Review on Pattern Recognition

• Reading Assignments

R. Duda, P. Hart, and D. Stork, *Pattern Classification*, John-Wiley, 2nd edition, 2001 (chapter 1, hard-copy).

C. Bishop, *Neural Networks for Pattern Recognition*, Oxford, 1997 (pp. 1-17, hard-copy).

A. Jain, R. Duin, and J. Mao, "Statistical Pattern Recognition: A Review", *IEEE Transactions on Pattern Analysis and Machine Intelligenve*, vol. 22, no. 1, pp. 4-37, 2000 (read pp. 4-17, skip what you do not understand, on-line)

Review on Pattern Recognition

• What is PR ?

It is the study of how machines **(1)** can observe the environment **(2)** learn to distinguish patterns of interest from their background, **(3)** make sound and reasonable decisions about the categories of the patterns.

• What is a pattern ?

- Some authors define a pattern as "*the opposite of a chaos; it is an entity, vaguely defined, that could be given a name*".

- A pattern can be a fingerprint image, a human face, a speech signal.

• Emerging applications in PR

• Is a single approach enough ?

- In many of the emerging applications, it is clear that no single approach for classification is "optimal".

- It has been common practice to combine several sensing modalities and classifiers.

• Important issues in the design of a PR system

- Definition of pattern classes.
- Sensing environment.
- Pattern representation.
- Feature extraction and selection.
- Cluster analysis.
- Selection of training and test examples.
- Performance evaluation.

• A example of a PR system

<Fig -IntroPaper>

The Complexity of the PR Problem

• An example

- Separate *sea bass* from *salmon* using optical sensing.

• How can we separate them ?

- Length
- Lightness
- Width
- Number and shape of fins.
- Position of the mouth.

• What are the difficulties ?

- Variations in lighting

- Position of the fish on the conveyor belt.

- "Static" noise from camera.

• Goal of Pattern Recognition

- Hypothesize the *models* that describe the two populations.

- Process the sensed data to eliminate noise.

- Given a sensed pattern, choose the model that best represents it.

• Components of a prototype system

- *Preprocessing* (e.g., adjust lighting, segmentation etc.)
- *Feature Extraction* (reduce data by measuring certain features).
- *Classification* (evaluate the evidence presented and make a final decision).

• **Examples of possible models**

- A model based on *length* information.

- Choose the optimal threshold using a number of *training examples*.

Fig p6

- A model based on *average lightness*.

Fig p6

• Classification error (cost)

- There are two possible classification errors.

  (1) deciding the fish was a sea bass when it was a salmon.
  (2) deciding the fish was a salmon when it was a sea bass.

- Which error is more important ?

• Decision Theory

- Find a *decision rule* that minimizes classification error.

- Single features might not yield the best performance.

- Combinations of features might yield better performance.

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

x_1: lightness
x_2: width

- Partition the *feature space* into two regions by finding the *decision boundary* that minimizes the error.

Fig p7

• How many features and which ?

- Correlated features do not improve performance.

- It might be difficult to extract certain features.

- It might be computationally expensive to extract certain features.

• The curse of dimensionality

- Adding too many features can, paradoxically, lead to a worsening of performance.

- Let’s see why:

*Divide each of the input features into a number of intervals, so that the value of a feature can be specified approximately by saying in which interval it lies.

* The whole input space is divided into a large number of cells.

* If each input feature is divided into M divisions, then the total number of cells is M^d (d: number of features) and this grows exponentially with d.

* Since each cell must contain at least one point, the number of training data grows exponentially !!

Fig. p8 Bishop

• **Model complexity**

- We can get perfect classification performance on the training data by choosing complex models.

- Complex models are *tuned* to the particular training samples, rather than on the characteristics of the true model.

Fig p8

• **Generalization**

- The ability of the classifier to produce correct results on *novel* patterns.

- How can we improve generalization performance ?

(1) More training examples (i.e., better pdf estimates).
(2) Simpler models (i.e., simpler classification boundaries) usually yield better performance.

Fig p9

• **Central questions in pattern recognition**

- How should we quantify and favor simpler classifiers ?

- Can we *predict* how well the system will generalize to novel patterns ?

• **Can we build general purpose pattern recognition systems ?**

- Different decision tasks may require different features.

