PMR5406 Redes Neurais e Lógica Fuzzy

Aula 5
Alguns Exemplos

APPLICATIONS

- Two examples of real life applications of neural networks for pattern classification:
  - RBF networks for face recognition
  - FF networks for handwritten recognition

FACE RECOGNITION

- **The problem:**
  - **Face recognition of persons of a known group in an indoor environment.**
- The approach:
  - Learn face classes over a wide range of poses using an RBF network.
- PhD thesis by Jonathan Howelland, Sussex University
  http://www.cogs.susx.ac.uk/users/jonh/index.html

Dataset

- **Sussex database** (university of Sussex)
  - **100 images of 10 people** (8-bit grayscale, resolution 384 x 287)
  - for each individual, 10 images of head in different pose **from face-on to profile**
  - Designed to asses performance of **face recognition techniques** when pose variations occur

Robustness to shift-invariance, scale-variance

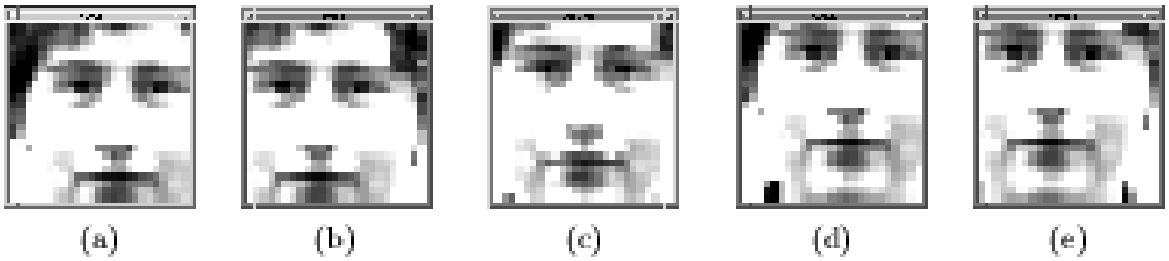


Figure 2: **Shift-varying** data for the 'face on' view of one individual: (a) top left (b) top right (c) normal view (d) bottom left (e) bottom right

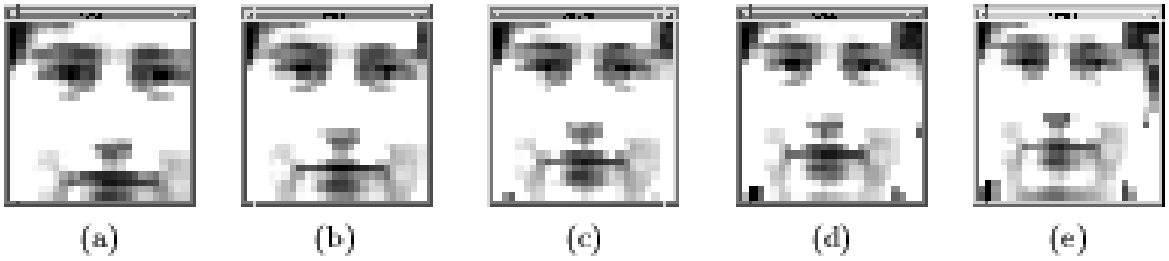
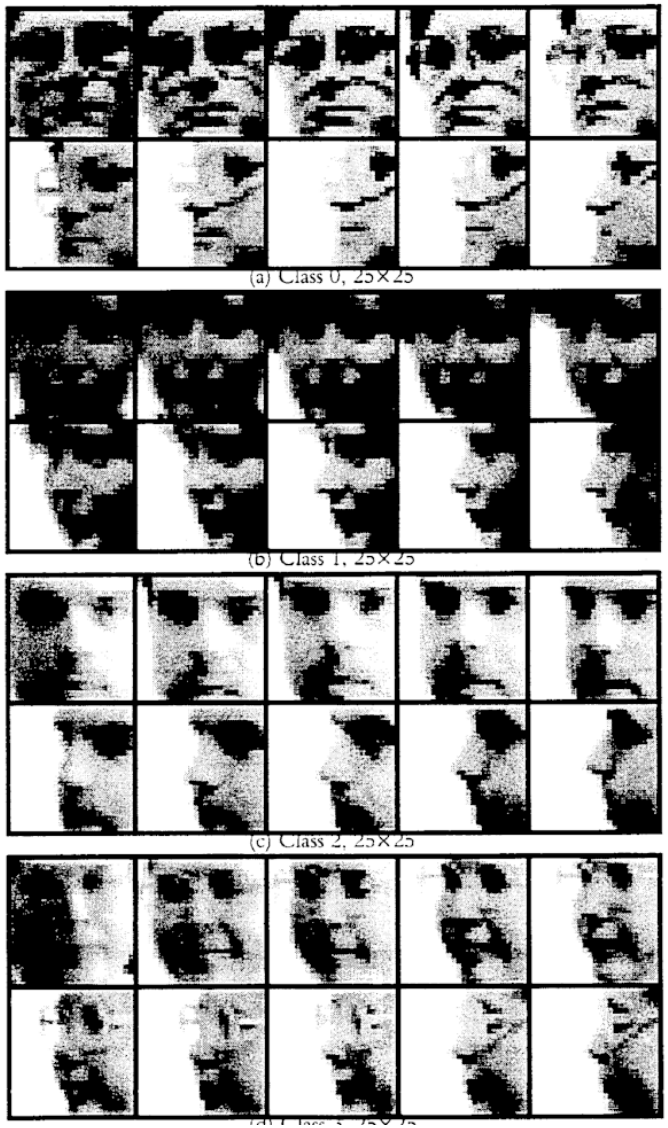


