Chapter 19 – Visual pattern recognition
What will we learn?

- What is visual pattern recognition and how does it relate to general pattern recognition?
- What are patterns and pattern classes?
- What is a pattern classifier?
- Which steps are normally needed to design, build, and test a visual pattern classifier?
- How can the performance of visual pattern classifiers be evaluated?
Introduction

• The **goal** of pattern classification techniques: to assign a class to each image (or object within an image) based on a numerical representation of the image's (or object's) properties that is most suitable for the problem at hand.

• Pattern classification techniques:
  • **statistical**
    • Each object or class can be represented as a *feature vector* and make decisions on which class to assign to a certain pattern based on distance calculations or probabilistic models.
  • **structural** (or syntactic).
Fundamentals

- The common goal of pattern classification techniques is to assign a class to an unknown pattern based on previously acquired knowledge about objects and the classes to which they belong.
A statistical pattern classifier processes numerical information from the feature vectors, computes a series of distances or probabilities, and uses those results to make decisions regarding which class label $C(x)$ should be assigned to each input pattern $x$. 

By Oge Marques

Copyright © 2011 by John Wiley & Sons, Inc. All rights reserved.
Design and implementation of a visual pattern classifier

Input image $f(x,y)$ → Pre-processing → Feature Extraction → Feature Selection → Pattern Classification

Application-specific feedback
Design and implementation of a visual pattern classifier

Steps:

1. Define the problem and determine the number of classes involved.
2. Extract features that are most suitable to describe the images and allow the classifier to label them accordingly.
3. Select a classification method or algorithm.
4. Select a dataset.
5. Select a subset of images and use them to train the classifier.
6. Test the classifier.
7. Refine and improve the solution.
Patterns and pattern classes

- A pattern can be defined as an arrangement of descriptors or features.
  - Patterns are usually encoded in the form of feature vectors, strings, or trees.
- A class is a set of patterns that share some common properties.
  - An ideal class is one in which its members are very similar to one another (i.e., the class has high intra-class similarity) and yet significantly different from members of other classes (i.e., inter-class differences are significant).
Patterns and pattern classes

- Sumo wrestlers and table tennis players
Patterns and pattern classes

- Data preprocessing
  - **Noise removal**: data samples that deviate too far from the average value for a class are removed, under the rationale that: (a) there may have been a mistake while measuring (or extracting) that particular sample; (b) the sample is a poor example of the underlying structure of the class.
  - **Normalization**: feature vectors may need to be normalized before distance, similarity, and probability calculations take place.
  - **Insertion of missing data**: (optional).
Training and test sets

- The process of development and testing of pattern classification algorithms usually requires that the dataset be divided into two subgroups:
  - **training set**: used for algorithm development and fine-tuning
  - **test set**: used to evaluate the algorithm's performance.

- The training set contains a small (typically 20% or less), representative subsample of the dataset, selected manually or automatically.

- The size of the training set and the method used to build it are often dependent on the selected pattern classification technique.

- The goal of having two separate sets is to avoid bias in reporting the success rates of the approach.
Confusion matrix

- A 2D array of size $K \times K$ (where $K$ is the total number of classes) used to report raw results of classification experiments.
- The value in row $i$, column $j$ indicates the number of times an object whose true class is $i$ was labeled as belonging to class $j$.
- The main diagonal of the confusion matrix indicates the number of cases where the classifier was successful; a perfect classifier would show all off-diagonal elements equal to zero.
Confusion matrix

- Example 19.1:

<table>
<thead>
<tr>
<th></th>
<th>$\omega_1$</th>
<th>$\omega_2$</th>
<th>$\omega_3$</th>
<th>$\omega_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\omega_1$</td>
<td>97</td>
<td>0</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>$\omega_2$</td>
<td>0</td>
<td>89</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>$\omega_3$</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>$\omega_4$</td>
<td>0</td>
<td>3</td>
<td>5</td>
<td>92</td>
</tr>
</tbody>
</table>

- Example 19.2:
  - Overall error rate: 5.5 % (success rate = 94.5%)
Hit rates, false alarm rates, and ROC curves

• A classical computer vision example: the *object detection* task
  • A CV algorithm is presented with an image and the question: "Is the object present in this image or not?"
    • If the algorithm successfully answers *yes* (and points to where in the image the object is located) when the object is present, it is called a *true positive*.
    • If the algorithm correctly answers *no* when the object is absent, it is called a *true negative*.
• There are two possible errors the algorithm can make:
  • Answering *yes* in the absence of an object (a *false alarm* or *false positive*)
  • Answering *no* when the object is present, i.e., missing the object (a *false negative*).
Hit rates, false alarm rates, and ROC curves

- The cost of a false positive or a false negative is application-dependent.

