机器学习综述（Machine Learing Overview）

一、机器学习的发展

机器学习（Machine Learning）的各个阶段发展历程列表如下。

时间段	机器学习理论	代表性成果
二十世纪五十年代初	人工智能研究处于推理期	A. Newell和H. Simon的“逻辑理论家”（Logic Theorist）程序证明了数学原理，以及此后的“通用问题求解”（General Problem Solving）程序。
已出现机器学习的相关研究	1952年，阿瑟·萨缪尔（Arthur Samuel）在IBM公司研制了一个西洋跳棋程序，这是人工智能下棋问题的由来。	
二十世纪五十年代中后期	开始出现基于神经网络的“连接主义”（Connectionism）学习	F. Rosenblatt提出了感知机（Perceptron），但该感知机只能处理线性分类问题，处理不了“异或”逻辑。还有B. Widrow提出的Adaline。
二十世纪六七十年代	基于逻辑表示的“符号主义”（Symbolism）学习技术蓬勃发展	P. Winston的结构学习系统，R. S. Michalski的基于逻辑的归纳学习系统，以及E. B. Hunt的概念学习系统。
以决策理论为基础的学习技术		
强化学习技术	N. J. Nilson的“学习机器”。	
统计学习理论的一些奠基性成果	支持向量，VC维，结构风险最小化原则。	
二十世纪八十年代至九十年代中期	机械学习（死记硬背式学习） 示教学习（从指令中学习） 类比学习（通过观察和发现学习） 归纳学习（从样例中学习）	学习方式分类
从样例中学习的主流技术之一：（1）符号主义学习	（1）决策树（decision tree）。 （2）归纳逻辑程序设计（Inductive Logic Programming, ILP）具有很强的知识表示能力，可以较容易地表达出复杂的数据关系，但会导致学习过程面临的假设空间太	

时间段	机器学习理论	代表性成果
（2）基于逻辑的学习	大，复杂度极高，因此，问题规模稍大就难以有效地进行学习。	
从样例中学习的主流技术之二：基于神经网络的连接主义学习	1983年，J. J. Hopfield利用神经网络求解“流动推销员问题”这个NP难题。1986年，D. E. Rumelhart等人重新发明了BP算法，BP算法一直是被应用得最广泛的机器学习算法之一。	
二十世纪八十年代是机器学习成为一个独立的学科领域，各种机器学习技术百花初绽的时期	连接主义学习的最大局限是“试错性”，学习过程涉及大量参数，而参数的设置缺乏理论指导，主要靠手工“调参”，参数调节失之毫厘，学习结果可能谬以千里。	
二十世纪九十年代中期	统计学习（Statistical Learning）	支持向量机（Support Vector Machine，SVM），核方法（Kernel Methods）。
二十一世纪初至今	深度学习（Deep Learning）	深度学习兴起的原因有二：数据量大，机器计算能力强。

二、机器学习的分类

机器学习大致可以分为四种类型：

1. **监督学习**
2. **无监督学习**
3. **半监督学习**
4. **强化学习**

2.1 监督学习

从**给定的训练数据集**中学习出一个函数（模型参数），当新的数据到来时，可以**根据这个函数预测结果。监督学习的训练集要求包括输入输出，也可以说是特征和目标。**训练集中的目标是由人标注的。**监督学习就是最常见的分类（注意和聚类区分）问题**，通过已有的训练样本（即**已知数据及其对应的输出**）去训练得到一个最优模型（**这个模型属于某个函数的集合**，**最优表示某个评价准则下是最佳的**），再利用这个模型将所有的输入映射为相应的输出，**对输出进行简单的判断从而实现分类的目的。**也就具有了对未知数据分类的能力。监督学习的目标往往是让计算机去学习我们已经创建好的分类系统（模型）。

监督学习是训练神经网络和决策树的常见技术。这两种技术高度依赖事先确定的分类系统给出的信息，对于**神经网络，分类系统利用信息判断网络的错误，然后不断调整网络参数。**对于**决策**

