第 3 章
自主性和人工智能基础

3.1 引言

在学习自主智能无人系统的自主定位、运动规划等内容之前，需要了解与自主性及人工智能相关的基础知识，这些基础知识是实现无人系统自主性和智能性的理论基础。

自主性和人工智能基础的重要内容包括机器学习和神经网络等。机器学习主要分为有监督学习、无监督学习和强化学习三大类。有监督学习常用于分类和回归问题的求解，而无监督学习常用于聚类与降维等任务的求解，强化学习主要应用于那些需要与环境进行交互的决策和控制策略的求解。神经网络则是一种模仿人脑神经元工作原理的计算模型，其可以通过学习和调整权重实现自适应性，从而完成复杂的分类、回归、聚类等任务。神经网络是实现机器学习的一种方法，实际应用中常常使用深度神经网络进行监督学习（如分类或回归任务）、无监督学习（如聚类或降维任务）以及强化学习（如决策或控制任务）。

本章将对自主智能无人系统中与人工智能相关的基础知识展开介绍。首先，对机器学习中的基本定义和典型求解方法作简要介绍，包括监督学习与无监督学习。随后，介绍马尔可夫决策过程和机器学习中的基本概念和求解方法。进一步地，总结概括几种典型人工神经网络模型的结构与工作原理，包括循环神经网络、长短期记忆网络、生成对抗网络及注意力机制等。最后，将神经网络和机器学习方法应用于一个无人驾驶轨迹预测问题中，以便读者更好地理解前述内容。

3.2 机器学习基础

本节主要介绍机器学习中涉及的基本概念以及两类经典学习方法：监督学习和无监督学习。在基本概念部分，介绍机器学习的定义以及常用的基本术语；在监督学习和无监督学习部分，分别介绍相关概念和一些常用算法的基本原理。

3.2.1 基本概念

1. 机器学习三要素

机器学习 (machine learning) 是一门研究如何利用数据和计算方法来改善系统性能的学科。它一般包含 3 个要素：**模型** (model)、**策略** (strategy) 和**算法** (algorithm)。

模型：用于对数据进行预测与分析。

策略：学习过程中判断模型效果优劣的准则。

算法：学习过程中模型的具体计算方法。

2. 训练、测试和验证

训练、测试和验证的关系如图 3.1 所示。

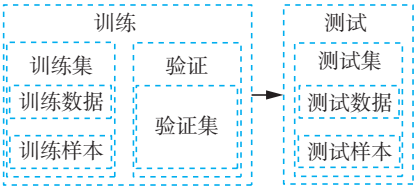

图 3.1 训练、测试和验证的关系

通过设计不同的策略和算法，并提供相应的数据，就可以学习得到不同的模型。其中，学习过程称为**训练** (training)，提供的数据称为**训练数据** (training data)，训练数据中的每个样本称为**训练样本** (training sample)，训练样本的集合称为**训练集** (training set)。

通过训练得到模型以后，向该模型输入相应的数据，就可以输出预测结果。其中，预测的过程称为**测试** (testing)，用于测试的数据称为**测试数据** (testing data)，测试数据中的每个样本称为**测试样本** (testing sample)，测试样本的集合称为**测试集** (testing set)。

在训练过程中，存在一个反馈过程，该过程被称为**验证** (validation)。验证是指在训练过程中，使用不同于训练集的数据集进行测试，并根据测试结果进行参数调整和模型选择的过程。其中，用于验证的数据集称为**验证集** (validation set)。

3. 模型评估和误差

测试结束后，需要将模型的预测输出与样本的真实输出进行对比，这个过程称为**模型评估**。通常，将模型实际预测输出与样本的真实输出两者的差异称为**误差** (error)，在训练集上的误差称为**训练误差** (training error) 或**经验误差** (empirical error)，在测试集上的误差称为**测试误差** (testing error)，在新样本上的误差称为**泛化误差** (generalization error)。

