Finding faces

- Faces "look like" templates (at least when they're frontal).
- General strategy:
  - search image windows at a range of scales
  - Correct for illumination
  - Present corrected window to classifier
- Issues
  - How corrected?
  - What features?
  - What classifier?
  - what about lateral views?

Details optional spring 2004

Naive Bayes

- (Important: naive not necessarily perjorative)
- Find faces by vector quantizing image patches, then computing a histogram of patch types within a face
- Histogram doesn't work when there are too many features
  - features are the patch types
  - assume they're independent and cross fingers
  - very effective for face finders
    - why? probably because the examples that would present real problems aren't frequent.

Many face finders on the face detection home page
http://home.t-online.de/home/Robert.Frischholz/face.htm

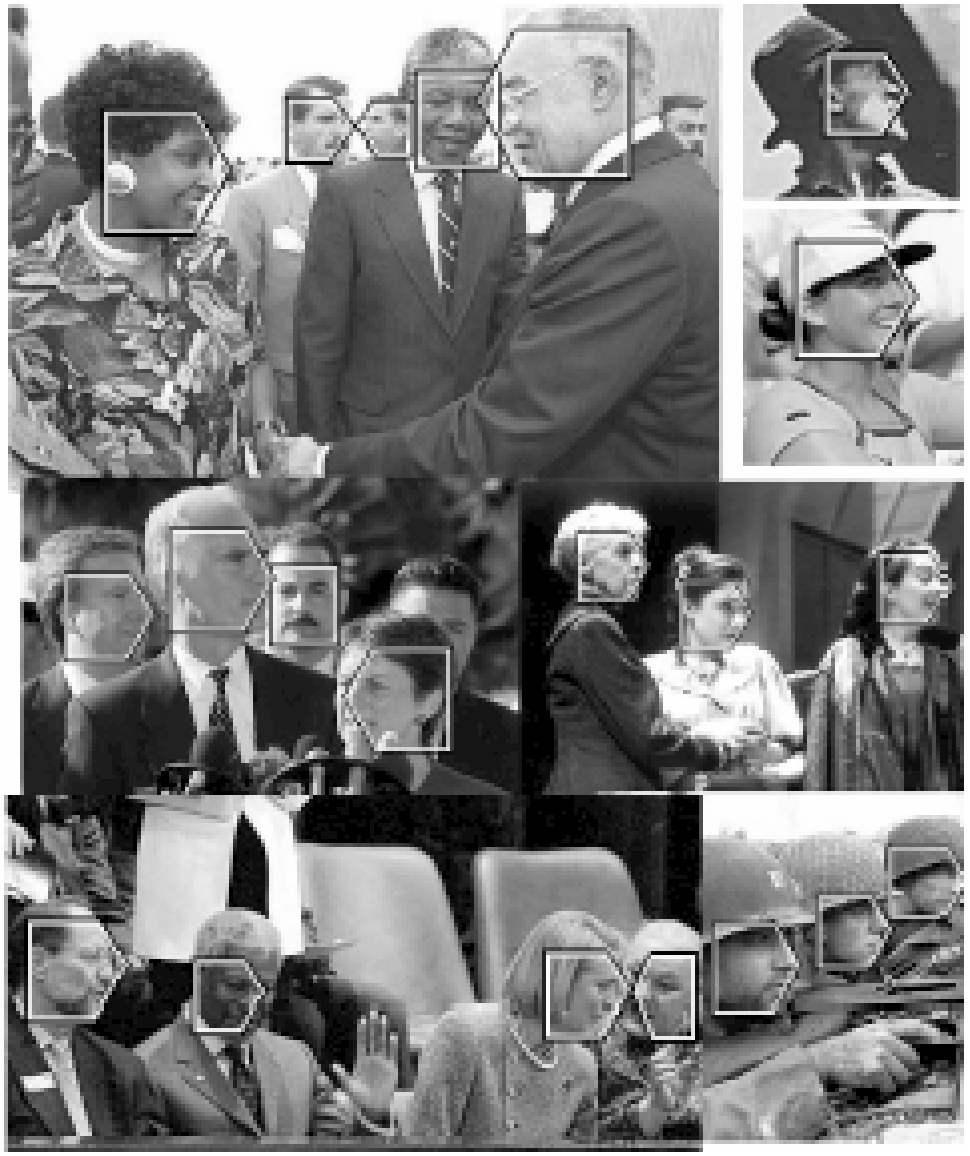


Figure from A Statistical Method for 3D Object Detection Applied to Faces and Cars, H. Schneiderman and T. Kanade, Proc. Computer Vision and Pattern Recognition, 2000, copyright 2000, IEEE

Face Recognition

- Whose face is this? (perhaps in a mugshot)
- Issue:
  - What differences are important and what not?
  - Reduce the dimension of the images, while maintaining the "important" differences.
- One strategy:
  - Principal components analysis (Eigenfaces)

- Many face recognition strategies at http://www.cs.rug.nl/users/peterkr/FACE/face.html

Details Optional (those specializing in vision will have to learn it sometime)

Assume we have a set of n feature vectors $\boldsymbol{x}_i$ $(i = 1, \ldots, n)$ in $\mathbb{R}^d$. Write

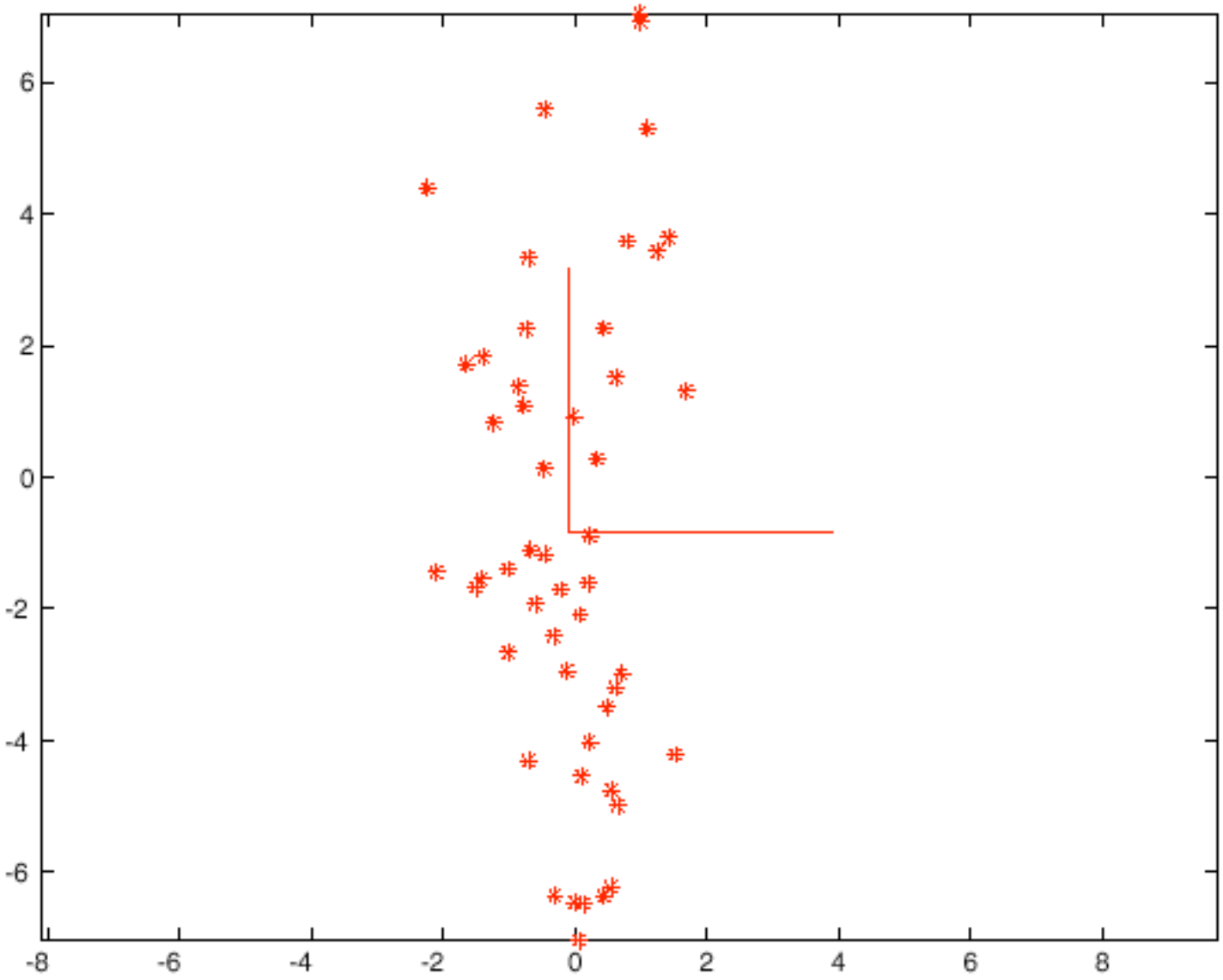
$$\boldsymbol{\mu} = \frac{1}{n}\sum_i \boldsymbol{x}_i$$

