Pattern Recognition

(EE5907R)

Jiashi FENG

Email: elefjia@nus.edu.sg

Outlines

- Unsupervised Feature Extraction (PCA, NMF,…)
- Supervised Feature Extraction (LDA, GE, …)
- Clustering and Applications
- **Gaussian Mixture Model and Boosting**
- Support Vector Machine
- Deep Learning

Generative vs. Discriminative

- We want to classify the data x into labels y. A generative model learns the joint probability distribution $p(x, y)$ and a discriminative model learns the conditional probability distribution $p(y|x)$.
- Suppose we have the following data in the form (x, y): (1,0), (1,0), (2,0), (2, 1)
- $p(x, y)$ is

	y = 0	y = 1
x = 1	½	0
x = 2	¼	¼

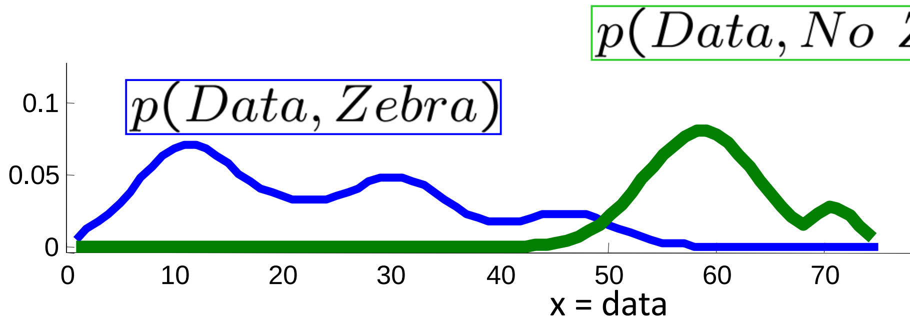
- $p(y|x)$ is

	y = 0	y = 1
x = 1	1	0
x = 2	½	½

Generative vs. Discriminative

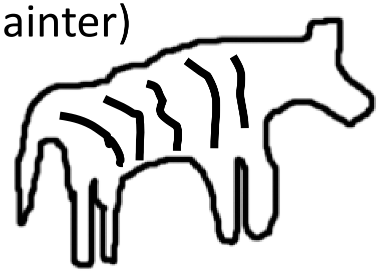
- **Generative model**

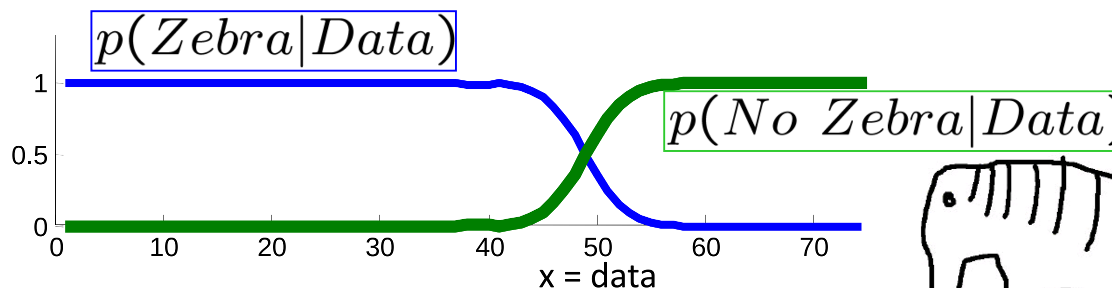
(The artist)

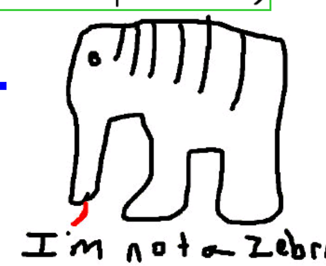


- **Discriminative model**

(The lousy painter)

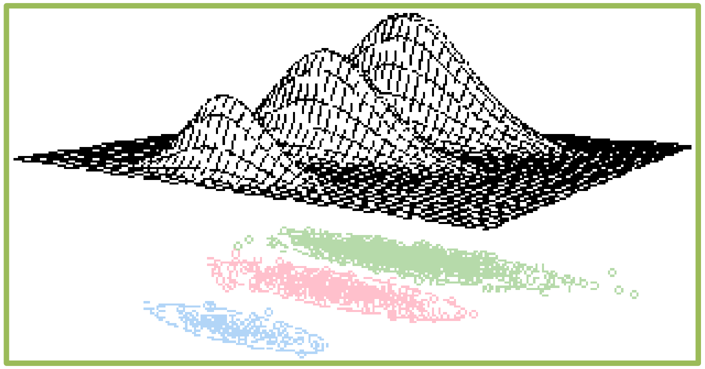




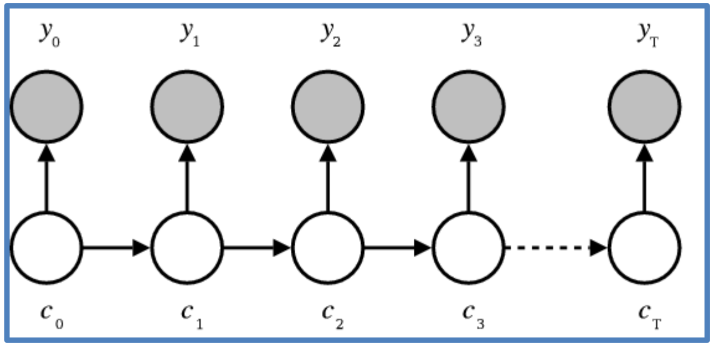


Generative Models

- Gaussian Mixture Model and other mixture model



- Hidden Markov Model



Discriminative Models

Nearest neighbor

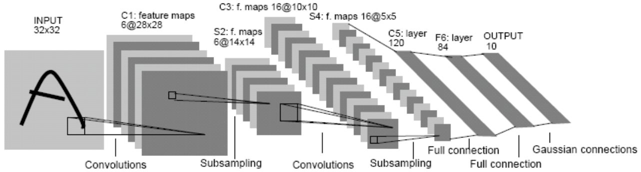
10^6 examples

Shakhnarovich, Viola, Darrell 2003

Berg, Berg, Malik 2005

…

Neural Networks

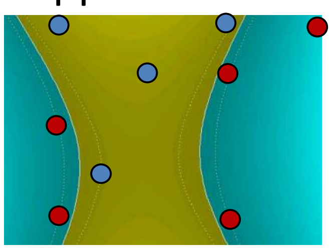


LeCun, Bottou, Bengio, Haffner 1998

Rowley, Baluja, Kanade 1998

…

Support Vector Machines

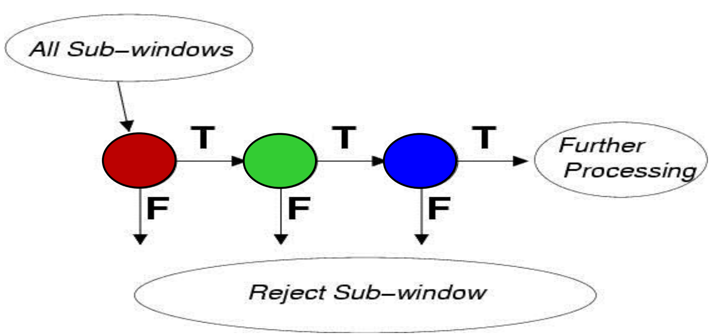


Guyon, Vapnik

Heisele, Serre, Poggio, 2001

…

Boosting



Generative: Gaussian Mixture Model (GMM)

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

$$p(x) = \pi_1 f_1(x) + \pi_2 f_2(x) + \ldots + \pi_K f_K(x)$$

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

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

Gaussian Mixture Models

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

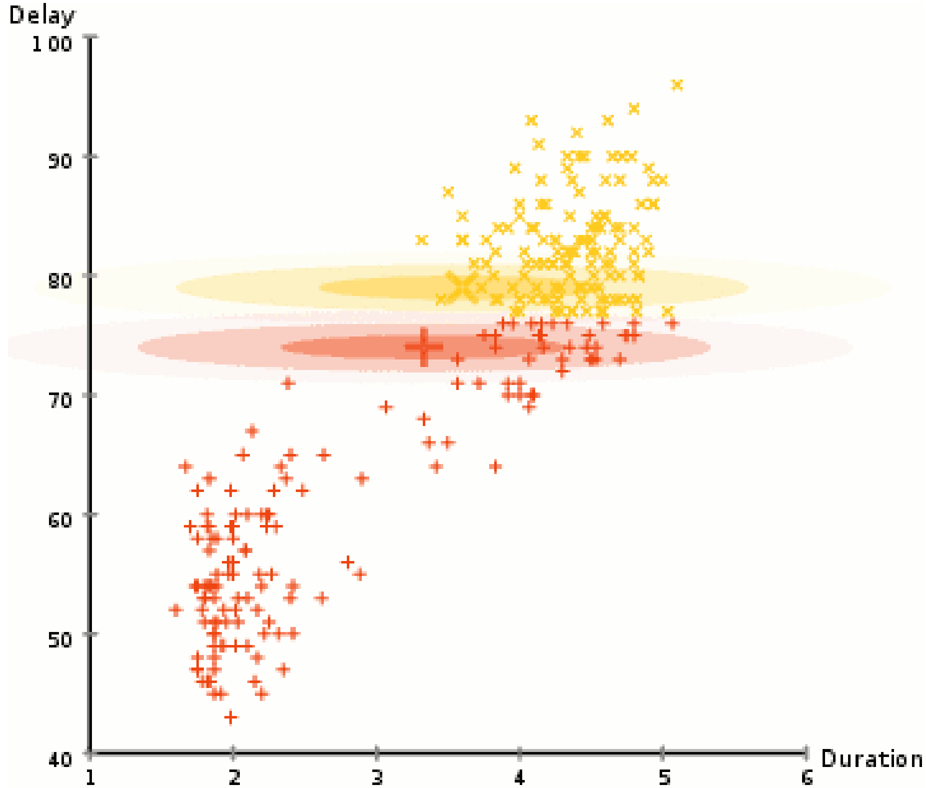
$$p(x) = \pi_1 N(x|\mu_1, \Sigma_1) + \pi_2 N(x|\mu_2, \Sigma_2) + \ldots + \pi_K N(x|\mu_K, \Sigma_K)$$

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

$$p(x) = \sum_{i=1}^{K} \pi_i N(x|\mu_i, \Sigma_i)$$

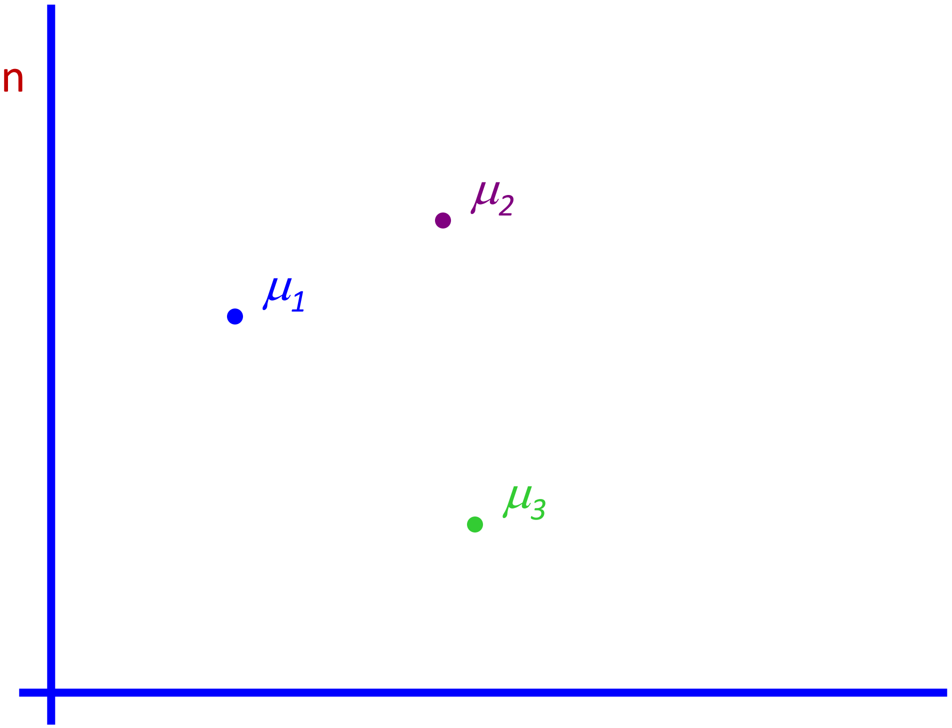
Gaussian Mixture Models

- Rather than identifying clusters by “nearest” centroids
- Fit a Set of K Gaussians to the **unlabeled** data
- Maximum Likelihood over a mixture model



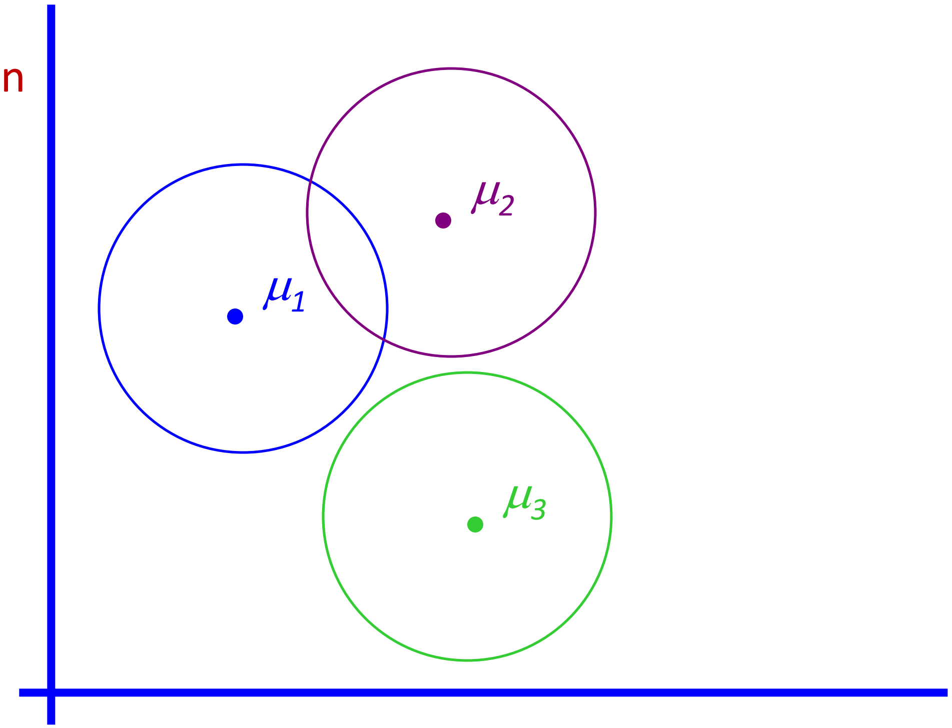
The GMM Assumption

- There are K components. The i'th component is called ω_i
- Component ω_i has an associated mean vector μ_i



The GMM Assumption

- There are K components. The i'th component is called ω_i
- Component ω_i has an associated mean vector μ_i
- Each component generates data from a Gaussian model with mean μ_i and covariance matrix $\sigma^2 \boldsymbol{I}$

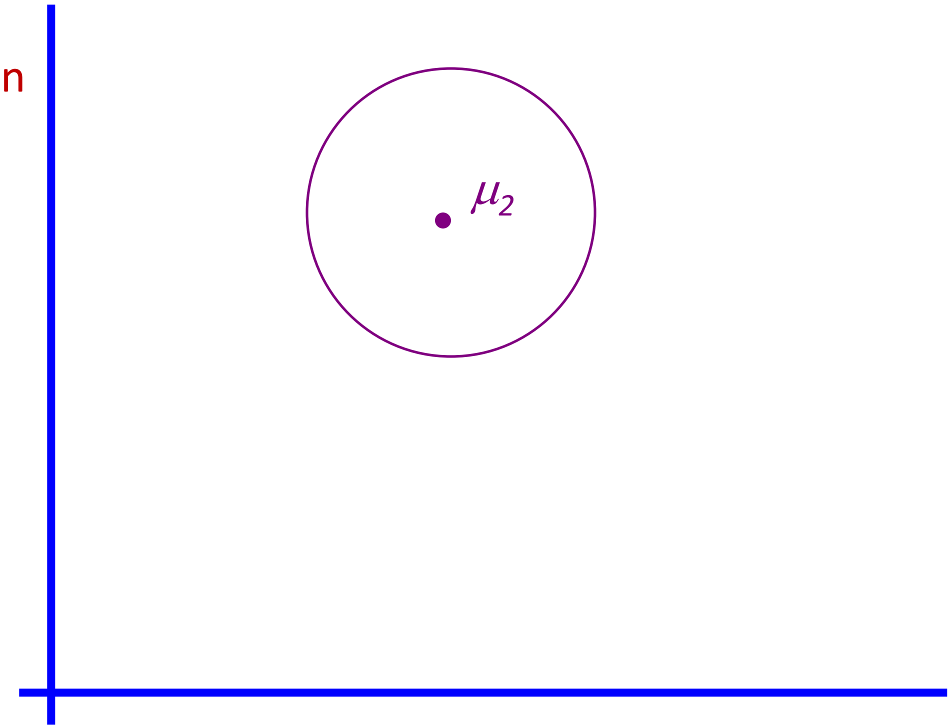


The GMM Assumption

- There are K components. The i'th component is called ω_i
- Component ω_i has an associated mean vector μ_i
- Each component generates data from a Gaussian model with mean μ_i and covariance matrix $\sigma^2 \mathbf{I}$