4. 数据集和样本

训练集、测试集和验证集三者构成了机器学习中的完整**数据集** (data set)。数据集的基本单位称为**样本** (sample) 或**示例** (instance)，它由用于描述自身性质的记录组成，这个记录被称为**属性** (attribute) 或**特征** (feature)，属性的取值被称为**属性值** (attribute value)。在某些情况下，样本可能会带有对应的真实输出记录，这个记录被称为**标记信息**或**标签** (label)。在不讨论强化学习的情况下，根据训练数据是否拥有标签信息，可以将机器学习分为**监督学习**

(supervised learning) 和**无监督学习** (unsupervised learning)。其中，训练数据含有标记信息的机器学习方法称为监督学习，训练数据不含标记信息的机器学习方法称为无监督学习。

3.2.2 监督学习

监督学习的目标是通过带有标记信息的训练数据（即每个数据点都有对应的输出结果或“标签”）来学习一个关于输入（特征）与输出（标签）之间关系的模型，并应用该模型去预测新输入的可能输出。这个模型的一般形式为**决策函数**或**条件概率分布**。

决策函数： $Y = f(X)$。

条件概率分布： $P(Y|X)$。

其中，X 和 Y 分别为输入变量和输出变量。

分类 (classification) 和**回归** (regression) 是监督学习的两项经典任务。

（1）**分类：**当输出变量 Y 取有限个离散值时，该监督学习任务便称为分类问题。此时，输入变量 X 可以是离散的，也可以是连续的。根据分类的类别数量，可以把分类任务分为**二分类问题**和**多分类问题**。二分类问题表示分类类别仅有两种，多分类问题则存在多个不同的分类类别。

（2）**回归：**回归旨在预测输入变量 X 和输出变量 Y 之间的关系，特别是输出变量随输入变量变化的关系。根据输入变量的个数，可以把回归任务分为**一元回归**和**多元回归**。其中，一元回归的输入变量只有一个，而多元回归的输入变量有多个。同时，回归任务也可以根据输入变量与输出变量之间的关系类型分为**线性回归**和**非线性回归**。

下面是一些经典的监督学习算法。

1. 线性回归算法

线性回归 (linear regression) **算法**是一种回归算法，核心思想是建立预测值与所有样本属性之间的线性关系，即

$$f(\boldsymbol{x}) = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + b$$

其中，$\boldsymbol{x}$ 可表示为 $\boldsymbol{x} = [x_1, x_2, \cdots, x_n]^{\mathrm{T}}$，$x_1, x_2, \cdots, x_n$ 分别为样本 $\boldsymbol{x}$ 的 n 个属性上的取值，$w_1, w_2, \cdots, w_n, b$ 为要学习的参数，$f(\boldsymbol{x})$ 为预测值。向量形式是

$$f(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x} + b$$

其中，$\boldsymbol{w} = [w_1, w_2, \cdots, w_n]^{\mathrm{T}}$。

线性回归的目标是最小化模型预测值与训练集 D 中每个样本标注之间的误差平方，即

$$(\boldsymbol{w}^*, b^*) = \underset{(\boldsymbol{w}, b)}{\operatorname{argmin}} \sum_{i=1}^{m} \left(f(\boldsymbol{x}_i) - \boldsymbol{y}_i\right)^2$$

其中，$\boldsymbol{x}_i$ 和 $\boldsymbol{y}_i$ 分别为训练集 D 中第 i 个样本和它对应的标注，$f(\boldsymbol{x}_i)$ 为模型对于样本 $\boldsymbol{x}_i$ 的预测值。

可以通过**最小二乘法** (ordinary least squares) 求解上述问题，线性回归算法的基本流程如算法 1 所示。

算法 1 线性回归算法 LinearRegression(D)

输入： 训练集 $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_m, y_m)\}$.

1: 令 $\boldsymbol{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} & 1 \\ x_{21} & x_{22} & \cdots & x_{2n} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} & 1 \end{bmatrix}$

其中 x_{ij} 是第 i 个样本的第 j 个属性

2: 令 $\boldsymbol{y} = [y_1, y_2, \cdots, y_m]^{\mathrm{T}}$

3: 计算 $\hat{\boldsymbol{w}}^* = (\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{y}$，其中 $\hat{\boldsymbol{w}}^* = [\boldsymbol{w}^*, b^*]^{\mathrm{T}}$

4: 令 $\hat{\boldsymbol{x}}_i = [\boldsymbol{x}_i, 1]^{\mathrm{T}}$

5: 线性回归模型 $f(\hat{\boldsymbol{x}}_i) = \hat{\boldsymbol{x}}_i^{\mathrm{T}}\hat{\boldsymbol{w}}^*$

输出： 线性回归模型 $f(\hat{\boldsymbol{x}}_i)$.

