实验报告：

实证分析中的稳健性检验：

基于机器学习方法

（2024-2025 学年第一学期）

学　　院　　经济学院

课程名称　R 语言编程基础与金融数据分析

任课教师　　王皓

小　　组　　08

代码地址：　https://sr6688.github.io/

小组报告

目录

研究背景 4
作为监督学习的线性回归模型....4
线性回归是最优预测吗？ 4
基准模型说明....9
数据获取与清洗结果....11
基准回归结果....13
研究 1：线性回归参数估计的误差修正 14
Lasso 回归估计结果 18
弹性网回归估计结果....19
研究 2：使用非参数方法对模型参数估计进行优化 20
分类树....20
随机森林....24
自适应提升....28
分类树回归估计结果....31
随机森林回归估计结果....32
自适应提升法回归估计结果....33
研究 3：使用非参数方法对模型参数估计进行优化 34
向量机....34
感知机....42
神经网络....43
支持向量机回归估计结果....48
人工神经网络回归估计结果....48
研究 4：使用非监督学习方法对模型聚类进行优化 49
主成分分析....49
聚类分析....53
主成分分析估计结果....55

图表索引

图表 1 最小二乘法的正交性.....6
图表 2 多项式对正弦函数的拟合.....7
图表 3 偏差与方差的权衡.....9
图表 4 变量定义.....10
图表 5 本文数据集结构.....13
图表 6 基准回归结果.....13
图表 7 岭回归估计量与 OLS 估计量在方差与偏差权衡下的选择.....14
图表 8 弹性网约束条件的几何解释.....18
图表 9 Lasso 回归估计结果.....19
图表 10 弹性网回归估计结果.....20
图表 11 作为分段常值函数的决策树.....22
图表 12 变量重要性图（Variable Importance Plot）.....26
图表 13 偏依赖图（Partial Dependence Plot）.....28
图表 14 分类树回归估计结果.....32
图表 15 随机森林回归估计结果.....33
图表 16 自适应提升法回归估计结果.....34
图表 17 分离超平面.....35
图表 18 观测数据到分离超平面的符号距离.....36
图表 19 特征变换.....39
图表 20 人工神经网络的神经元示意图.....42
图表 21 多个输出结果的感知机.....44
图表 22 多层感知机.....45
图表 23 双隐藏层神经网络.....46
图表 24 支持向量机回归估计结果.....48
图表 25 主成分分析示意图.....50
图表 26 陡坡图（scree plot）.....52
图表 27 累积 PVE 图（Cumulative PVE）.....53
图表 28 主成分分析结果.....56

研究背景

作为监督学习的线性回归模型

我们试图通过训练数据 $\{\boldsymbol{x}_i, y_i\}_{i=1}^n$ 来学得一个函数 $f(\boldsymbol{x}_i)$，并以 $f(\boldsymbol{x}_i)$ 预测 y_i。当然, 这种预测通常不可能完全准确。对于观测数据的生成过程（data generating process，简记 DGP），可以作很一般的假设：

$$y_i = f(\boldsymbol{x}_i) + \varepsilon_i$$

其中， ε_i 为“随机扰动项”（stochastic disturbance），或“误差项”（error term）。一般将 $f(\boldsymbol{x}_i)$视为“信号”（signal），即来自 $\boldsymbol{x}_i$ 对于 y_i 的 “系统信息”（systematic information）；而把 ε_i 看作随机扰动的“噪声”（noise）。

如何学得 $f(\boldsymbol{x}_i)$ ？大致可分为两种方法, 即“非参数方法”（nonparametric approach） 以及“参数方法”（parametric approach）。如果不对 $f(x_i)$ 的函数形式（functional form） 作任何假设，则为非参数方法；因为既然不去假设函数形式，则自然没有待估计的参数。非参数方法包括了 K 近邻法、决策树、基于决策树的装袋法、随机森林与提升法等。反之，参数方法则对 $f(\boldsymbol{x}_i)$ 的函数形式作了具体的假设。此时，模型一般可写为

$$y_i = f(\boldsymbol{x}_i; \boldsymbol{\beta}) + \varepsilon_i$$

其中，$\boldsymbol{\beta}$ 为待估计的未知参数向量。而函数 $f(\boldsymbol{x}_i; \boldsymbol{\beta})$ 的具体形式既可以是非常简单的线性函数，也可以是复杂的神经网络模型。一般来说，参数方法与非参数方法适用于不同的场景。

线性回归是最优预测吗？

如果使用 $\boldsymbol{x}_i$ 预测 y_i, 是否存在一个最优的预测函数 $g(\boldsymbol{x}_i)$？这取决于最优预测的定义。对于回归问题，若使用 $g(\boldsymbol{x}_i)$ 预测连续的响应变量 y_i，则一

般地使用“均方误差”（mean squared error, 简记 MSE）作为对其预测优良程度的度量：

$$MSE = E[y - g(\boldsymbol{x})]^2$$

其中，期望算子 $\mathrm{E}(\cdot)$ 为同时对 $(\boldsymbol{x}, y)$ 求期望。下面将证明此最优预测函数为 $\mathrm{E}(y \mid \boldsymbol{x})$，也就是上文的 $f(\boldsymbol{x})$ 。

对于 OLS 的损失函数可写为

$$\min_{\tilde{\boldsymbol{\beta}}} \mathrm{SSR}(\tilde{\boldsymbol{\beta}}) = \sum_{i=1}^{n} e_i^2 = \boldsymbol{e}'\boldsymbol{e} = \| \boldsymbol{e} \|^2$$

