Region Description for Recognition

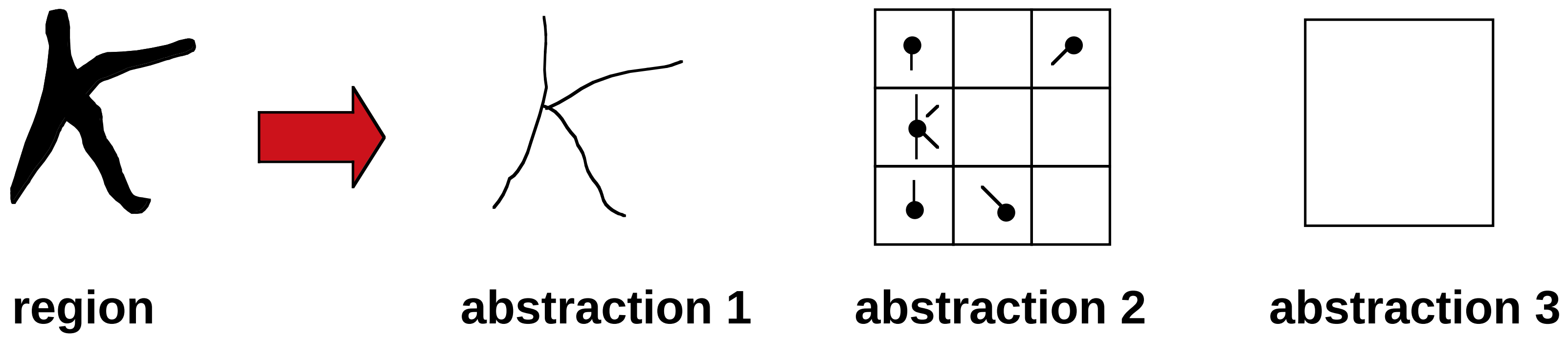
For object recognition, descriptions of regions in an image have to be compared with descriptions of regions of meaningful objects (models).

The general problem of object recognition will be treated later.

Here we learn basic region description techniques for later stages in image analysis (including recognition).

Typically, region descriptions suppress (abstract from) irrelevant details and expose relevant properties. What is "relevant" depends on the task.

Example: OCR (Optical Character Recognition)



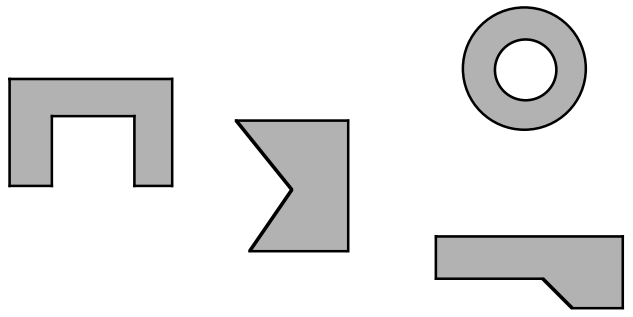
Simple 2D shape features

For industrial recognition tasks it is often required to distinguish

- a small number of different shapes
- viewed from a small number of different view points
- with a small computational effort.

In such cases simple 2D shape features may be useful, such as:

- area
- boxing rectangle
- boundary length
- compactness
- second-order momentums
- polar signature
- templates



Features may or may not have invariance properties:

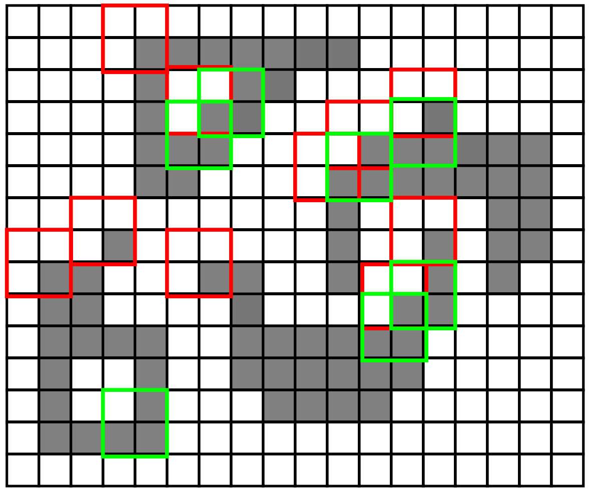
- 2D translation invariance
- 2D rotation invariance
- scale invariance

Euler number

The Euler number is the difference between the number of disjoint regions and the number of holes in an image.

P = number of parts
H = number of holes
E = P - H

Example:
P = 5
H = 2
E = 3

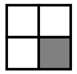


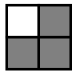
Surprisingly, E (but not P or H) can be computed by simple local operators.

Operators for regions with asymmetric connectivity:

4-connected NE and SW

8-connected NW and SE

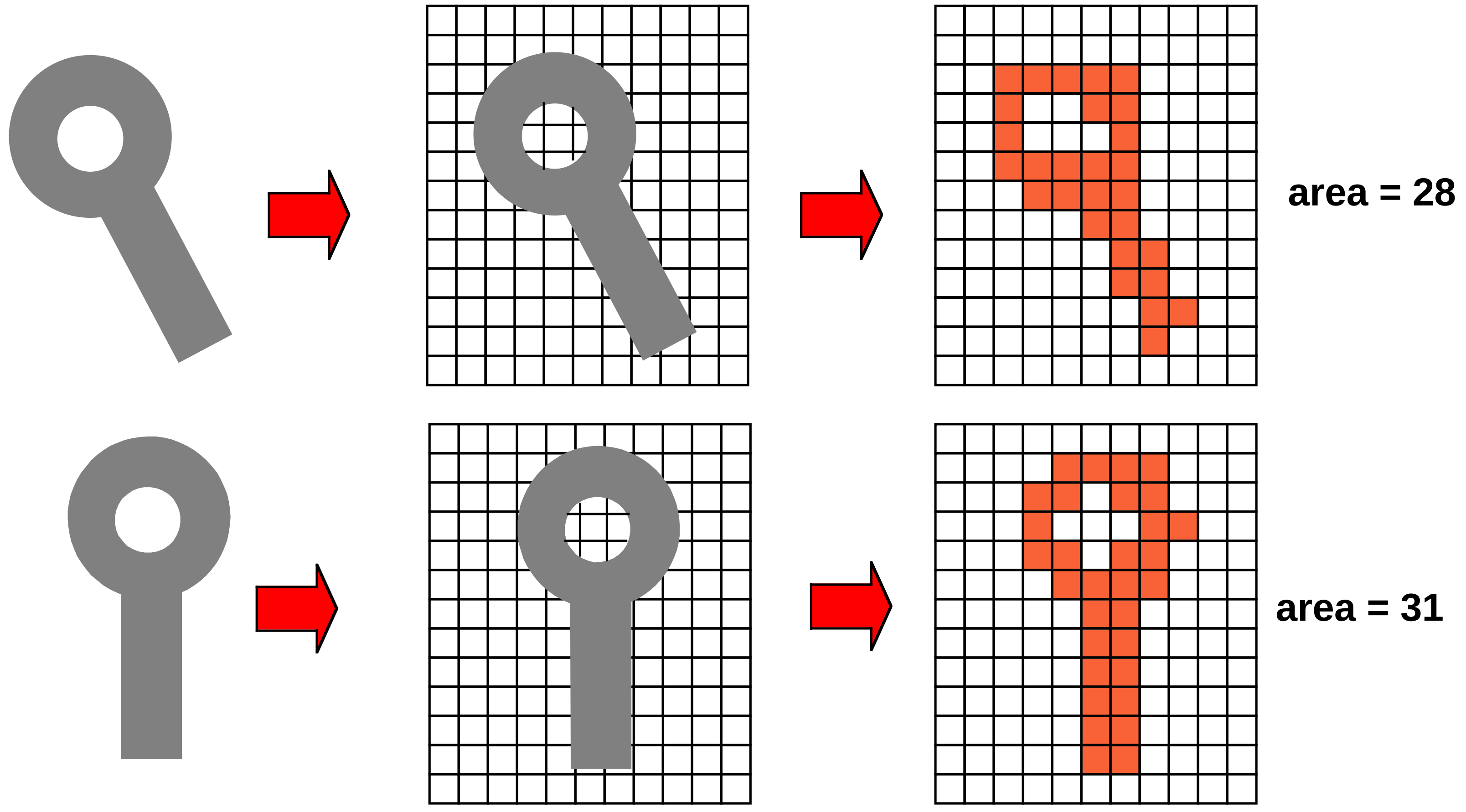
pattern1 = 

pattern2 = 

E = (count of pattern1) - (count of pattern2)

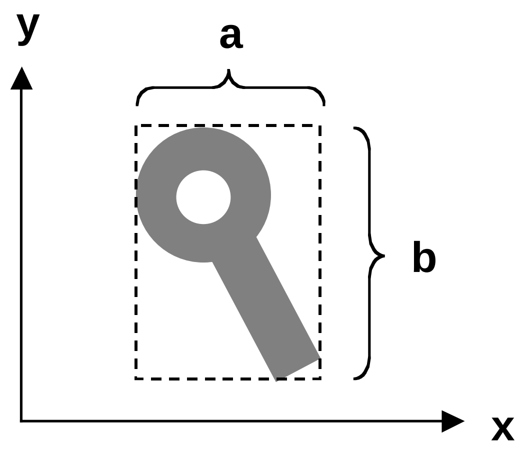
Area

The area of a digital region is defined as the number of pixels of the region.
For an arbitrarily fine resolution, area is translation and rotation invariant.
In praxis, discretization effects may cause considerably variations.



Boxing rectangle

Boxing rectangle = width of a shape in x- and y-direction



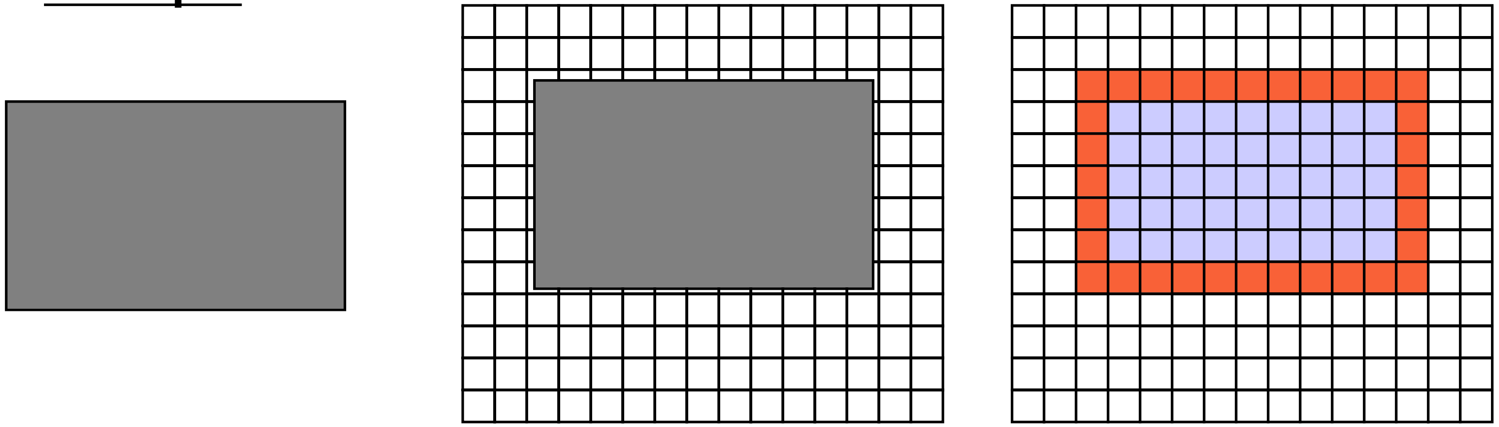
- easy to compute
- <u>not</u> rotation invariant

To achieve rotation invariance, the rectangle must be fitted parallel to an innate orientation of the shape. Orientation can be determined as the axis of least inertia (see second order moments).

Boundary length

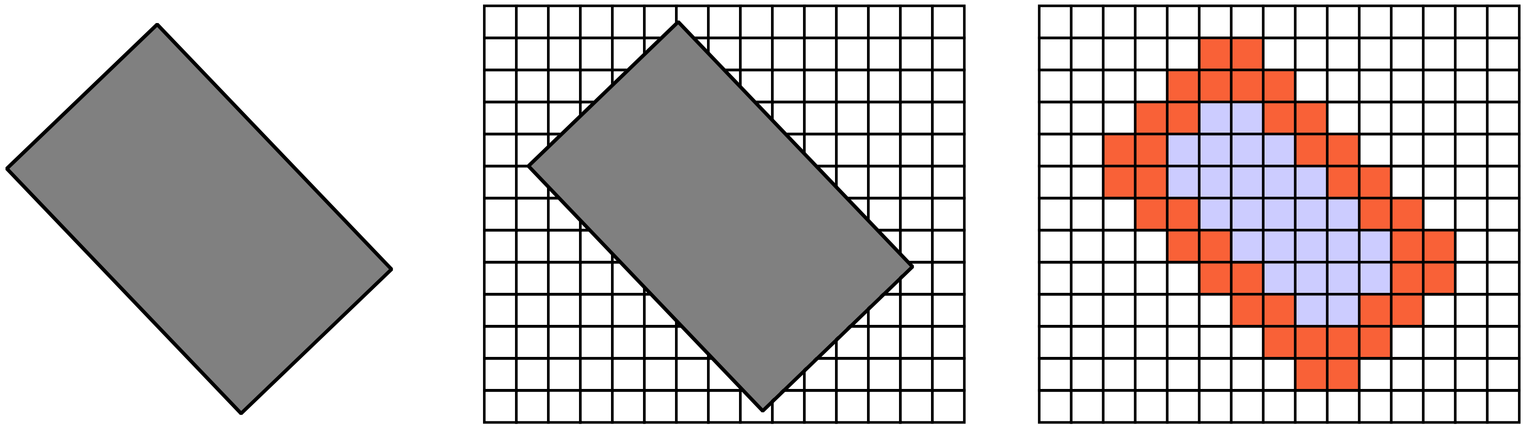
The boundary length is defined as the number of pixels which constitute the boundary of a shape.

<u>Example</u>:



area = 77

boundary length = 32



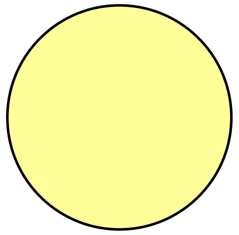
area = 69

boundary length = 40

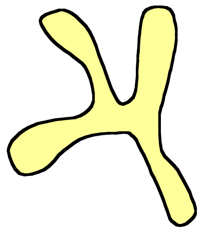
Compactness

$$\text{compactness} = \frac{(\text{boundary length})^2}{\text{area}}$$

Compactness describes analog shapes independent of linear transformations.



very compact



not very compact

Compactness for discrete shapes is in general not translation, rotation or scale invariant due to discretization effects.