2. 对数几率回归算法

对数几率回归 (logistic regression) **算法**是一种分类算法，主要用于解决二分类问题。若将二分类问题的输出标记为 $y \in \{0, 1\}$，则对数几率回归算法的核心思想是：通过应用**对数几率函数** (logistic function)，将线性回归的预测值 $z = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b$ 转换为 0/1 值, 再将其作为预测结果 y。

若只考虑将实值 z 转换为 0/1 值，则最理想的是**单位阶跃函数** (unit-step function)

$$y = \begin{cases} 0 & , z < 0 \\ 0.5 & , z = 0 \\ 1 & , z > 0 \end{cases}$$

对于单位阶跃函数，预测值 $z > 0$ 时，判为正例；$z < 0$ 时，则判为反例；$z = 0$ 时，分类可自由选择。然而，单位阶跃函数的不连续性可能会在后续的求导过程中引发问题。因此，需要一个既能近似单位阶跃函数，又能满足单调可微性质的替代函数。**对数几率函数**正是这样一种广泛使用的替代函数：

$$y = \frac{1}{1 + \mathrm{e}^{-z}}$$

类似地，对于对数几率函数，当预测值 $z > 0$ 时，判为正例；当 $z < 0$ 时，判为反例；当 $z = 0$ 时，y 的值可自由决定。

将线性回归预测值 z 转换为对数几率回归预测值 y 之后，需要求解 $z = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b$ 中的 $\boldsymbol{w}$ 和 b。由于此时数据集 D 中的 y 不是连续值，而是离散的 0/1 值，因此不能直接使用线性回归算法的最小二乘法进行求解。针对这一离散型随机变量的参数估计问题，可以采用**极大似然法** (maximum likelihood method) 构建优化模型，并采用**梯度下降法** (gradient descent

method) 或者**牛顿法** (Newton method) 等数值优化算法求解出最优解。基于梯度下降法的对数几率回归算法基本流程如算法 2 所示。

算法 2 对数几率回归算法 Logistic Regression(D, α,T)

输入： 训练数据集 $D=\{(\boldsymbol{x}_1,y_1),(\boldsymbol{x}_2,y_2),\cdots,(\boldsymbol{x}_m,y_m)\}$;
学习率 α; 迭代次数 T.

令 $\hat{\boldsymbol{x}}_i=[\boldsymbol{x}_i,1]^{\mathrm{T}}=[x_{i1},x_{i2},\cdots,x_{in},1]^{\mathrm{T}}$，其中，$x_{ij}$ 是第 i 个样本的第 j 个属性
令 $\boldsymbol{X}=[\hat{\boldsymbol{x}}_1,\hat{\boldsymbol{x}}_2,\cdots,\hat{\boldsymbol{x}}_m]$
令 $\boldsymbol{y}=[y_1,y_2,\cdots,y_m]^{\mathrm{T}}$
初始化模型参数 $\hat{\boldsymbol{w}}$，其中，$\hat{\boldsymbol{w}}=[\boldsymbol{w},b]^{\mathrm{T}}=[w_1,w_2,\cdots,w_n,b]^{\mathrm{T}}$
for $t=1,2,\cdots,T$ **do**
    计算线性组合: $\boldsymbol{z}=[z_1,z_2,\cdots,z_m]^{\mathrm{T}}=\boldsymbol{X}^{\mathrm{T}}\hat{\boldsymbol{w}}$
    计算预测值: $\boldsymbol{h}=[\mathrm{LF}(z_1),\mathrm{LF}(z_2),\cdots,\mathrm{LF}(z_m)]$，其中，LF() 代表对数几率函数
    计算梯度：$\mathbf{grad}=\boldsymbol{X}^{\mathrm{T}}(\boldsymbol{h}-\boldsymbol{y})/m$
    更新模型参数: $\hat{\boldsymbol{w}}=\hat{\boldsymbol{w}}-\alpha*\mathbf{grad}$
end for
对数几率回归模型 $f(\hat{\boldsymbol{x}}_i)$ =LF($\hat{\boldsymbol{w}}^{\mathrm{T}}\hat{\boldsymbol{x}}_i$)

输出： 对数几率回归模型 $f(\hat{\boldsymbol{x}}_i)$.

