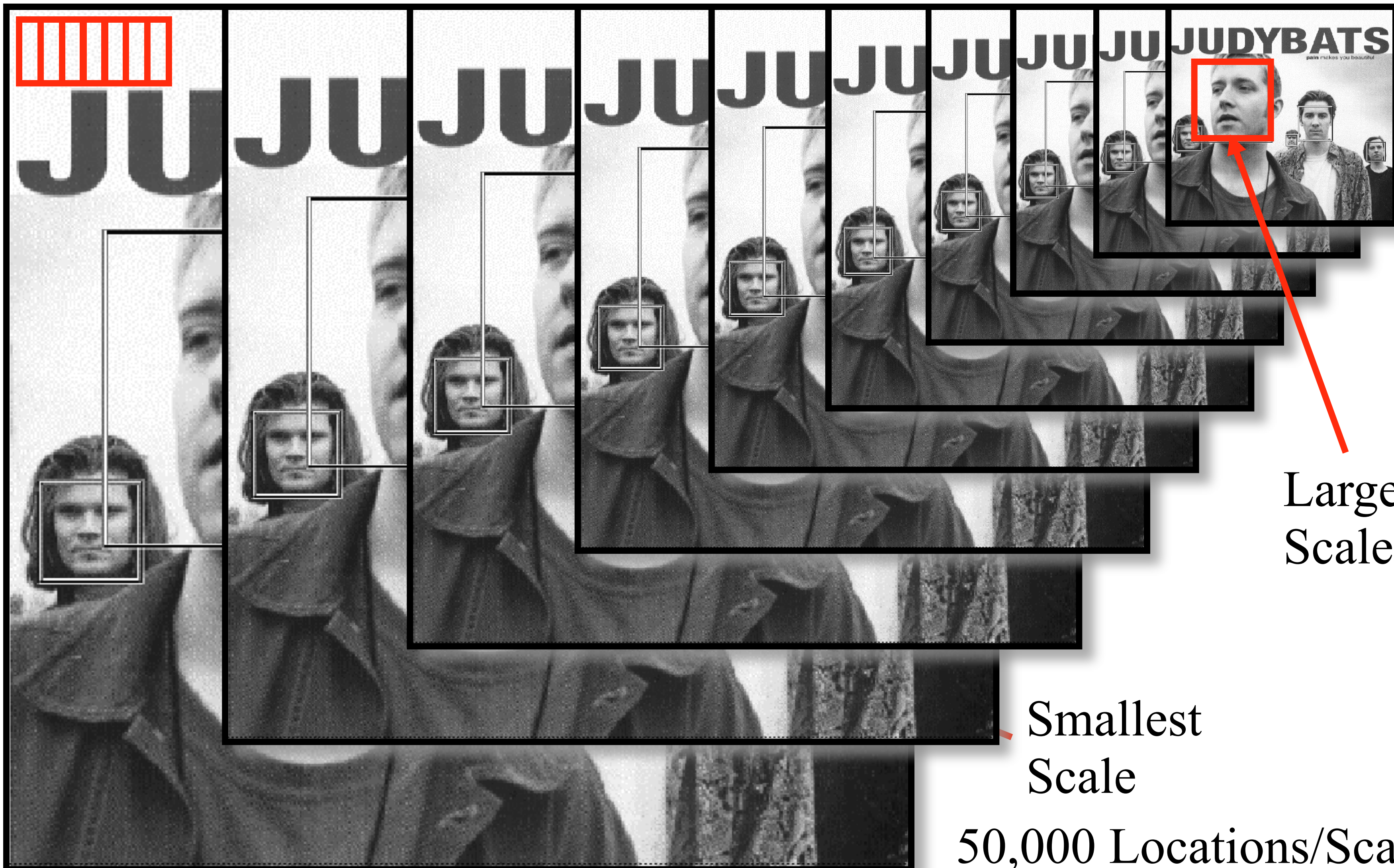
# Face Detection / Viola and Jones



- Live demo in 2001 conference.
- Higher accuracy than manually designed state of art
- Real -time using 2001 laptop

Many Uses
- Standard feature in cameras
- User Interfaces
- Security Systems
- Video Compression
- Image Database Analysis