$$\Sigma = \frac{1}{n-1}\sum_i (\boldsymbol{x}_i - \boldsymbol{\mu})(\boldsymbol{x}_i - \boldsymbol{\mu})^T$$

The unit eigenvectors of Σ — which we write as $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_d$, where the order is given by the size of the eigenvalue and $\boldsymbol{v}_1$ has the largest eigenvalue — give a set of features with the following properties:

- They are independent.
- Projection onto the basis $\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k\}$ gives the k-dimensional set of linear features that preserves the most variance.

Algorithm 22.5: *Principal components analysis identifies a collection of linear features that are independent, and capture as much variance as possible from a dataset.*


6
4
2
0
-2
-4
-6
-8
-6
-4
-2
0
2
4
6
8

Difficulties with PCA

- Projection may suppress important detail
  - smallest variance directions may be important
- Method does not take discriminative task into account
  - typically, we wish to compute features that allow good discrimination, not the same as largest variance

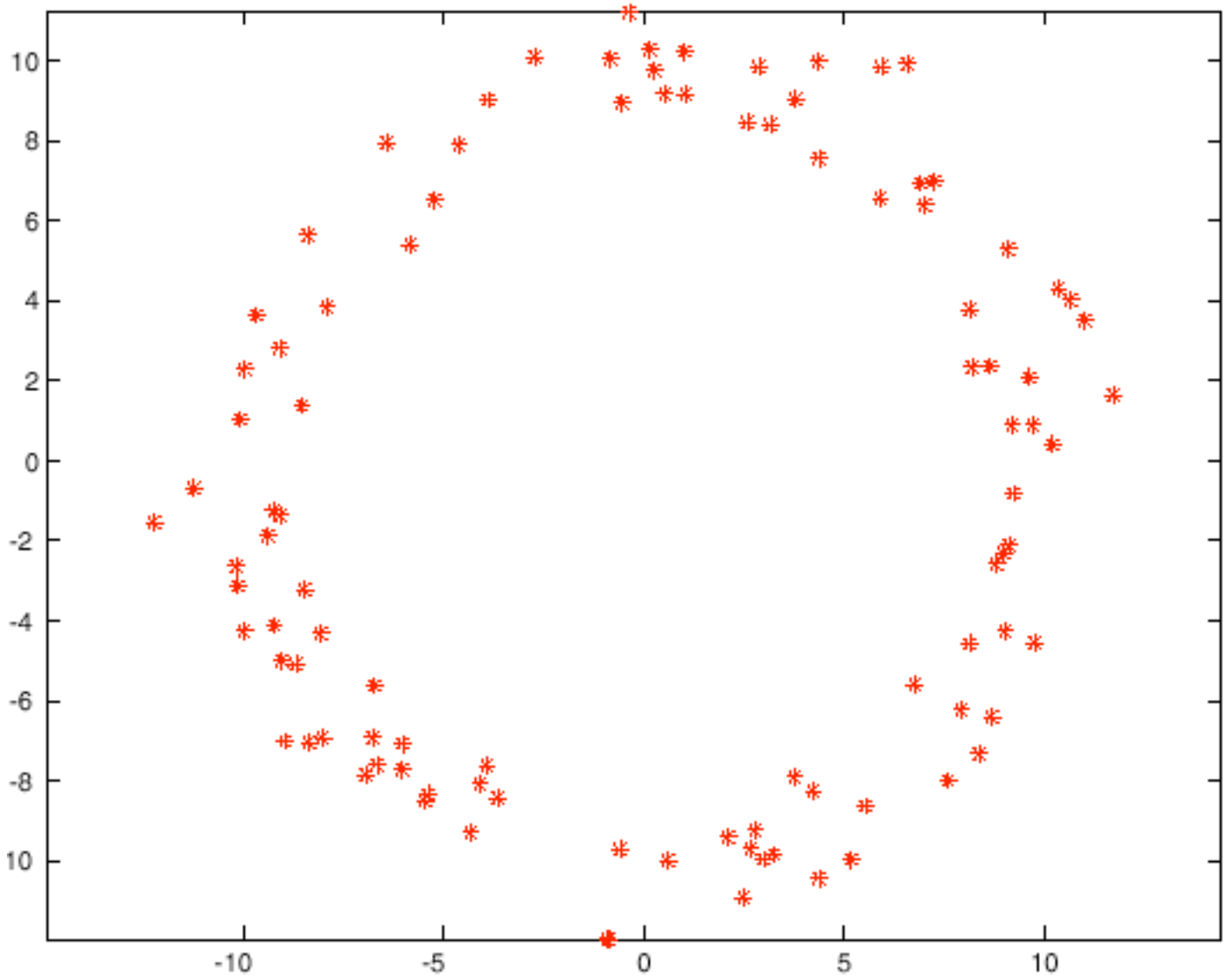


Fig 22.7 Principal components will give a very poor representation of this data set