Assume that each data point is generated according to the following recipe:

1. Pick a component at random. Choose component i with probability $P(\omega_i)$.

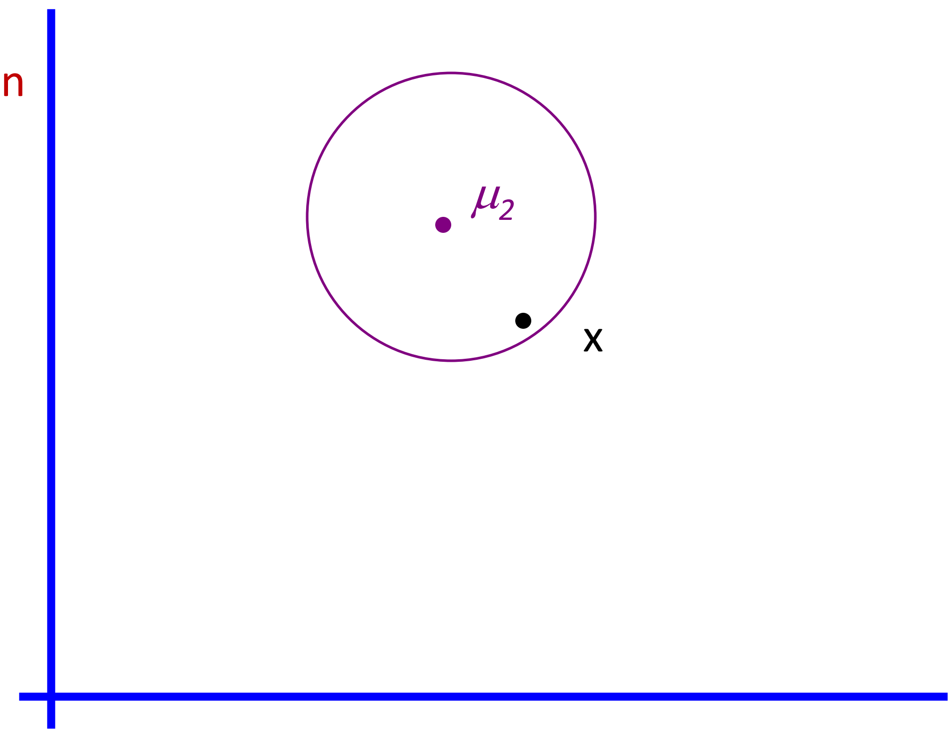


The GMM Assumption

- There are K components. The i'th component is called ω_i
- Component ω_i has an associated mean vector μ_i
- Each component generates data from a Gaussian model with mean μ_i and covariance matrix $\sigma^2 \mathbf{I}$

Assume that each data point is generated according to the following recipe:

1. Pick a component at random. Choose component *i* with probability $P(\omega_i)$.
2. Data point ~ $N(\mu_i, \sigma^2 \mathbf{I})$

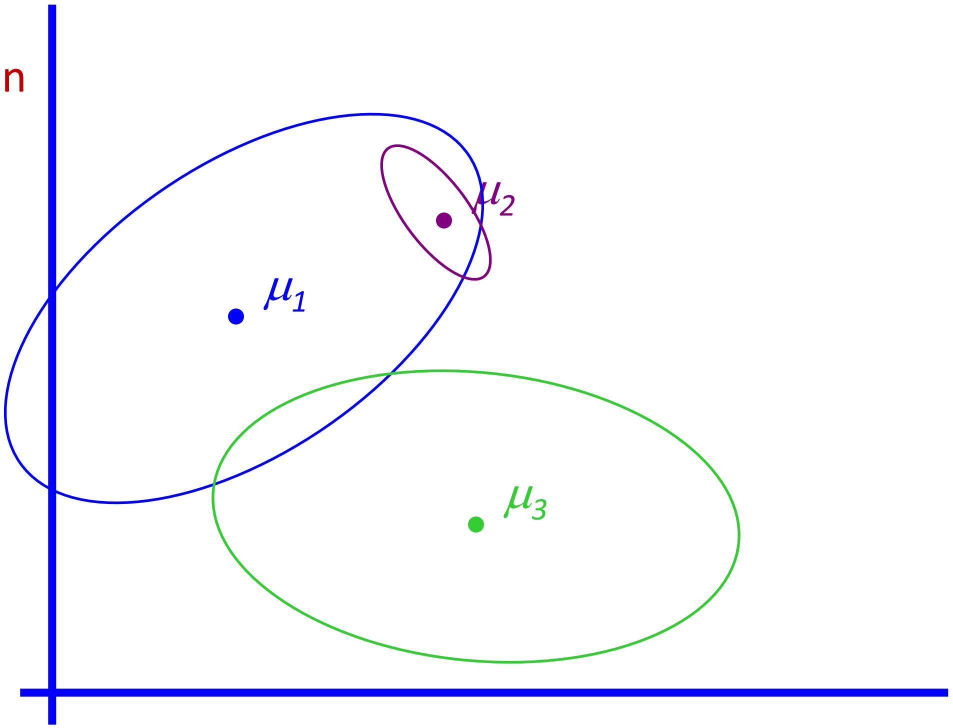


The General GMM Assumption

- There are K components. The i'th component is called ω_i
- Component ω_i has an associated mean vector μ_i
- Each component generates data from a Gaussian model with mean μ_i and covariance matrix Σ_i

Assume that each data point is generated according to the following recipe:

1. Pick a component at random. Choose component i with probability $P(\omega_i)$.
2. Data point ~ $N(\mu_i, \Sigma_i)$



The EM Algorithm

- **Expectation-maximization** (**EM**) is a method for finding maximum likelihood (or maximum a posteriori) estimate of parameter(s) in statistical model, where the model depends on **unobserved** latent variables.

Latent variables are the key properties for EM.

The EM Algorithm

- **EM** is an **iterative** method which alternates between performing an **E**xpectation (**E**) step and a **M**aximization (**M**) step
  - **E-step** computes the expectation of the log-likelihood evaluated using the current estimated distributions for the latent variables based on the parameters inferred from previous step
  - **M-step** computes parameters maximizing the expected log-likelihood from the E-step. These parameter-estimates are then used to determine the distribution of the latent variables in the next E-step.

Latent variables become constants here.

Simple Example

Let events be “grades in a class”

w_1 = Gets an A P(A) = ½

w_2 = Gets a B P(B) = μ

w_3 = Gets a C P(C) = 2μ

w_4 = Gets a D P(D) = ½-3μ

(Note $0 \le \mu \le 1/6$)

Assume we want to estimate μ from data. In a given class, there were

a A’s
b B’s
c C’s
d D’s

What’s the maximum likelihood estimate of μ given a, b, c, d?

Trivial Statistics

P(A) = ½ P(B) = μ P(C) = 2μ P(D) = ½-3μ

$P(a, b, c, d \mid \mu) = K(\tfrac{1}{2})^a(\mu)^b(2\mu)^c(\tfrac{1}{2}-3\mu)^d$

$\log P(a, b, c, d \mid \mu) = \log K + a\log \tfrac{1}{2} + b\log \mu + c\log 2\mu + d\log (\tfrac{1}{2}-3\mu)$

$$\text{FOR MAX LIKE } \mu\text{, SET } \frac{\partial \text{LogP}}{\partial \mu} = 0$$

$$\text{K}=\frac{(a+b+c+d)!}{a!b!c!d!}$$

$$\frac{\partial \text{LogP}}{\partial \mu} = \frac{b}{\mu} + \frac{2c}{2\mu} - \frac{3d}{1/2-3\mu} = 0$$

Gives max like $\boxed{\mu = \frac{b+c}{6(b+c+d)}}$

So if class got

A	B	C	D
14	6	9	10

$$\text{Max like } \mu = \frac{1}{10}$$

Same Problem with Latent Information

REMEMBER
P(A) = ½
P(B) = μ
P(C) = 2μ
P(D) = ½-3μ

Someone tells us that

Number of High grades (A's + B's) = *h*

Number of C's = *c*

Number of D's = *d*

What is the max likelihood estimate of μ now?

$$\log P(h, c, d \mid \mu, \underbrace{b}_{\text{latent variable}}) = \log K(h-b,b,c,d) + (h-b)\log \tfrac{1}{2} + b\log\mu + c\log 2\mu + d\log(\tfrac{1}{2}-3\mu)$$

Same Problem with Latent Information

REMEMBER
P(A) = ½
P(B) = μ
P(C) = 2μ
P(D) = ½-3μ

Someone tells us that

Number of High grades (A's + B's) = *h*

Number of C's = *c*

Number of D's = *d*

What is the max likelihood estimate of μ now?

We can answer this question circularly:

EXPECTATION

If we know the value of μ we could compute the expected value of *b*

Since the ratio a:b should be the same as the ratio ½ : μ

$$E_\mu(b) = \frac{\mu}{\frac{1}{2}+\mu} h$$

MAXIMIZATION

If we know the expected values of *b* we could compute the maximum likelihood value of μ

$$\mu = \frac{E_\mu(b)+c}{6\left(E_\mu(b)+c+d\right)}$$

Already computed as in slide #19

EM for This Problem

REMEMBER
P(A) = ½
P(B) = μ
P(C) = 2μ
P(D) = ½-3μ

We begin with a guess for μ

We iterate between **EXPECTATION** and **MAXIMIZATION** to improve our estimates of b and μ.

Define $\mu(t)$ the estimate of μ on the t'th iteration

$b(t)$ the estimate of b on t'th iteration

$$\mu(0) = \text{initial guess}$$

$$b(t) = \frac{\mu(t)h}{\frac{1}{2} + \mu(t)} = \mathrm{E}\left[b \mid \mu(t)\right]$$

E-step

$$\mu(t+1) = \frac{b(t)+c}{6\left(b(t)+c+d\right)}$$

$$= \text{max like est of } \mu \text{ given } b(t)$$

M-step

Continue iterating until converged.
Good news: Converging to local optimum is assured.
Bad news: We have "local" optimum.

EM Convergence

- Convergence proof based on fact that Prob(data | μ) must increase or remain same between each iteration [NOT OBVIOUS, BUT NOT STUDY HERE]
- But it can never exceed 1
- So it must therefore converge

In our example, suppose we had

$h = 20$

$c = 10$

$d = 10$

$\mu(0) = 0$

t	μ(t)	b(t)
0	0	0
1	0.0833	2.857
2	0.0937	3.158
3	0.0947	3.185
4	0.0948	3.187
5	0.0948	3.187
6	0.0948	3.187

Back to Learning of GMM

Remember:

We have unlabeled data $\boldsymbol{x}_1\ \boldsymbol{x}_2\ \dots\ \boldsymbol{x}_N$

We know there are K components

We know $P(\omega_1)\ P(\omega_2)\ P(\omega_3)\ \dots\ P(\omega_k)$, σ

We **<u>don't</u>** know $\boldsymbol{\mu}_1\ \boldsymbol{\mu}_2 \ldots \boldsymbol{\mu}_k$

Hidden variables z_k^n, indicating which component k the datum n is sampled from

Compute Likelihood

- We define:

$$\pi_i = P(\omega_i) \quad \text{where} \quad \sum_i \pi_i = 1$$

$$z_i = p(\omega_i|x) = \frac{P(\omega_i)p(x|\omega_i)}{\sum_{j=1}^{K} P(\omega_j)p(x|\omega_j)}$$

$$z_k^n = p(\omega_k|x_n)$$

- Identify a likelihood function

$$p(x_1, \dots, x_N|\pi, \mu) = \prod_{n=1}^{N} p(x_n|\pi, \mu)$$

x_n's were drawn independently

$$= \prod_{n=1}^{N} \sum_{k=1}^{K} p(x_n|\omega_k, \mu_k)P(\omega_k)$$

Maximum Likelihood over a GMM

- Identify a log-likelihood function

$$\ln p(x_1, \dots, x_n|\pi, \mu) = \sum_{n=1}^{N} \ln\left[\sum_{k=1}^{K} p(x_n|\omega_k, \mu_k)P(\omega_k)\right]$$

- Compute and set partials to 0

$$\frac{\partial \ln p(x_1, \dots, x_n|\pi, \mu)}{\partial \mu_k} = \sum_{n=1}^{N} \frac{1}{p(x_n|\pi, \mu)} \frac{\partial \sum_{k=1}^{K} N(x_n|\mu_k)P(\omega_k)}{\partial \mu_k}$$

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

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

$$p(x_n|\pi, \mu) = \sum_{k=1}^{K} p(x_n|\omega_k, \mu_k)P(\omega_k)$$

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

see next slide

Maximum Likelihood over a GMM

$$\frac{\partial \ln p(x_1, \dots, x_n | \pi, \mu)}{\partial \mu_k} = \sum_{n=1}^{N} p(\omega_k | x_n) \frac{\partial \ln \exp\left(-\frac{1}{2\sigma^2}(x_n - \mu_k)^2\right)}{\partial \mu_k}$$

$$= \sum_{n=1}^{N} z_k^n \left(-\frac{1}{2\sigma^2} \frac{\partial (x_n - \mu_k)^2}{\partial \mu_k}\right)$$

$$= \sum_{n=1}^{N} z_k^n \frac{x_n - \mu_k}{\sigma^2} = 0$$

set partials to 0

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

EM for General GMMs

We don't know $P(\omega_1), P(\omega_2), \dots, P(\omega_K), \mu_1, \mu_2, \dots, \mu_{\mathrm{K}}, \Sigma_1, \Sigma_2, \dots, \Sigma_K$

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

$$\mu_k = \frac{\sum_{n=1}^{N} z_k^n x_n}{\sum_{n=1}^{N} z_k^n}$$

$$\Sigma_k = \frac{\sum_{n=1}^{N} z_k^n (x_n - \mu_k)(x_n - \mu_k)^T}{\sum_{n=1}^{N} z_k^n}$$

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

Summary: EM for GMMs

- Initialize the parameters
  - Evaluate the log likelihood

- Expectation-step: Compute the expectation

- Maximization-step: Re-estimate Parameters
  - Evaluate the log likelihood
  - Check for convergence

