In summary

- C1={skin}
- C2={~skin}
- Given x=[R,G,B], is it skin or ~skin?
- Approach 1: nearest neighbors
  - How about high dimensional point x, e.g., a face image?
- Approach 2: compute p(C1|x) and p(C2|x)
  - P(C1|x) = P(x|C1)P(C1)/p(x)
  - P(C2|x) = P(x|C2)P(C2)/p(x)
    - How to compute P(x|C1) and P(x|C2)?
      - » Histogram
      - » How about high dimensional point x, e.g., a face image?

Modeling P(x|C) for high dimensional x

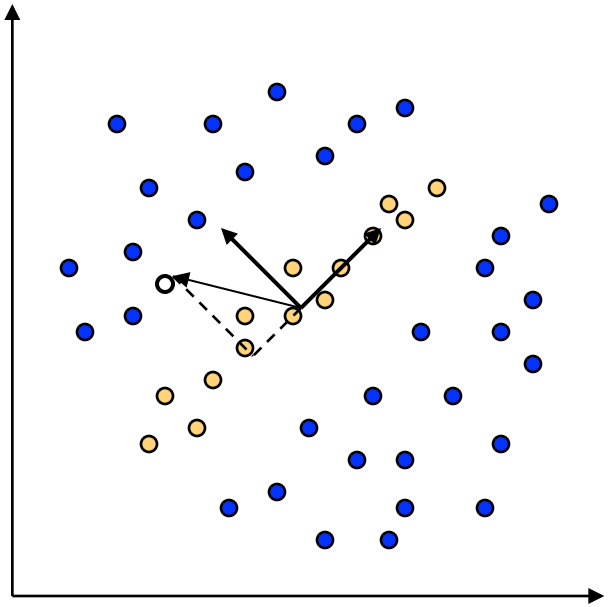
- Assuming P(x|C) is Gaussian
- 1-dimensional; $$p(x \mid C) = \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{1}{2\sigma^2}(x-\mu)^2)$$
  - how to estimate mean and variance?
- D-dimensional;

$$p(x \mid C) = (2\pi)^{-\frac{D}{2}} \mid \Sigma \mid^{-\frac{1}{2}} \exp(-\frac{1}{2}(x-\mu)^{\mathrm{T}} \Sigma^{-1}(x-\mu))$$

  - how to estimate mean and co-variance?

- What's wrong with testing p(x|C) given the estimated mean and covariance?
  - Degeneracy
  - Efficiency

Linear subspaces



convert **x** into $\mathbf{v}_1$, $\mathbf{v}_2$ coordinates

$$\mathbf{x} \rightarrow ((\mathbf{x} - \overline{x}) \cdot \mathbf{v_1}, (\mathbf{x} - \overline{x}) \cdot \mathbf{v_2})$$

What does the $\mathbf{v}_2$ coordinate measure?

- distance to line
- use it for classification—near 0 for orange pts

What does the $\mathbf{v}_1$ coordinate measure?

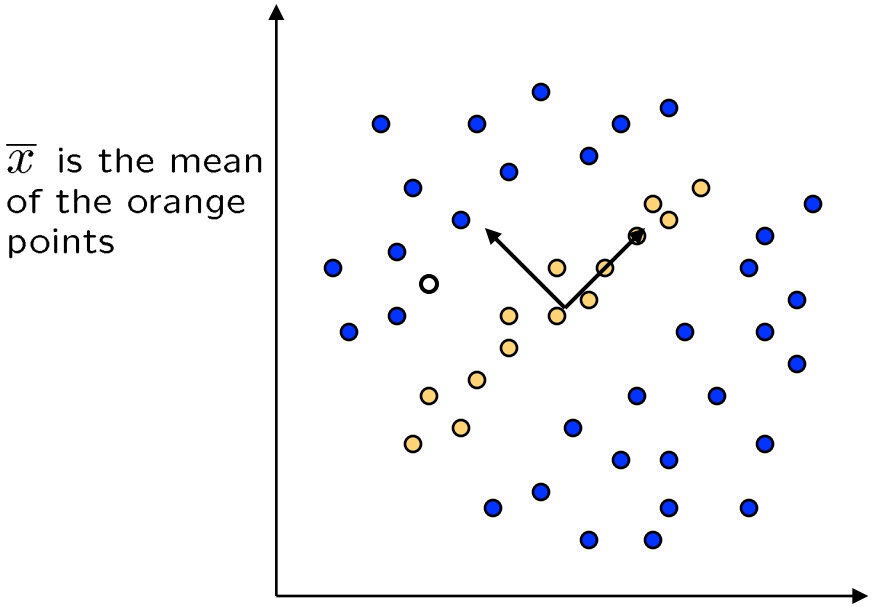
- position along line
- use it to specify which orange point it is