树，分类系统用它来判断哪些属性提供了最多的信息。

总结：监督学习需要有人为提供的训练数据集，其中包括特征和目标，监督学习常用语分类和预测问题。

2.2 无监督学习

与监督学习不同，无监督学习并非告诉计算机怎么做，而是让计算机自己学习怎么做。

无监督学习输入的数据没有标记，并且输出没有明确的结果，样本分类也是不确定的，需要通过样本之间的相似性对样本集进行分类（聚类，clustering）。试图使 **类内差距最小化，类间差距最大化** 通俗点将就是实际应用中，不少情况下**无法预先知道样本的标签**，也就是说没有训练样本对应的类别，因而只能从原先没有样本标签的样本集开始学习分类器设计。

典型的无监督学习方法是奖励和惩罚机制，让计算机自行决定如何做，在指导agent时不为其指定明确的分类，而是在成功时采用某种形式的激励记录。这种问题通常存在于决策机制的框架里，因为**它的目标不是为了产生一个分类系统，而是做出最大回报的决定**，这种思路很好的概括了现实世界，agent可以对正确的行为做出激励，而对错误行为做出惩罚。

2.3 半监督学习

半监督学习是监督学习和非监督学习的结合，其**在训练阶段使用的是未标记的数据和已标记的数据，不仅要学习属性之间的结构关系，也要输出分类模型进行预测。**

半监督学习旨在通过少量的有标签数据和大量的无标签数据来对模型进行训练，通过**充分利用无标签数据来对提升模型的泛化能力，同时通过有标签数据来引导模型确定正确的决策边界。**

2.4 强化学习

强化学习（Reinforcement Learning, RL），又称再励学习、评价学习或增强学习，是机器学习的范式和方法论之一，用于**描述和解决智能体（agent）在与环境的交互过程中通过学习策略以达成回报最大化或实现特定目标的问题.**

强化学习中的**核心概念**:
- **代理（Agent）**：强化学习中的决策者，它通过观察环境状态并采取行动来学习最优策略。
- **环境（Environment）**：代理所存在的外部条件，能够对代理的行动提供反馈（奖励或惩罚）。
- **状态（State）**：描述环境的一种方式，通常包含代理做出决策所需的全部信息。
- **动作（Action）**：代理在给定状态下可以采取的行为。
- **奖励（Reward）**：环境对代理行动的反馈，通常是标量值，表示该行动对达到目标的好坏程度。

