Face detection and Recognition

→ Face detection < Integral images
Ada-boost classification algorithm. →Face Recognition - Eigen face.

→ To detect face detection, the idea is to use sliding windows. step1 is to define a rectangle size and slide it horizontally throughout the image. step 2 defines if there is an face in that rectangle or not. We can have different sizes of the rectangle.

→ There can be times when in an image there would be more places where nothing is present (i.e. negative windows). Our whole idea is to spend as little time as possible on the negative windows for computational efficiency.

→ How we ~~say~~ classify that in an image what is face and what is not? By defining feature space (by having discriminative classification).

→ One prominent detector is the Viola/Jones detector. It's main features :-

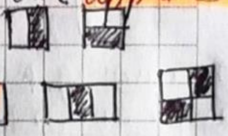
① Training is slow, but detection is fast.

3 main ideas {
② Integral images for fast feature evaluation
③ Boosting for feature selection.
④ Attentional cascade for fast rejection of non-face windows.

→ Image features are also called "Haar-like" features because they are similar to 2D Haar wavelets. Image features are "Rectangle filters" which are applied throughout the image.

$$\boxed{\text{Value} = \Sigma(\text{pixels in white area}) - \Sigma(\text{pixels in black area})}$$

of Rectangle



Different kinds of rectangles can be defined. This is, however time consuming.

→ For fast computation, Viola/Jones introduced the concept of "Integral images". The integral image computes a value at each pixel (x,y) that is the sum of pixel values above and to the left.

Eg.

0.1	0.1	0.2	0.1	0.7	0.1
0.2	0.3	0.2	0.7	0.8	0.2
0.1	0.4	0.3	0.3	0.1	0.3
0.1	0.5	0.1	0.1	0.2	0.8
0.1	0.8	0.8	0.5	0.6	0.5

original image

→

0.1	0.2	0.4	0.5	1.2	1.3
.3	.7	1.1	1.9	3.4	3.7
.4	1.2	1.9	3	4.6	5.2
.5	1.7	2.5	3.7	5.3	6.7
.6	2.3	3.9	5.6	8	9.9

integral image

Cumulative row sum : $s(x,y) = s(x-1,y) + i(x,y)$

Integral image

$ii(x,y) = ii(x,y-1) + s(x,y)$

→ Scaling is not required in integral ~~image~~ Image because it enables us to evaluate all the rectangle sizes in constant time. Alternatively, we can scale the rectangular features instead.

→ AdaBoost → focusses on Adaptive Boosting. It is a learning algorithm. Also focusses on combining the weak classifiers to make a strong classifier.

Boosting → is a classification scheme which works by combining weak learners to ensemble classifier. The learner has to do better than chance (Δ). Consist multiple boosting rounds, in which we select a weak learner that does better perform the examples "Hardness" is captured by weights attached to the training examples.

→ AdaBoost concept → ① combining weak classifiers (which are better than random classifier)

$$\underbrace{\left.\begin{array}{l} h_1(x) \in \{-1,+1\} \\ h_2(x) \in \{-1,+1\} \\ \vdots \\ h_T(x) \in \{-1,+1\} \end{array}\right\}}_{W.C} \quad \underbrace{\boxed{H_T(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)}}_{SC}$$

→ Boosting is a sequential procedure.

① weak learners from the family of lines

② Each data point has a class label Eg. $y_t = \begin{cases} +1 (\bullet) \\ -1 (\circ) \end{cases}$ and a weight $W_t = 1$

③ we define a line that has a chance of 50%.

④ we, then modify our classifier by changing the line position, and each doing better than chance. we finally found one.

⑤ we accept this and change the problem by changing the weight $w_t = w_t$ exp

⑥ we then repeat the steps until we get a good approximation.

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

→ AdaBoost Algorithm 3 steps

→ Weak learners for face detection

$$\underset{\substack{\downarrow \\ \text{window}}}{h_t(x)} = \left\{\begin{array}{ll} 1 & \text{if } \overset{\text{value of rectangle feature}}{f_t(x)} > \overset{\text{threshold}}{\theta_t} \\ 0 & \text{otherwise} \end{array}\right\}$$

→ Boosting method for face/Nonface examples →

① Training set consists of face and non-face examples [initially with equal weights]

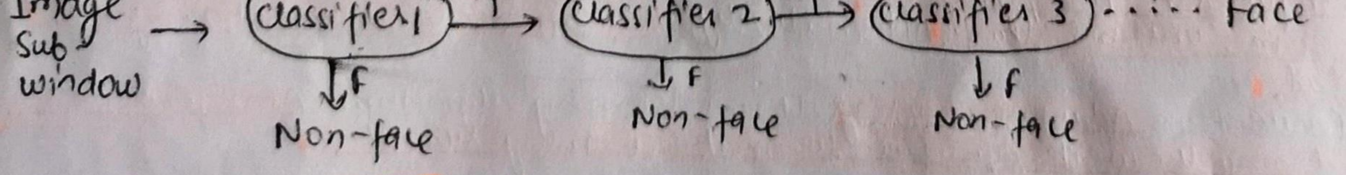
② for each round of Boosting:
- → Evaluate each rectangle filter on each example.
- → select best threshold for each filter.
- → select best filter/threshold combination.
- → Reweight examples.

