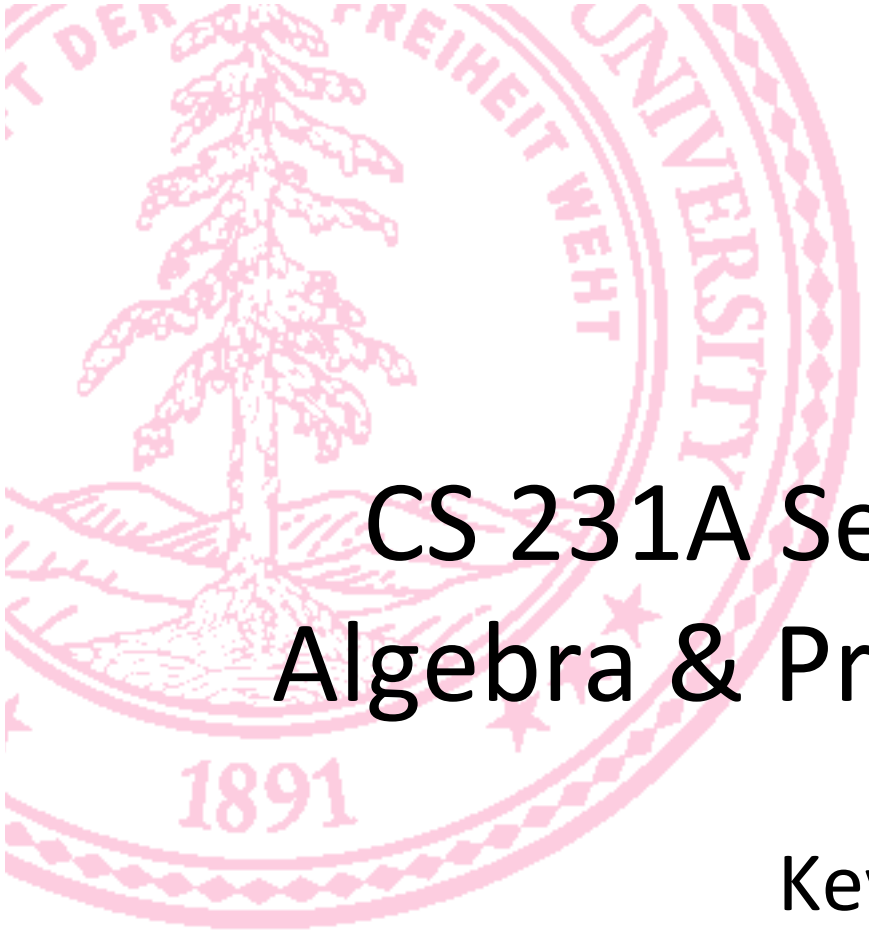


CS 231A Section 1: Linear Algebra & Probability Review

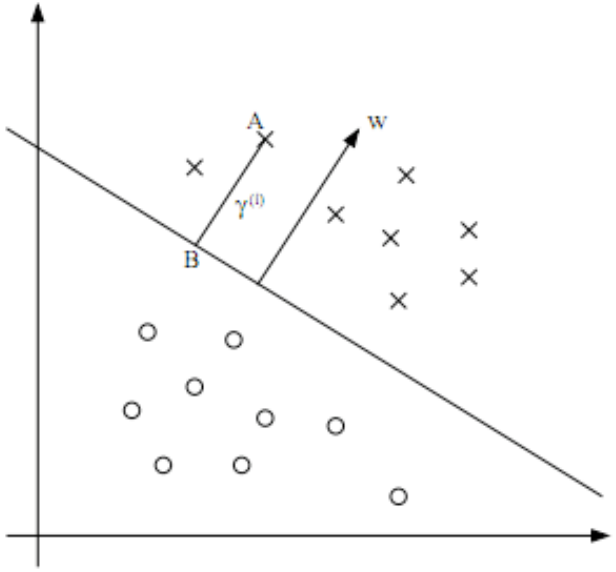
Kevin Tang

Topics

- Support Vector Machines
- Boosting
  - Viola-Jones face detector
- Linear Algebra Review
  - Notation
  - Operations & Properties
  - Matrix Calculus
- Probability
  - Axioms
  - Basic Properties
  - Bayes Theorem, Chain Rule

Support Vector Machines (SVM)

- Wish to perform binary classification, i.e. find a linear classifier
- Given data $\{x^{(1)}, \ldots, x^{(n)}\}$ and labels $\{y^{(1)}, \ldots, y^{(n)}\}$ where $y^{(i)} \in \{-1, 1\}$
- When data is linearly separable we can solve the optimization problem to find our linear classifier