Figure 3: **Scale-varying** data for the 'face on' view of one individual: (a) +25% (uses 111×111 window) (b) +12.5% (107×107) (c) normal view (100×100) (d) -12.5% (94×94) (e) -25% (87×87)

Datasets (Sussex)

All ten images for classes 0-3 from the Sussex database, nose-centred and subsampled to 25x25 before preprocessing



Pre-Processing

- Raw data can be used, but with pre-processing.
- Possible approaches:
  - Difference of Gaussians (DoG) Pre-Processing.
  - Gabor Pre-Processing.

Some justification

of scales. One way of thinking about these input representations and mapping them onto our RBF networks is to use the analogy with visual neurons. The receptive field of such a neuron is the area of the visual field (image) where the stimulus can influence its response. For the different classes of these neurons, a receptive field function $f(x, y)$ can be defined. For example, retinal ganglion cellas and lateral geniculate cells early in the visual processing have receptive fields which can be implemented as Difference of Gaussian filters (Marr & Hildreth 1980). Later, the receptive fields of the simple cells in the primary visual cortex are oriented and have characteristic spatial frequencies. Daugman (1988) proposed that these could be modelled as complex 2-D Gabor filters. Petkov et al. (1993) successfully implemented a face recognition scheme based on Gabor wavelet input representations to imitate the human vision system. The question we want to ask here is whether these later stages of processing make more information explicit than the earlier DoG filters for our face recognition task.

DoG

5 Difference of Gaussians (DoG) Pre-Processing

Where there is a change of intensity in an image, peaks or troughs are found in the first derivative of the intensity, and zero-crossings in the second derivative. To isolate the latter, Marr & Hildreth (1980) suggested the $\nabla^2 G$, or *Laplacian of the Gaussian*, operator, which can be closely apprimated by a *Difference of Gaussians* (DoG) operator, constructed from two Gaussians G of the form:

$$G(x, y) = \frac{1}{\sigma^2} \exp(-\frac{x^2 + y^2}{2\sigma^2}), \qquad (1)$$

where the space constants σ have a ratio of 1:1.6. The DoG masks were constructed using the POPVISION CONVOLVE_DoG_2D routines. Figure 5 shows these masks at various scale values, whilst Figure 2 shows the result of their convolution with an image at a fixed resolution.

