Object Recognition General Overview

Several approaches to classification/recognition. Choose the same class as objects with:

- **Properties** - similar properties
- **Appearance** - similar pixel values
- **Geometric** - similar structures in similar places with similar parameters
- **Graph** - similar part relationships
- **Bag of Words** - enough similar descriptions

Object recognition key points

- Classification by comparing the relative probability of a shape belonging to different classes.
- Use Bayes rule to calculate the class probabilities.
- Class model is multivariate Gaussian distribution.
- Estimate the distribution parameters from the data.

The story so far...

Preprocessing:

1. Capture image
2. Threshold to isolate object
3. Locate binary region
4. Measure properties
   \[
   \vec{x} = (\text{compactness}, c_{i1}, c_{i2}, c_{i3}, c_{i4}, c_{i5}, c_{i6})'
   \]

Probabilistic Object Recognition

\( p(c|d) \) is the probability that \( c \) was the class given that we observed evidence \( d \)

We select most probable class \( c \) (i.e. \( p(c|d) \) is the highest) or perhaps none if the probability for all classes is too low.
Computing $\text{prob}(c|d)$? Bayes Classifier

$p(c)$ is the \textit{a priori} (before any observations) probability of observing class $c$

$p(d|c)$ is the probability that evidence $d$ would have been observed if $c$ was the class

Bayes rule:

$$p(c|d) = \frac{p(d|c)p(c)}{p(d)} = \frac{p(d|c)p(c)}{\sum_k p(d|k)p(k)}$$

Advantage: we learn $p(c)$ and $p(d|c)$ from examples

$\text{prob}(d|c)$? Gaussian Distribution

Data $d$ is feature vector $\vec{x} = (f_1, f_2, \ldots, f_n)'$.

Expect some variation in property values, perhaps not independent between variables.

Commonly joint probability distribution of $d$ is

Multivariate Normal/Gaussian Distribution

For 2 properties, $\vec{x} = (f_1, f_2)'$ we have:

2D Gaussian Distribution

Characterised by mean $(m_1, m_2)'$ and covariance matrix

$$\begin{bmatrix}
(\sigma_1)^2 & \rho_{ij}\sigma_1\sigma_2 \\
\rho_{ij}\sigma_1\sigma_2 & (\sigma_2)^2
\end{bmatrix}$$

$s_i$ - standard deviation of $i^{th}$ property

$\rho_{ij}$ - cross correlation coefficient between $i$ and $j$

Multivariate Normal/Gaussian Distribution

For each class $c$ need:

- Mean vector $\vec{m}_c$ of dimension $n$ - the average value of the $n$ properties for class $c$

- Covariance matrix $\mathcal{A}_c$ - the $n \times n$ matrix of joint variation between each pair of components of the vector.

Then, the probability of observing feature vector $\vec{x}$ is:

$$p(\vec{x}|c) = \frac{1}{(2\pi)^{\frac{n}{2}} \det(\mathcal{A}_c)^{\frac{1}{2}}} e^{-\frac{1}{2}((\vec{x}-\vec{m}_c)'\mathcal{A}_c^{-1}(\vec{x}-\vec{m}_c))}$$
**Estimating the Distribution Parameters - the Class Model**

Given $k > n$ known instances of class $c$ with properties $\{\vec{x}_i\}$.

Estimated Mean vector: $\vec{m}_c = \frac{1}{k} \sum_i \vec{x}_i$

Estimated Covariance matrix:
\[
A_c = \frac{1}{k-1} \sum_i (\vec{x}_i - \vec{m}_c)(\vec{x}_i - \vec{m}_c)'$

Estimate $p(c)$ from the distribution of known samples

**Example continued**

Class 1 if $p(1|\vec{x}) > p(2|\vec{x})$

Bayes rule:
\[
p(1|\vec{x}) = \frac{p(\vec{x}|1)p(1)}{p(\vec{x})} > \frac{p(\vec{x}|2)p(2)}{p(\vec{x})} = p(2|\vec{x})
\]
\[
0.6 \times p(\vec{x}|1) > 0.4 \times p(\vec{x}|2)
\]
\[
p(\vec{x}|1) = \frac{1}{2\pi det(A_1)^{1/2}} e^{-\frac{1}{2}[(\vec{x} - \vec{m}_1)' A_1^{-1}(\vec{x} - \vec{m}_1)]}
\]

**Probability Example**

$n = 2$ classes. *a priori* probabilities $p(1) = 0.6$, $p(2) = 0.4$

Cls 1: $\vec{m}_1 = \begin{bmatrix} 2 \\ 6 \end{bmatrix}$, $A_1 = \begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix}$, $A_1^{-1} = \frac{1}{5} \begin{bmatrix} 3 & -1 \\ -1 & 2 \end{bmatrix}$