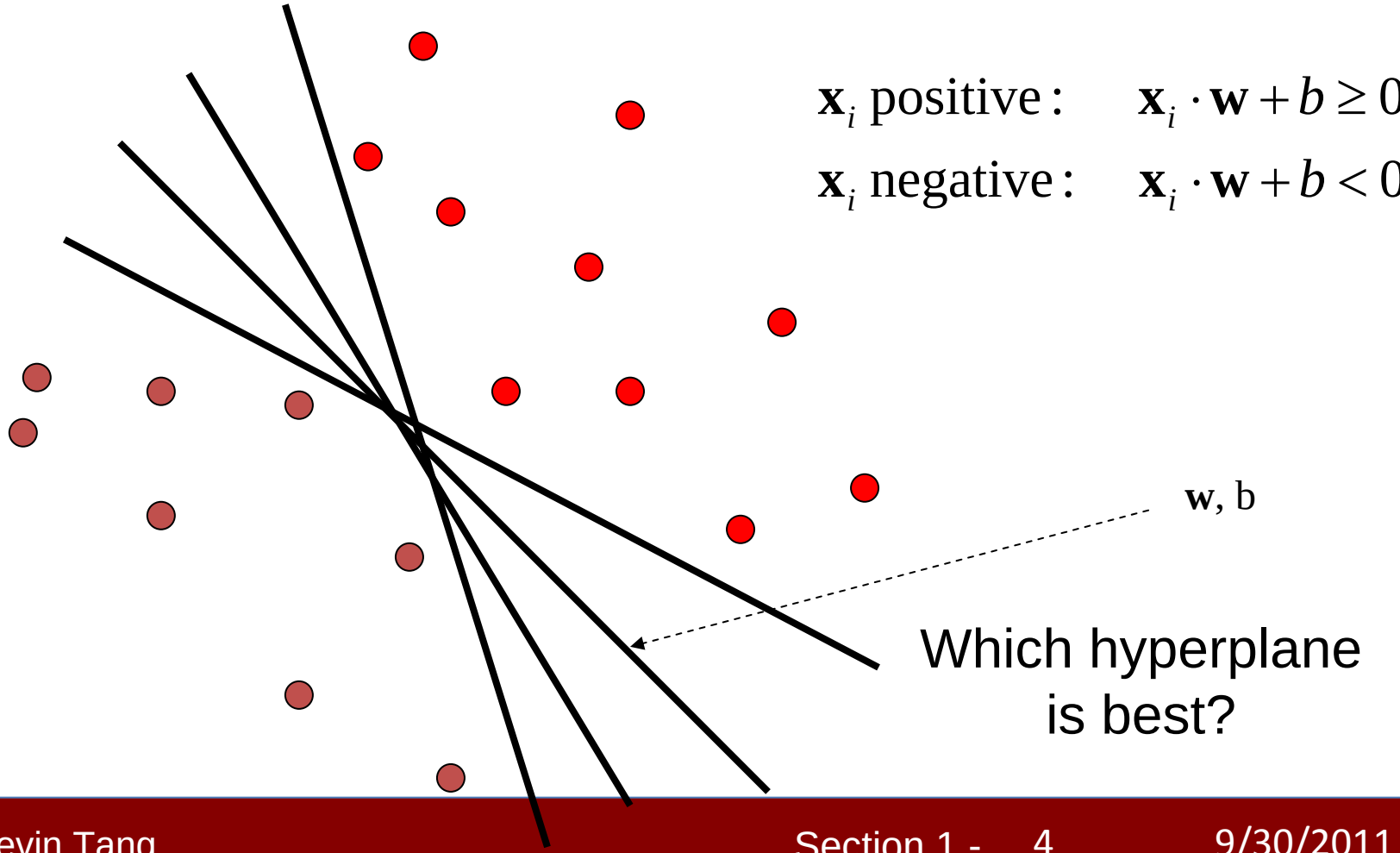


$$\min. \ \frac{1}{2}\|w\|^2$$

$$\text{s.t. } y^{(i)}(w^T x^{(i)} + b) \geq 1, \ i = 1, \ldots, n$$

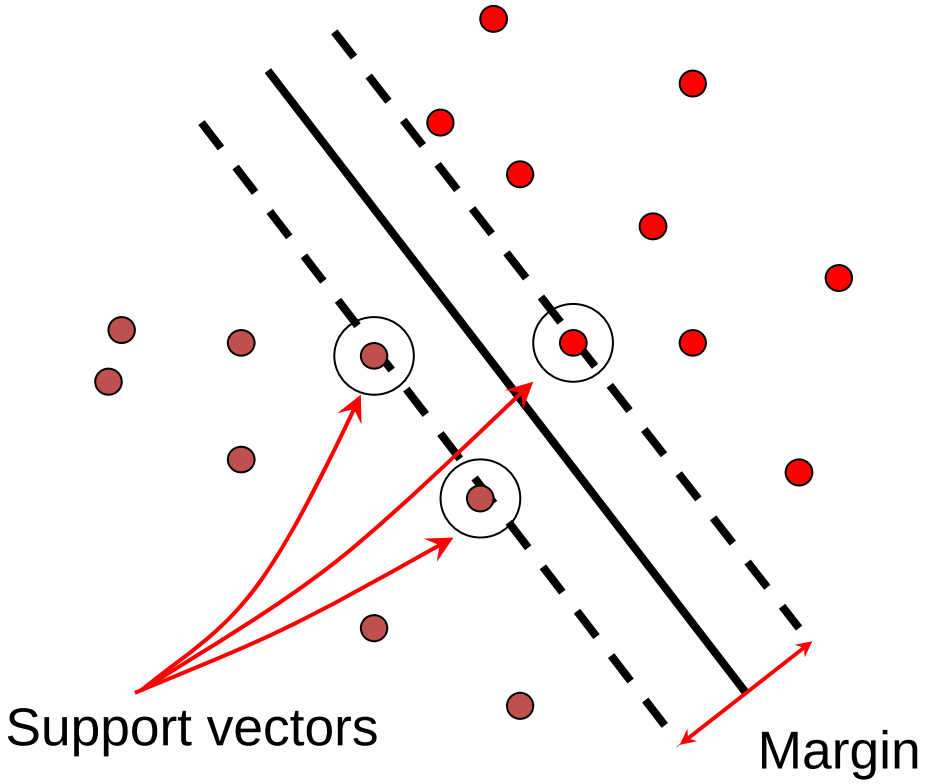
Linear classifiers

- Find linear function (*hyperplane*) to separate positive and negative examples



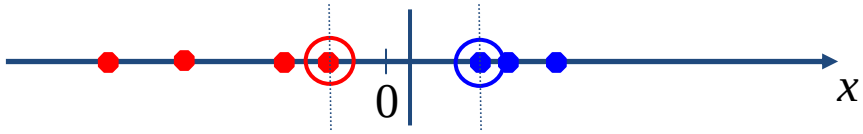
Support vector machines

- Find hyperplane that maximizes the *margin* between the positive and negative examples



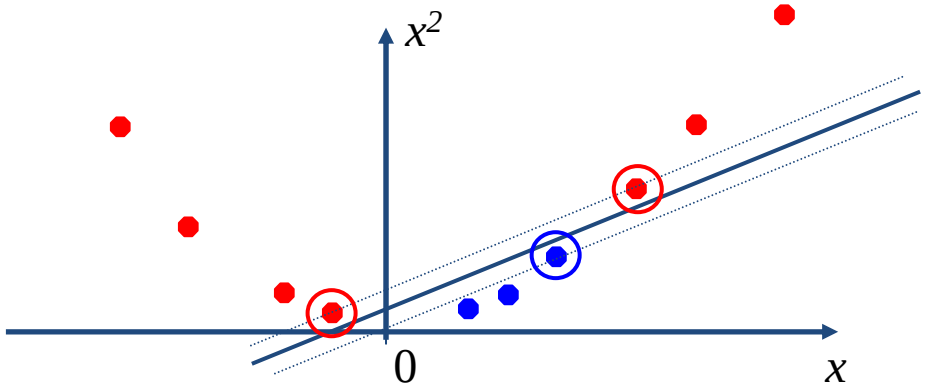
Nonlinear SVMs

- Datasets that are linearly separable work out great:



- But what if the dataset is just too hard?

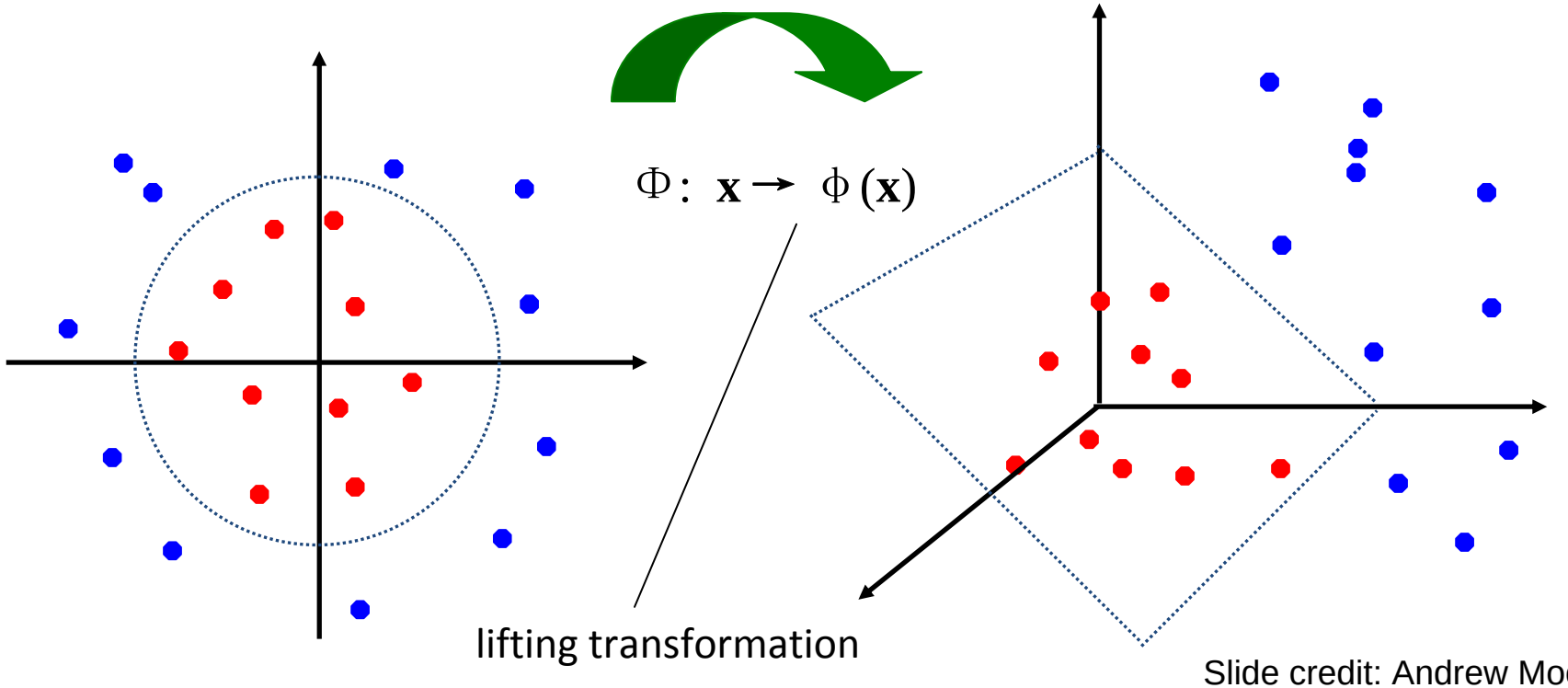
- We can map it to a higher-dimensional space:



Slide credit: Andrew Moore

Nonlinear SVMs

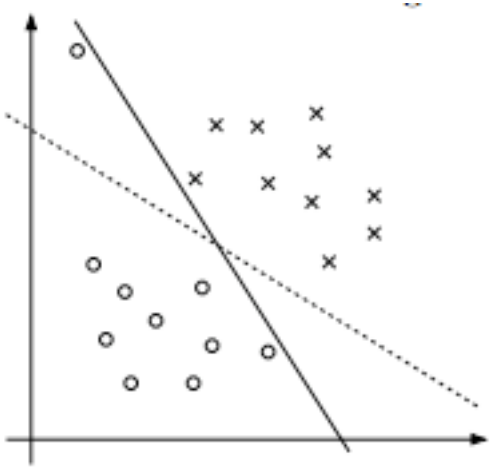
- General idea: the original input space can always be mapped to some higher-dimensional feature space where the training set is separable:



Slide credit: Andrew Moore

SVM – l_1 regularization

- What if data is not linearly separable?
- Can use regularization to solve this problem
- We solve a new optimization problem and "tune" our regularization parameter C



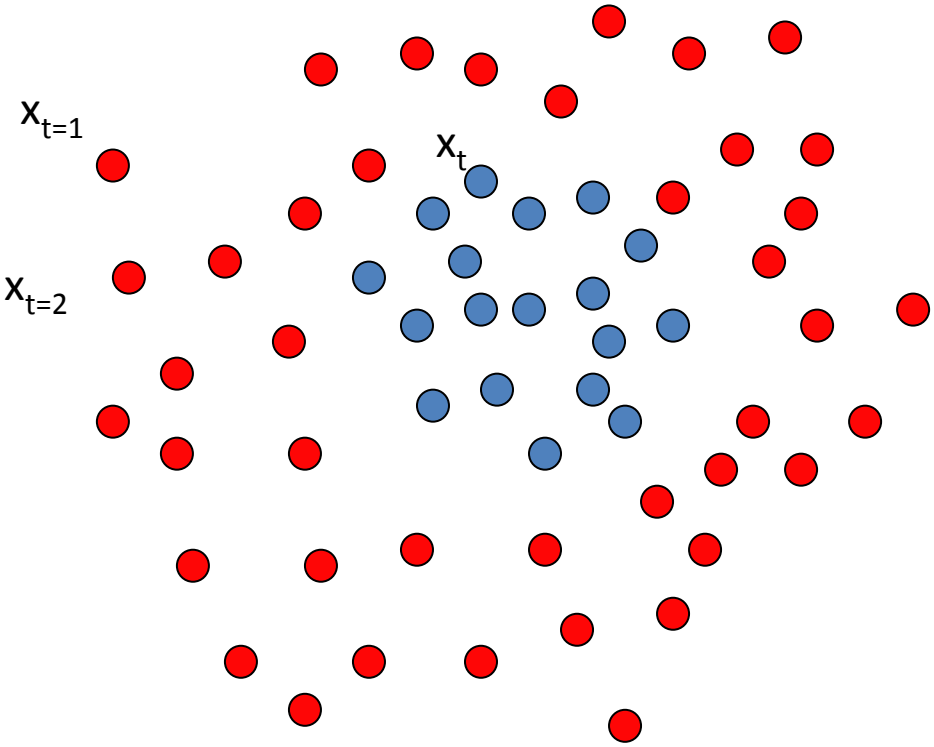
$$\begin{aligned}
&\text{min. } \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i \\
&\text{s.t. } y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i,\ i = 1, \ldots, n \\
&\qquad \xi_i \geq 0,\ i = 1, \ldots, n
\end{aligned}$$

Solving the SVM

- There are many different packages for solving SVM's
- In PS 0 we have you use the liblinear package. This is an efficient implementation but can only use a linear kernel
  http://www.csie.ntu.edu.tw/~cjlin/liblinear/
- If you wish to have more flexibility with your choice of kernel you can use the LibSVM package
  http://www.csie.ntu.edu.tw/~cjlin/libsvm/

Boosting

Y. Freund and R. Schapire, A short introduction to boosting, *Journal of Japanese Society for Artificial Intelligence*, 14(5):771-780, September, 1999.



Each data point has

a class label:

$$y_t = \begin{cases} +1 \text{ (●)} \\ -1 \text{ (●)} \end{cases}$$

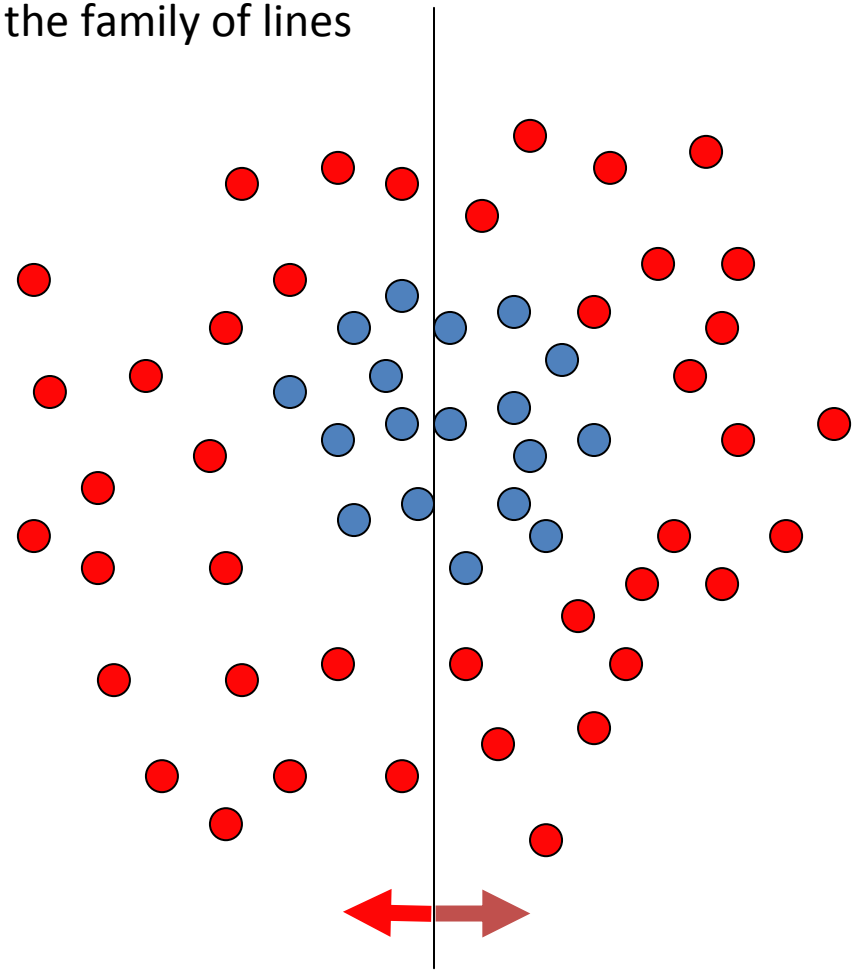
and a weight:

$$w_t = 1$$

- It is a sequential procedure:

Toy example

Weak learners from the family of lines



Each data point has

a class label:

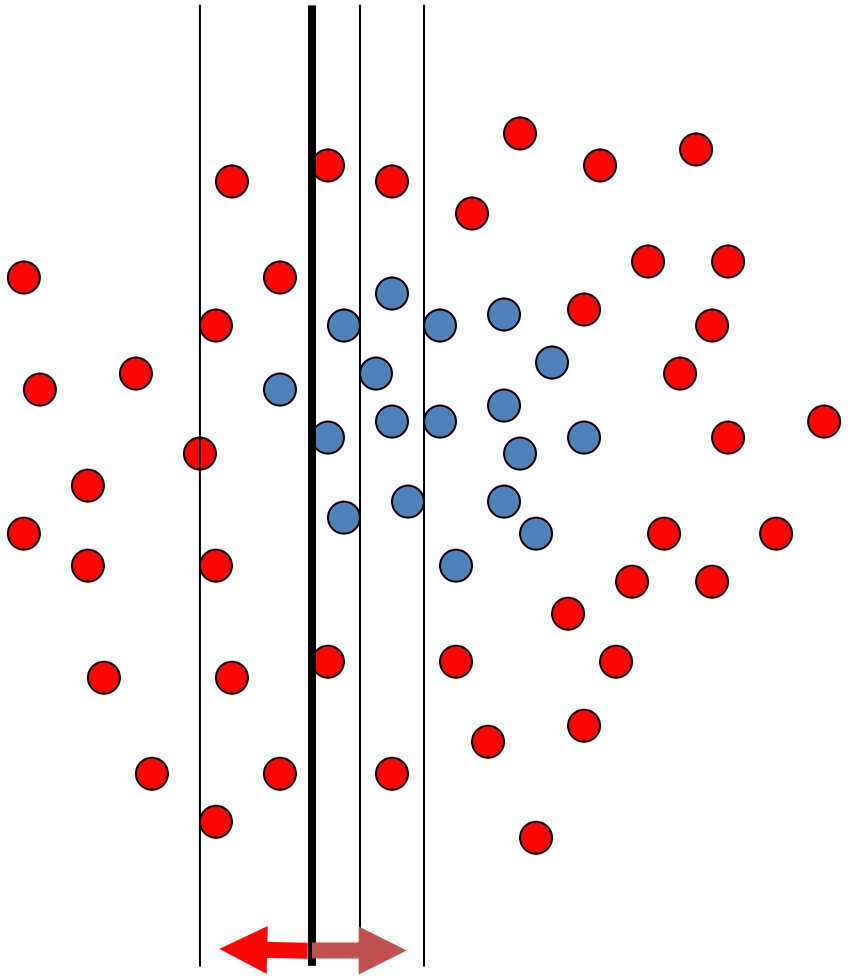
$$y_t = \begin{cases} +1 \text{ (●)} \\ -1 \text{ (●)} \end{cases}$$

