# CAP 5610: Machine Learning

## Lecture 12:
## Boosting

Instructor: Dr. Gita Sukthankar
Email: gitars@eecs.ucf.edu

Reference Materials From: GuoJun Qi

# HW1 Notes

MLE vs. MAP

Difference is whether a prior distribution is maintained over the value of the variables or whether everything is learned as a hyperparameter

- Step 2: Estimate the prior distribution

$$P(y = d) = \theta_d \, with \sum_{d=0}^{9} \theta_d = 1$$

- Solution 1: Maximum likelihood estimation

$$\theta_d = \frac{\#\ of\ digit\ d\ in\ traing\ set}{\#\ of\ total\ digits\ in\ training\ set}$$

- Solution 2: Maximum A Posterior parameter estimation
  - Imposing a prior $\mathrm{P}(\theta)$ on the parameter of prior distribution $\mathrm{P}(\mathrm{y}|\theta)$ : Prior on prior
  - Instead of only estimating a single point for $\theta$, we estimate a posterior distribution $\mathrm{P}(\theta|D_Y)$ over all possible $\theta$, where $D_Y$ is the class labels for training examples
  - Dirichlet distribution $\mathrm{P}(\theta) \propto \theta_1^{\alpha_1 - 1} \ldots \theta_9^{\alpha_9 - 1}$, conjugate to $\mathrm{P}(\mathrm{y}|\theta)$
  - By the property of conjugate distribution, we have

$$\underset{\theta_d}{\mathrm{argmax}}\ \mathrm{P}(\theta_d|D_Y) = \frac{\#\ of\ digit\ d\ in\ trainng\ set + \alpha_d}{\#\ of\ \ trainng\ examples + \sum_{d=0}^{9} \alpha_d}$$

# Next Week’s Schedule

- Class cancelled on Tuesday.
- Exam review on Thursday.

# Reading

Marsland Chapter 13

Bishop Chapter 14

Murphy Chapter 16.4

# Bootstrap Estimation and Bagging

- Repeatedly draw *n* samples from *D*
- For each set of samples, estimate a statistic
- The bootstrap estimate is the mean of the individual estimates
- Used to estimate a statistic (parameter) and its variance

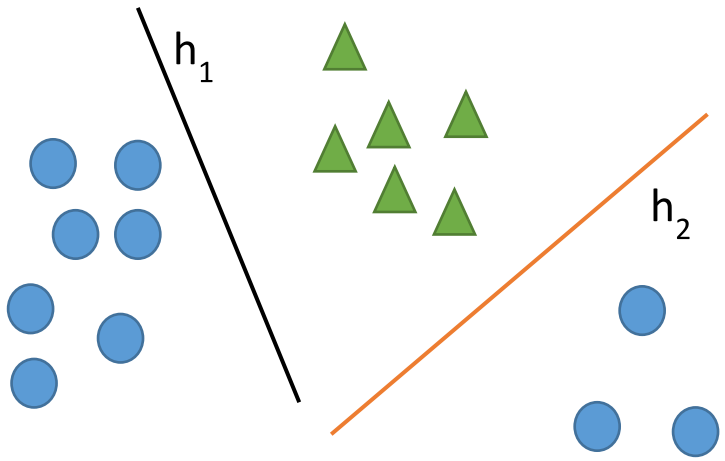
**Bagging (Bootstrap Aggregation)** trains classifiers on bootstrap samples and then uses voting to decide on the class.

# Weak classifiers are useful!

- Weak classifiers
  - Decision stump – one layer decision tree
  - Naive Bayes – A classifier without feature correlations
  - Linear classifier – logistic regression
- Weak classifiers usually have larger training error but smaller variance.
- A single weak classifier is usually not adequate in real applications, but it is possible to combine an ensemble of weak classifiers to build a strong one.

# Idea: Weighted Voting

- Combining an ensemble of weak classifiers by weighted voting
  - Learning an ensemble of weak classifiers
    - Although each weak classifier is not adequate of classifying the whole feature space, it can still output good result on certain parts of feature spaces.
  - Each weak classifier is given a weight based on its performance
    - Better classifiers will vote with more weight.
    - Weighted voting usually generates better performance by combining complementary classifiers  good at classifying different parts of feature spaces.



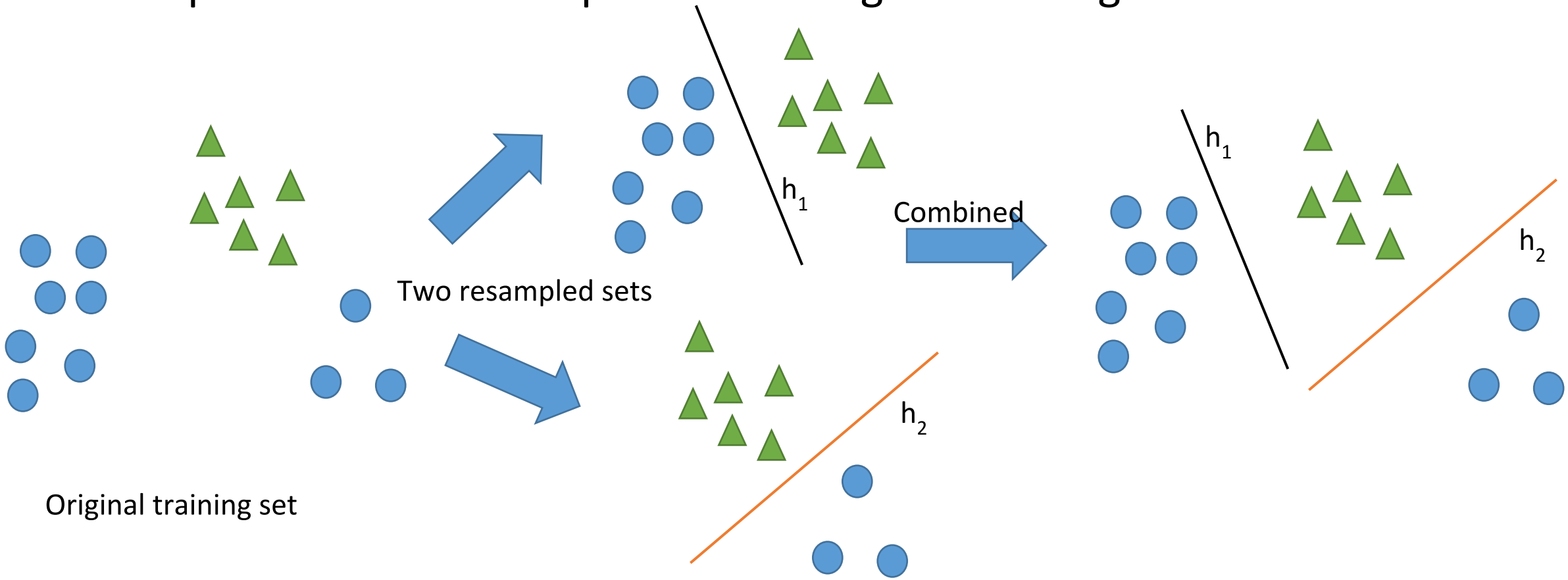

Combined classifier: $f(x) = \text{sign}\left(\alpha_1 h_1(x) + \alpha_2 h_2(x)\right)$

# Problems to solve

- How are an ensemble of weak classifiers learned?
  - Decide which aspect of the task each weak classifier focuses on.
- How is the weight of each weak classifier decided?

