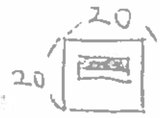
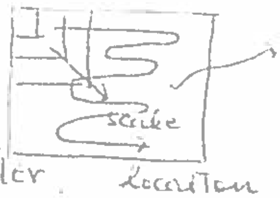
1. a-i) face images cropped and localized, reflecting variations
[9] on illumination, expression, pose, identities $\rightarrow$ resized to 20x20 $\rightarrow$ vectorized $x \in \mathbb{R}^{D=20\times 20}$ with $t = 1$

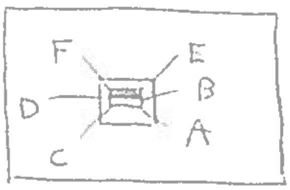
random crops not including faces, typically a larger number of this than that of face class. $\rightarrow$ resized and vectorized to $x \in \mathbb{R}^{400}$ with $t = -1$ //

Haar-basis like functions e.g.  or , we vary the typ and scale / location to generate a large feature pool, e.g. 45K $= M'$. The filter responses are fast computed on an Integral image //

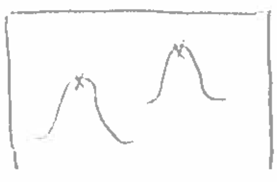
$$y_m(x) = \begin{cases} +1 & \text{if the filter response i.e. } x^T \boxed{} \geq \theta \\ -1 & \text{otherwise} \end{cases}$$, (also "α_m" as weights)

The examples chosen are as above, cause they capture characteristi patterns of objects e.g. eyes, eye brows, etc of face. $M << M'$ //

a-ii) Compute the integral image $II(x, y)$, scan every possible
[5] window in the image , pick a window and apply the boosting classifier.



For the example, we read corner values of A ~ E on the integral image then $\{A - B - C + D\} - \{B - E - D + F\}$ to compute the filter response. $\rightarrow y_m(x) \rightarrow \text{sign}(\sum^M \alpha_m y_m(x)) \rightarrow$ a response map



We do non-local maxima suppression to do (ma detection.

1. b-i) $y_m(x)$ that minimises $J_m = \sum_{n=1}^{N} w_n^{(m)} I(y_m(x) \neq t_n)$.

[5] $E = e^{\alpha_m/2} \sum_{n \in T_m} w_n^{(m)} + e^{\alpha_m/2} \sum_{n \in M_m} w_n^{(m)}$

$= (e^{\alpha_m/2} - e^{-\alpha_m/2}) \underline{\sum_{n=1}^{N} w_n^{(m)} I(y_m(x) \neq t_n)} + e^{-\alpha_m/2} \sum_{n=1}^{N} w_n^{(m)}$

the 2nd term is constant as to $y_m(x)$, it becomes minimising

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

b-ii) $w_n^{(m+1)} = w_n^{(m)} \exp\{ \alpha_m I(y_m(x) \neq t_n)\}$.

[5] $\frac{\partial E}{\partial w_n} = \exp\{-\frac{1}{2} t_n \alpha_m y_m(x_n)\}$, $\rightarrow$ $w_n = w_n \frac{\partial E}{\partial w_n}$ $\rightarrow$ the answer //

c) # of data points = # of ~~scan~~ scan windows = W x H x # of scales

[5] thus number is typically huge. we are given ~~a~~ only a little amoun

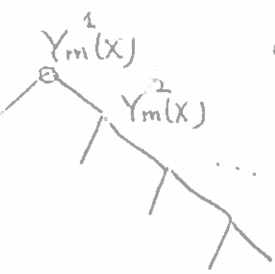
time per point = $\frac{\text{the process time}}{W \times H \times \#\text{ of scales}}$ $\rightarrow$ need a higly efficient classifie

