EE5907: Pattern Recognition
&
EE5026: Machine Learning for Data Analytics

Dr. WANG Si

Email: si.wang@nus.edu.sg

Office: E1a-04-01

Outlines

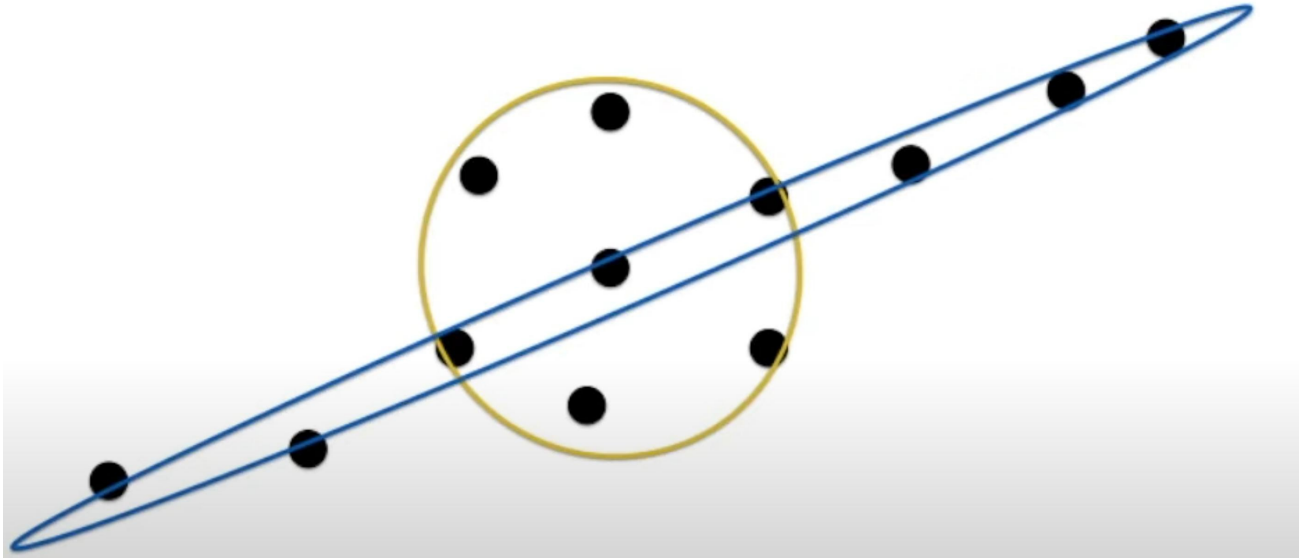
- Pattern Representation Learning
  - Unsupervised Representation Learning (week 7)
  - Supervised Representation Learning (week 8)
  - Unified Framework: Graph Embedding (week 8)
- Patter Recognition Models
  - Clustering (K-means, Agglomerative)(week 9)
  - Gaussian Mixture Model and Boosting (week 10)
  - Support Vector Machines (week 11)
- Deep Learning (week 12)
- Revision and Q&A (week 13)

Some slides are from Antonio Torralba

- We will discuss:
  - GMM (definition, assumption, optimization, etc.)
  - Boosting (AdaBoost & GentleBoost)
  - Object Detection using Boosting
- At the end of this lecture, you should be able to:
  - Know the rationales behind GMM and Boosting
  - Build GMM and Boosting
  - Understand strengths and limitations

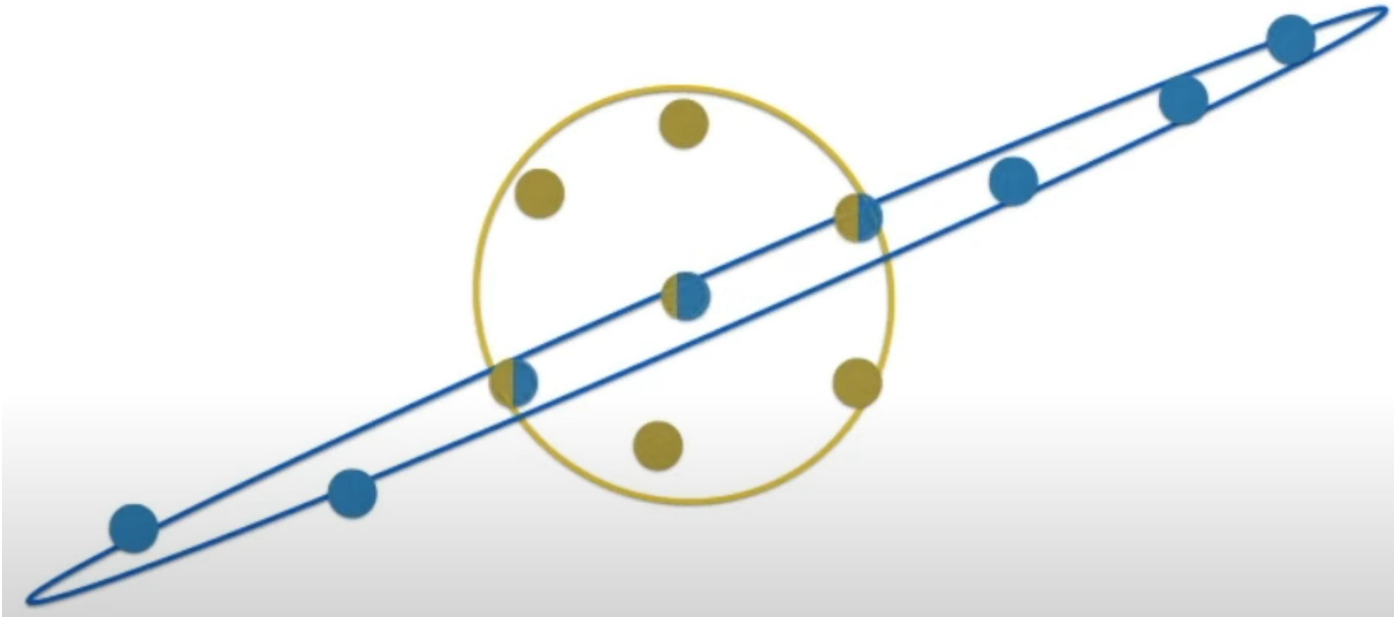
Gaussian Mixture Model (GMM)

Soft Clustering



A real-world example: Customer segmentation in an E-commerce Platform

A customer might behave like both a “bargain hunter” and a “frequent shopper”



Mixture Models

- Formally a Mixture Model is the weighted sum of a number of probability density functions (pdfs) where the weights are determined by a distribution, π

$$p(\boldsymbol{x}) = \pi_1 f_1(\boldsymbol{x}) + \pi_2 f_2(\boldsymbol{x}) + \cdots + \pi_K f_K(\boldsymbol{x})$$

$$= \sum_{i=1}^{K} \pi_i f_i(\boldsymbol{x})$$

where $\sum_{i=1}^{K} \pi_i = 1$

Gaussian Mixture Models

- GMM: the weighted sum of a number of **Gaussians** where the weights are determined by a distribution, π

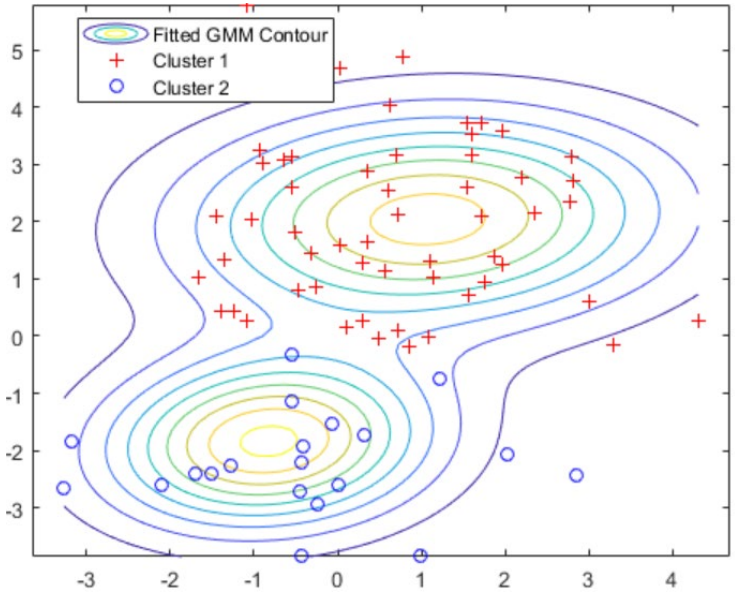
$$p(\boldsymbol{x}) = \pi_1 N(\boldsymbol{x}|\boldsymbol{u}_1, \boldsymbol{\Sigma}_1) + \pi_2 N(\boldsymbol{x}|\boldsymbol{u}_2, \boldsymbol{\Sigma}_2) + \cdots + \pi_K N(\boldsymbol{x}|\boldsymbol{u}_K, \boldsymbol{\Sigma}_K)$$

$$= \sum_{i=1}^{K} \pi_i N(\boldsymbol{x}|\boldsymbol{u}_i, \boldsymbol{\Sigma}_i)$$

where $\sum_{i=1}^{K} \pi_i = 1$

Gaussian Mixture Models

- Fit a Set of K Gaussians to the unlabeled data
- Maximum Likelihood over a GMM
- Assumption: the underlying data is generated from mixture of Gaussians.



Compute Likelihood

- We define:

$\pi_i = P(\boldsymbol{\omega}_i)$ where $\boldsymbol{\omega}_i$ is a Gaussian component $(\boldsymbol{u}_i, \boldsymbol{\Sigma}_i)$, $\sum_{i=1}^{K} \pi_i = 1$

(Prior)

$$z_i = P(\boldsymbol{\omega}_i|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\boldsymbol{\omega}_i)P(\boldsymbol{\omega}_i)}{p(\boldsymbol{x})} = \frac{p(\boldsymbol{x}|\boldsymbol{\omega}_i)P(\boldsymbol{\omega}_i)}{\sum_{j=1}^{K} P(\boldsymbol{\omega}_j)p(\boldsymbol{x}|\boldsymbol{\omega}_j)}$$

(Posterior)

Bayes' Theorem:
P(A|B)=P(B|A)P(A)/P(B)

$$z_k^n = P(\boldsymbol{\omega}_k|\boldsymbol{x}_n)$$

- Identify a likelihood function:

$$p(\boldsymbol{x_1}, \dots, \boldsymbol{x}_N|\boldsymbol{\pi}, \boldsymbol{\omega}) = \prod_{n=1}^{N} p(\boldsymbol{x}_n|\boldsymbol{\pi}, \boldsymbol{\omega})$$

x_n was drawn independently

$$= \prod_{n=1}^{N} \sum_{k=1}^{K} p(\boldsymbol{x}_n|\boldsymbol{\omega}_k)P(\boldsymbol{\omega}_k)$$

Maximum Likelihood over a GMM

- Identify a log-likelihood function:

$$\ln p(\boldsymbol{x_1}, \dots, \boldsymbol{x_N}|\boldsymbol{\pi}, \boldsymbol{\omega}) = \ln \prod_{n=1}^{N} p(\boldsymbol{x}_n|\boldsymbol{\pi}, \boldsymbol{\omega}) = \sum_{n=1}^{N} \ln p(\boldsymbol{x}_n|\boldsymbol{\pi}, \boldsymbol{\omega})$$

- Compute and set partials to 0:

$$\frac{\partial \ln p(\boldsymbol{x_1}, \dots, \boldsymbol{x_N}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})}{\partial \pi_k} = 0 \qquad \frac{\partial \ln p(\boldsymbol{x_1}, \dots, \boldsymbol{x_N}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})}{\partial \boldsymbol{\mu}_k} = 0 \qquad \frac{\partial \ln p(\boldsymbol{x_1}, \dots, \boldsymbol{x_N}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})}{\partial \boldsymbol{\Sigma}_k} = 0$$

$$\frac{\partial \ln p(\boldsymbol{x_1}, \dots, \boldsymbol{x_N}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})}{\partial \boldsymbol{\mu}_k} = \sum_{n=1}^{N} \frac{\partial \ln p(\boldsymbol{x}_n|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})}{\partial \boldsymbol{\mu}_k}$$

$$\frac{\partial \ln f(x)}{\partial x} = \frac{1}{f(x)} \frac{\partial f(x)}{\partial x}$$

$$= \sum_{n=1}^{N} \frac{1}{p(\boldsymbol{x}_n|\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Sigma})} \frac{\partial p(\boldsymbol{x}_n|\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Sigma})}{\partial \boldsymbol{\mu}_k}$$

$$= \sum_{n=1}^{N} \frac{1}{p(x_n|\pi,\mu,\Sigma)} \frac{\partial \sum_{i=1}^{K} p(\boldsymbol{x}_n|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) P(\boldsymbol{\omega}_i)}{\partial \boldsymbol{\mu}_k}$$

see next slide

Maximum Likelihood over a GMM

$$\frac{\partial \ln p(\boldsymbol{x_1}, \ldots, \boldsymbol{x_N}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})}{\partial \boldsymbol{\mu_k}} = \sum_{n=1}^{N} \frac{1}{p(\boldsymbol{x_n}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})} \frac{\partial p(\boldsymbol{x_n}|\boldsymbol{\mu_k}, \boldsymbol{\Sigma_k}) P(\boldsymbol{\omega_k})}{\partial \boldsymbol{\mu_k}}$$

$$\frac{\partial \ln f(x)}{\partial x} = \frac{1}{f(x)} \frac{\partial f(x)}{\partial x}$$

$$= \sum_{n=1}^{N} \frac{P(\boldsymbol{\omega_k}) p(\boldsymbol{x_n}|\boldsymbol{\mu_k}, \boldsymbol{\Sigma_k})}{p(\boldsymbol{x_n}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})} \frac{\partial \ln p(\boldsymbol{x_n}|\boldsymbol{\mu_k}, \boldsymbol{\Sigma_k})}{\partial \boldsymbol{\mu_k}}$$

Bayes' Theorem:
P(A|B)=P(B|A)P(A)/P(B)

$$= \sum_{n=1}^{N} \underbrace{P(\boldsymbol{\omega_k}|\boldsymbol{x_n})}_{z_k^n} \frac{\partial \ln N(\boldsymbol{x_n}|\boldsymbol{\mu_k}, \boldsymbol{\Sigma_k})}{\partial \boldsymbol{\mu_k}}$$

If univariate: $N(x_n|\mu_k, \Sigma_k) = \frac{1}{\sqrt{2\pi\Sigma_k}} \exp(-\frac{(x_n - \mu_k)^2}{2\Sigma_k})$

$$\frac{\partial \ln p(x_1, \ldots, x_N|\pi, \mu, \Sigma)}{\partial \mu_k} = \sum_{n=1}^{N} z_k^n \left(-\frac{1}{2\Sigma_k} \frac{\partial (x_n - \mu_k)^2}{\partial \mu_k}\right)$$

set partials to 0

$$= \sum_{n=1}^{N} z_k^n \frac{x_n - \mu_k}{\Sigma_k} = 0 \quad \Rightarrow \quad \mu_k^* = \frac{\sum_{n=1}^{N} z_k^n x_n}{\sum_{n=1}^{N} z_k^n}$$

Maximum Likelihood over a GMM

$$\frac{\partial \ln p(\boldsymbol{x_1}, \ldots, \boldsymbol{x_N}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})}{\partial \boldsymbol{\mu}_k} = \sum_{n=1}^{N} \frac{1}{p(\boldsymbol{x}_n|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})} \frac{\partial p(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) P(\boldsymbol{\omega}_k)}{\partial \boldsymbol{\mu}_k}$$

$$\frac{\partial \ln f(x)}{\partial x} = \frac{1}{f(x)} \frac{\partial f(x)}{\partial x}$$

$$= \sum_{n=1}^{N} \frac{P(\boldsymbol{\omega}_k) p(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{p(\boldsymbol{x}_n|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})} \frac{\partial \ln p(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\partial \boldsymbol{\mu}_k}$$