# Boosting [Schapire 1998]

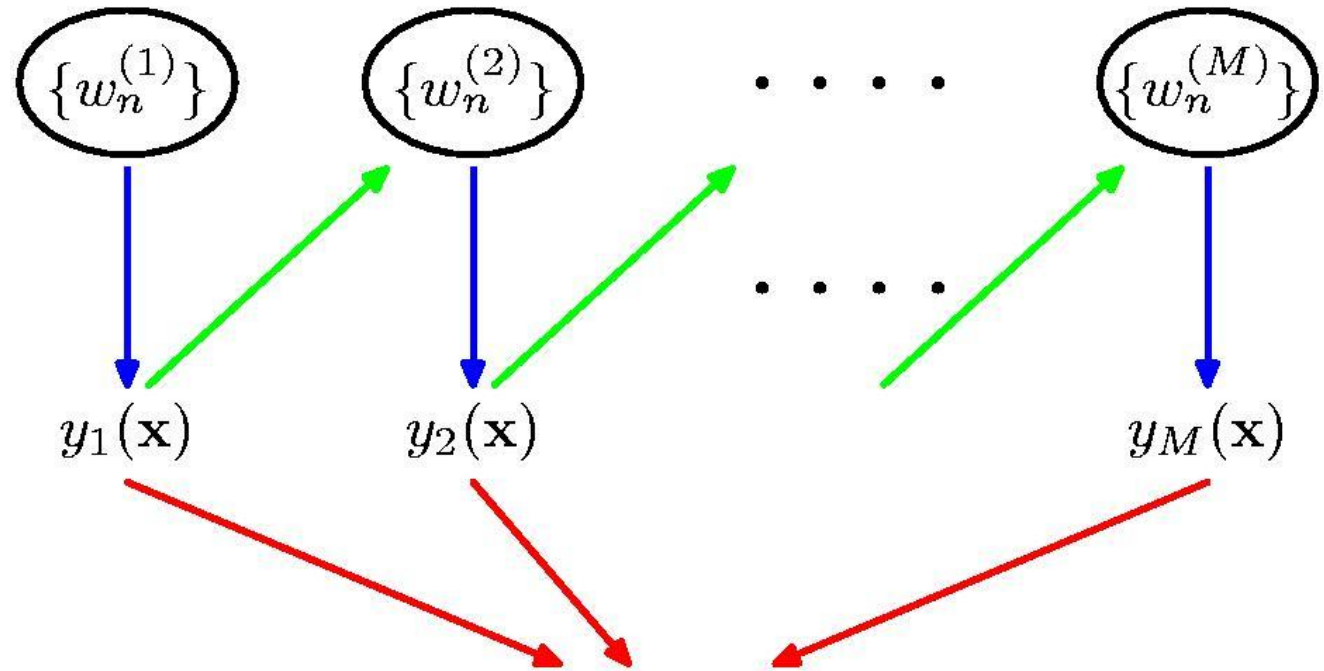
- Idea: learning a pool of weak classifiers (usually of the same type e.g., stump, logistic regression), on different sets of training examples resampled from different parts of an original training set



# Boosting – The Algorithm

- On each iteration t:
  - Weight each training example by how correctly it is classified so far
  - Learn a weak classifier $h_t$ that best classifies the weighted training examples.
  - Decide a strength for this weak classifier $\alpha_t$
- Final classifier: $f(x) = \text{sign}(\sum_t \alpha_t h_t(x))$

# Boosting Procedure




$$Y_M(\mathbf{x}) = \text{sign}\left(\sum_m^M \alpha_m y_m(\mathbf{x})\right)$$

(note: assumes binary classification problem but there have been multi-class extensions)

# Learning from Weighted Training Examples

- Consider a weighted dataset
  - D(i) – weight of i-th training example $\left(x^{(i)}, y^{(i)}\right)$
  - Interpretations:
    - *i*-th example is counted as D(i) examples
    - *i*-th example is resampled from training set by weight D(i)
- Two ways to learn a weak classifier from weighted training examples
  - Resampling the training set by D(i), and train a weak classifier from the resampled set
  - Learn a weak classifier directly from the weighted samples, e.g., a weighted logistic regression classifier

$$h = \min_{\mathrm{h}} \sum_{i} D(i) loss(h(x^{(i)}), y^{(i)})$$

# Decide the combination weight for each weak classifier

- Weight of weak classifier $\alpha_t = \frac{1}{2}\log\frac{1-\epsilon_t}{\epsilon_t}$

Where $\epsilon_t$ is the weighted training error

$$\epsilon_t = \sum_i D(i)\delta(h_t(x_i) \neq y_i)$$

If a classifier is better than a random guess, $\epsilon_t < 0.5$, and $\alpha_t > 0$; otherwise, $\alpha_t < 0$. For the latter case, it is an adverse classifier rather than a weak classifier.

Don’t want any classifier with exactly 0.5 error since it is non-informative.

# AdaBoost (Freund and Shapire '95)

- 1. Initialize the data weighting coefficients $\{w_n\}$ by setting $w_n^{(1)} = 1/N$ for $n = 1, \ldots, N$
- 2. For $m = 1, \ldots, M$ :
- (a) Fit a classifier $y_m(x)$ to the training data by minimizing the weighted error function

$$J_m = \sum_{n=1}^{N} w_n^{(m)} I(y_m(x_n) \neq t_n)$$

- Where $I(y_m(x_n) \neq t_n)$ is the indicator function and equals 1 when $y_m(x_n) \neq t_n$ and 0 otherwise.

# AdaBoost

- (b) Evaluate the quantities

$$\varepsilon_m = \frac{\sum_{n=1}^{N} w_n^{(m)} I(y_m(x_n) \neq t_n)}{\sum_{n=1}^{N} w_n^{(m)}}$$

and then use these to evaluate

$$\alpha_m = \ln\{\frac{1-\varepsilon_m}{\varepsilon_m}\}$$

this is used to weight the classifier

# AdaBoost

- (c) Update the data weighting coefficients

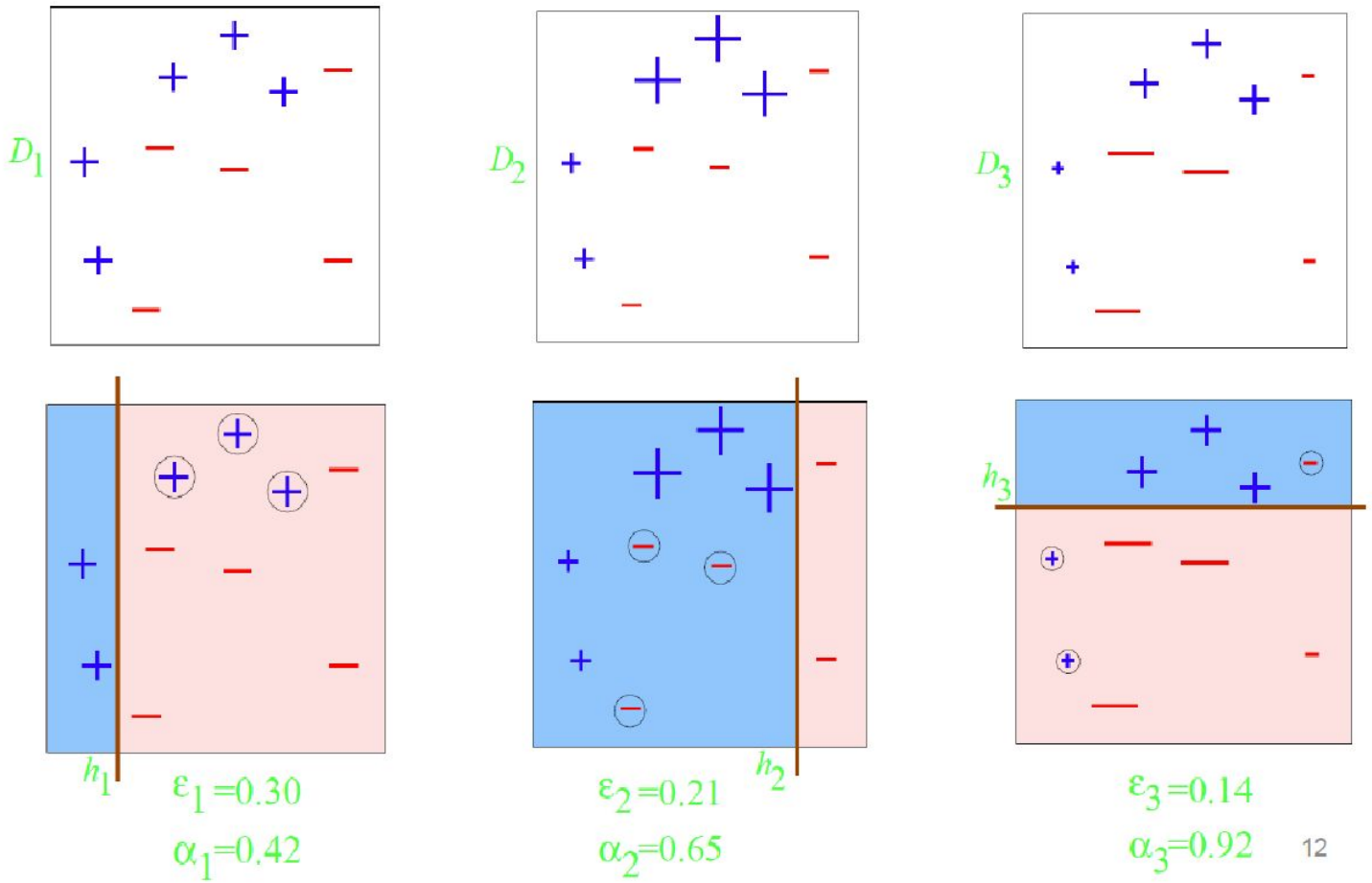
$$w_n^{(m+1)} = w_n^{(m)} \exp\{\alpha_m I(y_m(x_n) \neq t_n)\}$$