We have a few positive samples and all the rest (W x H x # of s)

as negative samples from an image. → need a high precision-rec

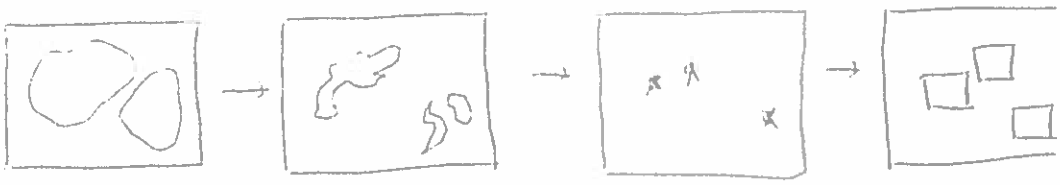
otherwise often miss target objects or encounter too many false

alarms. Boosting cascade has multiple boosting classifiers in a



coarse-to-fine ~~man~~ manner in an imbalanced tree

structure $Y_m(x)$, M = 2, → 5, 10, 20, ·· 100 e.g



coarse → fine

final detection

It applies a coarse to all pixels, then finer to selected

pixels, rejecting majority of negative samples quickly, accelerating

the overall run-time. //

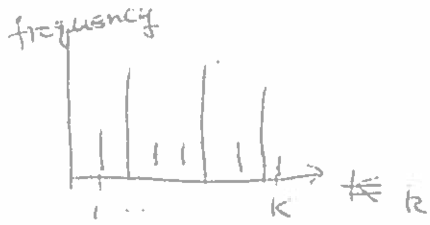
2 a-i) We collect M images of different classes, apply IP detector

[3] to the images, and harvest small image patches around the IPs. Say N patches per image. We then raster scan the patches to form vectors, $x \in \mathbb{R}^D$, called "visual words". We manually set K, the codebook size, K usually $\ll M \times N$. If K is too small, we lose discriminative information (under-representation). If K is too large, overfitting happens (over-representation) and memory, time-complexity issues.

The output is the codebook of size K, i.e. μ_k, $k=1, \ldots, K$. //

a-ii) Take a new image, collect the visual words from the image

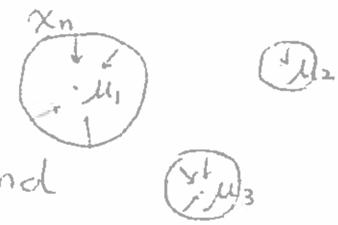
[4] in the way above. Say N words. Then, every word is compared with all μ_k $(k=1, \ldots K)$ and assigned to it nearest μ_k. The corresponding histogram bin is increased.



Thus costs $O(KND)$. //

b-i) r_{nk} is the membership indicator variable. If x_n is assigned

[4] to cluster k, then $r_{nk}=1$ and $r_{nj}=0$ for $j \neq k$. J measures the cluster compactness.



It tries to minimize the variance of each cluster and the average of all clusters variance

Input: x_n, $n=1, \ldots N$, K manually set, initial μ_k, $k=1, \ldots K$

Output: r_{nk}, $n=1, \ldots, N$, $K=1, \ldots K$, μ_k, $k=1, \ldots, K$

We optimize r_{nk}, μ_k by minimising J.

b-ii) w.r.t. r_{nk}, keeping μ_k fixed.

[5] $J = \ldots + r_{n1}\|x_n-\mu_1\|^2 + r_{n2}\|x_n-\mu_2\|^2 + \ldots + r_{nk}\|x_n-\mu_k\|^2 + \ldots$

r_{nk} is binary and $\sum_k r_{nk} = 1$. Minimising J w.r.t. r_{nk} means

$r_{n1} = 0$, $r_{n2} = 0$. $r_{nk} = 1$, ... , if $k = \arg\min_j \|x_n - \mu_j\|^2$ //

b-iii) minimise J w.r.t. μ_k, keeping r_{nk} fixed. The cost fth.