这意味着，我们的目标就是要最小化残差向量的长度 $\| \boldsymbol{e} \|$ 。而残差向量 $\boldsymbol{e}$ 的长度，其实就是 $\boldsymbol{y}$ 到超平面 $\mathrm{col}(\boldsymbol{X})$ 的距离。显然, 只有当残差向量 $\boldsymbol{e}$ 垂直于 $\mathrm{col}(\boldsymbol{X})$ 时，$\boldsymbol{y}$ 到 $\mathrm{col}(\boldsymbol{X})$ 的距离才能最小化。因此，残差向量 $\boldsymbol{e}$ 要正交于 $\boldsymbol{X}$，故 $\boldsymbol{X}'\boldsymbol{e} = \boldsymbol{0}$, 这正是 OLS 的正规方程组。总之，OLS 的实质就是响应变量 $\boldsymbol{y}$ 向数据矩阵 $\boldsymbol{X}$ 的列空间所作的投影。将 OLS 解 $\hat{\boldsymbol{\beta}} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{y}$ 代入方程可得：

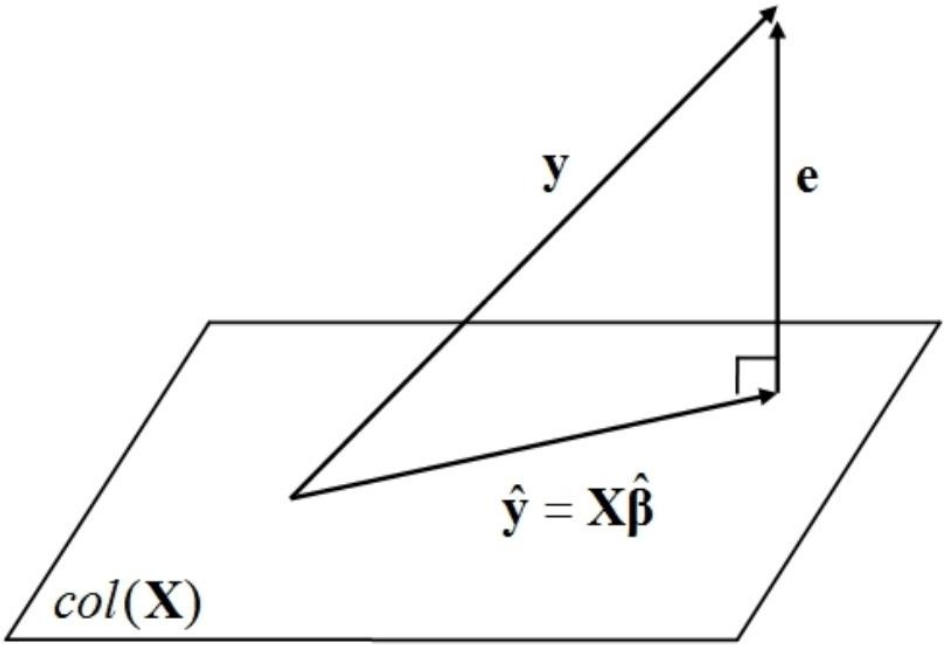
$$\hat{\boldsymbol{y}} = \boldsymbol{X}\hat{\boldsymbol{\beta}} = \underbrace{\boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'}_{=\boldsymbol{P}_{\boldsymbol{X}}} \boldsymbol{y} \equiv \boldsymbol{P}_{\boldsymbol{X}}\boldsymbol{y}$$

其中，$n \times n$ 矩阵 $\boldsymbol{P}_{\boldsymbol{X}} \equiv \boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'$ 称为 $\boldsymbol{X}$ 的“投影矩阵” (projection matrix)，其作用是将 $\boldsymbol{y}$ （或任何 n 维列向量） 投影到 $\boldsymbol{X}$ 的列空间 $\mathrm{col}(\boldsymbol{X})$ 。

进一步，残差向量 $\boldsymbol{e}$ 可写为

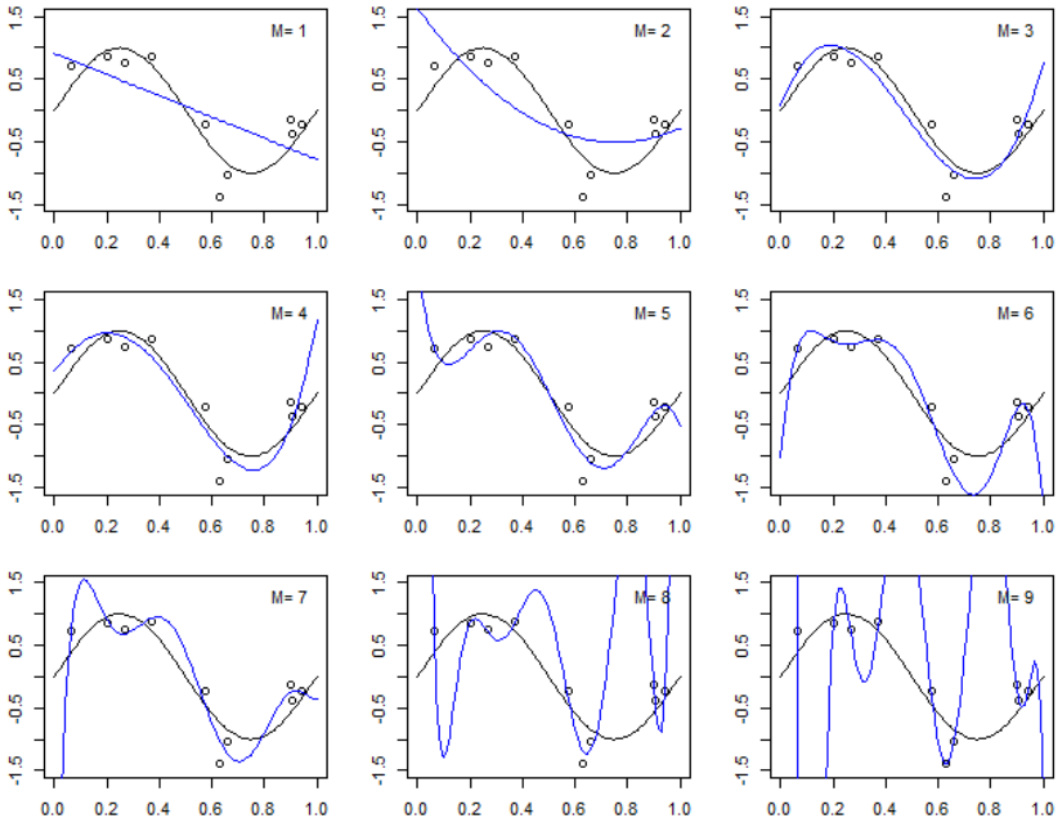
$$\boldsymbol{e} = \boldsymbol{y} - \hat{\boldsymbol{y}} = \boldsymbol{y} - \boldsymbol{P}_{\boldsymbol{X}}\boldsymbol{y} = \underbrace{(\boldsymbol{I} - \boldsymbol{P}_{\boldsymbol{X}})\boldsymbol{y}}_{=\boldsymbol{M}_{\boldsymbol{x}}} \boldsymbol{y} = \boldsymbol{M}_{\boldsymbol{X}}\boldsymbol{y}$$

