



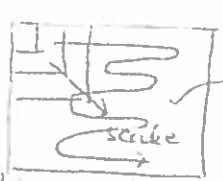
1. a-i) face images cropped and localized, reflecting variations
 [5] on illumination, expression, pose, identities \rightarrow resized to 20×20
 \rightarrow vectorized $x \in \mathbb{R}^D=20 \times 20$ with $t=1$

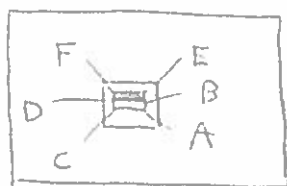
random crops not including faces, typically a larger number of
 this than that of face class. \rightarrow resized and vectorized to
 $x \in \mathbb{R}^{400}$ with $t=-1$ //

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

$y_m(x) = \begin{cases} +1 & \text{if the filter response i.e. } x^T \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix} \geq 0 \\ -1 & \text{otherwise, (also "x_m" as weights)} \end{cases}$

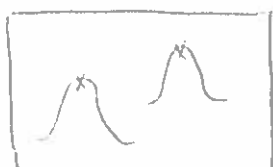
The examples chosen are as above, cause they capture characteristic
 patterns of objects e.g. eyes, eye brows, etc of face. $M \ll M'$

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. Location



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.

$\rightarrow y_m(x) \rightarrow \text{sign} \left(\sum^M x_m y_m(x) \right) \rightarrow$ a response map



We do non-local maxima suppression to do fine
 detection.

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

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

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

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

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

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

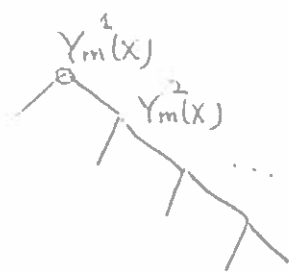
$$[5] \frac{\partial E}{\partial w_n} = \exp \left\{ -\frac{1}{2} t_n \alpha_m y_m(x_n) \right\}, \rightarrow w_n = w_n \frac{\partial E}{\partial w_n} \rightarrow \text{the answer} //$$

c) # of data points = # of scan windows = $W \times H \times \# \text{ of scales}$

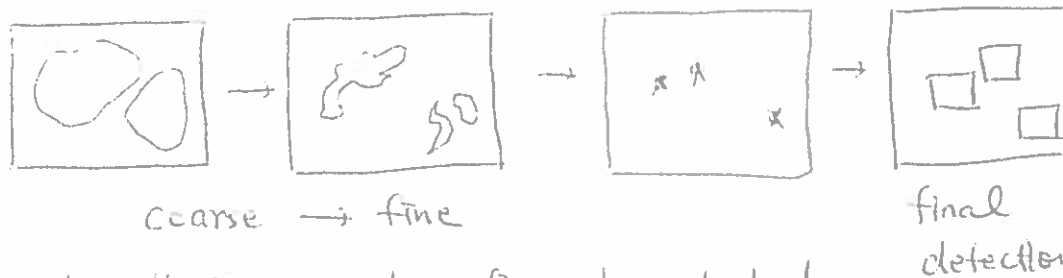
[5] this number is typically huge. we are given only a little amount

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

We have a few positive samples and all the rest ($W \times H \times \# \text{ of } s$) as negative samples from an image. \rightarrow need a high precision-recall otherwise often miss target objects or encounter too many false alarms. Boosting cascade has multiple boosting classifiers in a



coarse-to-fine manner in an imbalanced tree structure $Y_m(x)$, $M = 2, \rightarrow 5, 10, 20, \dots 100$ e.g



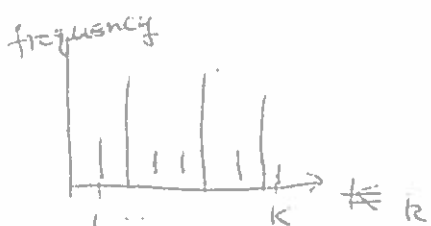
It applies a coarse to all pixels, then finer to selected pixels, rejecting majority of negative samples quickly, accelerating the overall run-time. //

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. If K is too large, overfitting happens and memory, time-complexity issues. (under-representation) (over-representation)

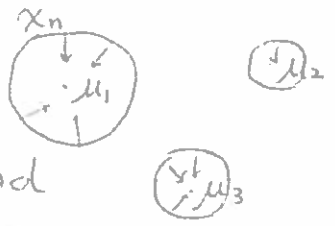
The output is the codebook of size K , i.e. $\mu_k, k=1, \dots, K$. //

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 and assigned to its nearest μ_k . The corresponding histogram bin is increased. $(k=1, \dots, K)$

Thus costs $O(KND)$. //



b-i) r_{nk} is the membership indicator variable. If x_n is assigned 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: $x_n, n=1, \dots, N'$, K manually set, initial $\mu_k, k=1, \dots, K$

Output: $r_{nk}, n=1, \dots, N', k=1, \dots, K, \mu_k, k=1, \dots, K$

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

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

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

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

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

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

[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: $\pi_k, \mu_k, \Sigma_k, k=1, \dots, K$.