Bayes' Theorem:
P(A|B)=P(B|A)P(A)/P(B)

$$= \sum_{n=1}^{N} \underbrace{p(\boldsymbol{\omega}_k|\boldsymbol{x}_n)}_{z_k^n} \frac{\partial \ln N(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\partial \boldsymbol{\mu}_k}$$

If multivariate: $N(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \frac{1}{\sqrt{2(\pi)^d \det(\boldsymbol{\Sigma}_k)}} \exp(-\frac{1}{2}(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1}(\boldsymbol{x}_n - \boldsymbol{\mu}_k))$

$$\frac{\partial \ln p(\boldsymbol{x_1}, \ldots, \boldsymbol{x_N}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})}{\partial \boldsymbol{\mu}_k} = \sum_{n=1}^{N} z_k^n \left(-\frac{1}{2} \frac{\partial (\boldsymbol{x}_n - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1} (\boldsymbol{x}_n - \boldsymbol{\mu}_k)}{\partial \boldsymbol{\mu}_k} \right)$$

set partials to 0

$$= \sum_{n=1}^{N} z_k^n \boldsymbol{\Sigma}_k^{-1}(\boldsymbol{x}_n - \boldsymbol{\mu}_k) = \boldsymbol{0} \Rightarrow \boldsymbol{\mu}_k^* = \frac{\sum_{n=1}^{N} z_k^n \boldsymbol{x}_n}{\sum_{n=1}^{N} z_k^n}$$

Maximum Likelihood over a GMM

Similarly, after computing the log-likelihood and take partials to 0, we have

$$\boldsymbol{\mu}_k^* = \frac{\sum_{n=1}^{N} z_k^n \boldsymbol{x}_n}{\sum_{n=1}^{N} z_k^n}$$

$\mathbb{E}_k[\boldsymbol{x}]$

(Weighted average of $\boldsymbol{x}$, using z_k^n as weights)

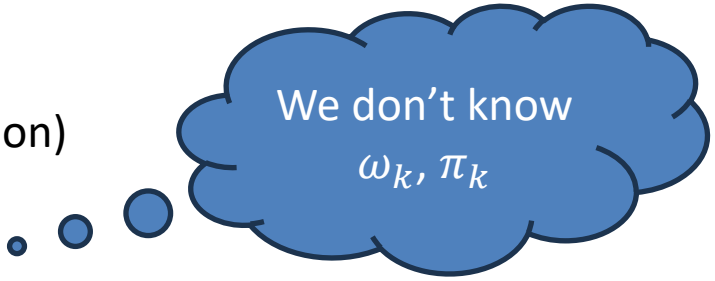
$$\boldsymbol{\Sigma}_k^* = \frac{\sum_{n=1}^{N} z_k^n (\boldsymbol{x}_n - \boldsymbol{\mu}_k^*)(\boldsymbol{x}_n - \boldsymbol{\mu}_k^*)^T}{\sum_{n=1}^{N} z_k^n}$$

$\mathbb{E}_k[(\boldsymbol{x} - \boldsymbol{\mu}_k^*)(\boldsymbol{x} - \boldsymbol{\mu}_k^*)^T]$

(Weighted average covariance of $\boldsymbol{x}$)

$$\pi_k^* = \frac{\sum_{n=1}^{N} z_k^n}{N}$$

(Effective proportion)



Where $z_k^n = P(\boldsymbol{\omega}_k|\boldsymbol{x}_n) = \frac{p(\boldsymbol{x}_n|\boldsymbol{\omega}_i)P(\boldsymbol{\omega}_i)}{\sum_{j=1}^{K} P(\boldsymbol{\omega}_j)p(\boldsymbol{x}_n|\boldsymbol{\omega}_j)}$

EM for GMMs

- Initialize the parameters $(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \pi_k)$

- Expectation-step: computes the posterior probabilities (z_k^n) based on current estimated distributions $(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \pi_k)$ derived from previous M-step

- Maximization-step: updates the parameters $(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \pi_k)$ maximizing the log-likelihood using z_k^n obtained from previous E-step

Gaussian Mixture Model Example: Start

1. Randomly select 3 Gaussians with equal $\pi_k = 1/3$

2. Calculate $z_k^n = P(\boldsymbol{\omega}_k|\boldsymbol{x}_n)$

$$= \frac{p(\boldsymbol{x}_n|\boldsymbol{\omega}_k)P(\boldsymbol{\omega}_k)}{\sum_{j=1}^{3} P(\boldsymbol{\omega}_j)p(\boldsymbol{x}_n|\boldsymbol{\omega}_j)}$$

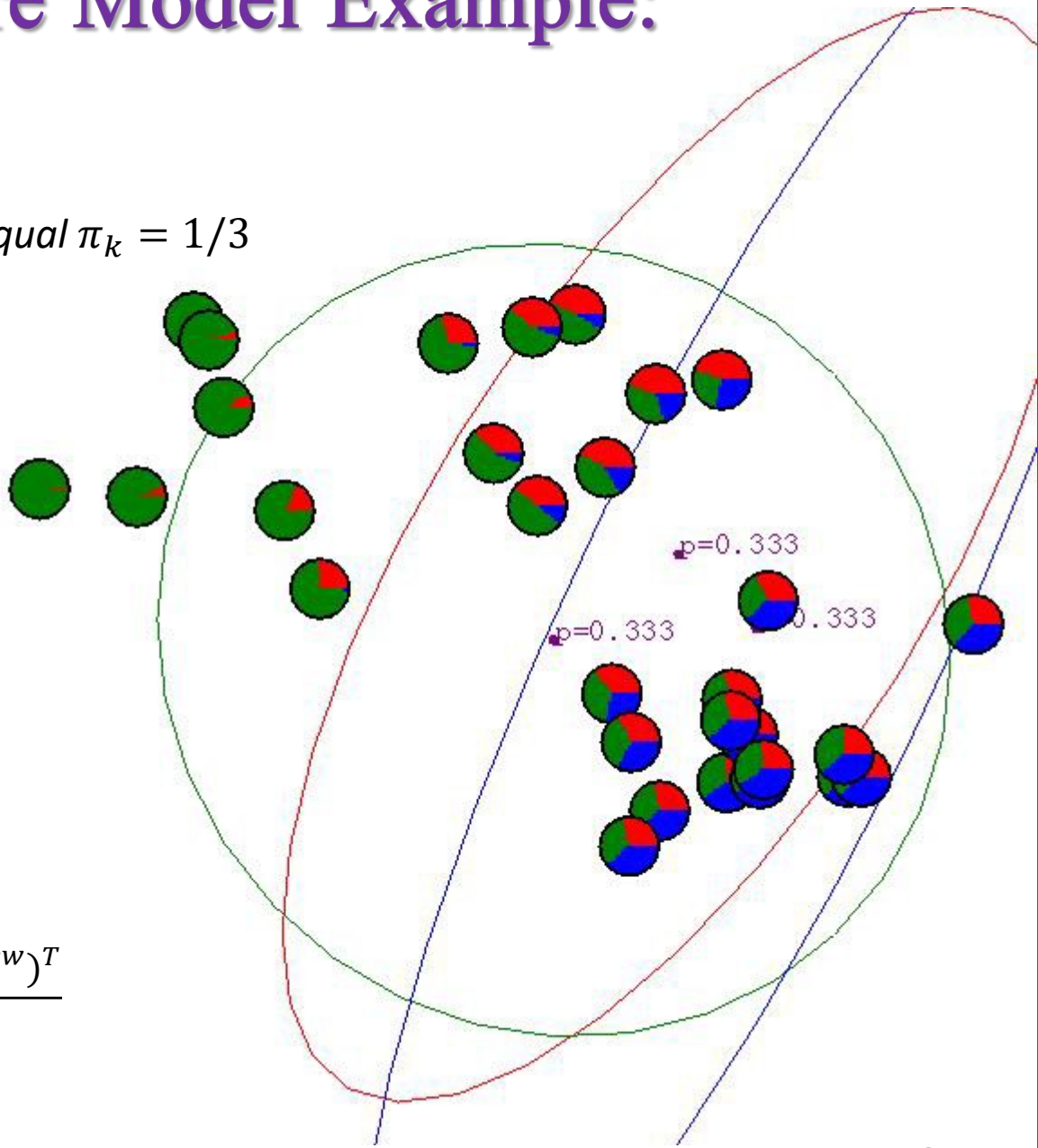
$$= \frac{N(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\pi_k}{\sum_{j=1}^{3} \pi_j N(\boldsymbol{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

3. Update parameters

$$\boldsymbol{\mu}_k^{new} = \frac{\sum_{n=1}^{N} z_k^n \boldsymbol{x}_n}{\sum_{n=1}^{N} z_k^n}$$

$$\boldsymbol{\Sigma}_k^{new} = \frac{\sum_{n=1}^{N} z_k^n (\boldsymbol{x}_n - \boldsymbol{\mu}_k^{new})(\boldsymbol{x}_n - \boldsymbol{\mu}_k^{new})^T}{\sum_{n=1}^{N} z_k^n}$$

