Topic 12: Pattern Recognition

Aim: This topics convers the basic of Object, or Pattern Recognition from template matching, the feature vector, basics of classifiers, and some examples of features including shape and moments.

Contents:

1. Basic Principles of Pattern Recognition
2. Template matching and Target Tracking
3. Feature Vectors
4. Simple Classifiers (Box and Linear)
5. Minimum Distance Classifier
6. Shape Features
7. Moments as Features.
8. Moments as invariant features

Principles of PR

Statistical Pattern Recognition:

Aim is to label a set of known objects, for example “letter” (character recognition).

1. Extract object from background (image processing)
2. Extract “features” (numbers) from the object.
3. Use these to assign the most probable “label” (assign a classification).

Syntactic Pattern Recognition:

Aim to describe an object in terms of “primitives” and then assign a label.

1. Extract object from background.
2. Extract primitives (typically lines or circles).
3. Use relationship between primitives for form a description of the object in terms of compound primitives (four lines can form a parallelogram, three parallelograms an form a cube etc).
4. Use description of object to assign name.

Statistical Pattern Recognition Types

Supervised Learning

We have examples of each predefined class, (a “training set”).

1. Extract “features” from each instance of the training set (assume they are correctly classified).
2. Use these to “train” the classifies (set the parameters to produce the lowest error rate).
3. To classify an unknown object, extract “features” and use classifies to assign the most probable label.

Once classifies is “trained” (parameters are fixed).

Unsupervised Learning

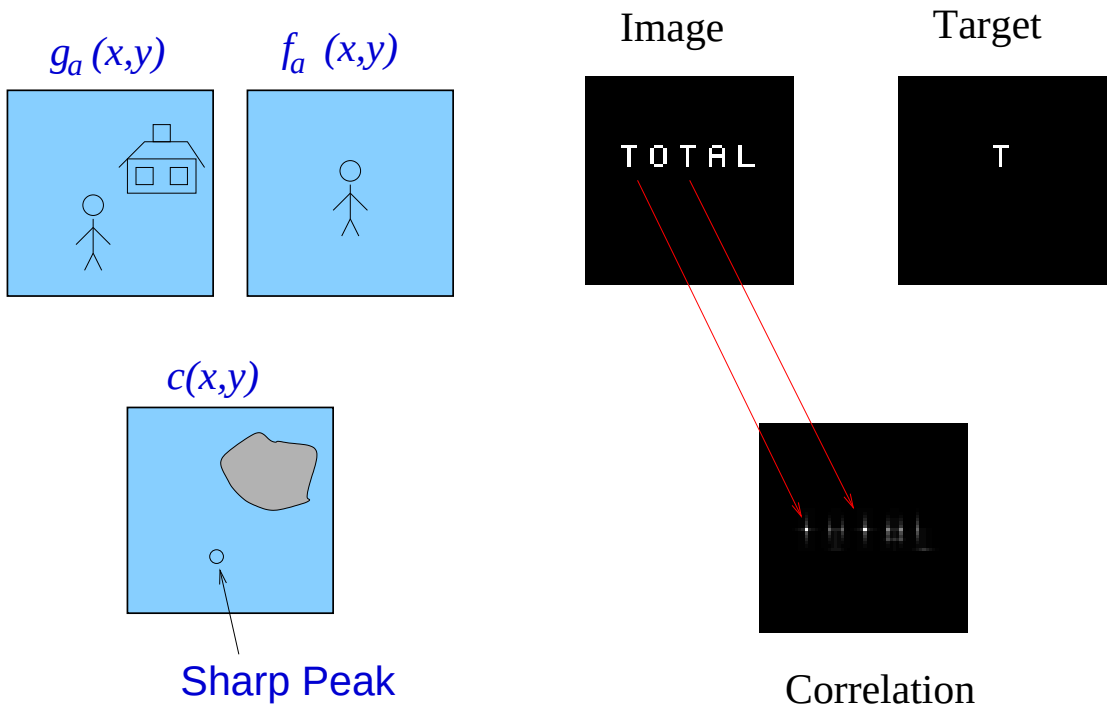
We have examples of classes, but without pre-defined labels.