3. 决策树算法

决策树 (decision tree) **算法**既可以用于分类任务，也可以用于回归任务。下面介绍**用于分类任务的决策树算法**。如图 3.2 所示，分类决策树模型由**内部节点** (internal node)、**叶节点** (leaf node) 和**有向边** (directed edge) 组成。

内部节点：表示一个属性。

叶节点：表示一个类别。

有向边：表示满足某一种属性值。

决策树的学习过程通常是一个递归的过程，该过程选择最优的划分属性，并根据该属性对训练数据集进行分割，目的是对每个子数据集进行最佳的分类。

假设给定训练数据集

$$D=\{(\boldsymbol{x}_1,y_1),(\boldsymbol{x}_2,y_2),\cdots,(\boldsymbol{x}_m,y_m)\}$$

该训练数据集对应的属性集为

$$A=\{a_1,a_2,\cdots,a_n\}$$

首先，将训练集 D 中的所有数据放在根节点处，并从属性集 A 中选取一个最优的划分属性作为根节点。接着，根据这个最优划分属性，将训练数据集分割成子集，在当前条件下为每个子集提供最佳的分类。该属性的不同取值作为一条有向边。

如果某个子集已经可以被基本正确分类，则构建叶节点并标记类别，然后将这些子集分配到对应的叶节点；如果还有子集不能被基本正确地分类，则在这些子集对应的新属性集中选择新的最优划分属性，继续划分子集，并构建新的节点和有向边。如此递归进行下去，直至所有训练数据均被划分至叶节点，或者无法选出合适的属性为止。决策树算法的基本流程如算法 3 所示。

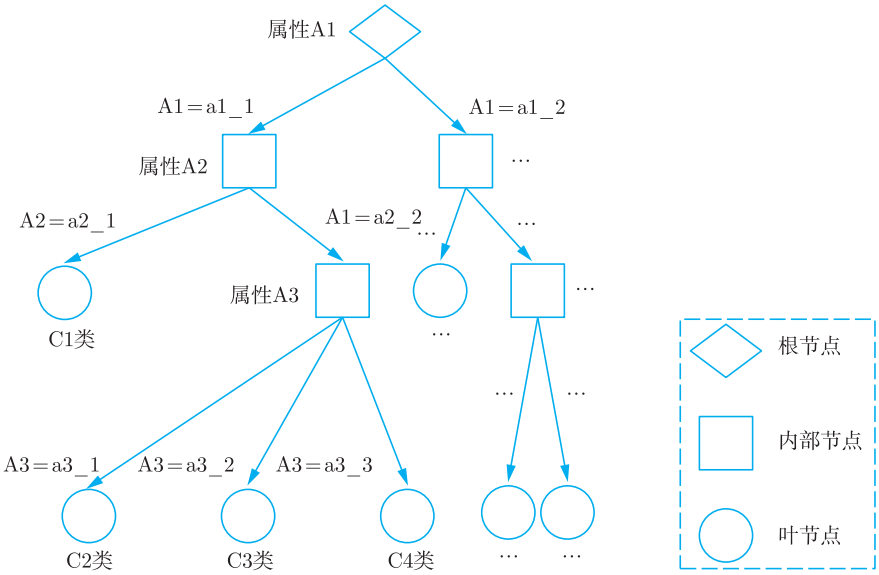

图 3.2　决策树模型

算法 3　决策树算法 DecisionTree(D,A)

输入： 训练数据集 $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_m, y_m)\}$;
　　属性集 $A = \{a_1, a_2, \cdots, a_n\}$.