$$\pi_k^{new} = \frac{\sum_{n=1}^{N} z_k^n}{N}$$

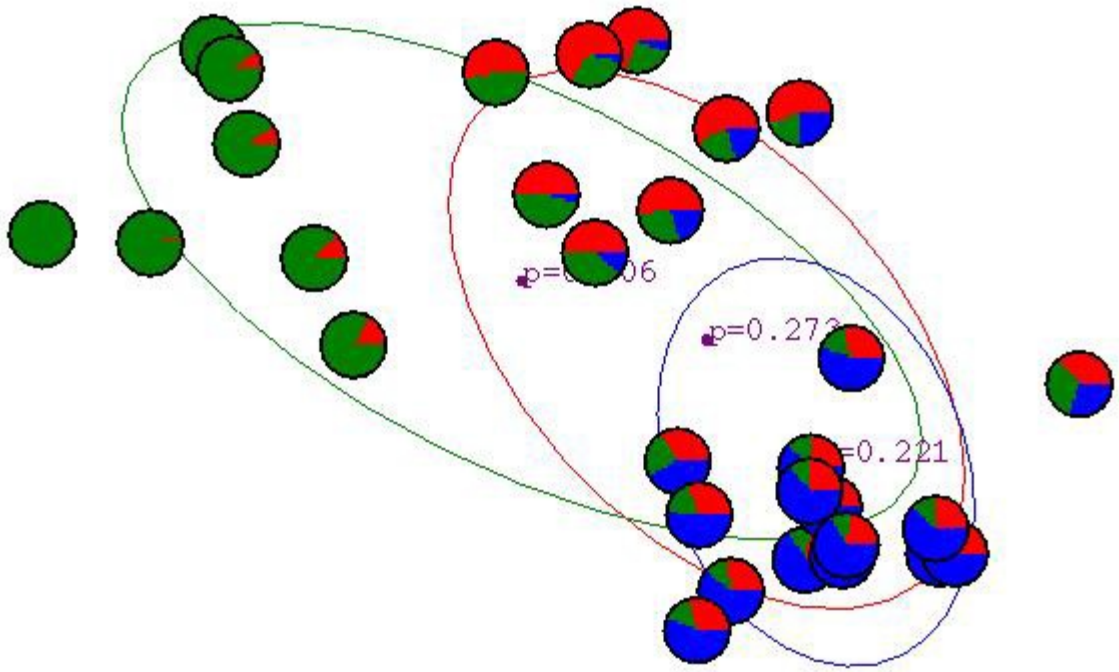


After 1st iteration

$\pi_1 = 0.506$

$\pi_2 = 0.273$

$\pi_3 = 0.221$

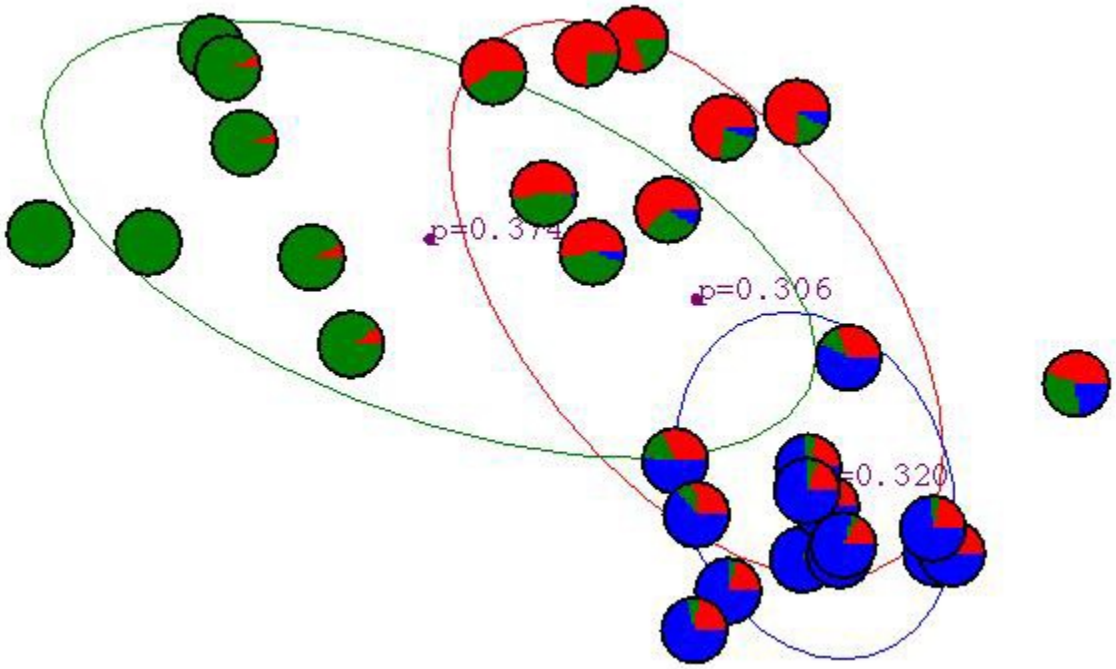


After 2nd iteration

$\pi_1 = 0.374$

$\pi_2 = 0.306$

$\pi_3 = 0.320$

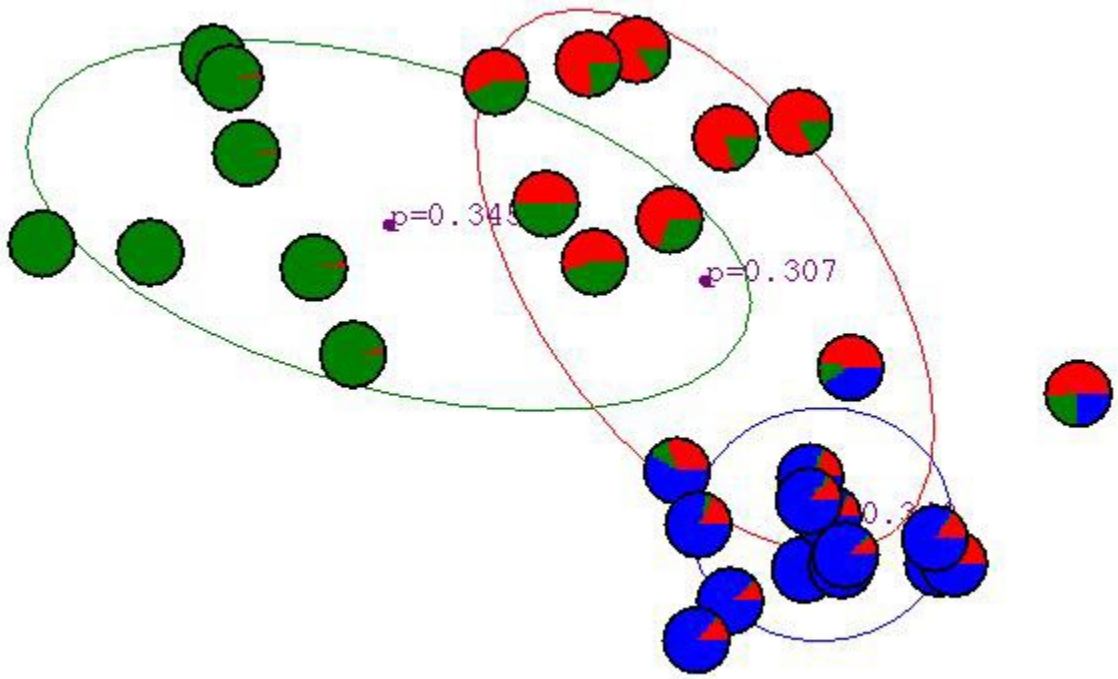


After 3rd iteration

$\pi_1 = 0.345$

$\pi_2 = 0.307$

$\pi_3 = 0.348$

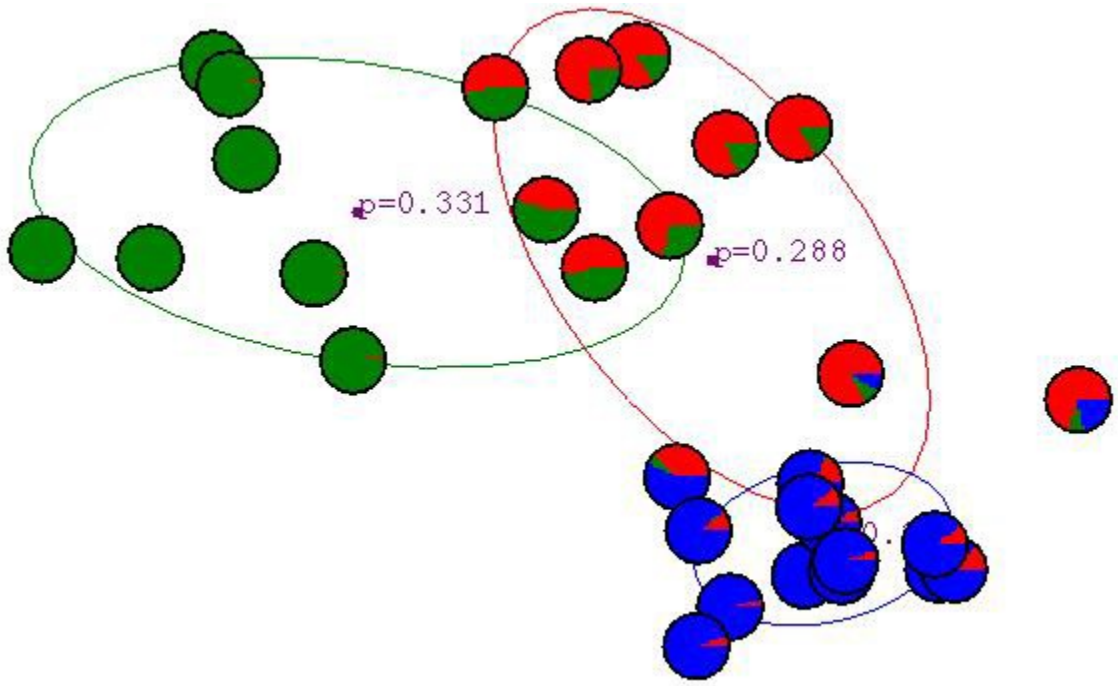


After 4th iteration

$\pi_1 = 0.331$

$\pi_2 = 0.288$

$\pi_3 = 0.381$

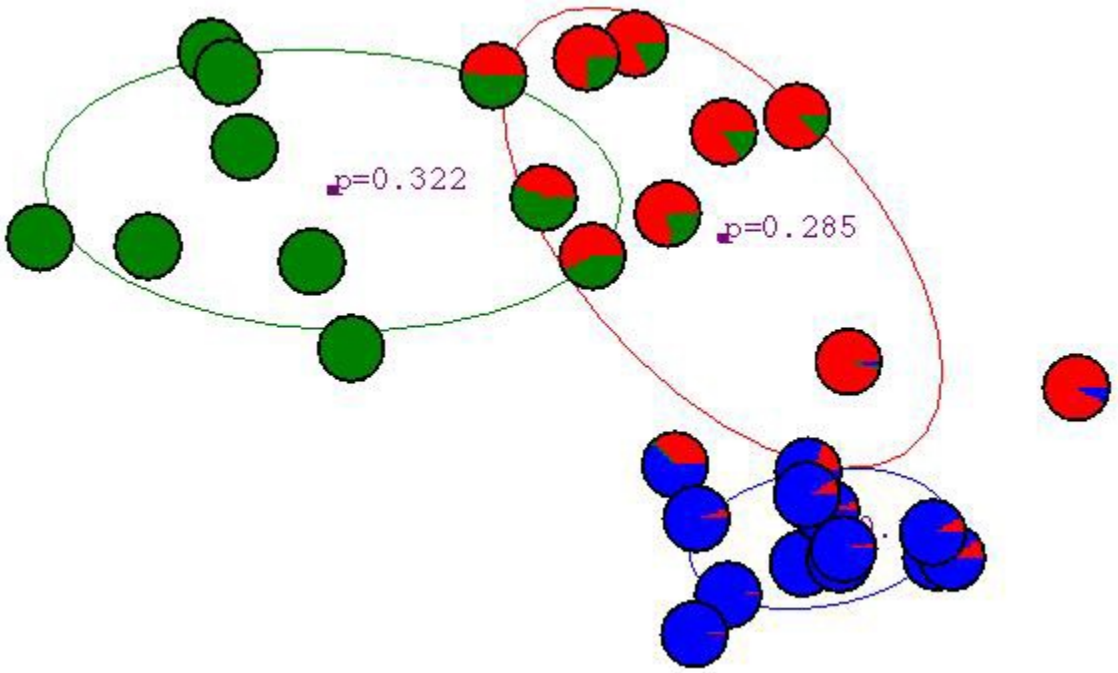


After 5th iteration

$\pi_1 = 0.322$

$\pi_2 = 0.285$

$\pi_3 = 0.393$

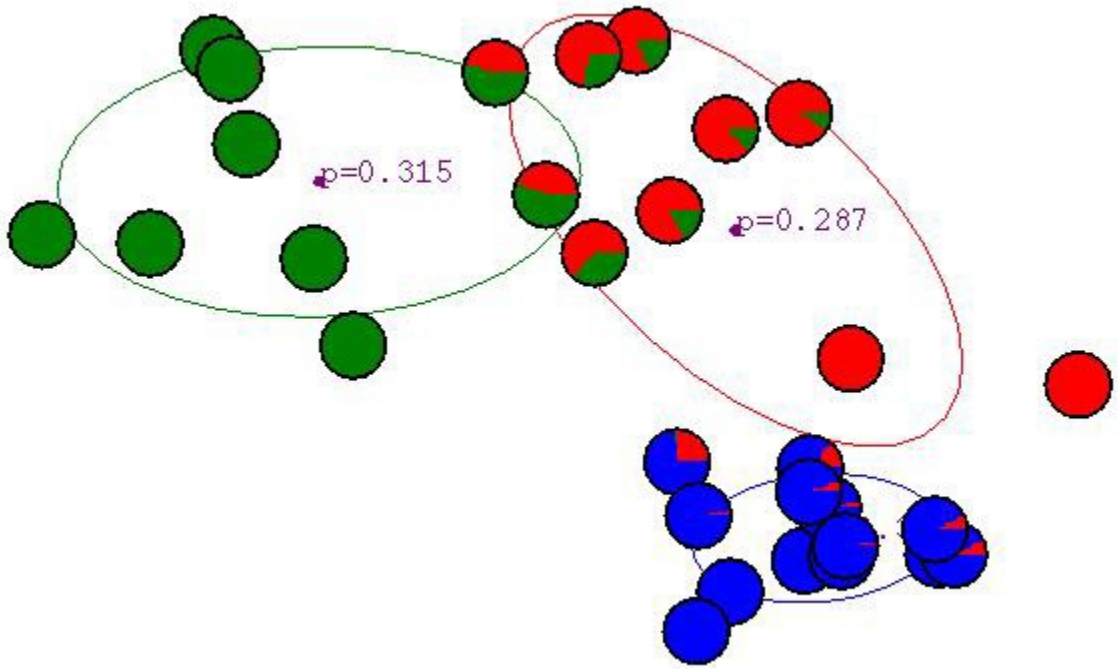


After 6th iteration

$\pi_1 = 0.315$

$\pi_2 = 0.287$

$\pi_3 = 0.398$

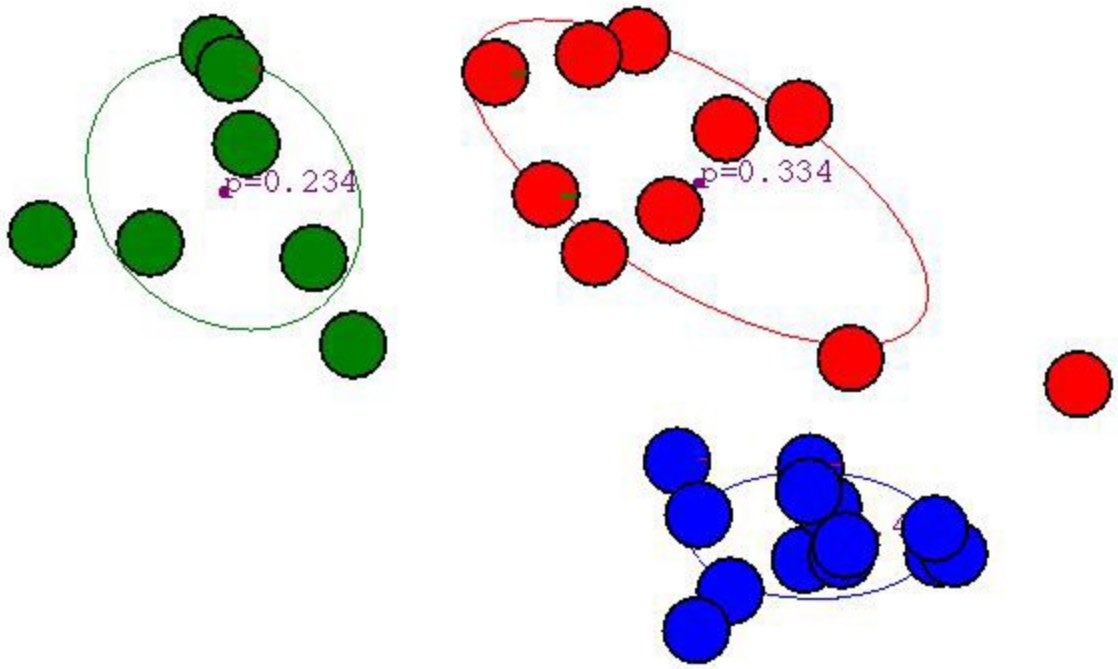


After 20th iteration

$\pi_1 = 0.234$

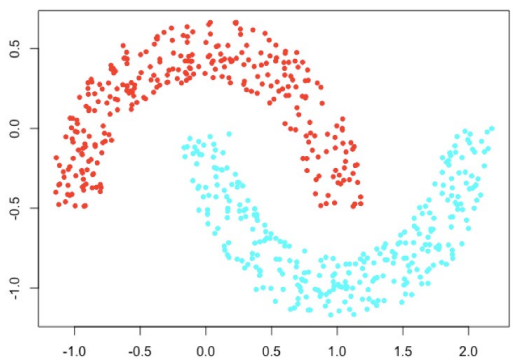
$\pi_2 = 0.334$

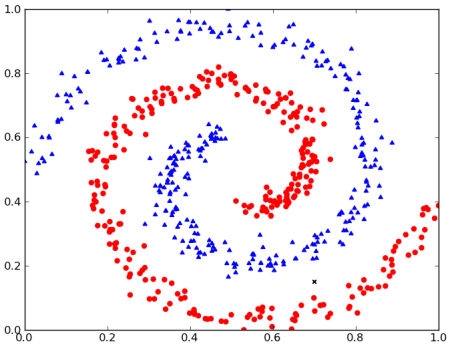
$\pi_3 = 0.432$



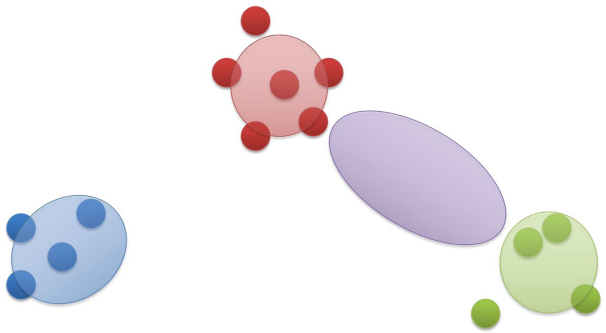
Limitations of GMM

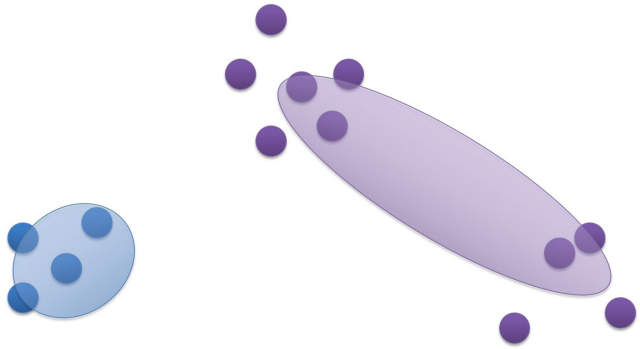
- *Assume Gaussian clusters*





- *Predefine K*



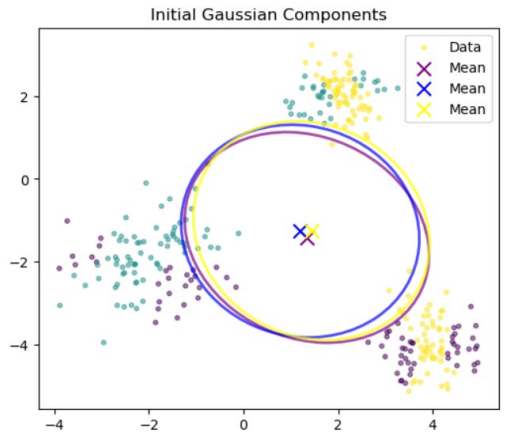


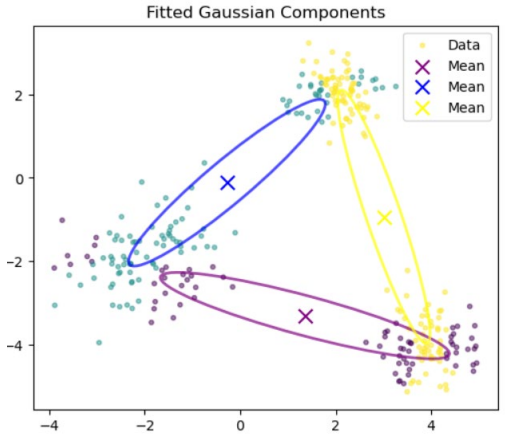
Possible Solution: Elbow method?