- 3. Make predictions using the final model, which is given by

$$Y_M(x) = sign(\sum_{m=1}^{M} \alpha_m y_m(x))$$

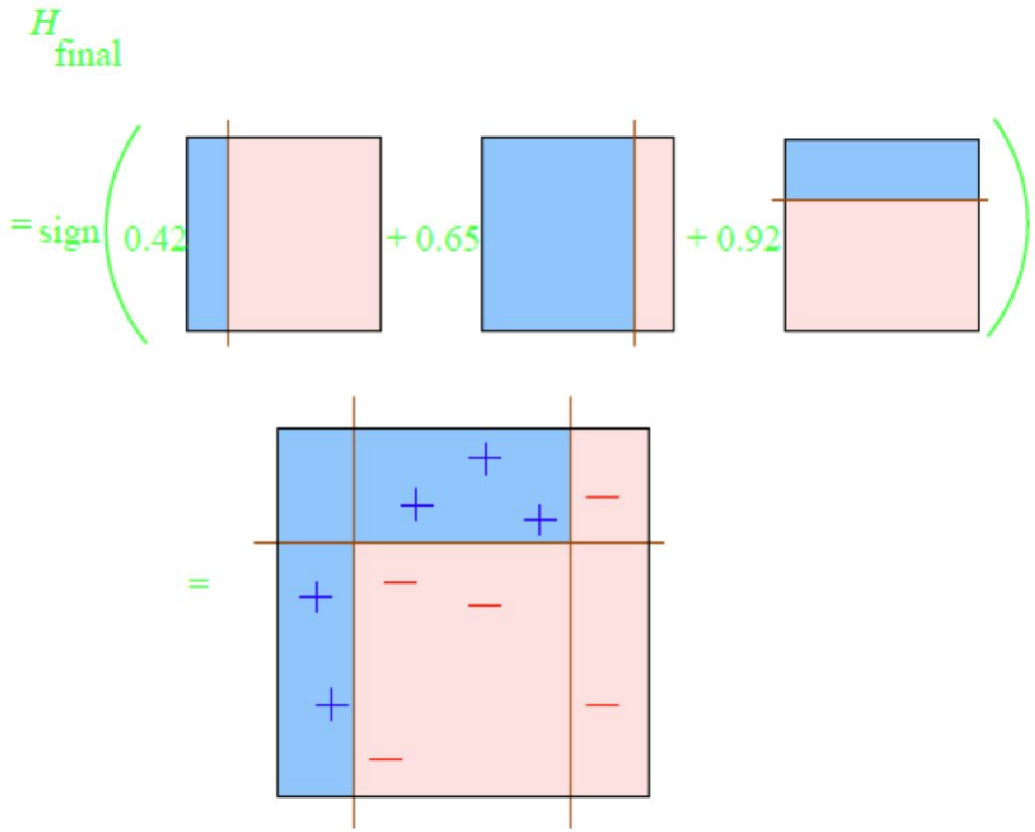
# Boosting Example (Decision Stump)

- Three weak classifiers



# Boosting Example (Decision Stump)

- Final classifier



# Algorithm recapitulation

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

- Train weak learner using distribution $D_t$. **Naïve bayes, decision stump**
- Get weak classifier $h_t : X \to \mathbb{R}$.
- Choose $\alpha_t \in \mathbb{R}$. **Magic (+ve)**
- Update:

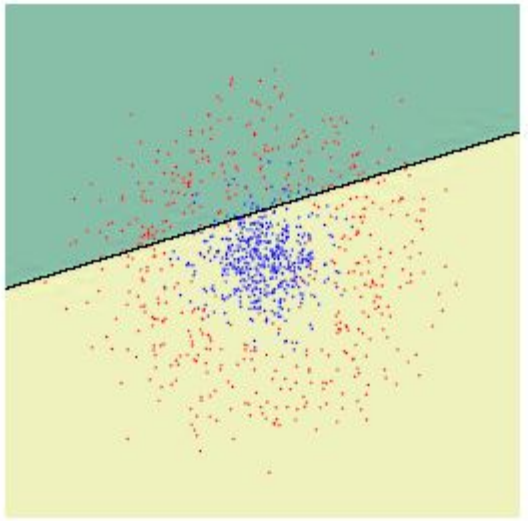
$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \begin{cases} e^{-\alpha_t} & \text{if } y_i = h_t(x_i) \\ e^{\alpha_t} & \text{if } y_i \neq h_t(x_i) \end{cases}$$

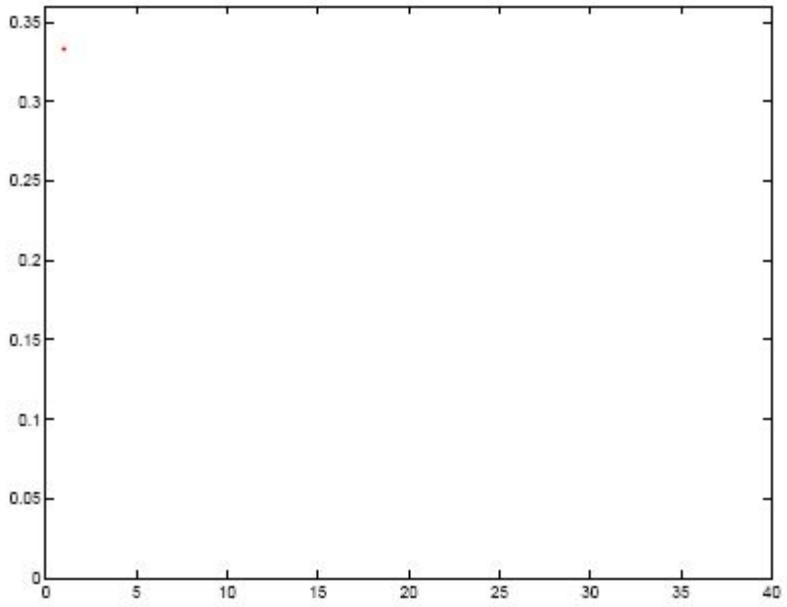
$$= \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