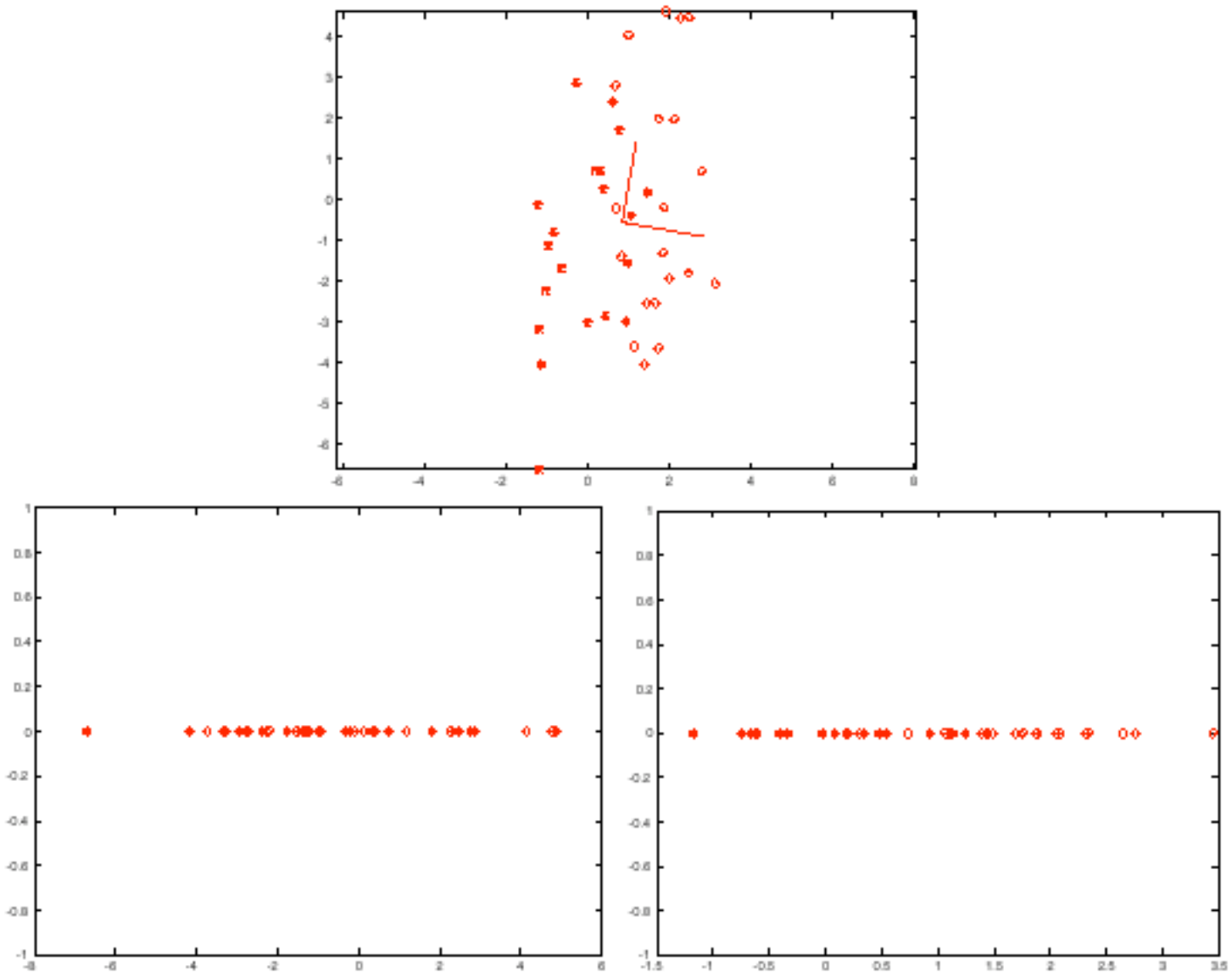


Fig 22.9 Principal components does not model the important property that the data might come from different classes

Details Optional

Linear Discriminant Analysis

- We wish to choose linear functions of the features that allow good discrimination.
  - Assume class-conditional covariances are the same
  - Want linear feature that maximizes the spread of class means for a fixed within-class variance

Optional

Algorithm 22.6: Canonical variates identifies a collection of linear features that separating the classes as well as possible.

Assume that we have a set of data items of g different classes. There are n_k items in each class, and a data item from the k'th class is $\boldsymbol{x}_{k,i}$, for $i \in \{1, \ldots, n_k\}$. The j'th class has mean $\boldsymbol{\mu}_j$. We assume that there are p features (i.e. that the $\boldsymbol{x}_i$ are p-dimensional vectors).

Write $\overline{\boldsymbol{\mu}}$ for the mean of the class means, i.e.

$$\overline{\boldsymbol{\mu}} = \frac{1}{g}\sum_{j=1}^{g}\boldsymbol{\mu}_j$$

Write

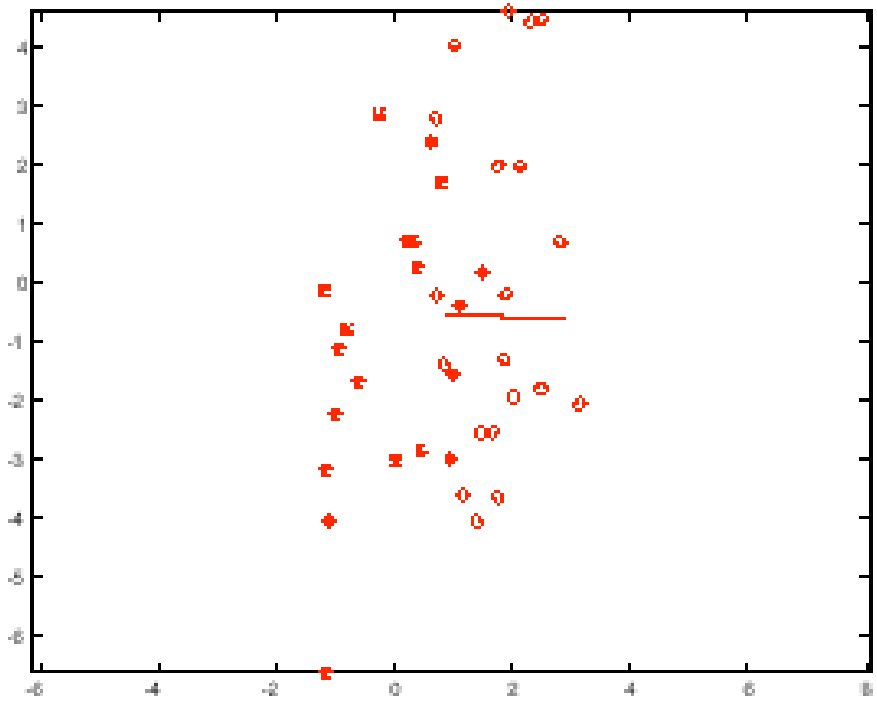
$$\mathcal{B} = \frac{1}{g-1}\sum_{j=1}^{g}(\boldsymbol{\mu}_j - \overline{\boldsymbol{\mu}})(\boldsymbol{\mu}_j - \overline{\boldsymbol{\mu}})^T$$