Center of gravity

Consider a 2D shape evenly covered with mass. Physical concepts such as

- center of gravity
- moments of inertia

may be applied.

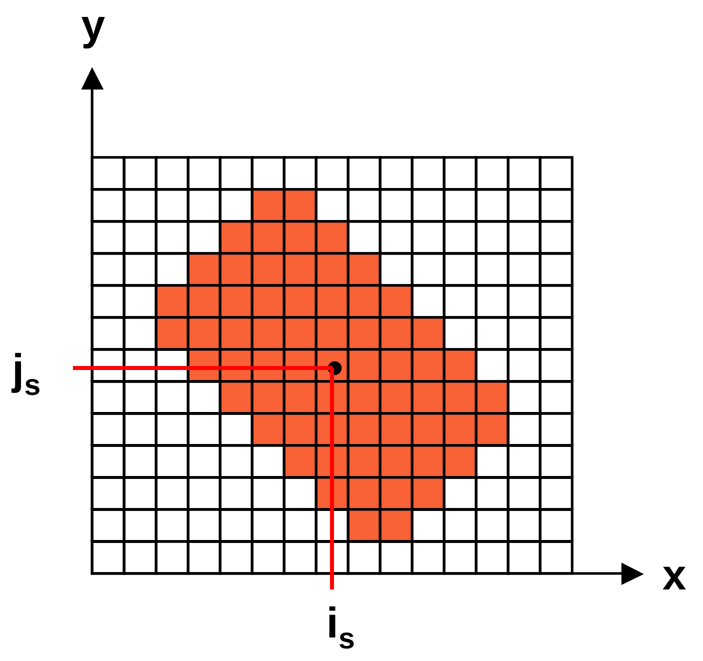
<u>Center-of-gravity coordinates</u>:

D = digital region

The center of gravity is the location where first-order moments sum to zero.

$$\sum_{ij \in D} (i - i_s) = 0 \qquad \sum_{ij \in D} (j - j_s) = 0$$

$$\Rightarrow \quad i_s = \frac{1}{|D|} \sum_{ij \in D} i \qquad j_s = \frac{1}{|D|} \sum_{ij \in D} j$$



Second-order moments

Second-order moments ("moments of inertia") measure the distribution of mass relative to axes through the center of gravity.

$$m_x = \sum_{ij\in D} (i - i_s)^2 = \sum_{ij\in D} i^2 - i_s^2 |D|$$

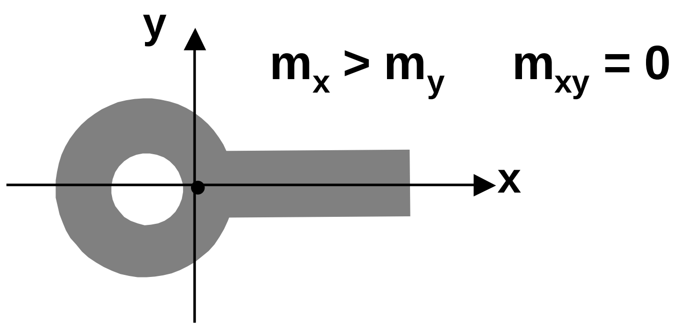
moment of inertia relative to y-axis through center of gravity

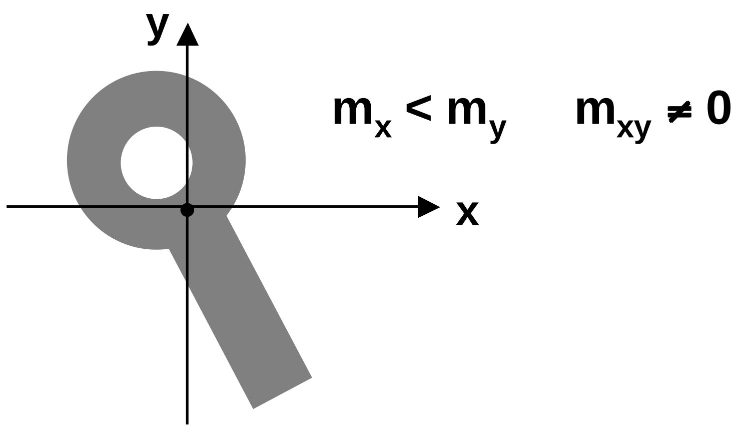
$$m_y = \sum_{ij\in D} (j - j_s)^2 = \sum_{ij\in D} j^2 - j_s^2 |D|$$

moment of inertia relative to x-axis through center of gravity

$$m_{xy} = \sum_{ij\in D} (i - i_s)(j - j_s) = \sum_{ij\in D} ij - i_s j_s |D|$$

"mixed"moment of inertia relative to x- and y-axis through center of gravity, zero if x- and y-axis are "main axes"



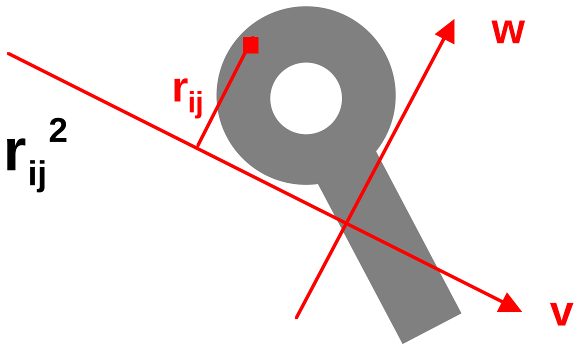


Axis of minimal inertia

The axis of minimal inertia can be used as an innate orientation of a 2D shape.

Inertia (= second order moment) relative to an axis is the sum of the squared distances between all pixels of the shape and the axis.

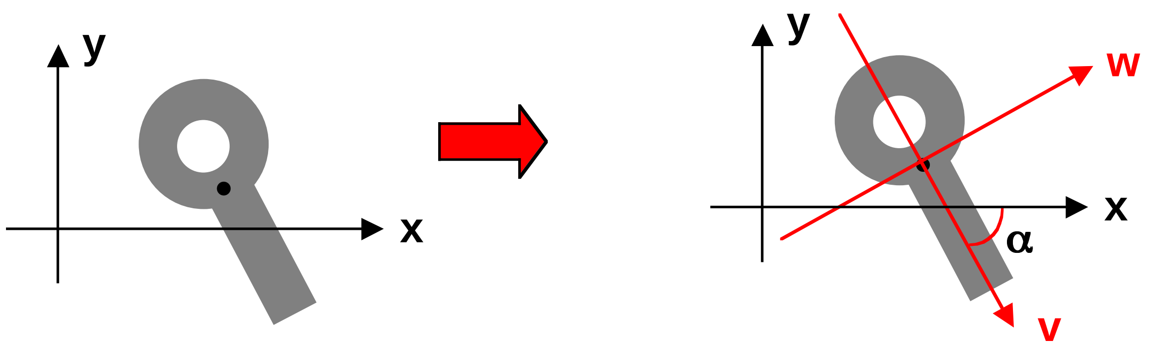
$$m_v = \sum_{ij \in D} r_{ij}^2$$



1. The axis of least inertia passes through the center of gravity
2. The mixed moment m_{vw} relative to the axes v and w must be zero

