Reconnaissance d'objets et vision artificielle

http://www.di.ens.fr/willow/teaching/recvis09

Lecture 6

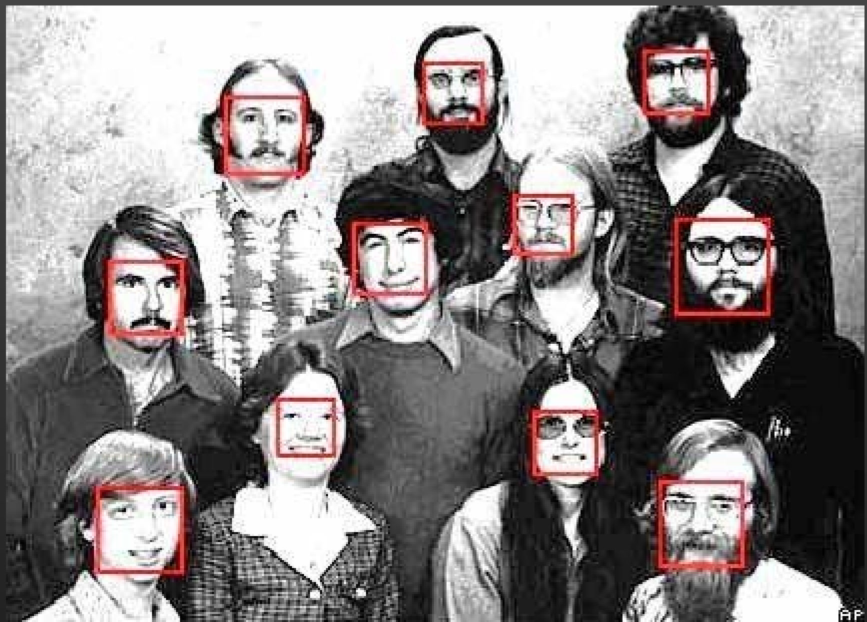
- Face recognition
- Face detection
- Neural nets

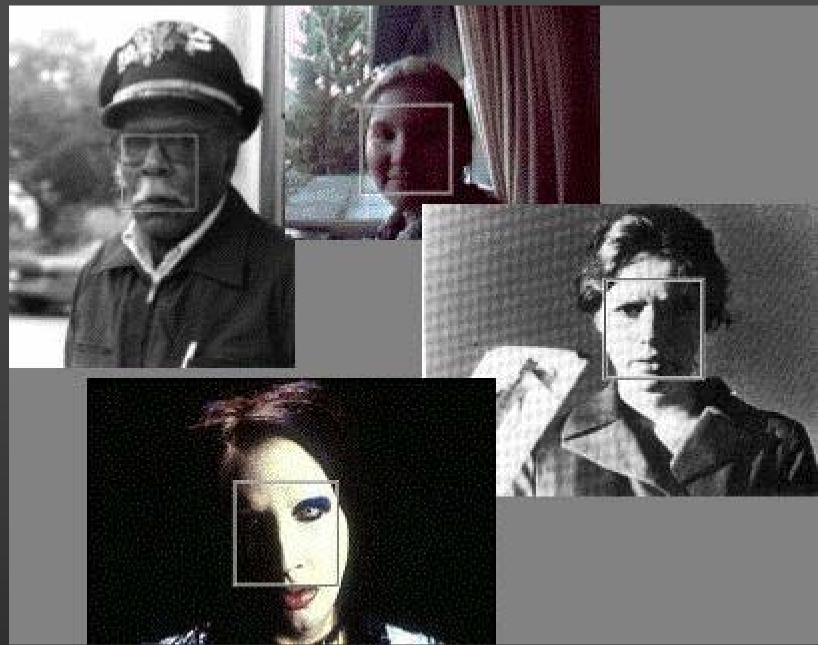
Attention!

Troisième exercice de programmation du le 24 novembre

http://www.di.ens.fr/willow/teaching/recvis09/assignment3/

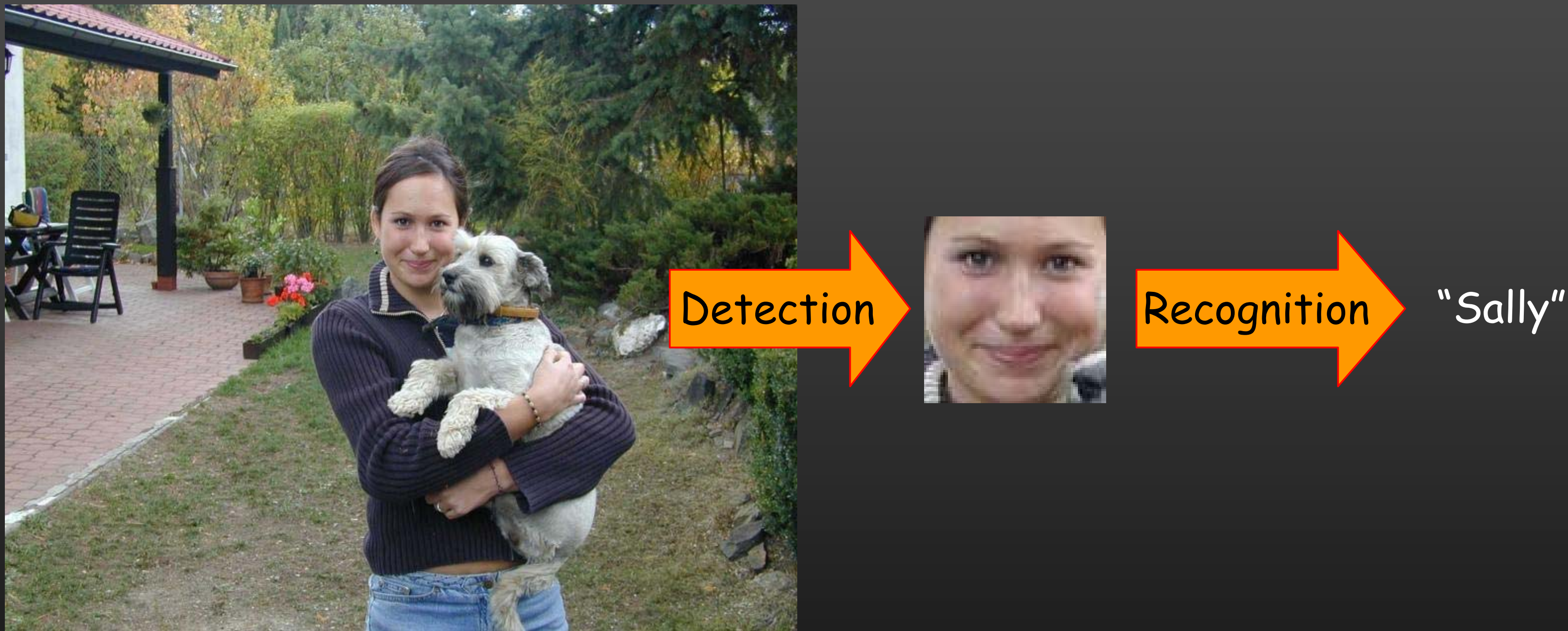
Face detection and recognition





Many slides adapted from S. Lazebnik,
K. Grauman and D. Lowe

Face detection and recognition

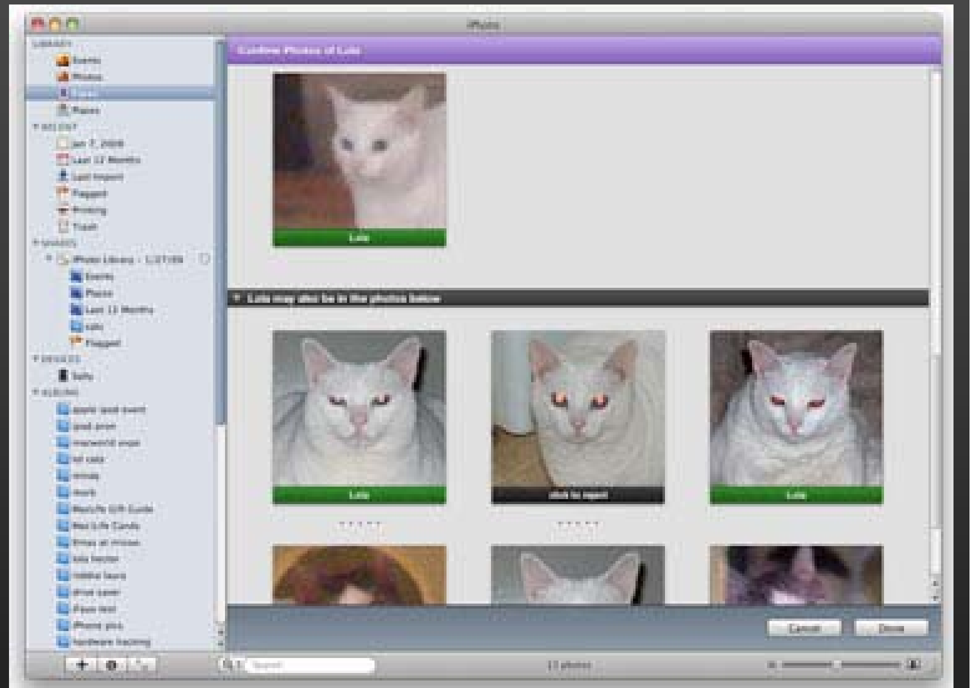


Consumer application: iPhoto 2009

http://www.apple.com/ilife/iphoto/

Consumer application: iPhoto 2009

Can be trained to recognize pets!



http://www.maclife.com/article/news/iphotos_faces_recognizes_cats

Consumer application: iPhoto 2009

[Things iPhoto thinks are faces]

History

- Early face recognition systems: based on features and distances
  Bledsoe (1966), Kanade (1973)
- Appearance-based models: eigenfaces
  Sirovich & Kirby (1987), Turk & Pentland (1991)
- Real-time face detection with boosting
  Viola & Jones (2001)

Outline

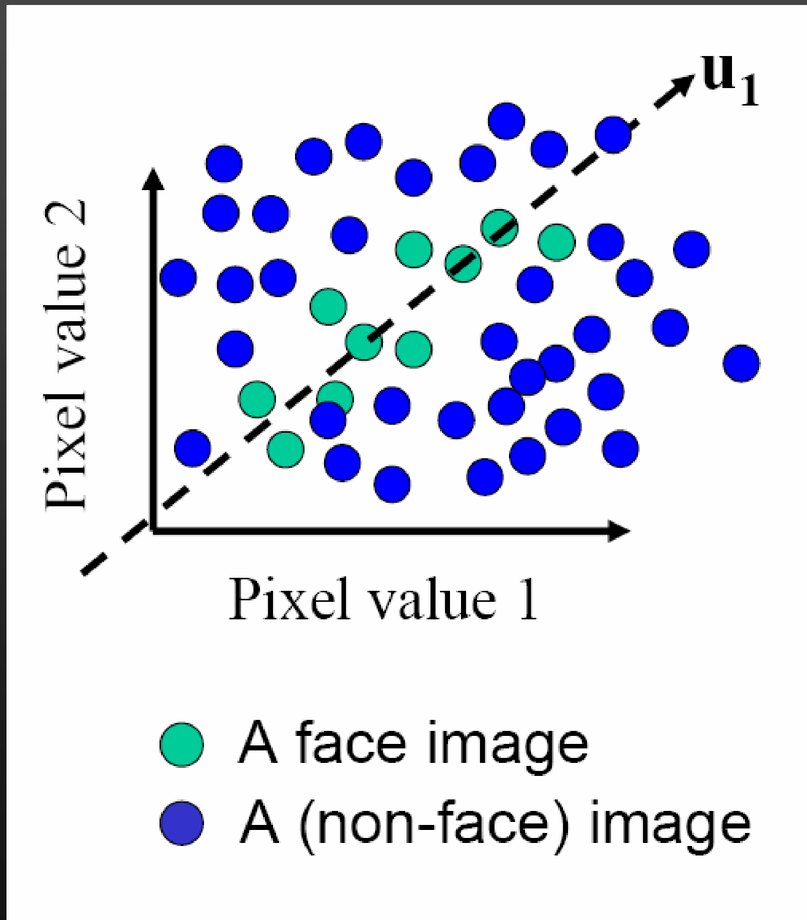
- Face recognition
  - Eigenfaces
- Face detection
  - The Viola & Jones system

The space of all face images

- When viewed as vectors of pixel values, face images are extremely high-dimensional
  - 100x100 image = 10,000 dimensions
- However, relatively few 10,000-dimensional vectors correspond to valid face images
- We want to effectively model the subspace of face images

The space of all face images

- We want to construct a low-dimensional linear subspace that best explains the variation in the set of face images



Principal Component Analysis

- Given: N data points $\mathbf{x_1}, \dots, \mathbf{x_N}$ in R^d

- We want to find a new set of features that are linear combinations of original ones:

$$u(\mathbf{x}_i) = \mathbf{u}^T(\mathbf{x}_i - \boldsymbol{\mu})$$

  ($\boldsymbol{\mu}$: mean of data points)

- What unit vector $\mathbf{u}$ in R^d captures the most variance of the data?

Principal Component Analysis

- Direction that maximizes the variance of the projected data:

$$var(u) = \frac{1}{N}\sum_{i=1}\underbrace{\mathbf{u}^{\mathrm{T}}(\mathbf{x}_i-\mu)}_{\text{Projection of data point}}(\mathbf{u}^{\mathrm{T}}(\mathbf{x}_i-\mu))^{\mathrm{T}}$$

$$= \mathbf{u}^{\mathrm{T}}\underbrace{\Big[\sum_{i=1}(\mathbf{x}_i-\mu)(\mathbf{x}_i-\mu)^{\mathrm{T}}\Big]}_{\text{Covariance matrix of data}}\mathbf{u}$$

$$= \mathbf{u}^{\mathrm{T}}\Sigma\mathbf{u}$$

The direction that maximizes the variance is the eigenvector associated with the largest eigenvalue of Σ

Eigenfaces: Key idea

- Assume that most face images lie on a low-dimensional subspace determined by the first k ($k<d$) directions of maximum variance
- Use PCA to determine the vectors $\mathbf{u}_1, \ldots \mathbf{u}_k$ that span that subspace:
  $\mathbf{x} \approx \boldsymbol{\mu} + w_1\mathbf{u}_1 + w_2\mathbf{u}_2 + \ldots + w_k\mathbf{u}_k$
- Represent each face using its “face space” coordinates $(w_1, \ldots w_k)$
- Perform nearest-neighbor recognition in “face space”

M. Turk and A. Pentland, Face Recognition using Eigenfaces, CVPR 1991

Eigenfaces example

Training images

$\mathbf{x}_1, \ldots, \mathbf{x}_N$

Eigenfaces example

Top eigenvectors:
$u_1, \ldots u_k$

Mean: μ

Eigenfaces example

- Face **x** in “face space” coordinates:

$$\mathbf{x} \rightarrow [\mathbf{u}_1^{\mathrm{T}}(\mathbf{x} - \mu), \ldots, \mathbf{u}_k^{\mathrm{T}}(\mathbf{x} - \mu)]$$

$$= \quad w_1, \ldots, w_k$$

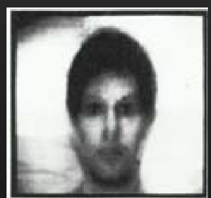
Eigenfaces example

- Face **x** in "face space" coordinates:

$$\mathbf{x} \rightarrow [\mathbf{u}_1^{\mathrm{T}}(\mathbf{x}-\mu), \ldots, \mathbf{u}_k^{\mathrm{T}}(\mathbf{x}-\mu)]$$

$$= \quad w_1, \ldots, w_k$$

- Reconstruction:

 =  + 

$$\hat{x} = \mu + w_1u_1 + w_2u_2 + w_3u_3 + w_4u_4 + \ldots$$

Summary: Recognition with eigenfaces

Process labeled training images:

- Find mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$
- Find k principal components (eigenvectors of $\boldsymbol{\Sigma}$) $\mathbf{u}_1,\ldots\mathbf{u}_k$
- Project each training image $\mathbf{x}_i$ onto subspace spanned by principal components:
  $(w_{i1},\ldots,w_{ik}) = (\mathbf{u}_1^T(\mathbf{x}_i - \boldsymbol{\mu}), \ldots , \mathbf{u}_k^T(\mathbf{x}_i - \boldsymbol{\mu}))$

Given novel image $\mathbf{x}$:

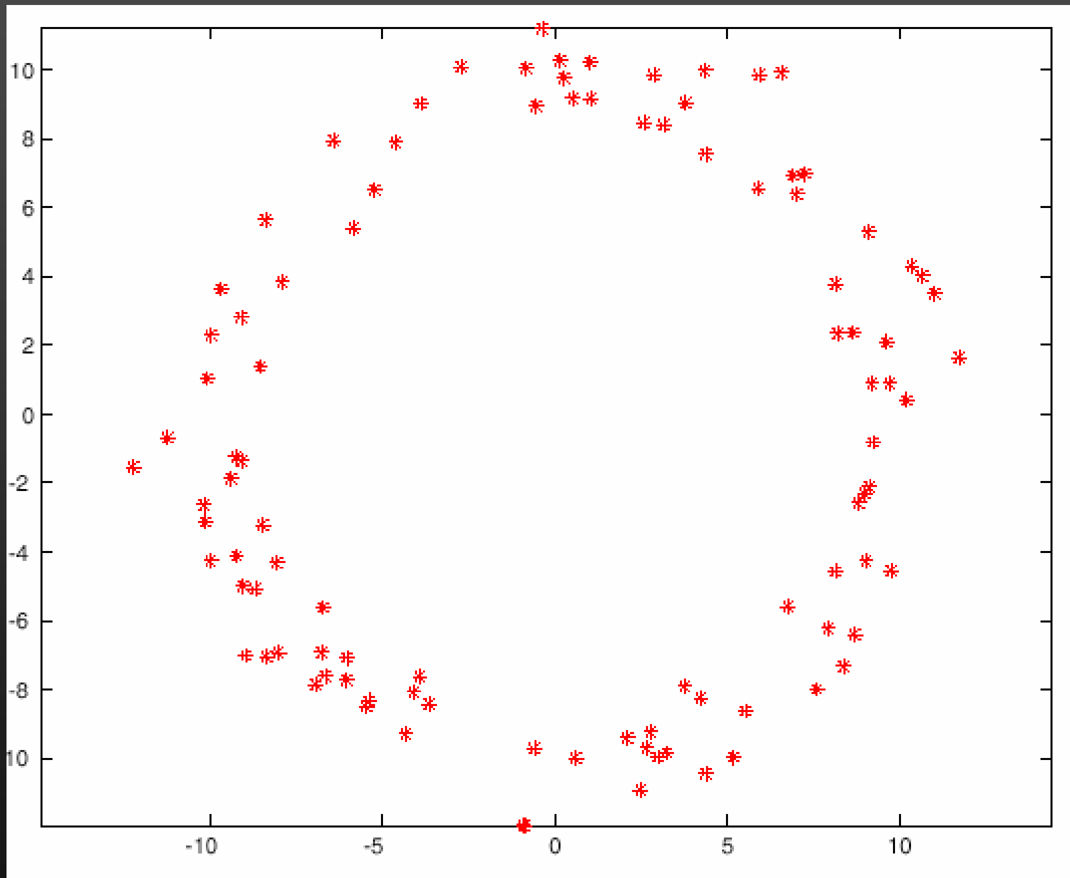
- Project onto subspace:
  $(w_1,\ldots,w_k) = (\mathbf{u}_1^T(\mathbf{x} - \boldsymbol{\mu}), \ldots , \mathbf{u}_k^T(\mathbf{x} - \boldsymbol{\mu}))$
- Optional: check reconstruction error $\mathbf{x} - \hat{\mathbf{x}}$ to determine whether image is really a face
- Classify as closest training face in k-dimensional subspace