- Classification can be expensive
  - Must either search (e.g., nearest neighbors) or estimate high-dim PDF's

Suppose the data points are arranged as above

- Idea—fit a line, classifier measures distance to line

Dimensionality reduction



How to find $\mathbf{v}_1$ and $\mathbf{v}_2$?
- PCA

Dimensionality reduction

- We can represent the orange points with *only* their $\mathbf{v}_1$ coordinates
  - since $\mathbf{v}_2$ coordinates are all essentially 0
- This makes it much cheaper to store and compare points
- A bigger deal for higher dimensional problems

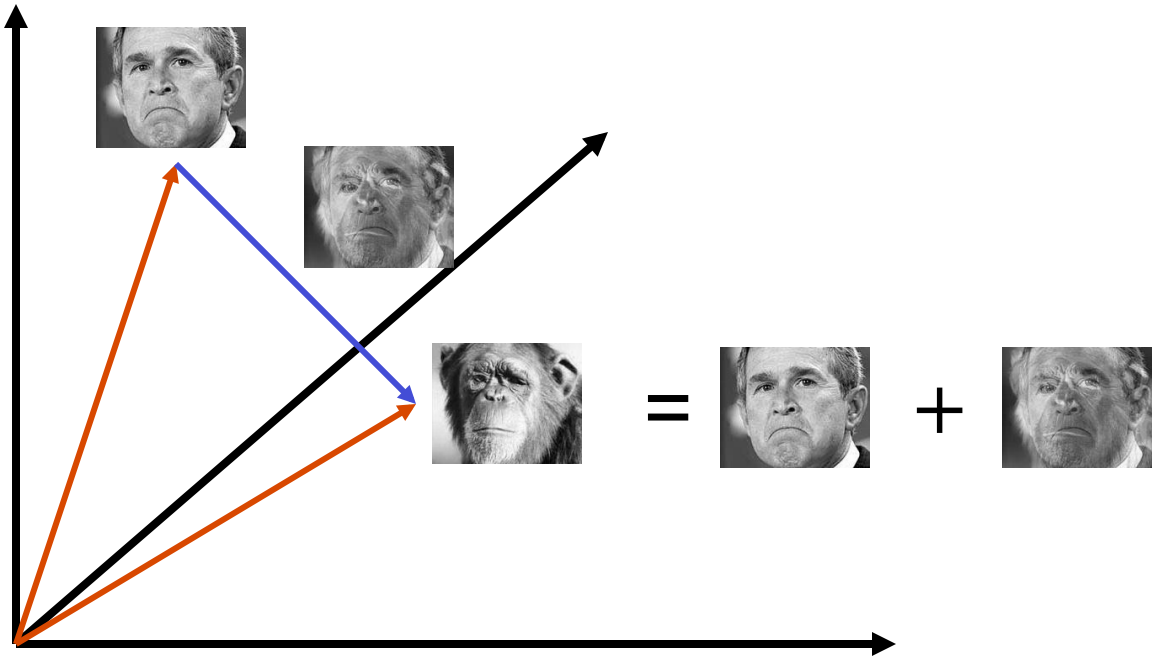
Principal component analysis

- Suppose each data point is N-dimensional
  - Same procedure applies:

$$\begin{aligned} var(\mathbf{v}) &= \sum_{\mathbf{x}} \|(\mathbf{x}-\overline{\mathbf{x}})^{\mathrm{T}} \cdot \mathbf{v}\| \\ &= \mathbf{v}^{\mathrm{T}}\mathbf{A}\mathbf{v} \quad \text{where } \mathbf{A} = \sum_{\mathbf{x}} (\mathbf{x}-\overline{\mathbf{x}})(\mathbf{x}-\overline{\mathbf{x}})^{\mathrm{T}} \end{aligned}$$