生成一个节点
if D 的全体样本均属于类别 C **then**
　将节点标记为类别为 C 的叶节点
　return
end if
if $A = \emptyset$ **or** D 中样本在 A 上的取值均相同 **then**
　将节点标记为叶节点，类别标记为 D 中样本数最多的类别
　return
end if
从 A 中选取最优划分属性 a^*
for 每个取值 $a_i^* \in a^*$ **do**
　生成一条有向边，并且令 D_i 为训练数据集 D 中在 a^* 上取值为 a_i^* 的样本集合
　if $D_i = \emptyset$ **then**
　　将节点标记为叶节点，类别标记为 D 中样本数最多的类别
　　return
　else
　　以 DecisionTree($D_i, A\backslash\{a^*\}$) 为内部节点
　end if
end for

输出： 以该节点为根节点的一棵决策树.

4. 支持向量机算法

支持向量机 (Support Vector Machine，SVM) **算法**既可以用于分类任务，也可以用于回归任务。下面介绍**线性可分支持向量机算法**。给定训练数据集 $D=\{(\boldsymbol{x}_1,y_1),(\boldsymbol{x}_2,y_2),\cdots,(\boldsymbol{x}_m,y_m)\}$，其中，$y_i\in\{-1,+1\}$。假设划分超平面 $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b=0$ 能正确地将训练样本分类，则有

$$\begin{cases}\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i+b>0, y_i=+1\\ \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i+b<0, y_i=-1\end{cases}$$

通过缩放变换，可以得到

$$\begin{cases}\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i+b\geqslant+1, y_i=+1\\ \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i+b\leqslant-1, y_i=-1\end{cases}\tag{3.1}$$

然后，如图 3.3 所示，样本空间中任意点 $\boldsymbol{x}$ 到划分超平面 $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b=0$ 的距离为

$$r=\frac{\left|\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b\right|}{\|\boldsymbol{w}\|}$$

而**支持向量** (support vector) 被定义为距离超平面最近的训练样本点，它们使得式 (3.1) 中的等号成立。两个不同类别的支持向量到超平面的距离之和被表示为图 3.3 中的 γ，它被称为**间隔** (margin)。支持向量机的核心思想就是在满足划分约束的情况下，找到具有**最大间隔** (maximum margin) 的划分超平面，即

$$\begin{aligned}&\max_{\boldsymbol{w},b}\frac{2}{\|\boldsymbol{w}\|}\\ &\text{s.t. } y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i+b)\geqslant1,\quad\forall i=1,2,\cdots,m\end{aligned}\tag{3.2}$$

此外，将式 (3.2) 等价变换后可得到**支持向量机的基本型**，即

$$\begin{aligned}&\min_{\boldsymbol{w},b}\frac{1}{2}\|\boldsymbol{w}\|^2\\ &\text{s.t. } y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i+b)\geqslant1,\quad\forall i=1,2,\cdots,m\end{aligned}$$

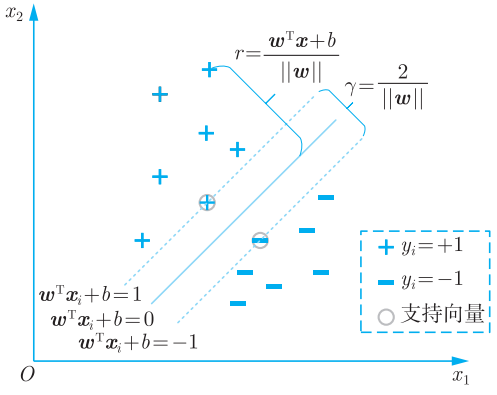

图 3.3　支持向量机原理图

支持向量机算法的基本流程如算法 4 所示。

算法 4 支持向量机算法 SupportVectorMachine(D)

输入： 训练数据集 $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_m, y_m)\}$.

1: 初始化模型参数 $\boldsymbol{w}$ 和 b

2: 构建优化问题：

$$\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2$$

$$\text{s.t. } y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) \geqslant 1, \quad \forall i = 1, 2, \cdots, m.$$

3: 利用**序列最小优化算法** (Sequential Minimal Optimization, SMO) 或**梯度下降法**等优化算法求解 $\boldsymbol{w}$ 和 b

4: 构建支持向量机模型 $f(\boldsymbol{x}_i) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b$

输出： 支持向量机模型 $f(\boldsymbol{x}_i)$.