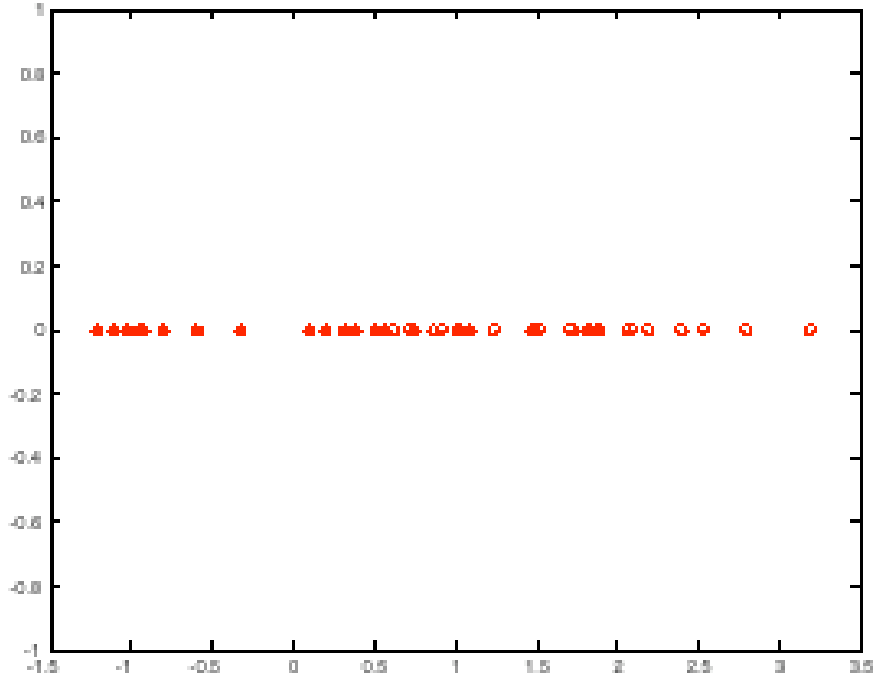
Assume that each class has the same covariance Σ, which is either known or estimated as

$$\Sigma = \frac{1}{N-1}\sum_{c=1}^{g}\left\{\sum_{i=1}^{n_c}(\boldsymbol{x}_{c,i} - \boldsymbol{\mu}_c)(\boldsymbol{x}_{c,i} - \boldsymbol{\mu}_c)^T\right\}$$

The unit eigenvectors of $\Sigma^{-1}\mathcal{B}$ — which we write as $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_d$, where the order is given by the size of the eigenvalue and $\boldsymbol{v}_1$ has the largest eigenvalue — give a set of features with the following property:

- Projection onto the basis $\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k\}$ gives the k-dimensional set of linear features that best separates the class means.


4
3
2
1
0
-1
-2
-3
-4
-5
-6
-6
-4
-2
0
2
4
6
8


1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5
3
3.5

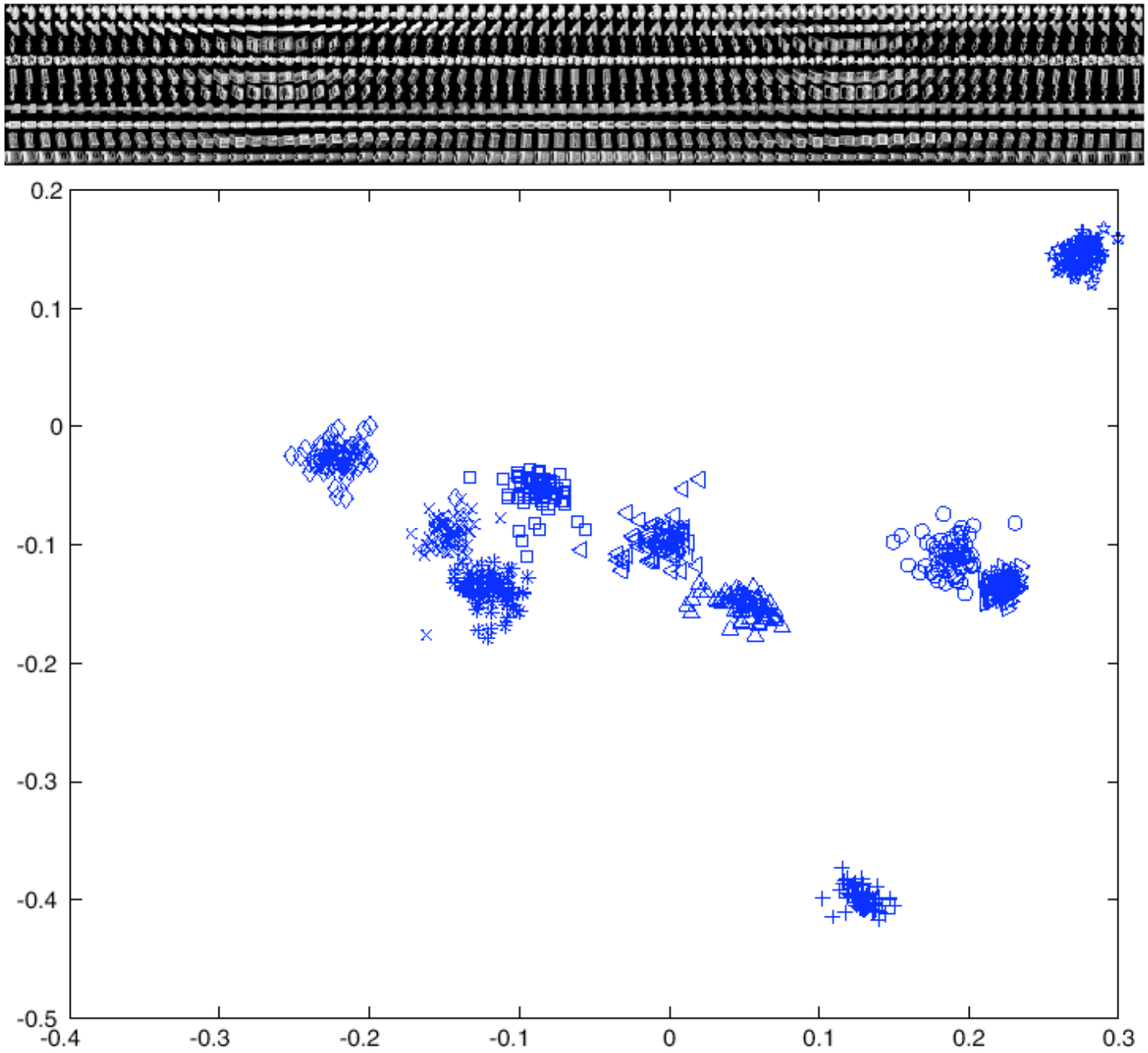
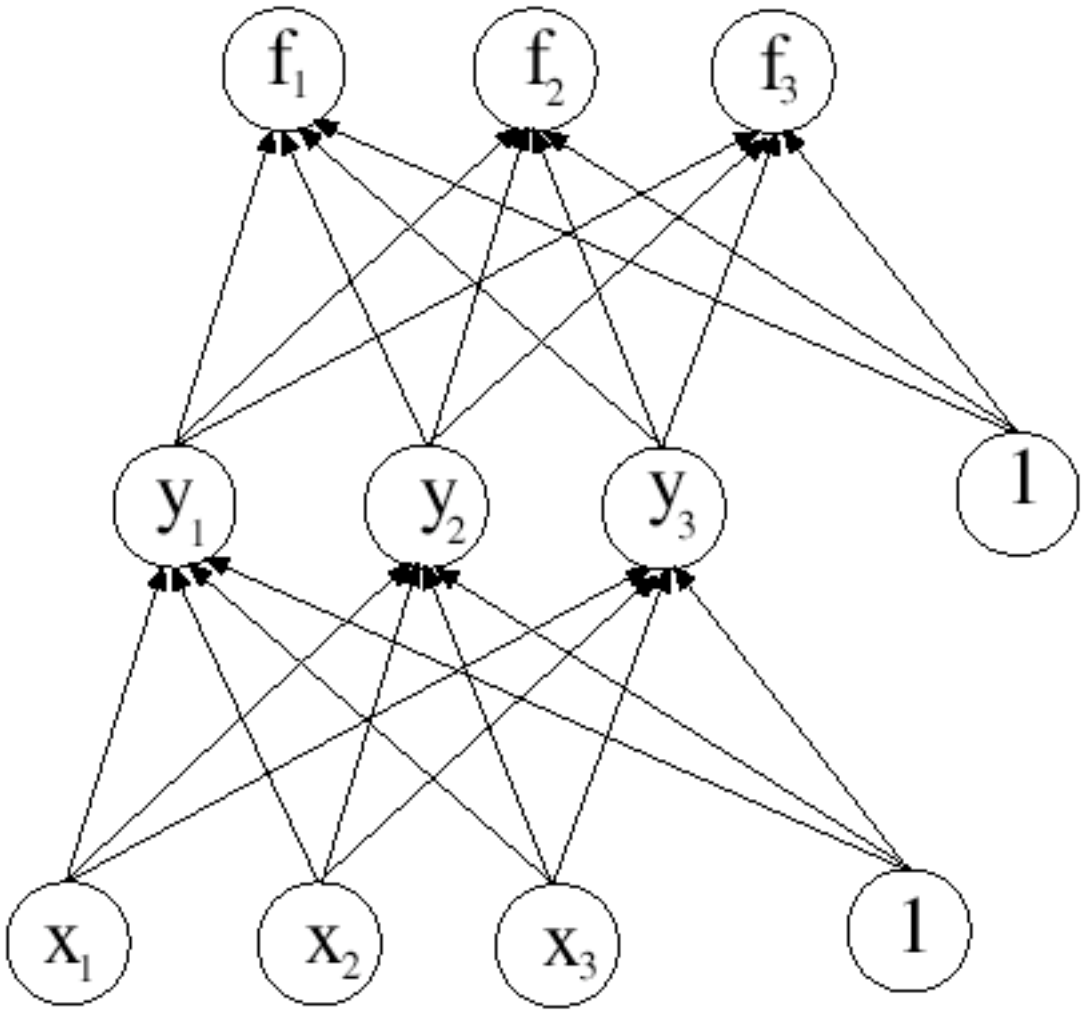