**Increase weight if wrong on pt i**
**$y_i h_t(x_i) = -1 < 0$**

where $Z_t$ is a normalization factor

$t = 1$





# Algorithm recapitulation

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

- Train weak learner using distribution $D_t$. **Naïve bayes, decision stump**
- Get weak classifier $h_t : X \to \mathbb{R}$.
- Choose $\alpha_t \in \mathbb{R}$. **Magic (+ve)**
- Update:

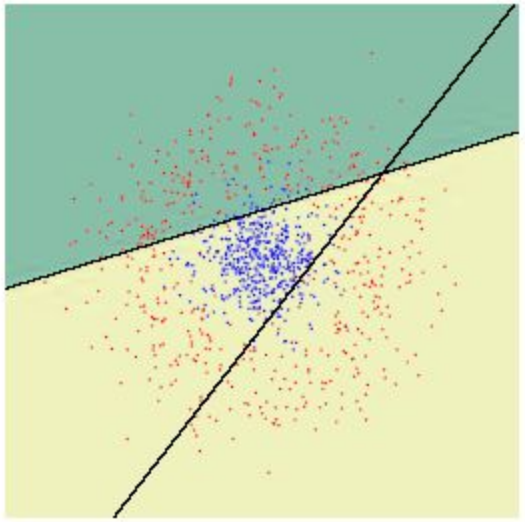
$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \begin{cases} e^{-\alpha_t} & \text{if } y_i = h_t(x_i) \\ e^{\alpha_t} & \text{if } y_i \neq h_t(x_i) \end{cases}$$

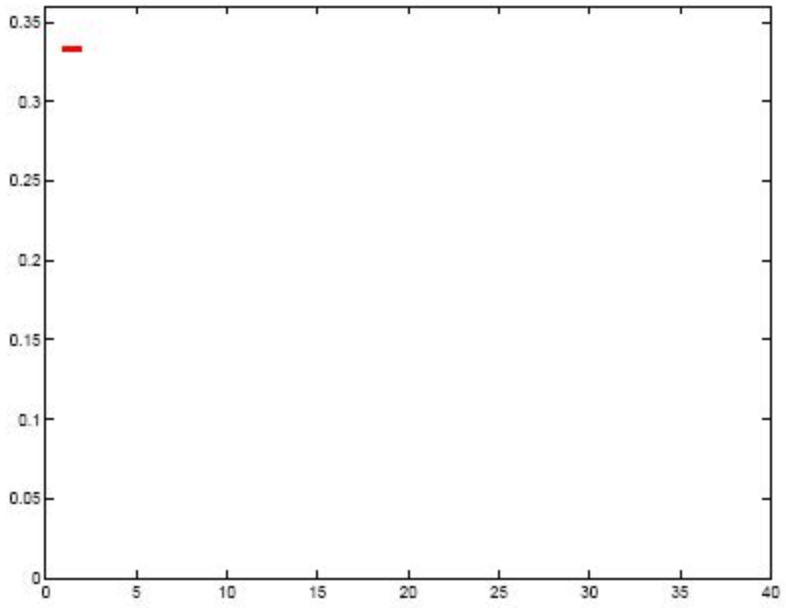
$$= \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

**Increase weight if wrong on pt i**
**$y_i h_t(x_i) = -1 < 0$**

where $Z_t$ is a normalization factor

$t = 2$





# Algorithm recapitulation

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

- Train weak learner using distribution $D_t$. **Naïve bayes, decision stump**
- Get weak classifier $h_t : X \to \mathbb{R}$.
- Choose $\alpha_t \in \mathbb{R}$. **Magic (+ve)**
- Update:

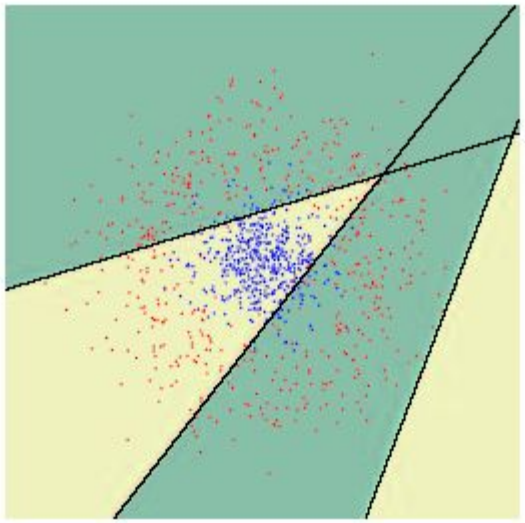
$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \begin{cases} e^{-\alpha_t} & \text{if } y_i = h_t(x_i) \\ e^{\alpha_t} & \text{if } y_i \neq h_t(x_i) \end{cases}$$

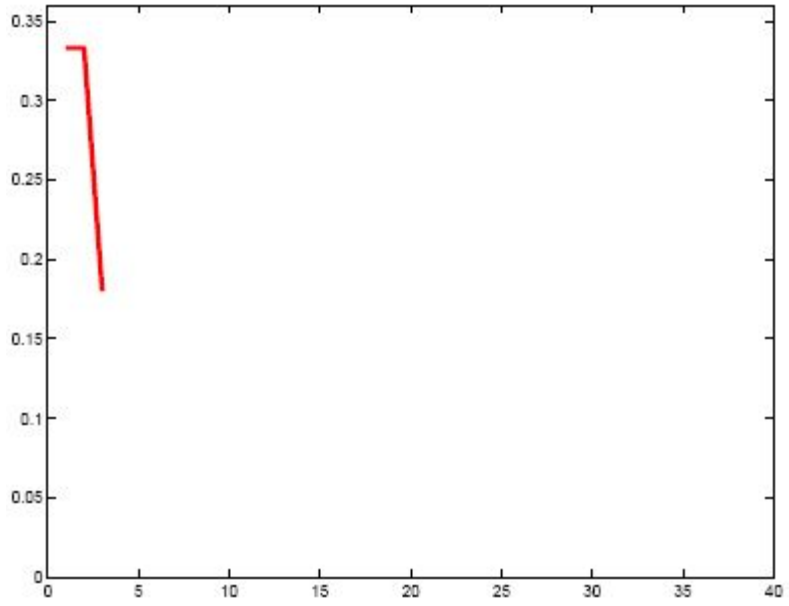
$$= \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

**Increase weight if wrong on pt i**
**$y_i h_t(x_i) = -1 < 0$**

where $Z_t$ is a normalization factor

$t = 3$





# Algorithm recapitulation

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

- Train weak learner using distribution $D_t$. **Naïve bayes, decision stump**
- Get weak classifier $h_t : X \to \mathbb{R}$.
- Choose $\alpha_t \in \mathbb{R}$. **Magic (+ve)**
- Update:

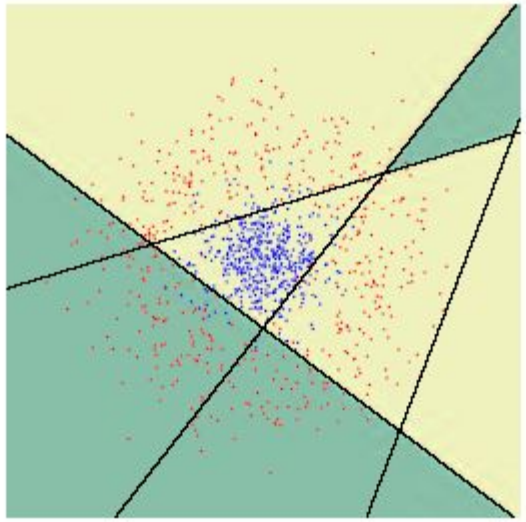
$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \begin{cases} e^{-\alpha_t} & \text{if } y_i = h_t(x_i) \\ e^{\alpha_t} & \text{if } y_i \neq h_t(x_i) \end{cases}$$