其中，$n \times n$ 矩阵 $\boldsymbol{M}_{\boldsymbol{X}} \equiv \boldsymbol{I} - \boldsymbol{P}_{\boldsymbol{X}} = [\boldsymbol{I} - \boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}']$ 称为 $\boldsymbol{X}$ 的“消灭矩阵”（annihilator matrix）, 其作用是将 $\boldsymbol{y}$ 向 $\boldsymbol{X}$ 的列空间 $\mathrm{col}(\boldsymbol{X})$ 投影，然后，得到此投影的残差。



图表 1 最小二乘法的正交性

直观上，数据可视为信号与噪声的组合。当样本内的拟合得越来越完美时，这意味着回归函数也拟合了大量的噪声（因为回归函数“跟”数据跟得太紧）;而噪声对于样本外的预测毫无意义。因此，在过拟合的情况下，虽然在样本内的拟合优度很高，但模型在样本外的预测能力反而下降。然而，机器学习目的恰恰是样本外的预测能力，即将模型运用于其未见过的数据（unseendata）时，所具有的推广预测能力，即模型的泛化能力（unseendata）。另一方面，对于模型已经见过并据此进行优化的训练数据，即使拟合得再好，也说明不了问题：它可能只是记住了训练数据，比如此例中 $M = 9$ 的多项式回归情形。



图表 2 多项式对正弦函数的拟合

随着模型复杂程度的增加，测试误差一般呈现出 U 形的曲线特征，即先下降而后上升。为什么测试误差一般呈 U 形？事实上，测试误差受到两种不同力量的影响，即偏差与方差。偏差（bias）是指估计量是否有系统误差（比如，系统性高估或低估）。给定 $\boldsymbol{x}$，则估计量 $\hat{f}(\boldsymbol{x})$ 的偏差定义为

$$Bias\left[\hat{f}(\boldsymbol{x})\right] \equiv E\left[\hat{f}(\boldsymbol{x})\right] - f(\boldsymbol{x})$$

可见，偏差度量的是在大量重复抽样过程中，$\hat{f}(\boldsymbol{x})$ 对于 $f(\boldsymbol{x})$ 的平均偏离程度。另一方面， 方差（variance）则衡量在大量重复抽样过程中，估计量 $\hat{f}(\boldsymbol{x})$ 本身围绕着其（条件）期望值 $E\hat{f}(\boldsymbol{x})$ 的波动幅度，其定义为

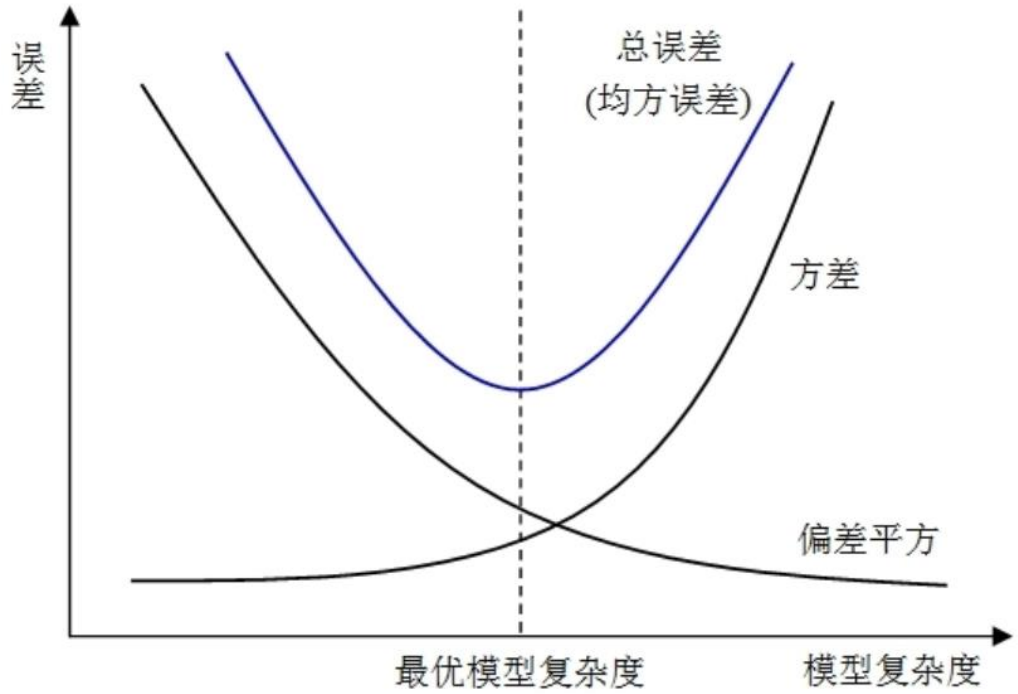
$$Var\left[\hat{f}(\boldsymbol{x})\right] \equiv E\left[\hat{f}(\boldsymbol{x}) - E\hat{f}(\boldsymbol{x})\right]^2$$