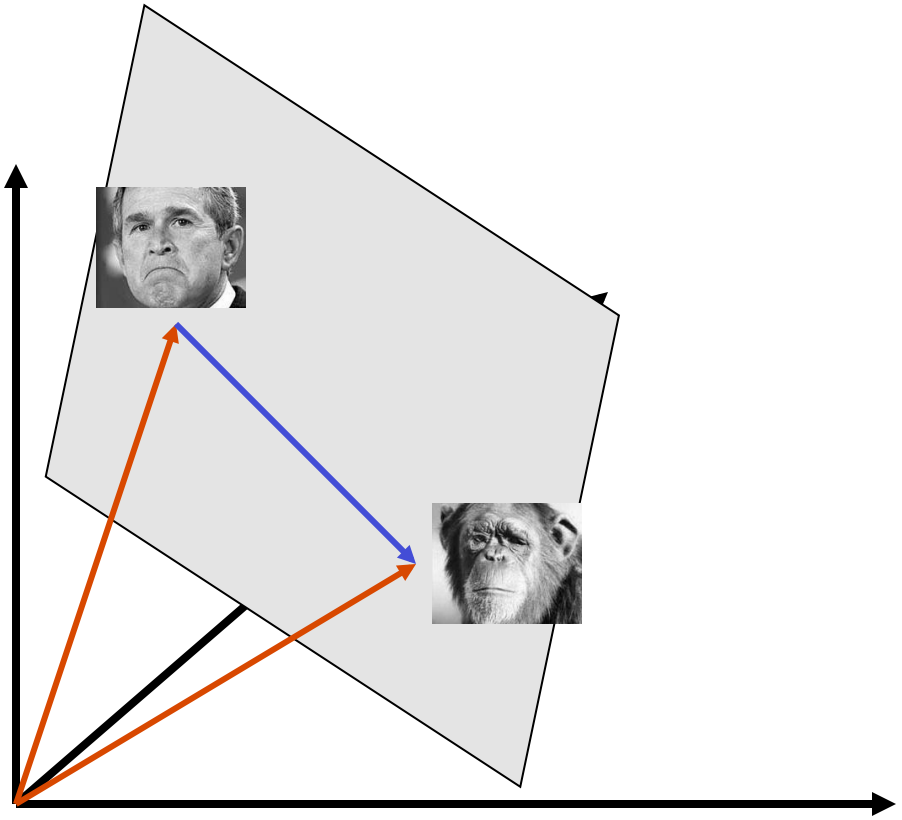
  - The eigenvectors of **A** define a new coordinate system
    - eigenvector with largest eigenvalue captures the most variation among training vectors **x**
    - eigenvector with smallest eigenvalue has least variation
  - We can compress the data by only using the top few eigenvectors
    - corresponds to choosing a "linear subspace"
      - » represent points on a line, plane, or "hyper-plane"
    - these eigenvectors are known as the ***principal components***

The space of faces



- An image is a point in a high dimensional space
  - An N x M image is a point in R^{NM}
  - We can define vectors in this space as we did in the 2D case