1. Extract “features” from each instance.
2. “Train” the classifier to break instances into distinct groups (clusters). Number of distinct classes usually known.
3. Manually assign label to the clusters.
4. Use classifier on new (unknown) object as for the Supervised case

Both scheme designed to *assign a pre-defined* name.

Template Matching

Is a specified target in a scene. Test by matching the target at each location,



Template Matching

Mathematically we form the correlation

$$c(x,y) = \iint f(s,t)\, g(s-x, t-y)\, \mathrm{d}s\mathrm{d}t$$

which will give a sharp peak where the target matches the scene.

Note Fourier relation for correlation, where

$$C(u,v) = F(u,v)\, G^*(u,v)$$

which simplifies the implementation.

The correlation gives us:

1. Height of correlation: probability of the target at a location. Find maxima for most probable target location.
2. Location of Maxima: Location on target

so one of the few scheme that also extracts the object and locates it in the test scene.

Template Matching II

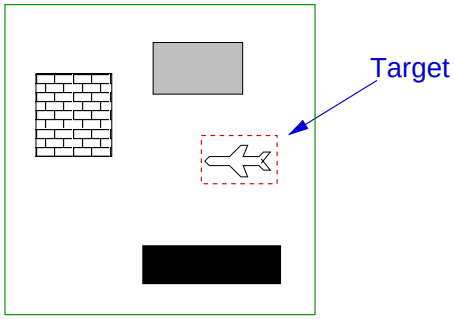
Practicality:

1. Easy to implement, give location of target without pre-processing.
2. Works for multiple targets in one scene, multiple peaks.
3. Requires close match to get good correlation (2% scale or 5° rotation typically gives 50% drop in correlation).
4. Difficult to extend to multiple target types (range of schemes but few reliable)

Fast, easy system to locate single “known” object, basis of target tracking.

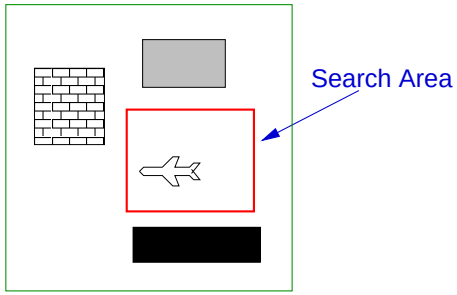
Target Tracking

Aim to track known (or pre-selected) object as it moves in scene.



1. Extract object from scene (usually manually selected), this becomes target.
2. At next time interval, form correlation over search area (know expected speed, we can specify maximum search area).
3. Relocate target

Target Tracking I



If object changes shape (rotation, scale change etc), then correlation peak will drop, but if changes small enough correlation still large.

1. Re-extract new object from the scene.
2. This become new target.
3. Use new target to track object in new search area.

Once direction of movement established we are able to trim search area to make updates faster so less that object changes shape too quickly.

Feature Vectors

To characterise an object we extract a series of measures, For image $f(x,y)$ to extract K measures,

$$v_1, v_2, \ldots, v_K$$

which we can treat as a vector of dimension K, by taking

$$\mathbf{v} = (v_1, v_2, \ldots, v_K)$$

So we can now thing of the object are being represented by a point in a K dimensional space with it coordinates given by **v**.

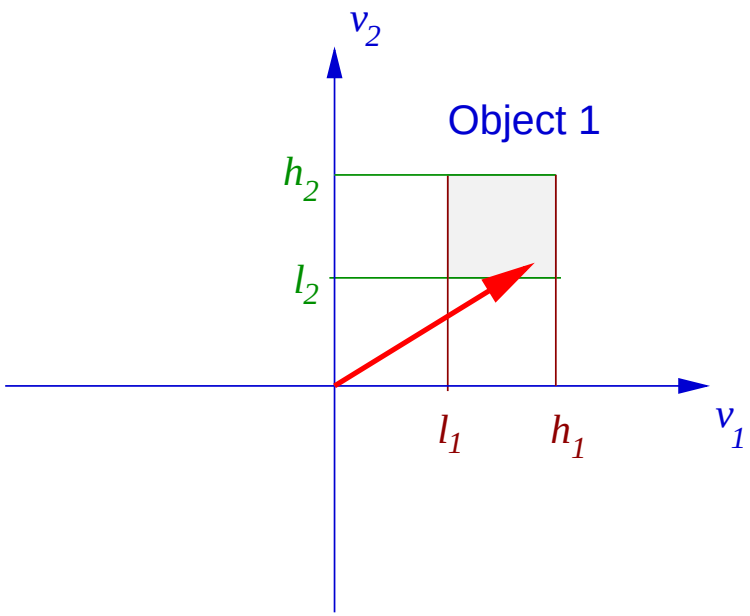
The aim is low to divide the K dimensional space up into regions associated with each known object type. Assign the label to each new object from which region is *feature vector* falls into.