EM for GMMs

- E-step: Compute "expected" classes of all data points for each class

$$z_k^n = \frac{\pi_k N(x_n|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j N(x_n|\mu_j, \Sigma_j)}$$

where $\pi_k = p(\omega_k)$

EM for GMMs

- M-Step: Re-estimate Parameters

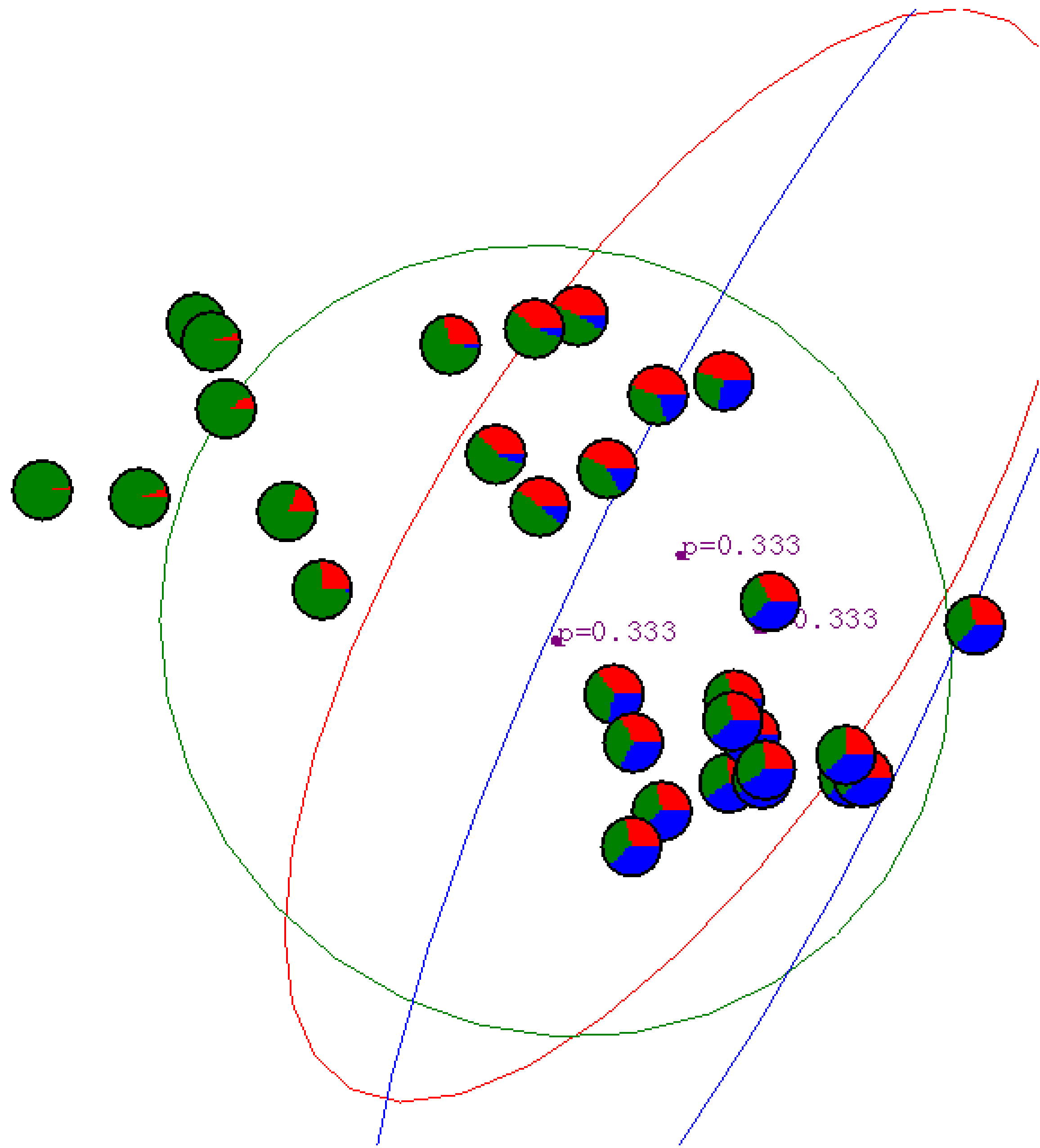
$$\mu_k^{new} = \frac{\sum_{n=1}^{N} z_k^n x_n}{\sum_{n=1}^{N} z_k^n}$$

$$\Sigma_k^{new} = \frac{\sum_{n=1}^{N} z_k^n (x_n - \mu_k^{new})(x_n - \mu_k^{new})^T}{\sum_{n=1}^{N} z_k^n}$$

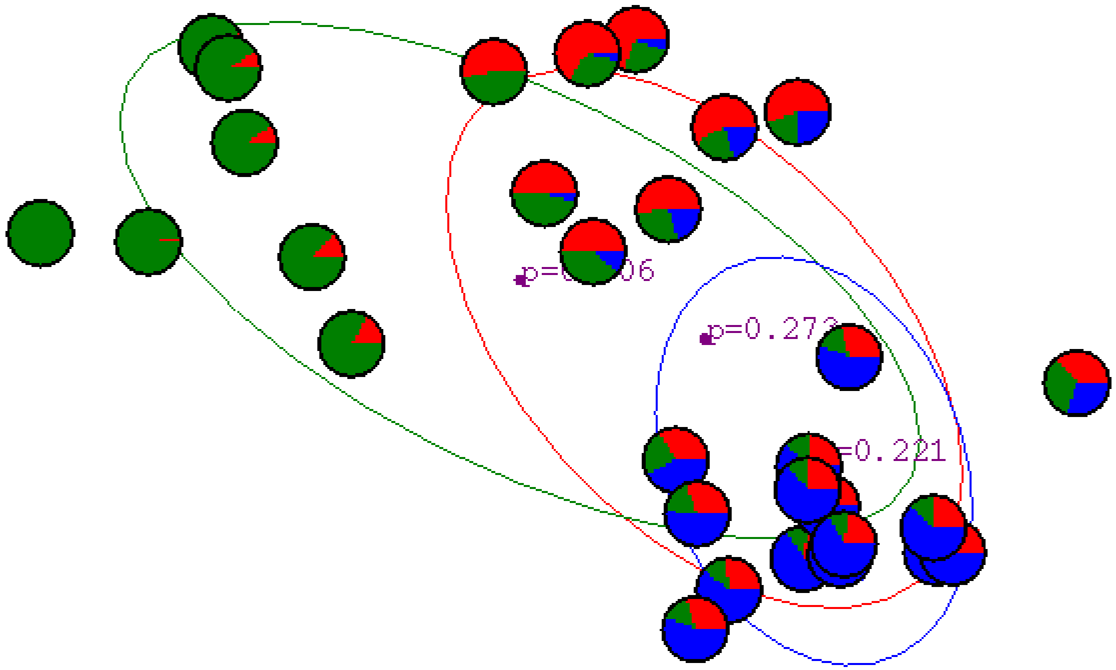
$$\pi_k^{new} = p(\omega_k)^{new} = \frac{\sum_{n=1}^{N} z_k^n}{N}$$

Latent variables become constants here.

