- **INTEGRAL IMAGE** is a matrix representation that holds global image information [Viola and Jones 01]
- values $ii(i, j)$ ... sums of all original image pixel-values left of and above (i, j):

$$ii(i, j) = \sum_{k \leq i, l \leq j} f(k, l) \,, \tag{4.1}$$

- efficiently computed in a single image pass using recurrences

Algorithm 4.2: Integral image construction

1. Let $s(i, j)$ denote a cumulative row sum, let $s(i, -1) = 0$.
2. Let $ii(i, j)$ be an integral image, let $ii(-1, j) = 0$.
3. Using a single row-by-row pass through the image, for each image pixel (i, j) calculate the cumulative row sums $s(i, j)$ and the integral image value $ii(i, j)$ using the recurrences

$$s(i, j) = s(i, j - 1) + f(i, j) \,, \tag{4.2}$$
$$ii(i, j) = ii(i - 1, j) + s(i, j) \,. \tag{4.3}$$

4. After reaching the lower right image corner pixel after a single pass through the image, the integral image ii is constructed.

- main use of integral image data structures is in rapid calculation of simple rectangle image features at multiple scales
- used for rapid object identification
- and for object tracking
- any rectangular sum can be computed using four array references
- a feature reflecting a difference between two rectangles requires eight references
- adjacent two-rectangle features require only six array references since the rectangles are adjacent
- three- and four-rectangle features of Figure 4.2c,d can be calculated using eight and nine references to the integral image values, respectively
- rectangle features can be computed extremely efficiently and in constant time once the integral image is formed

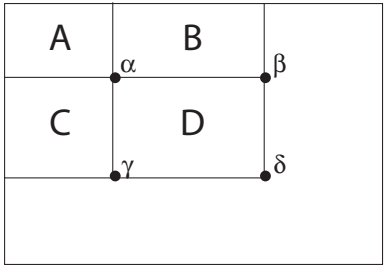

Figure 4.1: Calculation of rectangle features from an integral image. The sum of pixels within rectangle D can be obtained using four array references. $D_{sum} = ii(\delta) + ii(\alpha) - (ii(\beta) + ii(\gamma))$, where $ii(\alpha)$ is the value of the integral image at point α, i.e., the sum of image values within the rectangle A. Similarly, the value $ii(\beta)$ is the sum of values in rectangles A and B, etc.

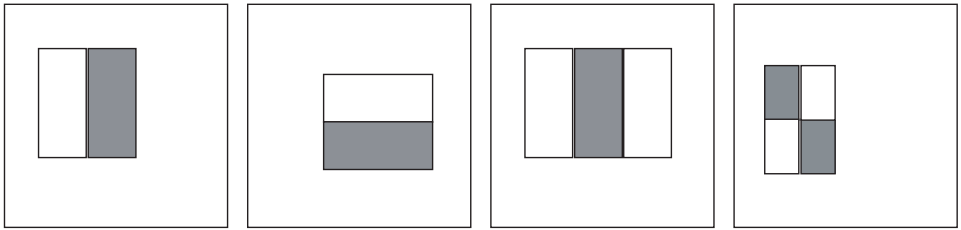

Figure 4.2: Rectangle-based features are calculated from an integral image. These features are calculated by subtraction of the sum of the shaded rectangle(s) from the non-shaded rectangle(s). The figure shows (a,b) two-rectangle, (c) three-rectangle, and (d) four-rectangle features. The four-rectangle features can be computed in two diagonal directions. The sizes of the individual rectangles can be varied to yield different features as well as features at different scales. The contributions from the shaded and non-shaded regions can be normalized to account for possibly unequal region sizes.

8.6 Boosting in pattern recognition

- single classifier rarely solve a problem completely, or even 'well enough'
- it is common to combine a number of independent classifiers to improve overall performance
- individual classifiers may be very weak [or **base**] in isolation
  (that is, in a two-class problem, a classifier may perform little better than 50%)
- simple rules are applied in turn, each time working with a different subset of the training examples
- achieving improved performance of any given learning algorithm is called **boosting** and is general
- after many rounds of calling these weak classifiers, the **boosting algorithm** combines the weak rule outcomes in a single classification rule that is much more accurate than any of the constituent weak rules

Questions to be solved:

- how to select training subsets submitted to the individual weak classifiers
- how to combine weak rules in a single **strong** rule
- $\Rightarrow$ a generally accepted approach ... let the weak classifiers function sequentially and place the most weight on "difficult" training examples, i.e., those that were misclassified in the previous round(s) of applying the other weak classifiers
- to combine the weak rules in a single strong rule ... a weighted majority vote of weak classifier outputs is an obvious strategy
- boosting produces very accurate classifications by combining classifications, which are only moderately accurate