DoG masks

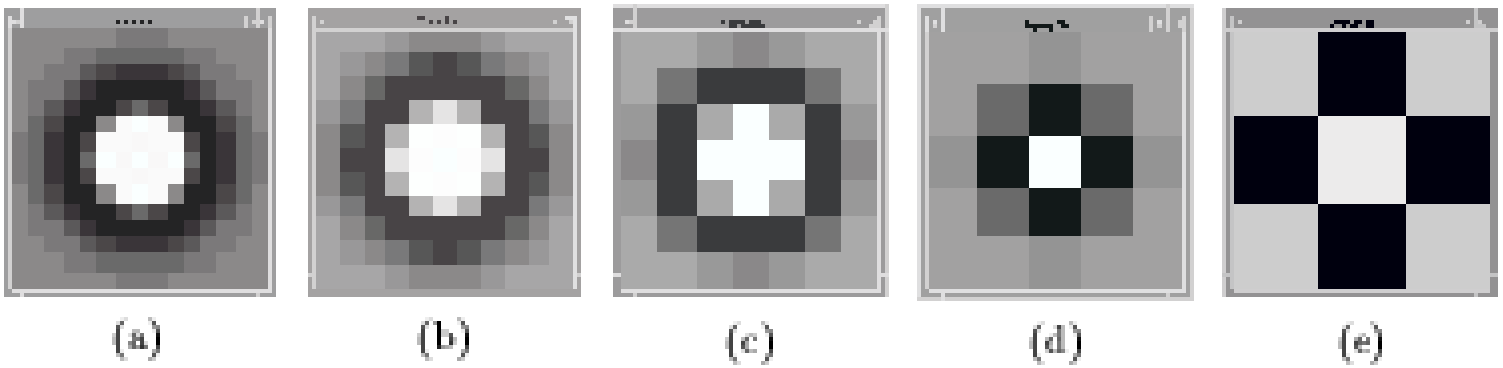


Figure 1: Masks created from various DoG scales (with mask sizes): (a) 1.6 (15×15) (b) 1.2 (11×11) (c) 0.8 (7×7) (d) 0.4 (5×5) (e) 0.15 (3×3)

Some examples:

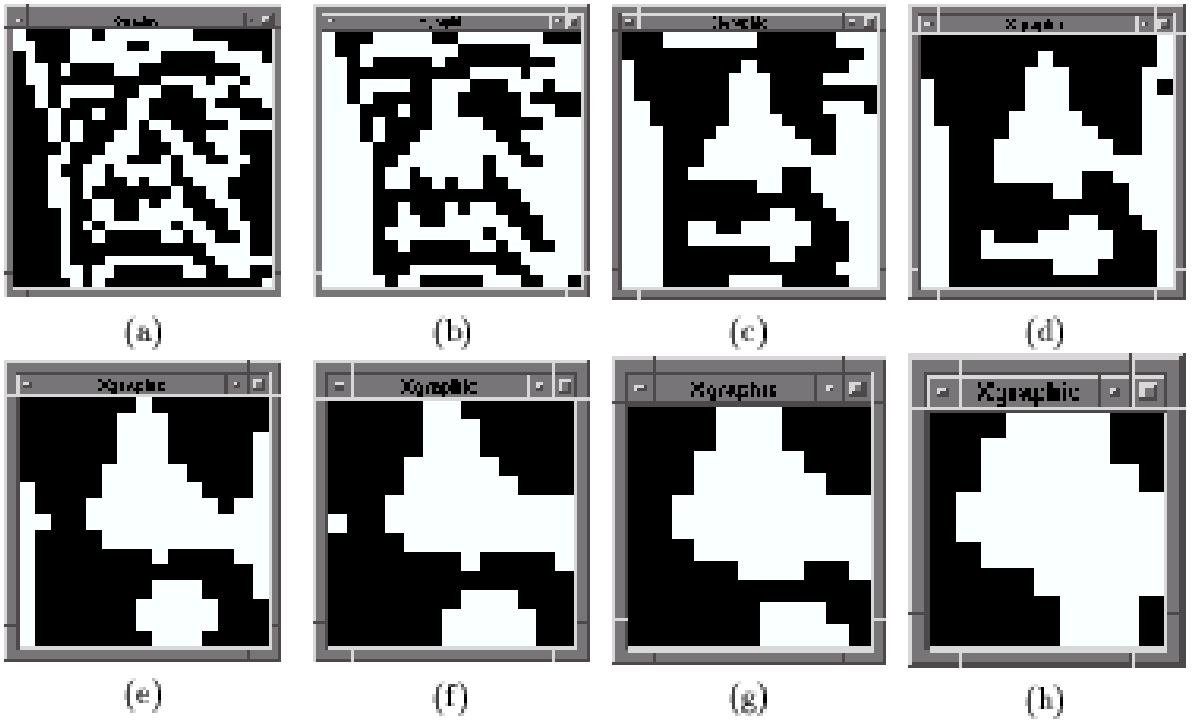


Figure 2: DoG scales applied to 25×25 image (with convolved image sizes): (a) 0.15 (23×23) (b) 0.4 (21×21) (c) 0.8 (19×19) (d) 1.0 (17×17) (e) 1.2 (15x15) (f) 1.4 (13x13) (g) 1.6 (11x11) (h) 1.9 (9x9)

Binarisation (1)

5.1 DoG Gradients *vs.* 'Zero-Crossings'

With a typical, grey level image, such as Figure 3(a), DoG convolution will give continuously-valued[2] gradient information, as shown in Figure 3(b). Where these values change from one sign to the other is the 'zero-crossing' point; if the values are thresholded at 0 into either 0 (for negative) and 1 (for positive), the boundaries between black and white are the zero-crossings for the image, as shown in Figure 3(c). To test how useful it was to explicitly concentrate only on this boundary point, preprocessing was carried out with and without this thresholding stage. This produced both gradient DoG (non-binarised) and zero-crossings DoG (binarised) information.

Binarisation (2)

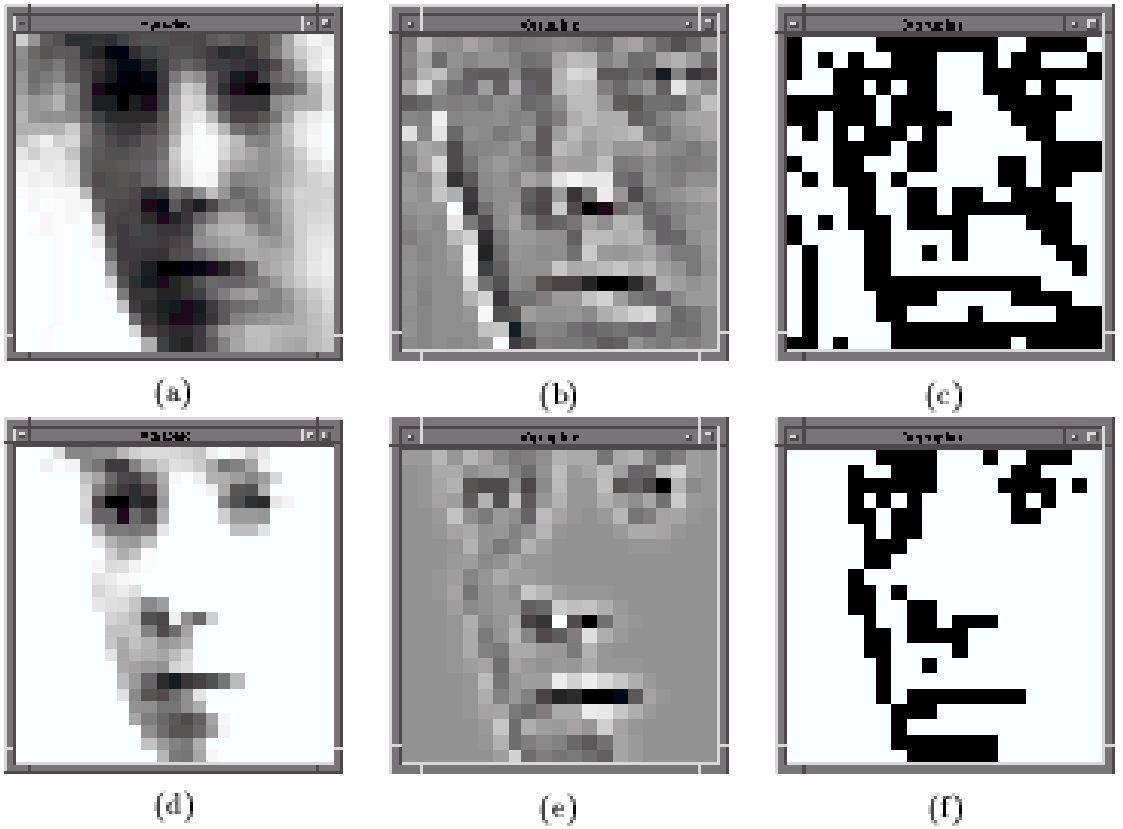


Figure 3: Effect of reducing range of grey-levels on 25×25 image (a) full range of grey-levels (b) after non-thresholded DoG (c) after thresholded DoG (d) reduced range of grey-levels (e) after non-thresholded DoG (f) after thresholded DoG