5. 监督学习总结

以上 4 种经典的监督学习算法拥有各自的特点、适用范围和优缺点，总结如表 3.1 所示。

表 3.1 监督学习算法对比表

算法	特点	适用范围	优点	缺点
线性回归算法	连续数值预测 线性模型	回归问题 线性关系	易于理解 计算效率高	适用范围受限 对异常值敏感
对数几率回归算法	二分类问题 线性模型 对数几率函数	分类问题 线性关系 近似线性关系	易于理解 计算效率高	适用范围受限 对异常值敏感
决策树算法	树状结构 递归分割	分类和回归问题 线性和非线性关系	适用范围广 可解释性强	容易过拟合 对噪声和小样本敏感
支持向量机算法	寻找最大间隔超平面	分类和回归问题 线性和非线性关系	适用范围广 泛化能力强	计算成本高 对大规模数据集敏感

3.2.3 无监督学习

无监督学习的目标是从未标记的训练数据中学习数据的内在结构、模式或关系。**聚类** (clustering) 和**降维** (dimensionality reduction) 是无监督学习的两个典型任务。

聚类：将样本集合中相似的样本分配到同一类别，不相似的样本分配到不同类别，一个类别被称作一个“簇”。根据样本是否只能属于一个类别，可以将聚类分为**硬聚类** (hard clustering) 和**软聚类** (soft clustering)。硬聚类意味着每个样本只能属于一个类别，而软聚类则允许一个样本属于多个类别。聚类示例如图 3.4 所示。

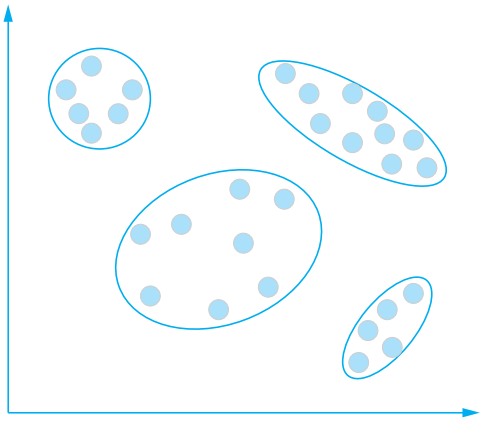

图 3.4　聚类示例

降维：将训练数据中的样本从高维空间转换到低维空间，在此过程中，要尽可能地减少样本信息的损失。降维示例如图 3.5 所示。

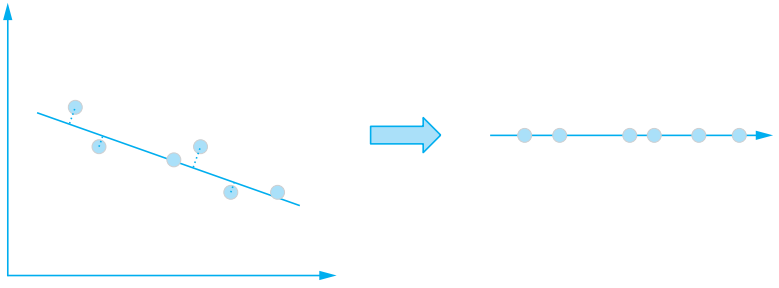

图 3.5　降维示例

下面是一些经典的无监督学习算法。

1. k **均值算法**

k **均值** (k-means) **算法**是一种聚类算法。给定样本集 $D=\{\boldsymbol{x}_1,\boldsymbol{x}_2,\cdots,\boldsymbol{x}_m\}$ 和聚类簇数 k，k 均值算法的目标是将数据分为 k 个簇，并针对聚类所得簇划分 $C=\{C_1,C_2,\cdots,C_k\}$ 最小化平方误差

$$E=\sum_{i=1}^{k}\sum_{\boldsymbol{x}\in C_i}\|\boldsymbol{x}-\boldsymbol{\mu}_i\|^2$$

其中，$\boldsymbol{\mu}_i=\dfrac{1}{|C_i|}\sum\limits_{\boldsymbol{x}\in C_i}\boldsymbol{x}$ 是簇 C_i 的均值向量。