- pattern space X
- training set of m patterns $\mathbf{x}_i$
- their corresponding class identifiers ω_i
- and assuming two-class classification ($\omega_i \in \{-1, 1\}$)
- weak classifiers W_k applied to training set in which importance of correctly classifying individual examples varies step by step
- in each step k, this importance is specified by a set of weights $D_k(i)$, so that $\sum_{i=1}^{m} D_k(i) = 1$
- initially, the weights are set equally but for each next step $k+1$, the weights of examples incorrectly classified in step k are increased (in a relative sense)
- consequently, the weak classifier W_{k+1} concentrates on the difficult examples, which were not correctly classified in the previous round(s)

Algorithm 8.3: AdaBoost

1. Initialize K, the number of weak classifiers to employ
2. Set $k = 1$, and initialize $D_1(i) = 1/m$.
3. For each step k, train a weak classifier W_k using the training set with a set of weights $D_k(i)$, so that a real number is assigned to each pattern $\mathbf{x}_i$; $W_k : X \to \mathcal{R}$.
4. Choose $\alpha_k > 0 \in \mathcal{R}$.
5. Update
$$D_{k+1}(i) = \frac{D_k(i) e^{-\alpha_k \, \omega_i \, W_k(\mathbf{x}_i)}}{Z_k} \, , \tag{8.6}$$
where Z_k is a normalization factor chosen so that $\sum_{i=1}^{m} D_{k+1}(i) = 1$.
6. Set $k = k + 1$.
7. If $k \leq K$, return to step 3.
8. The final strong classifier S is defined as
$$S(\mathbf{x}_i) = \text{sign} \left(\sum_{k=1}^{K} \alpha_k \, W_k(\mathbf{x}_i) \right) \, . \tag{8.7}$$

Notice at step 5 that the exponent is positive for misclassifications, lending more weight to the associated $D(i)$.

- In each step, the weak classifier W_k needs to be determined so that its performance is appropriate for the training set with the weight distribution $D_k(i)$.
- In the dichotomy classification case, the weak classifier training attempts to minimize the objective function ϵ_k

$$\epsilon_k = \sum_{i=1}^{m} P_{i\sim D_k(i)} \left[W_k(\mathbf{x}_i) \neq \omega_i \right], \tag{8.8}$$

  $P[\cdot]$ ... empirical probability observed on the training sample
- the error ϵ_k is calculated with respect to the weight distribution D_k (characterized as the sum of probabilities $P_{i\sim D_k(i)}$ in which the weight distribution $D_k(i)$ is considered together with the classification correctness achieved on the training patterns $\mathbf{x}_i$)
- misclassification of training patterns $\mathbf{x}_i$ for which $D_k(i)$ is low (patterns correctly classified in the previous weak classifier steps) increases the error value less than the misclassification of the patterns of focus of the weak classifier W_k
- $\Rightarrow$ individual weak classifiers are trained to classify different portions of the training set better than randomly

- value of α_k in step 5 can be – for a two-class classification problem

$$\alpha_k = \frac{1}{2} \ln \left(\frac{1 - \epsilon_k}{\epsilon_k} \right) \tag{8.9}$$

- behavior of the final strong classifier S is determined by the weighted majority vote of all K weak classifiers considering the classifier-specific weights α_k

- AdaBoost algorithm can achieve a classification accuracy that is arbitrarily close to 100%
  (as long as each of the weak classifiers is at least slightly better than random and assuming availability of sufficient training data)
- AdaBoost's ability to convert a weak learning algorithm into a strong learning algorithm has been formally proven

In addition to an ability to learn from examples, the ability to generalize and thus correctly classify previously unseen patterns is of basic importance. Theoretical considerations suggest that Adaboost may be prone to overfitting, but experimental results show that typically it does not overfit, even when run for thousands of rounds. More interestingly, it was observed that AdaBoost sometimes continues to drive down the classification error long after the training error had already reached zero (see Figure 8.12).

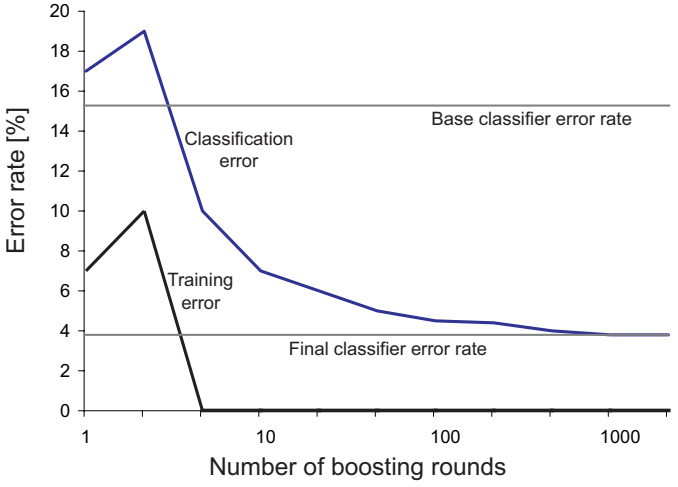

Figure 8.12: AdaBoost—observed training and testing error rate curves as a function of the number of boosting rounds. Note that the testing error keeps decreasing long after the training error has already reached zero. This is associated with a continuing increase in the margin that increases the overall classification confidence with additional rounds of boosting. For comparison, the horizontal lines indicate the error rate of the base and final classifiers.