Figure 22.12

Skipped spring 2004

Neural networks

- Linear decision boundaries are useful
  - but often not powerful enough
  - we seek an easy way to get more complex boundaries
- Compose linear decision boundaries from simpler ones
  - i.e. have several linear classifiers, and apply a classifier to their output
  - a nuisance, because sign(ax+by+cz) etc. isn't differentiable.
  - use a smooth "squashing function" in place of sign.



Skipped spring 2004

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

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

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

Skipped spring 2004

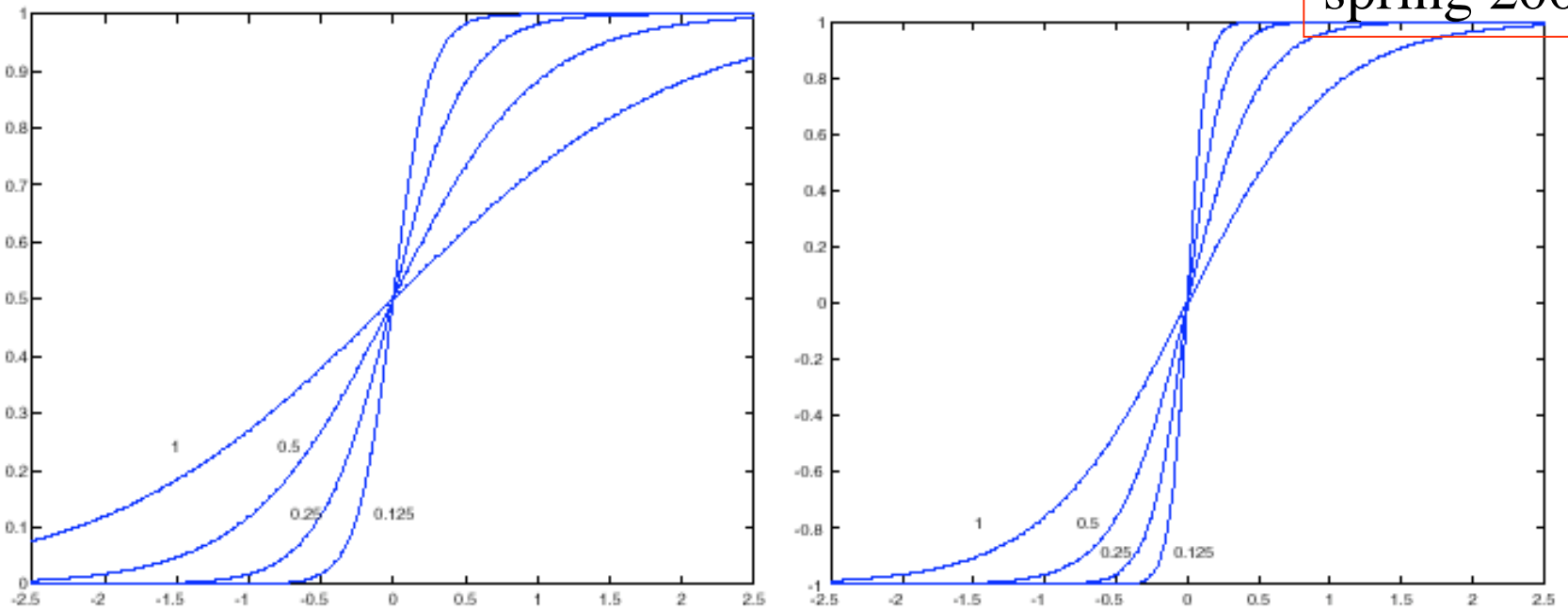


Figure 22.14. On the **left**, a series of squashing functions obtained using $\phi(x;\nu) = \frac{e^{x/\nu}}{1+e^{x/\nu}}$, for different values of ν indicated on the figure. On the **right**, a series of squashing functions obtained using $\phi(x;\nu,A) = A\tanh(x/\nu)$ for different values of ν indicated 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.