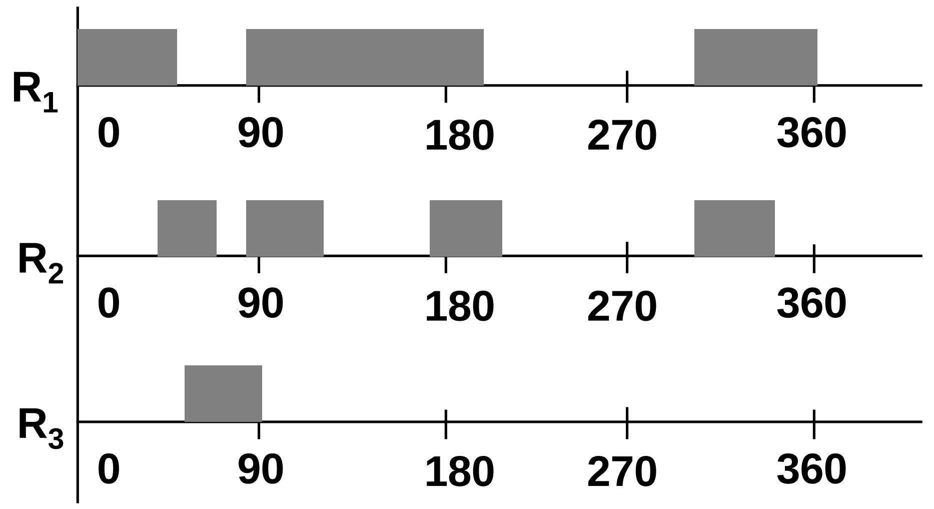
If the mixed moment is nonzero, the axis must be turned by the angle α:

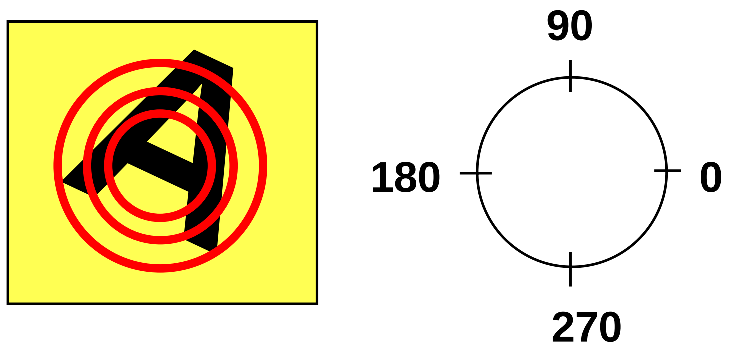
$$\tan 2\alpha = \frac{2m_{xy}}{m_y - m_x}$$



Polar signature

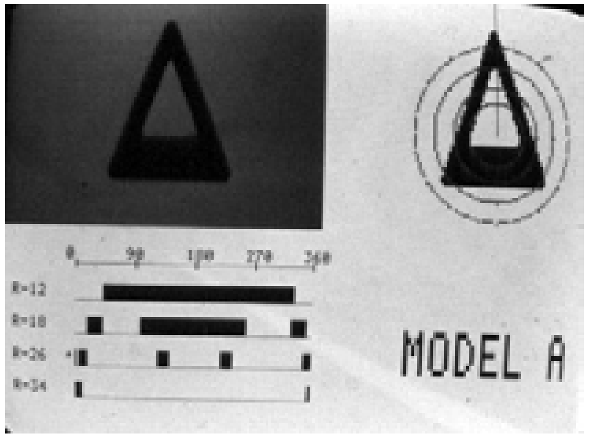
The polar signature records the angular segments where circles around the center of gravity lie within a shape.

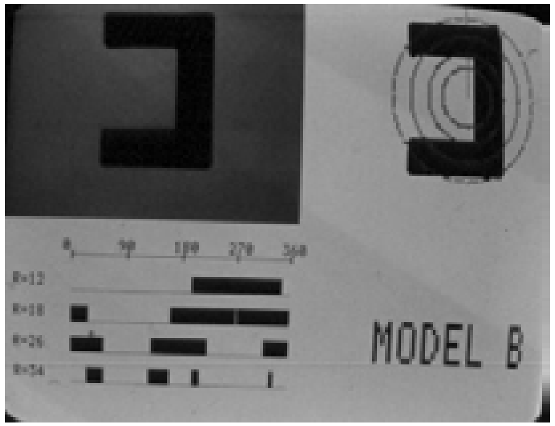


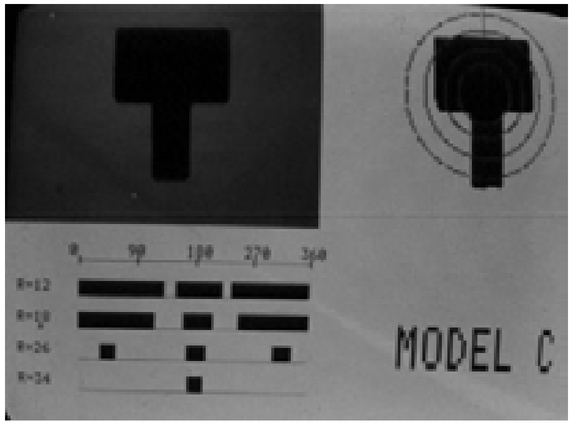


- **scalable from coarse to fine by appropriate number of circles**
- **radii of circles must be chosen judiciously**
- **translation-invariant**
- **rotation-invariance can be achieved by cyclic shifting**

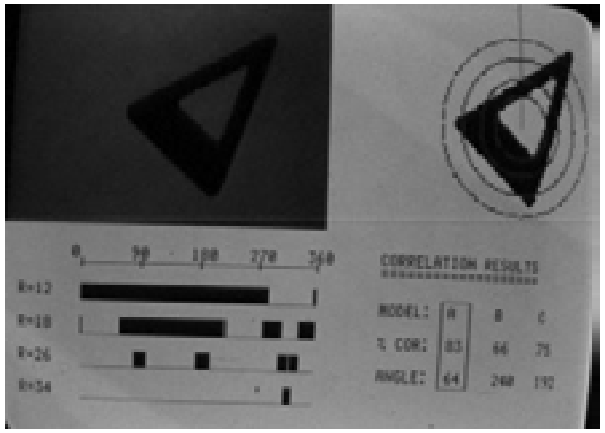
Object recognition using the polar signature