Limitations of GMM

- *Sensitive to initialization*

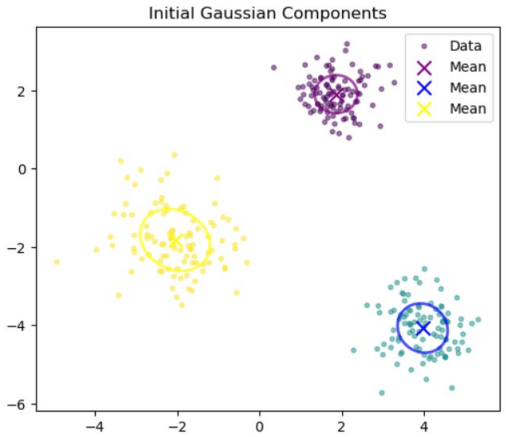
Initial

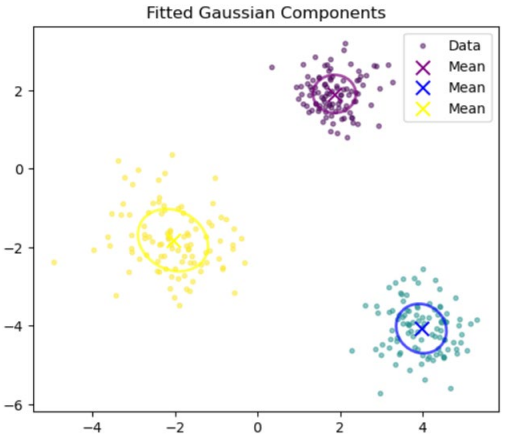




Fitted

Possible Solution: K-means for initialization





- ✓ Initial $\boldsymbol{\mu}_k$: centroid from K-means
- ✓ Initial $\boldsymbol{\Sigma}_k$: Covariance matrix for each cluster after K-means
- ✓ Initial π_k : Proportion of points in each cluster after K-means

Limitations of GMM

- *Sensitive to outliers*
- *Scales poorly compared to simpler methods like K-means*

Quick Question

Suppose you fit a Gaussian Mixture Model (GMM) with three components (K=3) to a dataset. After training, you find that a particular data point **(x_1)** has the following probabilities of belonging to each cluster:

- **Cluster 1:** 0.2
- **Cluster 2:** 0.5
- **Cluster 3:** 0.3

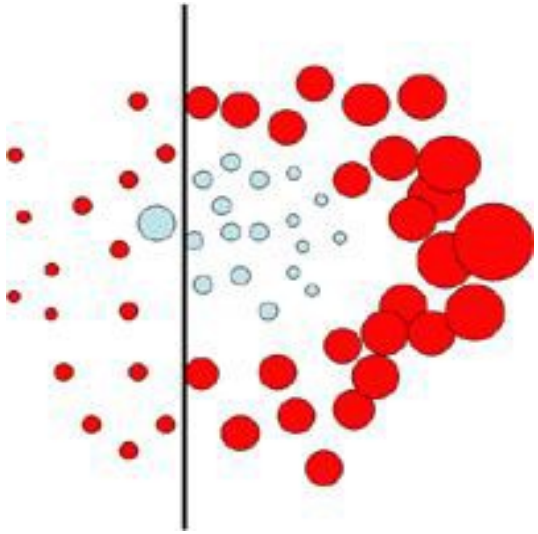
Which of the following do these probabilities represent?

A. z_k^1

B. π_k

C. $N(\boldsymbol{x}_1|\boldsymbol{u}_k, \boldsymbol{\Sigma}_k)$

D. None of the above



Boosting

(Supervised)

Boosting

- Combines multiple weak classifiers to form a strong classifier:

$$F(x) = \alpha_1 f_1(x) + \alpha_2 f_2(x) + \alpha_3 f_3(x) + \ldots$$

Strong classifier

Feature vector

Weight

Weak classifier

$f_k(x)$ from a family of weak classifiers

e.g., decision stump,
(one-level decision tree)

logistic regression (limited features),

linear SVM,

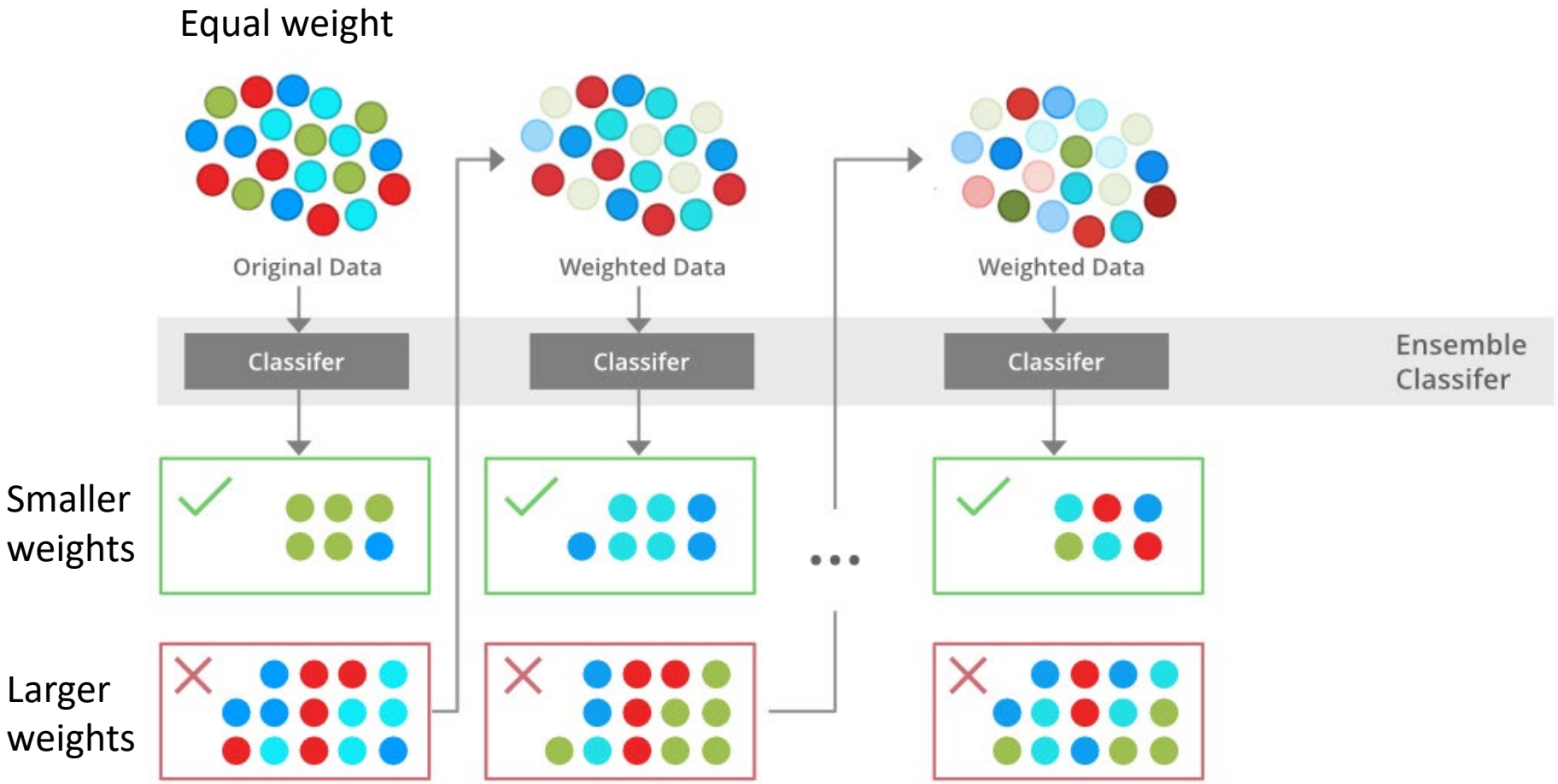
KNN (smaller K),

⋮

Weak Classifier: low prediction accuracy, slightly better than random guessing

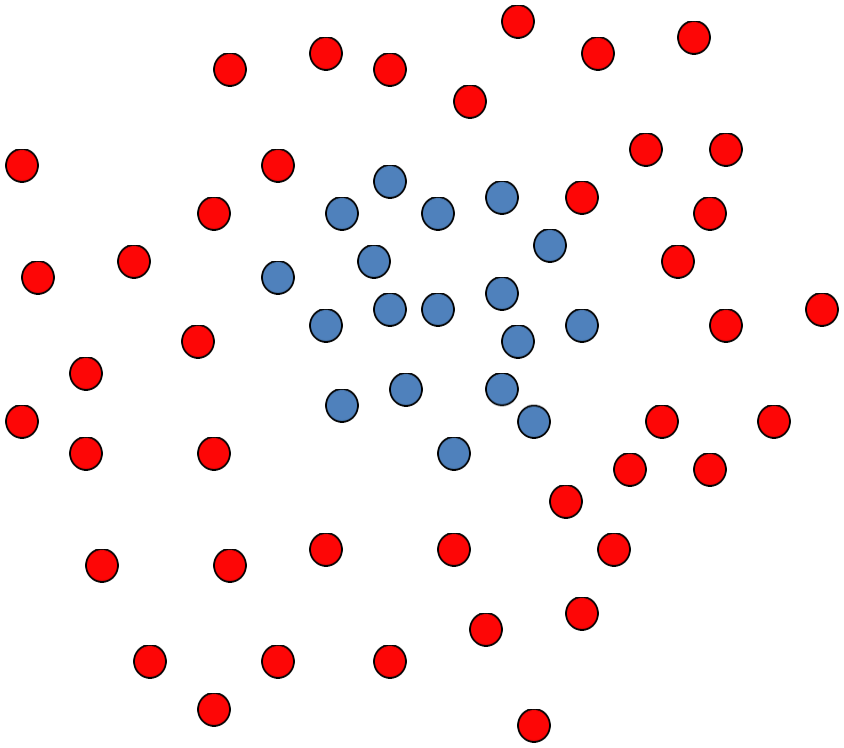
Strong Classifier: high prediction accuracy

General Training Process of Boosting



Toy Example

- It is a sequential procedure:



Each data point $\boldsymbol{x}_t$

has a class label:

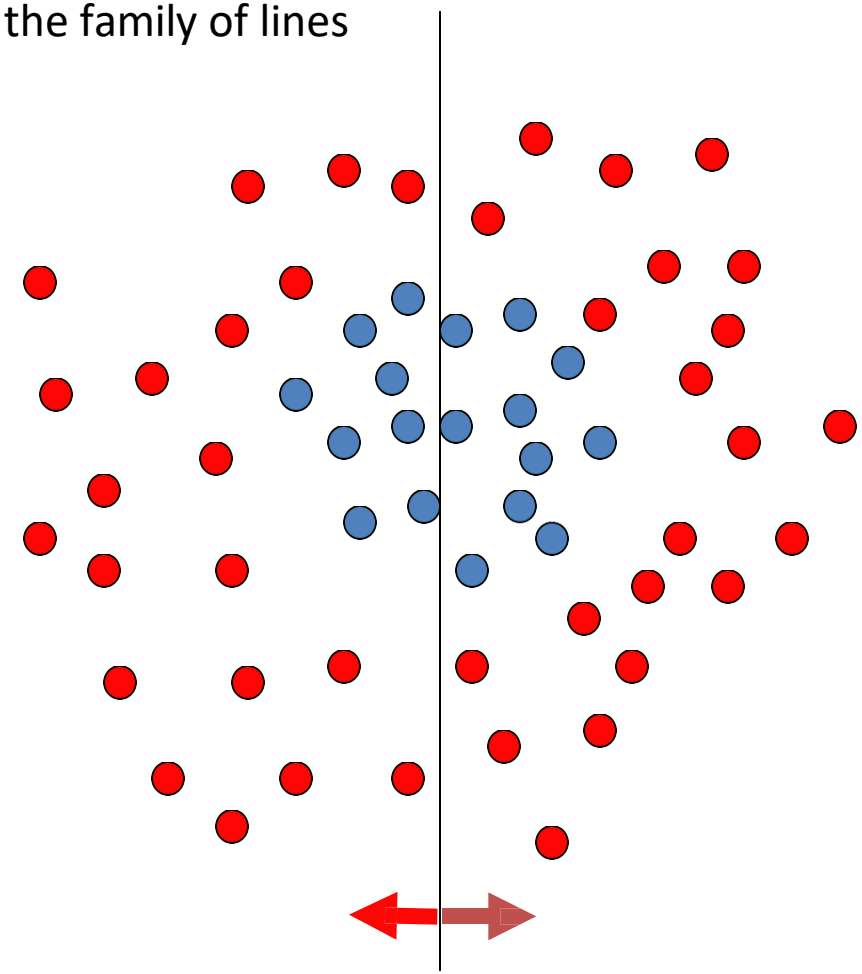
$$y_t = \begin{cases} +1 & (\bullet) \\ -1 & (\bullet) \end{cases}$$

and a weight:

$w_t = 1$

Toy Example

Weak learners from the family of lines



Each data point has

a class label:

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

and a weight:

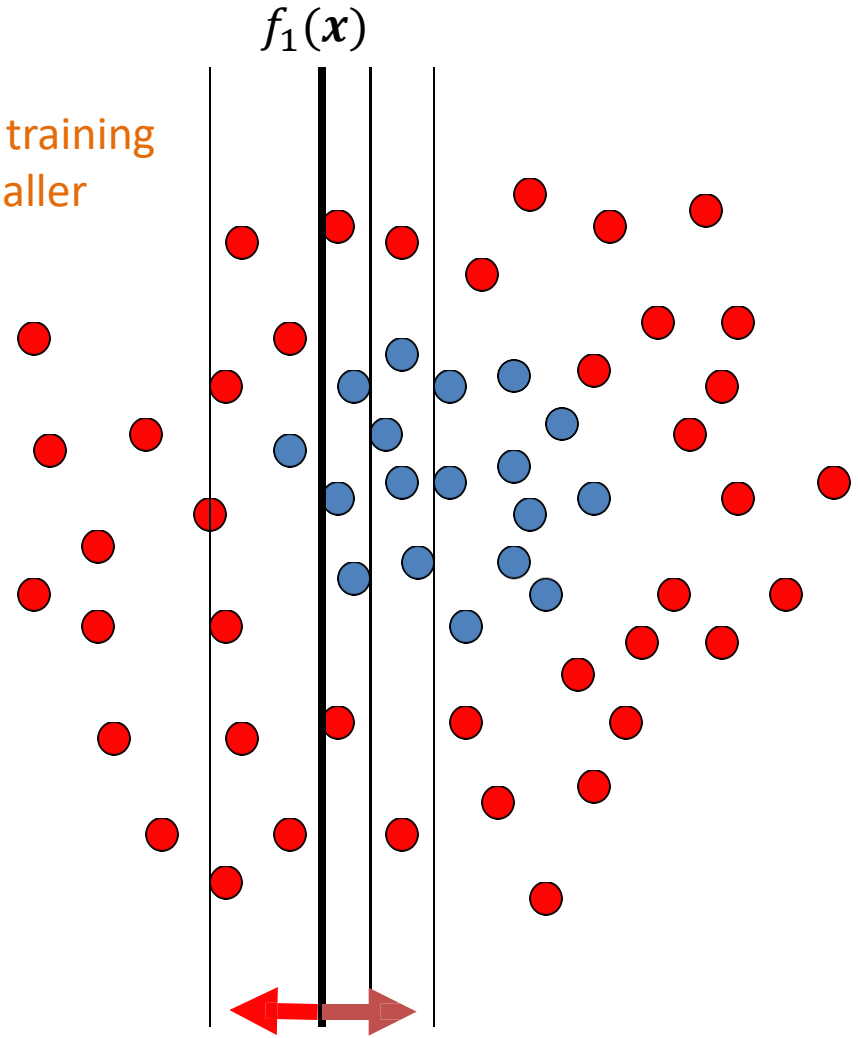
$$w_t = 1$$

p(error) = 0.5 it is at chance

Toy Example

$F(\boldsymbol{x}) = \alpha_1 f_1(\boldsymbol{x})$

where α_1 is larger if training error for $f_1(\boldsymbol{x})$ is smaller



This one seems to be the best

Each data point has

a class label:

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

and a weight:

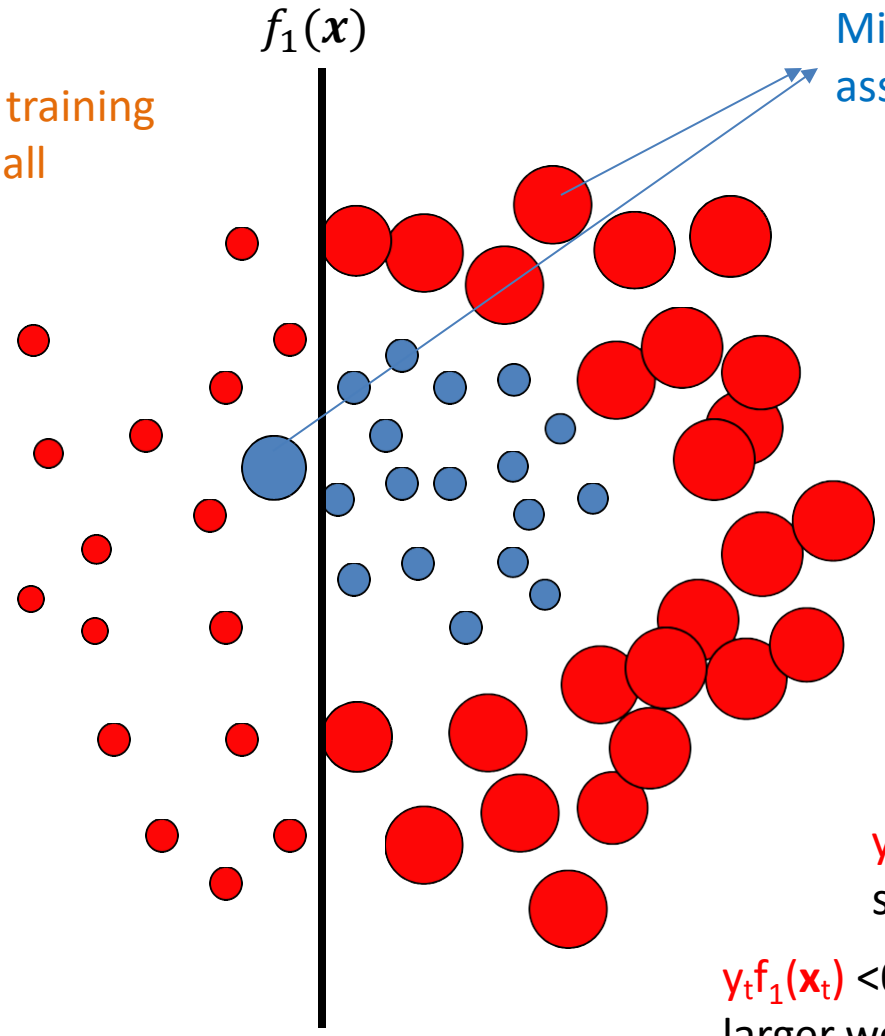
$w_t = 1$

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

Toy Example

$F(\boldsymbol{x}) = \alpha_1 f_1(\boldsymbol{x})$

where α_1 is larger if training error for $f_1(\boldsymbol{x})$ is small



Misclassified samples are assigned with larger weights.

Each data point has

a class label:

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

We update the weights:

$w_t \leftarrow w_t \exp\{-y_t f_1(\mathbf{x}_t)\}$

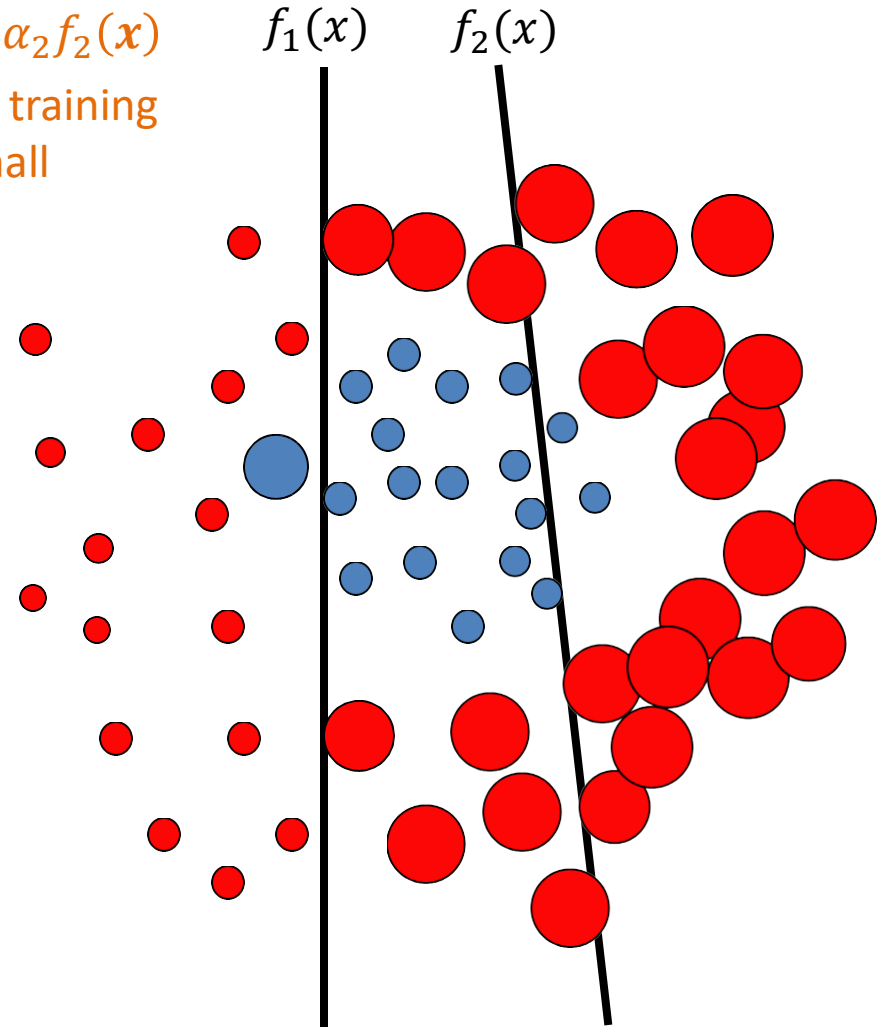
$y_t f_1(\mathbf{x}_t) > 0$: larger (similar), smaller weight

$y_t f_1(\mathbf{x}_t) < 0$: smaller (dissimilar), larger weight

Toy Example

$F(\boldsymbol{x}) = \alpha_1 f_1(\boldsymbol{x}) + \alpha_2 f_2(\boldsymbol{x})$

where α_2 is larger if training error for $f_2(\boldsymbol{x})$ is small



Each data point has a class label:

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

We update the weights:

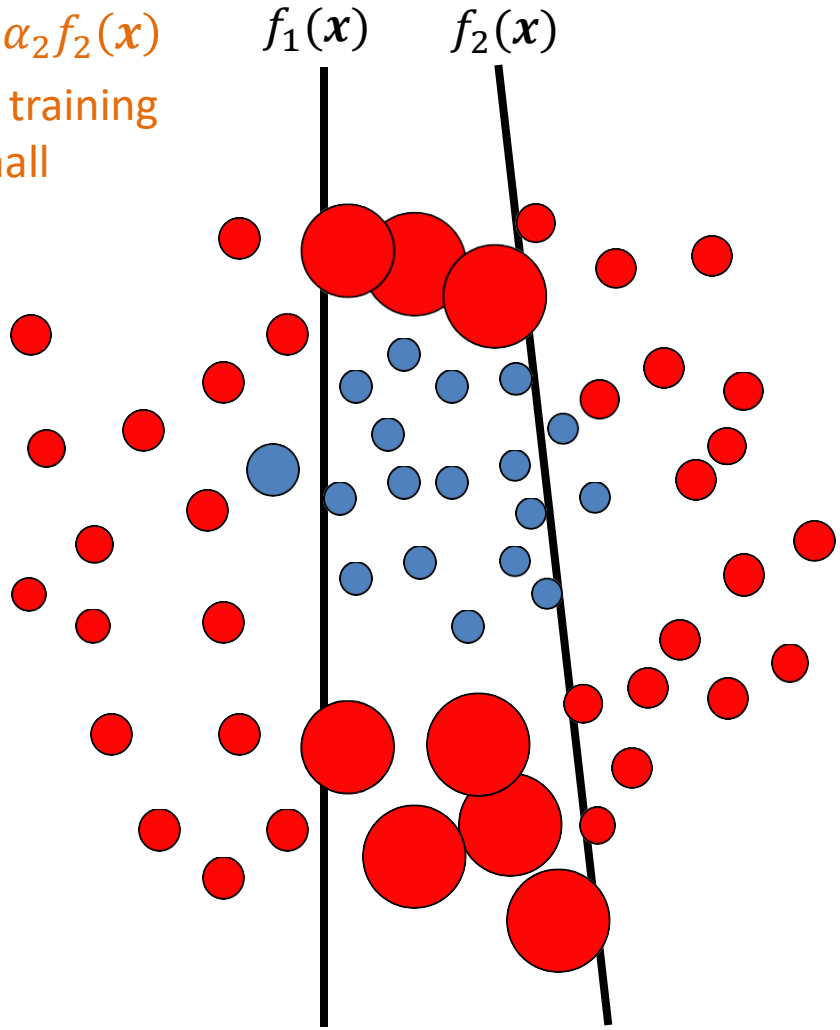
$$w_t \leftarrow w_t \exp\{-y_t f_2(\mathbf{x}_t)\}$$

Similarly, we learn another weak classifier

Toy Example

$F(\boldsymbol{x}) = \alpha_1 f_1(\boldsymbol{x}) + \alpha_2 f_2(\boldsymbol{x})$

where α_2 is larger if training error for $f_2(\boldsymbol{x})$ is small



Reweighting

Each data point has a class label:

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

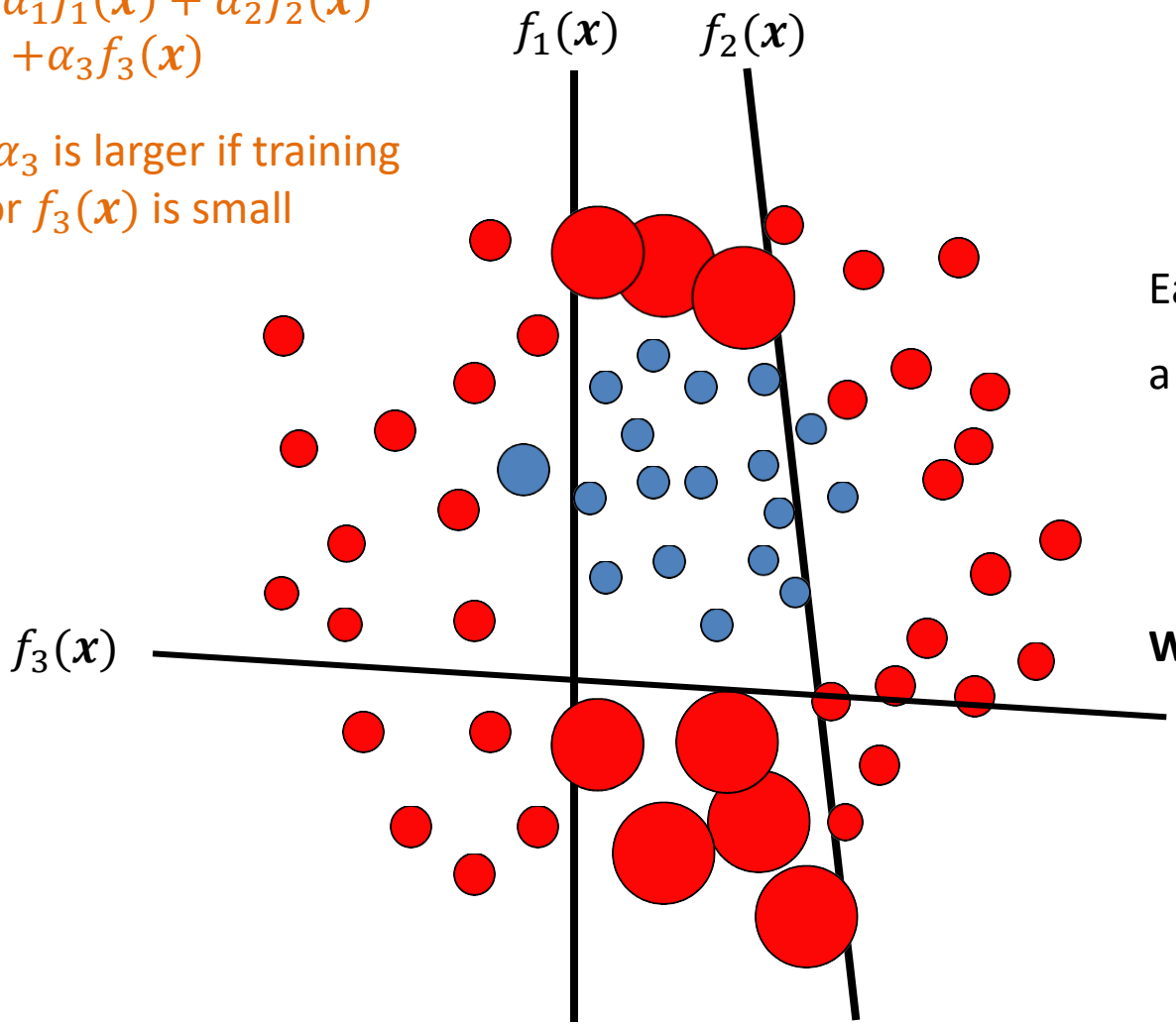
We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t f_2(\mathbf{x}_t)\}$$

Toy Example

$$F(\boldsymbol{x}) = \alpha_1 f_1(\boldsymbol{x}) + \alpha_2 f_2(\boldsymbol{x}) + \alpha_3 f_3(\boldsymbol{x})$$

where α_3 is larger if training error for $f_3(\boldsymbol{x})$ is small



Each data point has a class label:

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

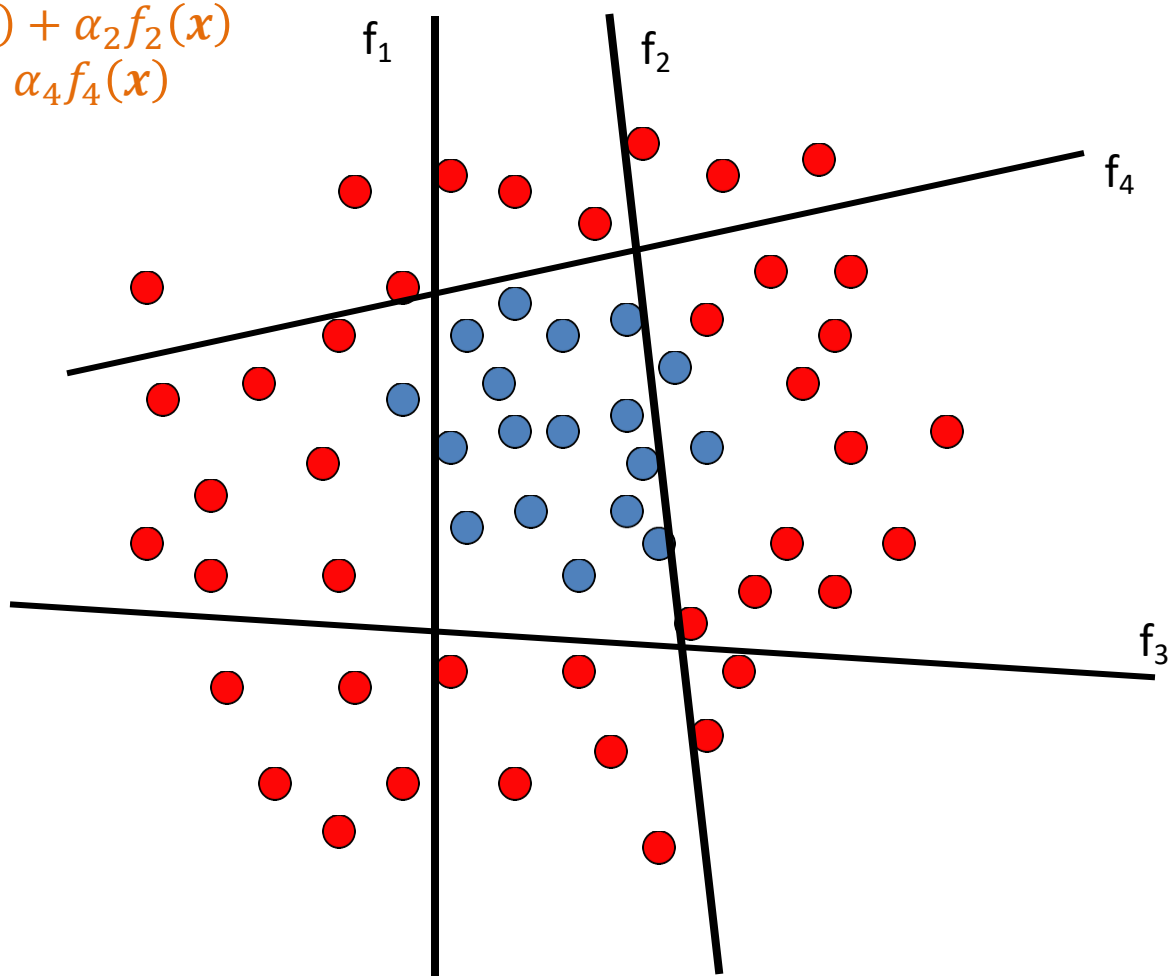
We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t f_3(\mathbf{x}_t)\}$$