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.

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

$$[8] \quad a-ii) \quad E(w) = \frac{1}{2} \sum_{n=1}^N \{ y(x_n, w) - t_n \}^2 //$$

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

$$[5] \quad 6 \quad E = \frac{1}{2} \sum_{n=1}^N (w_0 + w_1 x_n - t_n)^2 = \frac{1}{2} \sum (w_0^2 + w_1^2 x_n^2 + t_n^2 + 2w_0 w_1 x_n - 2w_0 x_n t_n - 2w_1 x_n t_n)$$

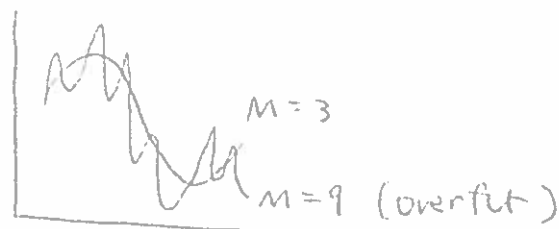
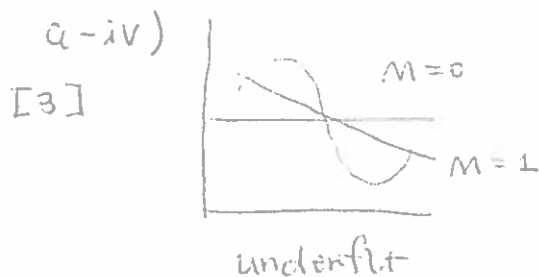
$$\frac{\partial E}{\partial w_0} = N w_0 + w_1 \sum_{n=1}^N x_n - \sum_{n=1}^N t_n = 0 //$$

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

solving the linear eqn system.

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

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

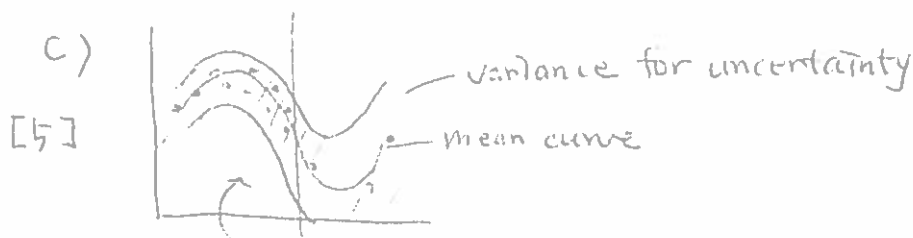


$$a-v) \quad E(w) = \frac{1}{2} \sum_{n=1}^N \{ y(x_n, w) - t_n \}^2 + \frac{\lambda}{2} \|w\|^2 //$$

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

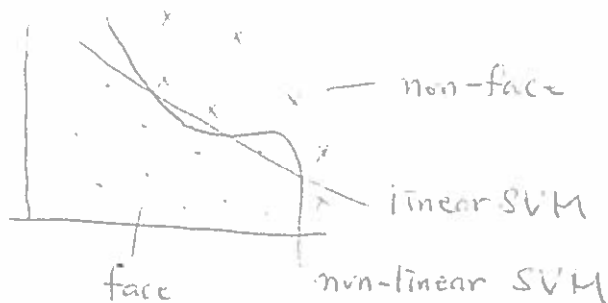
$$b-i) \quad m(p(t_{N+1}|t)) = k^T C_N^{-1} t$$

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



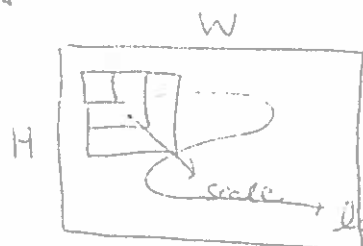
we have more data points here thus less variance.

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



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

new data points
training data point



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

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

Linear: $k(x, x_n) = x^T x_n \rightarrow y(x) = x^T \left(\sum_n a_n t_n x_n \right) + b$

$\rightarrow O(D=400)$ pre-computed

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

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

b-i) $z = U^T x$, where $U = [u_1 u_2 \dots u_M]$

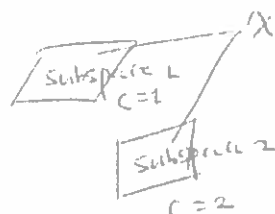
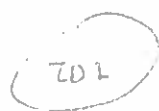
$M \times 1 \quad M \times D \quad D \times 1 \quad D \times M$

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

$D \times 1 \quad M \times 1 \quad M \times M \quad M \times 1$

b-iii)

[7]



assign arg min $\|x - \hat{x}_c\|$