[4] is quadratic, a unique global optimum.

$$\frac{\partial J}{\partial \mu_k} = 2 \sum_{n=1}^{N} r_{nk}(x_n - \mu_k) = 0 \rightarrow \mu_k = \frac{\sum_n r_{nk} x_n}{\sum_n r_{nk}}$$

b-iv) It stops when no change in μ_k or r_{nk} or J is saturated.

[2] Convergence proof exists, it is not global opti. solution, thus depends on initialisations.

b-v) GMM parameters: π_k, μ_k, Σ_k, $k=1, \ldots, K$.

K-m (hard clustering) vs GMM (soft clustering)

[3] K-m assigns data to nearest clusters, while GMM assigns data to Gaussian densities that best represent the data.

i) a-i) $y(x, w) = w_0 + w_1 x + w_2 x^2 + \cdots + w_M x^M = \sum_{j=0}^{M} w_j x^j$

[8]

2 a-ii) $E(w) = \frac{1}{2} \sum_{n=1}^{N} \{ y(x_n, w) - t_n \}^2$ //

[2]

a-iii) $E = \frac{N}{2} w_0^2 + \frac{1}{2} \sum_{n=1}^{N} (w_1^2 x_n^2 + t_n^2 + 2 w_0 w_1 x_n - 2 w_1 x_n t_n - 2w_0$

[8]

6 $E = \frac{1}{2} \sum_{n=1}^{N} (w_0 + w_1 x_n - t_n)^2 = \frac{1}{2} \sum (w_0^2 + w_1^2 x_n^2 + t_n^2 + 2 w_0 w_1 x_n - 2 w_1 x_n t_n - 2 w_0 t_n)$

$$\frac{\partial E}{\partial w_0} = N w_0 + w_1 \sum_{n=1}^{N} x_n - \sum_{n=1}^{N} t_n = 0 \quad //$$

$$\frac{\partial E}{\partial w_1} = w_1 \sum_{n=1}^{N} x_n^2 + w_0 \sum_{n=1}^{N} x_n - \sum_{n=1}^{N} x_n t_n = 0 \quad //$$

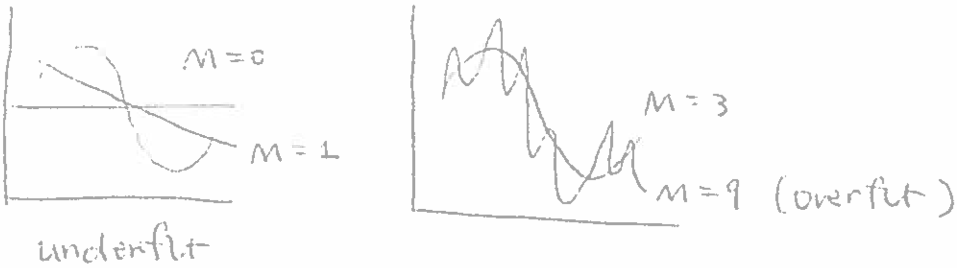
solving the linear eqn system.

$$w_1 = \frac{N \sum_{n=1}^{N} x_n t_n - \left(\sum_{n=1}^{N} x_n \right) \left(\sum_{n=1}^{N} t_n \right)}{\left(\sum_{n=1}^{N} x_n \right)^2 - N \left(\sum_{n=1}^{N} x_n^2 \right)} \quad //$$

$$w_0 = -\frac{1}{N} w_1 \sum_{n=1}^{N} x_n + \frac{1}{N} \sum_{n=1}^{N} t_n \quad //$$

a-iv)

[3]



a-v) $E(w) = \frac{1}{2} \sum_{n=1}^{N} \{ y(x_n, w) - t_n \}^2 + \frac{\lambda}{2} \|w\|^2$ //