Skipped spring 2004

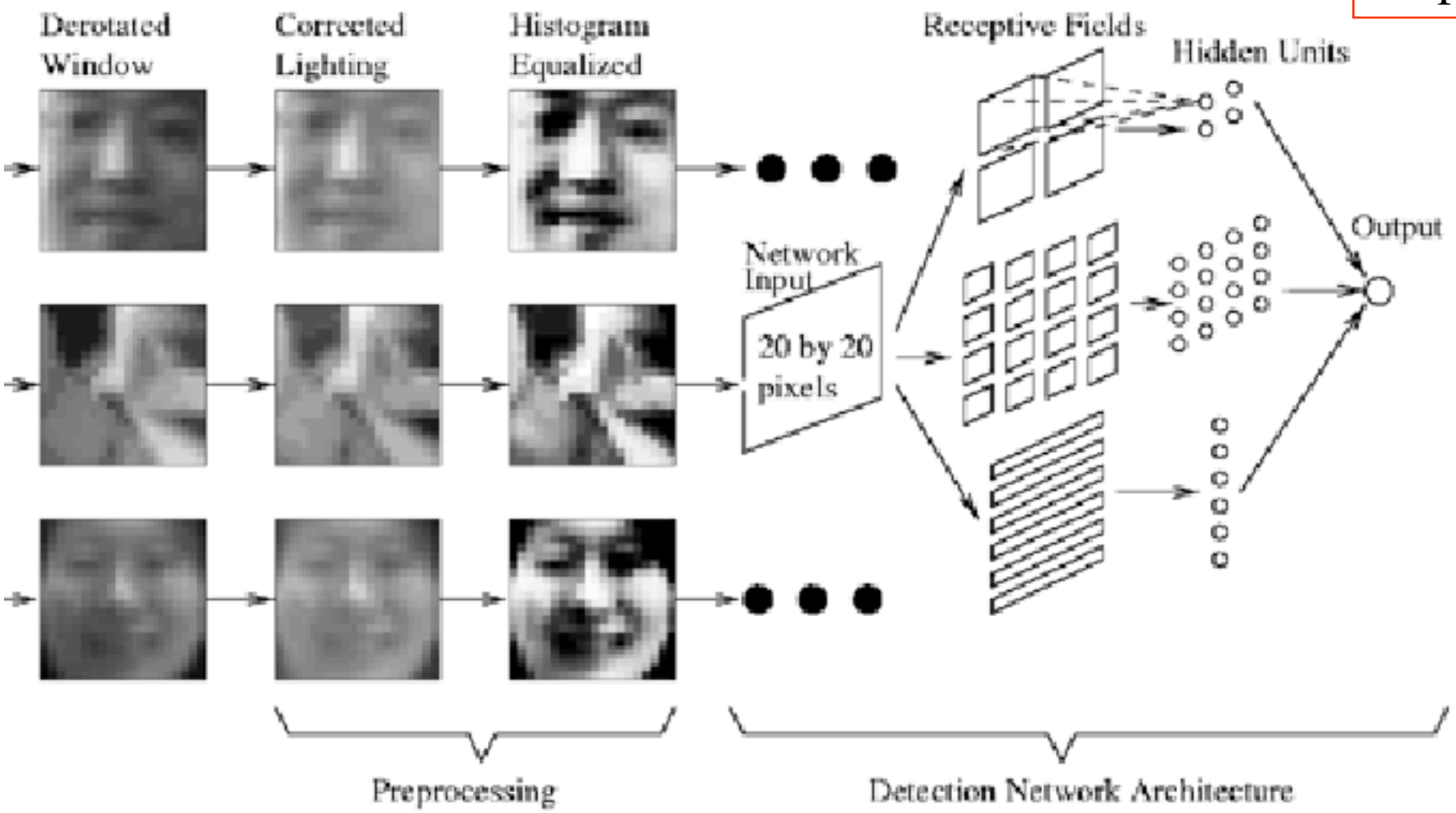
Training

- Choose parameters to minimize error on training set

$$Error(p) = \frac{1}{2} \sum_e \left(n(x^e; p) - o^e \right)$$

- Stochastic gradient descent, computing gradient using trick (backpropagation, aka the chain rule)
- Stop when error is low, and hasn't changed much

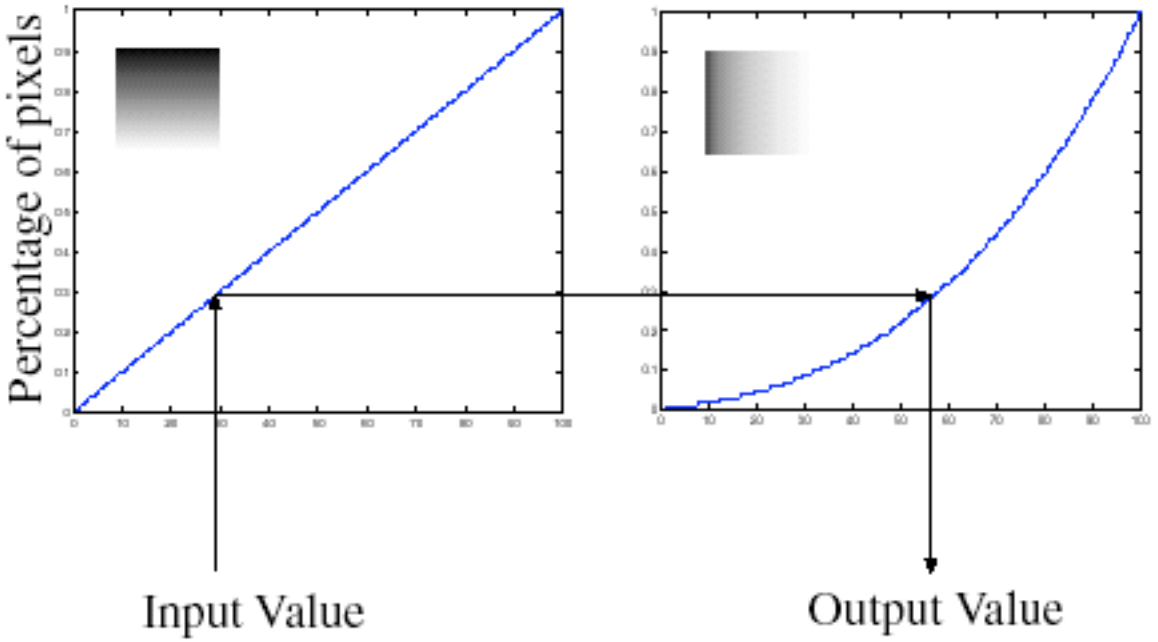
Skip



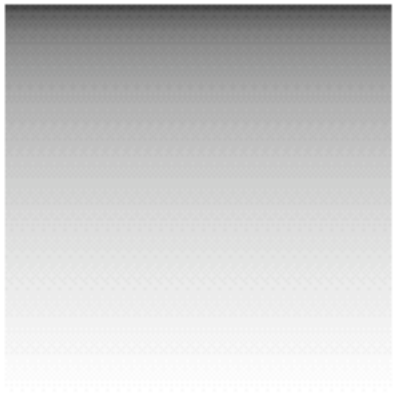
The vertical face-finding part of Rowley, Baluja and Kanade's system

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

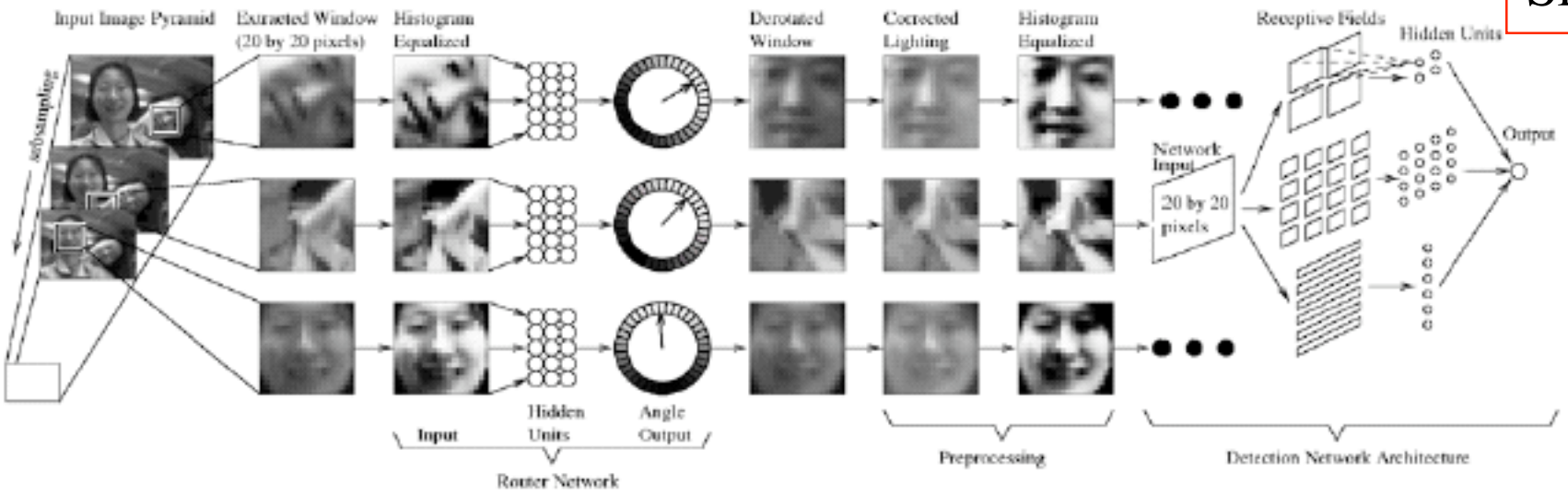
Skip



Histogram equalisation gives an approximate fix for illumination induced variability