# Face Detection as a Filtering process




Larger Scale

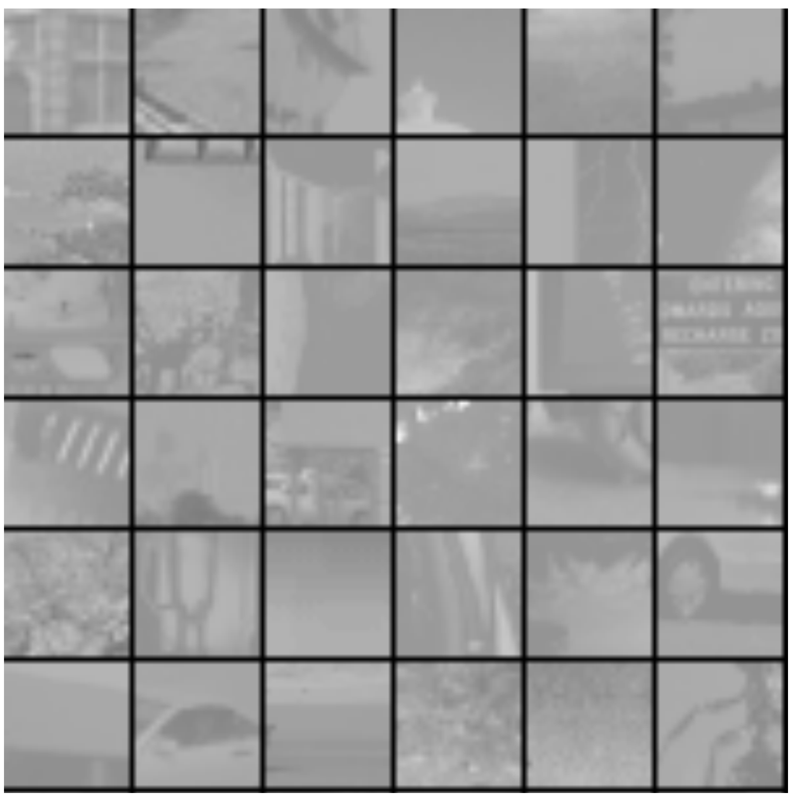
Smallest Scale

50,000 Locations/Scales

*Most Negative*

# Classifier is Learned from Labeled Data

- Example: 28x28 image patch
- Label: Face / Non face
- 5000 faces, $10^8$ non faces
- Faces are normalized
  - Scale, translation
  - Rotation remains…

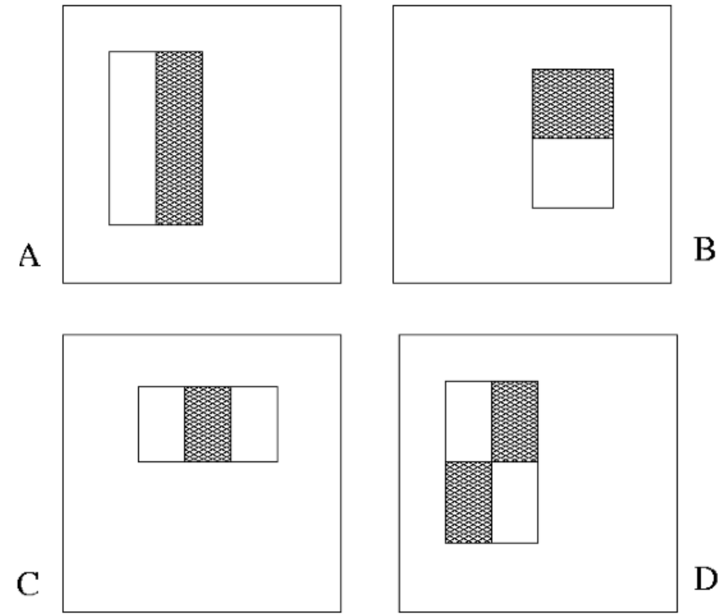


# Image Features

"Rectangle filters"

Similar to Haar wavelets

Papageorgiou, et al.




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

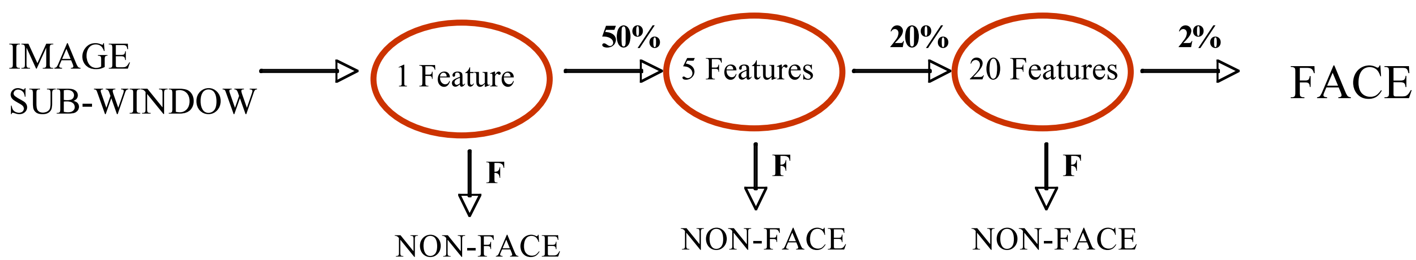
Very fast to compute using "integral image".

$$\boxed{\begin{array}{c} 60{,}000 \times 100 = 6{,}000{,}000 \\ \text{Unique Binary Features} \end{array}}$$

Combined using adaboost

# Cascaded boosting

- Features combined using Adaboost
  - Find best weak rule by exhaustive search.
  - Ran for 2 days on a 250 node cluster
- For detection, features combined in a cascade:




  - Runs in real time, 15 FPS, on laptop

# Paul Viola and Mike Jones

# Summary

- Boosting is a computational method for learning accurate classifiers
  - Resistance to over-fit explained by margins
  - Boosting has been applied successfully to a variety of classification problems