and a weight:

$$w_t = 1$$

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

Toy example



Each data point has a class label:

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

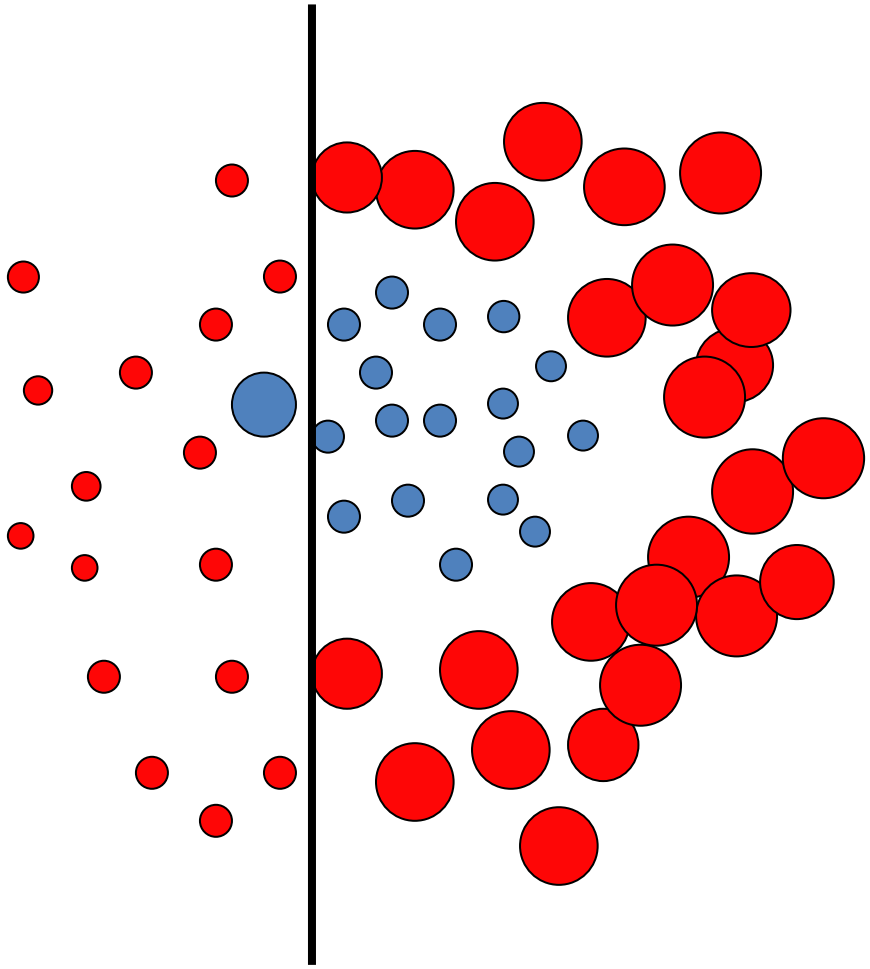
and a weight:

$$w_t = 1$$

This one seems to be the best

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

Toy example



Each data point has

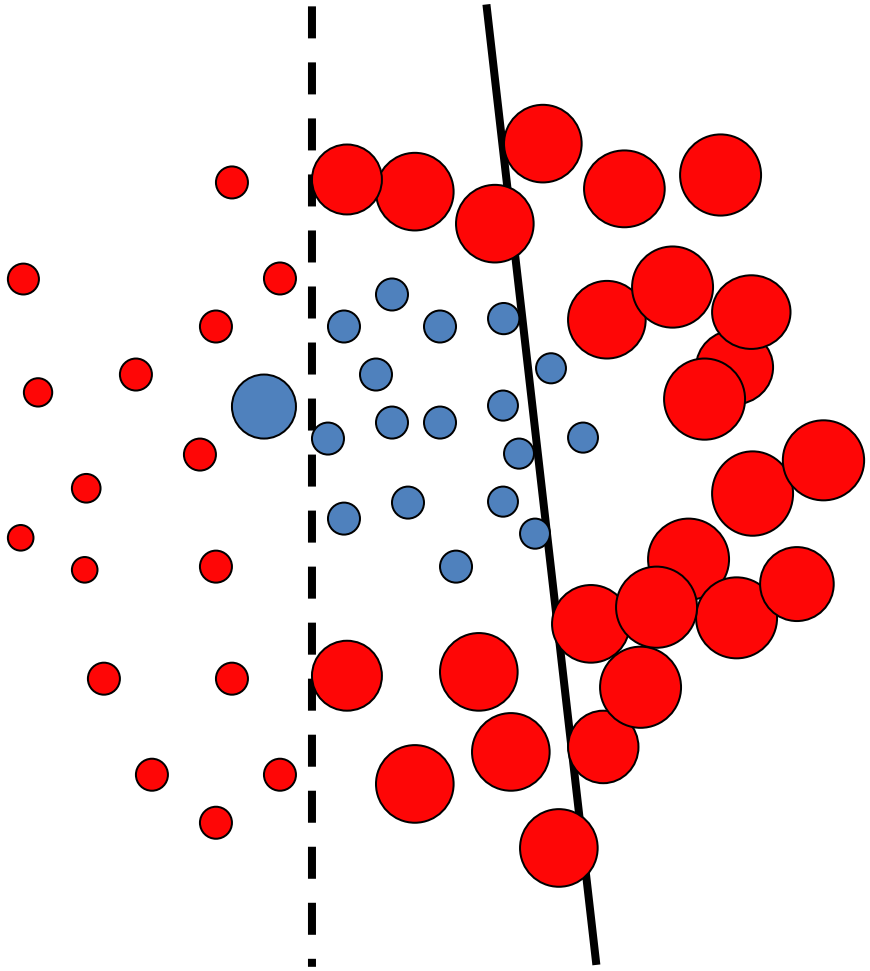
a class label:

$$y_t = \begin{cases} +1 \text{ (●)} \\ -1 \text{ (●)} \end{cases}$$

We update the weights:

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

Toy example



Each data point has

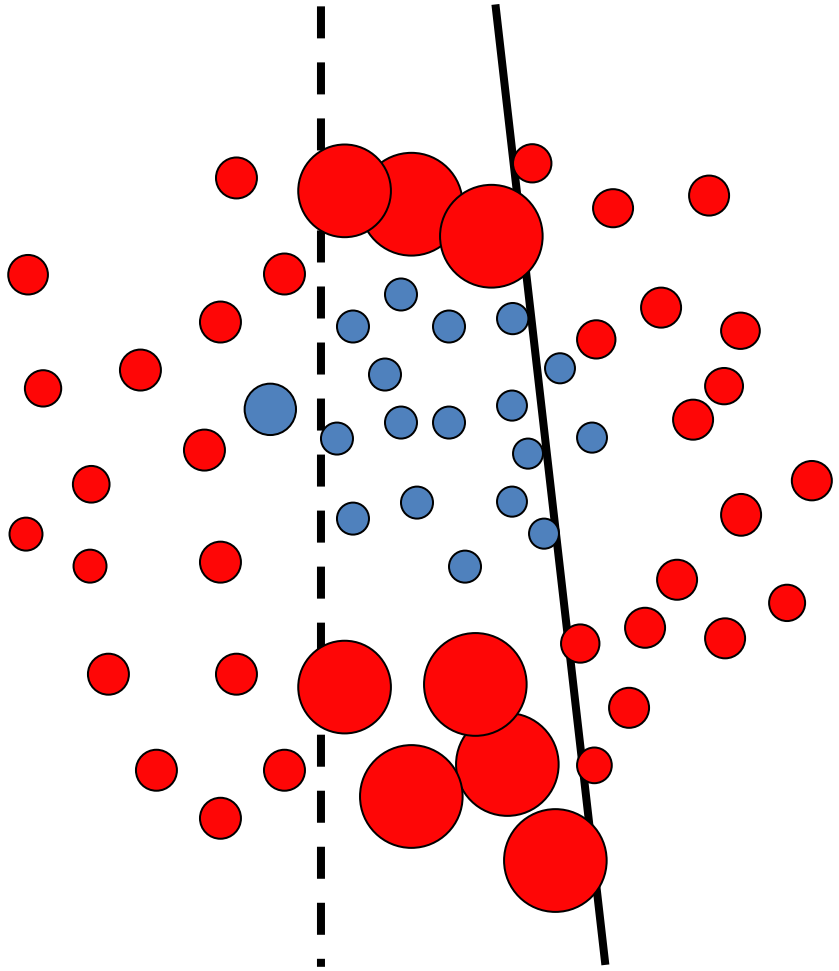
a class label:

$$y_t = \begin{cases} +1 \text{ (●)} \\ -1 \text{ (●)} \end{cases}$$

We update the weights:

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

Toy example



Each data point has

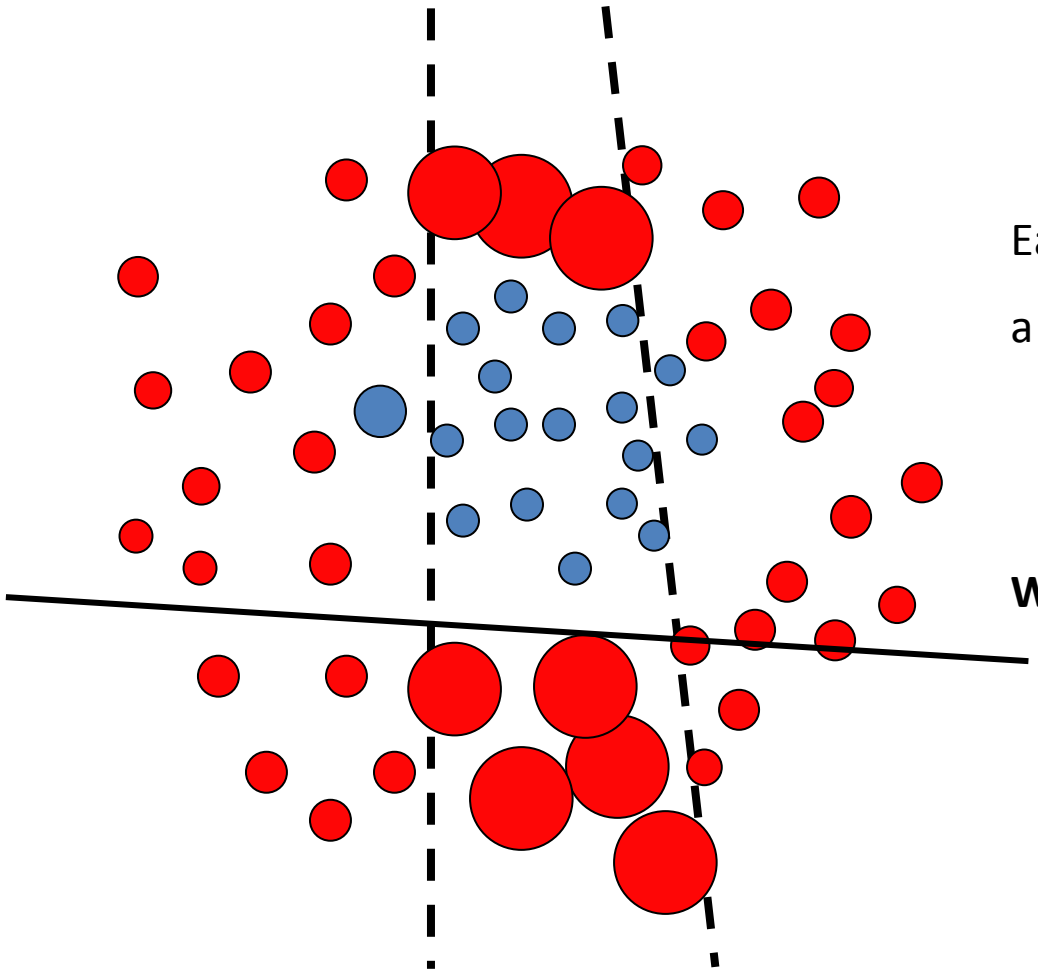
a class label:

$$y_t = \begin{cases} +1 \text{ (●)} \\ -1 \text{ (●)} \end{cases}$$

We update the weights:

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

Toy example



Each data point has

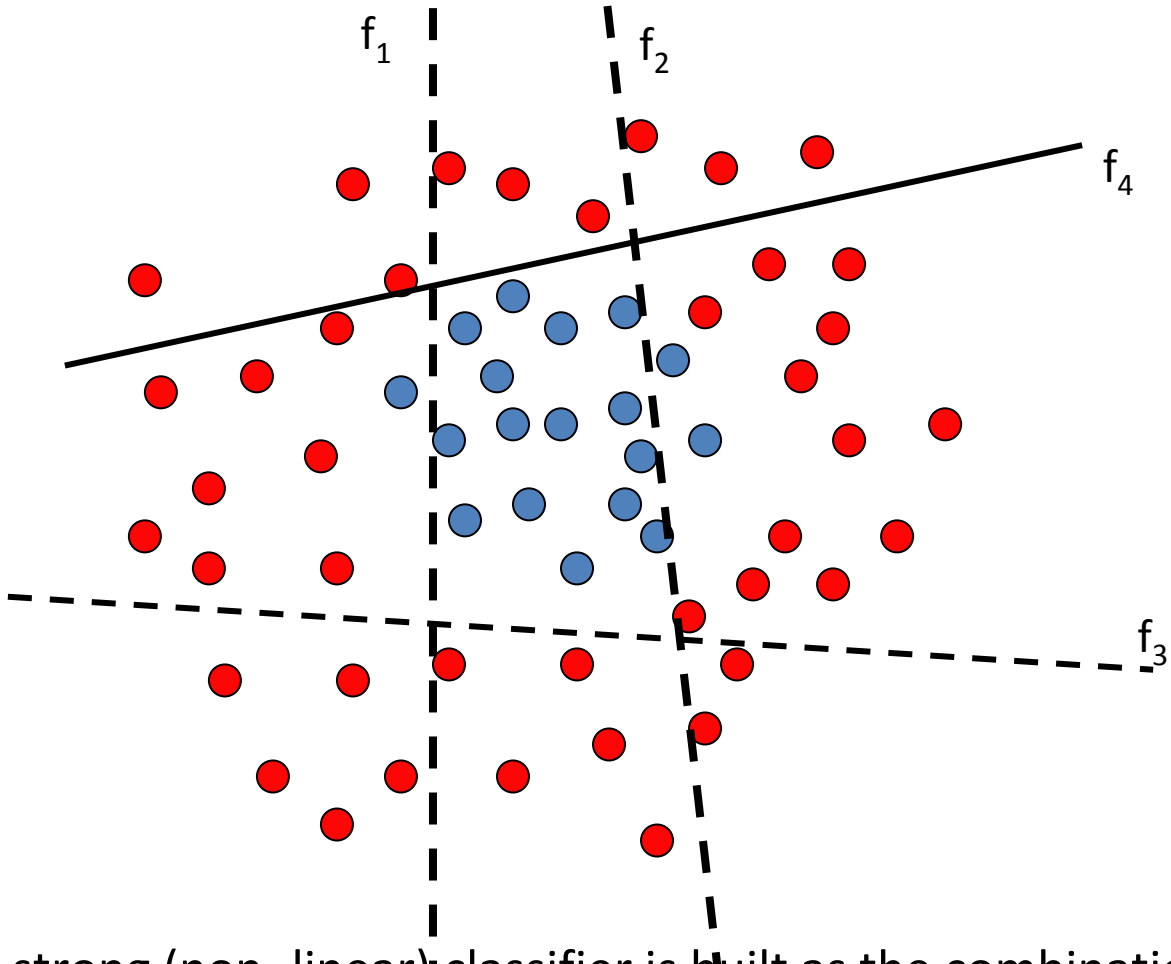
a class label:

$$y_t = \begin{cases} +1 \text{ (●)} \\ -1 \text{ (●)} \end{cases}$$

We update the weights:

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

Toy example



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

Boosting

- Defines a classifier using an additive model:

$$h(x) = \alpha_1 h_1(x) + \alpha_2 h_2(x) + \alpha_3 h_3(x) + \cdots$$

Strong classifier

Features vector

Weight

Weak classifier

Boosting

- Defines a classifier using an additive model:

$$h(x) = \alpha_1 h_1(x) + \alpha_2 h_2(x) + \alpha_3 h_3(x) + \cdots$$

Strong classifier

Features vector

Weight

Weak classifier

- We need to define a family of weak classifiers

$h_k(x)$ form a family of weak classifiers

Why boosting?

- A simple algorithm for learning robust classifiers
  - Freund & Shapire, 1995
  - Friedman, Hastie, Tibshhirani, 1998
- Provides efficient algorithm for sparse visual feature selection
  - *Tieu & Viola, 2000*
  - *Viola & Jones, 2003*
- Easy to implement, not requires external optimization tools.

Boosting - mathematics

- Weak learners

value of rectangle feature

$$h_j(x) = \begin{cases} 1 & \text{if } f_j(x) > \theta_j \\ 0 & \text{otherwise} \end{cases}$$

threshold

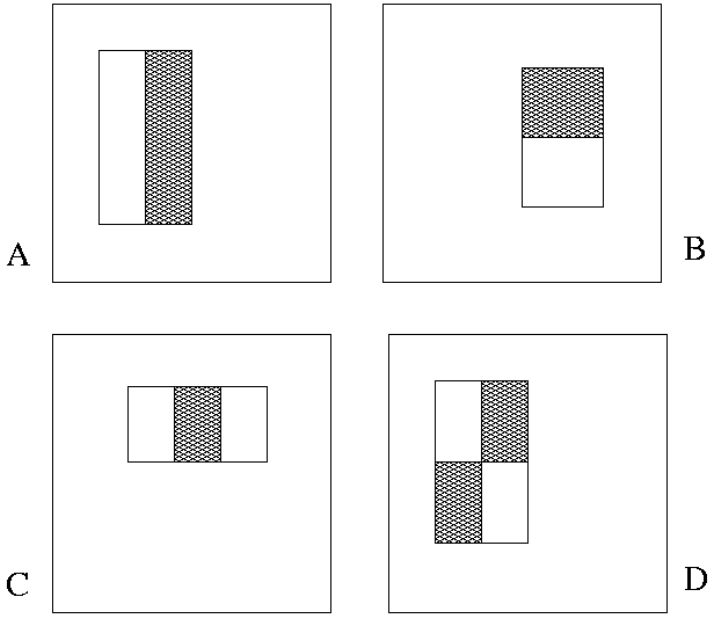
- Final strong classifier

$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

Weak classifier

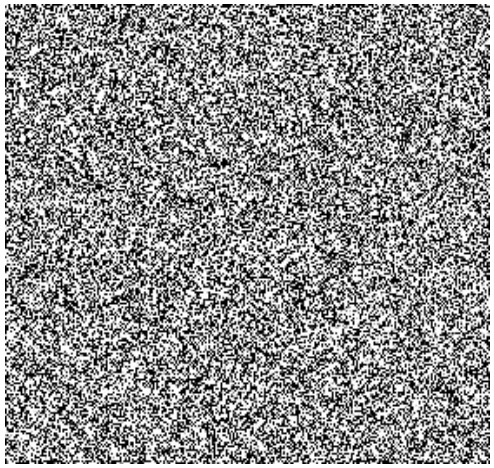
- 4 kind of Rectangle filters
- *Value =*