Gabor Pre-Processing

6 Gabor Pre-Processing

6.1 Gabor scales and orientations

We have selected Gabor filters (Daugman 1988) as an alternative preprocesing method, as it provides oriented information, which, we hope, will provide input information for the network in a more useful form than the previous methods. One disadvantage of isolated orientation-specific value is that if a full convolution of the image is carried out, more values are output than input (as there is a data value for each pixel for each orientation required). In addition, there are sine and cosine components of the Gabor filter, which doubles the number of coefficients produced.

The Gabor masks were constructed using the POPVISION GABORMASK routines, using three parameters: σ for width, p the period of the harmonic component ($\sigma = p/(2\sqrt{2})$), and o the orientation of the mask.

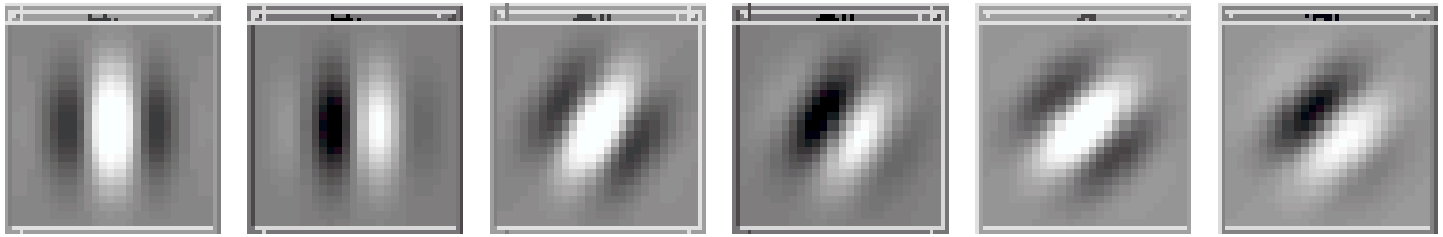
The real (cosine) component, C, of the Gabor mask is calculated as:

$$C(x,y) = N \exp(-\frac{r^2}{2\sigma^2}) cos(x'\omega), \tag{2}$$

where $r^2 = x^2 + y^2$, $x' = x\ cos(o) + y\ sin(o)$, and $\omega = (2\pi)/p$, and N is a real normalisation constant. The imaginary (sine) component, S, is:

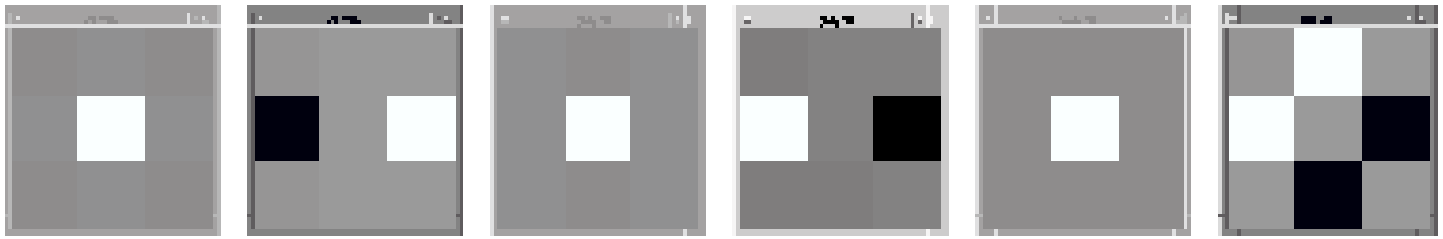
$$S(x,y) = N \exp(-\frac{r^2}{2\sigma^2}) sin(x'\omega). \tag{3}$$