Aside: For multi-spectral data with K bands each pixel is a vector of length K , we can classify each pixel by dividing up a K dimensional space.

Simple Classifiers

Box Classifier

Specify lower and upper bounds for each component of feature vector. For 2-D



$$\begin{aligned} \text{2-D} &\Rightarrow \text{Rectangle} \\ \text{3-D} &\Rightarrow \text{Rectangular cube} \\ > 3-\text{D} &\Rightarrow \text{Hyper-cube} \end{aligned}$$

Set low/upper bounds of each component from “training set” of pre-classified objects. Then use these bounds to classify object.

Linear Classifier

Divides the feature space up into regions using planes of dimension $K-1$.

Define a linear discrimination function,

$$g(\mathbf{v}) = w_0 + \mathbf{w}.\mathbf{v}$$

where vector **w** is a series of K weights,

$$\mathbf{w} = (w_1, w_2, \ldots, w_K)$$

that specify the importance of each component in the feature vector. We can now form a linear discrimination function for each label (or class) of object we wish to classify.

So for M classes of object we form

$$g_i(\mathbf{v}) \quad \text{for } i = 1, 2, \ldots, M$$

we then classify the object of being of class j if we have

$$g_j(\mathbf{v}) > g_i(\mathbf{v}) \quad \forall i \neq j$$

To form this classifier we need to find $K+1$ components of

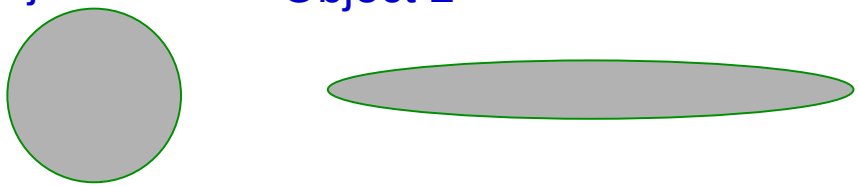
$$\mathbf{w} = (w_1, w_2, \ldots, w_K) \quad \text{and} \quad w_0$$

again typically from the training set such that there is least errors in the classification.

Linear Classifier Example

Take 2 types of object,

Object 1 Object 2

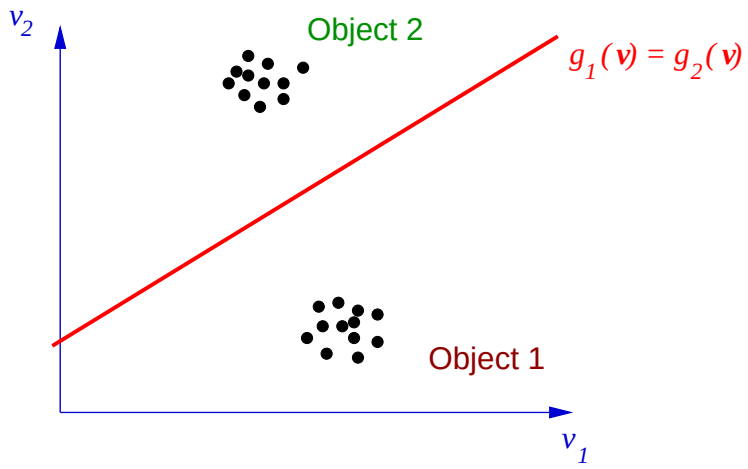


and extra the features,

$v_1 =$ Area and $v_2 =$ Boundary Length

Linear Classifier Example

So for range of instances we get



So the two discrimination functions $g_1()$ and $g_2()$ divide the region by a line.