Limitations

- Global appearance method: not robust to misalignment, background variation

Limitations

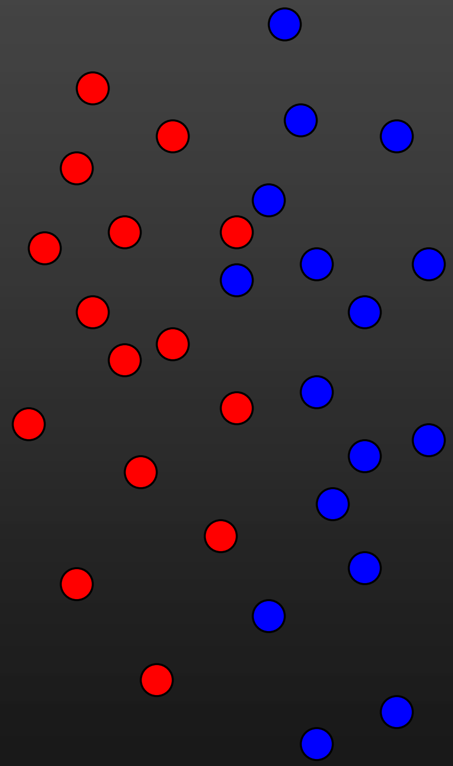
- PCA assumes that the data has a Gaussian distribution (mean μ, covariance matrix Σ)



The shape of this dataset is not well described by its principal components

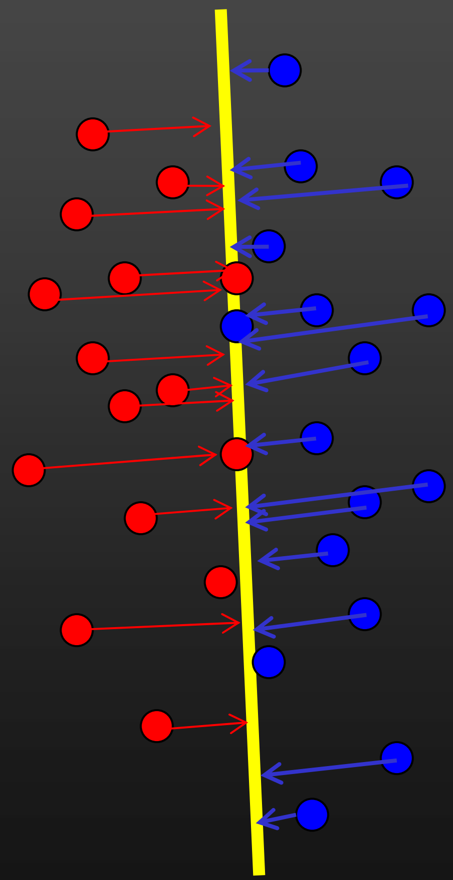
Limitations

- The direction of maximum variance is not always good for classification



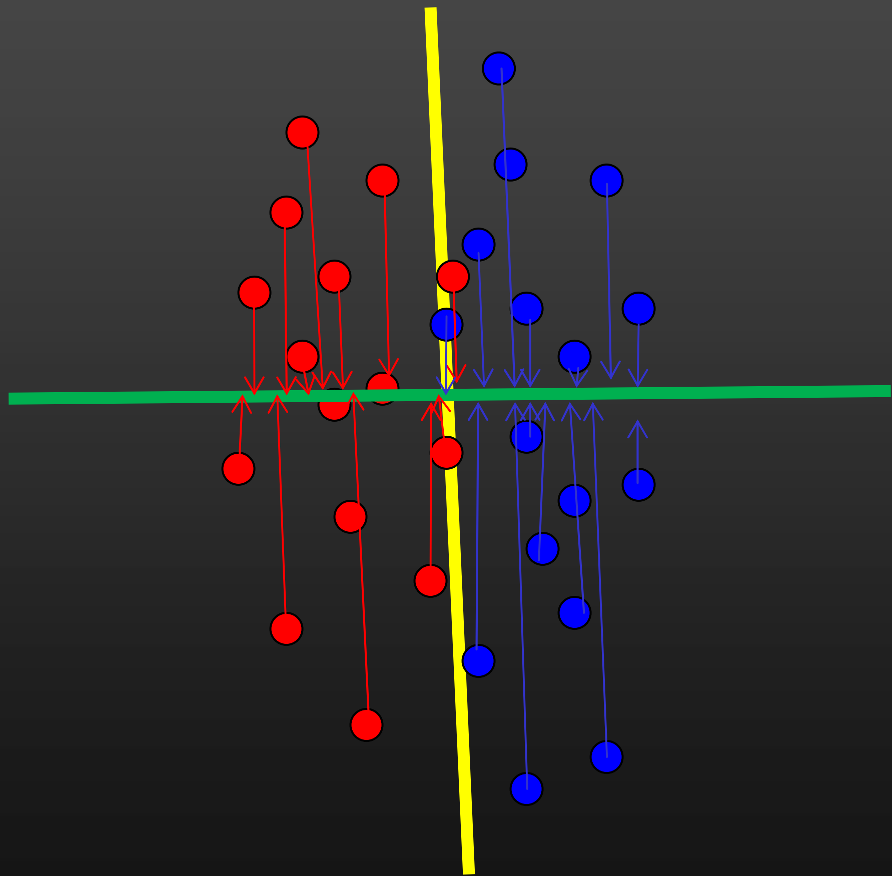
Limitations

- The direction of maximum variance is not always good for classification



Alternative (Belhumeur et al., 1997)

- Fisherfaces (aka linear discriminant analysis): Use the direction that maximizes the ratio of between-class scatter and within-class scatter



Alternative (Belhumeur et al., 1997)

- Fisherfaces (aka linear discriminant analysis): Use the direction that maximizes the ratio of between-class scatter and within-class scatter

Between-class scatter

$$S_B = \sum_{i=1}^{c} N_i (\boldsymbol{\mu}_i - \boldsymbol{\mu})(\boldsymbol{\mu}_i - \boldsymbol{\mu})^T$$

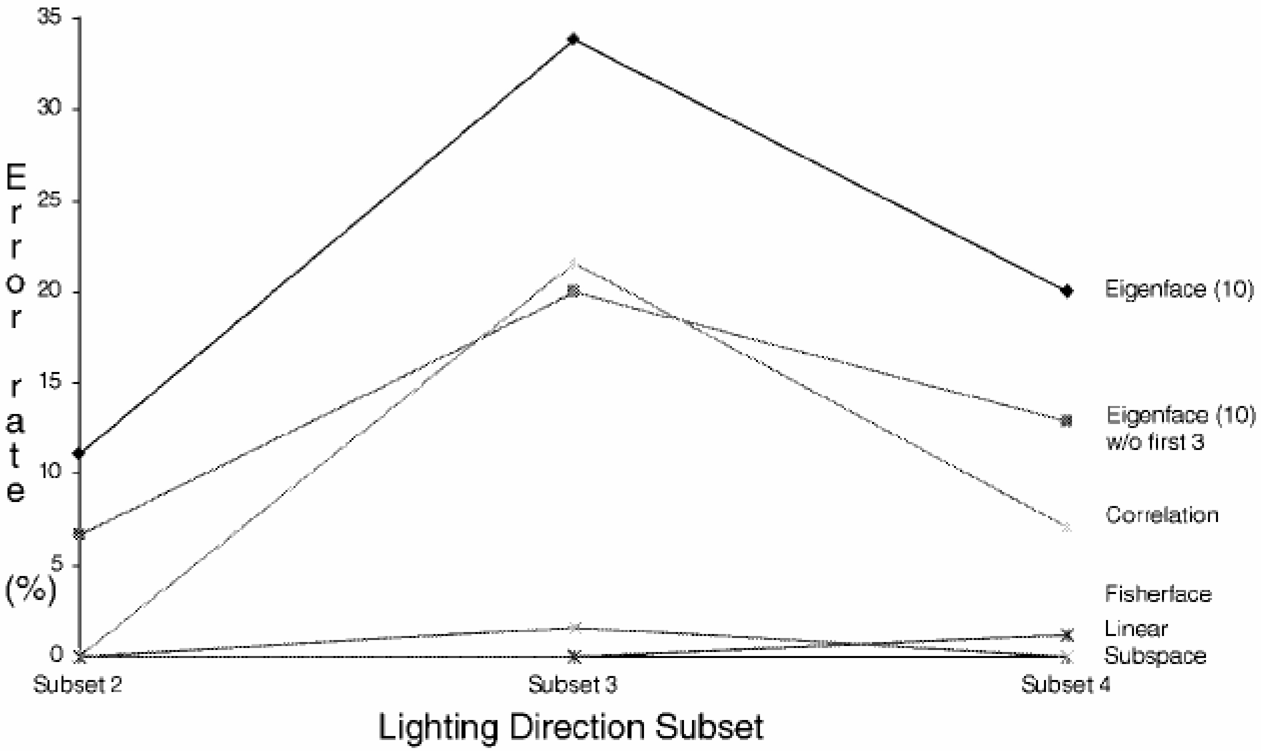
Within-class scatter

$$S_W = \sum_{i=1}^{c} \sum_{\mathbf{x}_k \in X_i} (\mathbf{x}_k - \boldsymbol{\mu}_i)(\mathbf{x}_k - \boldsymbol{\mu}_i)^T$$

Optimal direction

$$W_{opt} = \arg\max_W \frac{\left|W^T S_B W\right|}{\left|W^T S_W W\right|}$$

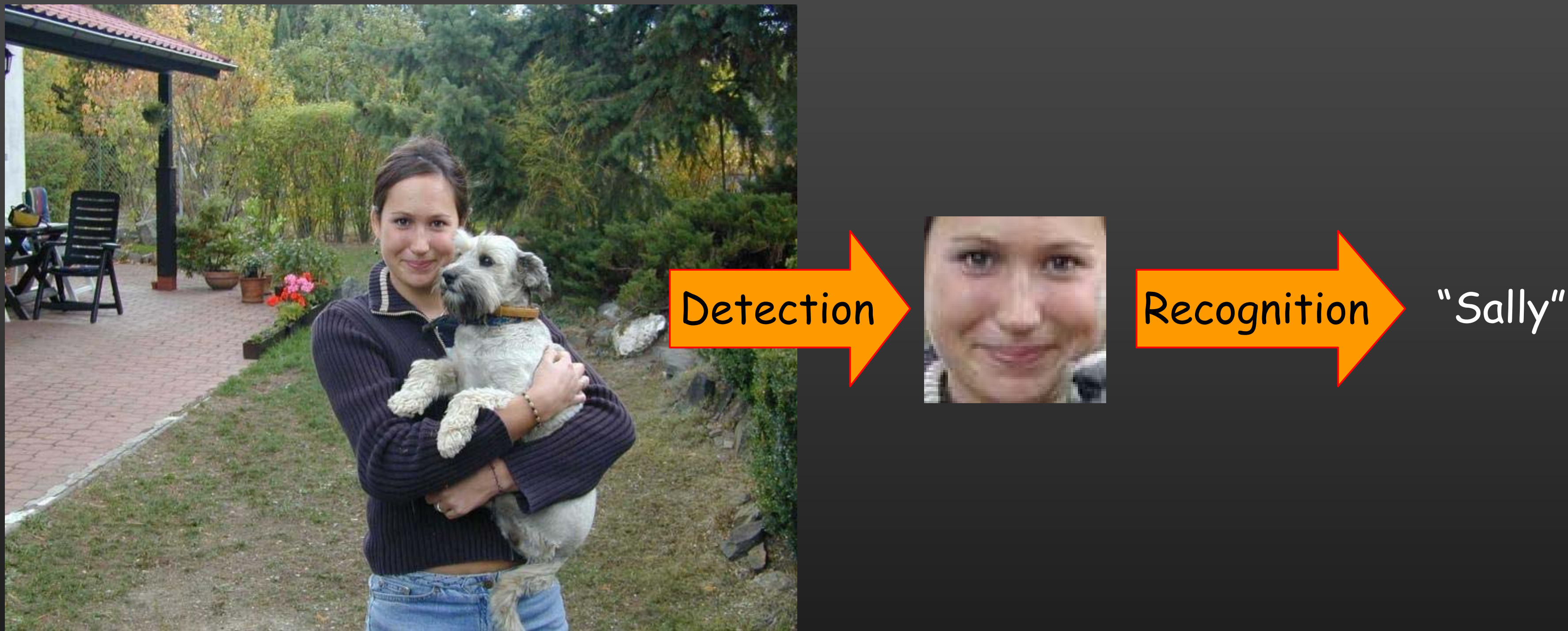
Generalized eigenvalue problem



Interpolating between Subsets 1 and 5				
Method	Reduced Space	Error Rate (%)		
		Subset 2	Subset 3	Subset 4
Eigenface	4 10	53.3 11.11	75.4 33.9	52.9 20.0
Eigenface w/o 1st 3	4 10	31.11 6.7	60.0 20.0	29.4 12.9
Correlation	129	0.0	21.54	7.1
Linear Subspace	15	0.0	1.5	0.0
Fisherface	4	0.0	0.0	1.2

Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection (Belhumeur, Hespanha, Kriegman, PAMI 19(7), 1997)

Face detection and recognition



Face detection

- Basic idea: slide a window across image and evaluate a face model at every location

Challenges of face detection

- Sliding window detector must evaluate tens of thousands of location/scale combinations
  - This evaluation must be made as efficient as possible
- Faces are rare: 0–10 per image
  - At least 1000 times as many non-face windows as face windows
  - This means that the false positive rate must be extremely low
  - Also, we should try to spend as little time as possible on the non-face windows

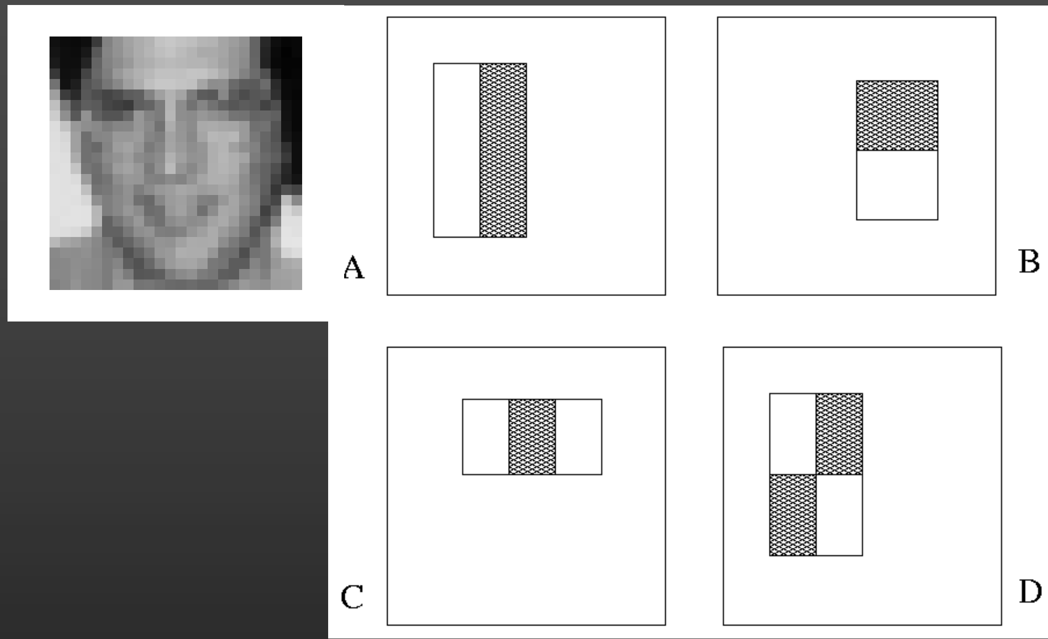
The Viola/Jones Face Detector

- A “paradigmatic” method for real-time object detection
- Training is slow, but detection is very fast
- Key ideas
  - *Integral images* for fast feature evaluation
  - *Boosting* for feature selection
  - *Attentional cascade* for fast rejection of non-face windows

P. Viola and M. Jones. *Rapid object detection using a boosted cascade of simple features*. CVPR 2001.

Image Features

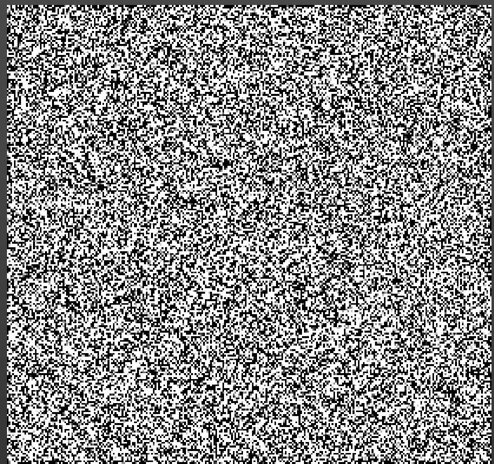
"Rectangle filters"



Value =

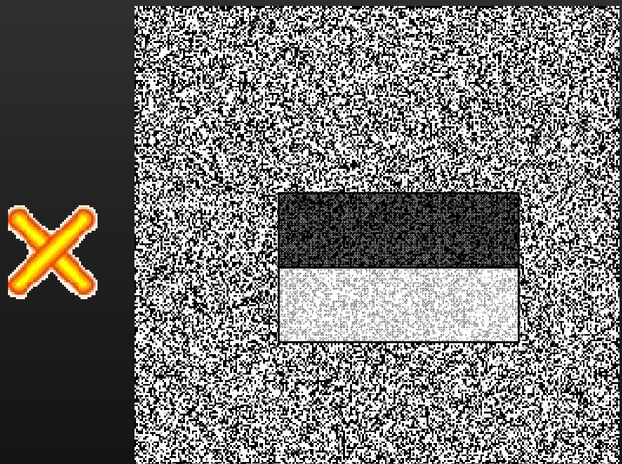
∑ (pixels in white area) –
∑ (pixels in black area)