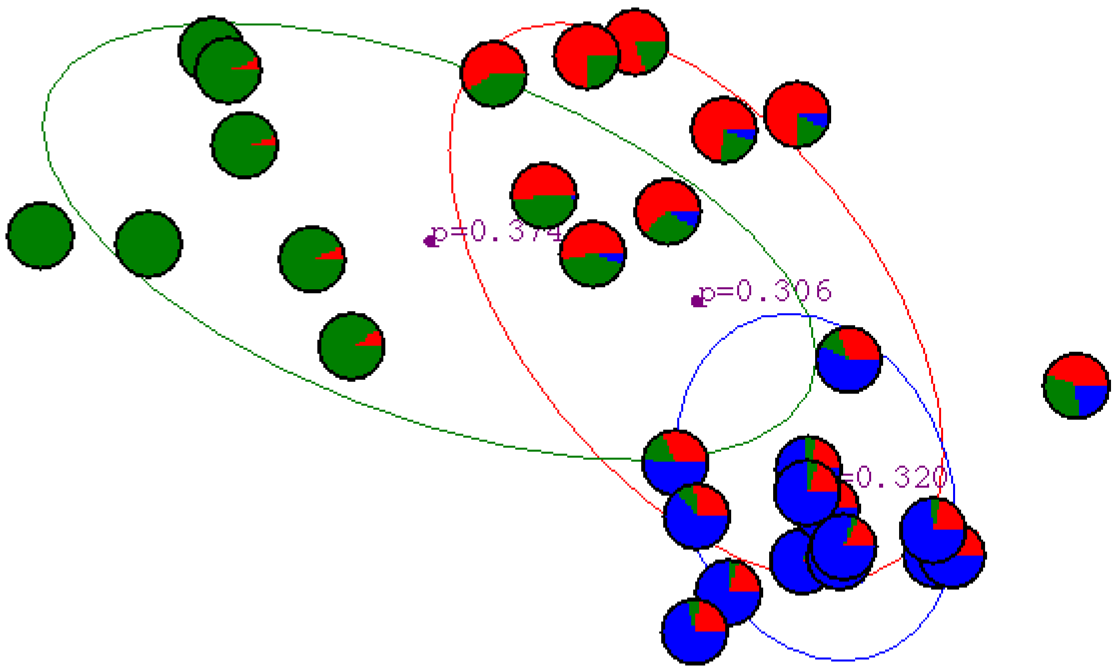
Gaussian Mixture Model Example: Start



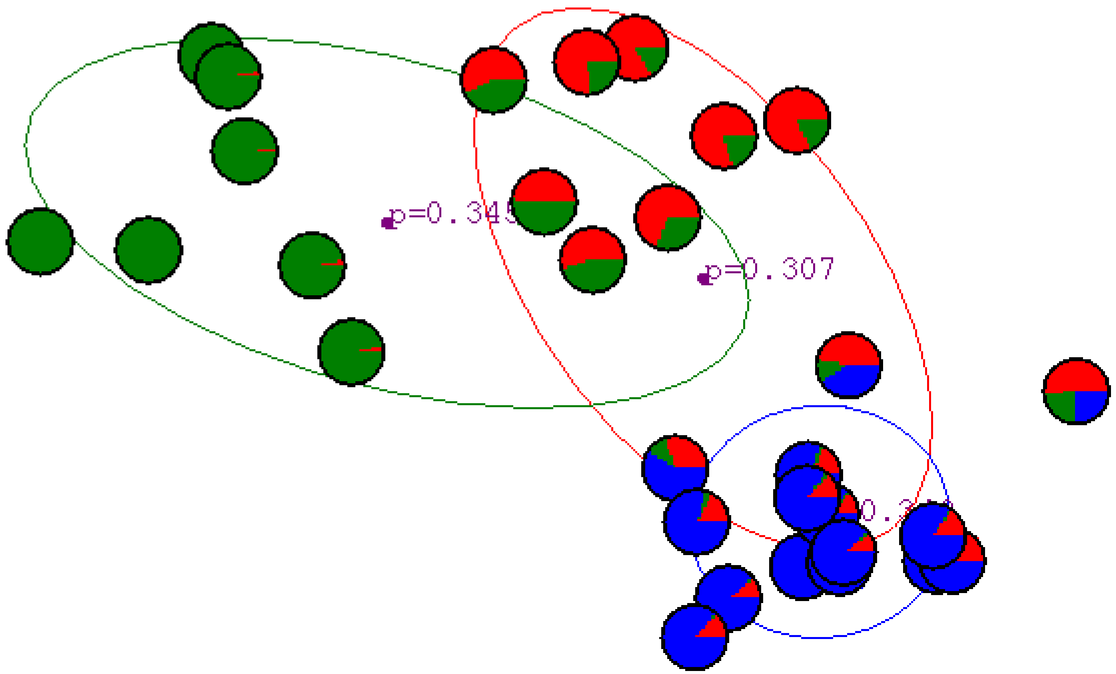
After 1st iteration



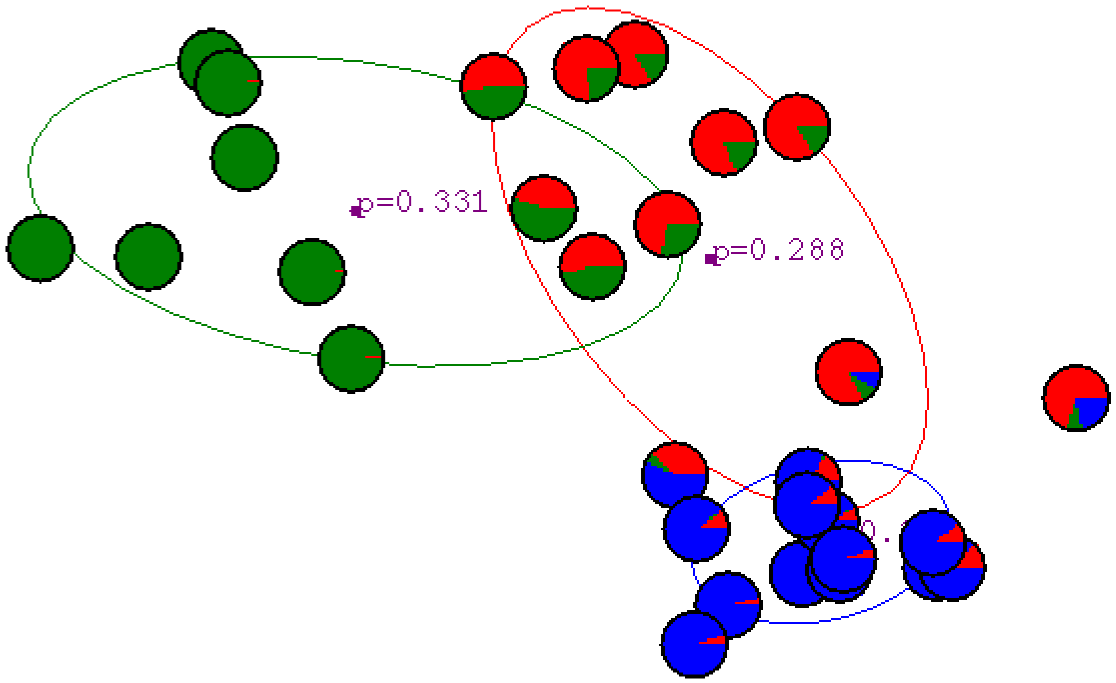
After 2nd iteration



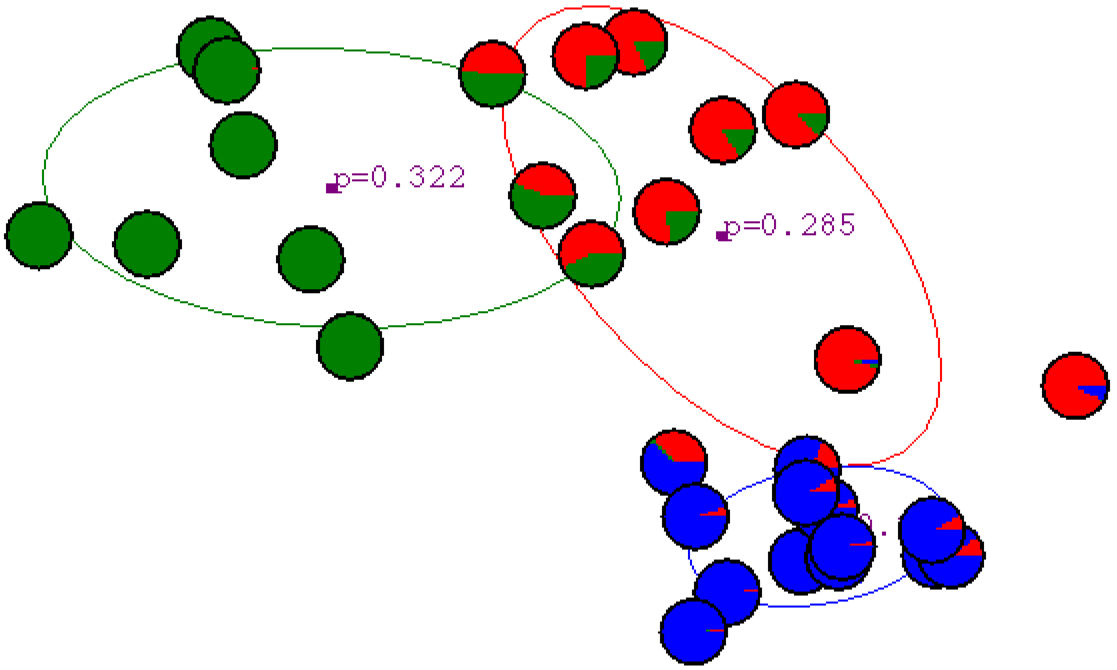
After 3rd iteration



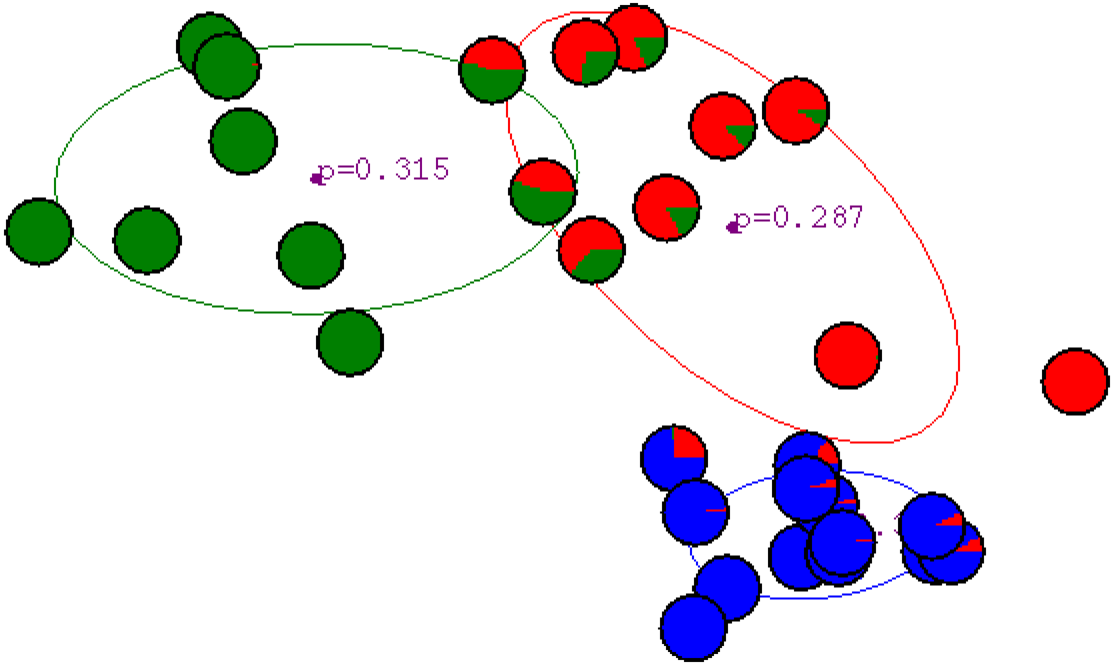
After 4th iteration



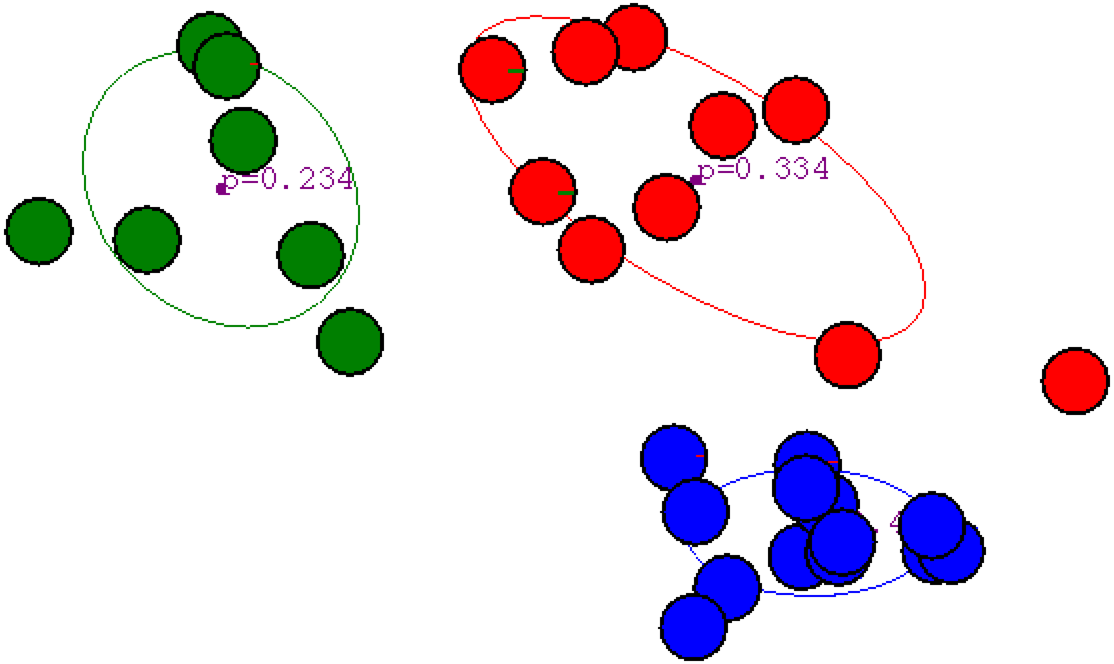
After 5th iteration



After 6th iteration



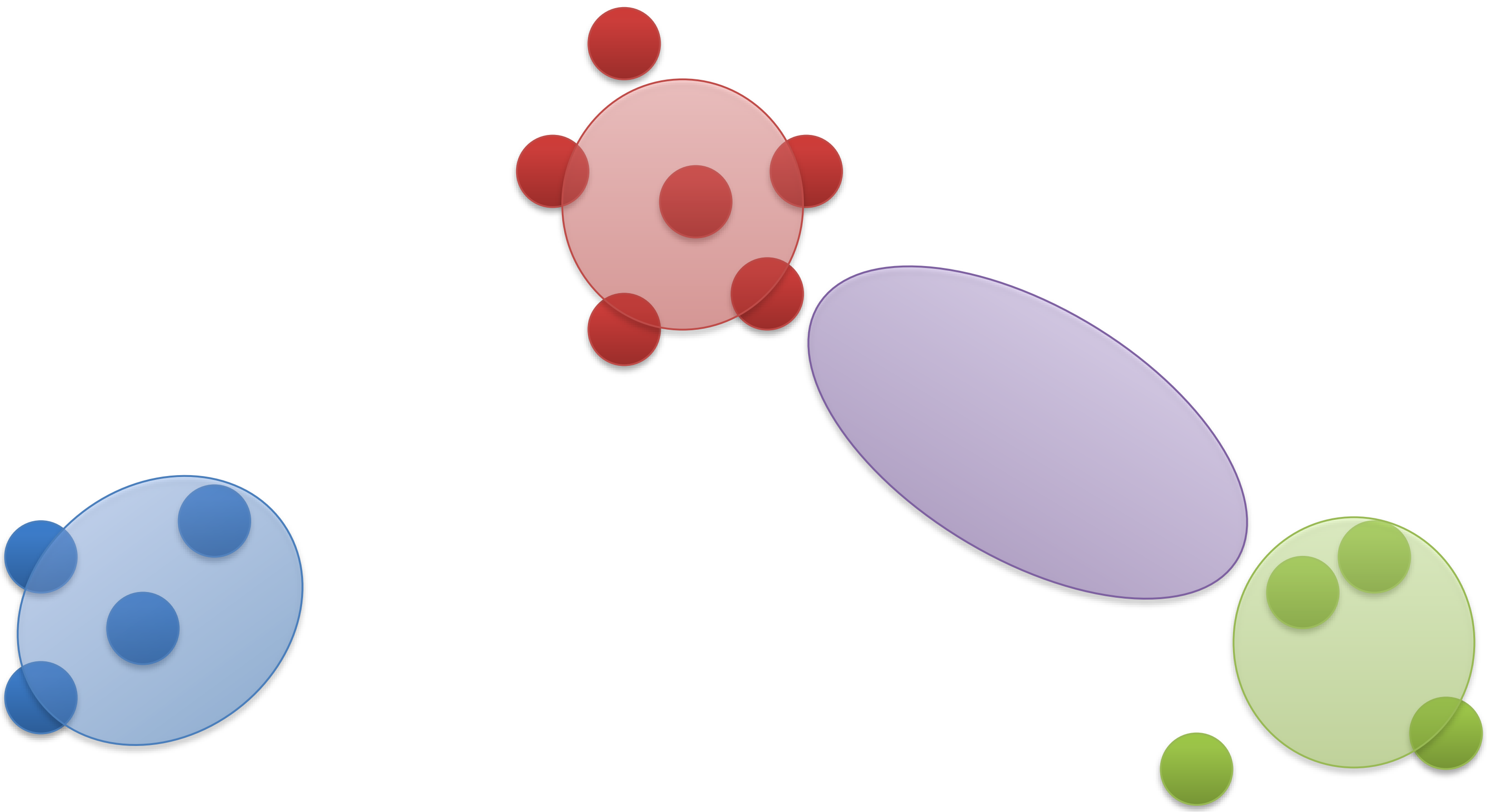
After 20th iteration



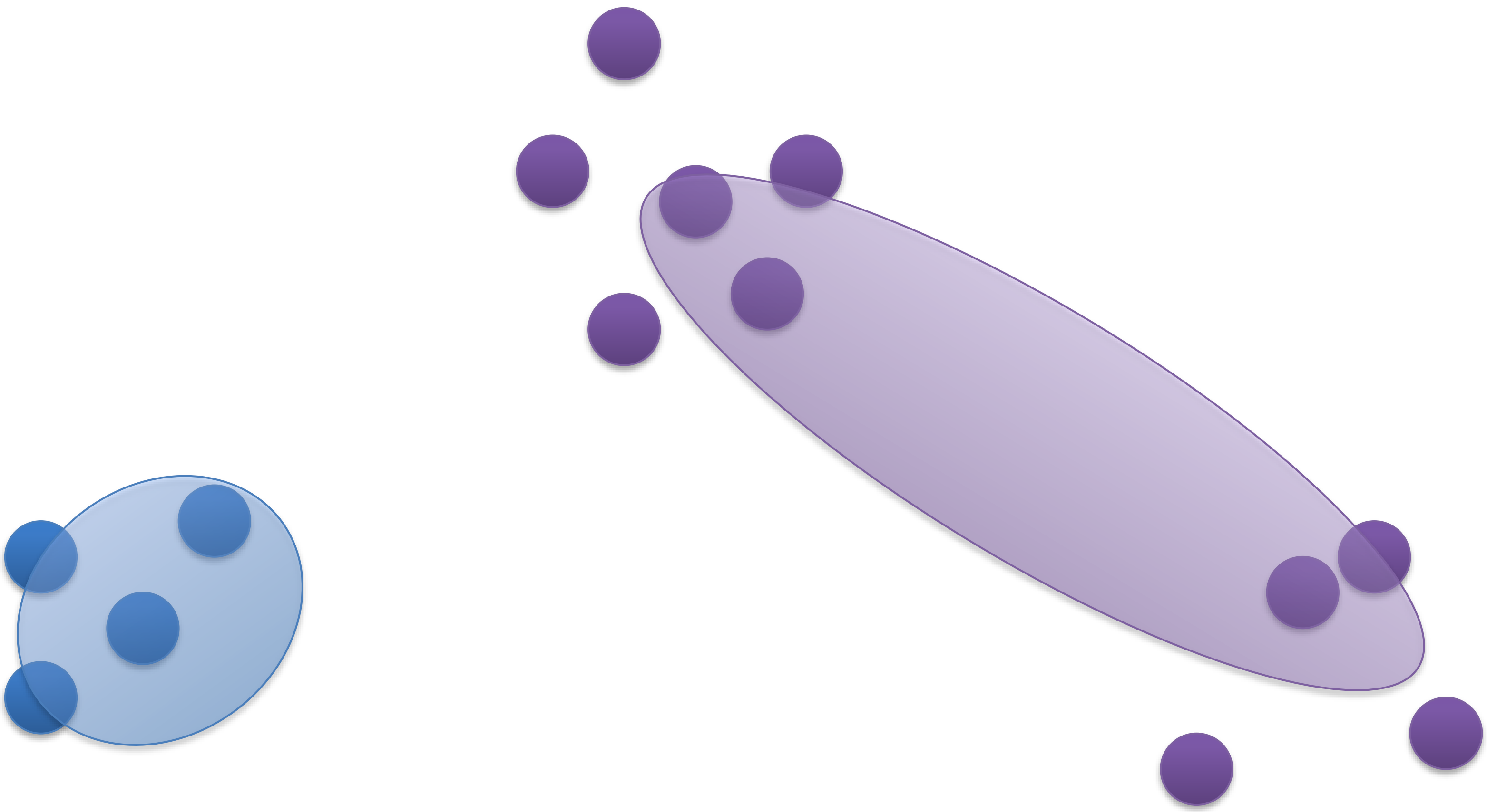
Relationship to K-means

- K-means makes **hard** decisions.
  - Each data point gets assigned to a single cluster.
- GMM makes **soft** decisions.
  - Each data point yields a posterior
- Potential problem:
  - Incorrect number of Mixture Components

Incorrect Number of Gaussians



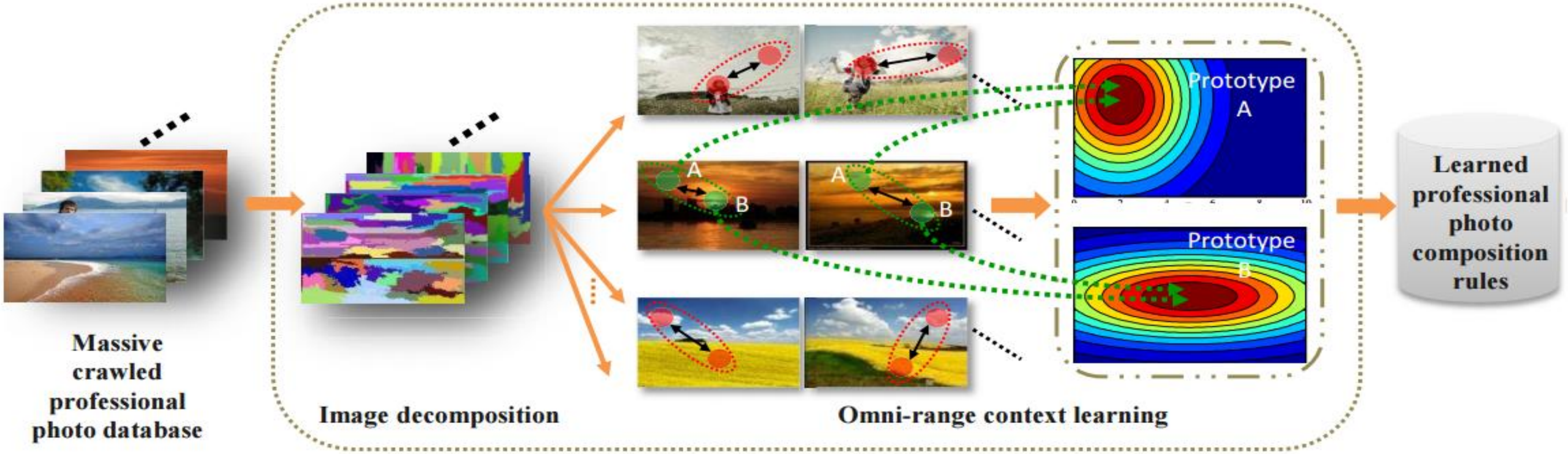
Incorrect Number of Gaussians

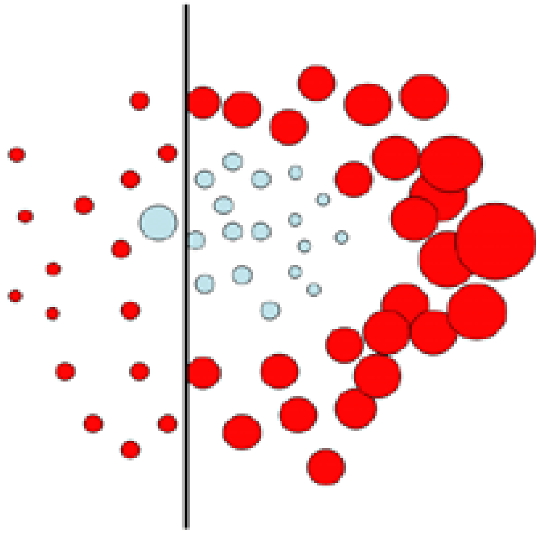


GMM for Classification

- Train universal GMM, and then adapt it for individual class, and finally do classification
  - Widely used in speech recognition
- Note that we can initiate GMM by using K-means

Another Application





Discriminative: Boosting

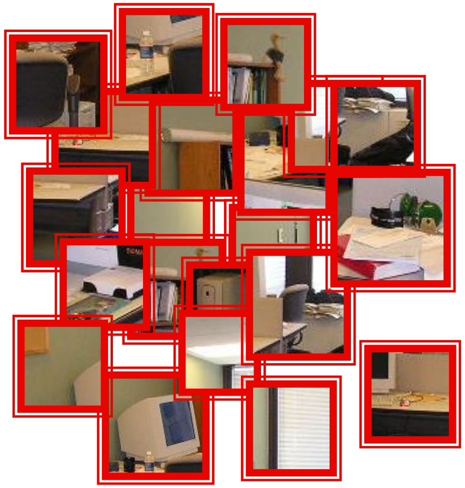
Example Task

Object detection and recognition is formulated as a classification problem.

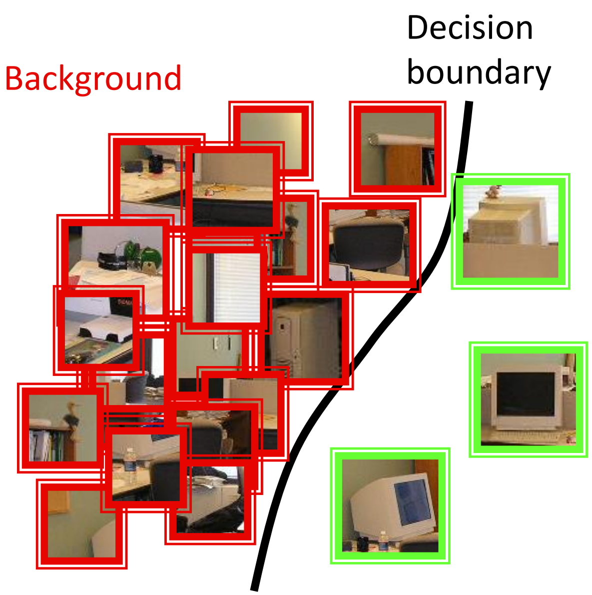
The image is partitioned into a set of overlapping windows

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

Where are the screens?