平方误差 E 的值越小，说明样本在簇内围绕簇均值向量 $\boldsymbol{\mu}=[\boldsymbol{\mu}_1,\boldsymbol{\mu}_2,\cdots,\boldsymbol{\mu}_k]$ 的分布越紧密，即簇内样本相似度越高。这正是 k 均值算法的核心思想。k 均值算法的基本流程如算法 5 所示。

算法 5 k 均值算法 KMeans(D,k)

输入： 样本集 $D=\{\boldsymbol{x}_1,\boldsymbol{x}_2,\cdots,\boldsymbol{x}_m\}$; 聚类簇数 k.

随机选择 k 个样本作为初始均值向量 $\boldsymbol{\mu}_1,\boldsymbol{\mu}_2,\cdots,\boldsymbol{\mu}_k$
while 均值向量仍存在更新 **do**
  初始化 $C=\{C_1,C_2,\cdots,C_k\}=\{\emptyset,\emptyset,\cdots,\emptyset\}$
  for 每个训练样本 $\boldsymbol{x}_i\in D$ **do**
    计算样本 $\boldsymbol{x}_i$ 与各均值向量 $\boldsymbol{\mu}_j(1\leqslant j\leqslant k)$ 的距离：$d_{ij}=\|\boldsymbol{x}_i-\boldsymbol{\mu}_j\|$
    将 $\boldsymbol{x}_i$ 的簇标记设置为距离最近的均值向量：$\lambda_i=\arg\min_{j\in\{1,2,\cdots,k\}}d_{ij}$
    将样本 $\boldsymbol{x}_i$ 归入对应的簇：$C_{\lambda_i}=C_{\lambda_i}\bigcup\{\boldsymbol{x}_i\}$
  end for
  for $i=1,2,\cdots,k$ **do**
    计算新的均值向量：$\boldsymbol{\mu}_i{}'=\dfrac{1}{|C_i|}\sum\limits_{\boldsymbol{x}\in C_i}\boldsymbol{x}$
    if $\boldsymbol{\mu}_i{}'\neq\boldsymbol{\mu}_i$ **then**
      将 $\boldsymbol{\mu}_i$ 更新为 $\boldsymbol{\mu}_i{}'$
    else
      保持 $\boldsymbol{\mu}_i$ 不变
    end if
  end for
end while

输出： 簇划分 $C=\{C_1,C_2,\cdots,C_k\}$.

2. 主成分分析算法

主成分分析 (Principal Component Analysis，PCA) **算法**是一种降维方法。对于给定训练数据集 $D=\{\boldsymbol{x}_1,\boldsymbol{x}_2,\cdots,\boldsymbol{x}_m\}$，假定其中的样本已经进行了**标准化** (standardization)：

$$(\boldsymbol{x}_i-\frac{1}{m}\sum_{i=1}^{m}\boldsymbol{x}_i)\Big/\sqrt{\frac{1}{m}\sum_{i=1}^{m}(\boldsymbol{x}_i-\frac{1}{m}\sum_{i=1}^{m}\boldsymbol{x}_i)^2}\to\boldsymbol{x}_i$$

接下来对样本进行正交变换，将其转换为由几个线性无关的新变量表示的样本。通过考虑方差大小，可以将新变量依次划分为第一主成分、第二主成分等。然后，利用更少数量的主成分近似表示原始数据，实现数据的降维。这就是主成分分析算法的核心思想。具体原理如图 3.6 所示。

假设数据样本已经进行了标准化处理，并且它们分布在图 3.6 中的椭圆之内。起初，样本由 x_1 和 x_2 表示，通过正交变换后，样本由 y_1 和 y_2 表示。在这里，y_1 轴的方向具有最大方差，y_2 轴的方向方差次之。因此，y_1 轴为第一主成分，而 y_2 轴为第二主成分。若主成分分析仅在一维空间中进行，则只选择 y_1 轴。这等价于将数据投影在 y_1 轴上，并用 y_1 轴上的投影点来表示样本，从而实现从二维空间到一维空间的降维。主成分分析算法的基本流程如算法 6 所示。