$\sum$ *(pixels in white area) –*
$\sum$ *(pixels in black area)*

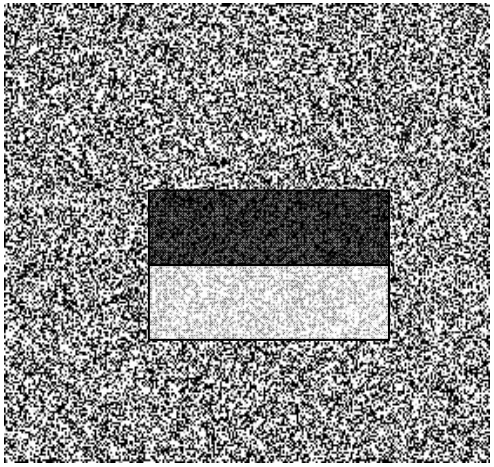


Credit slide: S. Lazebnik

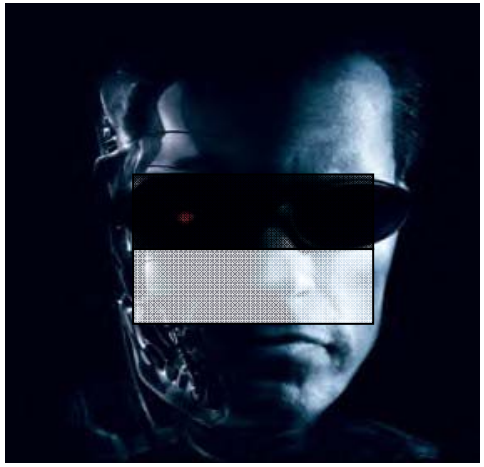
Weak classifier



Source



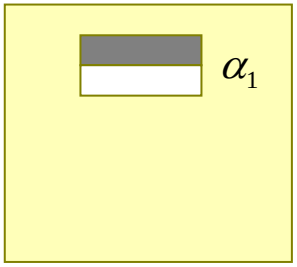
Result

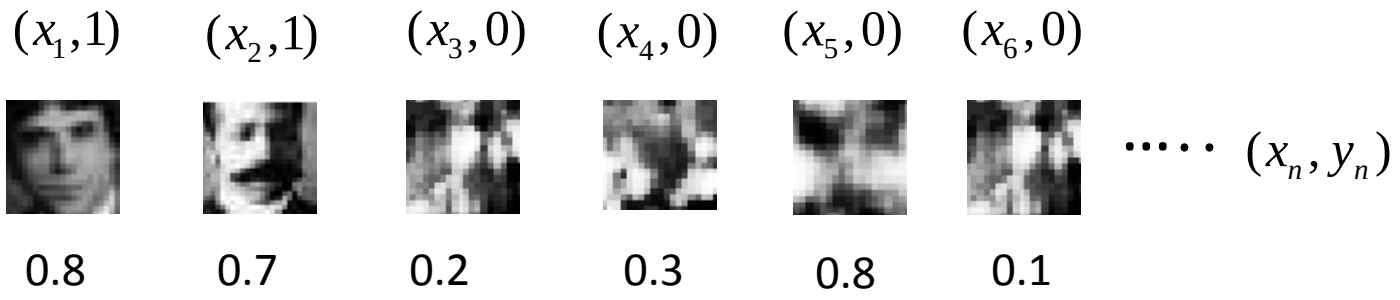


Credit slide: S. Lazebnik

Viola & Jones algorithm

1. Evaluate each rectangle filter on each example



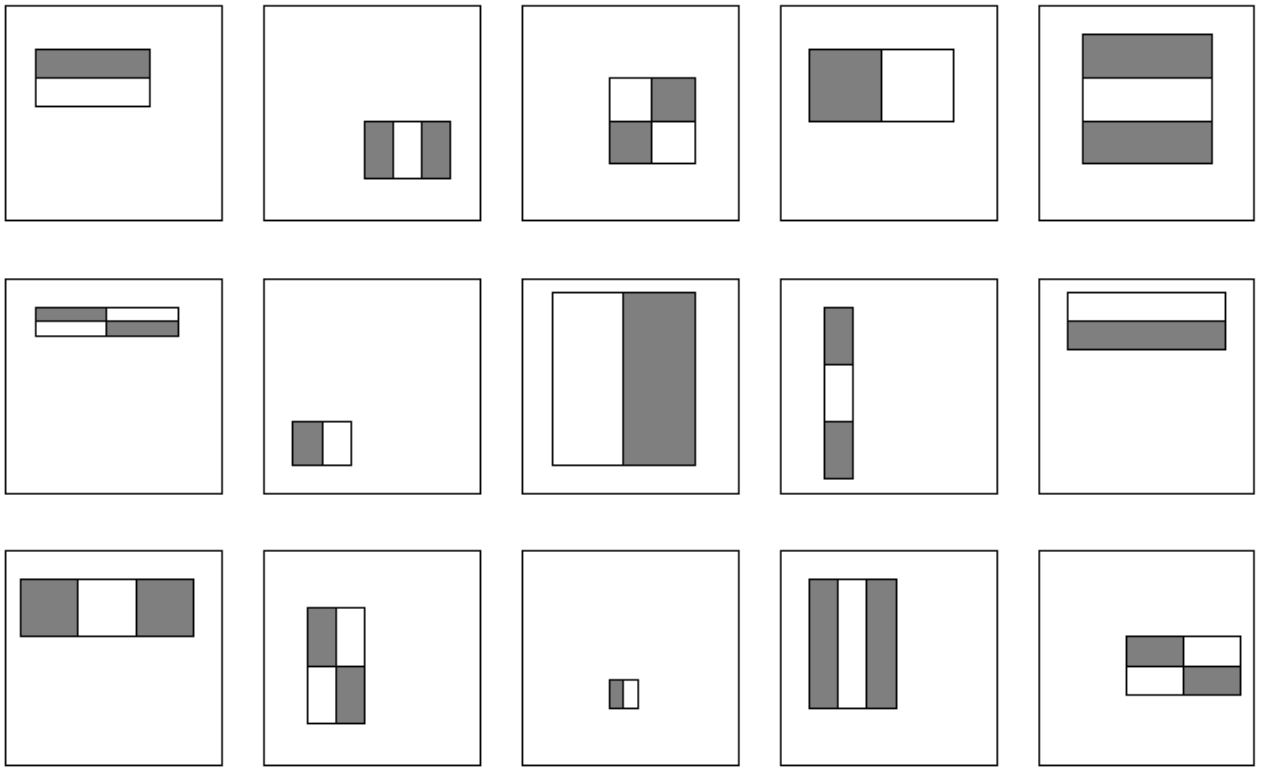


Weak classifier $h_j(x) = \begin{cases} 1 & \text{if } f_j(x) > \theta_j \\ 0 & \text{otherwise} \end{cases}$ ← threshold

P. Viola and M. Jones. *Rapid object detection using a boosted cascade of simple features*. CVPR 2001.

Viola & Jones algorithm

- For a 24x24 detection region,



P. Viola and M. Jones. *Rapid object detection using a boosted cascade of simple features*. CVPR 2001.

Viola & Jones algorithm

2. Select best filter/threshold combination

a. Normalize the weights $w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$

$$h_j(x) = \begin{cases} 1 & \text{if } f_j(x) > \theta_j \\ 0 & \text{otherwise} \end{cases}$$

b. For each feature, j $\varepsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right|$

c. Choose the classifier, h_t with the lowest error ε_t

3. Reweight examples

$$w_{t+1,i} = w_{t,i} \beta_t^{1-\left|h_t(x_i)-y_i\right|} \qquad \beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$$

P. Viola and M. Jones. *Rapid object detection using a boosted cascade of simple features*. CVPR 2001.

Viola & Jones algorithm

4. The final strong classifier is

$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

$$\alpha_t = \log \frac{1}{\beta_t}$$

The final hypothesis is a weighted linear combination of the T hypotheses where the weights are inversely proportional to the training errors

P. Viola and M. Jones. *Rapid object detection using a boosted cascade of simple features*. CVPR 2001.

Boosting for face detection

- For each round of boosting:
  1. Evaluate each rectangle filter on each example
  2. Select best filter/threshold combination
  3. Reweight examples

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

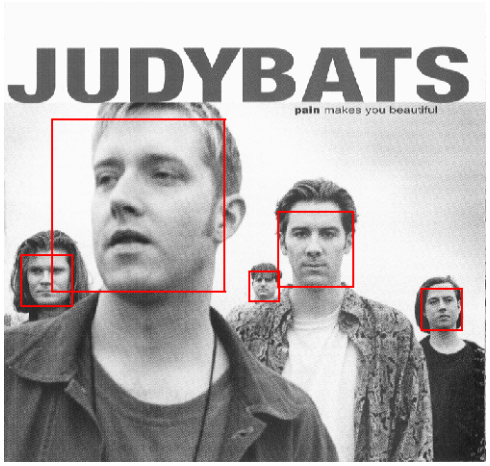
P. Viola and M. Jones. *Rapid object detection using a boosted cascade of simple features*. CVPR 2001.

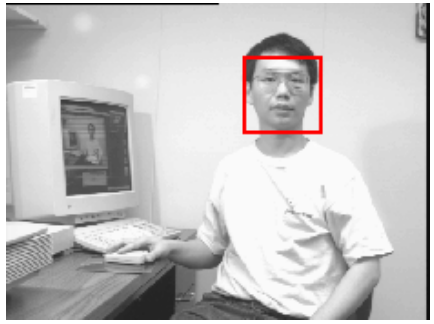
System performance

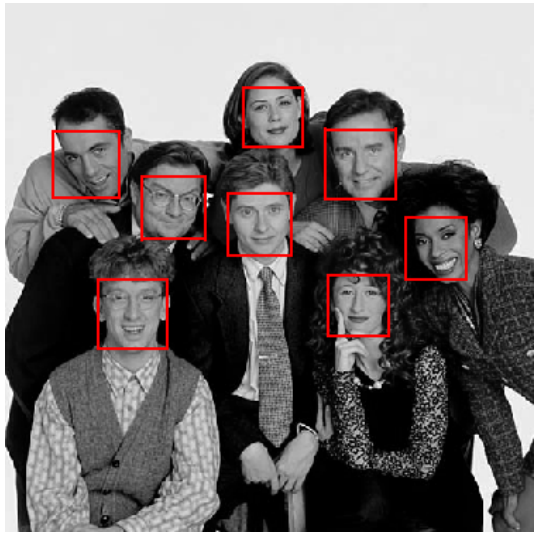
- Training time: “weeks” on 466 MHz Sun workstation
- 38 layers, total of 6061 features
- Average of 10 features evaluated per window on test set
- “On a 700 Mhz Pentium III processor, the face detector can process a 384 by 288 pixel image in about .067 seconds”
  - 15 Hz
  - 15 times faster than previous detector of comparable accuracy (Rowley et al., 1998)

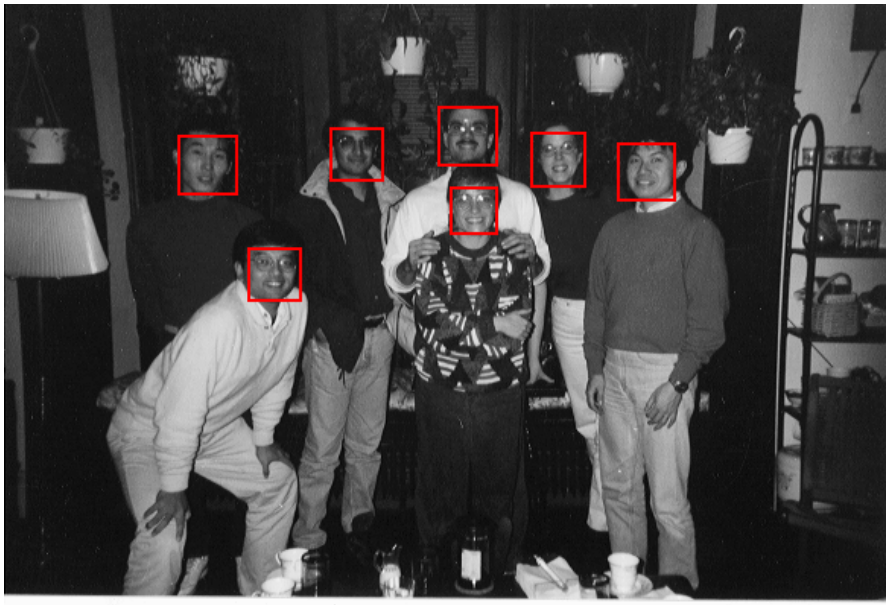
P. Viola and M. Jones. *Rapid object detection using a boosted cascade of simple features*. CVPR 2001.