Similarly, we learn another weak classifier

Toy Example

$$F(\boldsymbol{x}) = \alpha_1 f_1(\boldsymbol{x}) + \alpha_2 f_2(\boldsymbol{x}) + \alpha_3 f_3(\boldsymbol{x}) + \alpha_4 f_4(\boldsymbol{x})$$



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

Boosting

- For different cost function and minimization algorithm, the result is a different flavor of Boosting

- We shall discuss Adaboost
  - Proposed by Y. Freund & R. Schapire in 1995
  - First practical and widely used boosting
  - Exponential loss

- A variant of Adaboost: gentleBoost
  - More stable and robust alternative to AdaBoost
  - Squared error

Adaboost - Adaptive Boosting

Training dataset $\boldsymbol{D} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$, where $y_i \in \{-1, +1\}$

1. Start with weights $w_i^{(1)} = \frac{1}{N}, i = 1,2, \ldots, N, F(\boldsymbol{x}) = 0$

2. Repeat for *m*=1, 2,…, *M*:

(a) Train a weak classifier $f_m(\boldsymbol{x}) \in \{-1, +1\}$ using the weighted dataset, such that the following weighted classification error is minimized:

$$\epsilon_m = \sum_{i=1}^{N} w_i^{(m)} [1_{f_m(\boldsymbol{x}_i) \neq y_i}]$$

(b) Update $F(\boldsymbol{x}) \leftarrow F(\boldsymbol{x}) + \alpha_m f_m(\boldsymbol{x})$, where $\alpha_m = \frac{1}{2} \ln(\frac{1-\epsilon_m}{\epsilon_m})$

(c) Update $w_i^{(m+1)} \leftarrow w_i^{(m)} \exp(-\alpha_m y_i f_m(\boldsymbol{x}_i))$ and renormalize the weights

$$w_i^{(m+1)} \leftarrow \frac{w_i^{(m)} \exp(-\alpha_m y_i f_m(\boldsymbol{x}_i))}{\sum_{i=1}^{N} w_i^{(m)} \exp(-\alpha_m y_i f_m(\boldsymbol{x}_i))}$$

3. H(**x**)=sign(F(**x**))

Adaboost - Adaptive Boosting

Why $\alpha_m = \frac{1}{2}\ln\left(\frac{1-\epsilon_m}{\epsilon_m}\right)$? where $\epsilon_m = \sum_{i=1}^{N} w_i^{(m)}[1_{f_m(x_i)\neq y_i}]$

Objective Function: $J(F) = \mathbb{E}[e^{-yF(x)}]$

Objective Function of $f_m(x_i)$: $J(f_m) = J_m(F) = \mathbb{E}[e^{-yF_m(x)}]$

$$w_i^{(1)} = \frac{1}{N}$$

$$w_i^{(2)} = \frac{w_i^{(1)}\exp(-\alpha_1 y_i f_1(x_i))}{\sum_{i=1}^{N} w_i^{(1)}\exp(-\alpha_1 y_i f_1(x_i))} = \frac{\exp(-y_i F_1(x_i))}{\sum_{i=1}^{N}\exp(-y_i F_1(x_i))}$$

$$w_i^{(3)} = \frac{w_i^{(2)}\exp(-\alpha_1 y_i f_1(x_i))}{\sum_{i=1}^{N} w_i^{(2)}\exp(-\alpha_1 y_i f_1(x_i))} = \frac{\exp(-y_i F_2(x_i))}{\sum_{i=1}^{N}\exp(-y_i F_2(x_i))}$$

$$\vdots$$

$$w_i^{(m)} = \frac{\exp(-y_i F_{m-1}(x_i))}{\sum_{i=1}^{N}\exp(-y_i F_{m-1}(x_i))}$$

$$= \mathbb{E}[e^{-y\left(F_{m-1}(x)+\alpha_m f_m(x)\right)}]$$

$$= \mathbb{E}[e^{-yF_{m-1}(x)}e^{-y\alpha_m f_m(x)}]$$

$$= \sum_{i=1}^{N} \frac{e^{-y_i F_{m-1}(x_i)}}{\sum_{i=1}^{N} e^{-y_i F_{m-1}(x_i)}} e^{-\alpha_m y_i f_m(x_i)}$$

$$= \sum_{i=1}^{N} w_i^{(m)} e^{-\alpha_m y_i f_m(x_i)}$$

gentleBoosting

Training dataset ***D*** = $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$, where $y_i \in \{-1, +1\}$

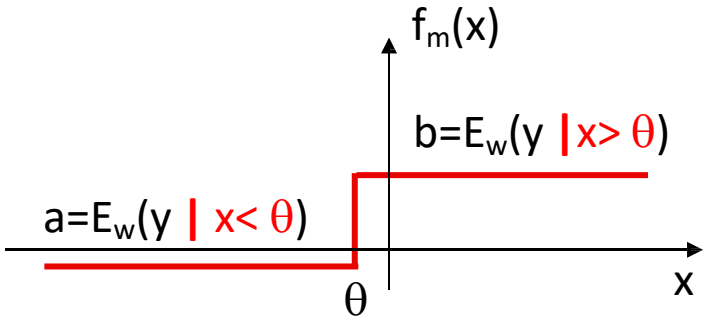
1. Start with weights $w_i^{(1)} = \frac{1}{N}, i = 1,2, \ldots, N, F(\boldsymbol{x}) = 0$

2. Repeat for *m*=1, 2,…, *M*:

   (a) Fit the regression stump $f_m(\boldsymbol{x})$ by weighted least-squares of y_i to $\boldsymbol{x}_i$ with weights $w_i^{(m)}$

$$f_m(x) = a[x_k < \theta] + b[x_k \geq \theta]$$

Indicator function



Four parameters: $\phi = [a, b, \theta, k]$

gentleBoosting

Training dataset $\boldsymbol{D} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$, where $y_i \in \{-1, +1\}$

1. Start with weights $w_i^{(1)} = \frac{1}{N}, i = 1,2, \ldots, N, F(\boldsymbol{x}) = 0$

2. Repeat for m=1, 2,…, M:

   (a) Fit the regression stump $f_m(\boldsymbol{x})$ by weighted least-squares of y_i to $\boldsymbol{x}_i$ with weights $w_i^{(m)}$

$$J(f_m) = \sum_{i=1}^{N} w_i^{(m)} (y_i - f_m(\boldsymbol{x}_i))^2 = \sum_{i=1}^{N} w_i^{(m)} (1 - y_i f_m(\boldsymbol{x}_i))^2$$

which is an approximation of $J(f_m) = \sum_{i=1}^{N} w_i^{(m)} e^{-y_i f_m(\boldsymbol{x}_i)}$

$$\approx \sum_{i=1}^{N} w_i^{(m)} (1 - y_i f_m(\boldsymbol{x}_i))^2$$

gentleBoosting

Training dataset ***D*** = $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$, where $y_i \in \{-1, +1\}$

1. Start with weights $w_i^{(1)} = \frac{1}{N}, i = 1,2, \ldots, N, F(\boldsymbol{x}) = 0$

2. Repeat for *m*=1, 2,…, *M*:

   (a) Fit the regression stump $f_m(\boldsymbol{x})$ by weighted least-squares of y_i to $\boldsymbol{x}_i$ with weights w_i

   (b) Update $F(\boldsymbol{x}) \leftarrow F(\boldsymbol{x}) + f_m(\boldsymbol{x})$

   (c) Update $w_i^{(m+1)} \leftarrow \exp(-y_i F(\boldsymbol{x}_i))$ and renormalize the weights

3. H(**x**)=sign(F(**x**))

$$w_i^{(m+1)} = \frac{\exp(-y_i F_m(\boldsymbol{x}_i))}{\sum_{i=1}^{N} \exp(-y_i F_m(\boldsymbol{x}_i))}$$

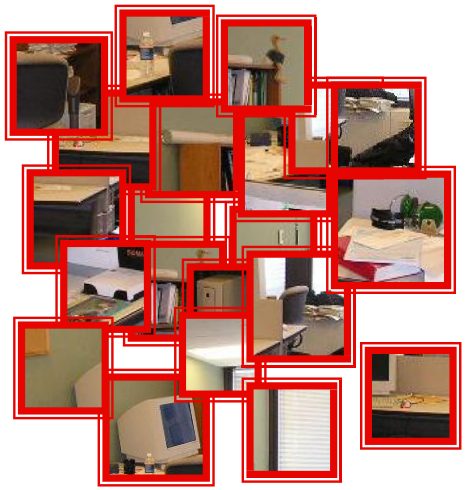
Boosting with Object Detection

Object detection is formulated as a classification problem.

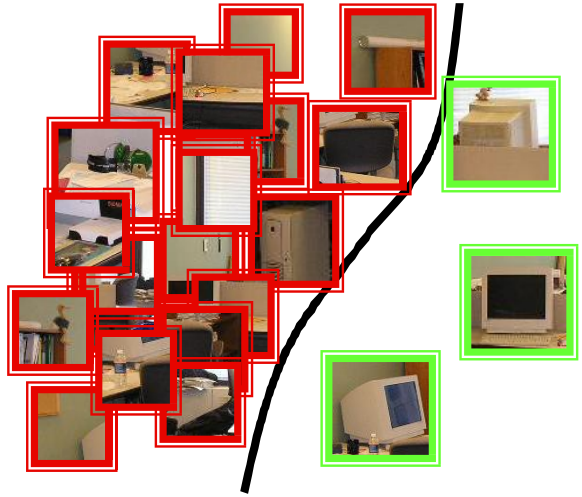
The image is partitioned into a set of overlapping windows,

… and a decision is taken at each window about if it contains a target object or not.

Where are the screens?



Bag of image patches



In some feature space

Boosting with Object Detection

- Formulation: binary classification

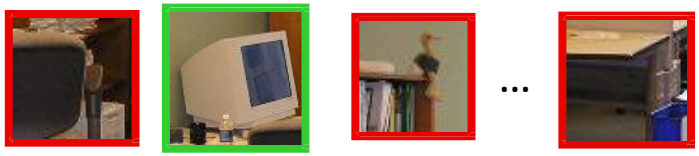



Image patch $\mathbf{x}$ = $\mathbf{x}_1$ $\mathbf{x}_2$ $\mathbf{x}_3$ … $\mathbf{x}_N$ $\mathbf{x}_{N+1}$ $\mathbf{x}_{N+2}$ … $\mathbf{x}_{N+M}$

Labels y = -1 +1 -1 -1 ? ? ?

Training data: each image patch is labeled as containing the object or background

Test data

- Extract critical features for each image patch

  --Train a weak detector that picks the best feature sequentially

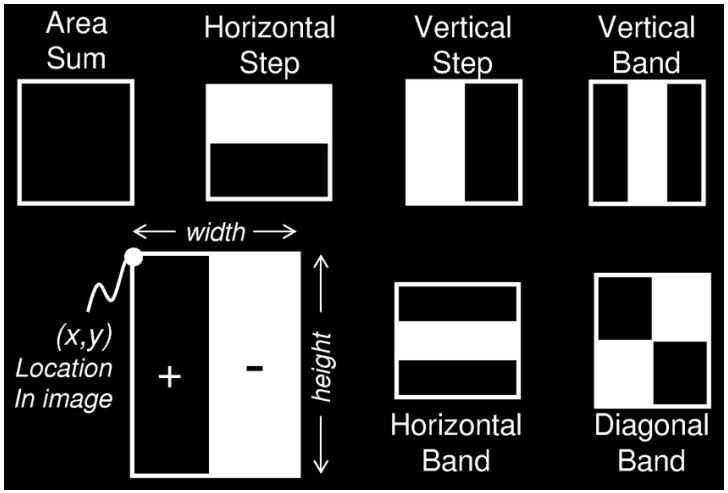
  --Aggregate the weak detectors

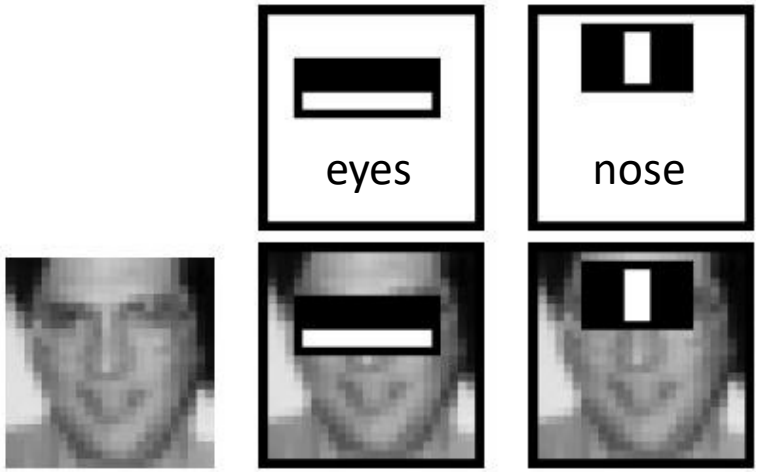
Features -> Weak Detectors

Haar filters and integral image

Viola and Jones, ICCV 2001

Rectangular regions of an image that compare intensity differences between adjacent regions