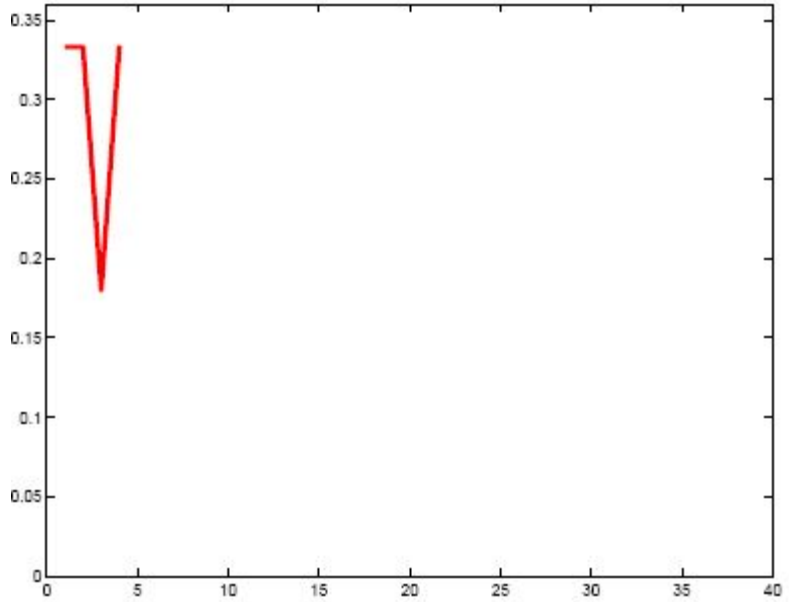
$$= \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

**Increase weight if wrong on pt i**
**$y_i h_t(x_i) = -1 < 0$**

where $Z_t$ is a normalization factor

$t = 4$





# Algorithm recapitulation

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

- Train weak learner using distribution $D_t$. **Naïve bayes, decision stump**
- Get weak classifier $h_t : X \to \mathbb{R}$.
- Choose $\alpha_t \in \mathbb{R}$. **Magic (+ve)**
- Update:

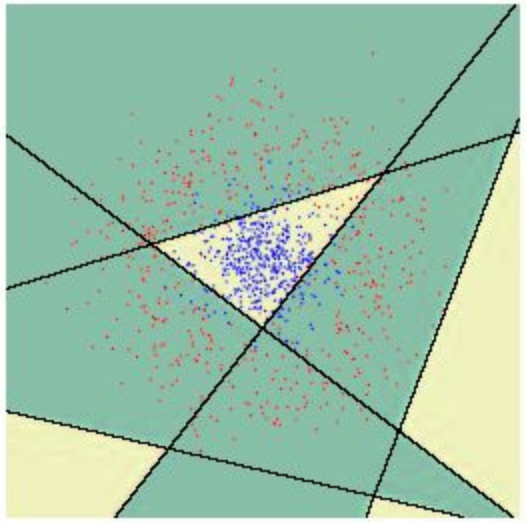
$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \begin{cases} e^{-\alpha_t} & \text{if } y_i = h_t(x_i) \\ e^{\alpha_t} & \text{if } y_i \neq h_t(x_i) \end{cases}$$

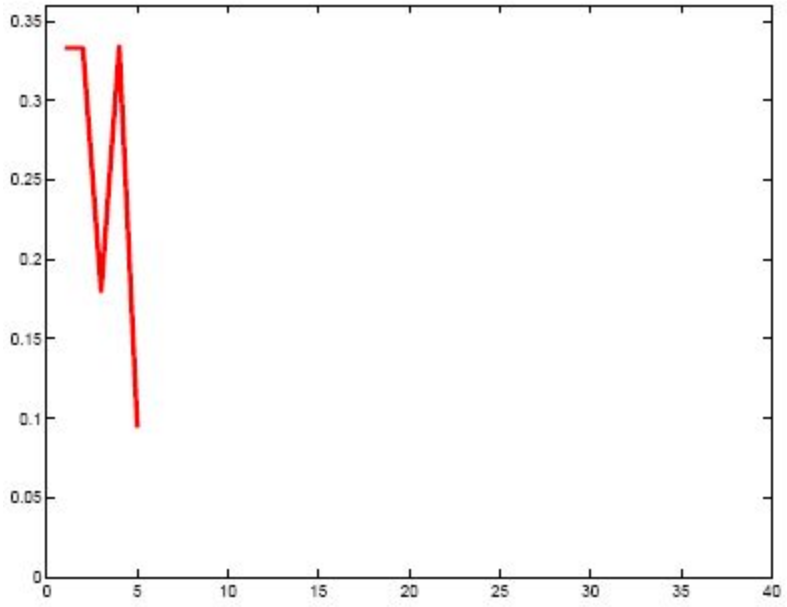
$$= \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

**Increase weight if wrong on pt i**
**$y_i h_t(x_i) = -1 < 0$**

where $Z_t$ is a normalization factor

$t = 5$





# Algorithm recapitulation

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

- Train weak learner using distribution $D_t$. **Naïve bayes, decision stump**
- Get weak classifier $h_t : X \to \mathbb{R}$.
- Choose $\alpha_t \in \mathbb{R}$. **Magic (+ve)**
- Update:

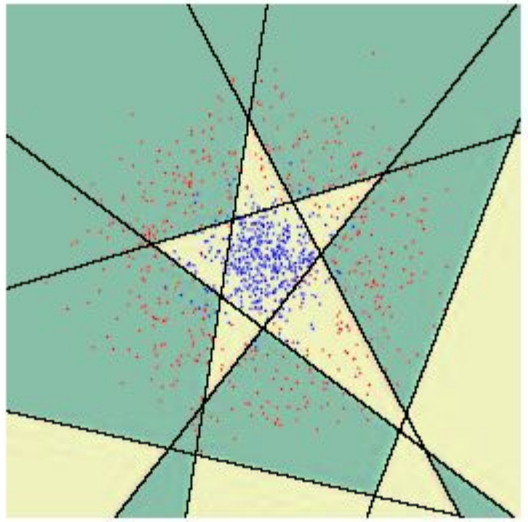
$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \begin{cases} e^{-\alpha_t} & \text{if } y_i = h_t(x_i) \\ e^{\alpha_t} & \text{if } y_i \neq h_t(x_i) \end{cases}$$

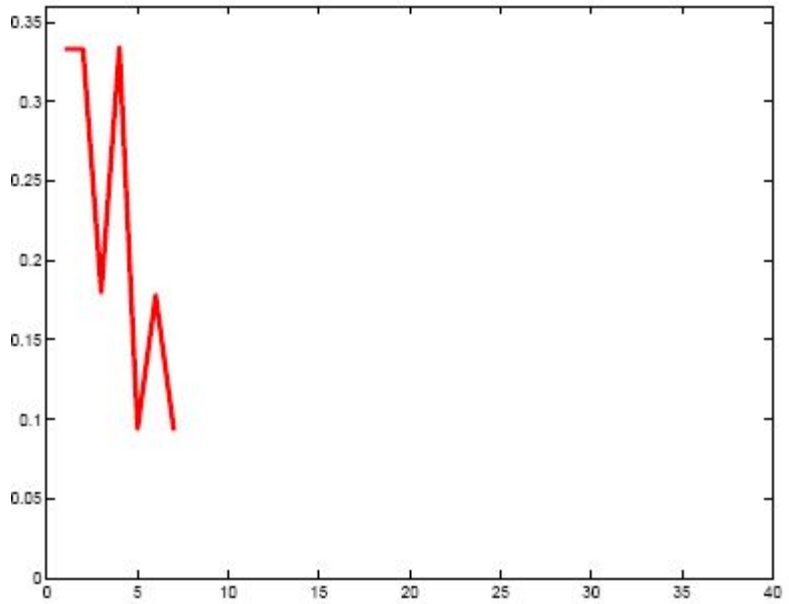
$$= \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