- Different features might yield different boundaries.

- Different tradeoffs (e.g., classification error) exist for different tasks.

• **Pattern representation schemes**

- Patterns can be represented as:

  (1) Vectors of real-numbers.
  (2) Lists of attributes.
  (3) Descriptions of parts and their relationships.

- Similar patterns should have similar representations.

- Patterns from different classes should have dissimilar representations.

- Choose features that are robust to noise.

- Favor features that lead to simpler decision regions.

• **Using domain knowledge in classifier design**

- When there is not sufficient training data, incorporate domain knowledge:

  Model how each pattern in generated (*analysis by synthesis*).
  (this is difficult !! e.g., recognize all types of chairs)

  Incorporate *some* knowledge about the pattern generation method.
  (e.g., optical character recognition (OCR) assuming characters are sequences of strokes)

Various areas of Pattern Recognition

• **Template matching**

- The pattern to be recognized is matched against a stored template while taking into account all allowable pose (translation and rotation) and scale changes.

• **Statistical pattern recognition**

- Focuses on the statistical properties of the patterns (i.e., probability densities).

• **Artificial Neural Networks**

- Inspired by biological neural network models.

• **Syntactic pattern recognition**

- Decisions consist of logical rules or grammars.

Summary of main problems in Pattern Recognition

• Feature Extraction

- It is a domain-specific problem which influences classifier's performance.

- Which features are most promising ?

- Are there ways to automatically learn which features are best ?

- How many should we use ?

• Noise

- Various types of noise (e.g., shadows, conveyor belt might shake, etc.)

- Noise can reduce the reliability of the feature values measured.

- Knowledge of the noise process can help improve performance.

• Overfitting

- Models complex than necessary lead to *overfitting* (i.e., good performance on the training data but poor performance on novel data).

- How can we adjust the complexity of the model ? (not very complex or simple).

- Are there principled methods for finding the best complexity ?

• Model Selection

- How do we know when to reject a class of models and try another one ?

- Is the model selection process just a trial and error process ?

- Can we automate this process ?

• **Prior Knowledge**

- Can be derived from information about the production of patterns.

- Can be derived from knowledge about the *form* of the underlying categories (e.g., chairs) or attributes of the patterns (e.g., faces).

• **Missing Features**

- Certain features might be missing (e.g., due to occlusion).

- How should the classifier make the best decision with missing features ?

- How should we train the classifier with missing features ?

• **Segmentation**

- Individual patterns have to be segmented.

  How can we segment without having categorized them first ?

  How can we categorize them without having segmented them first ?

- How do we "group" together the proper number of elements ?

• **Context**

- Input dependent information can improve classification.

- How precisely should we incorporate such information ?

• **Invariance**

- Useful classifiers should be invariant to transformations such as:
  translations, rotations, size, reflections, non-rigid deformations

- How do we build classifiers that are invariant to such complex changes ?

• **Evidence Pooling**

- Performance can be improved using a "pool" of classifiers.

- How should we combine multiple classifiers ?

• **Costs and Risks**

- Each classification is associated with a cost or risk (e.g., classification error).

- We design classifiers by minimizing some cost or risk.

- How can we incorporate knowledge about such risks ?

- Can we estimate the total risk ahead of time ?

- Can we estimate the *lowest* possible risk of *any* classifier ?

• **Computational Complexity**

- How does an algorithm *scale* with (i) the number of feature dimensions, (ii) number of patterns, or (iii) number of categories ?

- Brute-force approaches might lead to perfect classifications results but usually have impractical time and memory requirements.

- What is the tradeoff between computational ease and performance ?

Learning and Adaptation

- Learning from examples is an effective method for developing classifiers.

- Choose a model and use learning from training examples to estimate the parameters of the model.

• Supervised Learning

- A teacher provides a category label for each training example.

- We seek to reduce the sum of the costs of these patterns.

- How do we know if a learning algorithm is powerful enough to learn the solution ?

- Other issues: stability to initial conditions, convergence speed, overfitting etc.

• Unsupervised Learning

- There is no teacher.

- The system forms clusters or "natural groupings" of the input patterns.

- Requires hypothesizing the number of clusters.

• Reinforcement Learning

- No desired category is given, only teaching feedback that a classification is "correct" or "wrong".