Gabor Masks



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

Figure 6: 25×25 masks created from Gabor filter of period 13: (a) 0° real (b) 0° imaginary (c) 30° real (d) 30° imaginary (e) 45° real (f) 45° imaginary



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

Figure 7: 3×3 masks created from Gabor filter of period 1: (a) 0° real (b) 0° imaginary (c) 30° real (d) 30° imaginary (e) 45° real (f) 45° imaginary

Approach: Face unit RBF

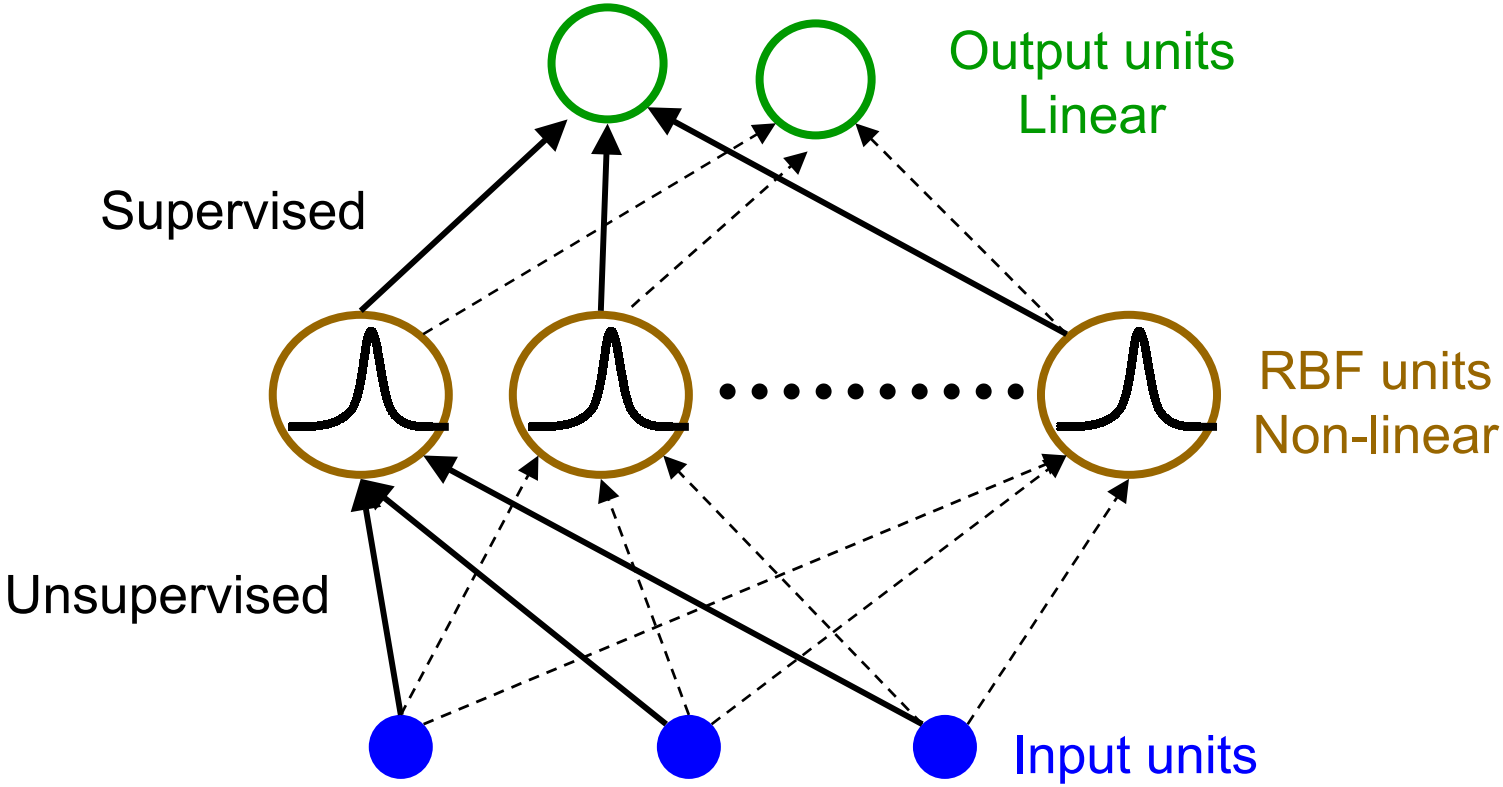
- A face recognition unit RBF neural networks is trained to recognize a single person.
- Training uses examples of images of the person to be recognized as positive evidence, together with selected confusable images of other people as negative evidence.