给定 $\boldsymbol{x}$（观测值），估计量 $\hat{f}(\boldsymbol{x})$ 的均方误差（$g(x) = \hat{f}(\boldsymbol{x})$）可作如下分解：

$$
\begin{aligned}
MSE\left[\hat{f}(\boldsymbol{x})\right] &= E_x E_y\left[y-\hat{f}(\boldsymbol{x})\right]^2 \\
&= E_x\left[\underbrace{f(\boldsymbol{x})+\varepsilon}_{y=f(\boldsymbol{x})+\varepsilon}-\hat{f}(\boldsymbol{x})\right]^2 \\
&= E_x\left[f(\boldsymbol{x})-E\hat{f}(\boldsymbol{x})+E\hat{f}(\boldsymbol{x})-\hat{f}(\boldsymbol{x})+\varepsilon\right]^2 \\
&= \underbrace{E_x\left[E\hat{f}(\boldsymbol{x})-f(\boldsymbol{x})\right]^2}_{Bias^2}+\underbrace{E_x[\hat{f}(\boldsymbol{x})-E\hat{f}(\boldsymbol{x})]^2}_{Variance}+\underbrace{E(\varepsilon^2)}_{E(\varepsilon^2|\boldsymbol{x})=E(\varepsilon^2)=Var(\varepsilon)} \\
&= \underbrace{Bias^2+Variance}_{reducible}+\underbrace{Var(\varepsilon)}_{ireducille}
\end{aligned}
$$

因此，估计量 $\hat{f}(\boldsymbol{x})$ 的均方误差可分解为偏差平方（$Bias^2$）、方差（$Variance$）与扰动项方差 $Var(\varepsilon)$ 之和。其中，偏差平方与方差均“可降低”（reducible）；在极端情况下，如果知道真实函数 $f(\boldsymbol{x})$，则偏差与方差均为 0。扰动项方差 $Var(\varepsilon)$ 则“不可降低”（irreducible），即使知道真实函数 $f(\boldsymbol{x})$，但 $Var(\varepsilon)$ 也依然存在。本质上，扰动项 ε 的方差 $Var(\varepsilon)$ 决定了预测问题困难程度。

通常来说，偏差平方与方差之间存在着此消彼长替代关系。当模型过于简单（比如多项式拟合案例中 M 为 1 或 2），则估计量 $\hat{f}(\boldsymbol{x})$ 的偏差较大，但方差较小；此时存在欠拟合。随着模型的复杂程度增加，估计量 $\hat{f}(\boldsymbol{x})$ 越来越接近于数据，故偏差变小；但由于数据包含噪声，紧跟数据使得方差随之增大，此时易出现过拟合。因此，在选择模型复杂程度时，存在偏差与方差的权衡（bias-variance trade-off），故须选择合适的模型复杂程度，使得模型的均方误差来达到最小值（比如上例中的 $M=3$）。



图表 3 偏差与方差的权衡

基准模型说明

设定以下模型研究企业数字化转型对于企业创新能力的影响

$$Innovation_{i,t+1} = \alpha_0 + \alpha_1 Digit_{i,t} + \sum \alpha_i Controls_{i,t} + \mu_t + \lambda_i + \varepsilon_{i,t}$$

其中，$Innovation_{i,t+1}$ 代表企业创新能力，$Digit_{i,t}$ 代表企业数字化转型程度，$Controls_{i,t}$ 包含一系列控制变量，μ_t 表示年份固定效应，λ_i 表示企业固定效应。本文关注的主要系数为 α_1。若企业数字化转型对企业创新具有显著的促进作用，则 α_1 应显著为正。

Variable	变量定义
Innovation	企业创新，用未来一期专利申请总数+1 的自然对数表示
Lndgt	数字化转型程度，用年报中数字化转型相关词汇的词频+1 的自然对数表示
Size	企业规模，用营业收入的自然对数表示
LEV	资产负债率，等于总负债/总资产
PPE	固定资产占比，等于固定资产/总资产
ROA	总资产报酬率，等于净利润/总资产
Cash	经营活动现金净额/总资产
SOE	产权性质，若为国有企业，则取值为 1，否则为 0
Boardsize	董事会规模，用董事会人数的自然对数表示
Indboard	独立董事占比
BM	账面市值比，等于股东权益总额/公司市值
Dual	两职合一，若董事长与总经理为同一人，则取值为 1，否则为 0
Age	企业年龄，等于企业成立年数

图表 4 变量定义

吴非等（2021）首次将上市公司年度报告中“数字化转型”相关词语的词频作为企业数字化转型程度的衡量指标。企业的年度报告作为总结性和导向性的对外报告，是企业向外部信息使用者传递信息的载体，企业积极开展数字化转型的意愿和成果很有可能反映在企业的年报中。因此，使用年度报告中“企业数字化转型”相关词语的词频来衡量数字化转型的程度具有合理性。