- The receiver operating characteristic (ROC) curve is a plot that shows the relationship between the correct detection (true positive) rate (also known as hit rate) and the false alarm (false positive) rate.

- The ideal ROC curve is one in which the “knee” of the curve is as close to the top-left corner of the graph as possible, suggesting hit rate close to 100% with a false alarm rate close to zero.
Hit rates, false alarm rates, and ROC curves

- Example of ROC curve
Certain image processing applications, notably image retrieval, have as their goal to retrieve relevant images while not retrieving irrelevant ones.

The measures of performance used in image retrieval borrow from the field of (document) information retrieval and are based on two primary figures of merit: precision and recall.

- **Precision** is the number of relevant documents retrieved by the system divided by the total number of documents retrieved (i.e., true positives plus false alarms).
- **Recall** is the number of relevant documents retrieved by the system divided by the total number of relevant documents in the database (which should, therefore, have been retrieved).
Precision and recall

- Example 19.3:

<table>
<thead>
<tr>
<th></th>
<th>$\omega_1$</th>
<th>$\omega_2$</th>
<th>$\omega_3$</th>
<th>$\omega_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\omega_1$</td>
<td>97</td>
<td>0</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>$\omega_2$</td>
<td>0</td>
<td>89</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>$\omega_3$</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>$\omega_4$</td>
<td>0</td>
<td>3</td>
<td>5</td>
<td>92</td>
</tr>
</tbody>
</table>

$$P = \frac{tp}{tp + fp}$$

$$R = \frac{tp}{tp + fn}$$

$$P_1 = \frac{97}{(97 + 0 + 0 + 0)} = 100\%$$
$$P_2 = \frac{89}{(0 + 89 + 0 + 3)} = 96.74\%$$
$$P_3 = \frac{100}{(2 + 10 + 100 + 5)} = 85.47\%$$
$$P_4 = \frac{92}{(1 + 1 + 0 + 92)} = 97.87\%$$
$$R_1 = \frac{97}{(97 + 0 + 2 + 1)} = 97\%$$
$$R_2 = \frac{89}{(0 + 89 + 10 + 1)} = 89\%$$
$$R_3 = \frac{100}{(0 + 0 + 100 + 0)} = 100\%$$
$$R_4 = \frac{92}{(0 + 3 + 5 + 92)} = 92\%$$
Precision and recall

- Precision can be interpreted as a measure of \textit{exactness}, whereas recall provides a measure of \textit{completeness}.
  - A perfect precision score of 1.0 means that every retrieved document (or image, in our case) was relevant, but does not provide any insight as to whether all relevant documents were retrieved.
  - A perfect recall score of 1.0 means that all relevant images were retrieved, but says nothing about how many irrelevant images might have also been retrieved.
Precision and recall

- **P-R graph**
  - Obtained by calculating the precision at various recall levels.
  - The ideal P-R graph shows perfect precision values at every recall level until the point where all relevant documents (and only those) have been retrieved; from that point on it falls monotonically until the point where recall reaches one.
Precision and recall

- $F_1$: a more compact representation of the precision and recall properties of a system.

$$F_1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$
Precision and recall

Example 19.4:

An image retrieval system produced the following 10 ranked results for a search operation against a database of 500 images, of which 5 are relevant to the query:

<table>
<thead>
<tr>
<th>Rank</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>R</td>
</tr>
<tr>
<td>2</td>
<td>R</td>
</tr>
<tr>
<td>3</td>
<td>N</td>
</tr>
<tr>
<td>4</td>
<td>R</td>
</tr>
<tr>
<td>5</td>
<td>N</td>
</tr>
<tr>
<td>6</td>
<td>N</td>
</tr>
<tr>
<td>7</td>
<td>N</td>
</tr>
<tr>
<td>8</td>
<td>R</td>
</tr>
<tr>
<td>9</td>
<td>N</td>
</tr>
<tr>
<td>10</td>
<td>R</td>
</tr>
</tbody>
</table>
Precision and recall