**Increase weight if wrong on pt i**
**$y_i h_t(x_i) = -1 < 0$**

where $Z_t$ is a normalization factor

$t = 7$





# Algorithm recapitulation

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

- Train weak learner using distribution $D_t$. **Naïve bayes, decision stump**
- Get weak classifier $h_t : X \to \mathbb{R}$.
- Choose $\alpha_t \in \mathbb{R}$. **Magic (+ve)**
- Update:

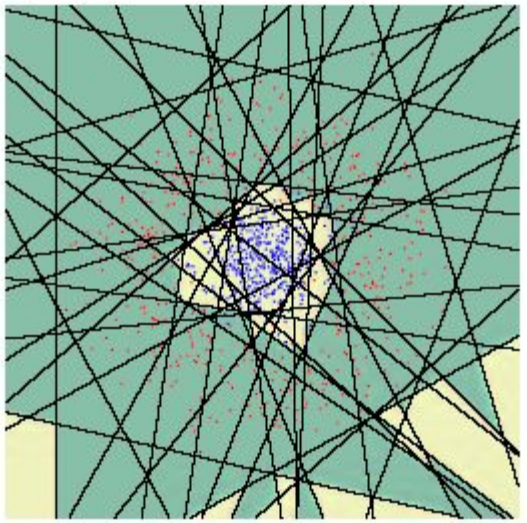
$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \begin{cases} e^{-\alpha_t} & \text{if } y_i = h_t(x_i) \\ e^{\alpha_t} & \text{if } y_i \neq h_t(x_i) \end{cases}$$

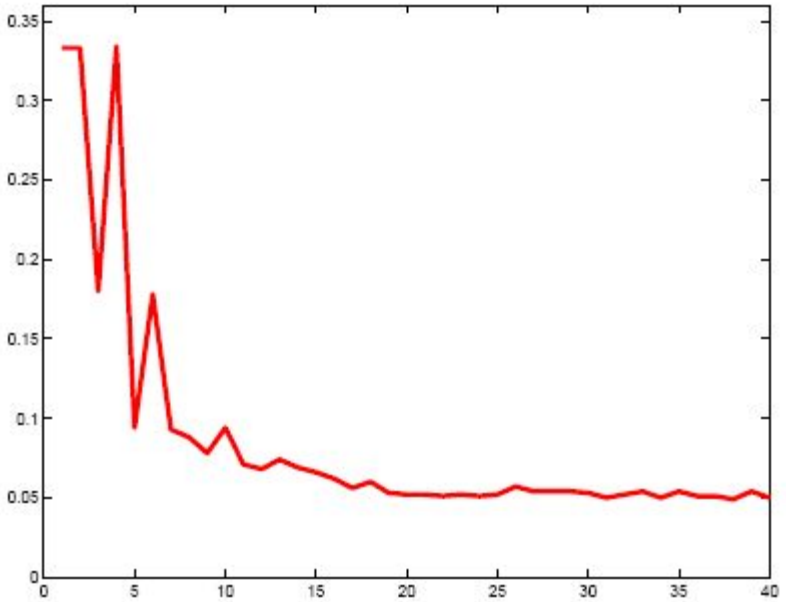
$$= \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

**Increase weight if wrong on pt i**
**$y_i\, h_t(x_i) = -1 < 0$**

where $Z_t$ is a normalization factor

$t = 40$

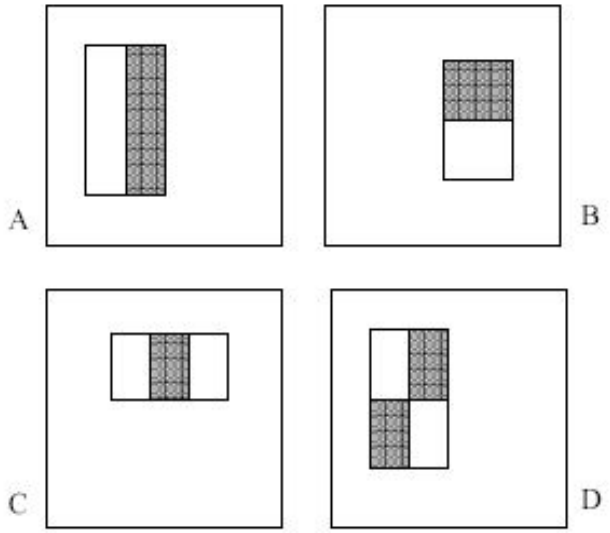




# Viola-Jones Face Detector

- Scan the input at many scales
- Starting at the base scale in which faces are detected at 24x24 pixels, a 384x288 pixel image is scanned at 12 scales each a factor 1.25 larger than the last
- Any rectangle feature can be evaluated at any scale and location in a few operations
- Face detection @15 fps for the entire image

# Image Features

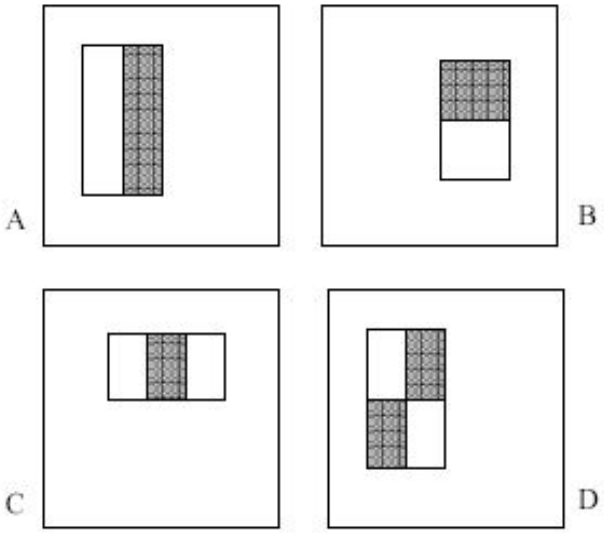



Rectangular filters

$$f(x,y) = \sum_i p_b(i) - \sum_i p_w(i)$$

Local features: Subtract sum of pixels in white area from the sum of pixels in black area; 2-rectangle features (A and B), 3-rectangle feature (C) and 4-rectangle feature (D)

# Image Features




Rectangular filters

$$f(x, \ldots) = \sum_i p_b(i) - \sum_i p_w(i)$$

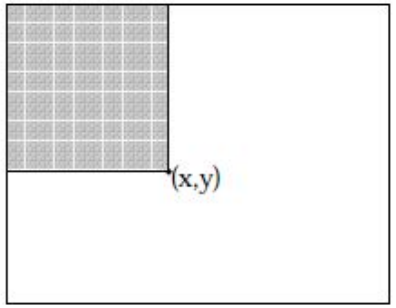
Local features: Subtract sum of pixels in black from sum of pixels in white

Too many features

In a 24 x 24 patch with 4x4 detector, there are over 160,000 locations for rectangles

# Image Features

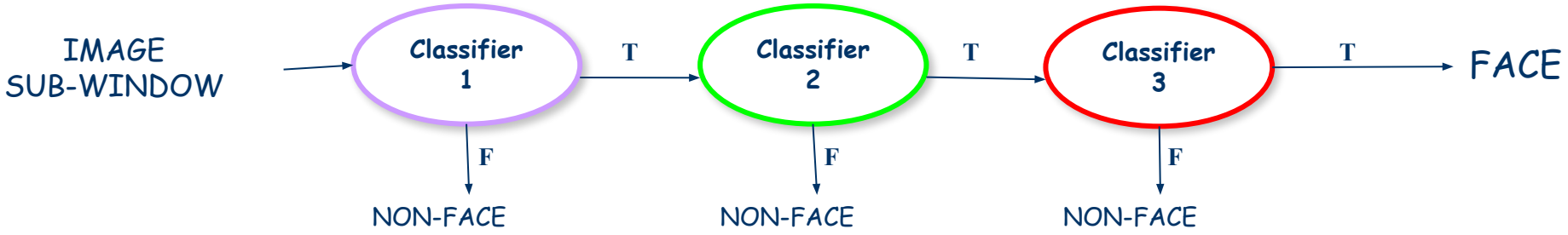
- Integral Image: An intermediate representation of the image for rapid calculation of rectangle features