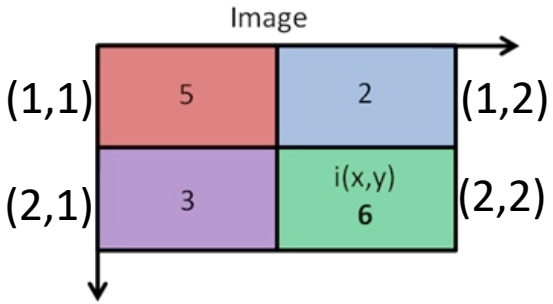
Features -> Weak Detectors

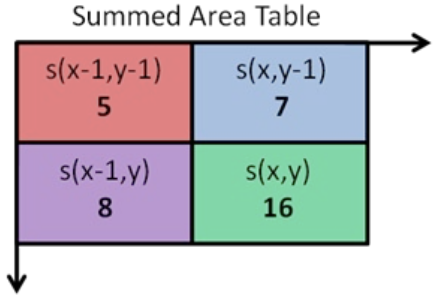
Haar filters and integral image

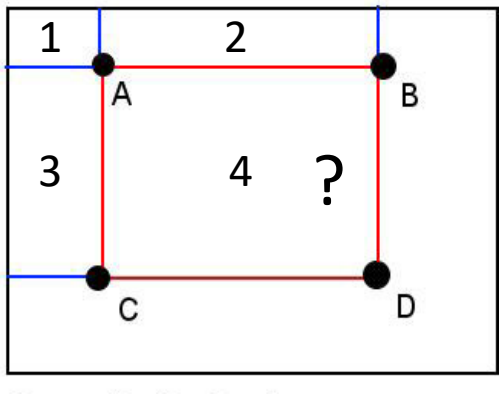
Viola and Jones, ICCV 2001

Intermediate representation to compute rectangle features rapidly

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



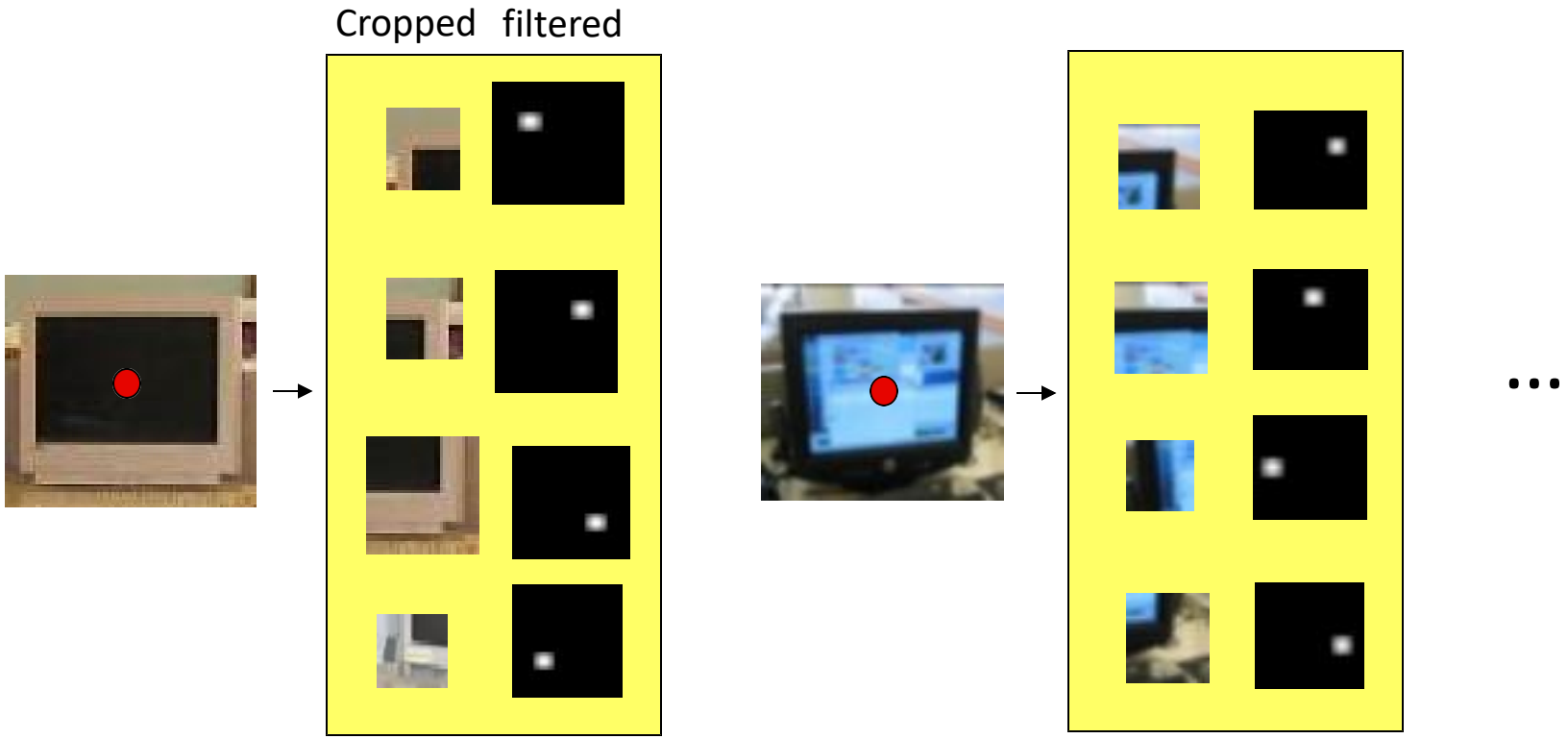




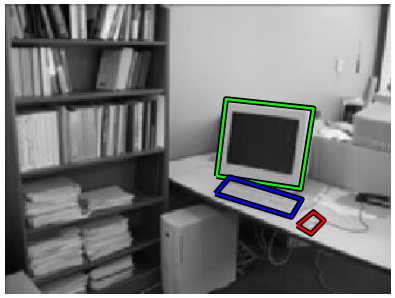
Suppose A, B, C, D are the integral images

Features -> Weak Detectors

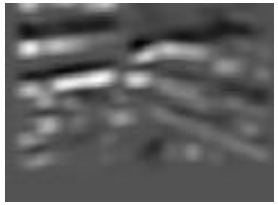
After extracting the features for each image patch using the Haar filters, the filtered responses highlight significant structures like edges, corners, etc.



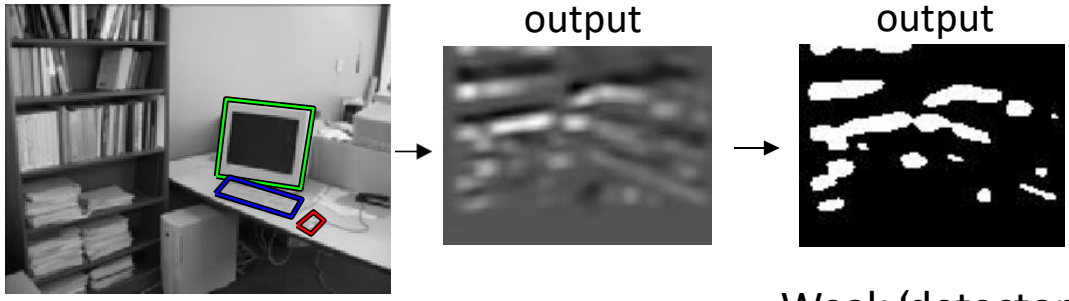
Features -> Weak Detectors



Feature output



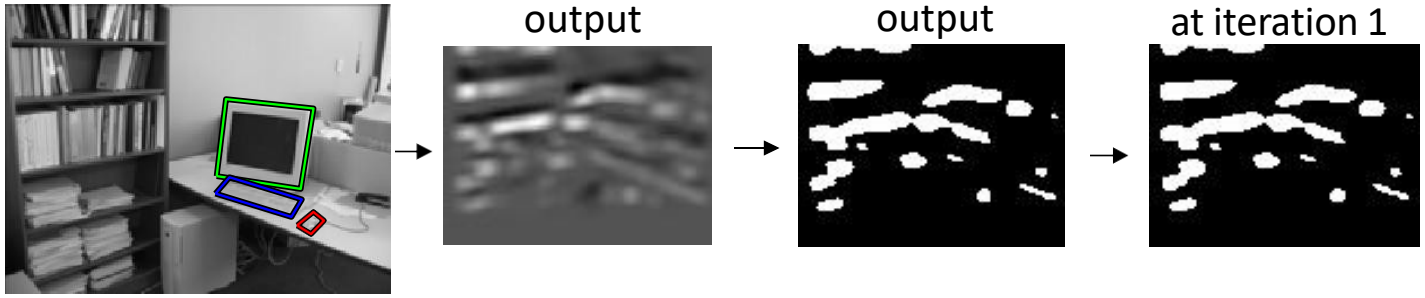
Features -> Weak Detectors



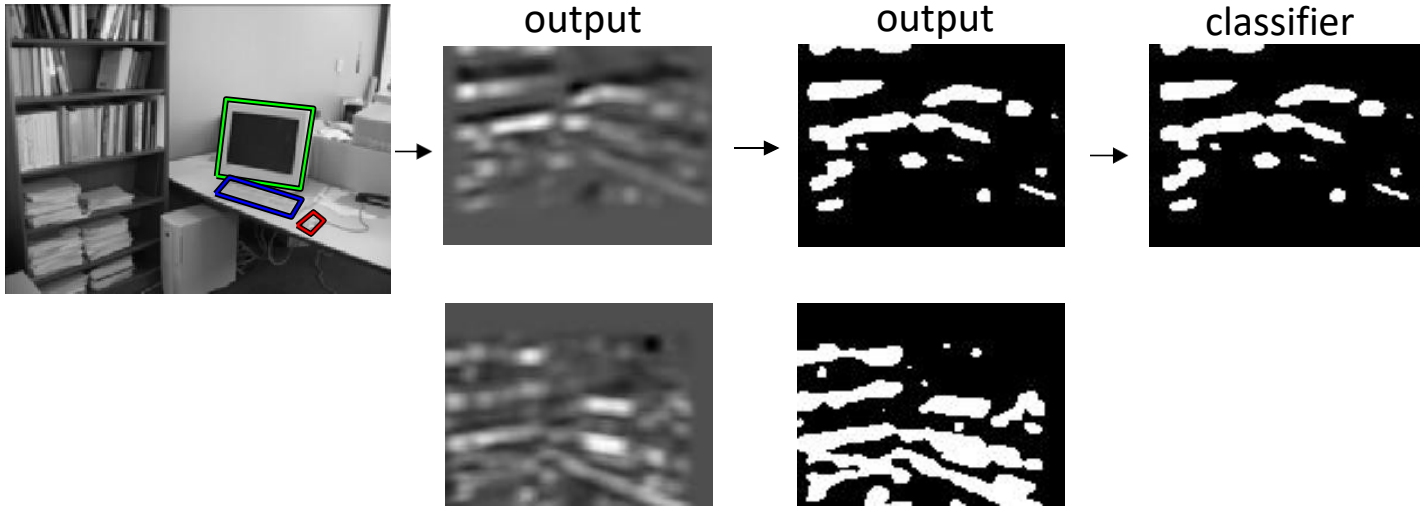
Weak ‘detector’

Produces many false alarms.

Features -> Weak Detectors



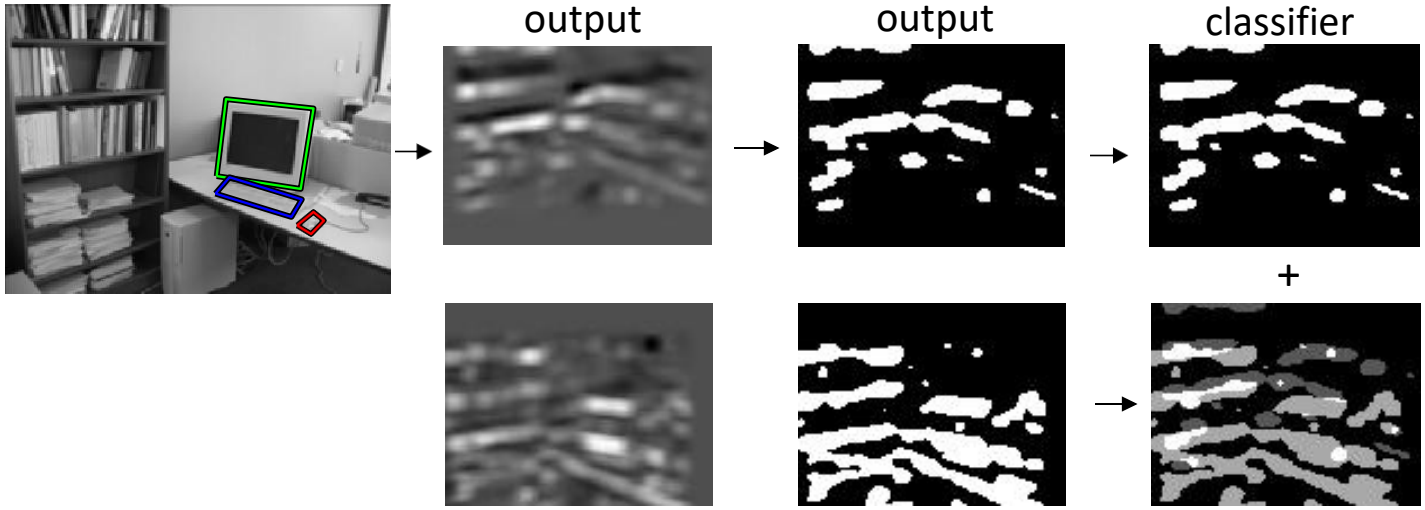
Features -> Weak Detectors



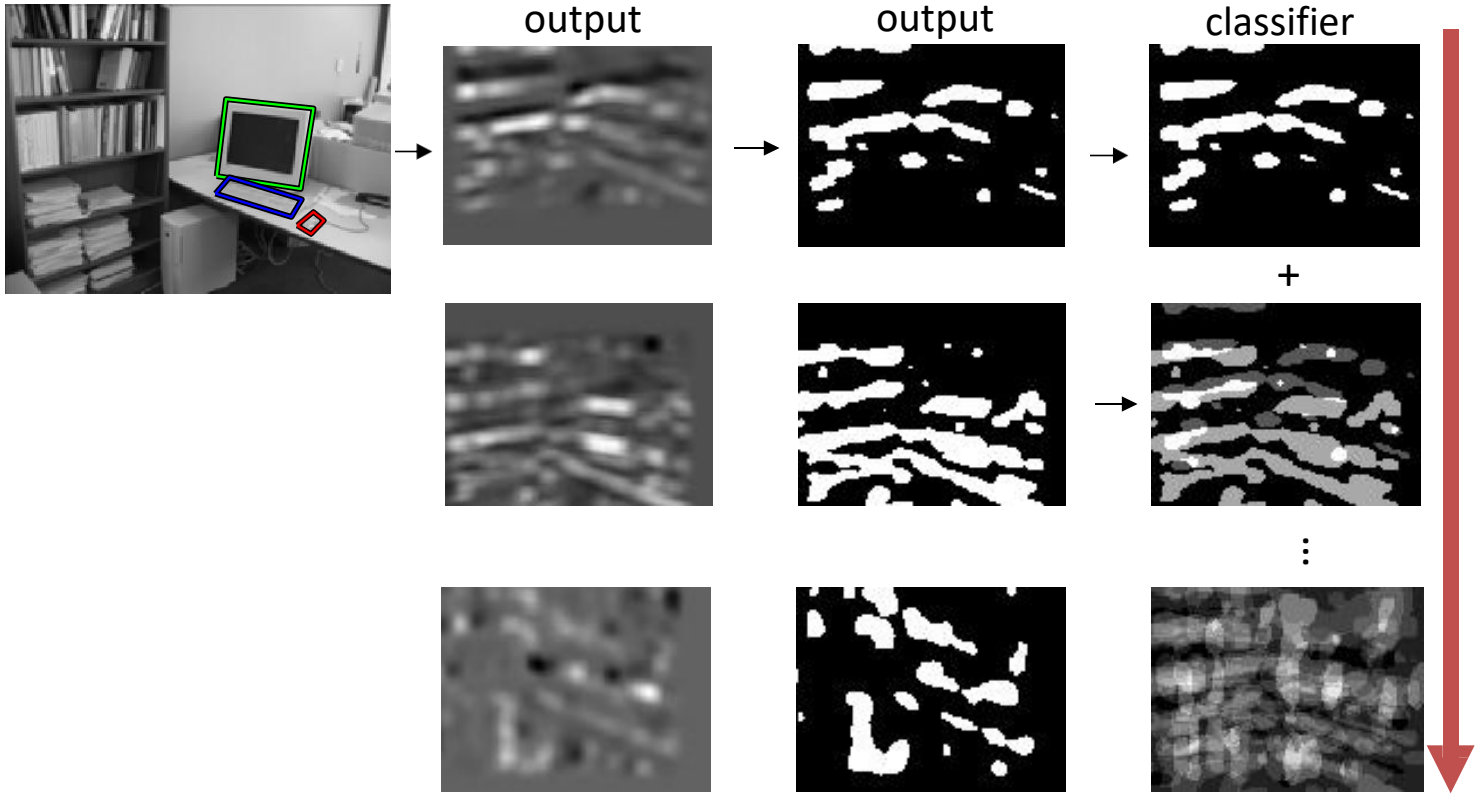
Second weak ‘detector’

Produces a different set of false alarms.