专利申请反映了企业投入于创新项目后的成果，能够衡量企业的创新能力。同时，专利申请量比专利授予量更接近创新产出的实际时间， 能更真实地反映创新水平。因此，本文参考李春涛等、黎文靖和郑曼妮的做法，将专利申请数作为企业创新的代理变量。专利数据也具有“右偏性”特征，本文将专利申请数加 1 后取自然对数。为缓解可能存在的反向因果问题，本文还将专利申请取未来一期处理。具体计算公式如下

$$Innovation_{i,t+1} = Ln(\ PatentApply_{i,t+1} + 1)$$

企业创新将受到融资约束、管理层激励、创新失败容忍度、代理问题、创新关注等多种因素的影响。创新项目失败率高、确定性低、资金需求大、周期长等特征，使现在的企业开始依靠数字化转型对企业创新的赋能作用。因此，本文选取控制变量有：公司规模（Size）、资产负债率（LEV）、固定资产占比（PPE）、总资产报酬率（ROA）、 经营活动现金净额 / 总资产（Cash）、产权性质（SOE）、董事会规模（Boardsize）、独立董事占比（Indboard）、账面市值

比（BM）、两职合一（Dual）和企业年龄（Age）作为控制变量。此外，还加入了年份固定效应和个体固定效应。

数据获取与清洗结果

```
setwd("/Users/apple/Downloads/同步空间/大三上课程/R/Pre/数据")
library(tidyverse)
library(readxl)
rawdata <- read_xlsx("data_raw.xlsx")
id <- read_xlsx("沪深 A 非 ST 非金融上市公司.xlsx",col_names    = FALSE)
colnames(id) <- c("股票代码","股票简称")
data <- inner_join(id,rawdata,by="股票代码")
data <- filter(data,上市状态=="正常上市")
data <- data    %>%
   select(1,2,3,5,6,7,12:27)
data <- drop_na(data)
data <- filter(data,between(年份,2012,2022))
data <- data    %>%
   select(1:9,12:22)
data <- data %>%
   mutate(Innovation=log(专利总共申请总量+1))
data <- data %>%
   mutate(Digit=log(数字化转型程度_A+1))
data <- data %>%
   mutate(Size=log(营业收入_元))
data <- data %>%
   mutate(Lev=资产负债率)
data <- data %>%
   mutate(PPE=固定资产总额占总资产之比)
data <- data %>%
```

```
  mutate(ROA=总资产净利润率)
data <- data %>%
  mutate(Cash=现金资产比率)
data <- data %>%
  mutate(SOE=控股股东性质)
data <- data %>%
  mutate(Boardsize=log(董事会规模_人))
data <- data %>%
  mutate(Indboard=独立董事占比)
data <- data %>%
  mutate(BM=账面市值比)
data <- data %>%
  mutate(Dual=两职合一)
data <- data %>%
  mutate(Age=企业年龄)
data <- data   %>%
  select(1:6,21:33)
data <- data %>%
  mutate(Id=1)
data[1:11,20] <- 0
data <- data %>%
  mutate(Year=年份-2012)
data <- na.omit(data)
```

```
> data
# A tibble: 30,448 × 19
   股票代码 股票简称.x  年份 行业名称 行业代码 省份  Innovation Digit  Size   Lev     PPE
      <dbl> <chr>      <dbl> <chr>    <chr>    <chr>      <dbl> <dbl> <dbl> <dbl>   <dbl>
 1        2 万科A       2012 房地产业 K70      广东…      1.79  1.10  25.4 0.783 0.00426
 2        2 万科A       2013 房地产业 K70      广东…      1.61  1.61  25.6 0.780 0.00444
 3        2 万科A       2014 房地产业 K70      广东…      3.26  3.18  25.7 0.772 0.00454
 4        2 万科A       2015 房地产业 K70      广东…      3.26  1.61  26.0 0.777 0.00804
 5        2 万科A       2016 房地产业 K70      广东…      2.94  1.10  26.2 0.805 0.00820
 6        2 万科A       2017 房地产业 K70      广东…      3.47  1.39  26.2 0.840 0.00609
 7        2 万科A       2018 房地产业 K70      广东…      3.69  2.08  26.4 0.846 0.00755
 8        2 万科A       2019 房地产业 K70      广东…      3.74  1.61  26.6 0.844 0.00717
 9        2 万科A       2020 房地产业 K70      广东…      3.89  2.20  26.8 0.813 0.00673
10        2 万科A       2021 房地产业 K70      广东…      3.66  2.40  26.8 0.797 0.00661
# ℹ 30,438 more rows
# ℹ 8 more variables: ROA <dbl>, Cash <dbl>, SOE <dbl>, Boardsize <dbl>, Indboard <dbl>,
#   BM <dbl>, Dual <dbl>, Age <dbl>
```

图表 5 本文数据集结构

基准回归结果

. encode(股票简称 x),gen(id)

. xtset id 年份

. xtreg innovation digit size lev ppe roa cash soe boardsize indboard bm dual age ,fe robust

```
Fixed-effects (within) regression               Number of obs      =        30,448
Group variable: id                              Number of groups   =         4,397

R-squared:                                      Obs per group:
     Within  = 0.1995                                         min =             1
     Between = 0.0354                                         avg =           6.9
     Overall = 0.0476                                         max =            11

                                                F(12, 4396)        =        166.98
corr(u_i, Xb) = -0.3378                         Prob > F           =        0.0000

                                    (Std. err. adjusted for 4,397 clusters in id)
------------------------------------------------------------------------------
             |               Robust
  innovation |      Coef.   Std. err.      t    P>|t|     [95% conf. interval]
-------------+----------------------------------------------------------------
       digit |   .0649232   .0106135     6.12   0.000     .0441154    .0857309
        size |   .3086007   .0213392    14.46   0.000     .2667651    .3504363
         lev |  -.0005683   .0041583    -0.14   0.891    -.0087207    .0075841
         ppe |  -.2427198   .1277654    -1.90   0.058    -.4932045    .0077648
         roa |   .0171784   .0180771     0.95   0.342    -.0182619    .0526187
        cash |  -.1950347    .082702    -2.36   0.018    -.3571723    -.032897
         soe |   .0103067   .0297877     0.35   0.729    -.0480922    .0687056
   boardsize |   .1062044   .0897143     1.18   0.237    -.0696808    .2820896
    indboard |  -.0003287   .0025724    -0.13   0.898    -.0053719    .0047144
          bm |   .1619138   .0421268     3.84   0.000      .079324    .2445036
        dual |   .0471588    .023517     2.01   0.045     .0010536    .0932639
         age |   .0801267   .0038337    20.90   0.000     .0726107    .0876427
       _cons |  -5.086779   .5185527    -9.81   0.000    -6.103404   -4.070155
-------------+----------------------------------------------------------------
     sigma_u |  1.5798192
     sigma_e |  .81313919
         rho |  .79056374   (fraction of variance due to u_i)
------------------------------------------------------------------------------
```