三、机器学习模型

机器学习的基本构成有如下三个方面：

机器学习 = 数据（data）+ 模型（model）+ 优化方法（optimal strategy）

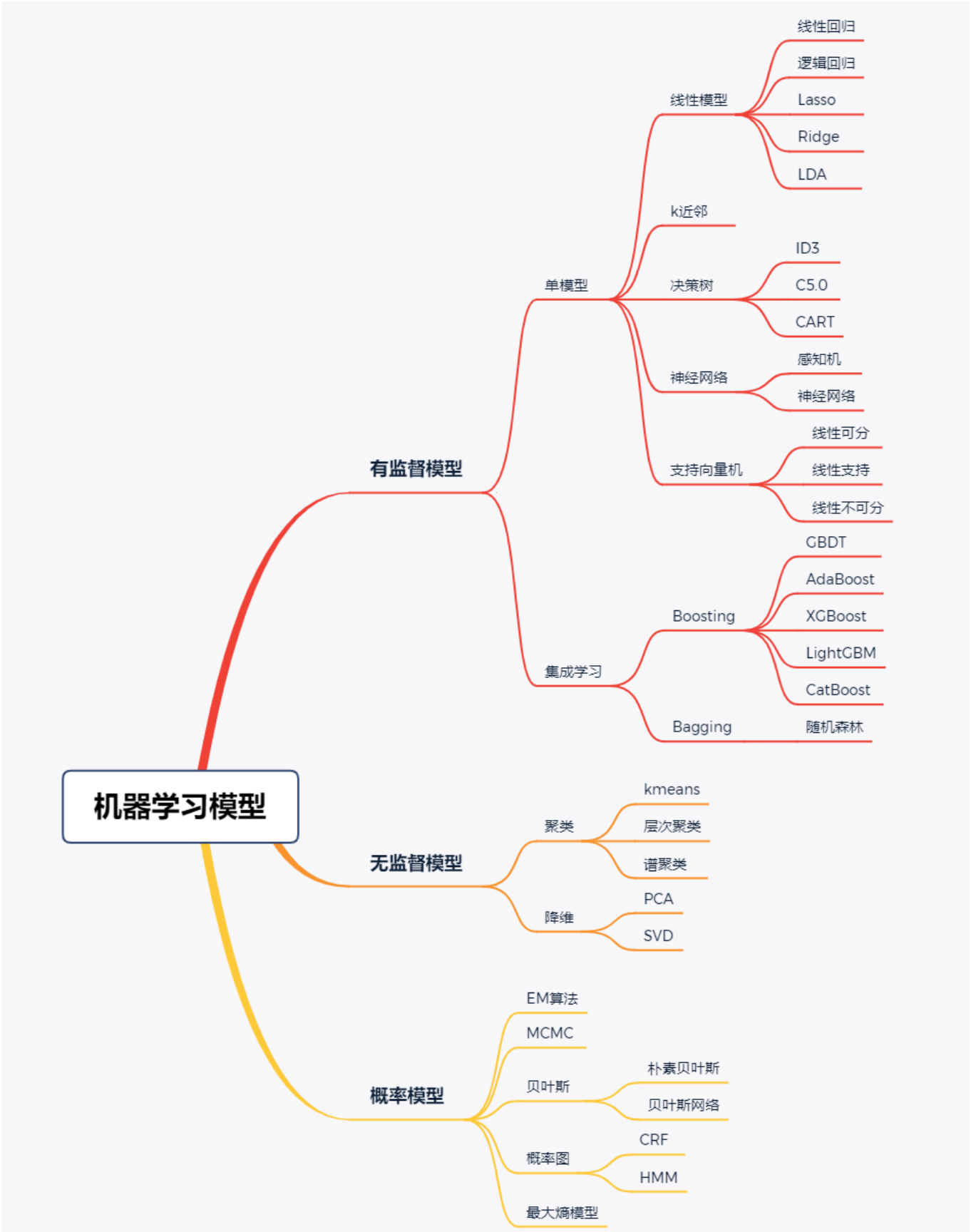

图1 机器学习模型图

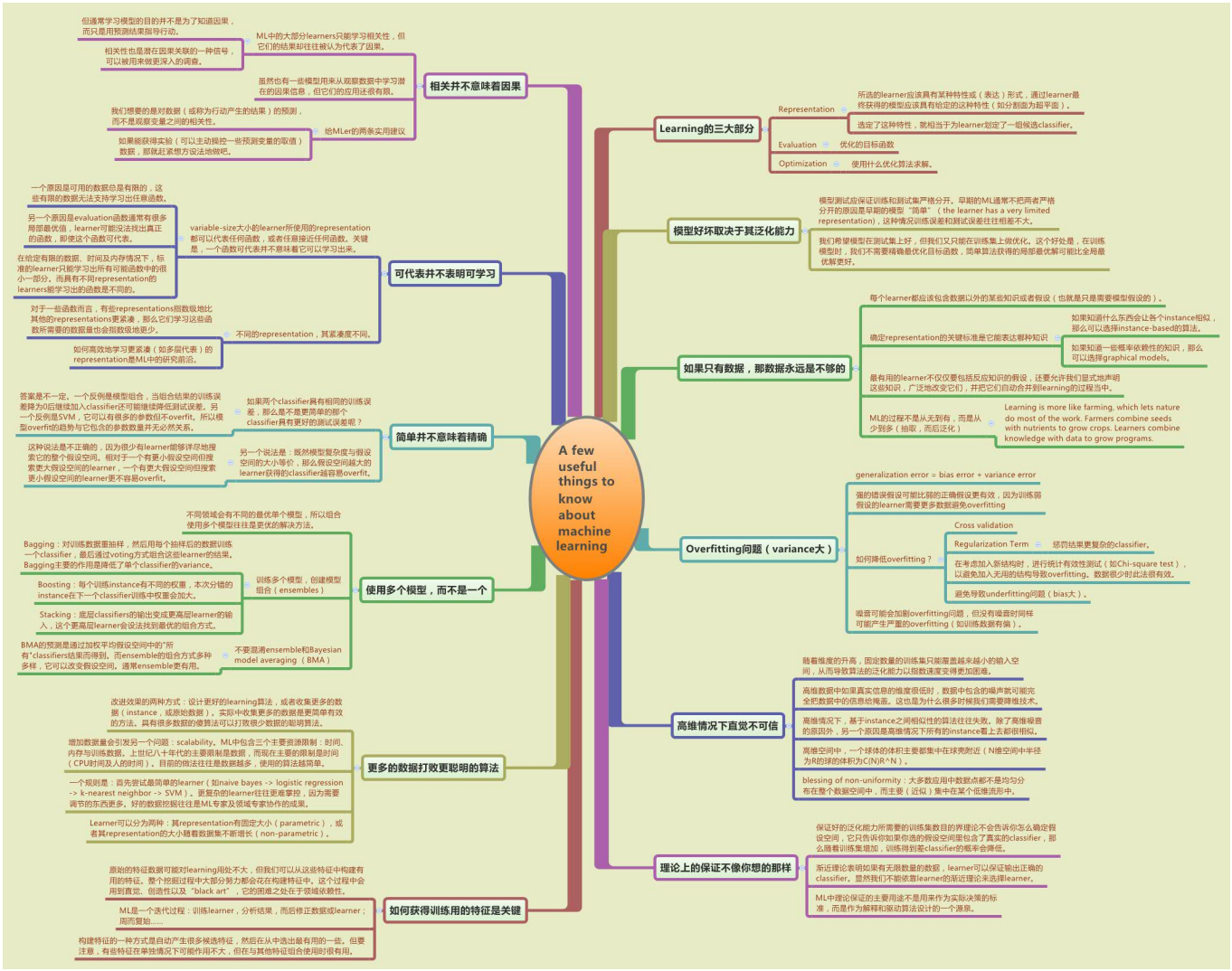

图2 机器学习注意事项

四、常见机器学习算法

4.1 线性算法（Linear Algorithms）

线性算法主要指基于**线性关系进行分析或者预测**的方法，这类算法的核心思想是**通过建立线性模型来描述自变量（特征）与因变量（目标变量）之间的关系。**

> **线性关系**：
> 线性关系指两个变量之间存在直线的关系，其中一个变量（自变量）的变化会导致另一个变量（因变量）以**恒定的速率**变化，线性关系可以通过直线方程来表示，其中的常数项（截距）和斜率（系数）描述了变量之间的相互关系。

> **非线性关系**
> 简单来说，两个变量之间的关系是一次函数的关系，即图像是直线的，就称为线性关系，而非线性关系就是两个变量之间的关系不是一次函数的关系，图像不是直线的。

线性算法主要包括四个：

1. 线性回归算法（Linear Regression）
2. 套索回归（Lasso Regression）
3. 岭回归（Ridge Regression）
4. 逻辑回归算法（Logistic Regression）

4.1.1 线性回归算法

