Lecture 1: Introduction to Pattern Recognition

Course: 2110573 Pattern Recognition

2026-01-08
version 0.1

1 What is Pattern Recognition?

Pattern recognition is a fundamental branch of machine learning. A common definition is:

> "Pattern recognition is a branch of machine learning that focuses on the recognition of patterns and regularities in data, although it is in some cases considered to be nearly synonymous with machine learning." — *Wikipedia*

This field is closely related to, and often overlaps with, other domains such as:

- Artificial Intelligence (AI)
- Data Mining (DM)
- Knowledge Discovery in Databases (KDD)
- Statistics
- Data Science

At their core, all these fields are concerned with the same fundamental question: **how do we learn from data?**

1.1 A Note on AI

- **Classical Definition:** A system that appears intelligent.
- **Modern Context:** The field is currently focused on "Narrow AI," which involves creating specialized systems that are very good at a single task (e.g., image recognition, language translation). This is largely synonymous with Machine Learning (ML).
- **Artificial General Intelligence (AGI):** This is the hypothetical ability of an agent to understand or learn any intellectual task a human can. It remains a topic of philosophical debate and long-term research.

2 Course Philosophy

The main goal of this course is to go beyond treating models as "black boxes." In this course, you will:

- Understand models on a deeper level.
- Implement algorithms from scratch to solidify your understanding.

As François Chollet states, "if you understand something clearly, you should be able to describe it in precise algorithmic terms to a computer."

3 Types of Machine Learning

There are three main categories of machine learning:

1. **Supervised Learning:** Learn a model from labeled data, i.e., pairs of (input, output).
2. **Unsupervised Learning:** Discover hidden structures in unlabeled data (input only, no output).
3. **Reinforcement Learning:** Train an agent to take actions in an environment to maximize a cumulative reward.

There is also a precursor to machine learning known as **rule-based systems**. An example is a 7-segment display decoder, where explicit 'IF-THEN' rules map the state of the 7 segments to a specific digit.

4 The Machine Learning Workflow

The goal of machine learning is to find the best function, $F(x)$, that maps an input x to an output y automatically from data. The typical workflow involves several key stages.

4.1 1. Feature Extraction

This is the process of transforming raw data into a numerical **feature vector** (x) that is informative and digestible for a model. This is one of the most critical steps, as the quality of the features directly impacts the model's performance. This adheres to the principle of "Garbage in, Garbage out."

4.2 2. Modeling (Training)

During the training phase, a learning algorithm uses a **training set** of feature vectors and their corresponding desired outputs (y) to learn a model (h).

$$\text{Training Data } (x, y) \rightarrow \text{Learning Algorithm} \rightarrow \text{Model } (h)$$

4.3 3. Evaluation (Testing)

In the testing phase, the trained model (h) is used to make predictions on new, unseen input data.

$$\text{New Input } (x) \rightarrow \text{Model } (h) \rightarrow \text{Predicted Output } (\hat{y})$$

The model's performance is then evaluated by comparing its predictions ($\hat{y}$) to the actual correct answers, or **ground truth** (y).

5 Model Evaluation

To compare different models, we need metrics to quantify their performance.

5.1 Ground Truth

We evaluate a model by comparing its output to the "correct answer" or ground truth. However, establishing ground truth can be difficult, especially for subjective tasks (e.g., rating a machine translation) or in cases with ambiguous data (e.g., labeling complex road signs or lane lines).

5.2 Metrics for Classification

For a binary classification or detection problem, there are four possible outcomes, often summarized in a **confusion matrix**:

	Detector Prediction	
Actual	**Positive**	**Negative**
Positive	True Positive (TP)	False Negative (FN)
Negative	False Positive (FP)	True Negative (TN)

From this, we derive several common metrics:

- **Accuracy:** $\frac{TP+TN}{TP+TN+FP+FN}$. Note: Accuracy can be misleading on biased datasets.
- **Precision:** $\frac{TP}{TP+FP}$. Of all the positive predictions, how many were actually correct? (High precision is critical for tasks like spam detection).
- **Recall (Sensitivity):** $\frac{TP}{TP+FN}$. Of all the actual positive cases, how many did the model correctly identify? (High recall is critical for tasks like cancer screening).
- **F1-Score:** $2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$. The harmonic mean of precision and recall, providing a single score that balances both.