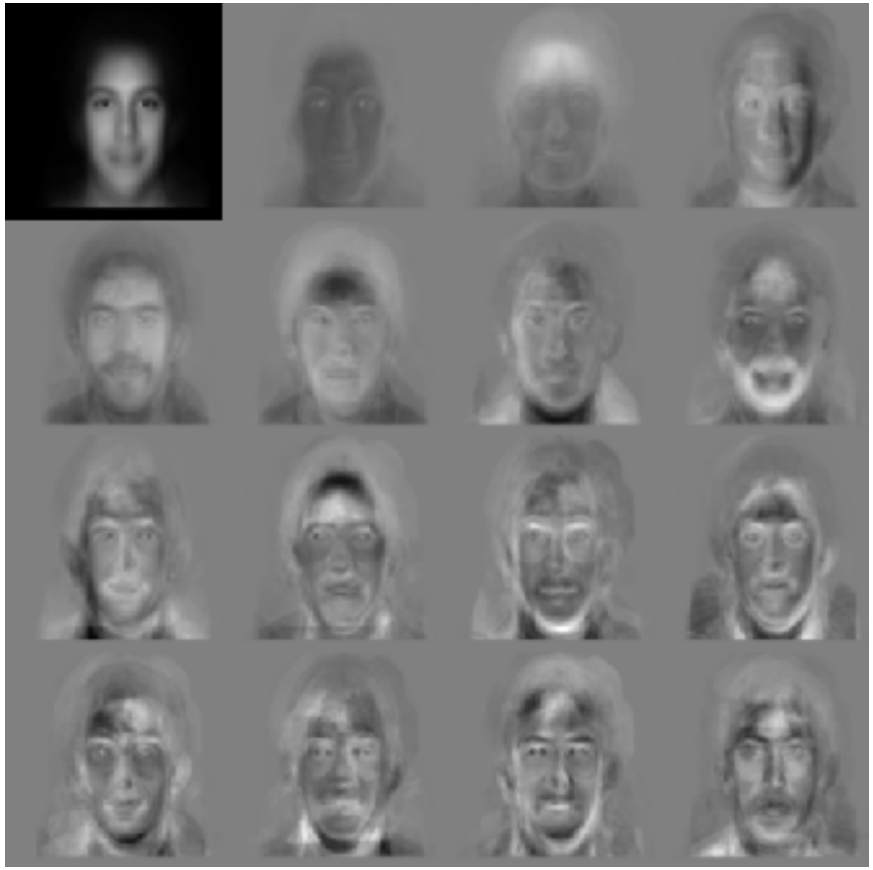
Dimensionality reduction



- The set of faces is a "subspace" of the set of images
  - Suppose it is K dimensional
  - We can find the best subspace using PCA
  - This is like fitting a "hyper-plane" to the set of faces
    - spanned by vectors $\mathbf{v_1}$, $\mathbf{v_2}$, ..., $\mathbf{v_K}$
    - any face $\mathbf{x} \approx \bar{\mathbf{x}} + a_1\mathbf{v_1} + a_2\mathbf{v_2} + \ldots + a_k\mathbf{v_k}$

Eigenfaces

- PCA extracts the eigenvectors of **A**
  - Gives a set of vectors $\mathbf{v}_1$, $\mathbf{v}_2$, $\mathbf{v}_3$, ...
  - Each one of these vectors is a direction in face space
    - what do these look like?



Projecting onto the eigenfaces

- The eigenfaces $\mathbf{v_1}$, ..., $\mathbf{v_K}$ span the space of faces
  - A face is converted to eigenface coordinates by

$$\mathbf{x} \rightarrow (\underbrace{(\mathbf{x}-\overline{\mathbf{x}})\cdot\mathbf{v_1}}_{a_1}, \underbrace{(\mathbf{x}-\overline{\mathbf{x}})\cdot\mathbf{v_2}}_{a_2}, \ldots, \underbrace{(\mathbf{x}-\overline{\mathbf{x}})\cdot\mathbf{v_K}}_{a_K})$$

$$\mathbf{x} \approx \overline{\mathbf{x}} + a_1\mathbf{v_1} + a_2\mathbf{v_2} + \ldots + a_K\mathbf{v_K}$$

Recognition with eigenfaces

- Algorithm
  1. Process the image database (set of images with labels)
     - Run PCA—compute eigenfaces
     - Calculate the K coefficients for each image
  2. Given a new image (to be recognized) **x**, calculate K coefficients