线性回归是一种基本的**预测建模技术**，他的目标是在自变量和因变量之间找到一条最佳直线，使得这条直线上的预测值与真实值之间的误差最小。

线性回归的基本假设：

1. **线性关系**：进行线性回归建模的前提，自变量和因变量之间存在线性关系。
2. **独立同分布**：观测值是独立同分布的。
3. **无多重共线性**:即自变量之间不存在多重共线性（自变量不应高度相关）
4. **误差项的正态性**：误差项（即，真实值与预测值之间的差异）服从正态分布。
5. **同方差性**：误差项的方差是恒定的，不随自变量的变化而变化。

> **独立同分布**：
> 独立同分布是概率统计论中的一个重要概念，指随机过程中的变量取值都是随机的，这些随机变量服从同一分布并且取值相互独立不影响，则称为独立同分布。（**同一分布、取值独立。**）

线性回归的基本模型：
线性回归的基本模型可以表示为：
$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n$$

其中：

- y 是因变量（响应变量、目标）
- $x_1, x_2, \ldots, x_n$ 是自变量（预测变量、特征）
- β_0 是截距项
- $\beta_1, \beta_2, \ldots, \beta_n$ 是回归系数（也称为斜率），它们表示每个自变量对因变量的影响程度

线性回归的意义：
线性回归是解决回归问题中最简单也是最常用的一种算法，通过建立自变量与因变量之间的线性关系模型可以通过给定的自变量值预测因变量值。
也就是说，线性回归用于解决预测和回归的问题。