5.3 Other Considerations

Besides predictive accuracy, other factors are important when evaluating models:

- Training and testing time
- Memory requirements
- Latency (time to make a single prediction)
- Parallelizability

6 Course Walkthrough

The course is structured into two main parts:

- **Traditional Machine Learning:** Covers foundational topics like K-Means, Regression, Naive Bayes, GMM, SVM, and basic Neural Networks (NN).
- **Deep Learning:** Moves into advanced topics including CNNs, Transformers, Generative Models (GANs, VAE, Diffusion), Self-supervised learning, and Reinforcement Learning.

The schedule includes regular homework assignments, a midterm, a final project, and a final exam.

7 Introduction to Unsupervised Learning

Unsupervised learning is a type of machine learning where the goal is to discover hidden patterns and structures in unlabeled data (i.e., data without a predefined target variable ‘y‘).

Key applications include:

- Customer/product segmentation
- General data analysis and exploration
- Identifying the number of speakers in a meeting recording
- Assisting supervised learning tasks

Clustering is a fundamental task in unsupervised learning that tries to automatically discover natural groupings within the data.

8 Nearest Neighbour Classification

Before diving into K-means, it’s helpful to understand classification using nearest neighbors.

8.1 K-Nearest Neighbour (KNN) Classification

The core idea is to classify a new data point based on the labels of its neighbors in the training data.

1. **Given a query data point:**
2. For every point in the training data, compute the distance to the query point.
3. Identify the 'K' nearest data points (neighbors).
4. Assign a label to the query point by taking a majority vote among its K neighbors.

A simple version (where K=1) is called Nearest Neighbour classification, but it is highly susceptible to noise. Using a larger K and a voting scheme makes the method more robust. For weighted k-NN, the votes of closer neighbors can be given more importance (e.g., weighted by the inverse of their distance).

8.2 Distance Metrics

To find the "closest" neighbors, we need a distance or similarity measure. Common measures include:

- **Euclidean Distance:** The straight-line distance between two points.

$$d(X_1, X_2) = \sqrt{\sum_{i=1}^{n} (x_{1,i} - x_{2,i})^2}$$

- **Cosine Similarity:** Measures the cosine of the angle between two vectors, indicating their orientation similarity.

$$\text{sim}(X_1, X_2) = \frac{X_1 \cdot X_2}{\|X_1\| \|X_2\|} = \frac{\sum_{i=1}^{n} x_{1,i} x_{2,i}}{\sqrt{\sum_{i=1}^{n} x_{1,i}^2} \sqrt{\sum_{i=1}^{n} x_{2,i}^2}}$$

Other measures like Jaccard distance and Earth Mover's distance also exist.

8.3 KNN Runtime

The runtime complexity for classifying J query points against a training set of N points is **O(JN)**. This can be computationally expensive for large datasets. Techniques like using centroids can speed this up.

9 K-means Clustering

K-means is an iterative algorithm that partitions a dataset into K distinct, non-overlapping clusters.

9.1 The Algorithm

The algorithm aims to minimize the distance between data points and their assigned cluster's centroid.

1. **Initialization:** Randomly select K data points from the dataset to serve as the initial centroids.
2. **Assignment Step:** Assign each data point in the dataset to the nearest centroid. The "nearest" is determined using a distance metric, typically Euclidean distance.
3. **Update Step:** Recalculate the centroid of each cluster by taking the mean of all data points assigned to it.
4. **Repeat:** Repeat the Assignment and Update steps until the centroids no longer change significantly, meaning the cluster assignments have stabilized.

9.2 Characteristics and Assumptions

- The number of clusters, **K**, must be specified in advance.
- The algorithm is guaranteed to converge, but it may converge to a **local minimum**.
- **Bad initializations** can lead to poor clustering results. A common solution is to run the algorithm multiple times with different random initializations and select the best outcome (e.g., the one with the lowest overall variance).
- The model implicitly assumes that clusters are spherical, evenly sized, and have similar density.