10.6 Boosted cascade of classifiers for rapid object detection

- rapid face detection led to a framework for general object detection and object tracking tasks [Viola and Jones 01]
- first stage
  - new image representation called an **integral image** allows fast computation of many simple image-based features (Section 4.2.1)
  - number of calculated features is far larger than the number of image pixels
  - over-complete representation results
  - subset of best features needs to be identified
- second stage
  - learning based on AdaBoost (Section 8.6) yields small number of well distinguishing features
  - and a set of efficient classifiers
  - feature selection performed using a modified AdaBoost algorithm in which the weak classifiers can only depend on a single feature [Tieu and Viola 04]
- third stage
  - classifiers are ordered in a cascade sequence starting with simple and thus fast classifiers used for quickly rejecting object-detection hypotheses, to

  employing more complex and therefore more powerful but slower classifiers that are applied only to the remaining, not-yet rejected hypotheses
- this strategy dramatically improves object detection speed.

- approach does not work directly with image intensity data
- *integral image* is used to quickly calculate responses of simple region-based filters at many scales
- specific filters are object dependent and can be used to encode problem-specific knowledge
- rapid computation of rectangle features – based on the integral image (Section 4.2.1 (see Figure 4.2))
- rectangular features computed in constant time using the integral image
- in one application ... three kinds of features were used for face detection calculated from two, three, or four rectangle configurations (Figure 4.2)
- despite the obvious simplicity
  despite the small number of directions
  features are very sensitive to edges, bars, and other simple image structures
- multi-scale rectangle feature determination provides a rich feature set facilitating effective learning

- large feature set available together with a training set of p positive and q negative examples (assuming a two-class problem)
- only a small number of these features can be used in combination to yield an effective classifier
- a single rectangle feature is first selected using a weak learning approach to best separate the positive and negative examples
- followed by additional features identified by the iterative boosting process
- for each selected feature, the weak learner finds an optimal threshold minimizing the number of misclassified examples from the training set

- each weak classifier is thus based on a single feature f_j and a threshold t_j

$$\begin{aligned} h_j(\mathbf{x}) = &\quad 1 \quad \text{if} \quad p_j f_j(\mathbf{x}) < p_j t_j \,, \\ &= -1 \quad \text{otherwise,} \end{aligned} \tag{10.26}$$

  p_j is a polarity indicating the direction of the inequality sign
  $\mathbf{x}$ is an image subwindow on which the individual rectangle features f_j are calculated

- while no single feature can typically perform the overall classification task with low error, the sequential character of feature selection means that the features picked first and their associated classifiers have a relatively high classification success on their respective training sets, say between 70% and 90%

- the classifiers trained in later rounds on the remaining more difficult examples yield classification correctness of 50–60%

Algorithm 10.11: AdaBoost feature selection and classifier learning

1. Consider a two-class problem, a training set of positive and negative examples $\mathbf{x}_i$, and their corresponding class identifiers $\omega_i \in \{-1, 1\}$.
2. Initialize K, the number of features to be identified.
3. Set $k = 1$; for each sample $\mathbf{x}_i$, initialize weights
$$\begin{aligned} w_{1,i} &= \frac{1}{2q} \quad \text{for} \quad \omega_i = -1\,, \\ &= \frac{1}{2p} \quad \text{for} \quad \omega_i = \quad 1\,. \end{aligned}$$
4. For $k \neq 1$, normalize the weights to produce a probability distribution
$$w_{k,i} := \frac{w_{k,i}}{\sum_{l=1}^{p+q} w_{k,l}}\,. \tag{10.27}$$
5. For each feature f_j, train a classifier $h_{k,j}$ restricted to using a single feature. Evaluate its classification error $\epsilon_{k,j}$ on the training set considering the current weights $w_{k,i}$ associated with each sample $\mathbf{x}_i$
$$\epsilon_{k,j} = \frac{1}{2}\sum_i w_{k,i}\big|h_j(\mathbf{x}_i) - \omega_i)\big|\,. \tag{10.28}$$