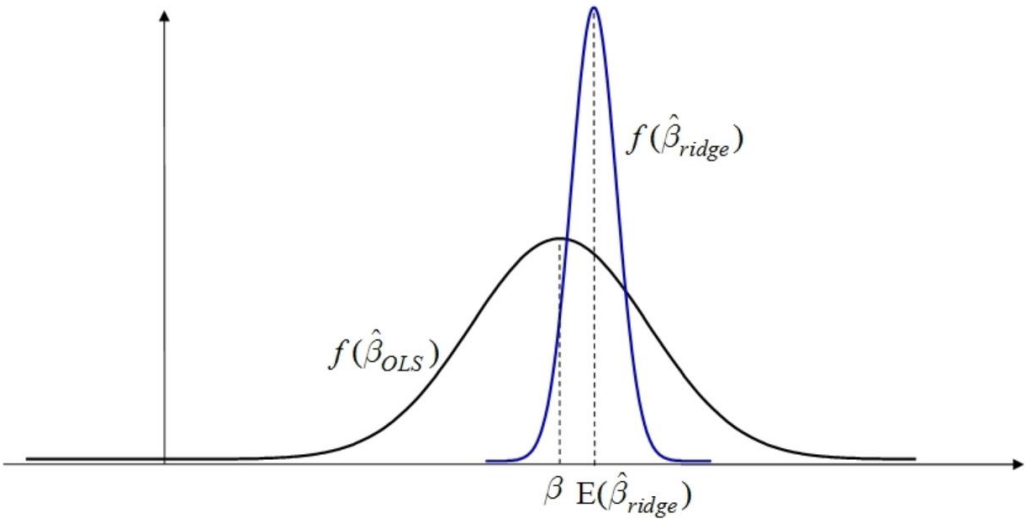
图表 6 基准回归结果

研究 1：线性回归参数估计的误差修正

对于任意估计量 $\hat{\boldsymbol{\beta}}$ ，其均方误差可分解为方差与偏差平方之和：

$$MSE(\hat{\boldsymbol{\beta}}) \equiv E[(\hat{\boldsymbol{\beta}}-\boldsymbol{\beta})(\hat{\boldsymbol{\beta}}-\boldsymbol{\beta})'] = Var(\hat{\boldsymbol{\beta}}) + [Bias(\hat{\boldsymbol{\beta}})][Bias(\hat{\boldsymbol{\beta}})]'$$

因此，使均方误差最小化，可视为在方差与偏差之间进行权衡（trade-off）。比如，一个无偏估计量（偏差为 0），如果方差很大，则可能不如一个虽然有偏差但是方差却很小的估计量。在（严格）多重共线性的情况下，虽然 OLS 估计量无偏，但其方差太大（无穷大），而岭回归虽有少量偏差，但可大幅减少方差，这使得岭回归估计量的均方误差（MSE）可能比 OLS 更小，参见图 6。



图表 7 岭回归估计量与 OLS 估计量在方差与偏差权衡下的选择

图中，横轴为待估参数 β 的可能取值，而纵轴为概率密度。虽然 OLS 估计量为无偏估计，概率密度函数 $f(\hat{\beta}_{\text{OLS}})$ 的中心位置正好在 β，但其方差较大。另一方面，虽然岭回归估计量有一定偏差 $E(\hat{\beta}_{ridge}) \neq \beta$，但其方差较小。这使得 $\text{MSE}(\hat{\beta}_{\text{ridge}})$ 小于 $\text{MSE}(\hat{\beta}_{\text{OLS}})$ 。

为得到参数向量 $\boldsymbol{\beta}$ 的唯一解，须对其取值范围有所限制，进行所谓“正则化”（regularization）。为此，考虑在损失函数（loss function） 中加入“惩罚项”（penalty term），进行惩罚回归（penalized regression）：

$$\min_{\boldsymbol{\beta}} L(\boldsymbol{\beta}) = \underbrace{(\boldsymbol{y}-\boldsymbol{X\beta})'(\boldsymbol{y}-\boldsymbol{X\beta})}_{SSR} + \underbrace{\lambda\|\boldsymbol{\beta}\|_2^2}_{penalty}$$

其中，损失函数的第 1 项依然为残差平方和（SSR）；而第 2 项为惩罚项，也称其为“正则项”（regularization）。 $\lambda \geqslant 0$ 称为“调节参数”（tuning parameter），控制惩罚的力度。 $\|\boldsymbol{\beta}\|_2$ 为参数向量 $\boldsymbol{\beta}$ 长度，即 $\boldsymbol{\beta}$ 到原点的欧氏距离，也称为“2 -范数（L_2 norm）：

$$\|\boldsymbol{\beta}\|_2 \equiv \sqrt{\beta_1^2 + \cdots + \beta_p^2}$$

将惩罚项写为 $\lambda\|\boldsymbol{\beta}\|_2^2 = \lambda\left(\beta_1^2 + \cdots + \beta_p^2\right) = \lambda\boldsymbol{\beta}'\boldsymbol{\beta}$，则损失函数可写为

$$\min_{\boldsymbol{\beta}} L(\boldsymbol{\beta}) = (\boldsymbol{y}-\boldsymbol{X\beta})'(\boldsymbol{y}-\boldsymbol{X\beta}) + \lambda\boldsymbol{\beta}'\boldsymbol{\beta}$$