10 Selecting the Number of Clusters (K)

Choosing the right value for K is a critical step in K-means clustering.

10.1 The Elbow Method

One of the most common methods is the Elbow Method.

1. Run the K-means algorithm for a range of K values (e.g., K=1 to 10).
2. For each K, calculate the **fraction of explained variance**.

$$\text{Fraction of explained variance} = \frac{\text{Between-cluster variance}}{\text{All-data variance}}$$

3. Plot the fraction of explained variance as a function of K.
4. The "elbow" of the curve—the point where the rate of increase sharply declines—is considered the optimal K. This represents the point of diminishing returns, where adding more clusters doesn't significantly explain more variance.

10.2 Other Methods

- **Variance Threshold:** Choose the minimum K that explains a certain percentage (e.g., 95%) of the total variance.
- **Performance Maximization:** If the clustering is part of a larger supervised learning task, choose the K that maximizes the performance (e.g., accuracy) on a validation or test set.

11 Notes of K-means and Other Clustering Methods

While K-means is powerful and widely used, other clustering methods exist that may be more suitable for different types of data.

- **K-mode and K-median:** Variations of K-means used for categorical data (K-mode) or to reduce sensitivity to outliers (K-median).
- **Spectral Clustering:** A technique that clusters data using the eigenvalues of a similarity matrix, effective for complex and non-convex cluster shapes.
- **DBSCAN (Density-Based Spatial Clustering of Applications with Noise):** A density-based method that is very robust, can find arbitrarily shaped clusters, and does not require the number of clusters to be specified beforehand.

12 Introduction to Regression

Regression analysis is a set of statistical processes for estimating the relationships between a dependent variable (often called the 'outcome' or 'target') and one or more independent variables (often called 'predictors', 'covariates', or 'features'). The goal is to predict a continuous output value.

For example, we might want to predict the amount of rainfall (our target, y) based on several features (x), such as the type of crops planted, temperature, etc. We are given a training dataset of m examples, like so:

Feature 1	**Feature 2**	...	**Feature n**	**Target (y)**
$x_{1,1}$	$x_{1,2}$	$\cdots$	$x_{1,n}$	y_1
$x_{2,1}$	$x_{2,2}$	$\cdots$	$x_{2,n}$	y_2
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
$x_{m,1}$	$x_{m,2}$	$\cdots$	$x_{m,n}$	y_m

13 The Linear Regression Model

In linear regression, we assume a linear relationship between the input features and the output target. Our model, or *hypothesis*, is represented by the function $h_\theta(x)$:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n \tag{1}$$

Here, the θ_i values are the *parameters* (or *weights*) of the model. To simplify this notation, we can introduce $x_0 = 1$. This allows us to write the hypothesis in a more compact vector form:

$$h_\theta(\mathbf{x}) = \sum_{j=0}^{n} \theta_j x_j = \boldsymbol{\theta}^T \mathbf{x} \tag{2}$$

where $\boldsymbol{\theta}$ and $\mathbf{x}$ are column vectors representing the parameters and the feature values, respectively. Our goal is to find the optimal values for the parameters $\boldsymbol{\theta}$ that make our predictions $h_\theta(\mathbf{x})$ as close to the actual values y as possible.

14 The Cost Function (Mean Squared Error)

To find the best parameters $\boldsymbol{\theta}$, we need a way to measure how well the model is performing. We do this using a *cost function*, also known as a *loss function*. A common choice for regression problems is the **Mean Squared Error (MSE)**.

The MSE measures the average of the squares of the errors, i.e., the average squared difference between the estimated values and the actual value. For easier computation of the gradient, we often use half of the sum of squared errors:

$$J(\boldsymbol{\theta}) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(\mathbf{x}_i) - y_i)^2 = \frac{1}{2m} \sum_{i=1}^{m} (\boldsymbol{\theta}^T \mathbf{x}_i - y_i)^2 \tag{3}$$

where m is the number of training examples. We want to find the $\boldsymbol{\theta}$ that minimizes $J(\boldsymbol{\theta})$.