6. Select the classifier $h_{k,j}$ with the lowest error $\epsilon_{k,j}$.
7. Update the weights for all samples $\mathbf{x}_i$

$$w_{k+1,i} = w_{k,i}\beta_k^{1-E_i} , \tag{10.29}$$

where $\beta_k = \epsilon_{k,j}/(1-\epsilon_{k,j})$ and

$$\begin{aligned} E_i &= 0 \quad \text{if } \mathbf{x}_i \text{ is classified correctly,} \\ &= 1 \quad \text{otherwise.} \end{aligned}$$

8. Set $k := k+1$.
9. If $k \leq K$, return to step 4.
10. The final strong classifier $S(\mathbf{x}_i)$ is defined as

$$\begin{aligned} S(\mathbf{x}_i) = \quad & 1 \quad \text{for } \sum_{k=1}^{K} \alpha_k h_{k,j}(\mathbf{x}_i) \geq \frac{1}{2}\sum_{k=1}^{K}\alpha_k , \\ = \quad & -1 \quad \text{otherwise,} \end{aligned}$$

where $\alpha_k = \log(1/\beta_k)$ and j denotes the single features, which are used in the K weak classifiers $h_{k,j}$, respectively.

The weighting update in step 7 means that any sample classified correctly in this round is not considered when determining the classification correctness in the next round.

- ... originally developed for frontal human face recognition
- the first and second features selected are easily interpretable
- first feature – eye region
- second feature – eyes are usually darker compared to the nose bridge

(a) (b) (c) (d) (e)

Figure 10.27: Two most significant feature determined by Algorithm 10.11 for face detection in subwindows of 24 × 24 pixels [Viola and Jones 01]. (a) Original image. (b) First most distinguishing feature corresponds to the frequent case when the eye region is darker than the region of the nose and cheeks. (c) The first feature overlaid over the original image. (d) Second most distinguishing feature is in agreement with the observation that the eyes are darker compared to the nose bridge located between them. (e) The second feature overlaid.

- previous step identifies the most distinguishing features
- $\Rightarrow$ cascade of classifiers to decrease processing time and increase performance
- early-stage simple classifiers are set so that their false negative rates (number of missed detections) is close to zero
- price paid for such behavior is an increase in the false positive rate
- but ... simpler early stage classifiers are used to quickly reject the majority of candidate locations (subwindows in which features are computed)
- increasingly more complex classifiers are employed in the locations that remain unrejected
- ultimately, the remaining non-rejected locations are marked as the locations of identified objects.

Figure 10.28:

- this cascade of classifiers ... a degenerate decision tree
- each classifier stage $n + 1$ is only called if the classifier n has not rejected the candidate location
- classifiers in all individual stages are trained using AdaBoost and adjusted to minimize false negatives

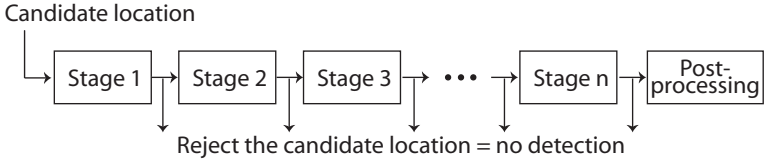

Figure 10.28: Detection cascade applied to each analyzed location–image subwindow. The early-stage simple classifiers reject less likely locations while maintaining a very low false negative rate. The later-stage more complex classifiers eliminate more and more false positives while they are set not to reject true positives.

Face detection

- powerful first stage classifier can be constructed from the two features identified above (Figure 10.27)
- this classifier detects 100% of face objects with about 40% false positives
- later-stage classifiers have increasingly higher false-positive rates
- since they are only applied to the subset of already-identified high-likelihood locations, the likelihood of invoking these classifiers is decreasing with the increased depth in the decision tree
- additional cascade stages added until the overall detection performance is sufficient
- since features are evaluated at multiple scales, subwindows are allowed to overlap – there may be overlapping multiple detections for each detected object
- multiple detections must be postprocessed to yield a single final detection per location

Figure 10.29: Some of the training face images used in the face detection system based on the boosted cascade of classifiers. *Courtesy of P. Viola, Microsoft Live Labs and M. Jones, Mitsubishi Electric Research Labs, ©2001 IEEE [Viola and Jones 01].*

Face detection

- 38 stages + over 6,000 features.
- required about 10 feature evaluations per subwindow
- high detection speed even when tested on difficult data with 75 million subwindows and over 500 faces present
- the approach itself is general and can be used for a variety of object detection and recognition tasks

(a) (b)

Figure 10.30: Example results of face detection using the described method of the boosted cascade classifiers. Each detected face is identified by an overlaying rectangle. *Courtesy of P. Viola, Microsoft Live Labs and M. Jones, Mitsubishi Electric Research Labs, ©2001 IEEE [Viola and Jones 01].*