- Example:
  - Recall and precision values
  - P-R graph

<table>
<thead>
<tr>
<th>Recall</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
<td>1.0000</td>
</tr>
<tr>
<td>0.4</td>
<td>1.0000</td>
</tr>
<tr>
<td>0.4</td>
<td>0.6667</td>
</tr>
<tr>
<td>0.6</td>
<td>0.7500</td>
</tr>
<tr>
<td>0.6</td>
<td>0.6000</td>
</tr>
<tr>
<td>0.6</td>
<td>0.5000</td>
</tr>
<tr>
<td>0.6</td>
<td>0.4286</td>
</tr>
<tr>
<td>0.8</td>
<td>0.5000</td>
</tr>
<tr>
<td>0.8</td>
<td>0.4444</td>
</tr>
<tr>
<td>1.0</td>
<td>0.5000</td>
</tr>
</tbody>
</table>
Distance and similarity measures

- Euclidean distance:

$$d_E = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2}$$

- Manhattan (or city-block) distance:

$$d_M = \sum_{i=1}^{n} |a_i - b_i|$$

- Minkowski distance:

$$d_M = \left[ \sum_{i=1}^{n} |a_i - b_i|^r \right]^{1/r}$$
Distance and similarity measures

• Similarity measures
  • Vector inner product:
    \[
    \sum_{i=1}^{n} a_i b_i = a_1 b_1 + a_2 b_2 + \ldots + a_n b_n
    \]

• Tanimoto metric:
Statistical pattern classification techniques

- So far, objects' properties are represented using *feature vectors* which are projected onto a *feature space*.
- The resulting points in the $n$-dimensional feature space should ideally be distributed in a way that correlates proximity in the feature space with similarity among the actual objects.
- In other words, feature vectors associated with objects from the same class will appear close together as *clusters* in the feature space.
Statistical pattern classification techniques

- The job of a statistical pattern classification technique is to find a discrimination curve (or a *hyper-surface*, in the case of an $n$-dimensional feature space) that can tell the clusters (and, therefore, classes) apart.
Statistical pattern classification techniques

- Three main techniques:
  - Minimum distance classifier
  - k-nearest neighbors
  - Bayesian classifier
Statistical pattern classification techniques

- Minimum distance classifier

\[ d_j(x) = \|x - m_j\| \]
\[ m_j = \frac{1}{N_j} \sum_{x \in \omega_j} x_j \]
Statistical pattern classification techniques

- k-nearest neighbors classifier
  - Works by computing the distance between an unknown pattern's feature vector and the $k$ closest points (not classes) to it in the feature space.
  - It then assigns the unknown pattern to the class to which the majority of the $k$ sampled points belong.
  - Advantages: simplicity (e.g., no assumptions need to be made about the probability distributions of each class) and versatility (e.g., it handles overlapping classes or classes with complex structure well).
  - Disadvantages: computational cost.
Statistical pattern classification techniques

- k-nearest neighbors classifier: example
Statistical pattern classification techniques

- **Bayesian classifier**
  - **Rationale:** A classification decision can be made based on the \textit{probability distributions} of the training samples for each class.
  - An unknown object is assigned to the class to which it is \textit{more likely} to belong based on the observed features.

\[
p(\omega_k|x) = \frac{p(x|\omega_k)P(\omega_k)}{p(x)} = \frac{\sum_{k=1}^{W} p(x|\omega_k)P(\omega_k)}{\sum_{k=1}^{W} p(x|\omega_k)P(\omega_k)}
\]
Hands-on

- Tutorial 19.1: Feature extraction and representation (page 491)
Tutorial 19.1 – Pattern classification

- OCR problem
  - Dataset: 1000 training images + 1000 test images
  - 10 classes: 0, 1, 2, ..., 9
- KNN classifier (courtesy of the STPRtool)
- Preprocessing
Tutorial 19.1 – Pattern classification

- 1\textsuperscript{st} attempt: using eccentricity and Euler number as features
- Results (training set)
- Partial conclusions
Tutorial 19.1 – Pattern classification

• Results (test set)

classification_result on test set data: 583 out of 1000 misclassified
class "0"  missclassified   66 times
class "1"  missclassified   46 times
class "2"  missclassified   61 times
class "3"  missclassified   68 times
class "4"  missclassified   77 times
class "5"  missclassified   71 times
class "6"  missclassified   67 times
class "7"  missclassified   55 times
class "8"  missclassified    2 times
class "9"  missclassified   70 times
Tutorial 19.1 – Pattern classification

- Results (test set)
Tutorial 19.1 – Pattern classification

- 2\textsuperscript{nd} attempt: using the 13 x 13 gray level values as FV

- Results (test set)

- Final conclusions