Network Architecture

- Input layer contains 25*25 inputs which represent the pixel intensities (normalized) of an image.
- Hidden layer contains p+a neurons:
  - p hidden pro neurons (receptors for positive evidence)
  - a hidden anti neurons (receptors for negative evidence)
- Output layer contains two neurons:
  - One for the particular person.
  - One for all the others.

The output is discarded if the absolute difference of the two output neurons is smaller than a parameter R.

RBF Architecture for one face recognition



The output o for hidden unit h (for a pattern l) can be expressed as:

$$o_h(l) = \exp[-\frac{|\mathbf{i}(l) - \mathbf{c}_h|^2}{2\sigma_h^2}], \tag{1}$$

the hidden layer output being unit-normalised, as suggested by [17]. For output unit i, the output is:

$$o_i(l) = \sum_h w_{ih}\, o_h(l). \tag{2}$$

Hidden Layer

- Hidden nodes can be:
  - Pro neurons: Evidence for that person.
  - Anti neurons: Negative evidence.
- The number of pro neurons is equal to the positive examples of the training set. For each pro neuron there is either one or two anti neurons.
- Hidden neuron model: Gaussian RBF function.

The Parameters

- Centers:
  - of a pro neuron: the corresponding positive example
  - of an anti neuron: the negative example which is most similar to the corresponding pro neuron, with respect to the Euclidean distance.
- Spread: average distance of the center vector from all other centers. If α, *h* hidden nodes, H total number of hidden nodes then:

$$\sigma_\alpha = \frac{1}{H\sqrt{2}} \sum_h \| t^\alpha - t^h \|$$

- Weights: determined using the pseudo-inverse method.
- A RBF network with 6 pro neurons, 12 anti neurons, and R equal to 0.3, discarded 23 pro cent of the images of the test set and classified correctly 96 pro cent of the non discarded images.

Handwritten Digit Recognition Using Convolutional Networks

- **Developed by Yann Lecun while working at the AT&T**

HANDWRITTEN DIGIT RECOGNITION

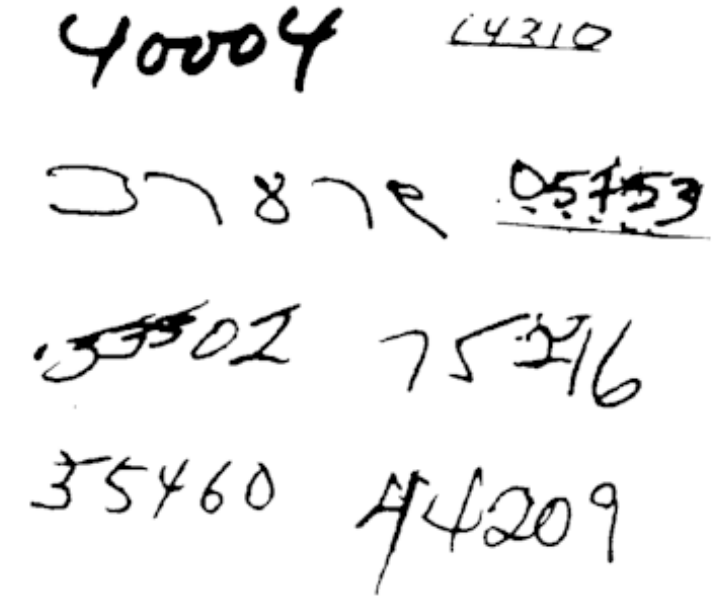


FIGURE 10.8
Examples of ZIP code image, and segmented and normalized numerals from the testing set. (*Source:* Reprinted with permission from Y. Le Cun, et al., "Backpropagation Applied to Handwritten Zip Code Recognition," *Neural Computation*, 1:541–551, 1989. ©1989 The MIT Press.)