③ computational complexity = O(MNK) M → rounds, N → examples, K = features

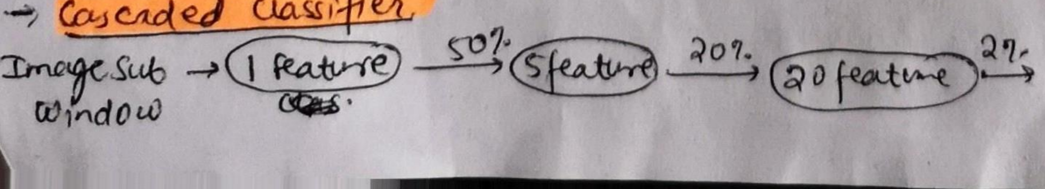
→ Attentional Cascading → ① we start with simple classifiers which reject many of the negative sub-windows while detecting almost all positive sub-windows.

② Positive response from the 1st classifier triggers the evaluation of a second (more complex) classifier, so on.

③ A negative outcome at any point leads to immediate rejection of the sub-window.



→ Cascaded classifier



2DIP → Adaboost and Eigenfaces

Training the cascade:

① set target detection and false positive rates for each stage.

② Keep adding features to the current stage until it's target rates have been met

→ Need to lower AdaBoost threshold to maximise detection.

→ Test on a validation set.

③ If the overall false positive rate is not low enough, then add another stage.

④ use false positives from current stage as the negative training eg. for the next stage.

Face Recognizing

challenges in Recognizing faces in an image are result of changes in expression, lighting, age, occlusion and viewpoint.

Some basic approaches to recognise face includes:-

(i) Project into a new sub-space (Eg. Eigenfaces = "PCA")
(ii) measure face features.
(iii) make 3d face model, compare shape + Appearances (e.g. AAM")

Simple approach for face recognition

(i) Treat face image as a vector of intensities □ → X

(ii) Recognize face by nearest neighbour in database $y_1 \cdots y_n$

$$\boxed{K = \underset{(K)}{\operatorname{argmin}} \| y_K - X \|}$$

problems:

① Scaling. (if an image would be zoomed, pixel values would change)

Eigenfaces idea → Construct a low-dimensional linear sub-space that best explains the variation in the set of face images.

PCA → Principal component analysis → The projection of x on the direction of u = $z = u^T x$

ow → Direction u that maximizes the variance of the projected data:

maximize → $\frac{1}{N} \sum_{i=1}^{N} \underbrace{u^T(x_i - \mu)}_{\text{Projection of data point}} (u^T(x_i - \mu))^T$ { Subject to $\|u\| = 1$

$$= \underbrace{u^T \left[\sum_{i=1}^{N} (x_i - \mu)(x_i - \mu)^T \right]}_{\text{covariance matrix of data}} u = \boxed{u^T \Sigma u}$$

e direction that maximizes the covariance is the eigenvector associated with the rgest Eigenvalue of Σ

Eigenfaces : idea → ① Assume that most face images lie on a low-dimensional subspace determined by the first K (K<d) directions of maximum variance

② use PCA to determine the vectors or " eigenfaces" $u_1 \dots u_k$ that span that subspace.

③ Represent all face images in the dataset as the linear combination of eigenfaces.

→ Recognition with Eigenfaces :-

① Process labelled training images:

① find mean(μ) and covariance matrix (Σ)

② Find K principle components (eigenvectors of Σ)

③ Project each training image (x_i) onto subspace spanned by principal components:

$$(w_{i1}, \dots w_{ik}) = (u_1^T(x_i-\mu), \dots u_k^T(x_i-\mu))$$

② Given novel image x:

① Project onto subspace $(w_1, \dots w_k) = (u_1^T(x-\mu), \dots u_k^T(x-\mu))$

② classify as closest training face in k dimensional sub-space.

→ limitations of pc

① we need a properly cropped image for overlapping.

② PCA assumes that the data has a gaussian distribution (mean μ, covariance matrix Σ)

③ the direction of maximum variance is not always good for classification.

→ The top k orthogonal direction that captures the most variance is the k eigenvecto associated with with the k largest eigenvalues.