15 Minimizing the Cost Function

15.1 Gradient Descent

Gradient Descent is an iterative optimization algorithm used to find the minimum of a function. The main idea is to take repeated steps in the opposite direction of the gradient (or derivative) of the function at the current point, because this is the direction of steepest descent.

The update rule for a single parameter θ_j is:

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) \tag{4}$$

where α is the *learning rate*, a small positive number that controls the step size.

For our linear regression cost function, the partial derivative is:

$$\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(\mathbf{x}_i) - y_i) x_{i,j} \tag{5}$$

So, the complete update rule for each parameter θ_j in the model is:

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(\mathbf{x}_i) - y_i) x_{i,j} \tag{6}$$

This update is performed simultaneously for all $j = 0, \ldots, n$. This method is known as **Batch Gradient Descent** because it uses all m training examples in each step. Variations include **Stochastic Gradient Descent (SGD)**, which uses one example at a time, and **Mini-batch Gradient Descent**, which uses a small subset of examples.

15.2 The Normal Equation: An Analytical Solution

Instead of an iterative algorithm, it's also possible to solve for the optimal $\boldsymbol{\theta}$ analytically. This method is called the Normal Equation. We can express the cost function in matrix form. Let $\mathbf{X}$ be the design matrix (with each row being a training example $\mathbf{x}_i^T$) and $\mathbf{y}$ be the vector of target values.

$$J(\boldsymbol{\theta}) = \frac{1}{2m} (\mathbf{X}\boldsymbol{\theta} - \mathbf{y})^T (\mathbf{X}\boldsymbol{\theta} - \mathbf{y}) \tag{7}$$

To minimize this, we take the gradient with respect to $\boldsymbol{\theta}$ and set it to zero.

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}} \frac{1}{2m} (\boldsymbol{\theta}^T \mathbf{X}^T \mathbf{X} \boldsymbol{\theta} - 2\boldsymbol{\theta}^T \mathbf{X}^T \mathbf{y} + \mathbf{y}^T \mathbf{y}) \\
&= \frac{1}{2m} (2\mathbf{X}^T \mathbf{X} \boldsymbol{\theta} - 2\mathbf{X}^T \mathbf{y}) \\
&= \frac{1}{m} (\mathbf{X}^T \mathbf{X} \boldsymbol{\theta} - \mathbf{X}^T \mathbf{y})
\end{aligned}$$

Setting the gradient to zero:

$$\begin{aligned}
\mathbf{X}^T \mathbf{X} \boldsymbol{\theta} - \mathbf{X}^T \mathbf{y} = 0 \\
\mathbf{X}^T \mathbf{X} \boldsymbol{\theta} = \mathbf{X}^T \mathbf{y}
\end{aligned}$$

This gives us the final equation for $\boldsymbol{\theta}$:

$$\boldsymbol{\theta} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y} \tag{8}$$

Pros and Cons

- **Gradient Descent**: Needs a learning rate α. Works well with a very large number of features. Is iterative.
- **Normal Equation**: No need for a learning rate. It's a one-step calculation. However, calculating the inverse of $\mathbf{X}^T\mathbf{X}$ is computationally expensive, with a complexity of roughly $O(n^3)$ where n is the number of features. It can be slow if n is very large. Also, $\mathbf{X}^T\mathbf{X}$ may not be invertible if features are redundant or if there are more features than training examples.

16 Summary

- Basic concept of learning from data
- We have learned two types of learning methods so far
  1. **Supervised learning – Linear Regression**
     - Learn a model F from pairs of (x,y)
     - Goal – find a model, use the model in applications
  2. **Unsupervised learning – K-mean clustering**

    - Discover the hidden structure in unlabeled data x (no y)
    - Goal - find insights, turn it into actions

- The concept of setting the loss function and optimize for it is a key concept in machine learning that will keep coming up in this course.