Skip



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

Skip

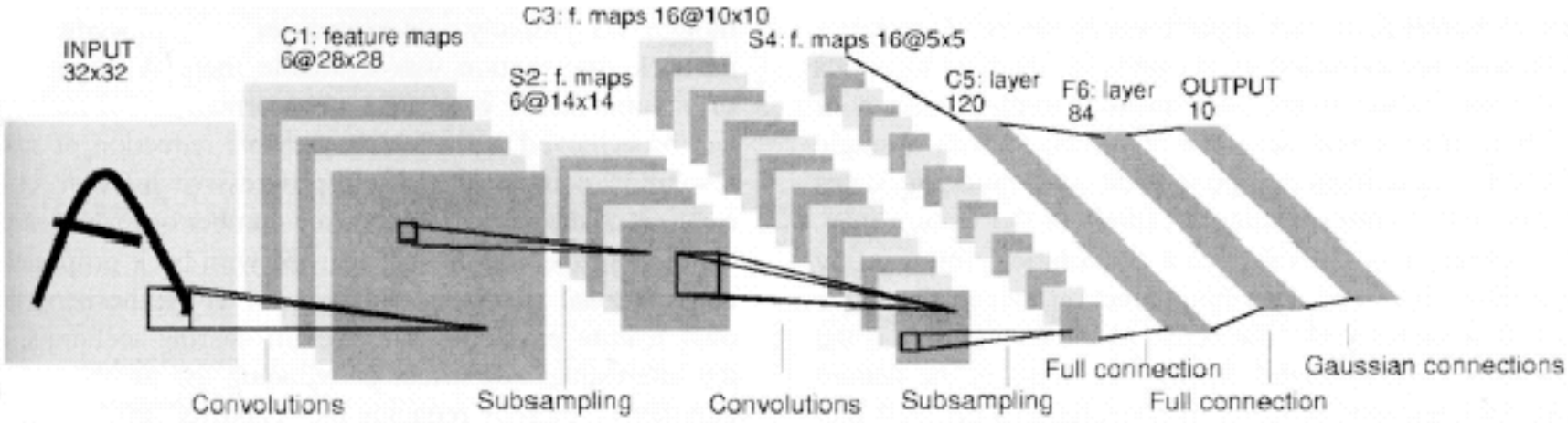
Figure from “Rotation invariant neural-network based face detection,” H.A. Rowley, S. Baluja and T. Kanade, Proc. Computer Vision and Pattern Recognition, 1998, copyright 1998, IEEE

Skip

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?

Skip



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

Skip

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

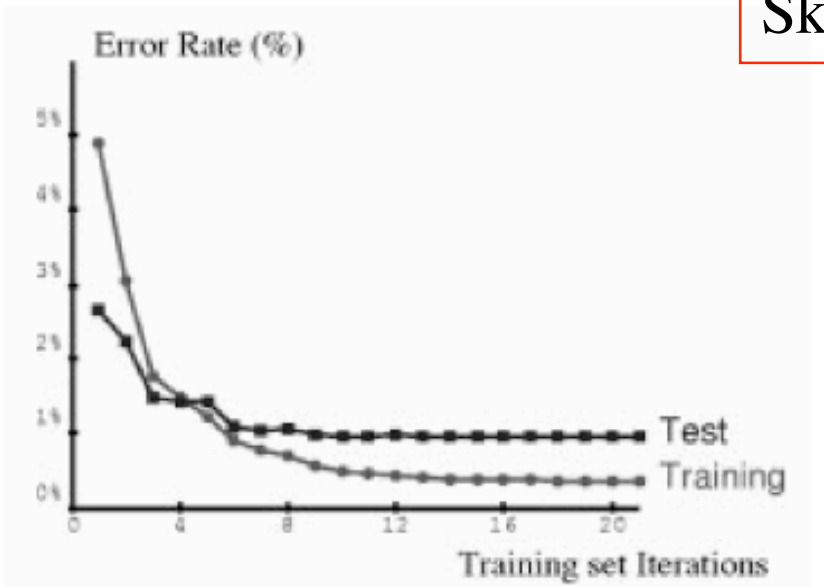


Fig. 5. Training and test error of LeNet-5 as a function of the number of passes through the 60 000 pattern training set (without distortions). The average training error is measured on-the-fly as training proceeds. This explains why the training error appears to be larger than the test error initially.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

LeNet is used to classify handwritten digits. Notice that the test error rate is not the same as the training error rate, because the test set consists of items not in the training set. Not all classification schemes necessarily have small test error when they have small training error.