For a new input object, the classification rule then becomes

$$g_1(\mathbf{v}) > g_2(\mathbf{v}) \quad \Rightarrow \quad \text{Object 1}$$
$$g_2(\mathbf{v}) > g_1(\mathbf{v}) \quad \Rightarrow \quad \text{Object 2}$$

so once trained, it becomes a very fast classification scheme that can be extended to many objects.

Minimum Distance Classifier

If we have a set of M classes, and have characteristic feature vector, $\mathbf{O}_i$ for $i = 1, 2, \ldots, M$, for each class.

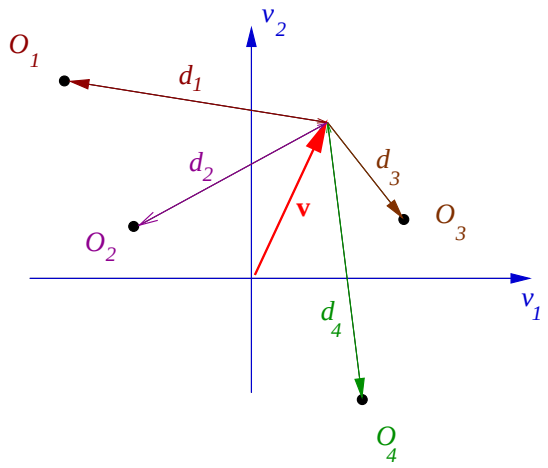
For a new object to be classified we extract its feature vector **v** then form the geometric distance to each of the class vectors by

$$d_i = \sqrt{\sum_{j=1}^{k} (v_j - O_{i_j})^2}$$

we then assign the new object to class j such that

$$d_j < d_i \quad \forall j \neq i$$

(so the closest class vector. In 2-D this looks like



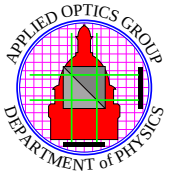
Training the Classifier

We use a pre-classified training set and extract the feature vectors for each class.

The class vector is then just the mean vector for each class, so training is very simple.

This assumes that the each class, each component of the feature vector is equally well clustered.

For this to be optimal it actually assumes an equal Gaussian spread in each component, which is very unlikely to be true!



Feature Selection

The most difficult, and least understood, aspect of pattern recognition is “how to choose features”. The required properties are:

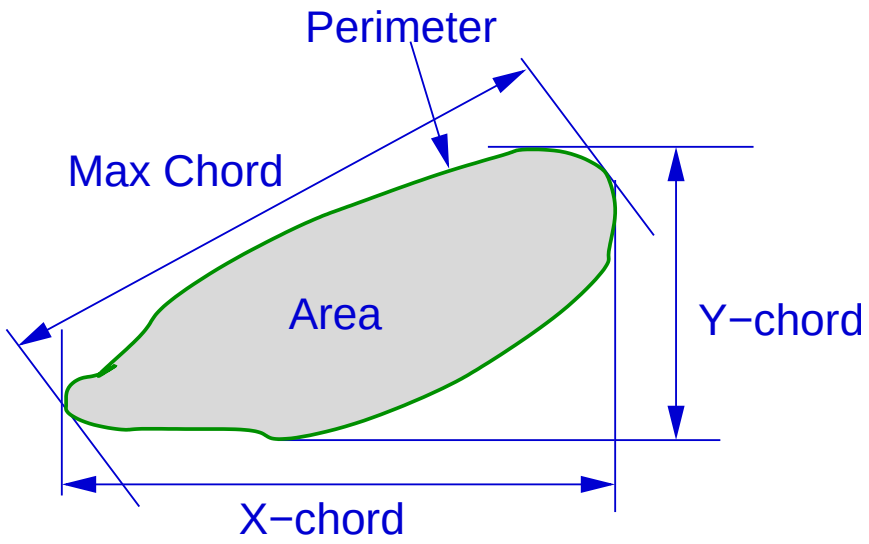
1. Fast and reliable to calculate.
2. Values of the features for one class of object is well clustered in space.
3. Values of features for different classes are well separated in space.
4. Small geometric changes in an object do not make significant difference to the feature values.