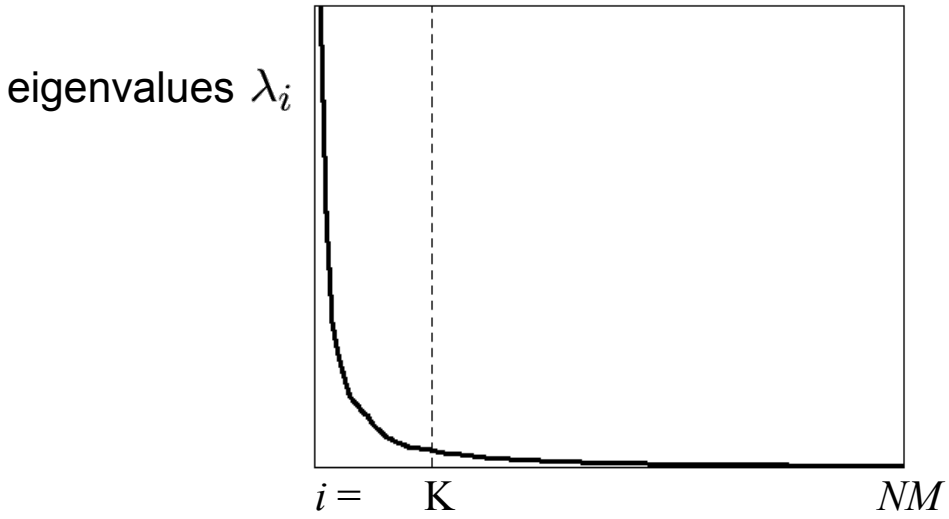
$$\mathbf{x} \rightarrow (a_1, a_2, \ldots, a_K)$$

  3. Detect if x is a face

$$\|\mathbf{x} - (\overline{\mathbf{x}} + a_1\mathbf{v_1} + a_2\mathbf{v_2} + \ldots + a_K\mathbf{v_K})\| < \text{threshold}$$

  4. If it is a face, who is it?
     - Find closest labeled face in database
       - nearest-neighbor in K-dimensional space

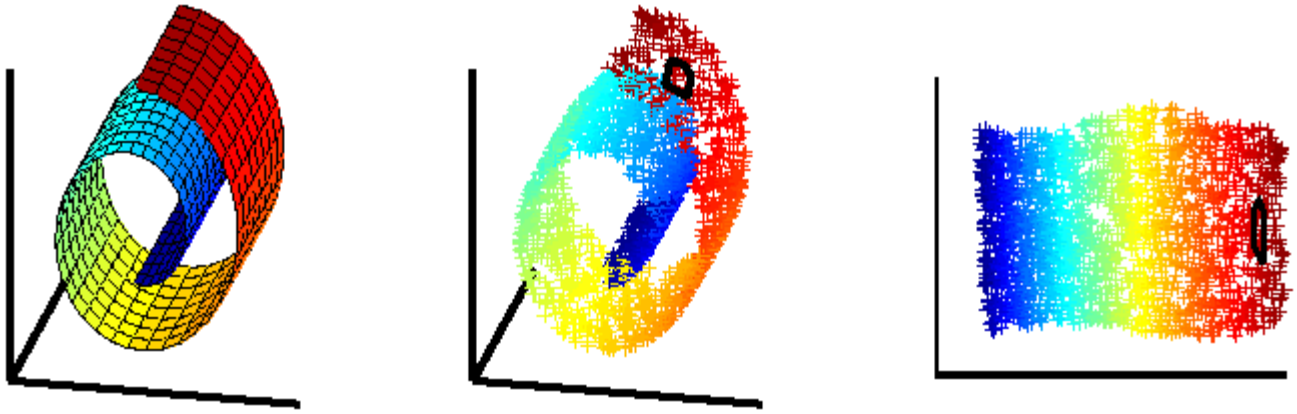
Choosing the dimension K



- How many eigenfaces to use?
- Look at the decay of the eigenvalues
  - the eigenvalue tells you the amount of variance “in the direction” of that eigenface
  - ignore eigenfaces with low variance