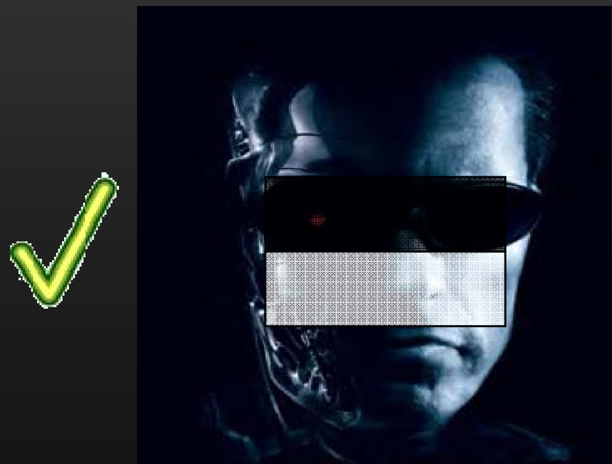
Example



Source

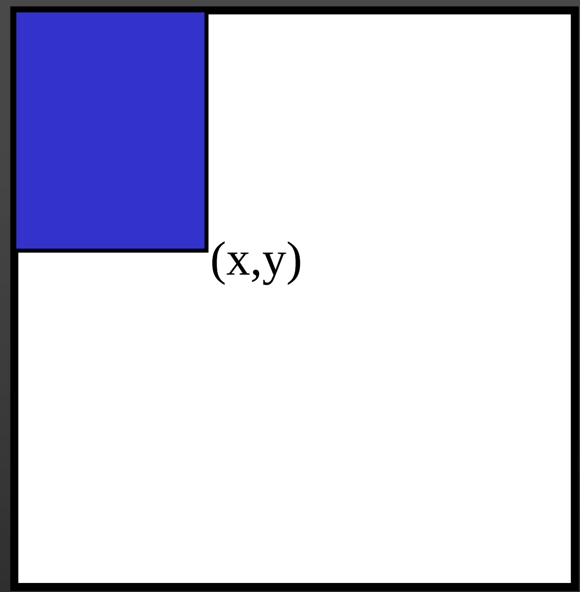
Result



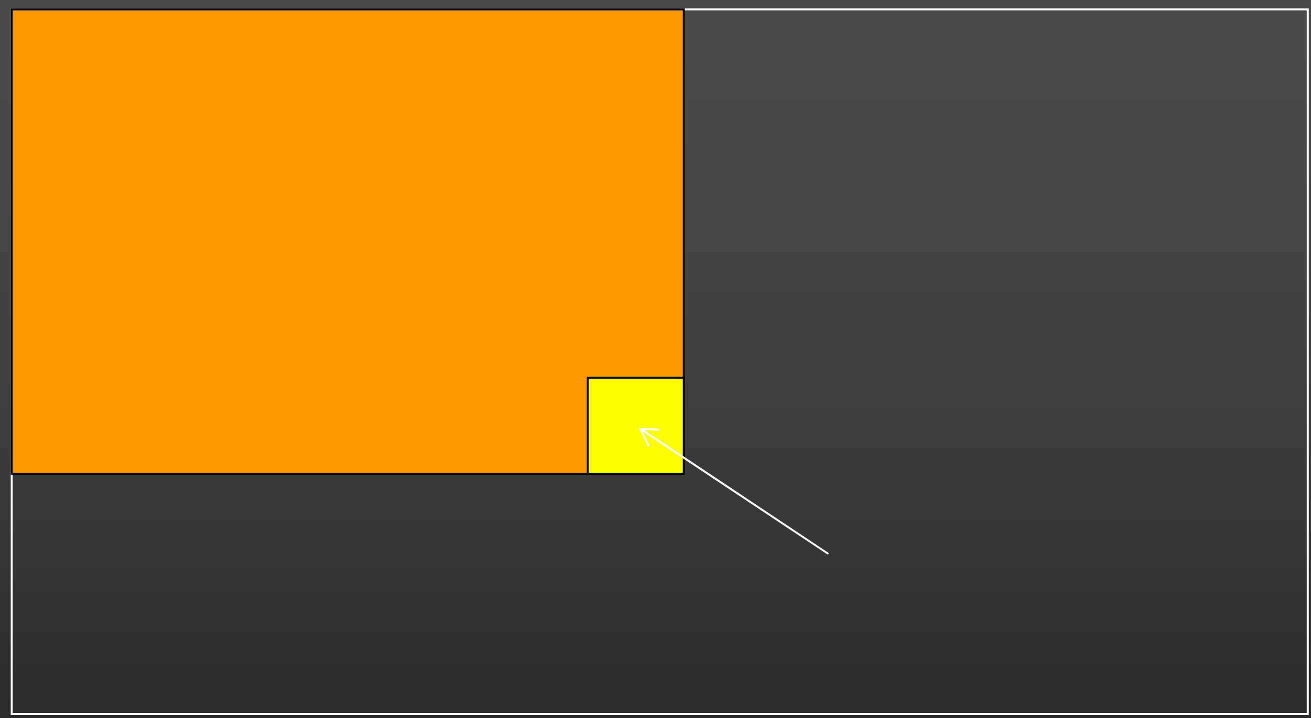


Fast computation with integral images

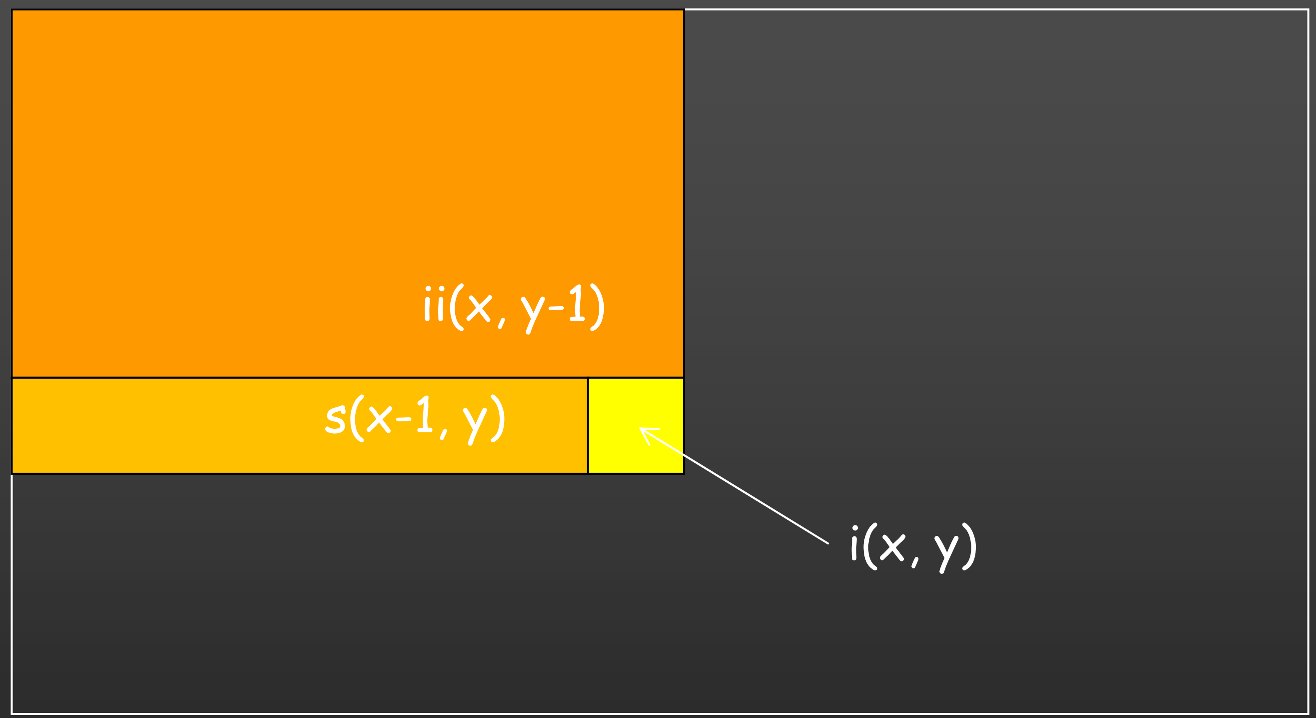
- The *integral image* computes a value at each pixel (x,y) that is the sum of the pixel values above and to the left of (x,y), inclusive
- This can quickly be computed in one pass through the image



Computing the integral image



Computing the integral image



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

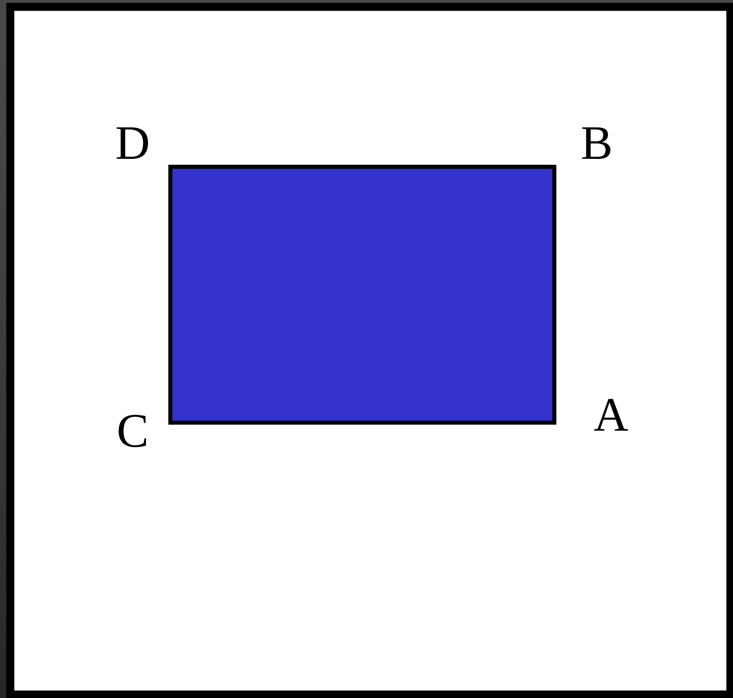
Integral image: $ii(x, y) = ii(x, y-1) + s(x, y)$

MATLAB: `ii = cumsum(cumsum(double(i)), 2);`

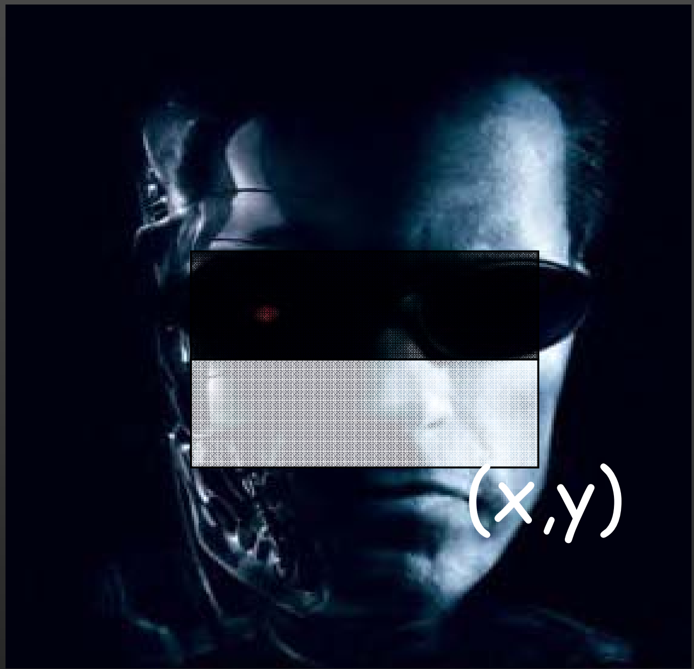
Computing sum within a rectangle

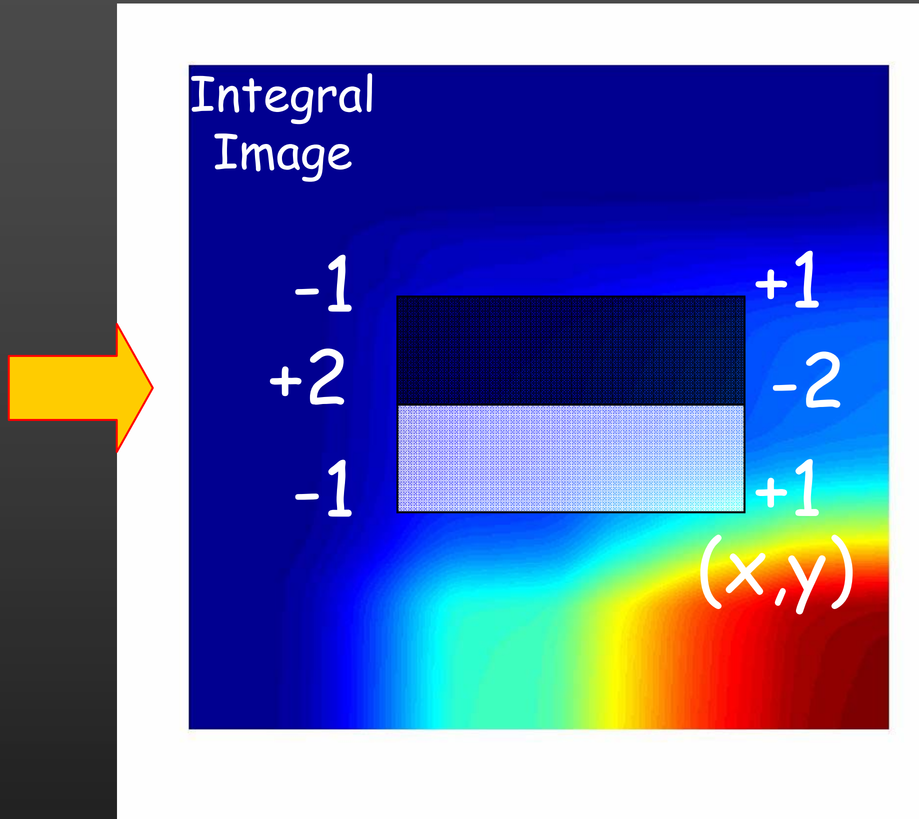
- Let A,B,C,D be the values of the integral image at the corners of a rectangle
- Then the sum of original image values within the rectangle can be computed as:

  sum = A – B – C + D
- Only 3 additions are required for any size of rectangle!
  - This is now used in many areas of computer vision



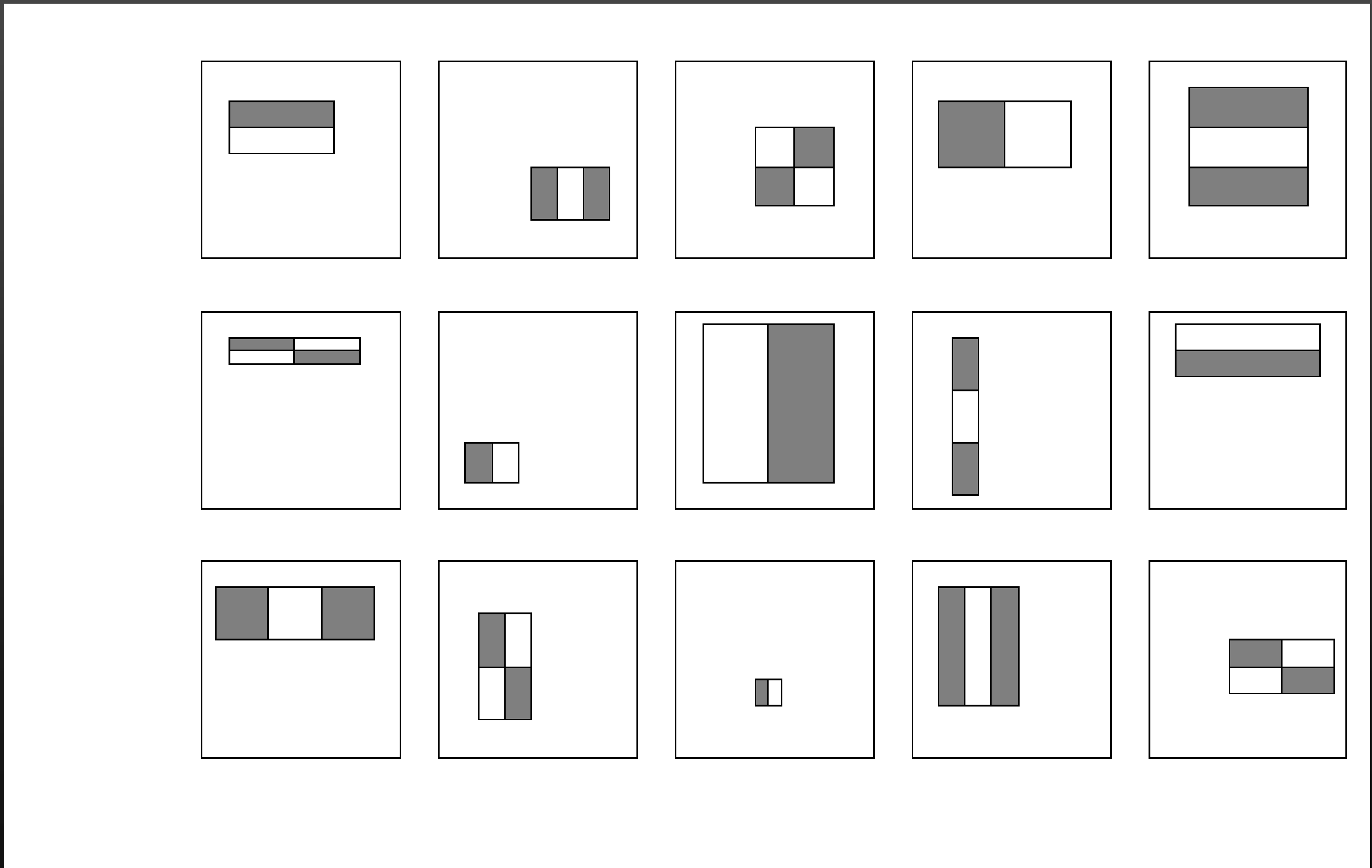
Example





Feature selection

- For a 24x24 detection region, the number of possible rectangle features is ~180,000!



Feature selection

- For a 24x24 detection region, the number of possible rectangle features is ~180,000!
- At test time, it is impractical to evaluate the entire feature set
- Can we create a good classifier using just a small subset of all possible features?
- How to select such a subset?