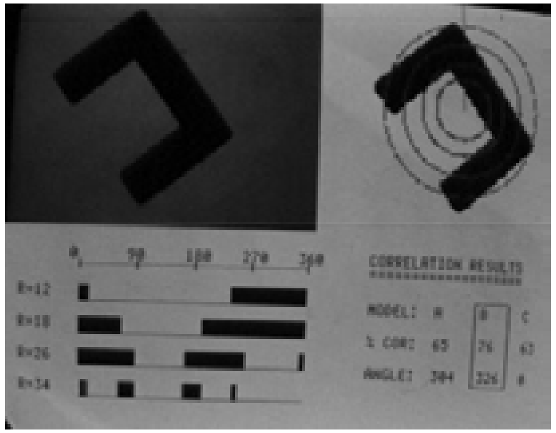


Model signatures



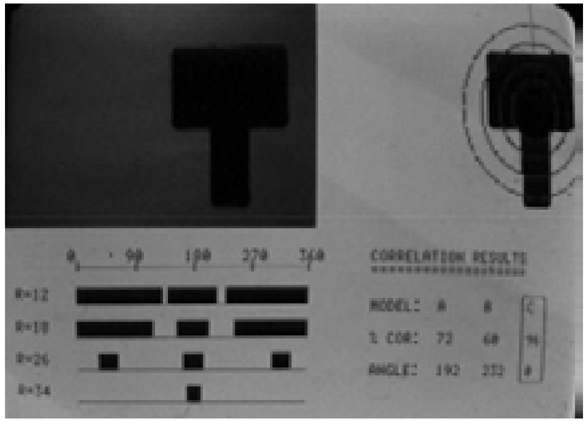
CORRELATION RESULTS

MODEL:	A	B	C
% COR:	83	66	75
ANGLE:	64	240	192



CORRELATION RESULTS

MODEL:	A	B	C
% COR:	65	76	61
ANGLE:	304	326	0



CORRELATION RESULTS

MODEL:	A	B	C
% COR:	73	60	96
ANGLE:	192	333	0

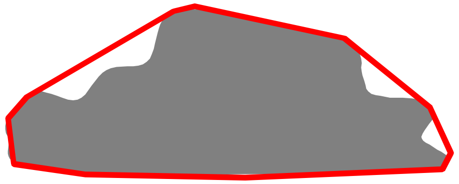
Recognition results

Convex hull

A region R is convex if the straight-line segment x_1x_2 between any two points of R lies completely inside of R.

For an arbitrary region R, the convex hull H is the smallest convex region which contains R.

Example of shape with convex hull:

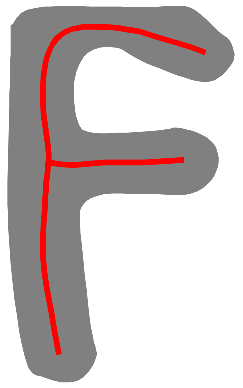


<u>Intuitive convex hull algorithm</u>:

1. **Pick lowest and left-most boundary point of R as starting point $P_k = P_1$. Set direction of previous line segment of convex-hull boundary to $\underline{v} = (0, -1)$.**
2. **Follow boundary of R from current point P_k in an anti-clockwise direction and compute angle θ_n of line P_kP_n for all boundary points P_n after P_k. The point P_q with $\theta_q = \min\{\theta_n\}$ is a vertex of the convex hull boundary.**
3. **Set $P_k = P_q$ and $\underline{v} = (P_kP_n)$ and repeat 2) and 3) until $P_k = P_1$.**

There are numerous convex hull algorithms in the literature. The most efficient is O(N) [Melkman 87], see Sonka et al. "Image Processing ...".

Skeletons



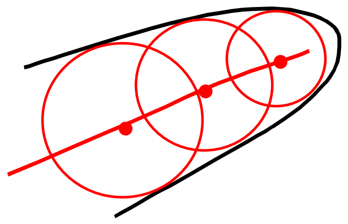
The skeleton of a region is a line structure which represents "the essence" of the shape of the region, i.e. follows elongated parts.

Useful e.g. for character recognition

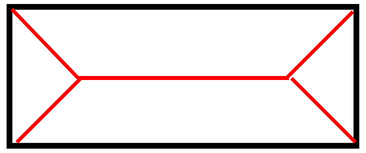
<u>Medial Axis Transform</u> (MAT) is one way to define a skeleton:

The MAT of a region R consists of all pixels of R which have more than one closest boundary point.

MAT skeleton consists of centers of circles which touch boundary at more than one point



MAT skeleton of a rectangle shows problems:



Note that "closest boundary point" depends on digital metric!

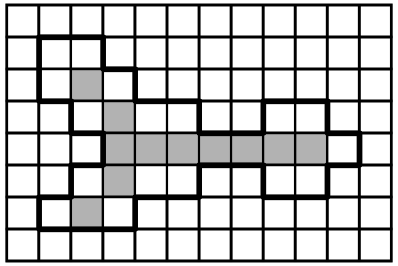
Thinning algorithm

Thinning algorithm by Zhang and Suen 1987
(from Gonzalez and Wintz: "Digital Image Processing")

Repeat A to D until no more changes:

A	**Flag all contour points which satisfy conditions (1) to (4)**
B	**Delete flagged points**
C	**Flag all contour points which satisfy conditions (5) to (8)**
D	**Delete flagged points**

Example:



Assumptions:
- **region pixels = 1**
- **background pixels = 0**
- **contour pixels 8-neighbours of background**

Neighbourhood labels:

p_9	p_2	p_3
p_8	p_1	p_4
p_7	p_6	p_5

Conditions:

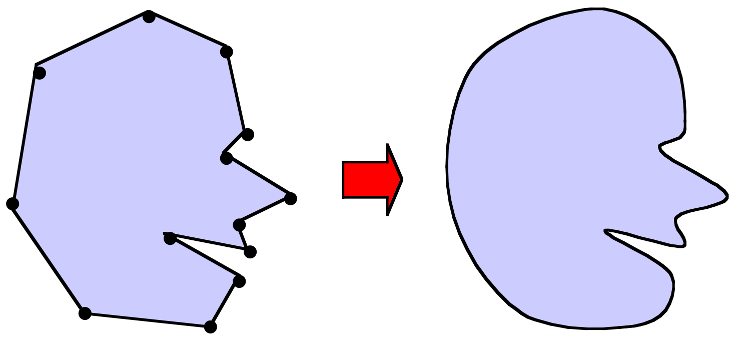
(1) $2 \leq N(p_1) \leq 6$	(5) $2 \leq N(p_1) \leq 6$
(2) $S(p_1) = 1$	(6) $S(p_1) = 1$
(3) $p_2 \cdot p_4 \cdot p_6 = 0$	(7) $p_2 \cdot p_4 \cdot p_8 = 0$
(4) $p_4 \cdot p_6 \cdot p_8 = 0$	(8) $p_2 \cdot p_6 \cdot p_8 = 0$

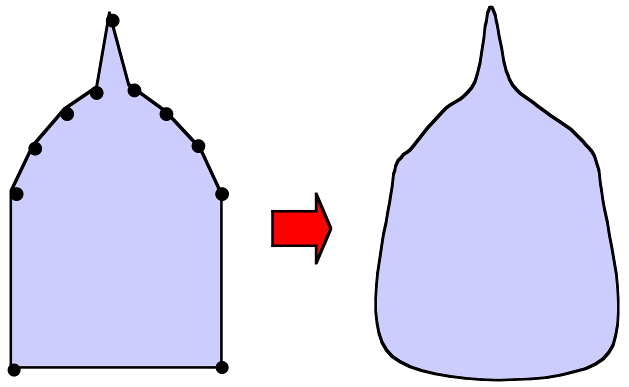
$N(p_1)$ = number of nonzero neighbours of p_1

$S(p_1)$ = number of 0 - 1 transitions in ordered sequence p_2, p_3, ...