Pytorch实现：

```python
import torch
import torch.nn as nn
import torch.optim as optim
import numpy as np
import matplotlib.pyplot as plt

# 设定随机数种子以便结果可复现
torch.manual_seed(0)

# 生成模拟数据
np.random.seed(0)
x_train = np.random.rand(100, 1) * 10  # 100个样本，特征值在0到10之间
y_train = 2 * x_train + 3 + np.random.randn(100, 1) * 0.3  # 通过自变量产生的标签值

# 将numpy数组转换为PyTorch张量
x_train_tensor = torch.from_numpy(x_train).float()
y_train_tensor = torch.from_numpy(y_train).float()


# 定义线性回归模型
class LinearRegression(nn.Module):
    def __init__(self, input_dim, output_dim):
        super(LinearRegression, self).__init__()
        self.linear = nn.Linear(input_dim, output_dim)

    def forward(self, x): # 前向传播
        out = self.linear(x)
        return out



# 初始化模型
input_dim = 1    # 线性回归算法的输入输出维度都是1
output_dim = 1
model = LinearRegression(input_dim, output_dim)

criterion = nn.MSELoss()    # 计算均方误差的损失函数
learning_rate = 0.01   # 学习率
# 创建SGD优化器用于自动更新参数，其中model.parameters包含模型中所有可训练参数的迭代器，lr是学习率的缩写，指定参数更新的步长。
optimizer = optim.SGD(model.parameters(), lr=learning_rate)

# 训练模型
num_epochs = 100   # 指定参数更新次数
for epoch in range(num_epochs):
```

```
    # 前向传播
    outputs = model(x_train_tensor)
    # 通过输出和标签值计算损失值
    loss = criterion(outputs, y_train_tensor)

    # 反向传播和优化
    optimizer.zero_grad()  # 清零梯度缓存
    loss.backward()  # 反向传播，计算梯度
    optimizer.step()  # 使用梯度下降更新权重

    if (epoch + 1) % 10 == 0:
        print(f'Epoch [{epoch + 1}/{num_epochs}], Loss: {loss.item()}')



# 获取模型的参数
print('Model parameters:')
for name, param in model.named_parameters():
    print(f"    Name: {name}, Value: {param.data.numpy()}")
```

4.1.2 套索回归（Lasso回归）

Lasso回归是什么：

Lasso回归是一种是一种广泛使用的**回归分析方法**，尤其在处理具有多重共线性或高维数据时表现出色。

Lasso回归的原理：

Lasso回归通过在目标函数中引入L1正则化项（即**变量系数的绝对值之和**），对系数的绝对值进行惩罚，使得一些不重要的系数值变为零。
Lasso回归的数学模型可以表示为：

$$\min_{\beta} \left\{ \frac{1}{2n} \sum_{i=1}^{n} (y_i - x_i^T \beta)^2 + \lambda \|\beta\|_1 \right\}$$

其中，y_i 是响应变量，x_i 是预测变量，β 是系数向量，$\|\beta\|_1$ 表示系数向量的L1范数（即系数的绝对值之和），λ 是正则化参数，控制着对系数的惩罚强度。

在Lasso回归中，λ是一个关键的参数，其值的大小直接影响到最终模型的表现。当λ为0时，Lasso回归就退化为普通的最小二乘回归。随着λ值的增加，越来越多的系数被压缩为零，这有助于特征选择和降低模型复杂度。然而，如果λ过大，它可能会导致模型过于简单，从而影响模型的预测能力。因此，选择一个合适的λ值是实现最佳模型性能的关键。

总的来说，Lasso回归就是在目标函数中引入了系数绝对值作为惩罚参数来达到降低模型复杂度和特征选择的目的。

Lasso回归的作用：

Lasso回归主要有两个重要的作用：**变量选择和模型简化。**
首先是变量选择，Lasso回归通过变量系数绝对值作为惩罚来使不重要的系数值变为零，从而只保留重要的系数项即变量。
另一方面，通过选择重要的变量和压缩系数能够有效的降低模型的复杂度。
同时通过保留重要系数项能够显示出数据集中的主要特征，达到特征选择的目的。

4.1.3 岭回归（Rideg Regression）