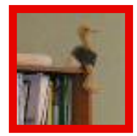
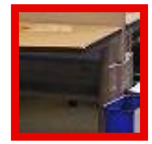
Bag of image patches



In some feature space

Formulation

- Formulation: binary classification

   …    

Features x =	x_1	x_2	x_3	…	x_N	x_{N+1}	x_{N+2}	…	x_{N+M}
Labels y =	-1	+1	-1		-1	?	?		?

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

Test data

- Classification function

$\hat{y} = F(x)$ where $F(x)$ belongs to some family of functions

- Minimize misclassification error

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

- Defines a classifier using an additive model:

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

Strong classifier

Weak classifier

Weight

Features vector

Boosting

- Defines a classifier using an additive model:

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

Strong classifier

Features vector

Weight

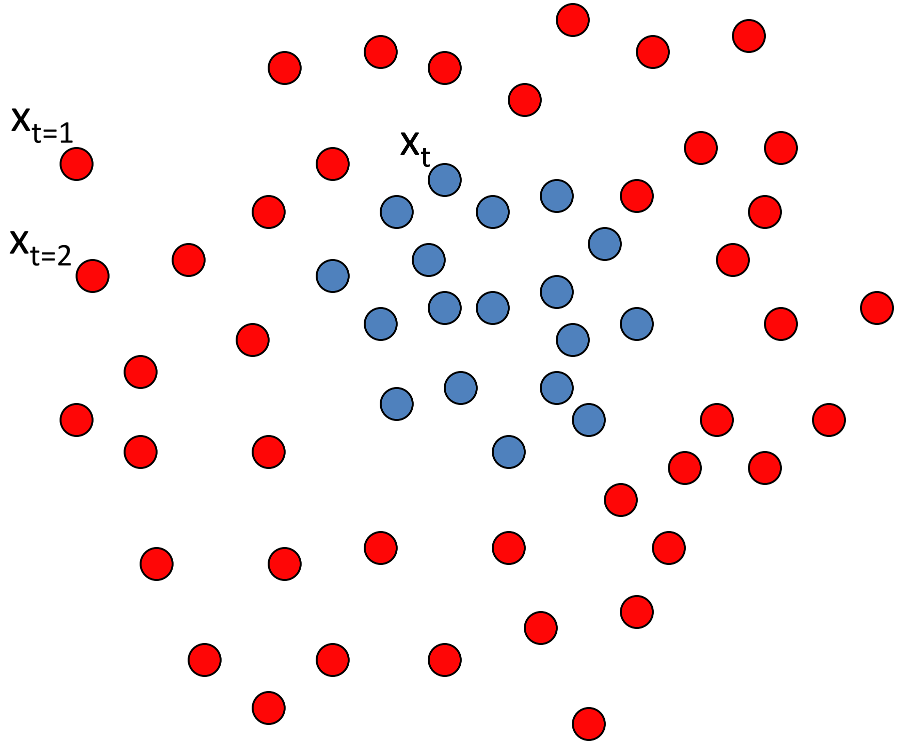
Weak classifier

- We need to define a family of weak classifiers

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

Toy Example

- It is a sequential procedure:



Each data point has

a class label:

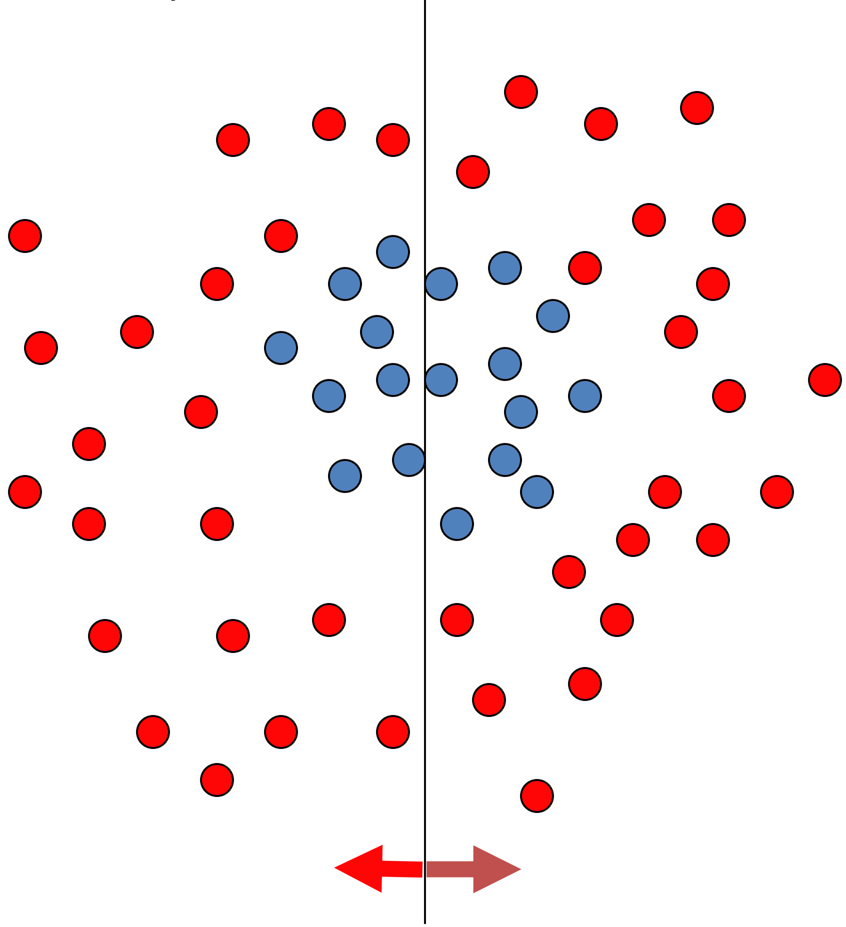
$$y_t = \begin{cases} +1 & (\ ) \\ -1 & (\ ) \end{cases}$$

and a weight:

$w_t = 1$

Toy Example

Weak learners from the family of lines



Each data point has

a class label:

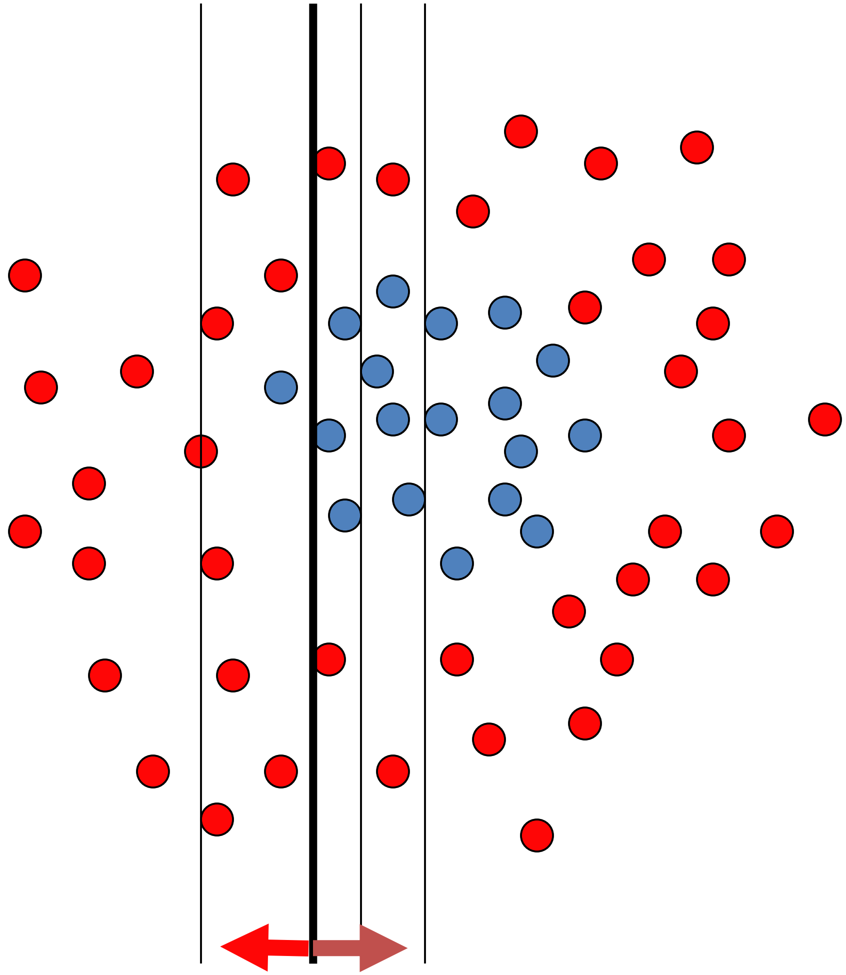
$$y_t = \begin{cases} +1 & (\ ) \\ -1 & (\ ) \end{cases}$$

and a weight:

$w_t = 1$

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

Toy Example



This one seems to be the best

Each data point has a class label:

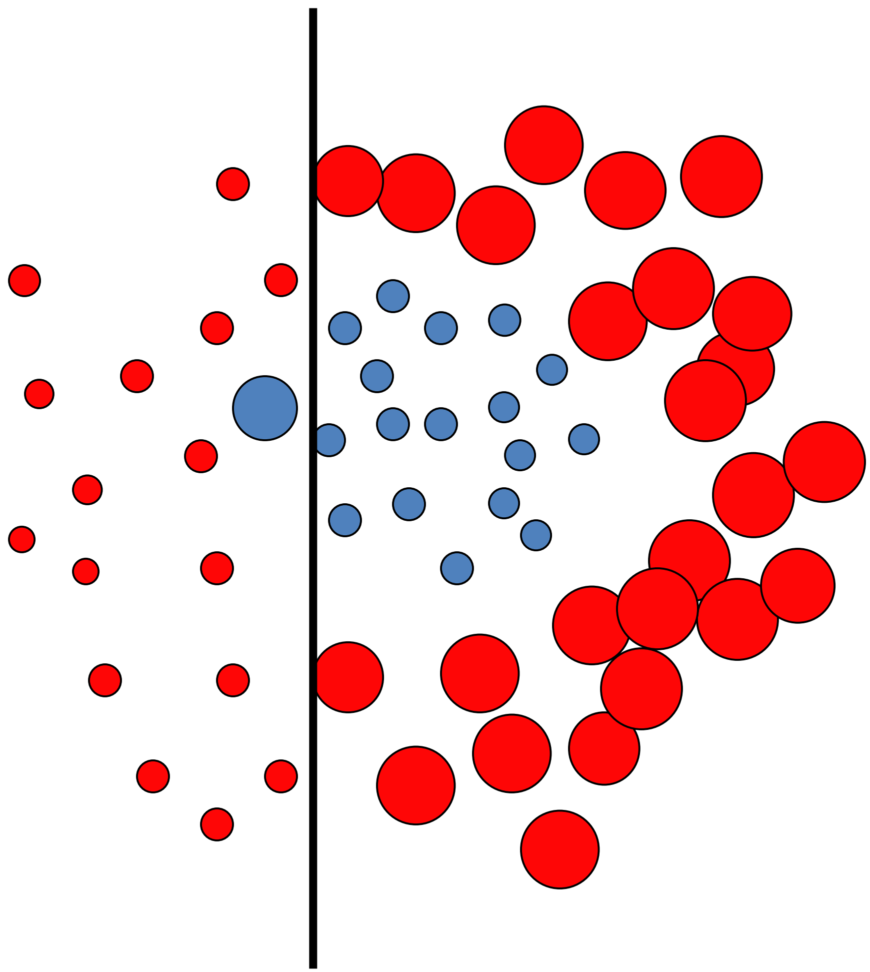
$$y_t = \begin{cases} +1 & (\text{red}) \\ -1 & (\text{blue}) \end{cases}$$

and a weight:

$w_t = 1$

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

Toy Example



Each data point has

a class label:

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

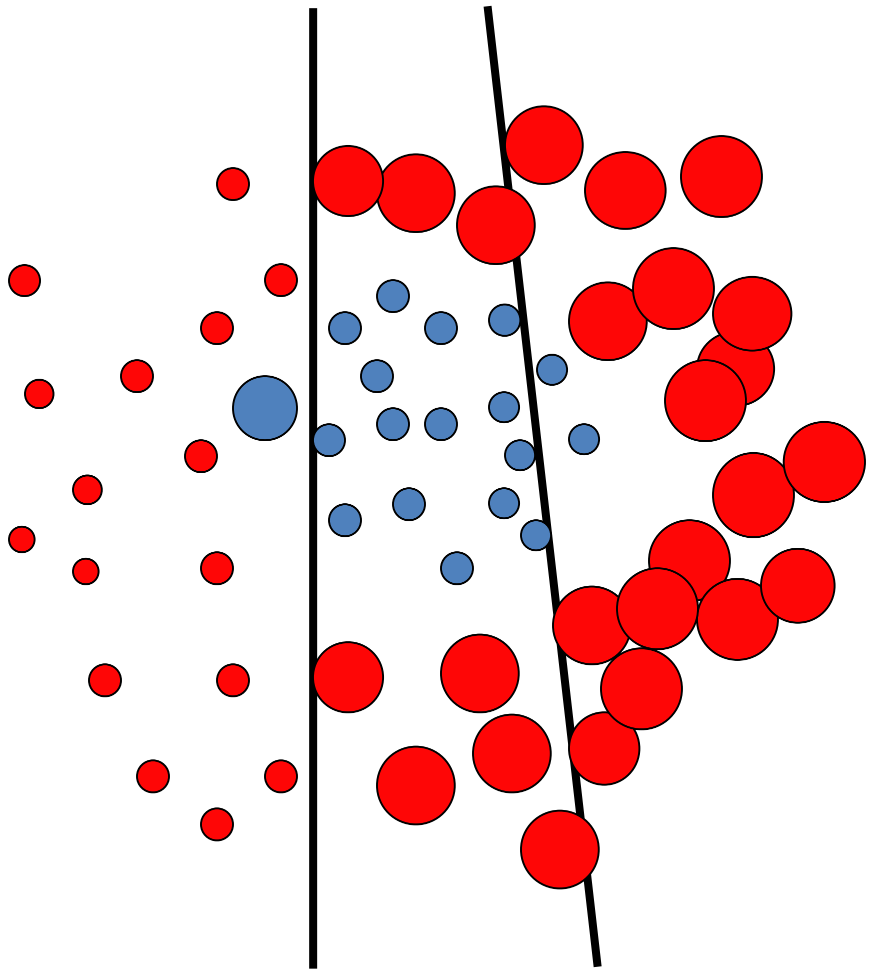
We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t F(x_t)\}$$

Current weak classifier

We set a new problem for which the current classifier performs at chance again

Toy Example



Each data point has

a class label:

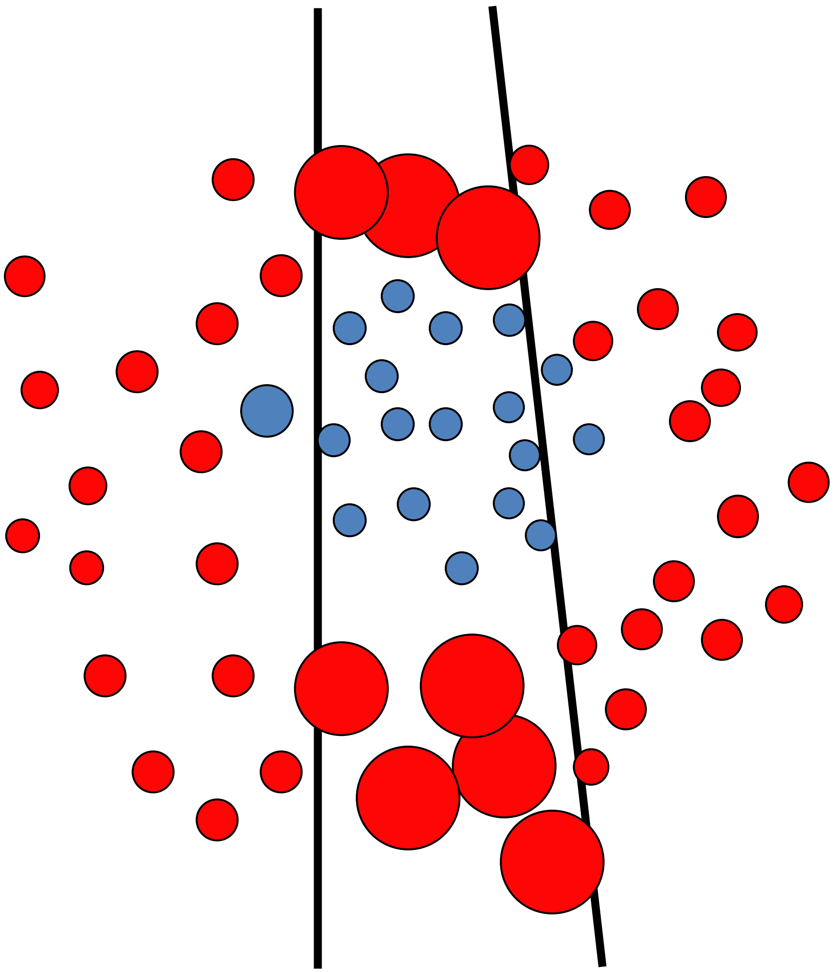
$$y_t = \begin{cases} +1 & (\text{red}) \\ -1 & (\text{blue}) \end{cases}$$