Output of Face Detector on Test Images









P. Viola and M. Jones. *Rapid object detection using a boosted cascade of simple features*. CVPR 2001.

Linear Algebra in Computer Vision

- Representation
  - 3D points in the scene
  - 2D points in the image (Images are matrices)
- Transformations
  - Mapping 2D to 2D
  - Mapping 3D to 2D

Notation

- We adopt the notation for a matrix

$$A \in \mathbb{R}^{m \times n}$$

  which is a real valued matrix with m rows, and n columns

- We adopt the notation for a column vector, and a row vector respectively

$$b \in \mathbb{R}^n$$

$$a^T \in \mathbb{R}^n$$

Notation

- To indicate the element in the ith row and jth column of a matrix we use

$$A_{ij} \text{ where } A \in \mathbb{R}^{m \times n}$$

- Similarly to indicate the ith entry in a vector we use

$$x_i \text{ where } x \in \mathbb{R}^n$$

Norms

- Intuitively the norm of a vector is the measure of its "length"
- The l_2 norm is defined as $\|x\|_2 = \|x\| = \sqrt{\sum_{i=1}^{n} x_i^2}$

  in this class we will use the l_2 norm unless otherwise noted. Thus we drop the 2 subscript on the norm for convenience.
- Note that

$$\|x\|^2 = x^T x$$

Linear Independence and Rank

- A set of vectors is linearly independent if no vector in the set can be represented as a linear combination of the remaining vectors in the set
- The rank of a matrix is the maximal number of linearly independent column or rows of a matrix
- $A \in \mathbb{R}^{m \times n}, \ \text{rank}(A) \leq \min(m, n)$
  $\text{if } \text{rank}(A) = \min(m, n), \ A \text{ is said to be full rank}$
- $\text{rank}(A) = \text{rank}(A^T)$
- $\text{rank}(AB) \leq \min(\text{rank}(A), \text{rank}(B))$
- $\text{rank}(A + B) \leq \text{rank}(A) + \text{rank}(B)$

Range and Nullspace

- The range of a matrix is the span of the columns of the matrix, denoted by the set

$$\mathcal{R}(A) = \{y \mid y = Ax\}$$

- The nullspace of a matrix, is the set of vectors that when multiplied by the matrix result in 0, given by the set

$$\mathcal{N}(A) = \{x \mid Ax = 0\}$$

Eigenvalues and Eigenvectors

- Given a matrix, $\lambda \in \mathbb{C}$ and $x \in \mathbb{C}^n$ are said to be an eigenvalue and the corresponding eigenvector of the matrix if

$$Ax = \lambda x, \;\; x \neq 0$$

- We can solve for the eigenvalues by solving for the roots of the polynomial generated by

$$|(\lambda I - A)| = 0$$

Eigenvalue Properties

- $|A| = \prod_{i=1}^{n} \lambda_i$
- The rank of a matrix is equal to the number of its non-zero eigenvalues
- Eigenvalues of a diagonal matrix, are simply the diagonal entries
- A matrix is said to be diagonalizable if we can write

$$A = X \Lambda X^{-1}$$

Eigenvalues & Eigenvectors of Symmetric Matrices

- Eigenvalues of symmetric matrices are real
- Eigenvectors of symmetric matrices are orthonormal
- Consider the optimization problem involving the symmetric matrix, A

$$\max. \ \frac{x^T A x}{x^T x}$$

the maximizing x is the eigenvector corresponding to the largest eigenvalue

Generalized Eigenvalues

- Generalized Eigenvalue problem

$$Av = \lambda Bv$$

- Generalized eigenvalues must satisfy

$$|A - \lambda B| = 0$$

- This reduces to the original eigenvalue problem when B^{-1} exists

$$B^{-1}Av = \lambda v$$

- Generalized eigenvalues are used in Fisherfaces

Singular Value Decomposition (SVD)

- The SVD of matrix is given by

$$A = U\Sigma V^T$$

- Where u_i are the columns of U and called the left singular vectors
  $\sum$ is a diagonal matrix whose values are σ_i and called the singular values
  v_i are the columns of V, and are called the right singular vectors

SVD

- If the matrix A has rank r, then A has r non-zero singular values
- $\{u_1, \ldots, u_r\}$ are an orthonormal basis for $\text{range}(A)$
- $\{v_{r+1}, \ldots, v_n\}$ are an orthonormal basis for $\text{null}(A)$
- Singular values of A are the square root of the non-zero eigenvalues of $A^T A$ or AA^T

Matlab

- [V,D] = eig(A)
  The eigenvectors of A are the columns of V. D is a diagonal matrix whose entries are the eigenvalues of A.
- [V,D] = eig(A,B)
  The generalized eigenvectors are the columns of V. D is a diagonal matrix whose entries of the generalized eigenvalues.
- [U,S,V] = svd(X)
  The columns of U are the left singular vectors of X. S is a diagonal matrix whose entries are the singular values of X. The columns of V are the right singular vectors of X. Recall X = U*S*V';

Matrix Calculus -- Gradient

- Let $f : \mathbb{R}^{m \times n} \rightarrow \mathbb{R}$
  then the gradient is given by

$$(\nabla_A f(A))_{ij} = \frac{\partial f(A)}{\partial A_{ij}}$$

- $\nabla_A f(A)$ is always the same size as A, thus if we just have a vector the gradient is simply

$$\nabla_x f(x) = \begin{bmatrix} \frac{\partial f(x)}{x_1} \\ \frac{\partial f(x)}{x_2} \\ \vdots \\ \frac{\partial f(x)}{x_n} \end{bmatrix}$$

Gradients

- From partial derivatives

$$\nabla_x(f(x) + g(x)) = \nabla_x f(x) + \nabla_x g(x)$$

$$\nabla_x(tf(x)) = t\nabla_x f(x)$$

- Some common gradients

$$\nabla_x b^T x = b$$

$$\nabla_x x^T A x = 2Ax, \text{ if } A \text{ symmetric}$$

Probability in Computer Vision

- Foundation for algorithms to solve
  - Tracking problems
  - Human activity recognition
  - Object recognition
  - Segmentation

Probability Axioms

- Sample space: The set of all the outcomes of a random experiment. Denoted by Ω
- Event space: A set whose elements are subsets of Ω . The event space is denoted by $\mathcal{F}$. For example

$$\text{if } A \in \mathcal{F} \text{ then } A \subseteq \Omega$$

- Probability measure: A function $P : \mathcal{F} \rightarrow \mathbb{R}$ that satisfies
  - $P(A) \geq 0, \text{ for all } A \in \mathcal{F}$
  - $P(\Omega) = 1$
  - for $A_1, A_2, \ldots, A_n$ disjoint $P(\cup_i A_i) = \sum P(A_i)$

Basic Properties

- if $A \subseteq B \Rightarrow P(A) \leq P(B)$
- $P(A \cap B) \leq \min(P(A), P(B))$
- $P(\Omega \setminus A) = 1 - P(A)$
- (Union Bound) $P(A \cup B) \leq P(A) + P(B)$
- if $A_1, A_2, \ldots, A_k$ are disjoint events such that $\cup_{i=1}^{k} A_i = \Omega$ then,

$$\sum_{i=1}^{k} P(A_i) = 1$$

Conditional Probability

- $P(A \mid B) \triangleq \frac{P(A \cap B)}{P(B)}$

- Two events are independent if

$$P(A \mid B) = P(A)$$

- Conditional Independence

$$P(A \cap B \mid C) = P(A \mid C)P(B \mid C)$$

Product Rule

- From the definition of conditional probability we can write

$$P(A \cap B) = P(B \mid A)P(A)$$

- From the product rule we can derive the chain rule of probability

$$P(\cap_{i=1}^{k} A_i) = \prod_{i=1}^{k} P(A_i \mid \cap_{j=1}^{i-1} A_j)$$

Bayes Theorem

- $$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$