B-splines (1)

B-splines are piecewise polynomial curves which provide an approximation of a polygon based on vertices.





precision depends on distances of vertices

Important properties:

- **eye-pleasing smooth approximation of control polygon**
- **change of control polygon vertex influences only small neighbourhood**
- **curve is twice differentiable (e.g. has well-defined curvature)**
- **easy to compute**

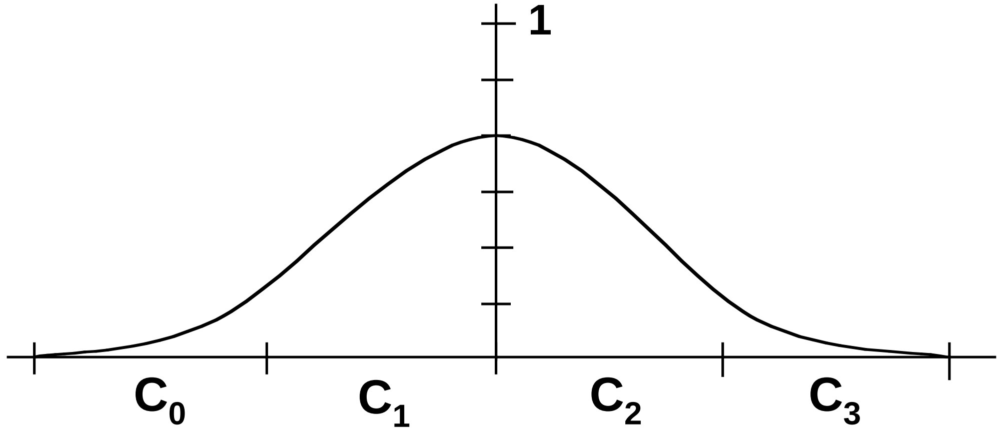
$\underline{x}(s) = \Sigma\ \underline{v}_i B_i(s) \quad i = 0\ ..\ N+1$

s	parameter, changing linearly from i to i+1 between vertices $\underline{v}_i$ and $\underline{v}_{i+1}$
$\underline{v}_i$	vertices of control polygon
$B_i(s)$	base functions, nonzero for s ‡$[i-2\ ,\ i+2]$

B-splines (2)

Each base function $B_i(s)$ consists of four parts:

$$\left.\begin{aligned}
C_0(t) &= \frac{t^3}{6} \\
C_1(t) &= \frac{-3t^3+3t^2+3t+1}{6} \\
C_2(t) &= \frac{3t^3-6t^2+4}{6} \\
C_3(t) &= \frac{-3t^3+3t^2-3t+1}{6}
\end{aligned}\right\}$$



$\underline{x}(s) = C_3(s-i)\underline{v}_{i-1} + C_2(s-i)\underline{v}_i + C_1(s-i)\underline{v}_{i+1} + C_0(s-i)\underline{v}_{i+2}$

<u>Example</u>: $s = 7.7 \quad i = 7$

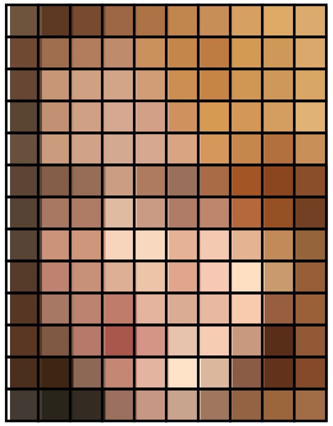
$\underline{x}(7.7) = C_3(0.7)\underline{v}_6 + C_2(0.7)\underline{v}_7 + C_1(0.7)\underline{v}_8 + C_0(0.7)\underline{v}_9$

Templates

A template is a translation-, rotation- and scale-variant shape desription. It may be used for object recognition in a fixed, reoccurring pose.

- A M-by-N template may be treated as a vector in MN-dimensional feature space
- Unknown objects may be compared with templates by their distance in feature space

Example:
Template for face recognition



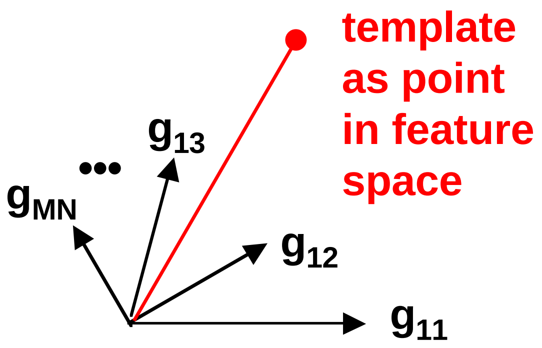
Distance measures:

g_{mn}	pixels of image
t_{mn}	pixels of template

$$d_e^2 = \sum_{mn} (g_{mn} - t_{mn})^2$$ squared Euclidean distance

$$d_a = \sum_{mn} |g_{mn} - t_{mn}|$$ absolute distance

$$d_b = \max_{mn} |g_{mn} - t_{mn}|$$ maximal absolute distance



Cross-correlation

$r = \sum_{mn} g_{mn} t_{mn}$ cross-correlation between image g_{mn} and template t_{mn}

Compare with squared Euclidean distance d_e^2:

$$d_e^2 = \sum_{mn} (g_{mn} - t_{mn})^2 = \sum_{mn} g_{mn}^2 + \sum_{mn} t_{mn}^2 - 2r$$

Image "energy" $\Sigma g_{mn}{}^2$ and template "energy" $\Sigma t_{mn}{}^2$ correspond to length of feature vectors.

$$r' = \frac{\sum_{mn} g_{mn} t_{mn}}{\sqrt{\sum_{mn} g_{mn}^2 \sum_{mn} t_{mn}^2}}$$

Normalized cross-correlation is independent of image and template energy. It measures the cosine of the angle between the feature vectors in MN-space.

<u>Cauchy-Schwartz Inequality</u>:

$|r'| \leq 1$ with equality iff $g_{mn} = c\, t_{mn}$, all mn

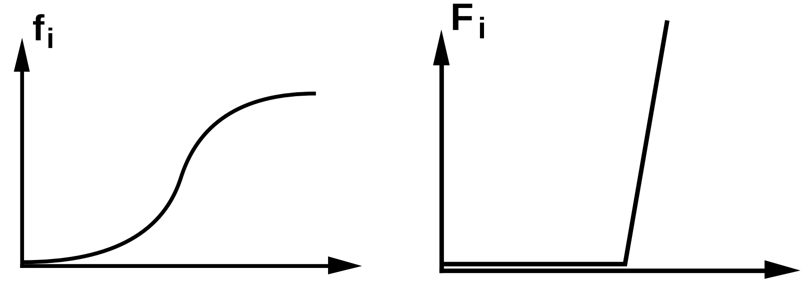
Artificial neural nets

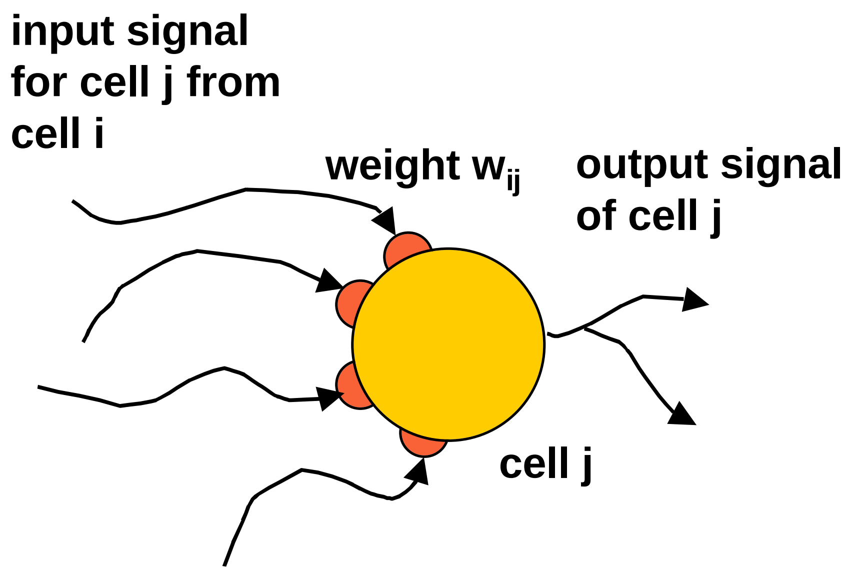
Information processing in biological systems is based on neurons with roughly the following properties:

- **the degree of activation is determined by incoming signals**
- **the outgoing signal is a function of the activation**
- **incoming signals are mediated by weights**
- **weights may be modified by learning**

net input for cell j	$\Sigma w_{ij}\ o_i(t)$
activation	$a_j(t) = f_j\ (a_j, \Sigma w_{ij}\ o_i(t))$
output signal	$o_j(t) = F_j\ (a_j)$

Typical shapes of f_i and F_i:



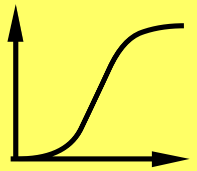


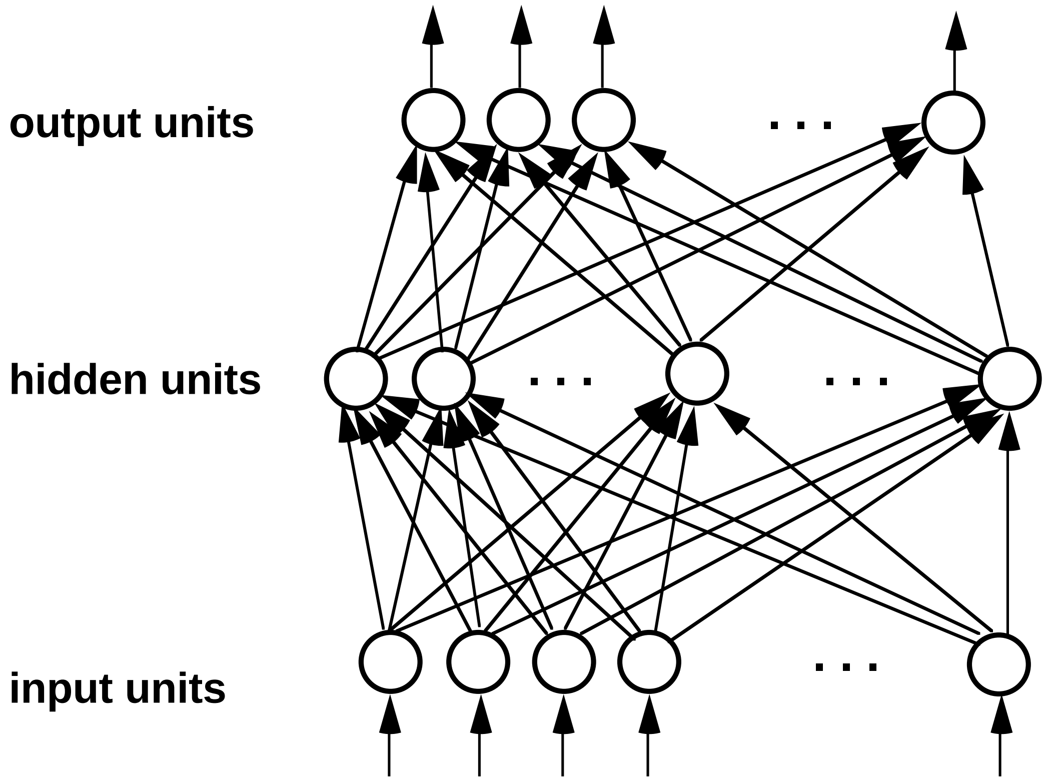
Multilayer feed-forward nets

<u>**Example**</u>**:**

3-layer net

- **each unit of a layer is connected to each unit of the layer below**
- **units within a layer are not connected**
- **activation function f is differentiable (for learning)**





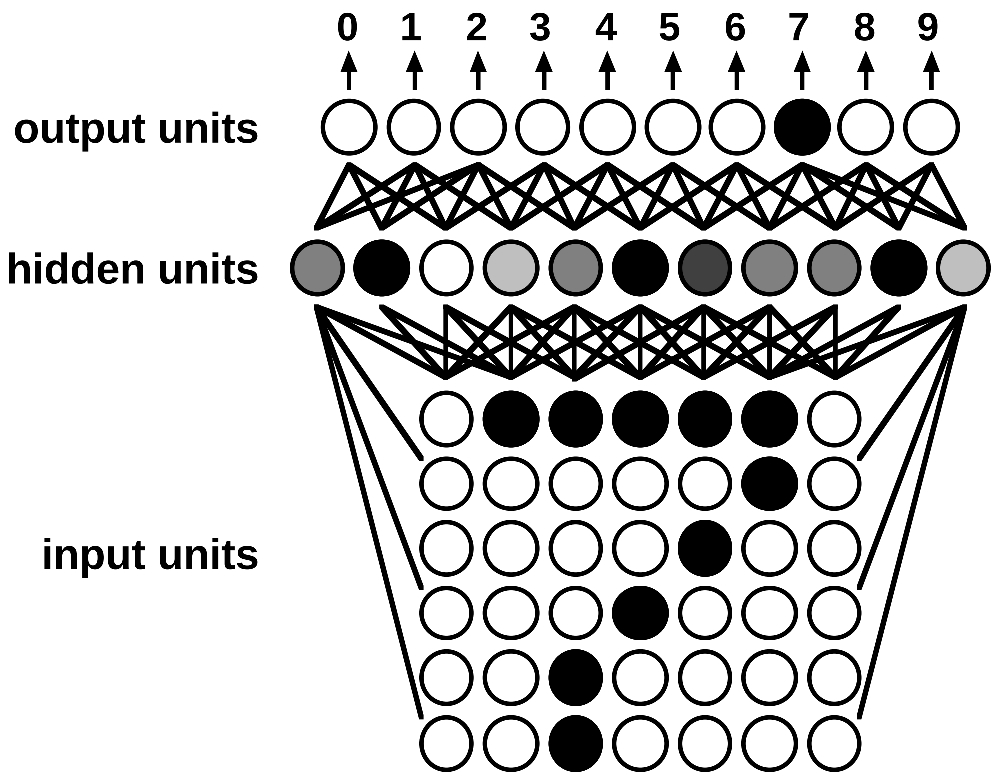
Character recognition with a neural net

Schematic drawing shows 3-layer feed-forward net:

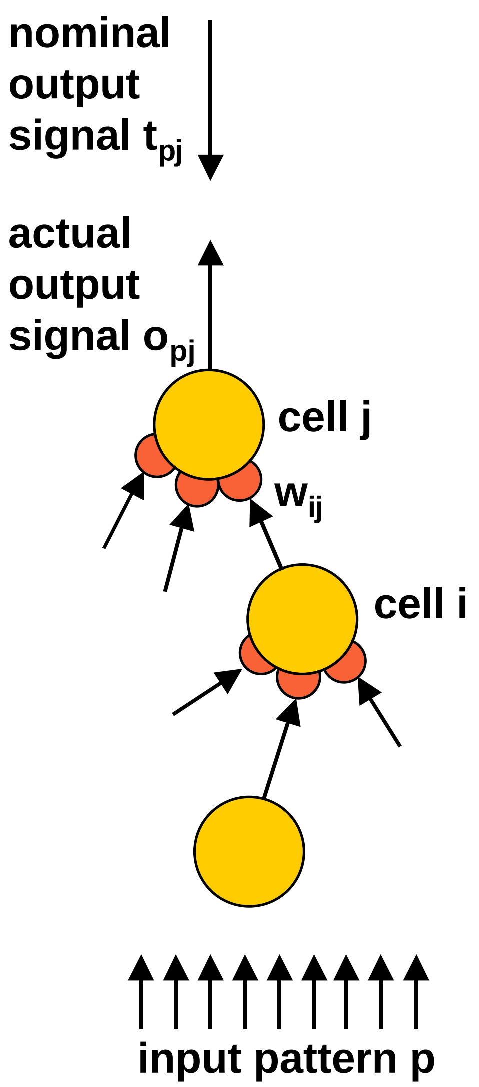
- **input units are activated by sensors and feed hidden units**
- **hidden units feed output units**
- **each unit receives weighted sum of incoming signals**

<u>Supervised learning</u>

Weights are adjusted iteratively until prototypes are classified correctly (-> backpropagation)



Learning by backpropagation



Supervised learning procedure:

- **present example and determine output error signals**
- **adjust weights which contribute to errors**

Adjusting weights:

- **Error signal of output cell j for pattern p is**

  $\delta_{pj} = (t_{pj} - o_{pj})\, f_j'(net_{pj})$

  $f_j'()$ is the derivative of the activation function f()

- **Determine error signal δ_{pi} for internal cell i recursively from error signals of all cells k to which cell i contributes.**

  $\delta_{pi} = f_i'(net_{pi})\, \Sigma_k\, \delta_{pk} w_{ik}$

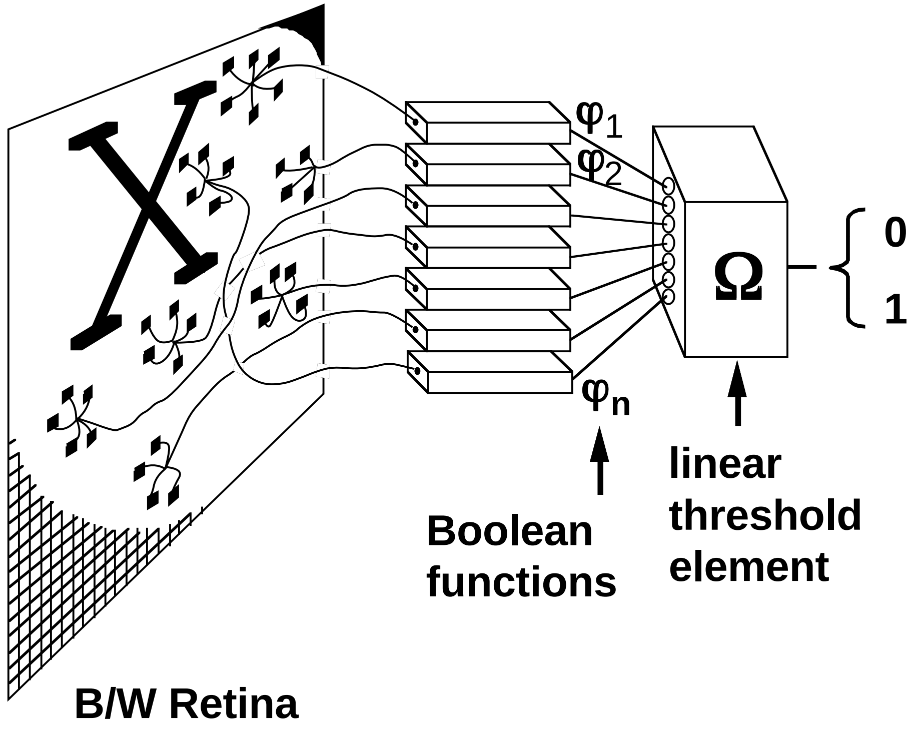
- **Modify all weights:** $\Delta_p w_{ij} = \eta \delta_{pj} o_{pi}$ **η is a positive constant**

The procedure must be repeated many times until the weights are "optimally" adjusted. There is no general convergence guarantee.

Perceptrons (1)

Which shape properties can be determined by combining the outputs of local operators?

A perceptron is a simple computational model for combining local Boolean operations. (Minsky and Papert, Perceptrons, ??)



φ_i Boolean functions with local support in the retina:
- limited diameter
- limited number of cells
output is 0 or 1

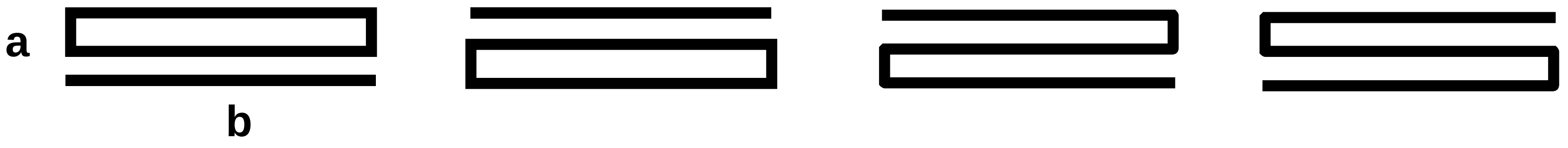
Ω compares weighted sum of the φ_i with fixed threshold θ:

$$\Omega = \begin{cases} 1 & \text{if } \Sigma w_i \varphi_i > \theta \\ 0 & \text{otherwise} \end{cases}$$

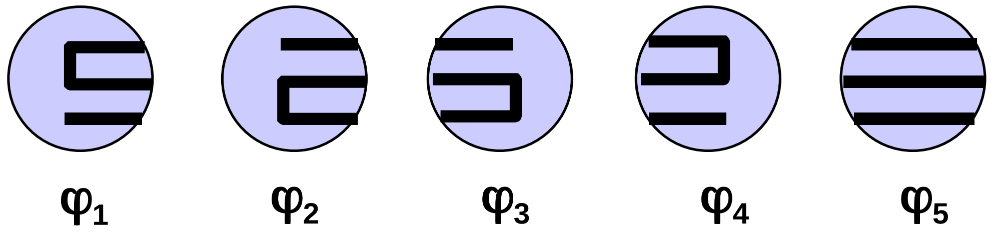
Perceptrons (2)

A limited-diameter perceptron cannot determine connectedness.

Assume perceptron with maximal diameter d for the support of each φ_i.
Consider 4 shapes as below with a < d and b >> d.



Boolean operators may distinguish 5 local situations:



φ_5 is clearly irrelevant for distinguishing between the 2 connected and the 2 disconnected shapes

For Ω to exist, we must have:

$w_1 \varphi_1 + w_4 \varphi_4 < \theta$
$w_2 \varphi_2 + w_3 \varphi_3 < \theta$
$\Rightarrow \Sigma w_i \varphi_i < 2\theta$

$w_2 \varphi_2 + w_4 \varphi_4 > \theta$
$w_1 \varphi_1 + w_3 \varphi_3 > \theta$
$\Rightarrow \Sigma w_i \varphi_i > 2\theta$

$\Rightarrow$ contradiction, hence Ω cannot exist