Issues: dimensionality reduction

- What if your space isn't *flat*?
  - PCA may not help



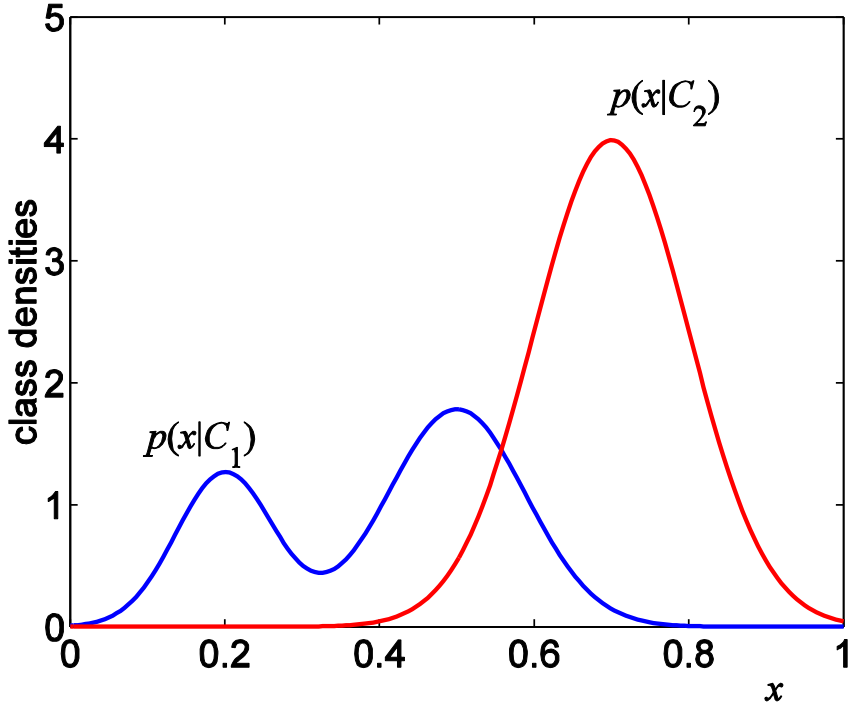
Nonlinear methods
LLE, MDS, etc.

Issues: data modeling

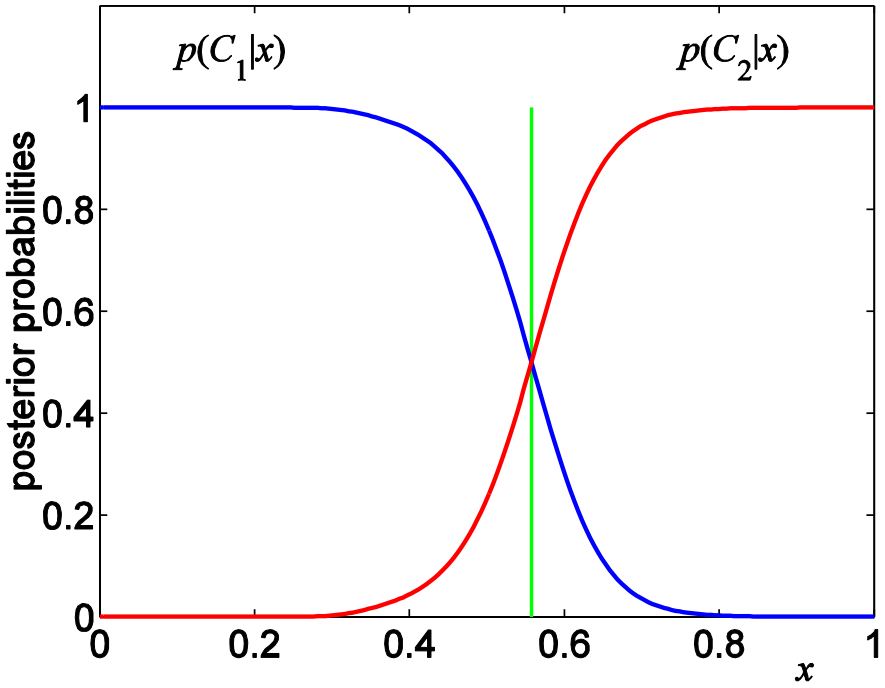
- Generative methods
  - model the "shape" of each class
    - histograms, PCA,
    - mixtures of Gaussians
    - ...

- Discriminative methods
  - model boundaries between classes
    - perceptrons, neural networks
    - support vector machines (SVM's)

Generative vs. Discriminative



Generative Approach
model individual classes, priors



Discriminative Approach
model posterior directly

from Chris Bishop

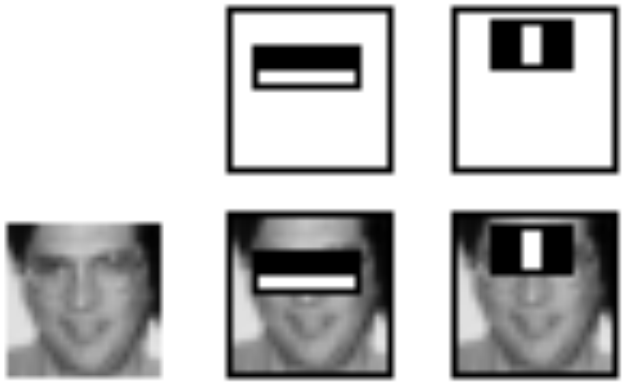
Issues: speed

- Case study: Viola Jones face detector
- Exploits three key strategies:
  - simple, super-efficient features
  - image pyramids
  - pruning (cascaded classifiers)