Boosting

- Boosting is a classification scheme that works by combining *weak learners* into a more accurate ensemble classifier
- *Weak learner*: classifier with accuracy that need be only better than chance
- We can define weak learners based on rectangle features:

value of rectangle feature

$$h_t(x) = \begin{cases} 1 & \text{if } p_t f_t(x) > p_t \theta_t \\ 0 & \text{otherwise} \end{cases}$$

window

parity threshold

Y. Freund and R. Schapire, A short introduction to boosting, *Journal of Japanese Society for Artificial Intelligence*, 14(5):771-780, September, 1999

Boosting

- Boosting is a classification scheme that works by combining *weak classifiers* into a more accurate ensemble classifier
- *Weak classifier*: classifier with accuracy that need be only better than chance
- We can define weak classifiers based on rectangle features:

value of rectangle feature

Parameters tuned to minimize misclassification error

$$h_t(x) = \begin{cases} 1 & \text{if } p_t f_t(x) > p_t \theta_t \\ 0 & \text{otherwise} \end{cases}$$

window

parity threshold

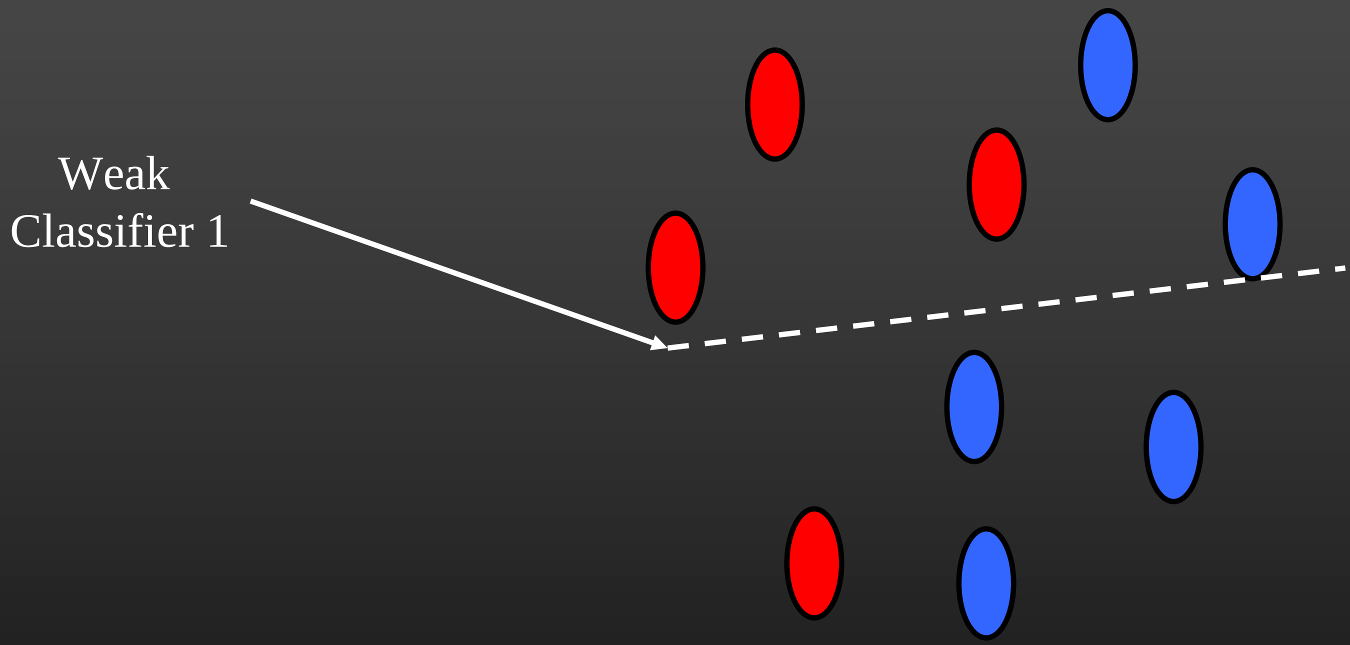
Y. Freund and R. Schapire, A short introduction to boosting, *Journal of Japanese Society for Artificial Intelligence*, 14(5):771-780, September, 1999

Boosting outline

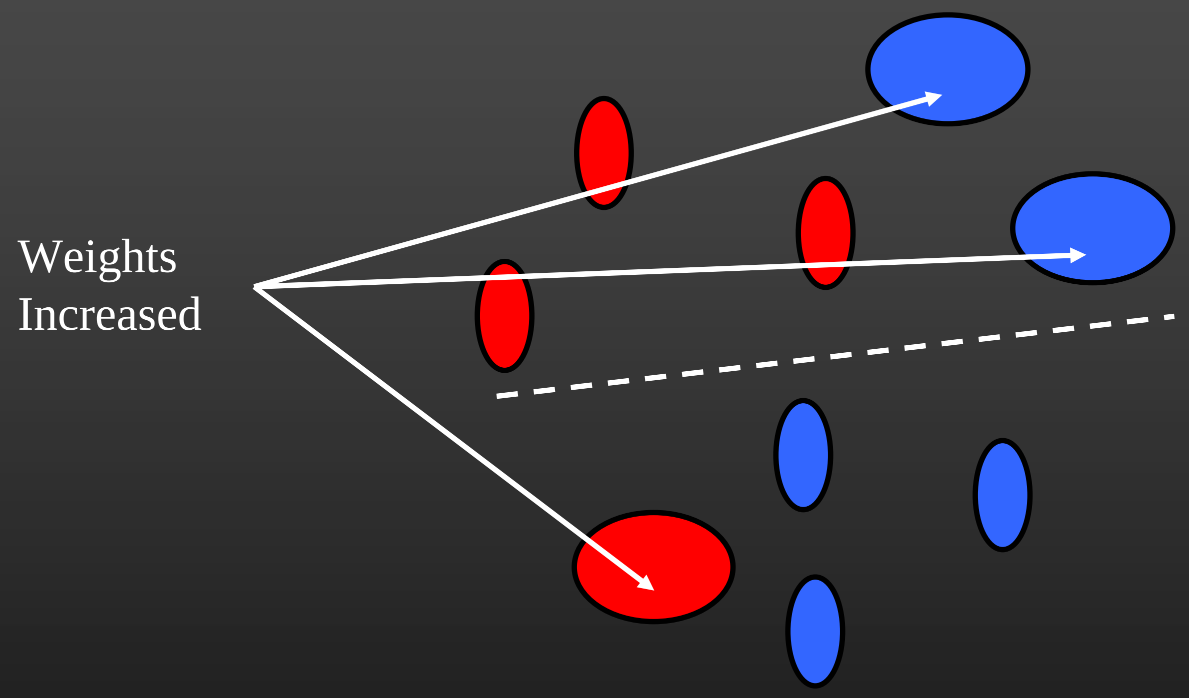
- Initially, give equal weight to each training example
- Iterative training procedure
  - Find best weak classifier for current weighted training set
  - Raise the weights of training examples misclassified by current weak learner
- Compute final classifier as linear combination of all weak classifiers (weight of each learner is related to its accuracy)

Y. Freund and R. Schapire, A short introduction to boosting, *Journal of Japanese Society for Artificial Intelligence*, 14(5):771-780, September, 1999

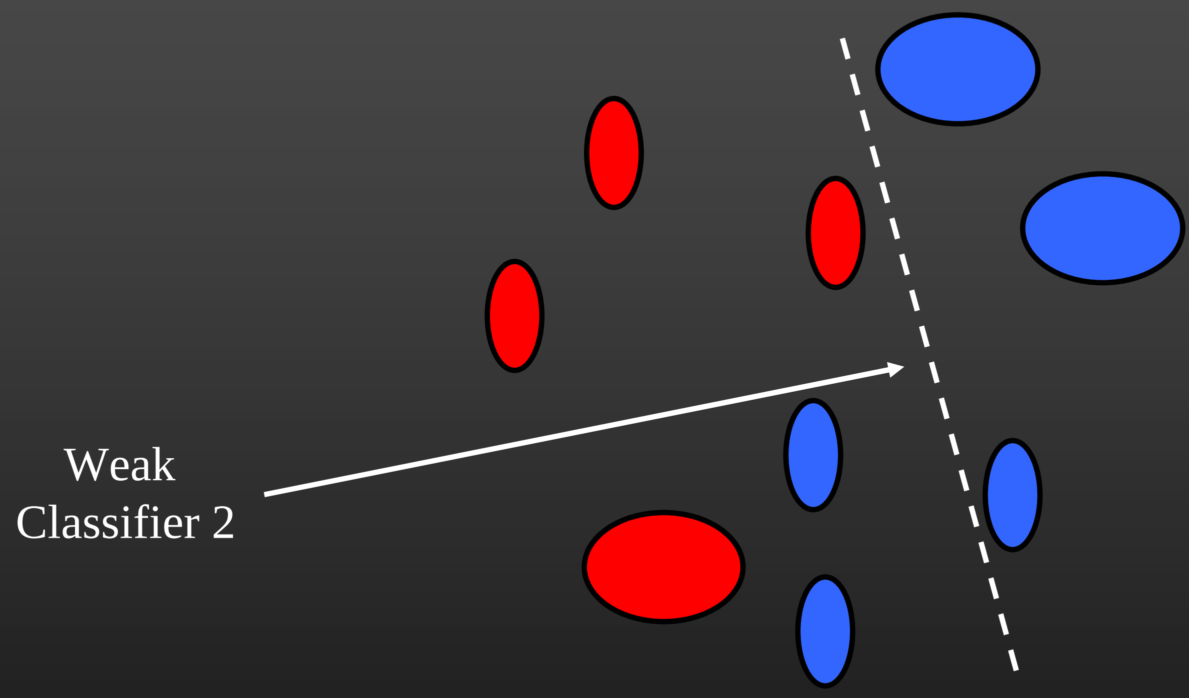
Boosting



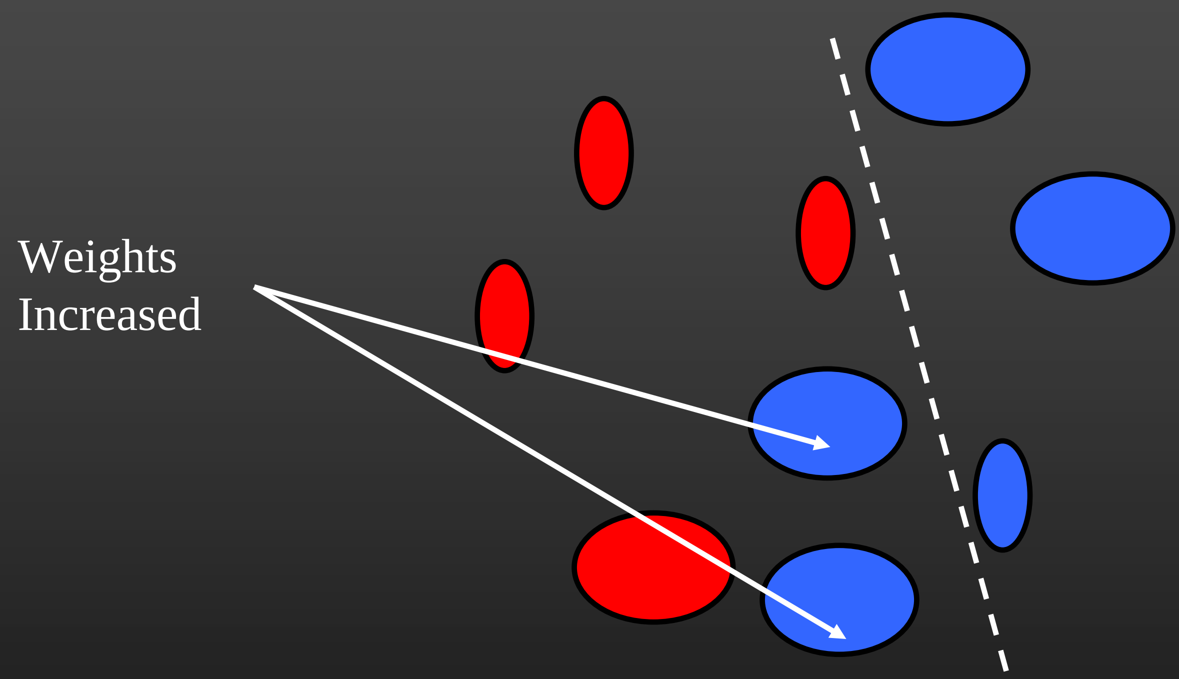
Boosting



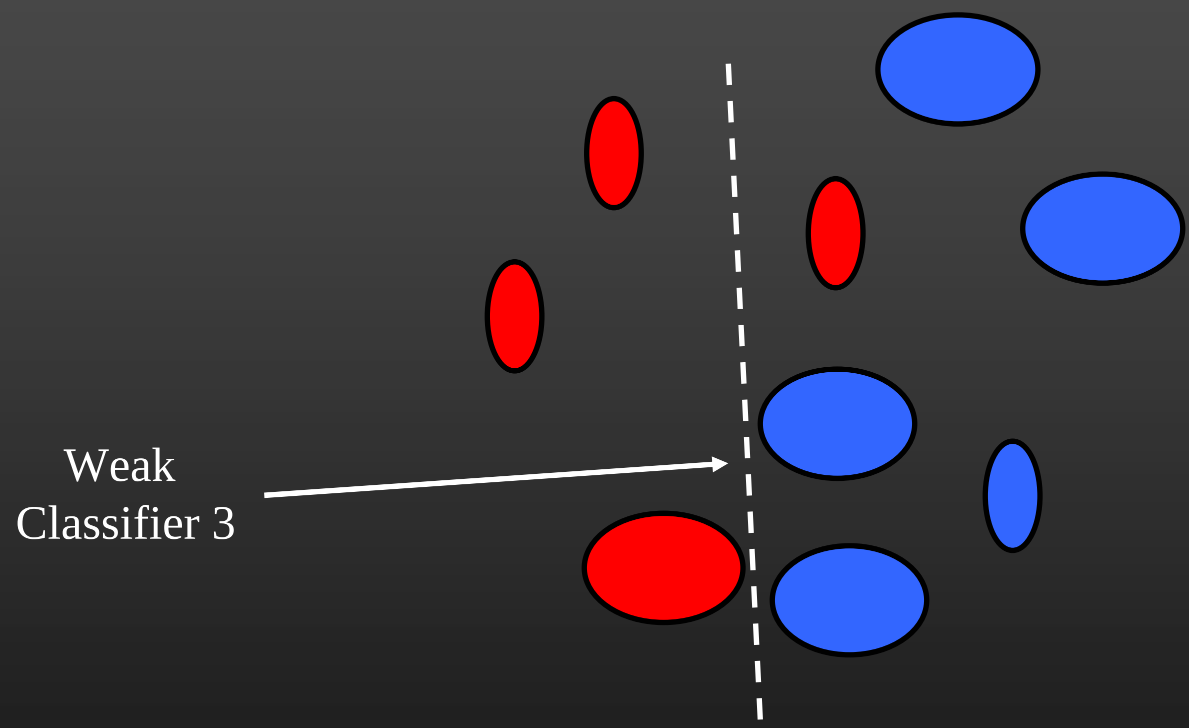
Boosting



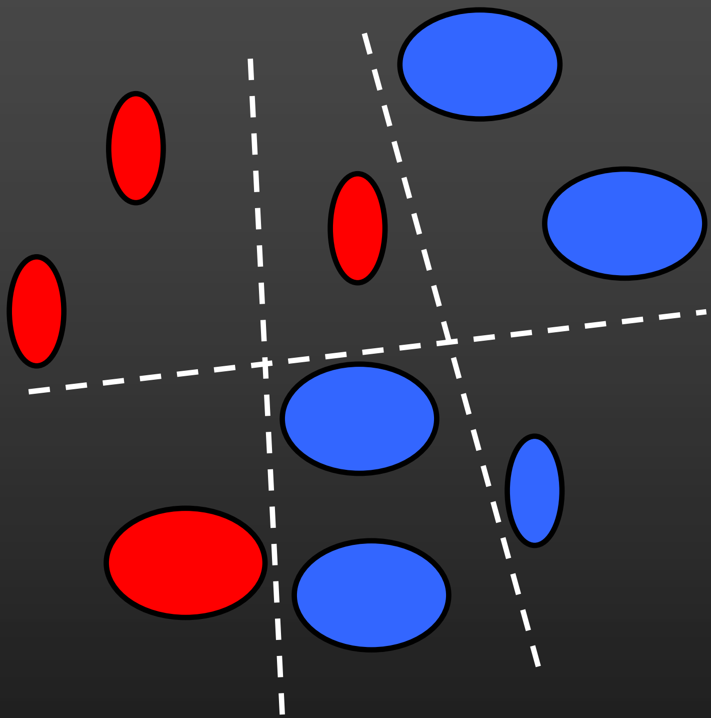
Boosting



Boosting



Boosting



Final classifier is
linear combination of weak
classifiers

AdaBoost algorithm: more details

Start with equal weights on each data point (i)

For t = 1 …,T

- Select weak classifier with minimum error

$$\epsilon_t = \sum_i \omega_i [h_t(x_i) \neq y_i] \quad \text{where } \omega_i \text{ are weights}$$

- Set

$$\alpha_t = \frac{1}{2} \ln \frac{1 - \epsilon_t}{\epsilon_t}$$

- Reweight examples (boosting) to give misclassified examples more weight

$$\omega_{t+1,i} = \omega_{t,i} e^{-\alpha_t y_i h_t(x_i)}$$

- Add weak classifier with weight α_t

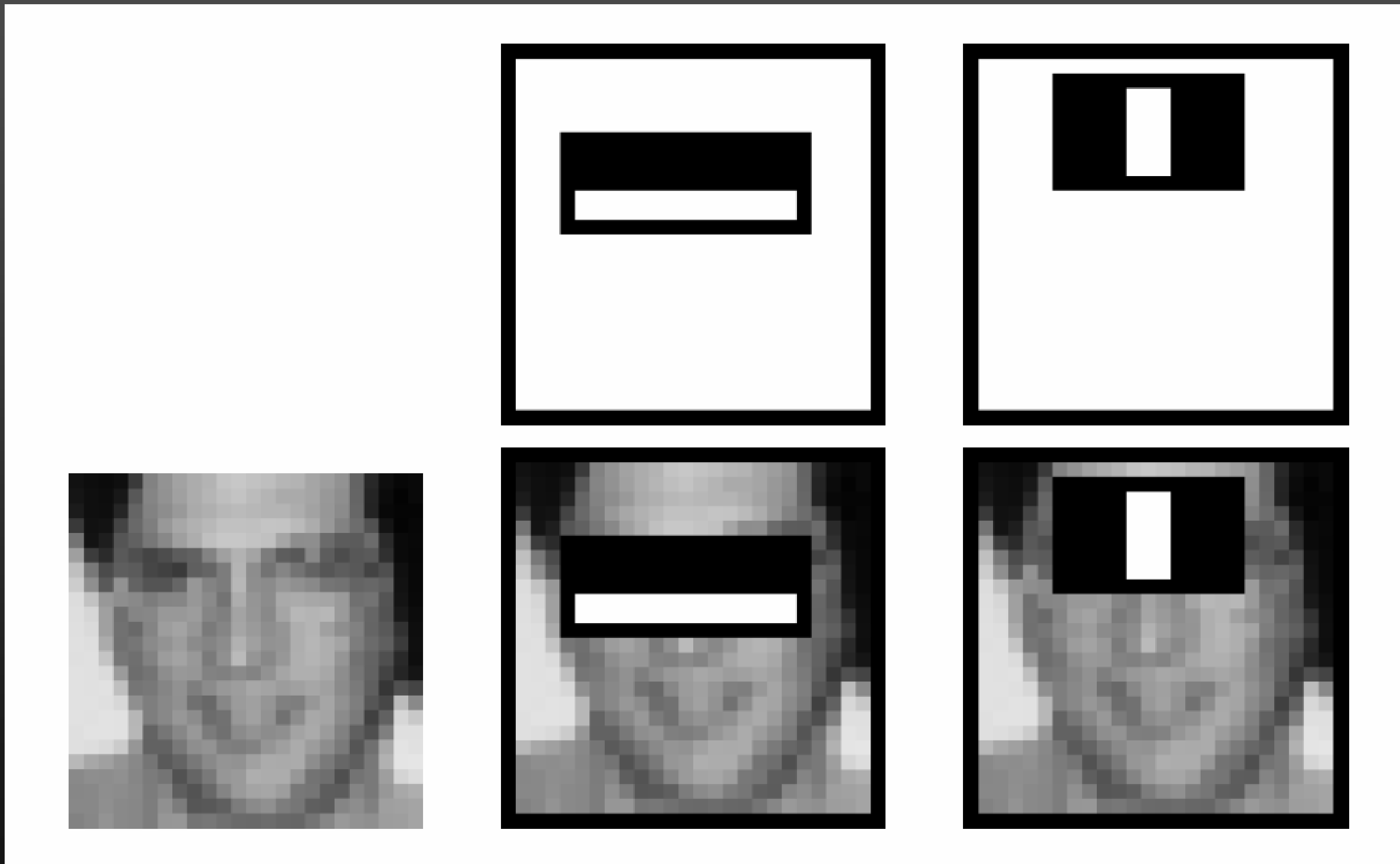
Final classifier:

$$H(x) = \text{sign} \sum_{t=1}^{T} \alpha_t h_t(x)$$

Boosting for face detection

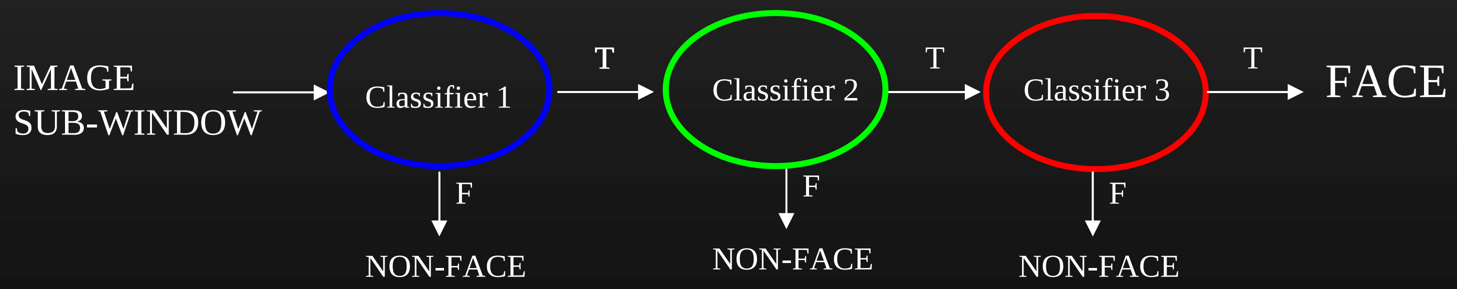
- For each round of boosting:
  - Evaluate each rectangle filter on each example
  - Select best threshold for each filter
  - Select best filter/threshold combination
  - Reweight examples
- Computational complexity of learning: $O(MNT)$
  - M filters, N examples, T thresholds