The selection of the correct features are critical to the operation.

There are no general rules, we will look at two typical feature types, one related to shape and the other given by the spatial moments of the object.

Shape Features

Extract binary shape by thresholding, then form



where we extract Area, Perimeter, and three chords. Write these as:

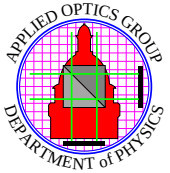
$$
\begin{aligned}
A &= \text{Area} \\
P &= \text{Perimeter} \\
X_c &= \text{X chord} \\
Y_c &= \text{Y chord} \\
M_c &= \text{Max chord}
\end{aligned}
$$

Shape Features

we can then form dimensionless features of

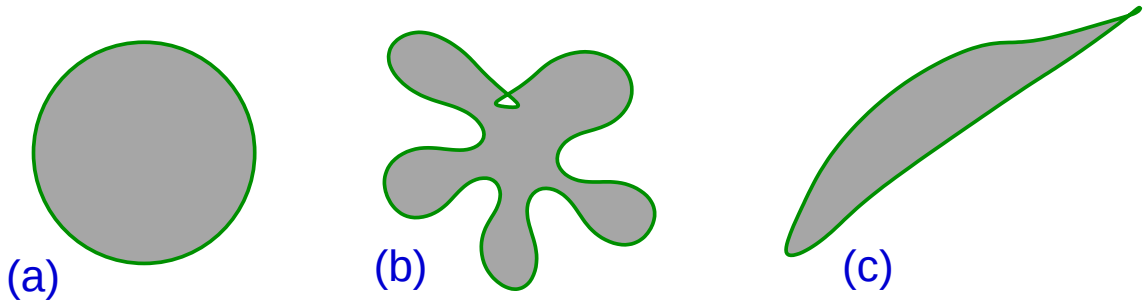
$$
\begin{aligned}
v_1 &= P^2/A \\
v_2 &= X_c/P \\
v_3 &= Y_C/P \\
V_4 &= M_C/P
\end{aligned}
$$

Note that by taking dimensionless combinations the feature vectors do not depend on size of the object but only on its shape.



Shape Features II

Typical results of:



(a)	(b)	(c)
v_1 small	v_1 large	v_1 large
v_2 large	v_2 small	v_2 variable
v_3 large	v_3 small	v_3 variable
v_4 large	v_4 small	v_4 variable

Useful features especially for medical applications, (counting and classification of cell types).

Moments as Features

Again assume we have isolated and extracted the object from the background scene. For image $f(x,y)$ define the 2-D moments are:

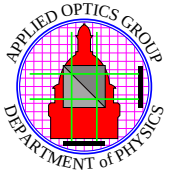
$$m_{p,q} = \iint x^p y^q f(x,y)\,\mathrm{d}x\mathrm{d}y$$

or for a discrete image $f(i,j)$, of size $N \times N$ we have

$$m_{p,q} = \frac{1}{N^2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} i^p j^q f(i,j)$$

So the moments are really an expansion of the image into the complete set of polynomials

$$x^p y^q$$



Moments as Features

Look at the first few moments, they are

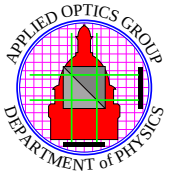
$$m_{0,0} = \iint f(x,y)\,\mathrm{d}x\mathrm{d}y = \mu \quad \text{mean value}$$

$$m_{1,0} = \iint x\, f(x,y)\,\mathrm{d}x\mathrm{d}y = \quad \text{First Order}$$

$$m_{0,1} = \iint y\, f(x,y)\,\mathrm{d}x\mathrm{d}y = \quad \text{First Order}$$

so that a single (isolated) object, the centre of mass is located at

$$\left(\frac{m_{1,0}}{m_{0,0}}, \frac{m_{0,1}}{m_{0,0}}\right) = (\bar{x}, \bar{y})$$



Moments as Features

Since it is useful if the features are independent of the location of the target it is more useful to define the central moments as

$$\mu_{p,q} = \int\int (x-\bar{x})^p (y-\bar{y})^q f(x,y)\,\mathrm{d}x\mathrm{d}y$$

which are the moments calculated about the centre-of-mass. We can then form a feature vector from these moments to give

$$\mathbf{v} = \underbrace{\mu_{0,0}}_{\text{Zero}}, \underbrace{\mu_{2,0}, \mu_{1,1}, \mu_{0,2}}_{\text{Second Order}}, \underbrace{\mu_{3,0}, \mu_{2,1}, \mu_{1,2}, \mu_{0,3}}_{\text{Third Order}} \ldots$$

so we can extend the feature vector to any moment order.