We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t F(x_t)\}$$

Similarly, we learn another weak classifier

Toy Example



Reweighting

Each data point has

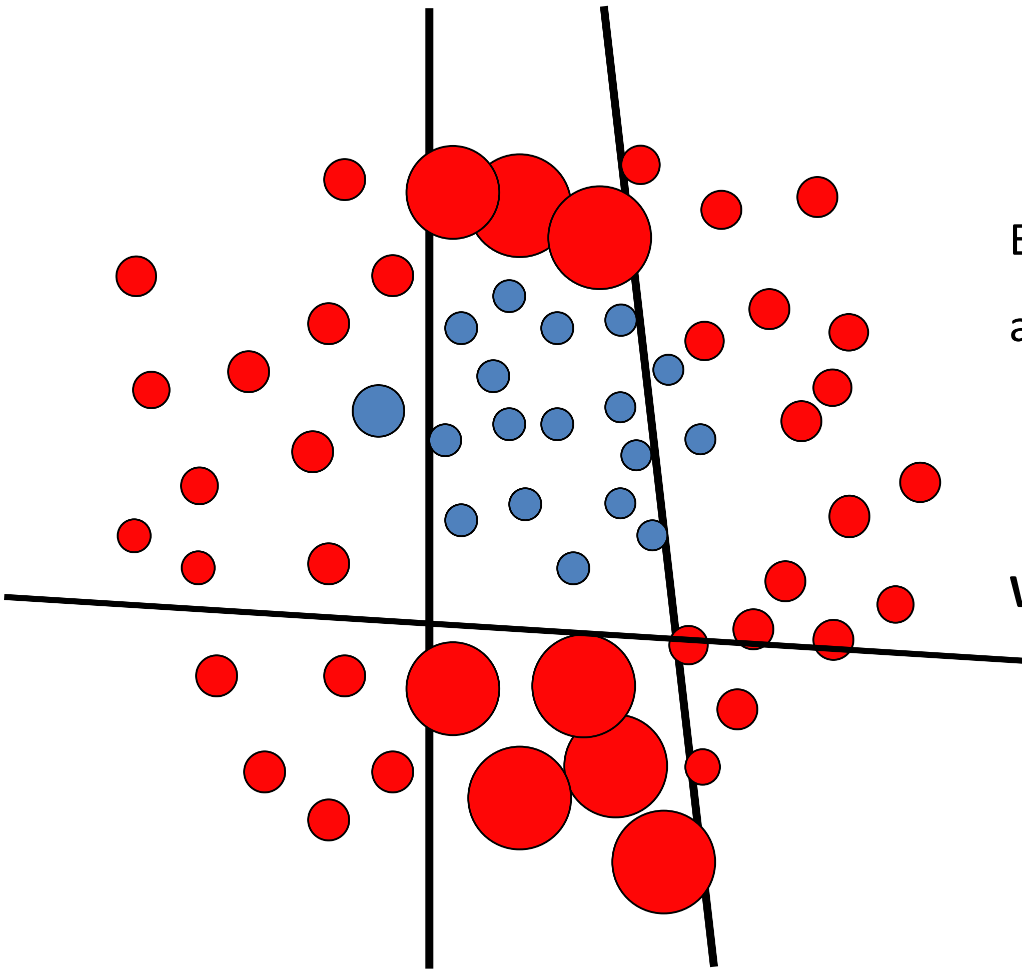
a class label:

$$y_t = \begin{cases} +1 & (\text{red}) \\ -1 & (\text{blue}) \end{cases}$$

We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t F(x_t)\}$$

Toy Example



Each data point has

a class label:

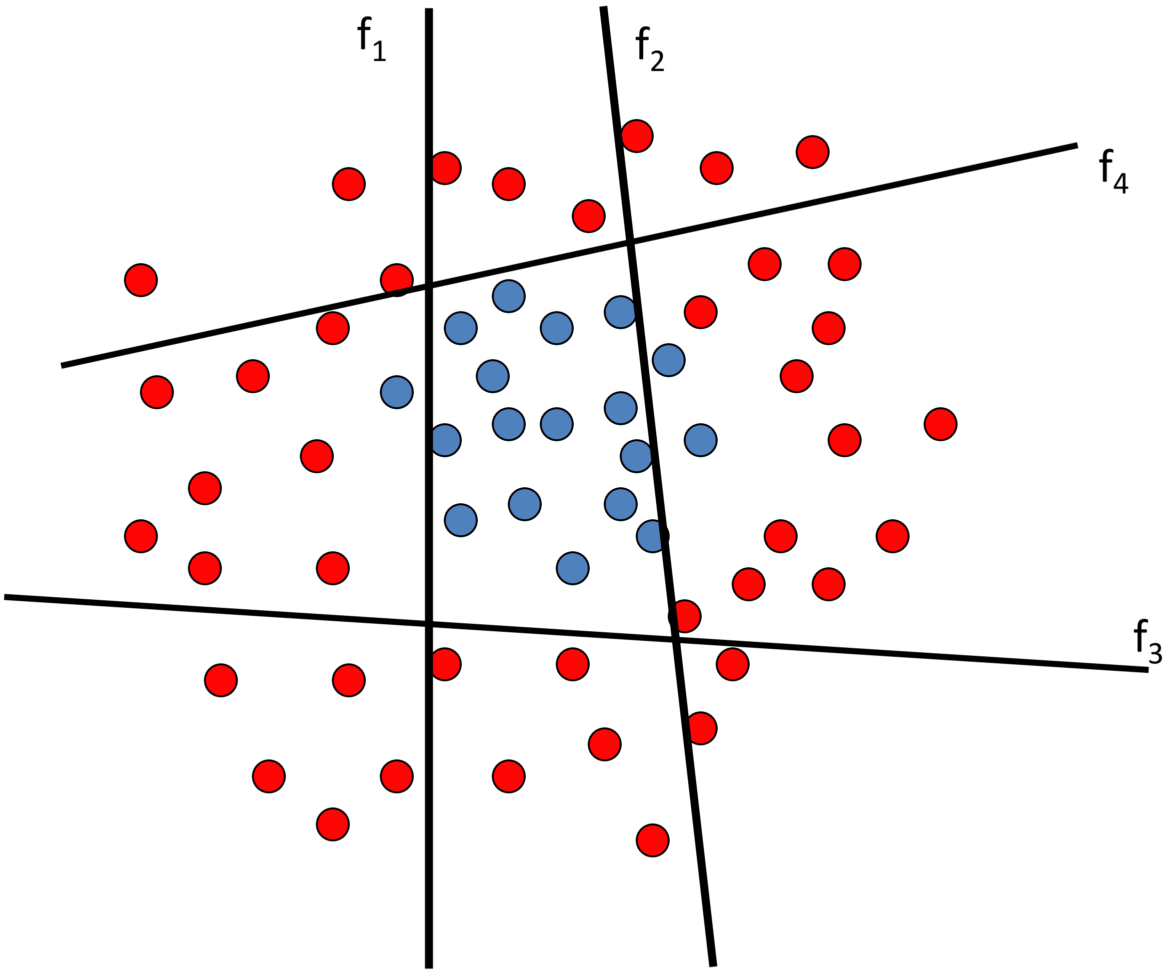
$$y_t = \begin{cases} +1 & (\ ) \\ -1 & (\ ) \end{cases}$$

We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t\, F(x_t)\}$$

Similarly, we learn another weak classifier

Toy Example



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 introduce gentleBoosting
  - It is simple to implement and numerically stable.

Boosting

Boosting fits the additive model

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

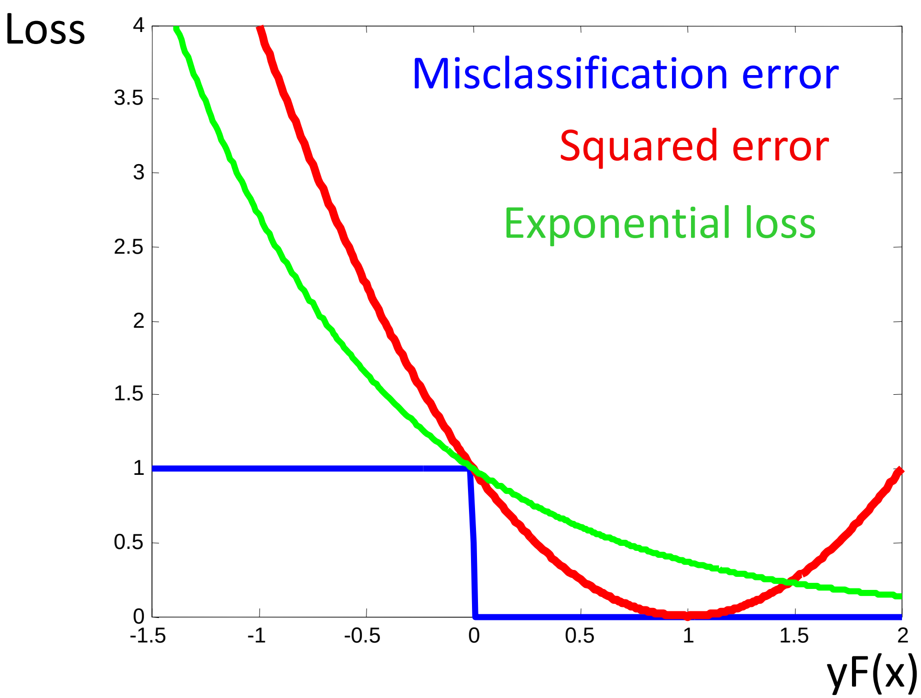
by minimizing the exponential loss

$$J(F) = \sum_{t=1}^{N} e^{-y_t F(x_t)}$$

Training samples

The exponential loss is a differentiable upper bound to the misclassification error.

Exponential Loss



Squared error

$$J = \sum_{t=1}^{N} [y_t - F(x_t)]^2$$

Exponential loss

$$J = \sum_{t=1}^{N} e^{-y_t F(x_t)}$$

Boosting

Sequential procedure. At each step m we add

$$F(x) \leftarrow F(x) + f_m(x)$$

to minimize the residual loss

$$(\phi_m) = \arg\min_{\phi} \sum_{t=1}^{N} J\left(y_t, F(x_t) + f(x_t; \phi)\right)$$

Parameters of the weak classifier

Desired output

input

For more details: Friedman, Hastie, Tibshirani. “Additive Logistic Regression: a Statistical View of Boosting” (1998)

gentleBoosting

- At each iteration:

  We chose $f_m(x)$ that minimizes the cost:

$$J(F + f_m) = \sum_{t=1}^{N} e^{-y_t(F(x_t)+f_m(x_t))}$$

  Instead of doing exact optimization, gentle Boosting minimizes the **approximation** of the error:

$$J(F) \propto \sum_{t=1}^{N} \underbrace{\boxed{e^{-y_t F(x_t)}}}_{\text{Weights at this iteration}} (y_t - f_m(x_t))^2$$

At each iterations we just need to solve a weighted least squares problem

For more details: Friedman, Hastie, Tibshirani. "Additive Logistic Regression: a Statistical View of Boosting" (1998)

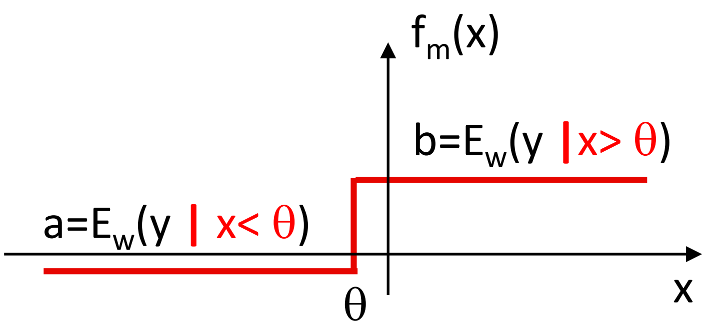
Weak Classifiers

- The input is a set of weighted training samples (x,y,w)

- Regression stumps: simple but commonly used in object detection.

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

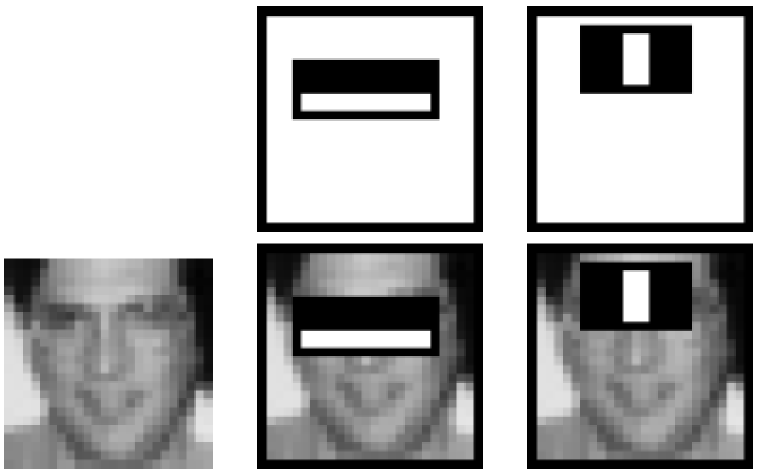
Four parameters: $\phi = [a, b, \theta, k]$

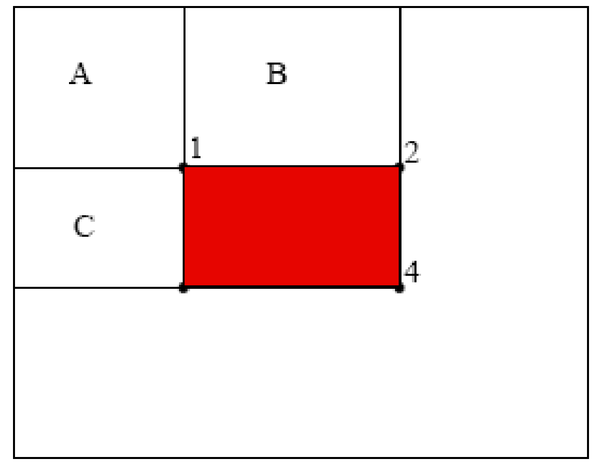


Features -> Weak Detectors

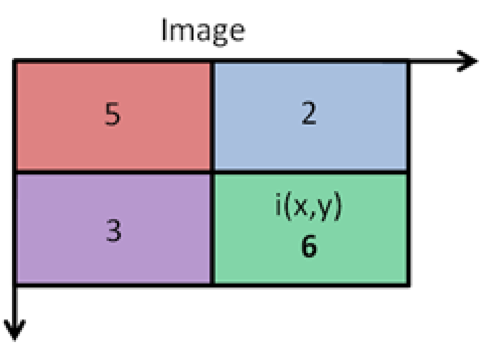
Haar filters and integral image

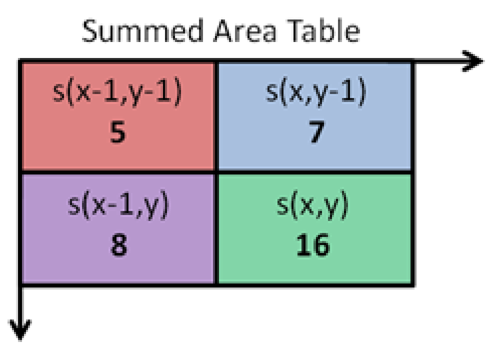
Viola and Jones, ICCV 2001

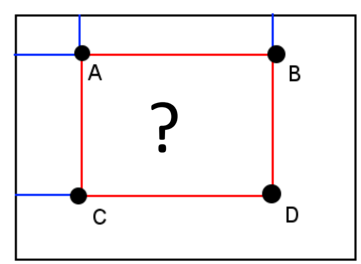




The average intensity in the block is computed with four sums independently of the block size.