First two features selected by boosting



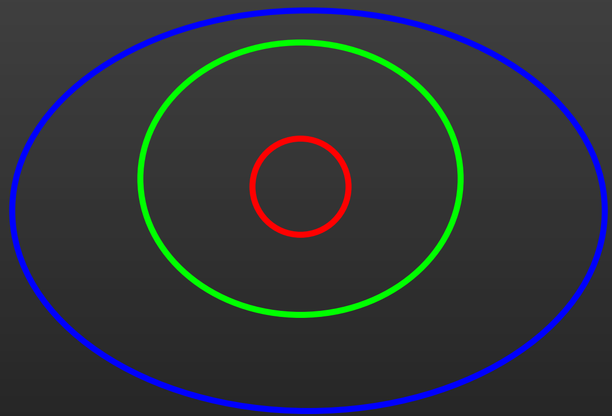
Cascading classifiers

- We start with simple classifiers which reject many of the negative sub-windows while detecting almost all positive sub-windows
- Positive results from the first classifier triggers the evaluation of a second (more complex) classifier, and so on
- A negative outcome at any point leads to the immediate rejection of the sub-window

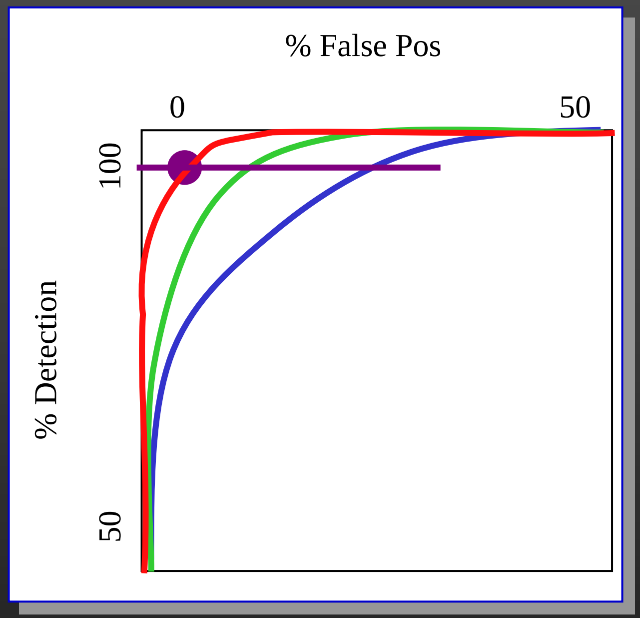


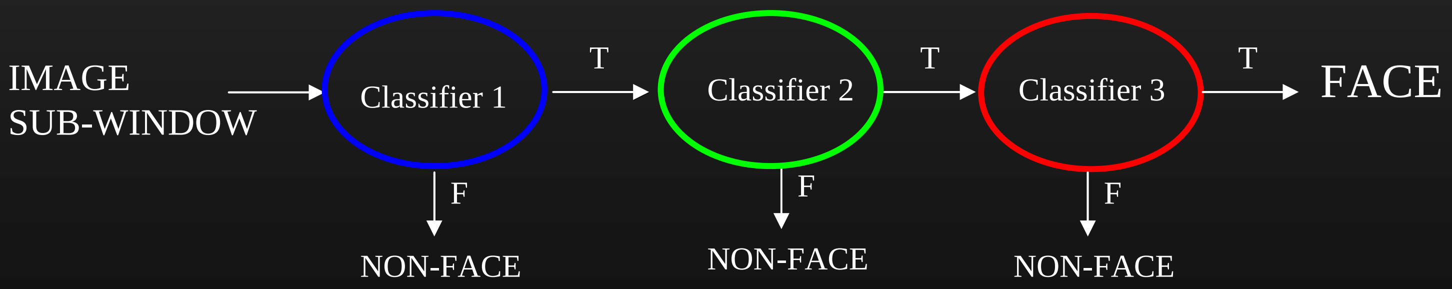
Cascading classifiers

- Chain classifiers that are progressively more complex and have lower false positive rates:



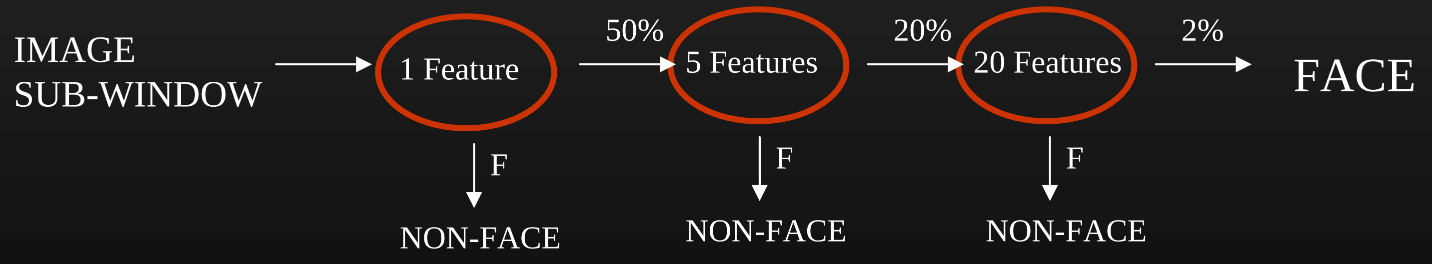
Receiver operating characteristic





Training the cascade

- Adjust weak learner threshold to minimize *false negatives* (as opposed to total classification error)
- Each classifier trained on false positives of previous stages
  - A single-feature classifier achieves 100% detection rate and about 50% false positive rate
  - A five-feature classifier achieves 100% detection rate and 40% false positive rate (20% cumulative)
  - A 20-feature classifier achieve 100% detection rate with 10% false positive rate (2% cumulative)



The implemented system

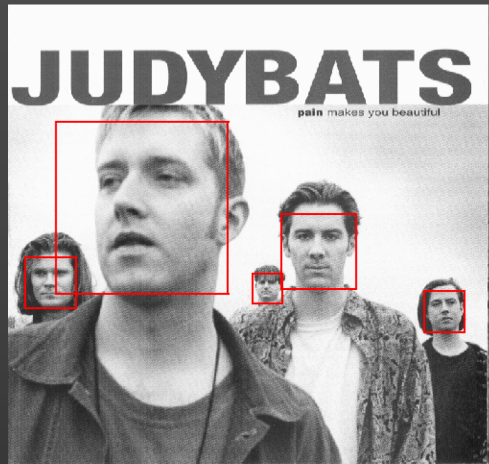
- Training Data
  - 5000 faces
    - All frontal, rescaled to 24x24 pixels
  - 300 million non-faces
    - 9500 non-face images
  - Faces are normalized
    - Scale, translation
- Many variations
  - Across individuals
  - Illumination
  - Pose

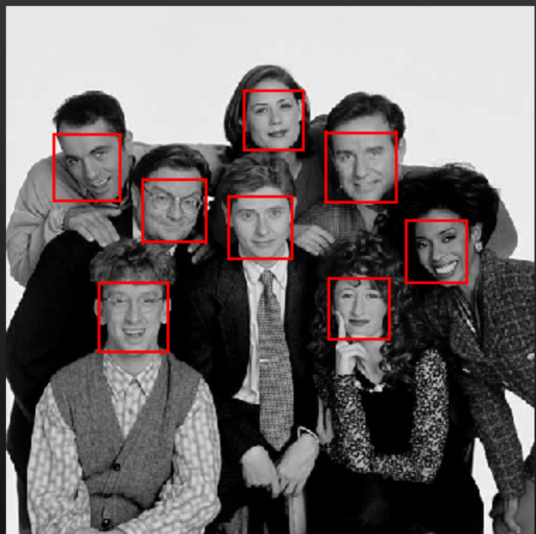
(Most slides from Paul Viola)

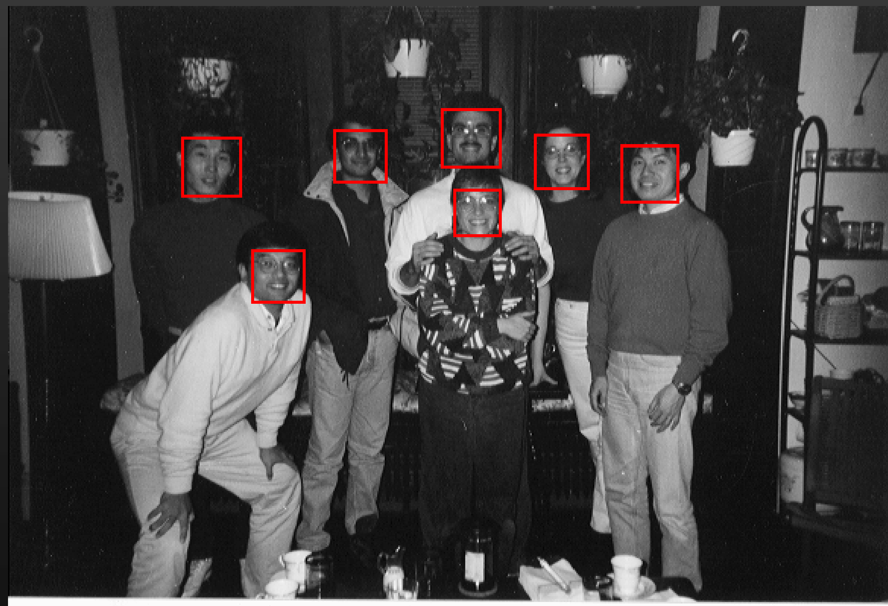
System performance

- Training time: “weeks” on 466 MHz Sun workstation
- 38 layers, total of 6061 features
- Average of 10 features evaluated per window on test set
- “On a 700 Mhz Pentium III processor, the face detector can process a 384 by 288 pixel image in about .067 seconds”
  - 15 Hz
  - 15 times faster than previous detector of comparable accuracy (Rowley et al., 1998)

Output of Face Detector on Test Images

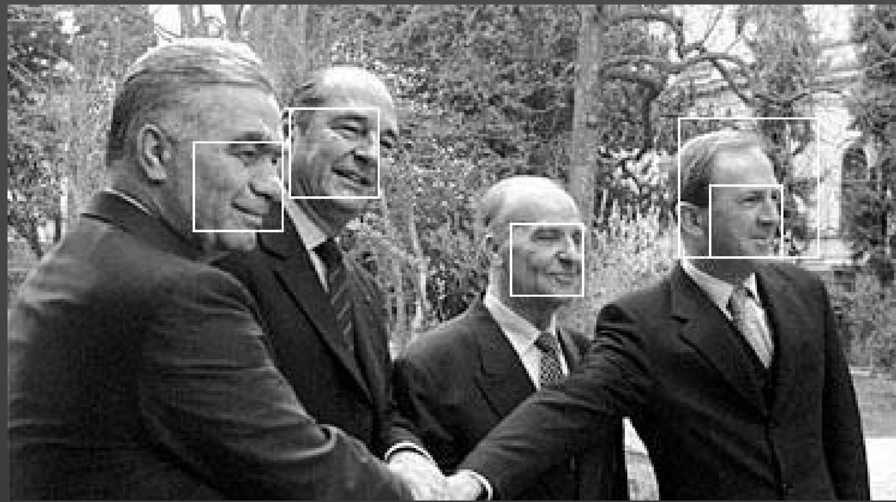






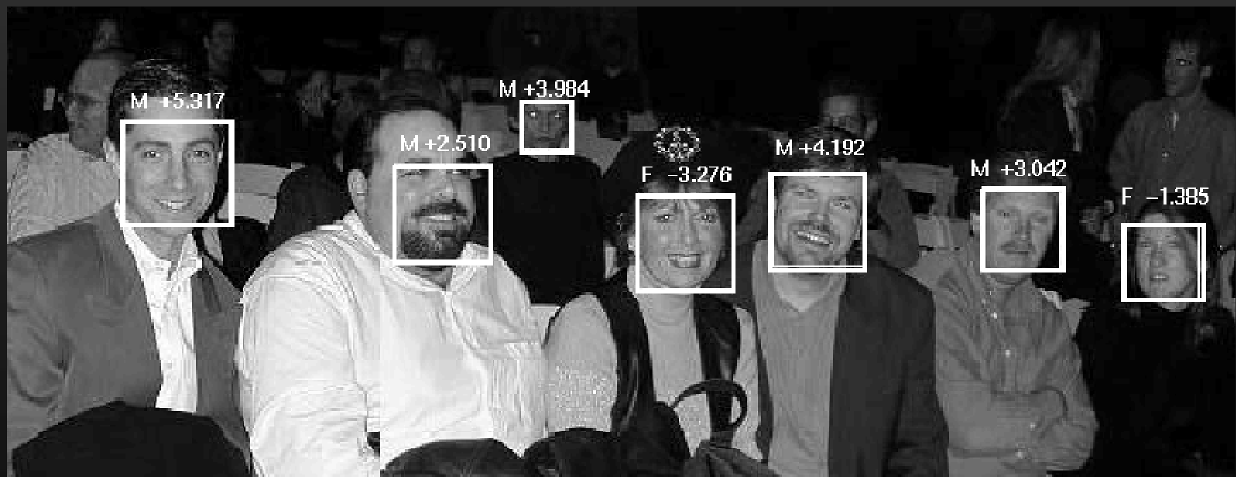
Other detection tasks

Facial Feature Localization



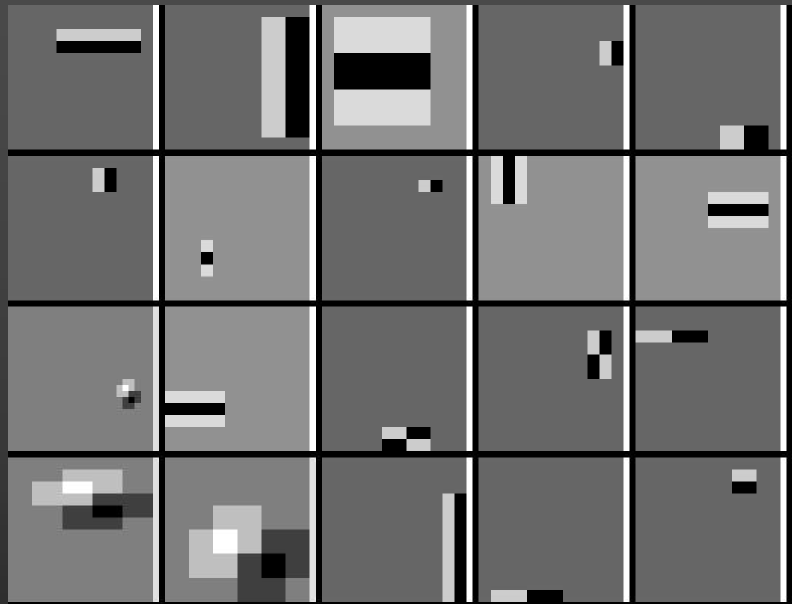
Profile Detection

Male vs.
female



Profile Detection

Profile Features

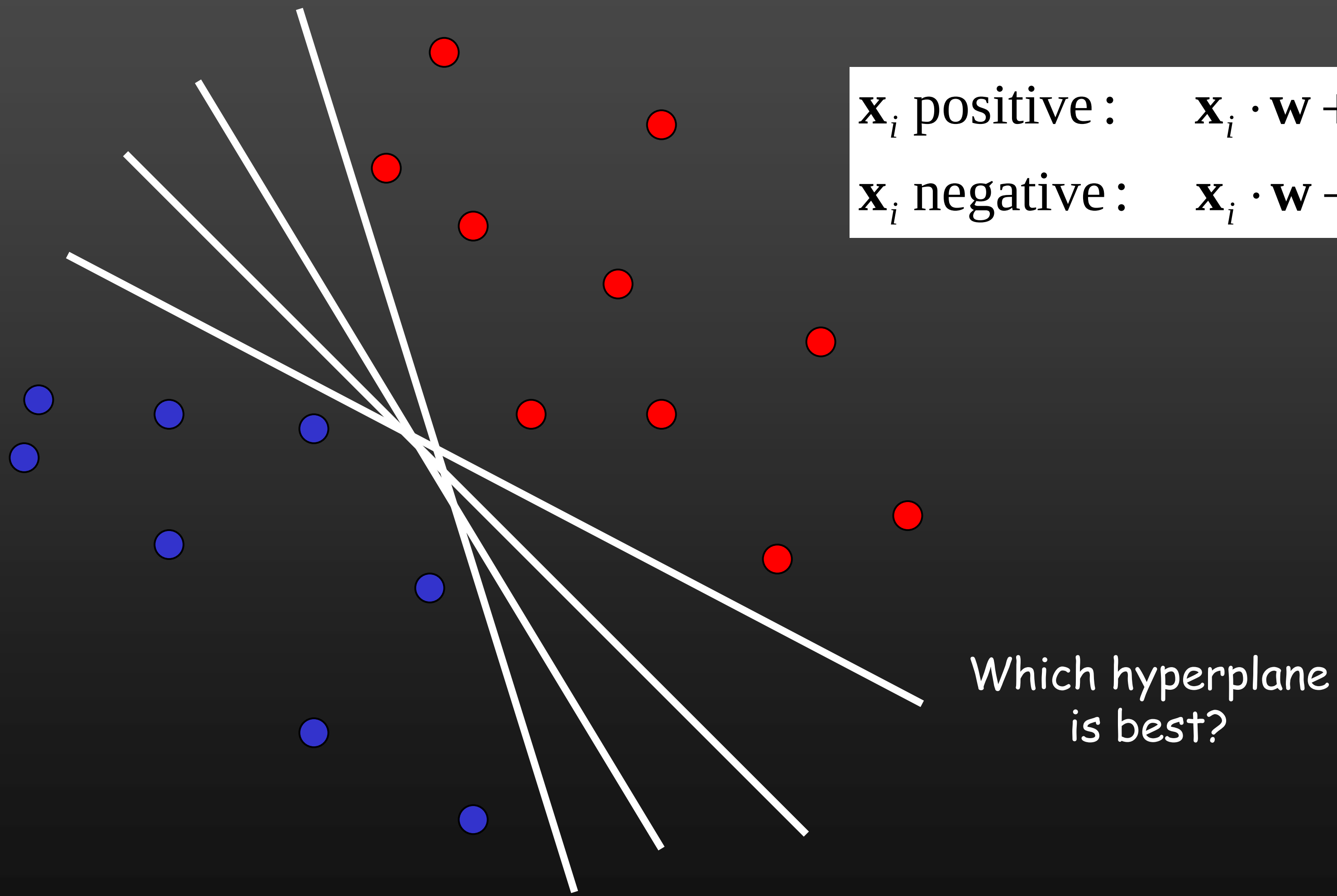


Summary: Viola/Jones detector

- Rectangle features
- Integral images for fast computation
- Boosting for feature selection
- Attentional cascade for fast rejection of negative windows