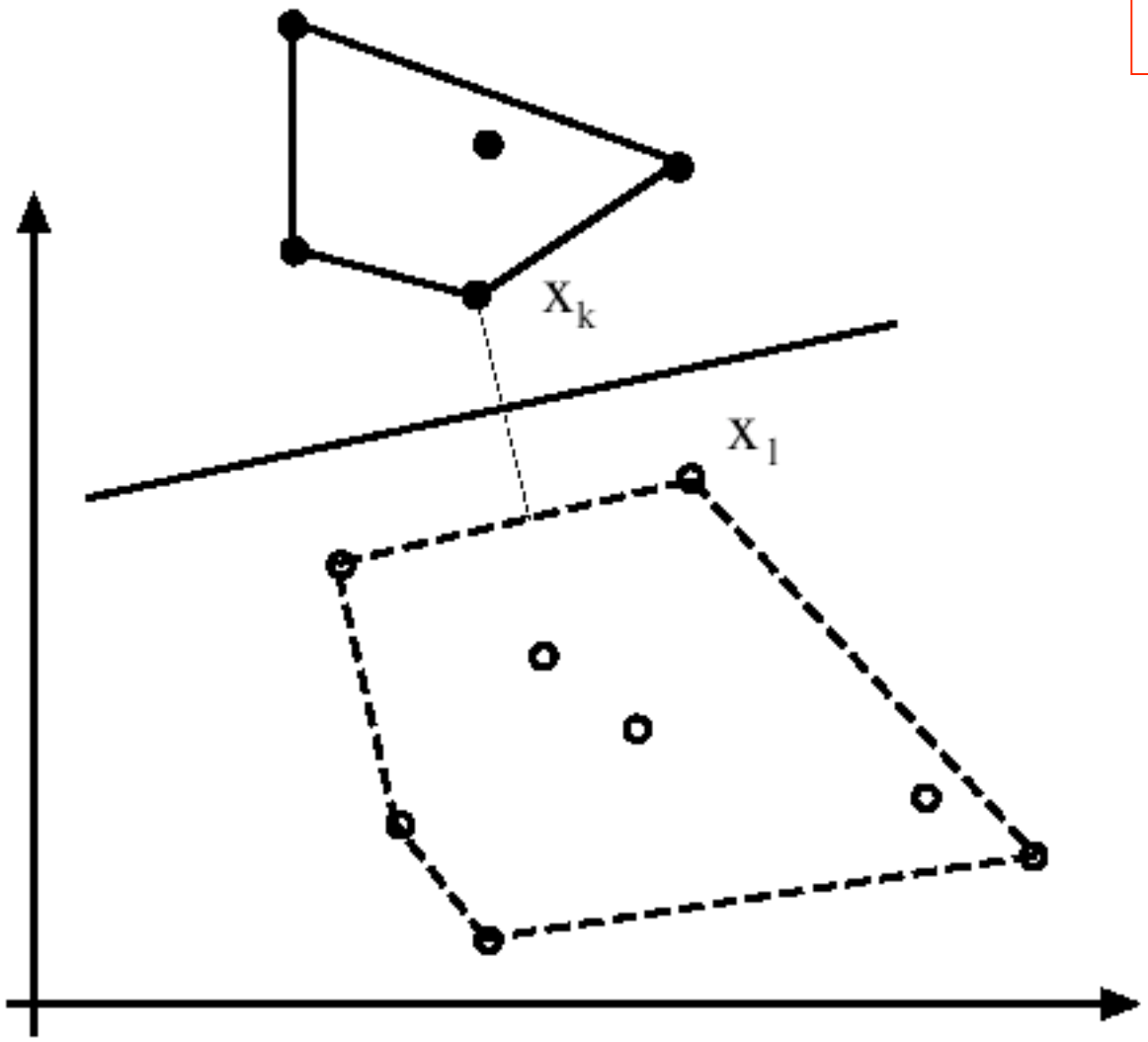
Figure from "Gradient-Based Learning Applied to Document Recognition", Y. Lecun et al Proc. IEEE, 1998 copyright 1998, IEEE

Skipped spring 2004

Support Vector Machines

- Neural nets try to build a model of the posterior, p(k|x)
- Instead, try to obtain the decision boundary directly
  - potentially easier, because we need to encode only the geometry of the boundary, not any irrelevant wiggles in the posterior.
  - Not all points affect the decision boundary

Skipped
spring 2004



Skipped spring 2004

Support Vector Machines

- Linearly separable data means

$$y_i \left(\boldsymbol{w} \cdot \boldsymbol{x}_i + b \right) > 0$$

- Can scale so that

$$y_i \left(\boldsymbol{w} \cdot \boldsymbol{x}_i + b \right) \geq 1$$

- Distance (for points with equality but on opposite

$$\begin{aligned} dist(\boldsymbol{x}_k, \text{hyperplane}) + dist(\boldsymbol{x}_l, \text{hyperplane}) &= \left(\frac{\boldsymbol{w}}{|\boldsymbol{w}|} \cdot \boldsymbol{x}_k + \frac{b}{|\boldsymbol{w}|}\right) - \left(\frac{\boldsymbol{w}}{|\boldsymbol{w}|} \cdot \boldsymbol{x}_1 + \frac{b}{|\boldsymbol{w}|}\right) \\ &= \frac{\boldsymbol{w}}{|\boldsymbol{w}|} \cdot (\boldsymbol{x}_1 - \boldsymbol{x}_2) = \frac{2}{|\boldsymbol{w}|} \end{aligned}$$

Skipped spring 2004

Support Vector Machines

$$\text{minimize} \quad (1/2)\boldsymbol{w} \cdot \boldsymbol{w}$$

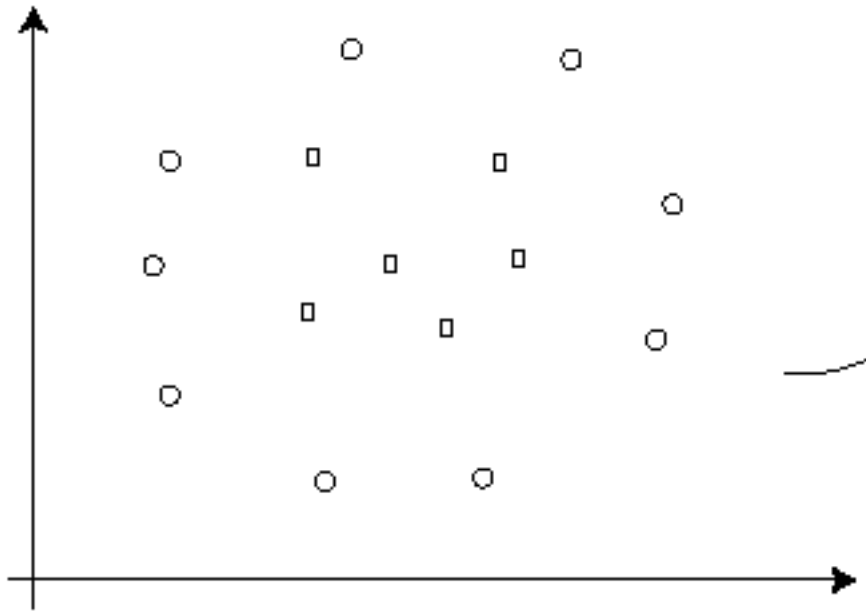
$$\text{subject to} \quad y_i \left(\boldsymbol{w} \cdot \boldsymbol{x}_i + b\right) \geq 1$$

Details on the solution left for another day!

By being clever about what x means, I can have much more interesting boundaries.

Skipped spring 2004

$$(x, y) \square \left(x^2, xy, y^2, x, y\right) = \left(u_0, u_1, u_2, u_3, u_4\right)$$



Space in which decision boundary is linear - a conic in the original space has the form

$$au_0 + bu_1 + cu_2 + du_3 + eu_4 + f = 0$$

Vision applications

- Reliable, simple classifier,
  - use it wherever you need a classifier
- Commonly used for face finding and many other things
- Currently one the latest popular hammers in a number of fields--you need to understand it in order to understand when it is approapriate and when it is NOT.

The future of vision is bright

- Computation is cheap
- Lots of images
  - cameras are cheap, many pictures are digital
- Lots of demand for “slicing and dicing” pictures
  - generate models
  - new movies from old
  - search
- Lots of “hidden value”
  - can’t do data mining for collections with images or image like data
- Scientific applications--stacks of scientific data that needs tools for analysis
- “Ubiquitous computing”

Recent flowering of vision

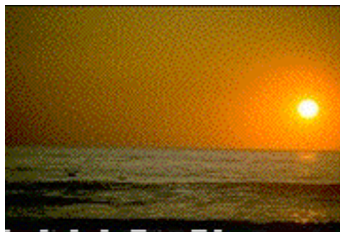
- can do (sort of!)
  - structure from motion
  - segmentation
  - video representation
  - model building
  - tracking
  - face finding
- will be able to do (sort of!)
  - face recognition
  - inference about people
  - character recognition
  - perhaps more

Big open problems

- Next step in structure from motion
- Really good missing variable formalism
- Methods for finding good parameters in complex posterior distributions
- Decent understanding of illumination, materials and shading
- Segmentation

- Representation for recognition
- Efficient management of relations
- Recognition processes for lots of objects

- A lot of this looks like applied statistics

The END