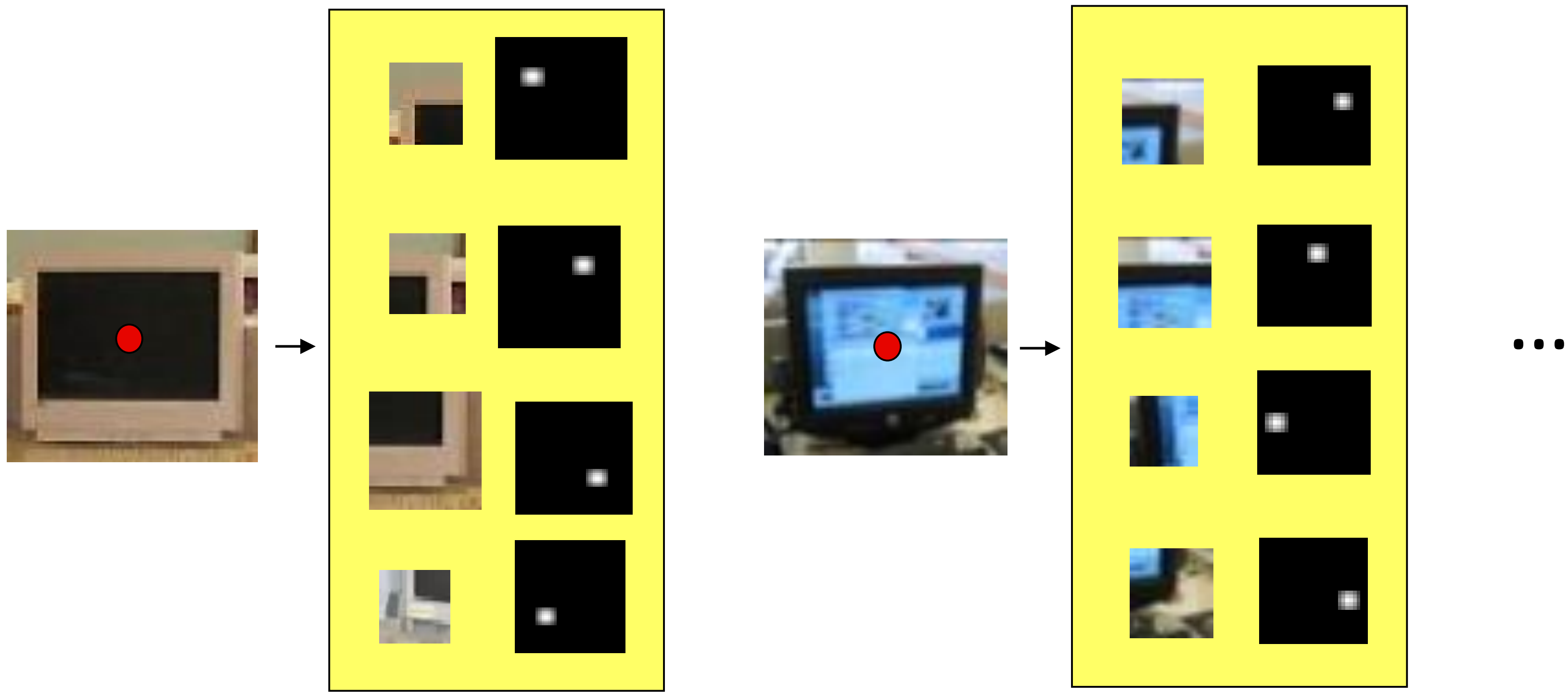




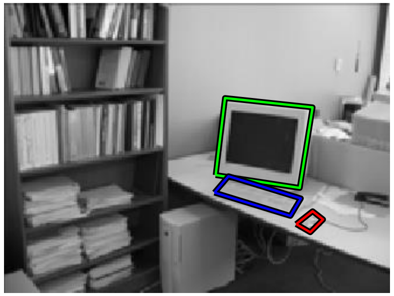
Sum = D - B - C + A

Features -> Weak Detectors

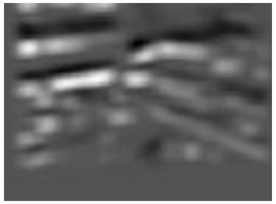
For screen detection, we may collect a set of part templates from a set of training objects to build feature set.



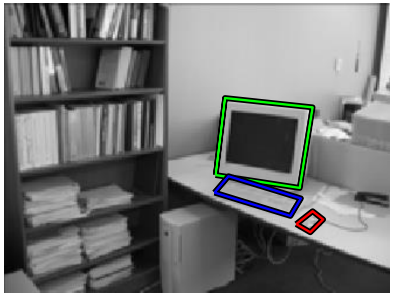
Example: Screen Detection



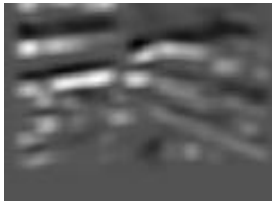
Feature output



Example: Screen Detection



Feature
output

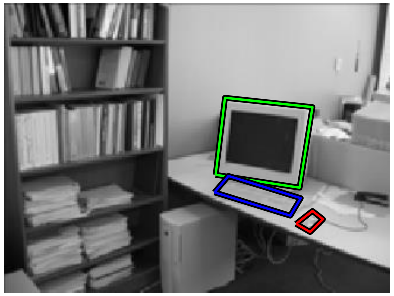


Thresholded
output

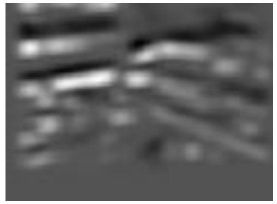
Weak ‘detector’

Produces many false alarms.

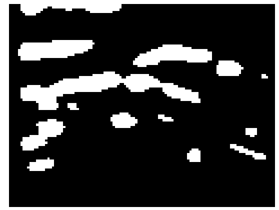
Example: Screen Detection



Feature output

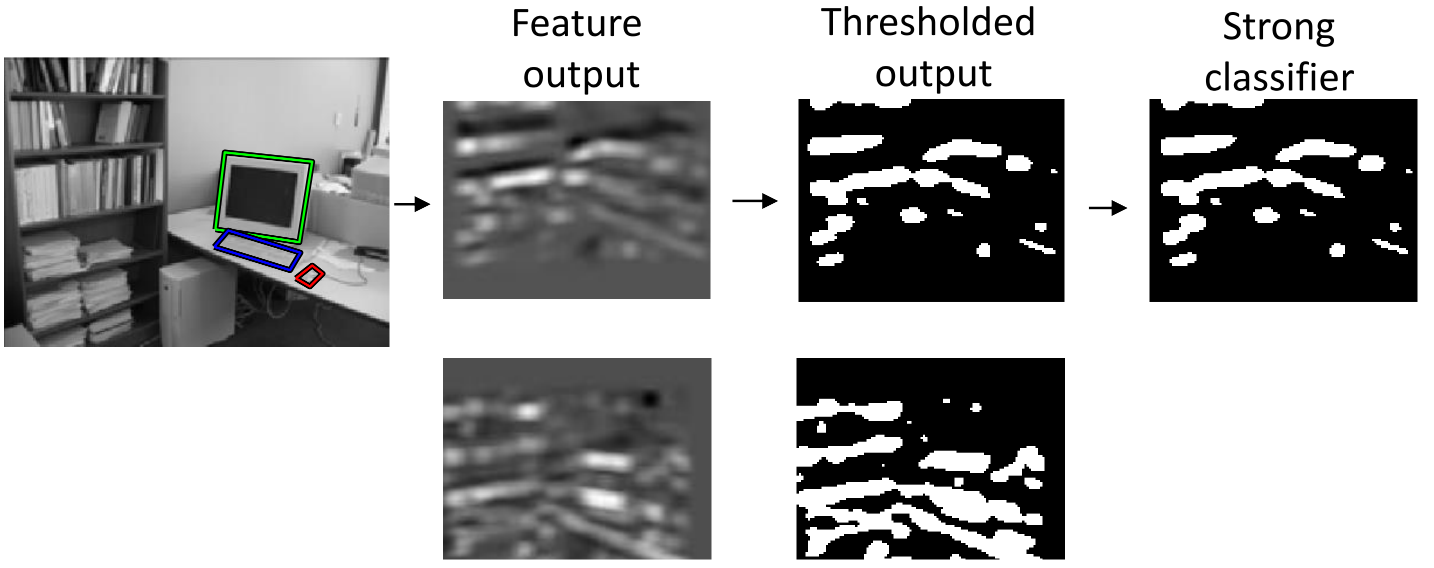


Thresholded output



Strong classifier at iteration 1

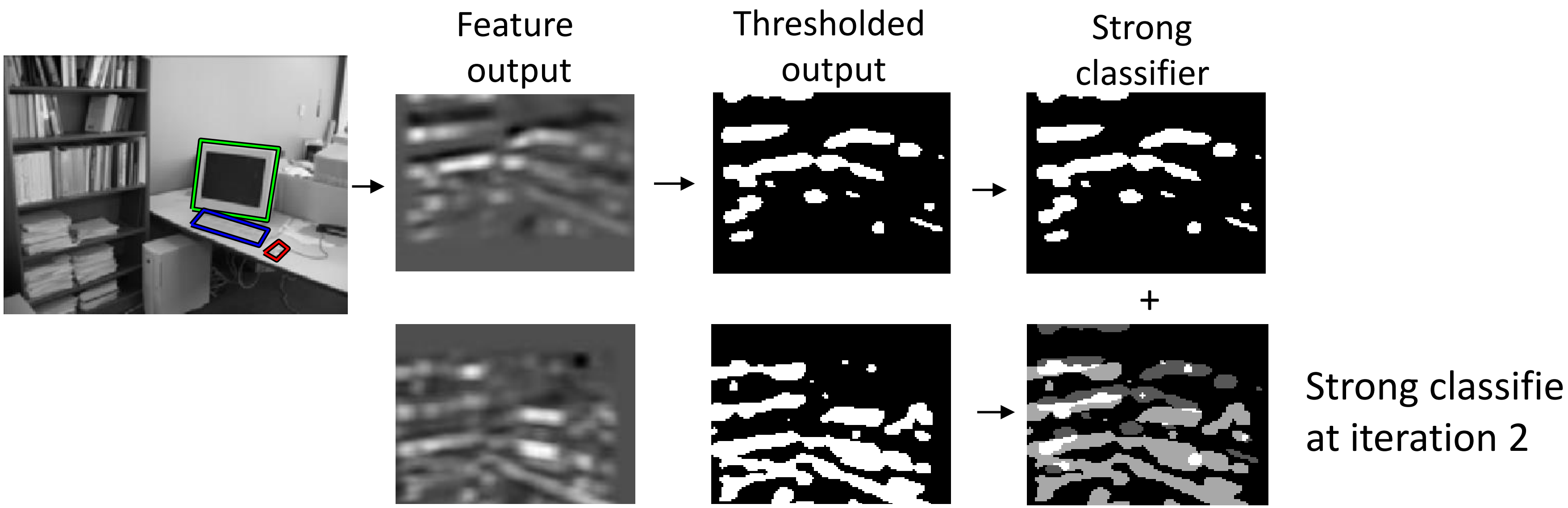
Example: Screen Detection



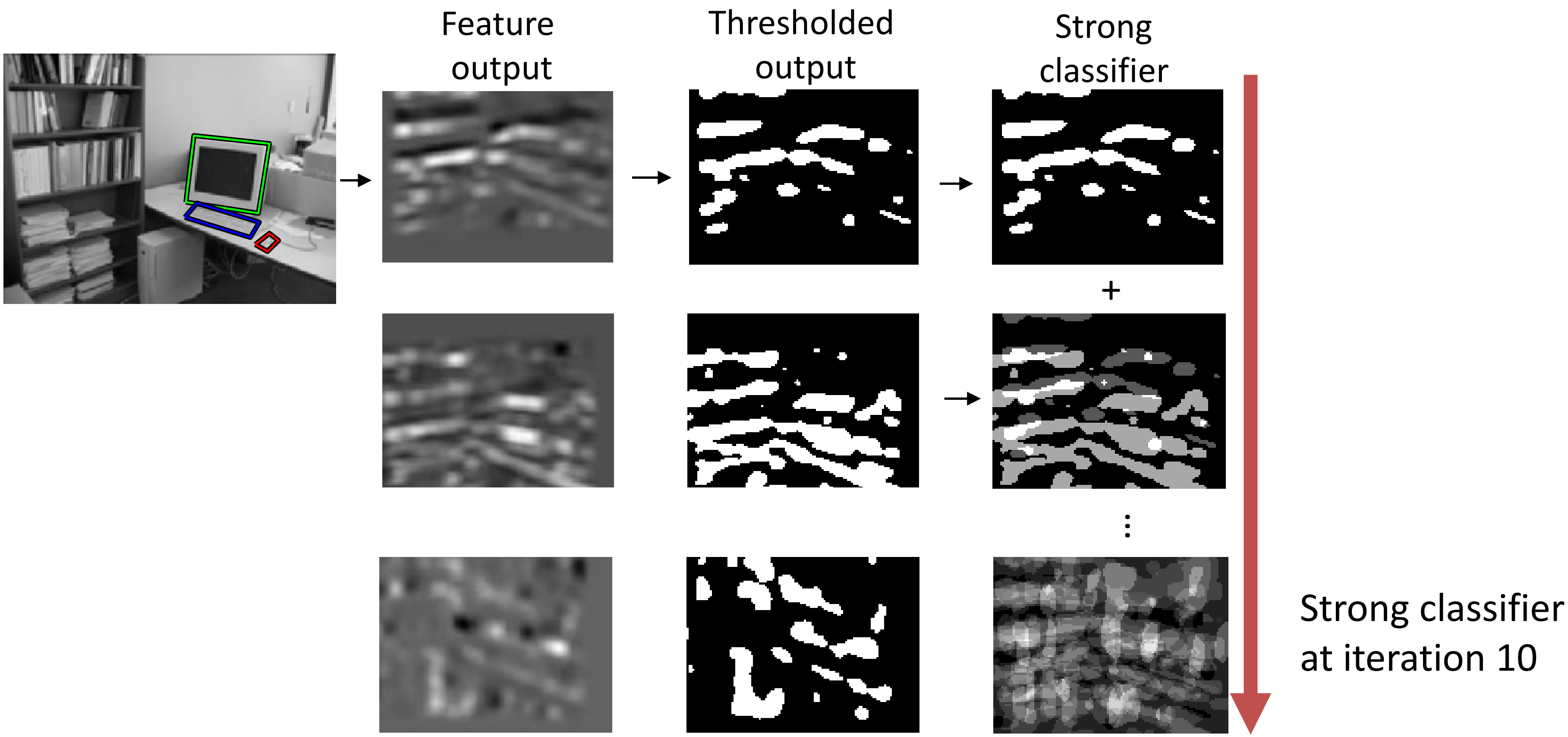
Second weak 'detector'

Produces a different set of false alarms.