Linear classifiers

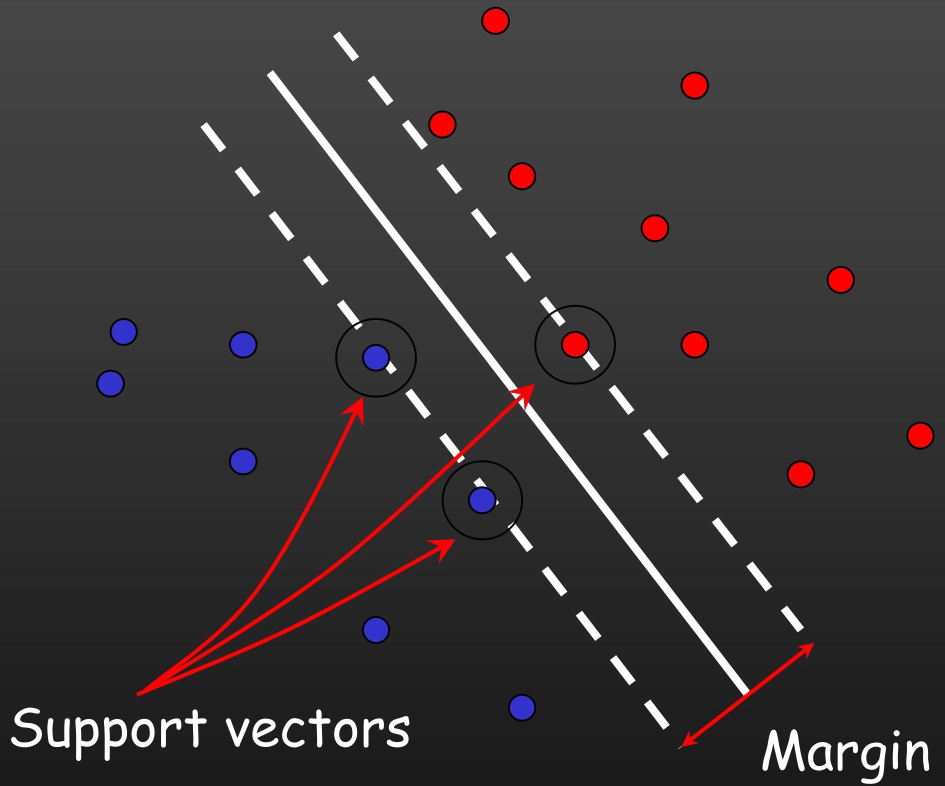
- Find linear function (*hyperplane*) to separate positive and negative examples



$$\mathbf{x}_i \text{ positive}: \quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 0$$
$$\mathbf{x}_i \text{ negative}: \quad \mathbf{x}_i \cdot \mathbf{w} + b < 0$$

Support vector machines

- Find hyperplane that maximizes the *margin* between the positive and negative examples



$$\mathbf{x}_i \text{ positive } (y_i = 1): \quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$$

$$\mathbf{x}_i \text{ negative } (y_i = -1): \quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$$

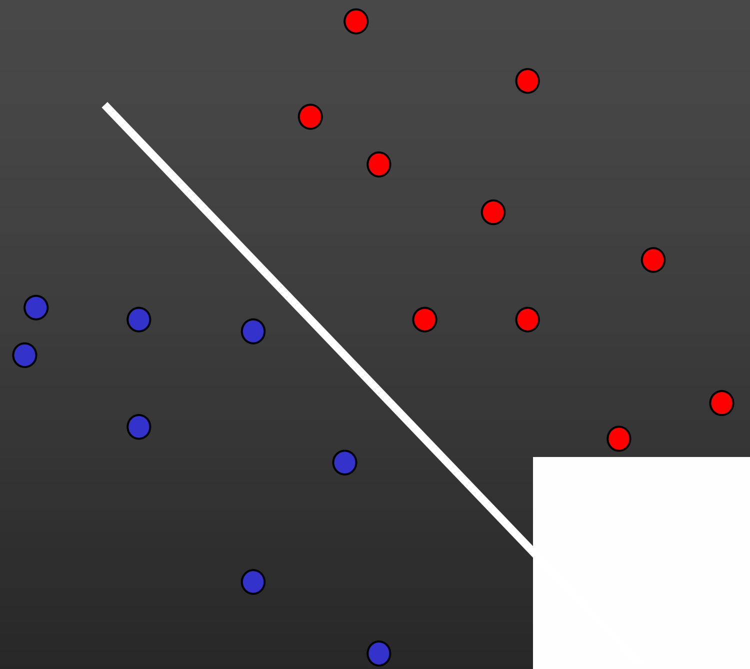
For support, vectors, $\mathbf{x}_i \cdot \mathbf{w} + b = \pm 1$

Distance between point and hyperplane: $\dfrac{|\mathbf{x}_i \cdot \mathbf{w} + b|}{\|\mathbf{w}\|}$

Therefore, the margin is $2 / \|\mathbf{w}\|$

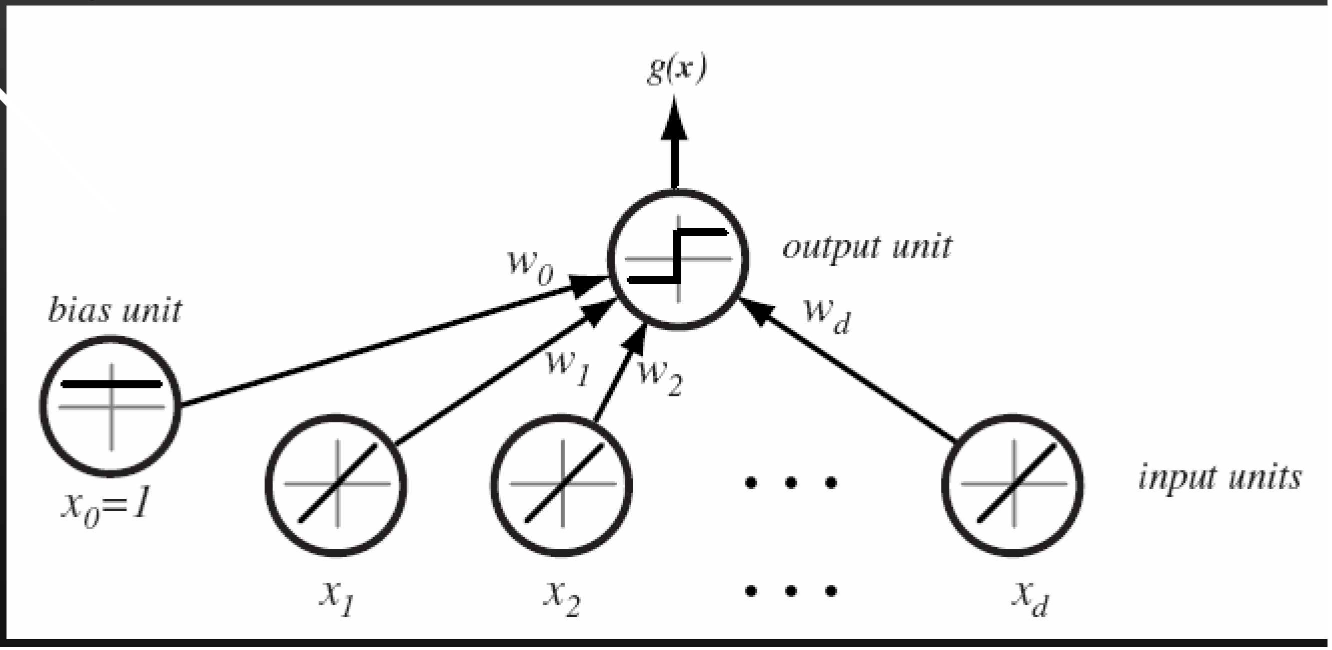
C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, 1998

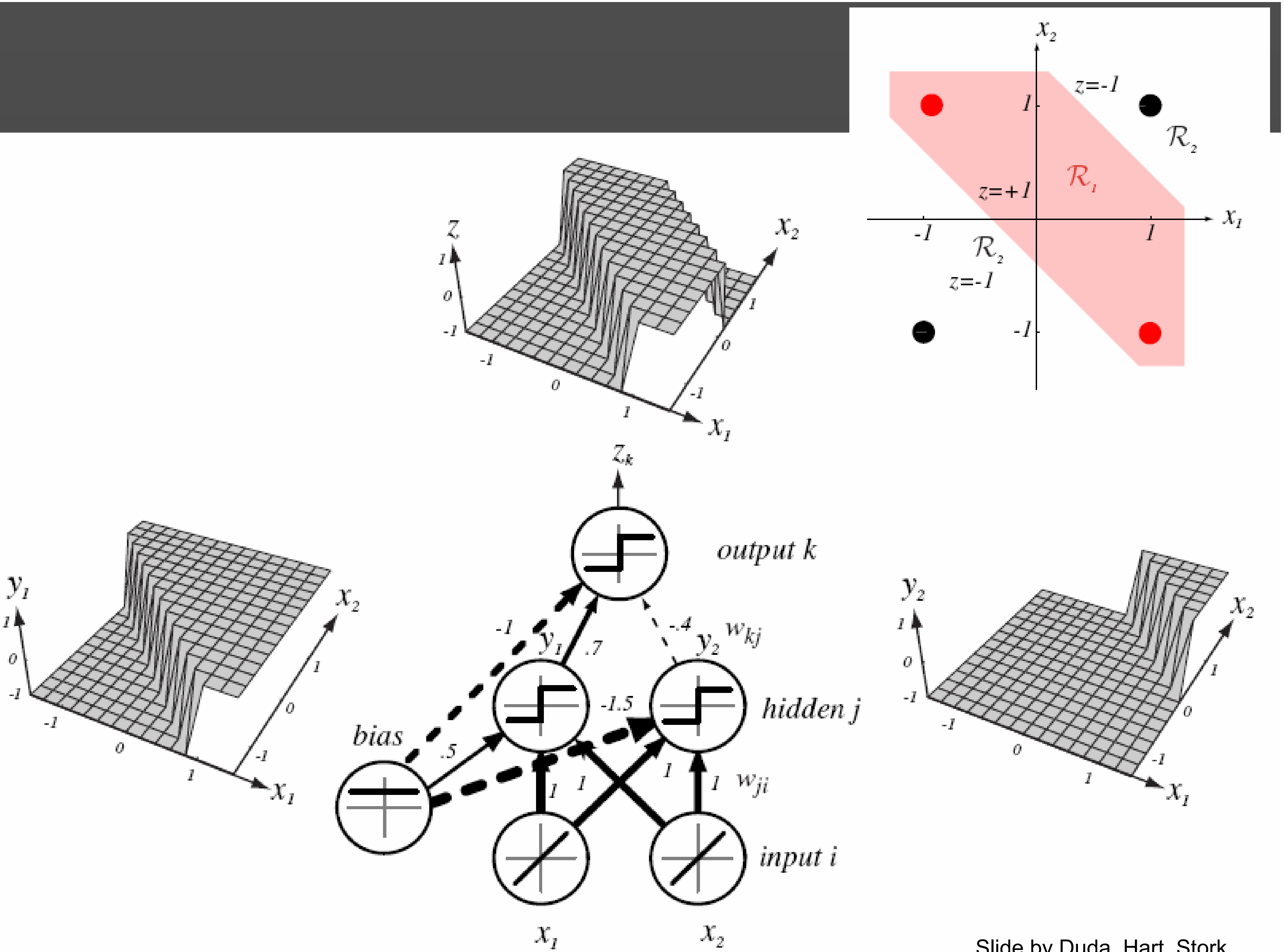
Linear classifiers



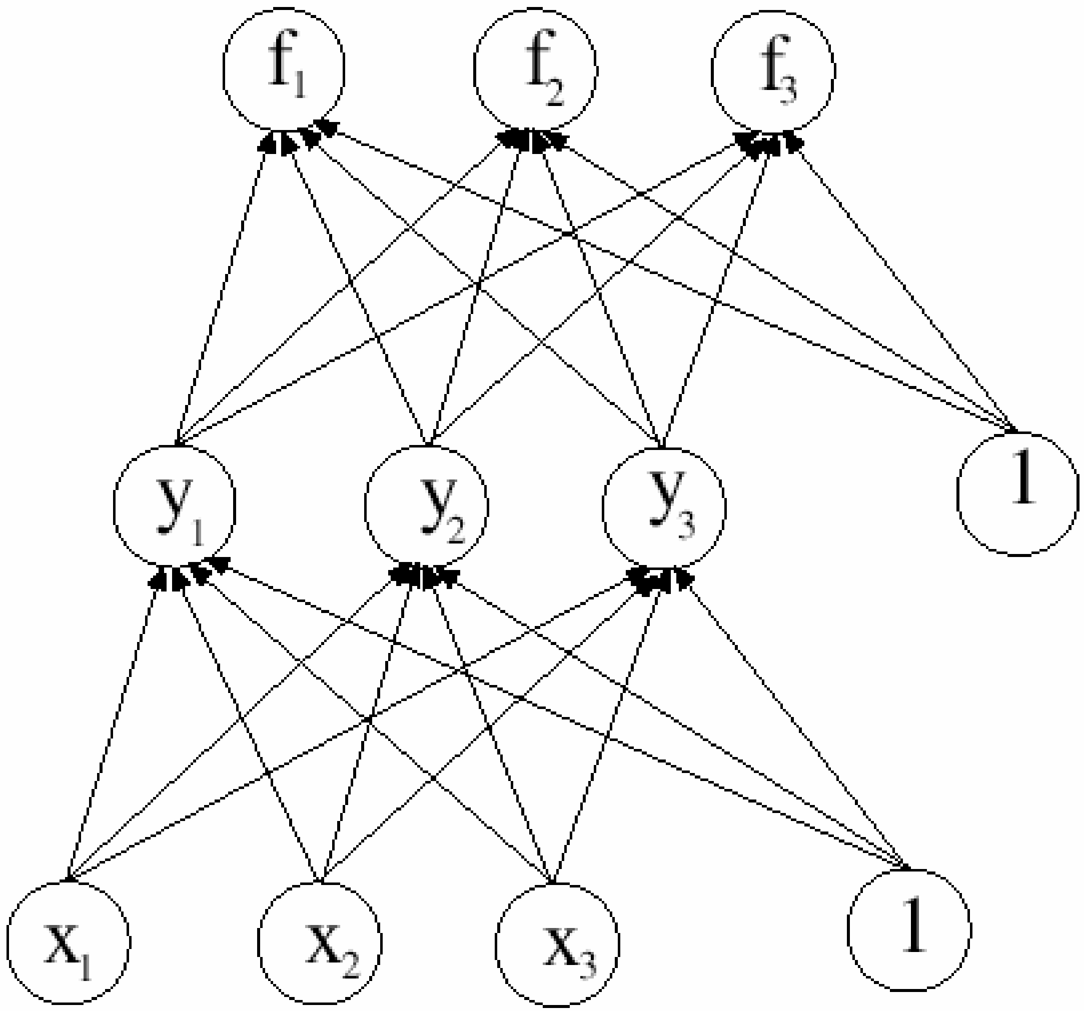
The perceptron
(Rosenblatt'57)

What the perceptron can learn, it will learn using a simple weight update rule.