由于惩罚项 $\lambda\boldsymbol{\beta}'\boldsymbol{\beta}$ 只是简单的二次型（其二次型矩阵为单位矩阵），故不难使用向量微分的规则得到相应的一阶条件：

$$\frac{\partial L(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = -2\boldsymbol{X}'(\boldsymbol{y}-\boldsymbol{X\beta}) + 2\lambda\boldsymbol{\beta} = 0$$

经移项整理可得：

$$(\boldsymbol{X}'\boldsymbol{X} + \lambda\boldsymbol{I})\boldsymbol{\beta} = \boldsymbol{X}'\boldsymbol{y}$$

所得最优解正是岭回归估计量：

$$\hat{\boldsymbol{\beta}}_{\text{ridge}}(\lambda) \equiv (\boldsymbol{X}'\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}'\boldsymbol{y}$$

岭回归的目标函数可等价地写为如下有约束的极值问题：

$$\min_{\boldsymbol{\beta}} (\boldsymbol{y}-\boldsymbol{X\beta})'(\boldsymbol{y}-\boldsymbol{X\beta})$$
$$\text{s.t.} \|\boldsymbol{\beta}\|_2^2 \leqslant t$$

其中，$t \geqslant 0$ 为某常数。对于此约束极值问题，可引入拉格朗日乘子函数，并以 λ 作为其乘子：

$$\min_{\boldsymbol{\beta},\lambda} \tilde{L}(\boldsymbol{\beta}) = (\boldsymbol{y} - \boldsymbol{X\beta})'(\boldsymbol{y} - \boldsymbol{X\beta}) + \lambda(\|\boldsymbol{\beta}\|_2^2 - t)$$

对 $\boldsymbol{\beta}$ 求偏导数，并注意到 $\|\boldsymbol{\beta}\|_2^2 = \boldsymbol{\beta}'\boldsymbol{\beta}$，可得一阶条件：

$$\frac{\partial \tilde{L}(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = -2\boldsymbol{X}'(\boldsymbol{y} - \boldsymbol{X\beta}) + 2\lambda\boldsymbol{\beta} = \boldsymbol{0}$$

显然，该约束极值问题与岭回归的最小化问题等价。

Tibshirani （1996） 提出 “套索估计量”（Least Absolute Shrinkage and Selection Operator，简记 LASSO），将岭回归惩罚项中 2-范数改为 1 -范数：

$$\min_{\boldsymbol{\beta}} L(\boldsymbol{\beta}) = \underbrace{(\boldsymbol{y} - \boldsymbol{X\beta})'(\boldsymbol{y} - \boldsymbol{X\beta})}_{SSR} + \underbrace{\lambda\|\boldsymbol{\beta}\|_1}_{panelty}$$

其中，$\|\boldsymbol{\beta}\|_1 = \sum_{j=1}^{p} |\beta_j|$ 为参数向量 $\boldsymbol{\beta}$ 的 1-范数（L_1 norm）

类似于岭回归，Lasso 最小化问题也可等价写为如下约束极值问题：

$$\min_{\boldsymbol{\beta}} (\boldsymbol{y} - \boldsymbol{X\beta})'(\boldsymbol{y} - \boldsymbol{X\beta})$$
$$\text{s.t. } \|\boldsymbol{\beta}\|_1 \leqslant t$$

其中，$t \geqslant 0$ 为某常数。

Zou and Hastie（2005）将 Lasso 与岭回归相结合，提出弹性网（elastic net）估计量。在弹性网估计量损失函数中，同时包含 L_1 与 L_2 惩罚项：

$$\min_{\boldsymbol{\beta}}(\boldsymbol{y}-\boldsymbol{X\beta})'(\boldsymbol{y}-\boldsymbol{X\beta})+\lambda_1\|\boldsymbol{\beta}\|_1+\lambda_2\|\boldsymbol{\beta}\|_2^2$$

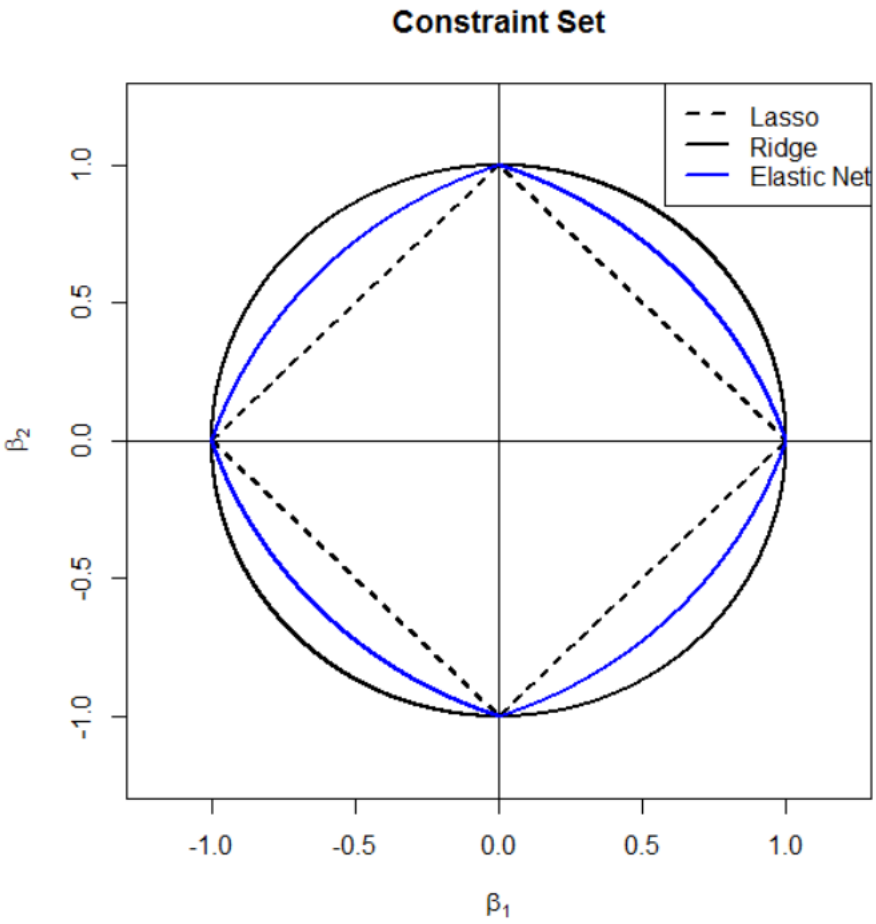
其中，$\lambda_1 \geqslant 0$ 与 $\lambda_2 \geqslant 0$ 都是调节参数。由于 λ_1 与 λ_2 的取值范围均为无穷，不便于使用交叉验证选择其最优值。为此，定义 $\lambda \equiv \lambda_1+\lambda_2, \alpha \equiv \frac{\lambda_1}{\lambda}$，可将损失函数等价地写为