Viola/Jones: features

"Rectangle filters"

Differences between sums of pixels in adjacent rectangles



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

$60,000 \times 100 = 6,000,000$
Unique Features

$$Y(x) = \sum \alpha_t y_t(x)$$

Select 200 by Adaboost

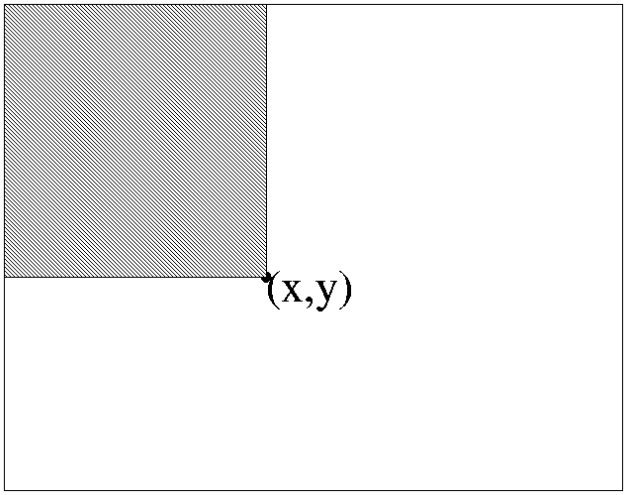
$$\text{Detection} = \begin{cases} \text{face}, & \text{if } Y(x) > 0 \\ \text{non-face}, & \text{otherwise} \end{cases}$$

Robust Realtime Face Dection, IJCV 2004, Viola and Jonce

Integral Image (aka. summed area table)

- Define the Integral Image

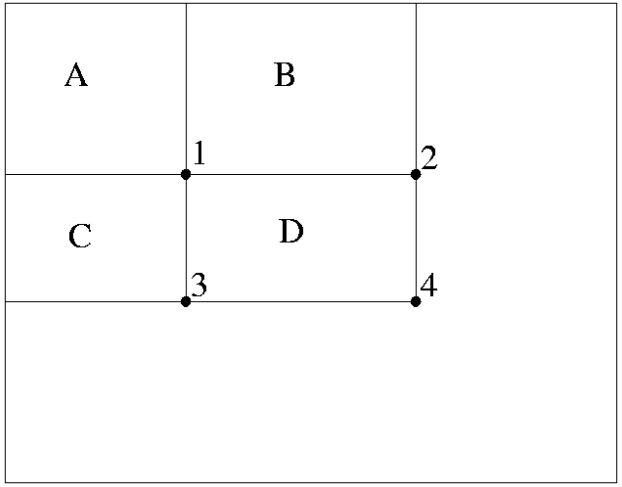
$$I'(x,y) = \sum_{\substack{x' \le x \\ y' \le y}} I(x',y')$$



- Any rectangular sum can be computed in constant time:

$$\begin{aligned} D &= 1 + 4 - (2 + 3) \\ &= A + (A + B + C + D) - (A + C + A + B) \\ &= D \end{aligned}$$

- Rectangle features can be computed as differences between rectangles



Feature selection (AdaBoost)

Given training data $\{x_n, t_n=+1/-1\}$, find $\{\alpha_t\}$ for $\{y_t(x)\}$ by Minimizing total error function:

$$E(Y = \sum_{t=1}^{M} \alpha_t y_t(x)) = \sum_{n=1}^{N} error(t_n Y(x_n))$$

Ideal function error(z) = z>0?0:1, hard to optimize.
Instead use error(z)=exp(-z) to make the optimization convex.

Define $f_m(x) = \frac{1}{2} \sum_{l=1}^{m} \alpha_l y_l(x)$

Basic idea:
first find $f_1(x)$ by minimizing $E(f_1)$
Then given $f_{m-1}(x)$, find $f_m(x)$ by searching for best α_m and $y_m(x)$