Slide by Duda, Hart, Stork



Multi-layer neural network

$$\boldsymbol{g}(\boldsymbol{x}) \approx \boldsymbol{f}(\boldsymbol{x}) = [\phi(\boldsymbol{w}_{21} \cdot \boldsymbol{y}), \phi(\boldsymbol{w}_{22} \cdot \boldsymbol{y}), \ldots \phi(\boldsymbol{w}_{2n} \cdot \boldsymbol{y})]$$

$$\boldsymbol{y}(\boldsymbol{z}) = [\phi(\boldsymbol{w}_{11} \cdot \boldsymbol{z}), \phi(\boldsymbol{w}_{12} \cdot \boldsymbol{z}), \ldots \phi(\boldsymbol{w}_{1m} \cdot \boldsymbol{z}), 1]$$

$$\boldsymbol{z}(\boldsymbol{x}) = [x_1, x_2, \ldots, x_p, 1]$$

Slide by D.A. Forsyth

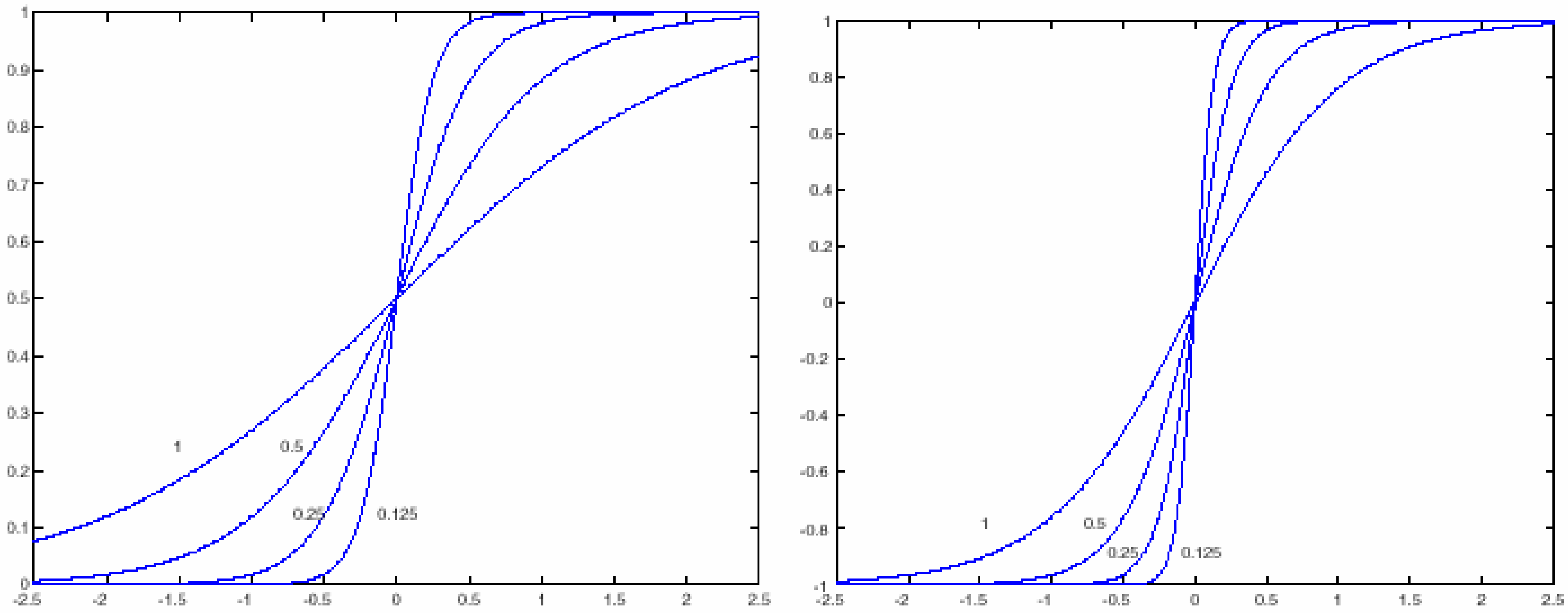
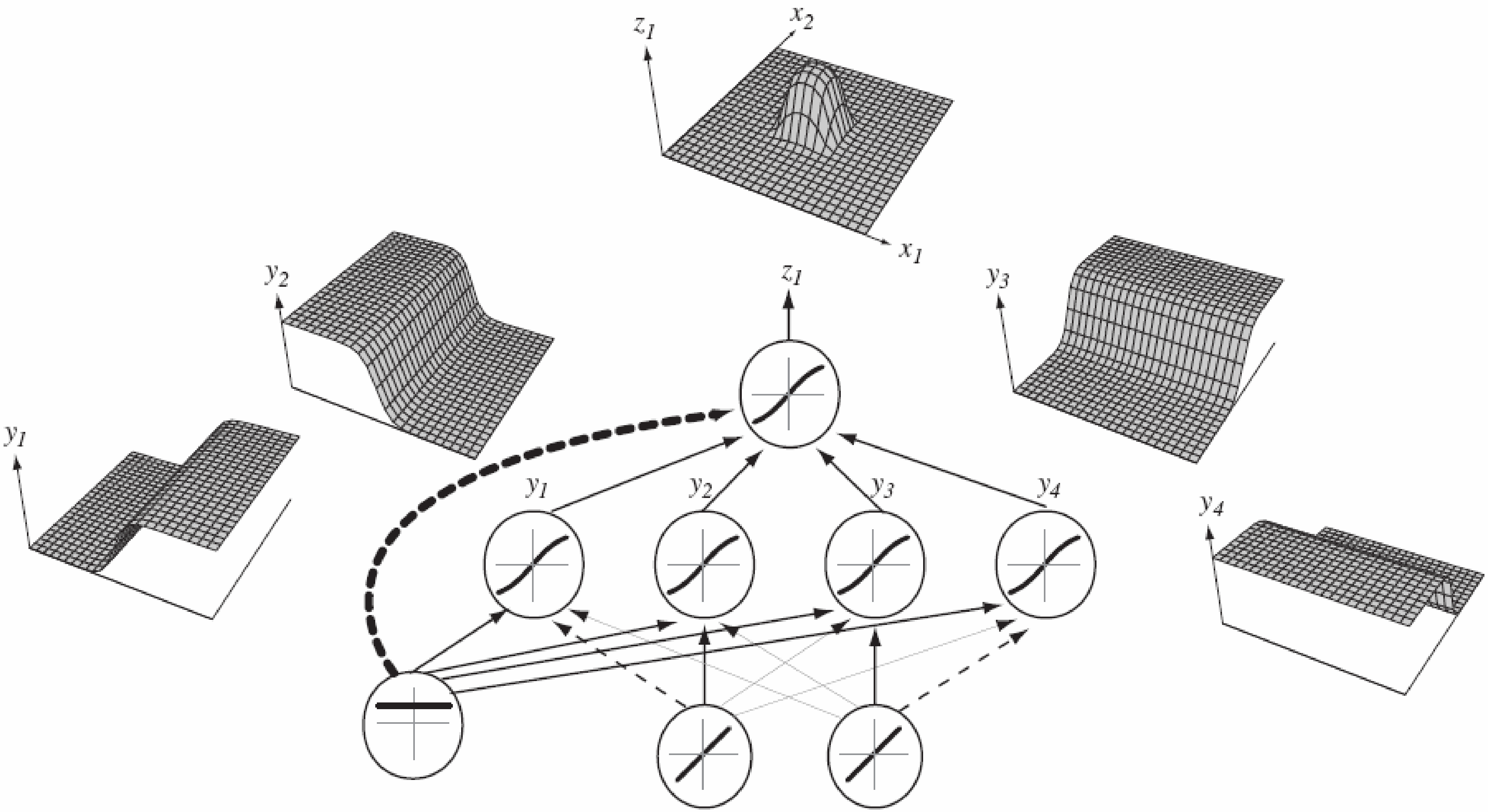


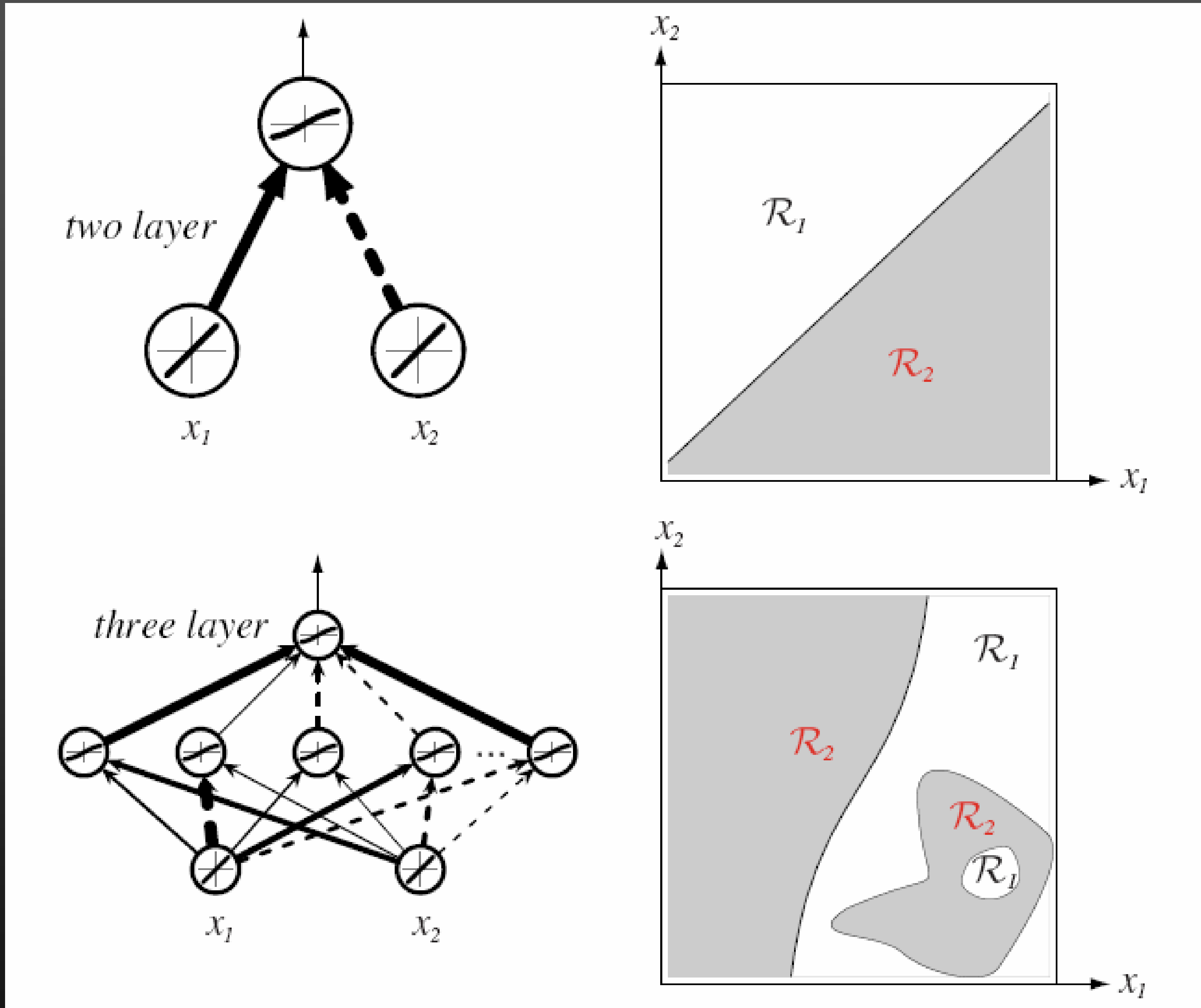
Figure 22.14. On the **left**, a series of squashing functions obtained using $\phi(x;\nu) = \frac{e^{x/\nu}}{1+e^{x/\nu}}$, for different values of ν indicated on the figure. On the **right**, a series of squashing functions obtained using $\phi(x;\nu,A) = A\tanh(x/\nu)$ for different values of ν indicated on the figure. Generally, for x close to the center of the range, the squashing function is linear; for x small or large, it is strongly non-linear.

Slides by D.A. Forsyth

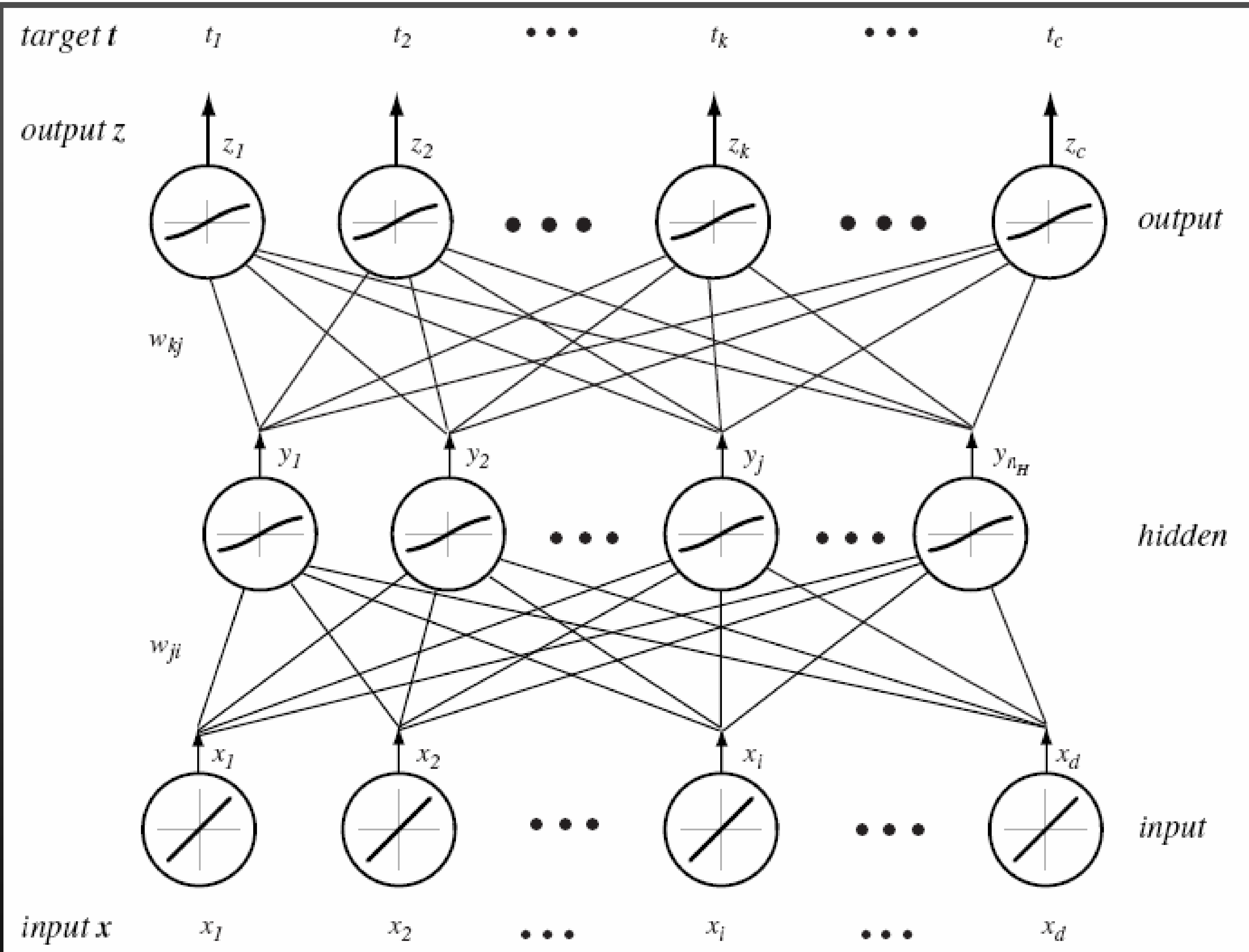


Any function can be learned by a 3-layer network with enough hidden units

Slide by Duda, Hart, Stork



Slide by Duda, Hart, Stork



Slide by Duda, Hart, Stork

Gradient-based supervised learning

- Parametric prediction function: $f(x, w) \rightarrow y$
- Learning: Minimize

$$E = \sum_i L(y_i, f(x_i, w))$$

- Recognition: $y = f(x, w)$

How can we minimize E? ..Gradient descent..

Gradient-based supervised learning II

- Gradient descent:
  - compute $\nabla E = (\partial E/\partial w_1, \ldots, \partial E/\partial w_n)$
  - $w_{k+1} \leftarrow w_k - \epsilon \nabla E$

- Stochastic gradient descent:
  - compute ∇E_i
  - $w_{k+1} \leftarrow w_k - \epsilon \nabla E_i$

  where E_i is the energy associated with some random training sample i

- The stochastic version works much better in practice.

Gradient-based supervised learning III

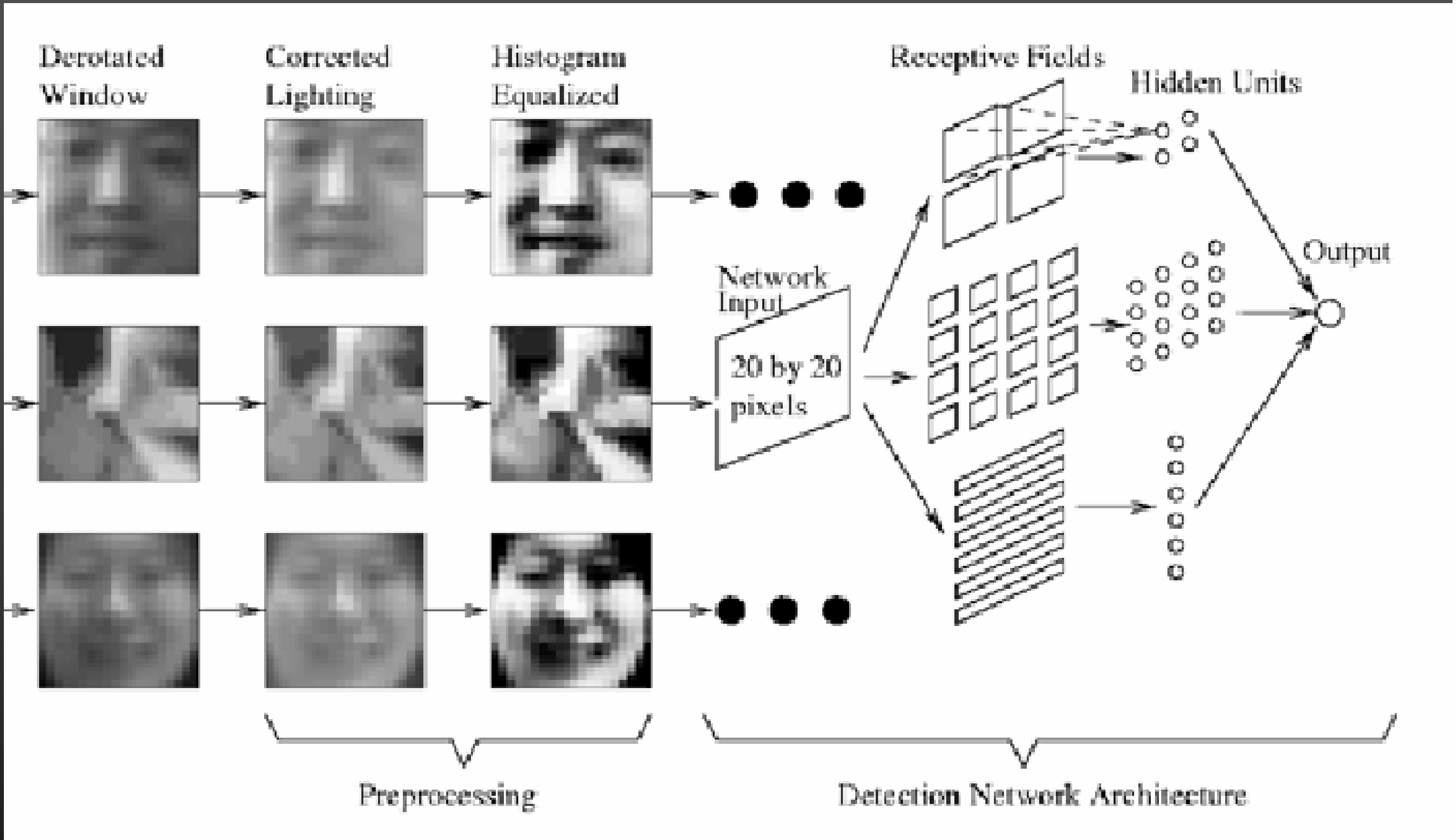
- Consider a feed-forward system composed of successive modules:
  $x_i = f_i\,(w_i, x_{i-1})$