Note:

$$\begin{aligned} \mu_{0,0} &\Rightarrow \text{Average Value} \\ \mu_{1,0}, \mu_{0,1} &\Rightarrow \text{Always zero} \\ \mu_{2,0}, \mu_{1,1}, \mu_{0,2} &\Rightarrow \text{Best fitting ellipse to object} \end{aligned}$$

Not easy to put physical interpretation on the higher order moments.

Scale Invariance

Consider what happens if we scale the object by a factor a is size,

$$f(x,y) \rightarrow f\left(\frac{x}{a},\frac{y}{a}\right)$$

so the moments become

$$m'_{p,q} = \int\int x^p y^q f\left(\frac{x}{a},\frac{y}{a}\right) \mathrm{d}x\mathrm{d}y$$

which we can re-arrange to give

$$m'_{p,q} = a^{p+q+2} \int\int s^p t^q f(s,t)\,\mathrm{d}s\mathrm{d}t = a^{p+q+2} m_{p,q}$$

also for the central moments, we can show that under scaling of a,

$$\mu'_{p,q} = a^{p+q+2} \mu_{p,q}$$

so noting that

$$m'_{0,0} = a^2 m_{0,0}$$

Scale Invariance

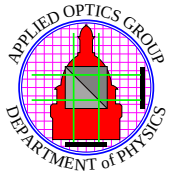
We can use this to normalise the central moments to make them invariant under scale shift to give,

$$\eta_{p,q} = \frac{\mu_{p,q}}{m_{0,0}^{(p+q+2)/2}}$$

Now we have that $\eta_{0,0} = 1$, so we form a scale and translation invariant feature vector by taking.

$$\mathbf{v} = \underbrace{\eta_{2,0}, \eta_{1,1}, \eta_{0,2}}_{\text{Second Order}}, \underbrace{\eta_{3,0}, \eta_{2,1}, \eta_{1,2}, \eta_{0,3}}_{\text{Third Order}}, \ldots$$

which we can extend to higher order if needed.



Rotation Invariance

The properties of moments under rotation can be determined and we can use the same method to normalise the effect out. To third order this leads to a set of 7 invariant features, the first four being,

$$
\begin{aligned}
\phi_1 &= \mu_{2,0} + \mu_{0,2} \\
\phi_2 &= (\mu_{2,0} - \mu_{0,2})^2 + 4\mu_{1,1} \\
\phi_3 &= (\mu_{3,0} - 3\mu_{1,2})^2 + (3\mu_{2,1} - \mu_{0,3})^2 \\
\phi_4 &= (\mu_{3,0} + \mu_{1,2})^2 + (\mu_{2,1} + \mu_{0,3})^2 \\
\phi_5 &= \\
\phi_6 &= \\
\phi_7 &=
\end{aligned}
$$

so we can form a set of features that are invariant to translation, scale, and rotation, depending only on the shape of the object.

In is not possible to directly extend this process to higher orders, but it is possible to use moments based on the complex Zernike polynomials to form a equivalent set of translation, scale and rotation invariant features.

Use of Moments

Moments and moment invariants do form a useful set of features for pattern recognition, but have some limitations, these being:

1. No fast calculation scheme.
2. Scale invariant moments $\eta_{p,q}$ have a huge dynamic range with results in rounding errors during calculation.
3. The higher order moments suffer from noise effects.
4. Some of the rotation invariant combinations are highly numerically unstable.

They have been “out of favour” for some time mainly due to the calculation overhead, but with faster computer hardware they have the potential to make a comeback!