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

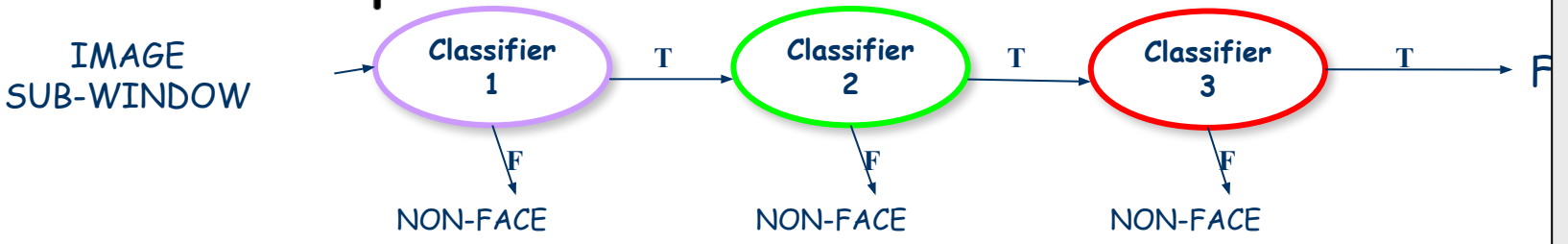
$$\sum_{\substack{x_0 < x \le x_1 \\ y_0 < y \le y_1}} i(x,y) = I(D) + I(A) - I(B) - I(C)$$

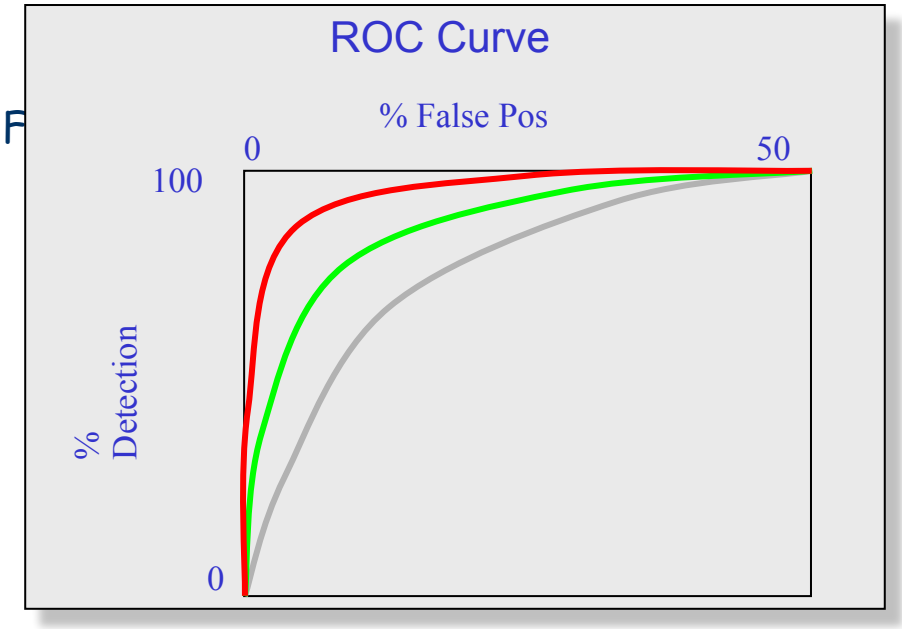
- Reject non-face windows through a cascade of classifiers and boosting



# Cascade of classifiers

- Train each classifier with increasing number of features until the target false positive and detection rates are achieved
- Use false positives from current stage as the negative training examples for the next stage
- Classifiers are progressively more complex and have lower false positive rates




False positive rate of cascade $F = \prod_{i=1}^{K} f_i$

Detection rate of cascade $D = \prod_{i=1}^{K} d_i$

# Viola Jones Face Detector

https://www.youtube.com/watch?v=hPCTwxF0qf4&t=95s

# How good can the boosting reduce the training error?

- If each weak classifier $h_t$ is slightly better than random guess with $\epsilon_t < 0.5$, then the training error of Adaboost can be reduced exponentially fast in the number of weak classifiers combined T,

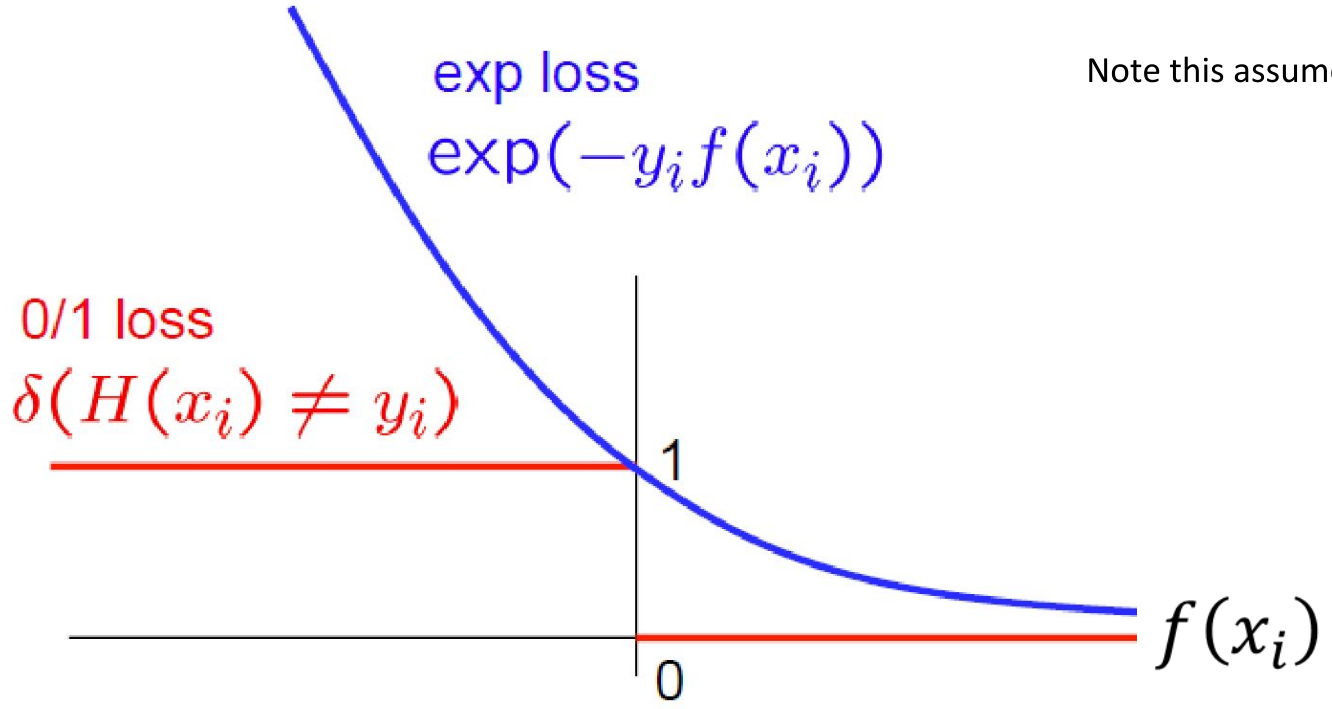
$$\frac{1}{m}\sum_{i=1}^{m}\delta(H(x_i) \neq y_i) \leq \exp\left(-2\sum_{t=1}^{T}(1/2-\epsilon_t)^2\right)$$

**Training Error**

- It is astounding result as AdaBoost can reduce the training error to arbitrarily close to zero.