  $E = L\,(y\ , x)$, with $x = x_n = f_n\,(w_n, x_{n-1})$

  $$\partial E/\partial w_n = \partial E/\partial x\ \partial f_n/\partial w_n$$

  $$\partial E/\partial x_{n-1} = \partial E/\partial x\ \partial f_n/\partial x_{n-1}$$

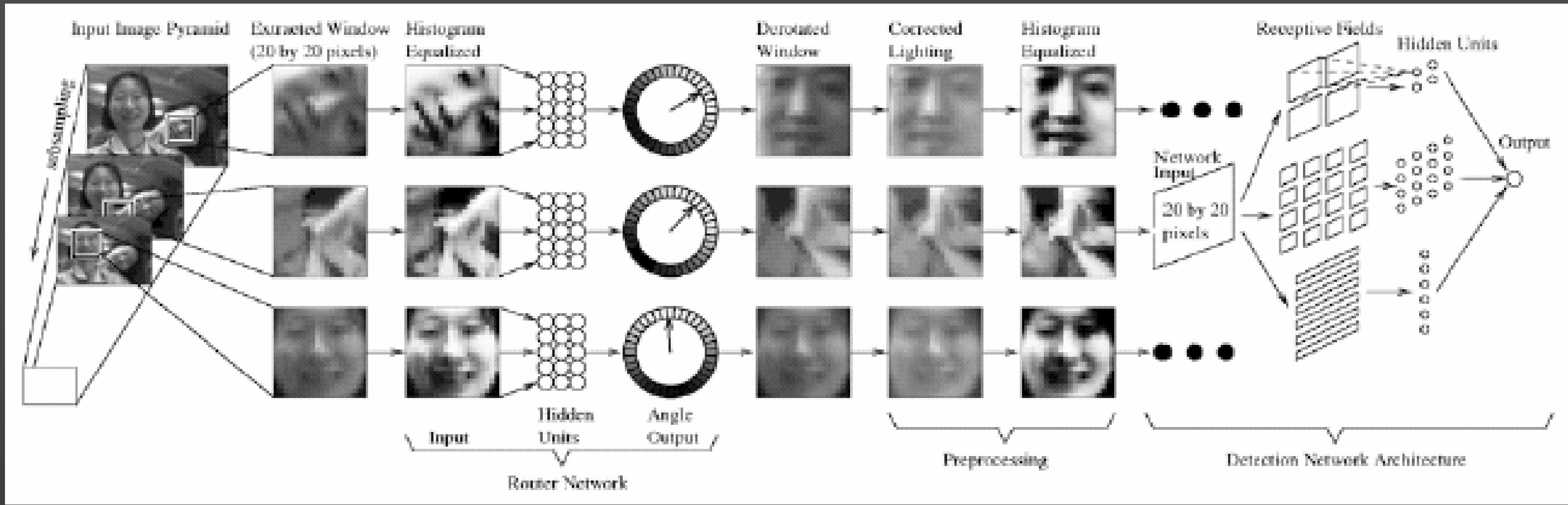
- Backward recursion: backpropagation



The vertical face-finding part of Rowley, Baluja and Kanade’s system

Figure from “Rotation invariant neural-network based face detection,” H.A. Rowley, S. Baluja and T. Kanade, Proc. Computer Vision and Pattern Recognition, 1998, copyright 1998, IEEE

Slide by D.A. Forsyth



Architecture of the complete system: they use another neural net to estimate orientation of the face, then rectify it. They search over scales to find bigger/smaller faces.

Figure from "Rotation invariant neural-network based face detection," H.A. Rowley, S. Baluja and T. Kanade, Proc. Computer Vision and Pattern Recognition, 1998, copyright 1998, IEEE

Slide by D.A. Forsyth

Figure from “Rotation invariant neural-network based face detection,” H.A. Rowley, S. Baluja and T. Kanade, Proc. Computer Vision and Pattern Recognition, 1998, copyright 1998, IEEE

Slide by D.A. Forsyth

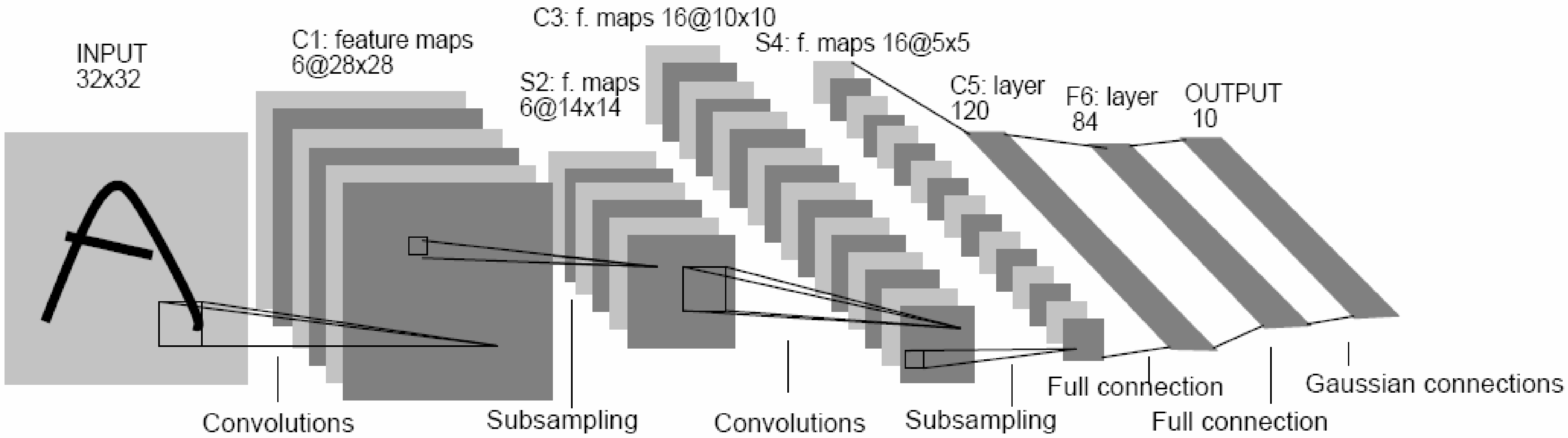
Convolutional neural networks

Template matching using NN classifiers seems to work

Natural features are filter outputs

- probably, spots and bars, as in texture
- but why not learn the filter kernels, too?

Slide by D.A. Forsyth



A convolutional neural network, LeNet; the layers filter, subsample, filter, subsample, and finally classify based on outputs of this process.

Figure from "Gradient-Based Learning Applied to Document Recognition", Y. Lecun et al Proc. IEEE, 1998 copyright 1998, IEEE

Slide by D.A. Forsyth

Fig. 4. Size-normalized examples from the MNIST database.

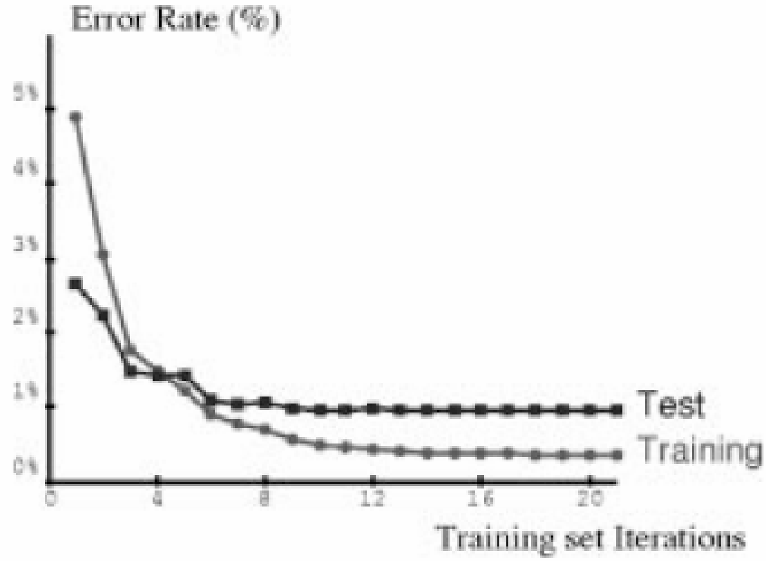
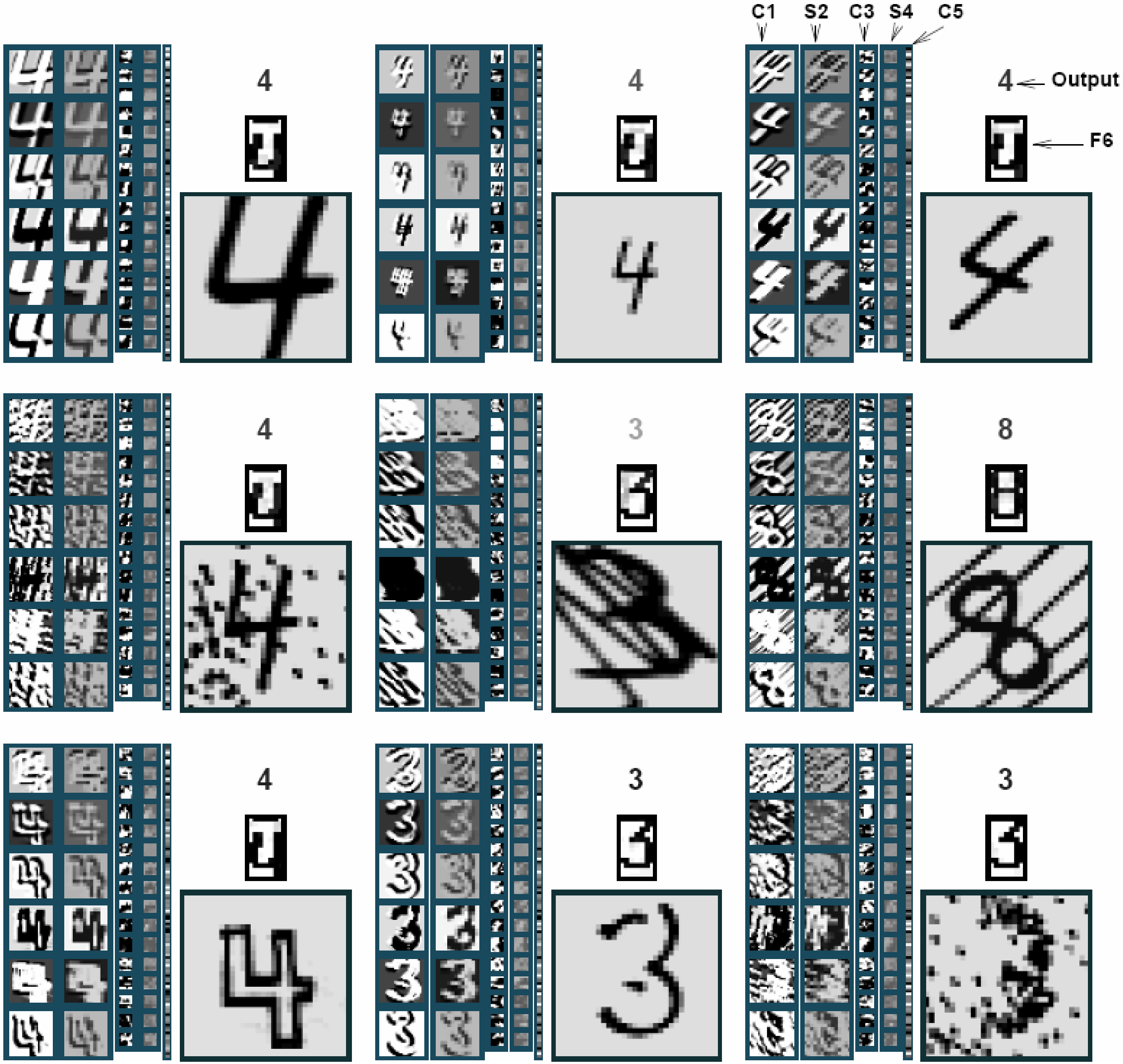


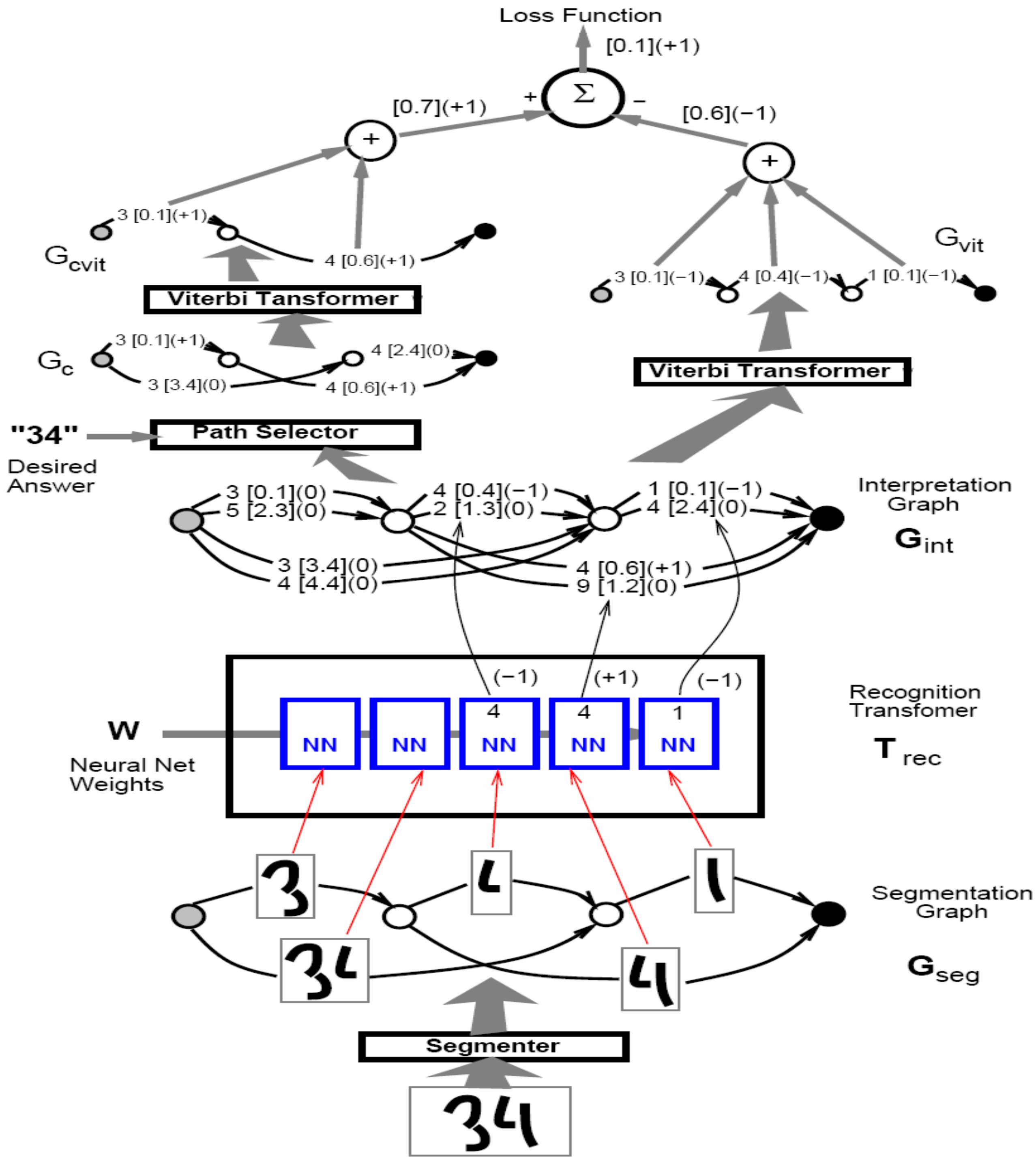
Fig. 5. Training and test error of LeNet-5 as a function of the number of passes through the 60 000 pattern training set (without distortions). The average training error is measured on-the-fly as training proceeds. This explains why the training error appears to be larger than the test error initially. Convergence is attained after 10–12 passes through the training set.

Figure from “Gradient-Based Learning Applied to Document Recognition”, Y. Lecun et al Proc. IEEE, 1998 copyright 1998, IEEE

Slide by D.A. Forsyth



C1
S2
C3
S4
C5
4
Output
F6
4
4
4
3
8
4
3
3



Loss Function
[0.1](+1)
[0.7](+1)
[0.6](−1)
Σ
3 [0.1](+1)
4 [0.6](+1)
G_{cvit}
Viterbi Tansformer
G_{vit}
3 [0.1](−1)
4 [0.4](−1)
1 [0.1](−1)
G_c
3 [0.1](+1)
3 [3.4](0)
4 [2.4](0)
4 [0.6](+1)
Viterbi Transformer
"34"
Desired Answer
Path Selector
3 [0.1](0)
5 [2.3](0)
4 [0.4](−1)
2 [1.3](0)
1 [0.1](−1)
4 [2.4](0)
3 [3.4](0)
4 [4.4](0)
4 [0.6](+1)
9 [1.2](0)
Interpretation Graph
G_{int}
W
Neural Net Weights
(−1)
(+1)
(−1)
NN
NN
4
NN
4
NN
1
NN
Recognition Transfomer
T_{rec}
Segmentation Graph
G_{seg}
Segmenter

Benchmarking SGD in Simple Problems

- The theory suggests that SGD is very competitive.
  - Many people associate SGD with trouble.

- SGD historically associated with back-propagation.
  - Multilayer networks are very hard problems (nonlinear, nonconvex)
  - What is difficult, SGD or MLP?

- Try PLAIN SGD on simple learning problems.
  - Support Vector Machines
  - Conditional Random Fields

Download from `http://leon.bottou.org/projects/sgd`.
These simple programs are very short.

See also (Shalev-Schwartz et al., 2007; Vishwanathan et al., 2006)

Text Categorization with SVMs

- **Dataset**
  - Reuters RCV1 document corpus.
  - 781,265 training examples, 23,149 testing examples.
  - 47,152 TF-IDF features.
- **Task**
  - Recognizing documents of category CCAT.
  - Minimize $E_n = \frac{1}{n}\sum_i \left(\frac{\lambda}{2} w^2 + \ell(\, w\, x_i + b,\; y_i\,) \right)$.
  - Update $w \leftarrow w - \eta_t \nabla(w_t, x_t, y_t) \;=\; w - \eta_t \left(\lambda w + \frac{\partial \ell(w\, x_t + b,\; y_t)}{\partial w} \right)$

Same setup as (Shalev-Schwartz et al., 2007) but plain SGD.

Text Categorization with SVMs

- **Results: Linear SVM**

  $\ell(\hat{y}, y) = \max\{0, 1 - y\hat{y}\} \quad \lambda = 0.0001$

	Training Time	Primal cost	Test Error
SVMLight	23,642 secs	*0.2275*	*6.02%*
SVMPerf	66 secs	0.2278	6.03%
SGD	1.4 secs	*0.2275*	*6.02%*

- **Results: Log-Loss Classifier**

  $\ell(\hat{y}, y) = log(1 + exp(-y\hat{y})) \quad \lambda = 0.00001$

	Training Time	Primal cost	Test Error
LibLinear ($\varepsilon = 0.01$)	30 secs	0.18907	*5.68%*
LibLinear ($\varepsilon = 0.001$)	44 secs	*0.18890*	5.70%
SGD	2.3 secs	0.18893	*5.66%*