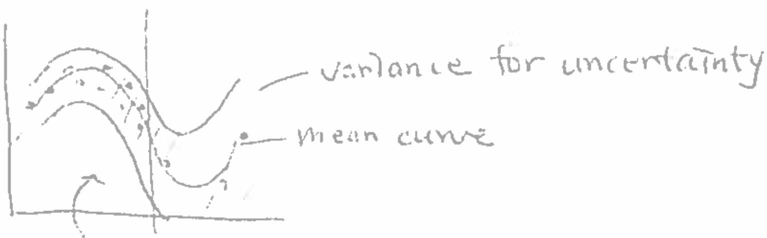
[2] $\|w\|^2 = w^T w = w_0^2 + w_1^2 + \cdots + w_M^2$, all elements of w are forced to be small.

b-i) $m(p(t_{N+1} | t)) = k^T C_N^{-1} t$

[5] $\sigma^2(p(t_{N+1} | t)) = c - k^T C_N^{-1} k$ //

c)

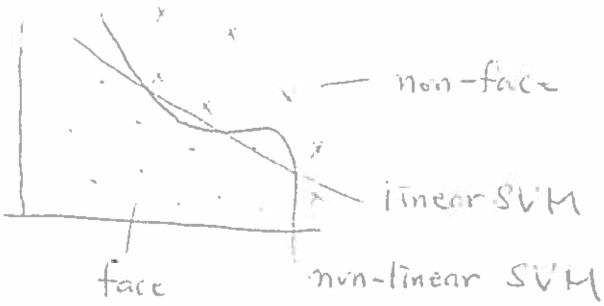
[5]



we have more data points here thus less variance.

4. a-i) We collect a large number of face images croped and normalized to, say 20×20 and a larger number of non-face image in the same size. They are vectorized to $x \in \mathbb{R}^{400}$, and labelled +1 for face, −1 for non-face.

[5]



a-ii) $y(x) = \sum_{n=1}^{N} a_n t_n k(x, x_n) + b$

new data point

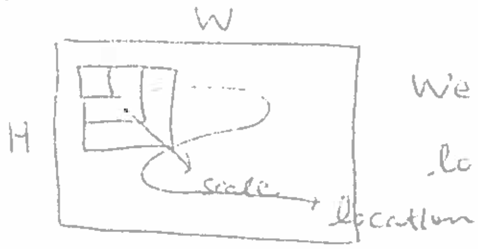
label

Lagrangian

bias

training data point

[7]



We do scanning windows at every scale and location. Each window is croped and sized to say 20×20, vectorized to x.

Then we have $y(x) = 1$ or -1.

Linear : $k(x, x_n) = x^T x_n \rightarrow y(x) = x^T \underbrace{\left(\sum_n \alpha_n t_n x_n \right)}_{\text{pre-computed}} + b$

$\rightarrow O(D = 400)$

Nonlinear : $k(x, x_n) = \exp\left(\frac{-\|x - x_n\|^2}{\sigma^2} \right)$ $O(D)$

$y(x) = \sum_n^N \alpha_n t_n k(x, x_n) + b \rightarrow O(ND)$

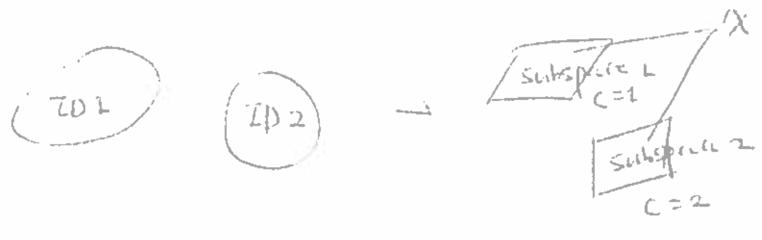
[3] b-i) $\underset{M\times 1}{z} = \underset{M\times D}{U^T} \underset{D\times 1}{x}$, where $U = \underset{D\times M}{[u_1\ u_2 \dots u_M]}$

b-ii) $\underset{D\times 1}{\tilde{x}} = \sum_{i=1}^{M} z_i u_i = \underset{D\times M}{U}\underset{M\times 1}{z}$

[3]

b-iii)

[7]



assign $\arg\min_c \|x - \hat{x}$