Cls 2: $\vec{m}_2 = \begin{bmatrix} 4 \\ 13 \end{bmatrix}$, $A_2 = \begin{bmatrix} 3 & 1 \\ 1 & 2 \end{bmatrix}$, $A_2^{-1} = \frac{1}{5} \begin{bmatrix} 2 & -1 \\ -1 & 3 \end{bmatrix}$

$det(A_1) = det(A_2) = 5$

Data $\vec{x} = \begin{bmatrix} area \\ perimeter \end{bmatrix} = \begin{bmatrix} 3 \\ 10 \end{bmatrix}$

Which is the most probable class?

\[
p(\vec{x}|1) = \frac{1}{2\pi \sqrt{5}} e^{-\frac{1}{2}[(\vec{x} - \vec{m}_1)' A_1^{-1}(\vec{x} - \vec{m}_1)]} = \frac{1}{2\pi \sqrt{5}} e^{-\frac{1}{2}[(\vec{x} - \vec{m}_1)' \frac{1}{5} \begin{bmatrix} 3 & -1 \\ -1 & 3 \end{bmatrix} (\vec{x} - \vec{m}_1)]}
\]

\[
p(\vec{x}|1) = \frac{1}{2\pi \sqrt{5}} e^{-\frac{1}{2} \begin{bmatrix} 1 \\ 4 \end{bmatrix}' \frac{1}{5} \begin{bmatrix} 3 & -1 \\ -1 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ 4 \end{bmatrix}} = \frac{1}{2\pi \sqrt{5}} e^{-\frac{1}{2} \begin{bmatrix} 1 \\ 4 \end{bmatrix}' \begin{bmatrix} 3 & -1 \\ -1 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ 4 \end{bmatrix}}
\]

\[
p(\vec{x}|1) = \frac{1}{2\pi \sqrt{5}} e^{-\begin{bmatrix} 1 \\ 4 \end{bmatrix}' A_1^{-1} \begin{bmatrix} 1 \\ 4 \end{bmatrix}} = \frac{1}{2\pi \sqrt{5}} e^{-\begin{bmatrix} 1 \\ 4 \end{bmatrix}' \frac{1}{5} \begin{bmatrix} 3 & -1 \\ -1 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ 4 \end{bmatrix}} = 4.78 \times 10^{-3}
\]
Similarly,

\[ p(\vec{x}|2) = \frac{1}{2\pi} \frac{1}{\sqrt{5}} e^{-\frac{23}{10}} = 7.15 \times 10^{-3} \]

So, class 1 if

\[ 0.6 \times p(\vec{x}|1) > 0.4 \times p(\vec{x}|2) \]
\[ 2.87 \times 10^{-3} > 2.85 \times 10^{-3} \]

Thus most likely (barely) to be class 1.

**Midlecture Problem**

The data has a probability of 0.1 for class 1 and a probability of 0.2 for class 2. But, class 1 is 3 times as likely to be observed \textit{a priori} as class 2.

Based on the two pieces of information, which is the most likely class to explain the observations?

**Matlab code for probability calculation**

```matlab
function prob = multivariate(Vec,Mean,...
    Invcor,apriori)

diff = Vec-Mean;
n = length(Vec);
wgt = sqrt(det(Invcor));
dist = diff*Invcor*diff';
prob = apriori * ( 1 / (2*pi)^(n/2)) ...
    * wgt * exp(-0.5*dist);
```

**Training**

Calculating classifier parameters \( p(c), \vec{m}_c, \mathcal{A}_c \)

Should split data into:

- Training set - used to estimate parameters (eg. 50% of data)
- Validation set - used to tell when to stop training (eg. 25% of data)
- Test set - used to test performance (eg. 25% of data)

Must have more samples than properties!
Training Example

For the batch-mode training used here, can merge Training and Validation sets.

In the example below, since we have only 4 samples of each class, we’ll be naughty and use all 4 samples for both the Training and Test sets.

Can use at most 3 properties.

Test Objects

A: B: C:

4 instances of each

\textit{a priori} probability = 0.33

Training Code

\begin{verbatim}
Dim = 3; % number of feature properties
modelfilename = input('Model file name (filename)\n?','s');
maxclasses = input('Number of classes (int)\n?');
trainfilestem = input('Training image file stem (filestem)\n?','s');
N = input('Number of training images (int)\n?');
for imagenum = 1 : N
    currentimagergb = imread([trainfilestem, int2str(imagenum),'.jpg'],'.jpg');
    currentimage = rgb2gray(currentimagergb);
vec(imagenum,:) =
    extractprops(currentimage,0,0,0,0,0);
trueclasses(imagenum) = input(['Train image ', int2str(imagenum),' true class (1..',int2str(maxclasses),')\n?']);
end
[Means,Invcors,Aprioris] = buildmodel(Dim,vec,N,
    maxclasses,trueclasses);
eval(['save ','modelfilename',' maxclasses ...
    Means Invcors Aprioris']);
\end{verbatim}
Building statistical model

function [Means, Invcors, Aprioris] = buildmodel(Dim, Vecs, N, Numclass, Classes)
    for i = 1 : Numclass
        samples = find(Classes == i); % locate cls i
        M = length(samples); % num in class
        classvecs = Vecs(samples,:); % get members
        mn = mean(classvecs);
        Means(i,:) = mn;
        diffs = classvecs - ones(M,1)*mn;
        Invcors(i,:,:) = inv(diffs'*diffs/(M-1));
        Aprioris(i) = M/N;
    end