Convolutional Network

- Convolutional network is a multilayer perceptron designed specifically to recognize two-dimensional shapes with a high degree of invariance to translation, scaling, skewing and other forms of distortions.

Characteristics (1)

1. Feature extraction: each neuron takes its synaptic input from a local receptive field from the previous layer. It extracts local features

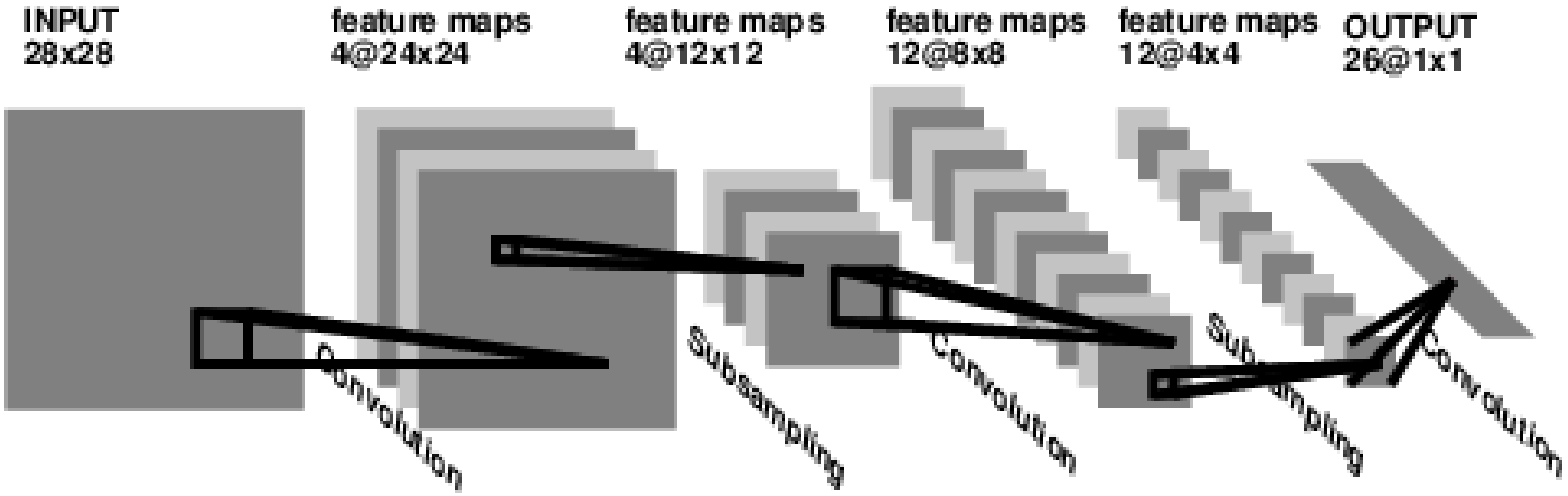
Characteristics (2)

2. Feature mapping: each computational layer is composed of multiple feature maps. In each feature, map neurons must share weights. This promote:
    - Shift invariance,
    - Reduction in the number of the free parameters.

Chracteristics (3)

3. Subsampling: each convolutional layer is followed by a computational layer that peforms local averaging and subsampling. This has the effect of reducing the sensitivity to shift´s and other forms of distortions.

Architecture (0)



Architecture (1)

- Input layer: 28x28 sensory nodes
- First hidden layer: convolution, 4@24x24 neurons feature map. Each neuron is assigned a receptive field of 5x5.
- Second hidden layer: subsampling and local averaging. 4@12x12 neurons feature map. Each neuron: 2x2 receptive field, weight, bias and sigmoid activation function

Architecture (2)

- Third hidden layer: convolution. 12@8x8 neurons feature map. Each neuron may have synaptic connections from several feature maps in the previous hidden layer.
- Fourth hidden layer: subsampling and averaging. 12@4x4 neurons feature map.

Architecture (3)

- Output layer: Convolution, 26@1x1neurons one for each character. Each neurons is connected to a receptive field of 4x4.

Architecture (4)

- Convolution -> subsampling -> convolution -> subsampling -> ...
- At each convolutional or subsampling layer, the number of feature maps is increased while the spatial resolution is reduced.
- 100.000 synaptic conections but only 2.600 free parameters.