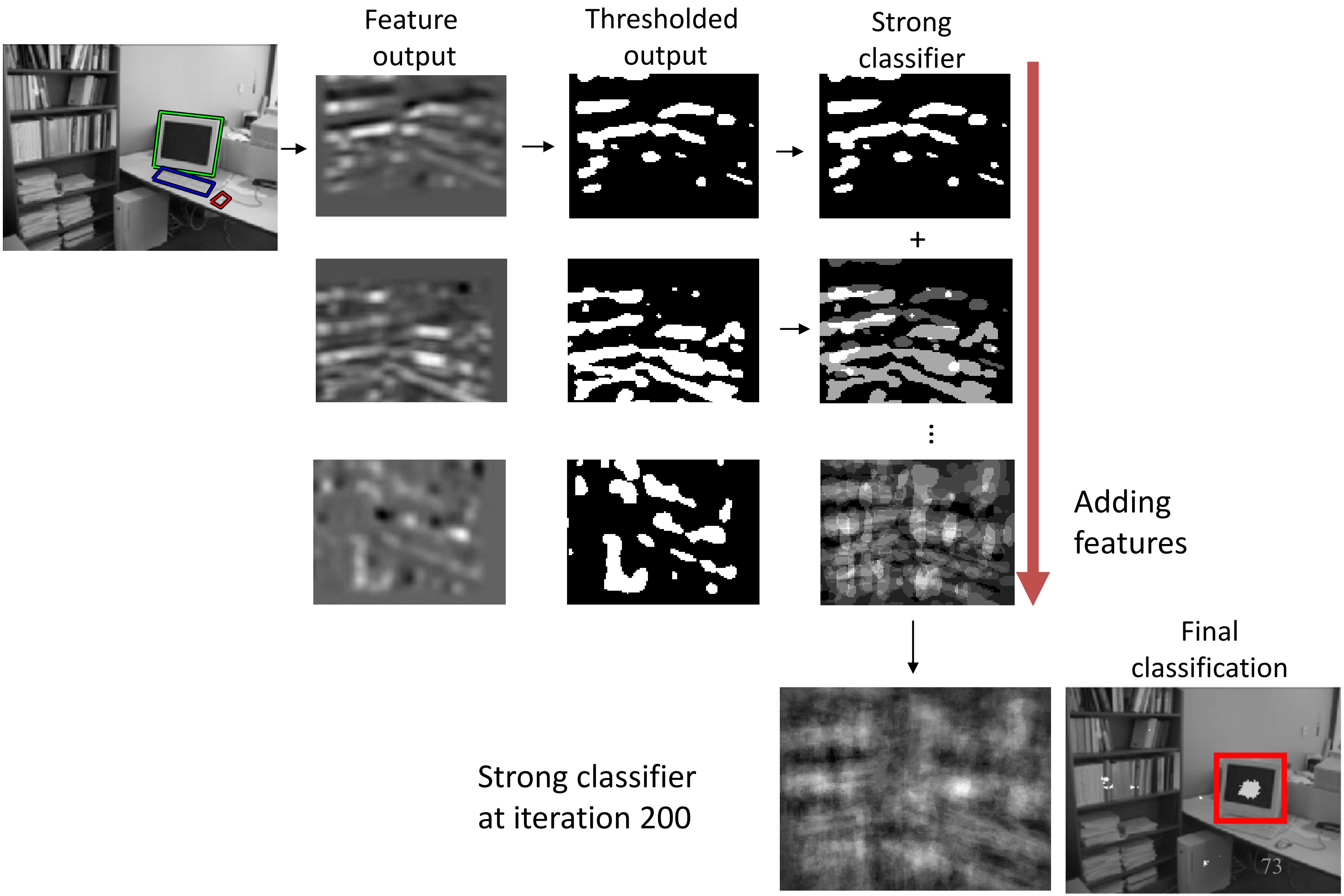
Example: Screen Detection



Example: Screen Detection

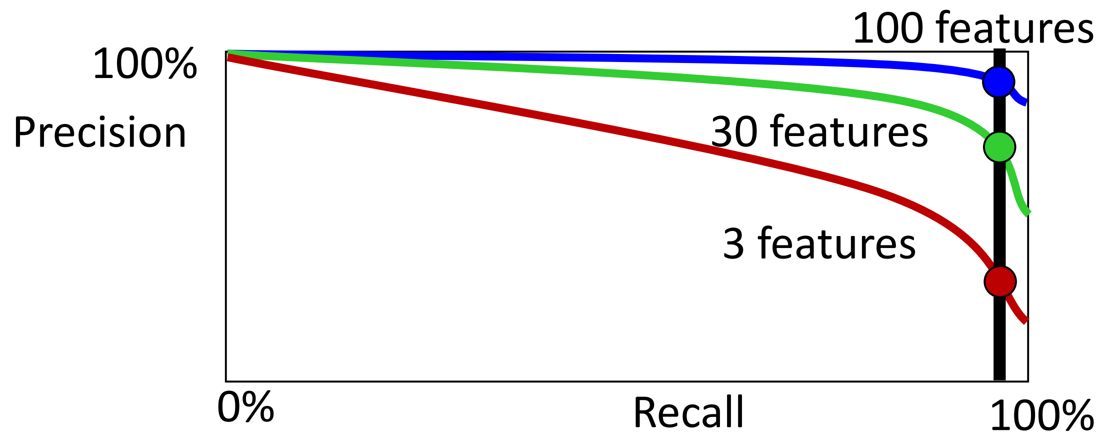


Example: Screen Detection

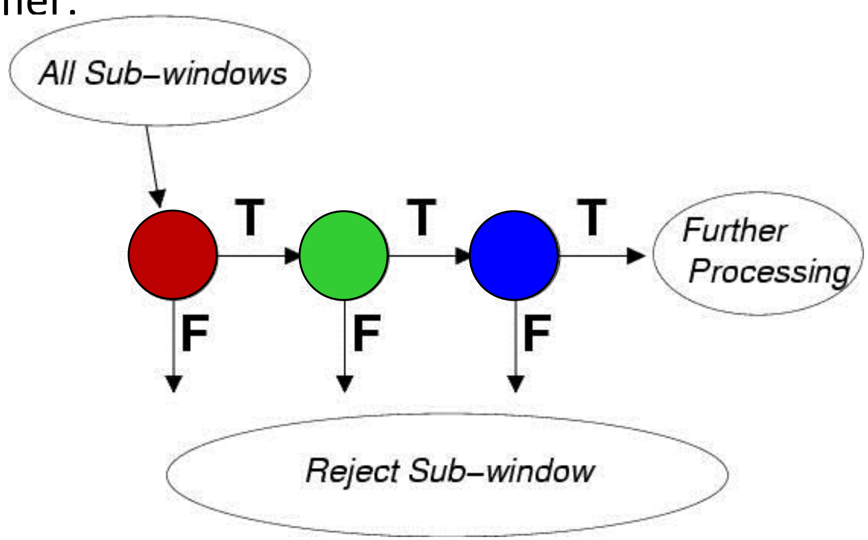


Cascade of classifiers

What is the motivation: some negative samples may be rejected based on few features!



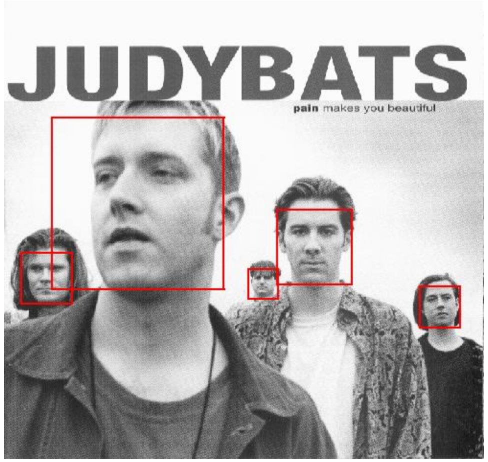
We want the complexity of the 3 features classifier with the performance of the 100 features classifier:

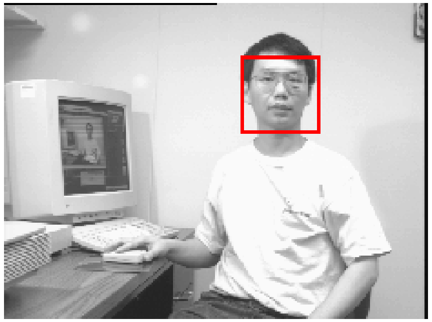


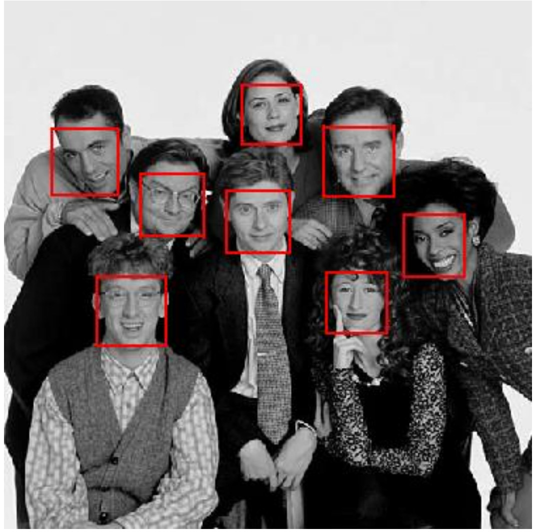
Select a threshold with high recall for each stage.

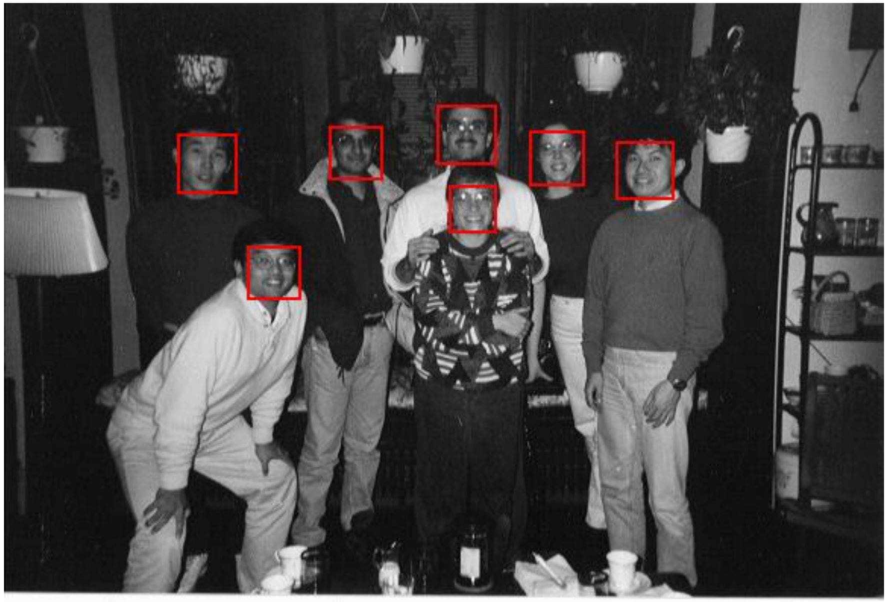
We increase precision using the cascade

Output of Face Detector on Test Images



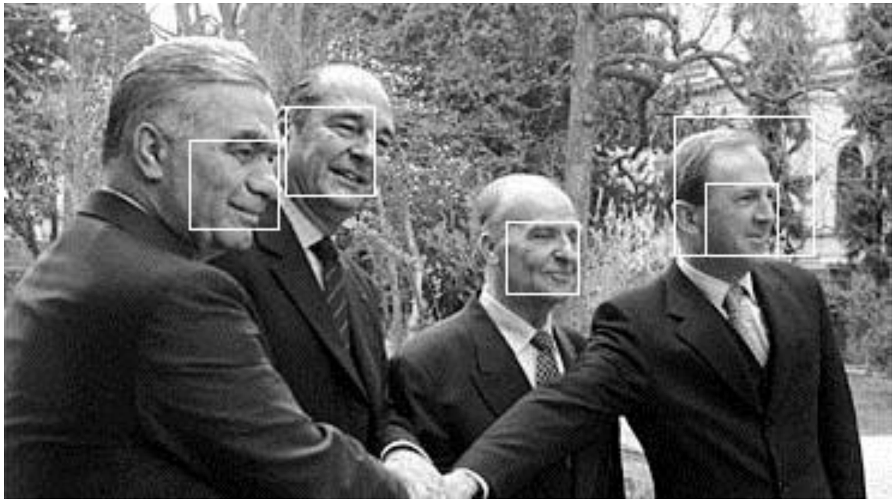






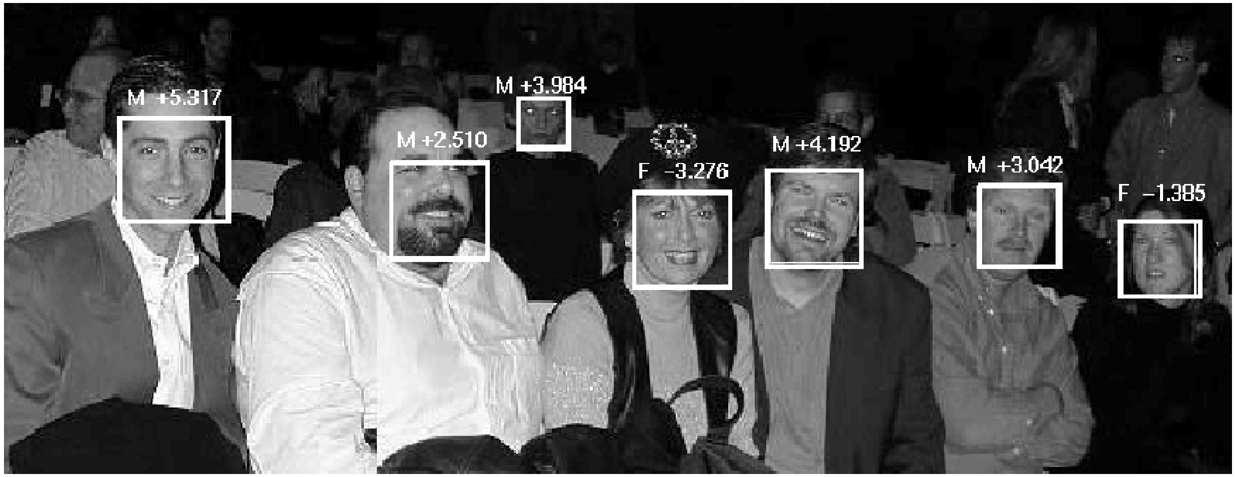
Other detection tasks

Facial Feature Localization



Profile Detection

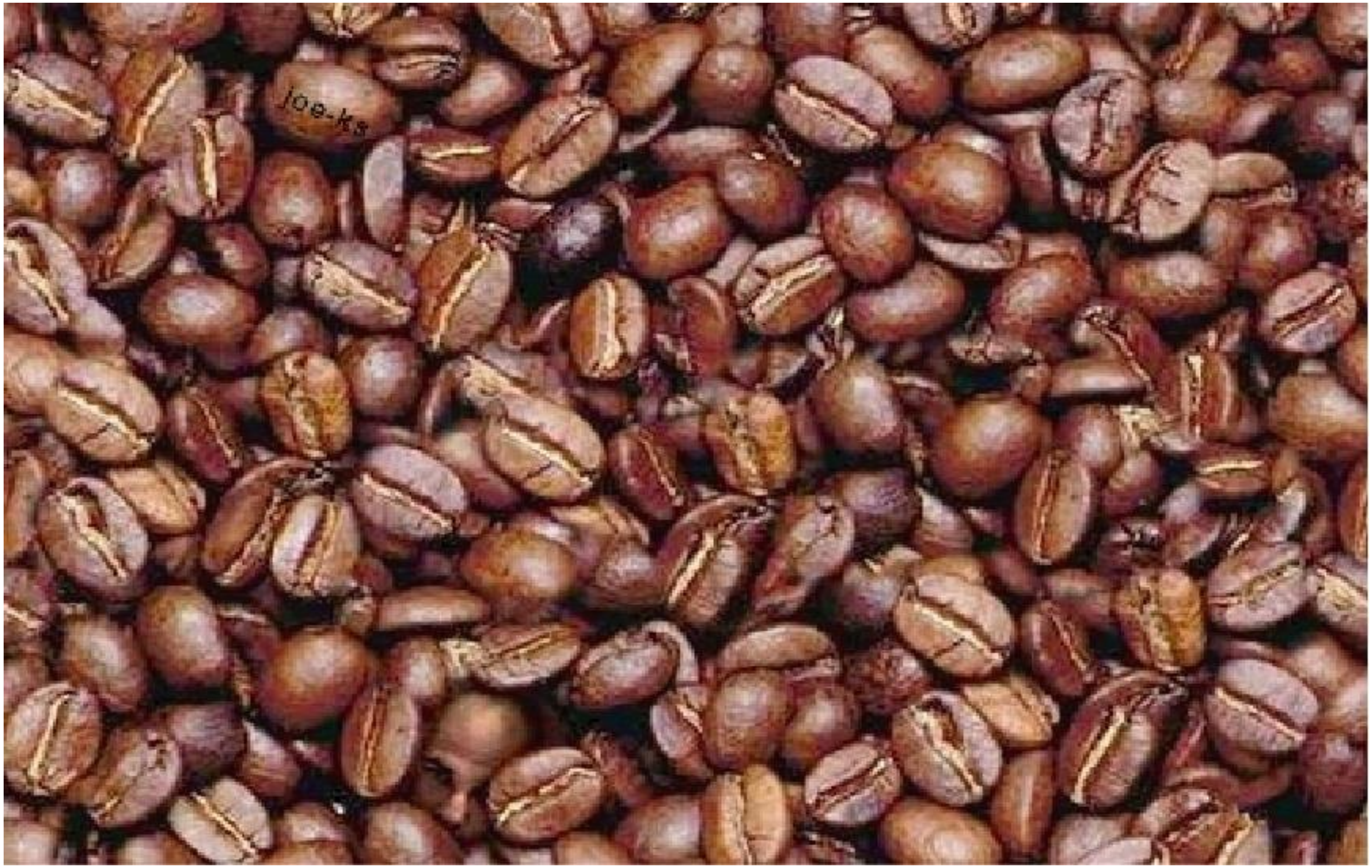
Male vs.
female



Profile Detection

“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)
- No mechanism for handling 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). Pdf

- 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.
  Postscript or gzipped postscript.

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