Training Log

doall(2)
Model file name (filename)?blocks
Number of classes (int)?3
Training image file stem (filestem)?TESTDATA1/f
Number of training images (int)?12
Train image 1 true class (1..3)?1
***

test Code

eval(['load ',modelfilename,...
     ' maxclasses Means Invcors Aprioris'])
imagestem = input('Test image file stem ...
     (filestem)\n?','s');
run=1;
imagenum=0;
while ~(run == 0)
    imagenum = imagenum + 1;
    currentimagergb = imread([imagestem, ...
            int2str(imagenum),'.jpg']);
    currentimage = rgb2gray(currentimagergb);
    vec = extractprops(currentimage,0,0,0,0);
    class=classify(vec,maxclasses,Means,Invcors,...
        Dim,Aprioris)
    run = input(['Want to process another image ',
        int2str(imagenum+1),'] (0,1)\n?']);
end
Recognition Performance

Confusion matrix:

<table>
<thead>
<tr>
<th>Computed Class</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
</tbody>
</table>

Discussion

- Simple process but well founded
- Works well because reliable feature extraction and features easily discriminate
- Choosing good features usually hardest part.

What We Have Learned

1. Bayes classifier
2. Multi-variate Gaussian distributions
3. Property-based recognition

Projective Geometry I

**Projection**: Any non-singular (ie. invertable) linear transformation \( P \) that maps points from one position to another

Plane \( \pi_2 \) shapes observed in a 3D position project onto image plane \( \pi_1 \) (2D \( \rightarrow \) 2D)

\[ P : \pi_2 \rightarrow \pi_1 \]
Projective Geometry II

3D→3D, 1D→1D also possible
Accounts for rotation, translation, scale, shear.
To do properly, use homogeneous coordinates
Augment point positions \((x, y)’\) to \((x, y, 1)’\)

\[
\begin{pmatrix} x \\ y \end{pmatrix} \rightarrow \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}
\]

Use homogeneous coordinates
\(P_1: \pi_2 \rightarrow \pi_1\) (ie. copy scene plane into image plane)
\(P_2: \pi_2 \rightarrow \pi_3\) (ie. copy scene plane into new plane)
Therefore \(P_3 = P_2 (P_1)^{-1}: \pi_1 \rightarrow \pi_3\)

Projection matrix \(P_i:\)

\[
P = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix}
\]
Remapping Algorithm

Input image \( I(x, y) \)
Remapped image \( R(u, v) \)

If projective relation \( P \) between planes known, then can map \((u, v)\) onto \((x, y)\) using:

\[
\begin{bmatrix}
\lambda x \\
\lambda y \\
\lambda 
\end{bmatrix}
= P
\begin{bmatrix}
u \\
v \\
1 
\end{bmatrix}
\]

for each \((u, v)\)
get \((x, y)\) from projection divided by \(\lambda\)

\( R(u, v) = I(x, y) \)

See remap.m

Estimating \( P \): \((u, v) \rightarrow (x, y)\)

Direct Linear Transform Method

\( N \geq 4 \) matched points: \( \{(x_i, y_i), (u_i, v_i)\}^{N}_{i=1} \)

Let \( \vec{p}' = (p_{11}, p_{12}, p_{13}, p_{21}, p_{22}, p_{23}, p_{31}, p_{32}, p_{33})' \)

Let \( A_i = 
\begin{bmatrix}
0 & 0 & 0 & -u_i & -v_i & -1 & y_i & u_i & y_i & v_i & y_i \\
u_i & v_i & 1 & 0 & 0 & 0 & -x_i & u_i & -x_i & v_i & -x_i 
\end{bmatrix}
\)

Construct \( A = 
\begin{bmatrix}
A_1 \\
A_2 \\
\ldots \\
A_N 
\end{bmatrix}
\)

Compute \( \text{SVD}(A) = UDV' \)

\( \vec{p} \) is last column of \( V \) (eigenvector of smallest eigenvalue of \( A \))

Repack \( \vec{p} \) back into matrix \( P \)

See esthomog.m
Projective Transfer Example

ORIGINAL    TARGET    REMAPPED