# Proof: Training error of AdaBoost

- Note the fact that exponential function $\exp(-y_i f(x_i))$ is an upper bound of the 0/1 loss $\delta(y_i \neq f(x_i))$ as




Note this assumes that y=+1/-1.

# Proof: Training error of AdaBoost

- The total training error is bounded by

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

where $f(x_i)=\sum_t \alpha_t h_t(x_i)$, and $H(x_i)=\text{sign}(f(x_i))$ is the final classifier.

# Proof: Training error of AdaBoost

- $D_1(i) = \frac{1}{m}$
- $D_2(i) = \frac{\exp(-y_i \alpha_1 h_1(x_i))}{m Z_1}$
- $D_3(i) = \frac{D_2(i)\exp\left(-y_i \alpha_2 h_2(x_i)\right)}{Z_2} = \frac{\exp(-y_i \alpha_1 h_1(x_i))\exp(-y_i \alpha_2 h_2(x_i))}{m Z_1 Z_2}$

By induction, we have

- $D_T(i) = \frac{D_{T-1}\exp(-y_i \alpha_T h_T(x_i))}{Z_T} = \frac{\exp(\sum_t -y_i \alpha_t h_t(x_i))}{m Z_1 Z_2 \ldots Z_T} = \frac{\exp(-y_i f(x_i))}{m Z_1 Z_2 \ldots Z_T}$

From $\sum_i D_T(i) = 1$, we have $\prod_t Z_t = \frac{1}{m}\sum_i \exp(-y_i f(x_i))$.

# Proof: Training error of AdaBoost

- So we have

$$\frac{1}{m}\sum_i \delta\big(y_i \neq f(x_i)\big) \leq \frac{1}{m}\sum_i \exp(-y_i f(x_i)) = \Pi_t Z_t$$

- If $Z_t$<1, the training error decays exponentially.

# Proof: Training error of AdaBoost

- Finding an optimal weight $\alpha_t$ by minimizing $Z_t$

$$Z_t = \sum_i D_t(i)\exp(-\alpha_t y_i h_t(x_i))$$
$$= \sum_{h_t(x_i)\neq y_i} D_t(i)\exp(\alpha_t) + \sum_{h_t(x_i)=y_i} D_t(i)\exp(-\alpha_t)$$
$$= \exp(\alpha_t)\,\epsilon_t + \exp(-\alpha_t)\,(1-\epsilon_t)$$

$$\frac{\partial Z_t}{\alpha_t} = \exp(\alpha_t)\,\epsilon_t - \exp(-\alpha_t)\,(1-\epsilon_t) = 0, \text{thus}$$
$$\text{the optimal } \alpha_t = \frac{1}{2}\log\frac{1-\epsilon_t}{\epsilon_t}, \text{and}$$
$$Z_t = 2\sqrt{\epsilon_t(1-\epsilon_t)} = \sqrt{1-(1-2\epsilon_t)^2}$$

# Proof: Training error of AdaBoost

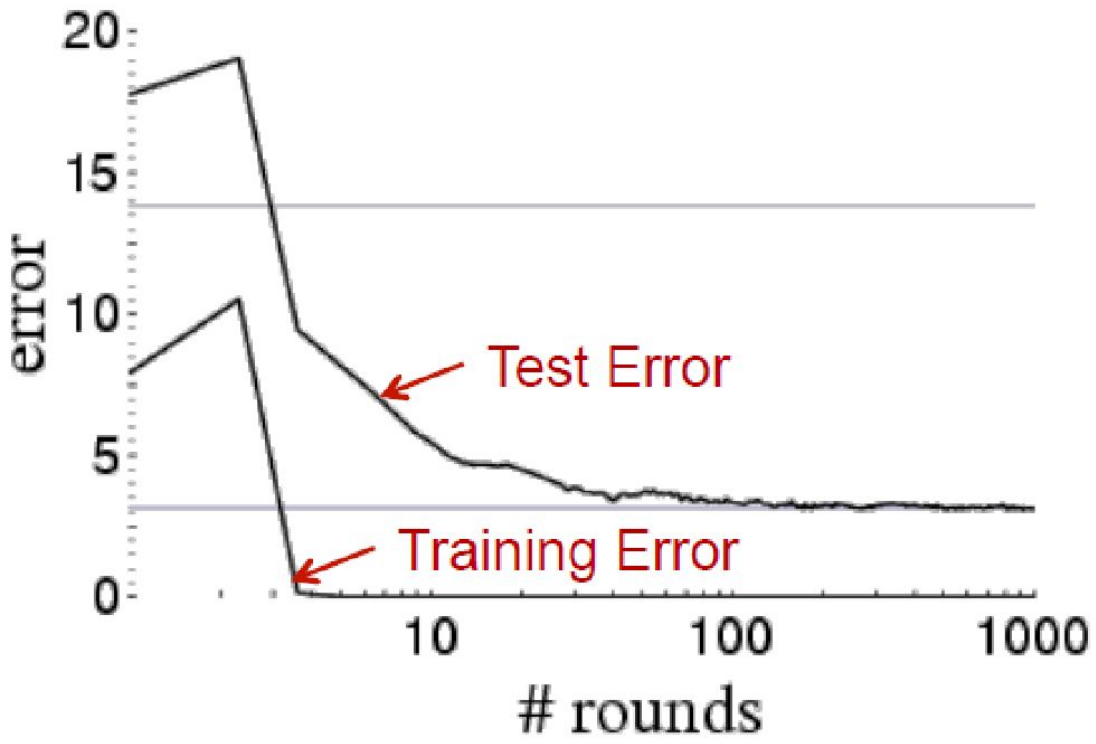
- Training error is bounded by

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

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

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

# Digit recognition




- Test error still decreases even after training error reaches zero.
- This shows boosting is robust to overfitting?

# Comparison between LR and Boosting

- Logistic Regression assumes

$$P(y_i = 1|x_i) = \frac{1}{1+\exp(-y_i f(x_i))}, f(x) = \sum_d w_d x_d + w_0$$

- Maximizing the data log likelihood

$$\max_w -\log(1 + \exp(-y_i f(x_i)))$$

Or

$$\min_w \log(1 + \exp(-y_i f(x_i)))$$

# Comparison between LR and Boosting

- Logistic Regression

$$\min_{w} \log(1 + \exp(-y_i f(x_i))), f(x) = \sum_{d} w_d x_d + w_0$$

- Boosting

$$\min_{\mathrm{h}} \frac{1}{m} \sum_i \exp(-y_i f(x_i)) = \prod_t Z_t, f(x) = \sum_t \alpha_t h_t(x)$$

- Comparing $\alpha_t \leftrightarrow w_d$ and $h_t \leftrightarrow x_d$, LR and Boosting becomes comparable.
  - For LR all $w_d$ are jointly learned, but for Boosting, $h_t$ are learned sequentially.
  - Note that LR is linear classifier, but Boosting is not.

# Attention

- A common mistake: Even if you choose a linear classifier for $h_t$, the final classifier is not linear in Boosting.
  - A sign function should be taken for $h_t(x) = sign(w^T x)$, otherwise a linear function of a set of linear functions

$$f(x) = \sum_t \alpha_t h_t(x)$$

  is still linear which we do not desire.

# Gradient Boosting

A top performer in many Kaggle competitions!

Typically uses decision trees as the basic classifier.

Fits h to the residual to correct errors of predecessors

Optimization is achieved using gradient descent

# Summary

- We have learned to use a set of weak classifiers to build a strong classifier, which
  - Can reduce the training error arbitrarily close to zero.
  - Even if the weak classifier is only slightly better than random guess
- A particular Boosting algorithm: Adaboost
- Compare the Logistic Regression and Boosting
  - Linear vs. Nonlinear
  - Joint optimization vs. iterative optimization
- Overview of gradient boosting