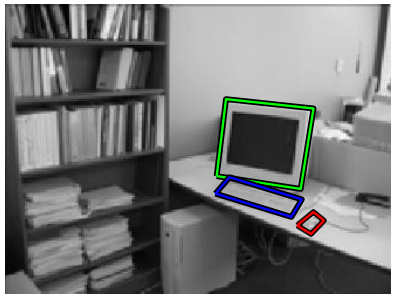
Features -> Weak Detectors



Features -> Weak Detectors



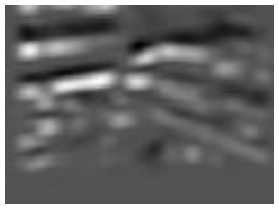
Features -> Weak Detectors



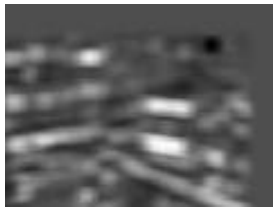
Feature output

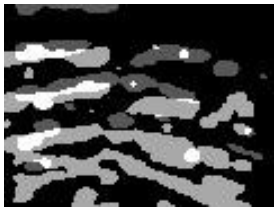
Thresholded output

Strong classifier

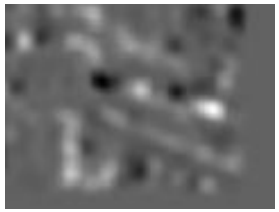


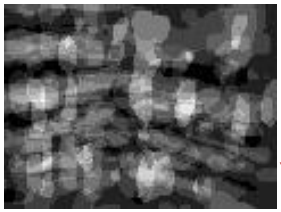
+





⋮

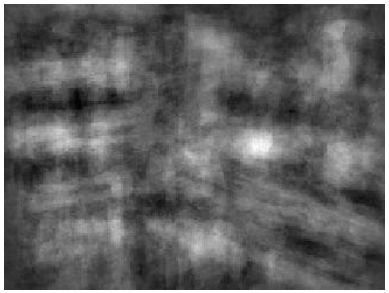


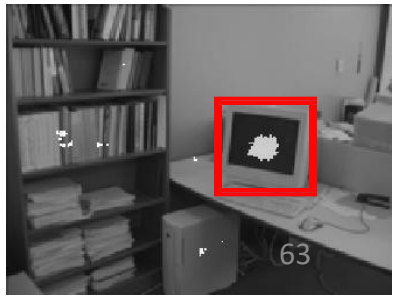


Adding features

Final classification

Strong classifier at iteration 200



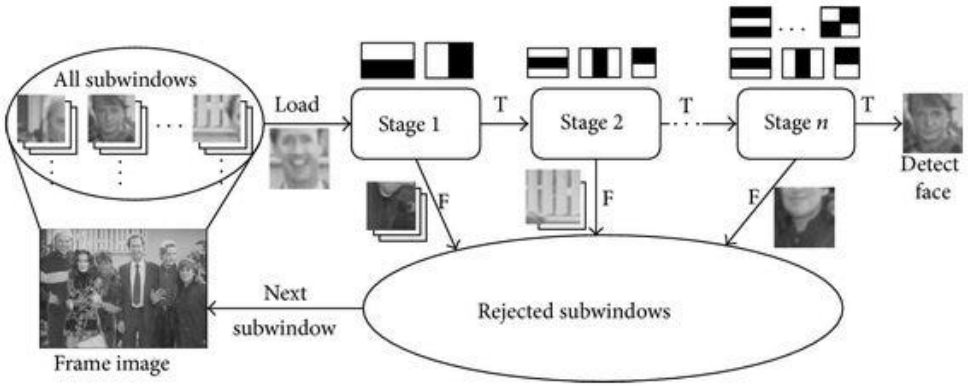


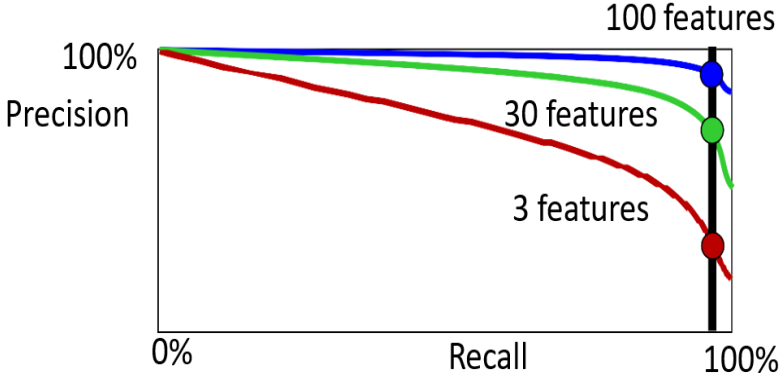
Cascade of classifiers

- Motivation: Some negative (background) samples may be rejected based on only few features
- How?

$$\frac{FP}{FP + TN}$$

Reduce false positive rate





$$\frac{FN}{TP + FN}$$

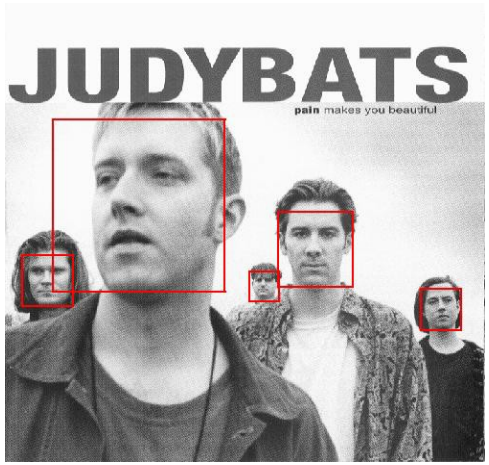
Select a threshold with a very low false negative rate for each stage

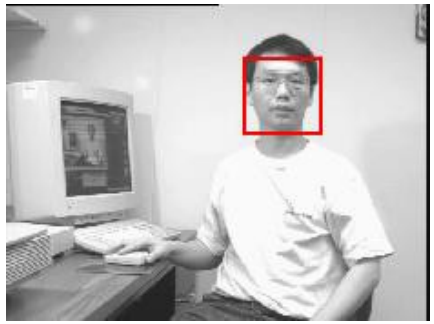
Computational complexity of the 3-features classifier with the performance of the 100-features classifier

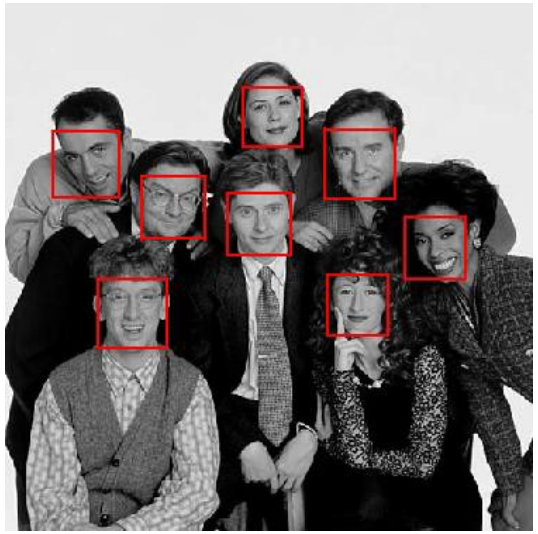
$$\text{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}}$$

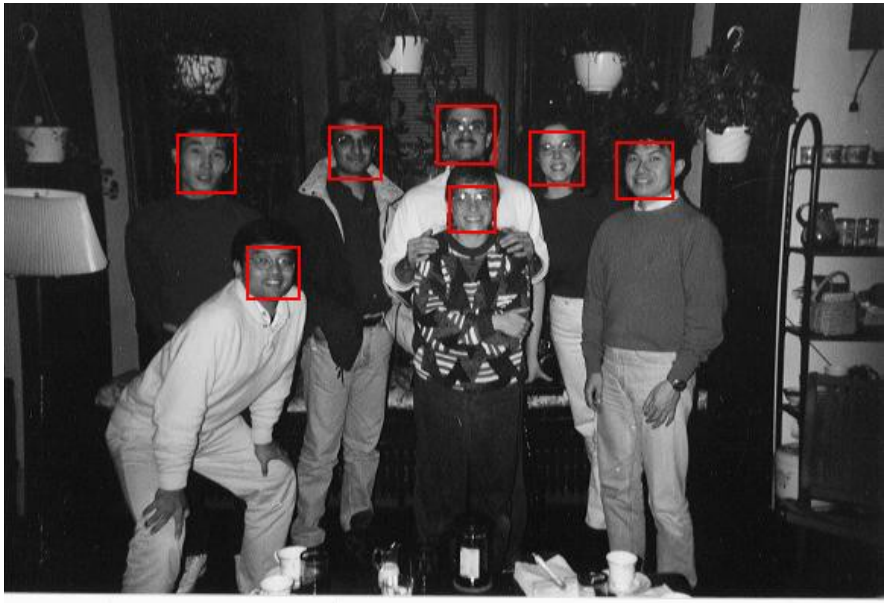
$$\text{Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}}$$

Output of Face Detector on Test Images







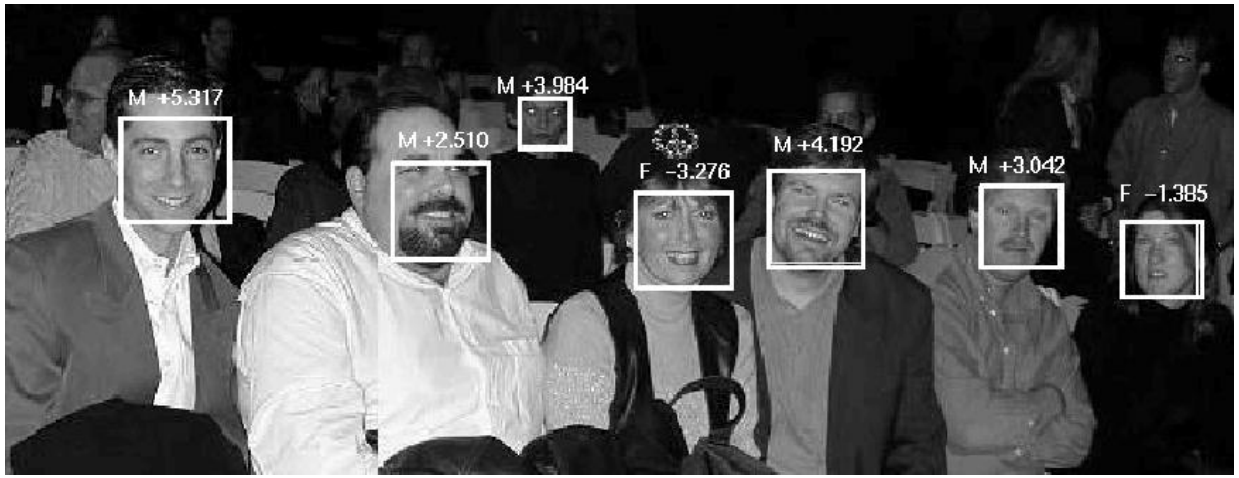


Other detection tasks

Facial Feature Localization

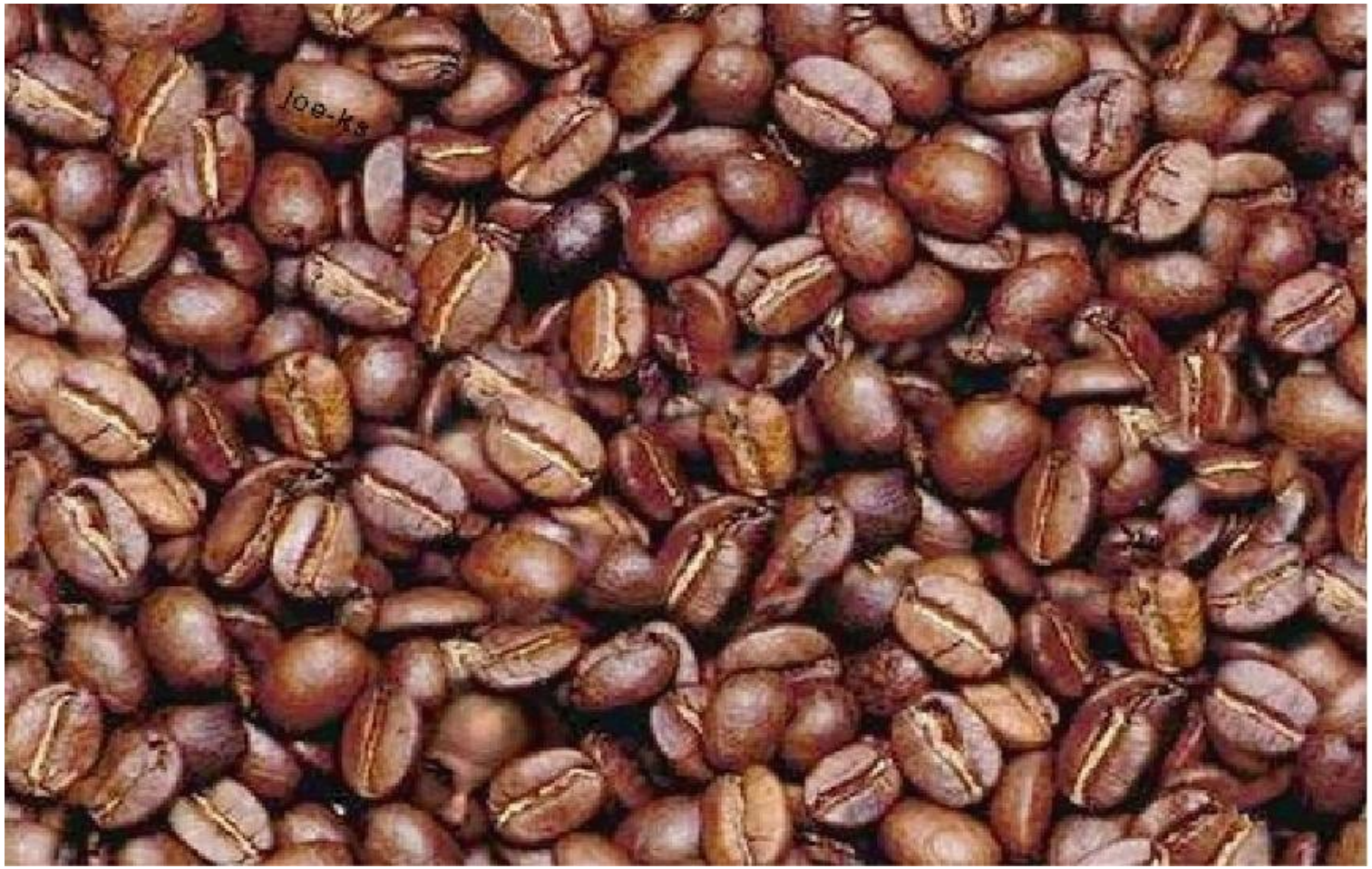
Profile Detection

Male vs. female



“Head in the coffee beans problem”

Can you find the head in this image?



Weakness of Boosting

- Features are extracted at fixed positions, and thus not deformable (not perfect for deformable objects)
- Difficulty in handling severe occlusion
- Extension to “deformable model” + “and/or model”?

Papers to Read and Study

- Friedman, Hastie, Tibshirani.

  Additive Logistic Regression: a Statistical View of Boosting (1998).

- Robert E. Schapire.
  The boosting approach to machine learning: An overview.
  In D. D. Denison, M. H. Hansen, C. Holmes, B. Mallick, B. Yu, editors,
  Nonlinear Estimation and Classification. Springer, 2003.

- Ron Meir and Gunnar Rätsch.
  An introduction to boosting and leveraging.
  In *Advanced Lectures on Machine Learning (LNAI2600)*, 2003.