$$\min_{\beta}(\boldsymbol{y}-\boldsymbol{X\beta})'(\boldsymbol{y}-\boldsymbol{X\beta})+\lambda[\alpha\|\boldsymbol{\beta}\|_1+(1-\alpha)\|\boldsymbol{\beta}\|_2^2]$$

其中，$\lambda \geqslant 0$ 与 $0 \leqslant \alpha \leqslant 1$ 为调节参数。由于调节参数 α 的取值局限于区间 [0,1]，故便于通过交叉验证选择其最优值。显然，如果 $\alpha=0$，则弹性网退化为岭回归；而如果 $\alpha=1$，则弹性网退化为 Lasso。因此，岭回归与 Lasso 都是弹性网的特例。如果 $0<\alpha<1$，则弹性网为岭回归与 Lasso 之间的折中。容易看出，式（37）可等价地写为以下约束极值问题:

$$\min_{\boldsymbol{\beta}}(\boldsymbol{y}-\boldsymbol{X\beta})'(\boldsymbol{y}-\boldsymbol{X\beta})$$
$$\text{s. t.}\quad \alpha\|\boldsymbol{\beta}\|_1+(1-\alpha)\|\boldsymbol{\beta}\|_2^2 \leqslant t$$

其中，$t \geqslant 0$ 为调节参数。显然，其约束集“$\alpha\|\boldsymbol{\beta}\|_1+(1-\alpha)\|\boldsymbol{\beta}\|_2^2 \leqslant t$”不同于岭回归或 Lasso 的约束集，但兼具二者的特点。仍以二元回归为例，$\boldsymbol{\beta}=(\beta_1,\beta_2)'$，则弹性网估计量的约束集为 $\alpha(|\beta_1|+|\beta_2|)+(1-\alpha)(\beta_1^2+\beta_2^2) \leqslant t$，图 9.10 展示了弹性网 $(\alpha=0.5)$，Lasso 及岭回归的约束集。



图表 8 弹性网约束条件的几何解释

Lasso 回归估计结果

```
install.packages("glmnet")
library(glmnet)
X <- as.matrix(data[, c("Digit", "Size", "Lev", "PPE", "ROA", "Cash", "SOE",
"Boardsize", "Indboard", "BM", "Dual", "Age","Id","Year")])
y <- data$Innovation
X <- scale(X)
model <- glmnet(X, y, alpha = 1) # LASSO 回归
cv_model <- cv.glmnet(X, y, alpha = 1)
best_lambda <- cv_model$lambda.min
```

coef(cv_model, s = best_lambda)

```
15 x 1 sparse Matrix of class "dgCMatrix"
                      s1
(Intercept)  2.767961014
Digit        0.123564265
Size         0.757993225
Lev          .
PPE         -0.000522285
ROA          .
Cash        -0.046287744
SOE          .
Boardsize    .
Indboard     .
BM          -0.079218247
Dual         .
Age         -0.246227581
Id           .
Year         0.190665034
```

图表 9 Lasso 回归估计结果

弹性网回归估计结果

model <- glmnet(X, y, alpha = 0.5) # 这里设置 alpha 为 0.5，表示弹性网回归

cv_model <- cv.glmnet(X, y, alpha = 0.5)

best_lambda <- cv_model$lambda.min

coef(cv_model, s = best_lambda)

```
15 x 1 sparse Matrix of class "dgCMatrix"
                      s1
(Intercept)  2.767961014
Digit        0.125660924
Size         0.742877070
Lev          .
PPE         -0.003882644
ROA          .
Cash        -0.051310192
SOE          .
Boardsize    .
Indboard     .
BM          -0.075428644
Dual         .
Age         -0.240701008
Id           .
Year         0.189944438
```

图表 10 弹性网回归估计结果

研究 2：使用非参数方法对模型参数估计进行优化

分类树

Breiman et al.（1984）研究了 UCSD 医学中心的一个案例。当心梗病人进入 UCSD 医学中心后 24 小时内，测量 19 个变量，包括血压、年龄以及 17 个排序或虚拟变量。数据集中包含 215 个病人，其中 37 人为高危病人。研究目的是为了快速预测哪些心梗病人为“高危病人”（无法活过 30 天），哪些是“低危病人”（可活过 30 天），从而更好地配置医疗资源。

Breiman et al. （1984）建立了如下分类树

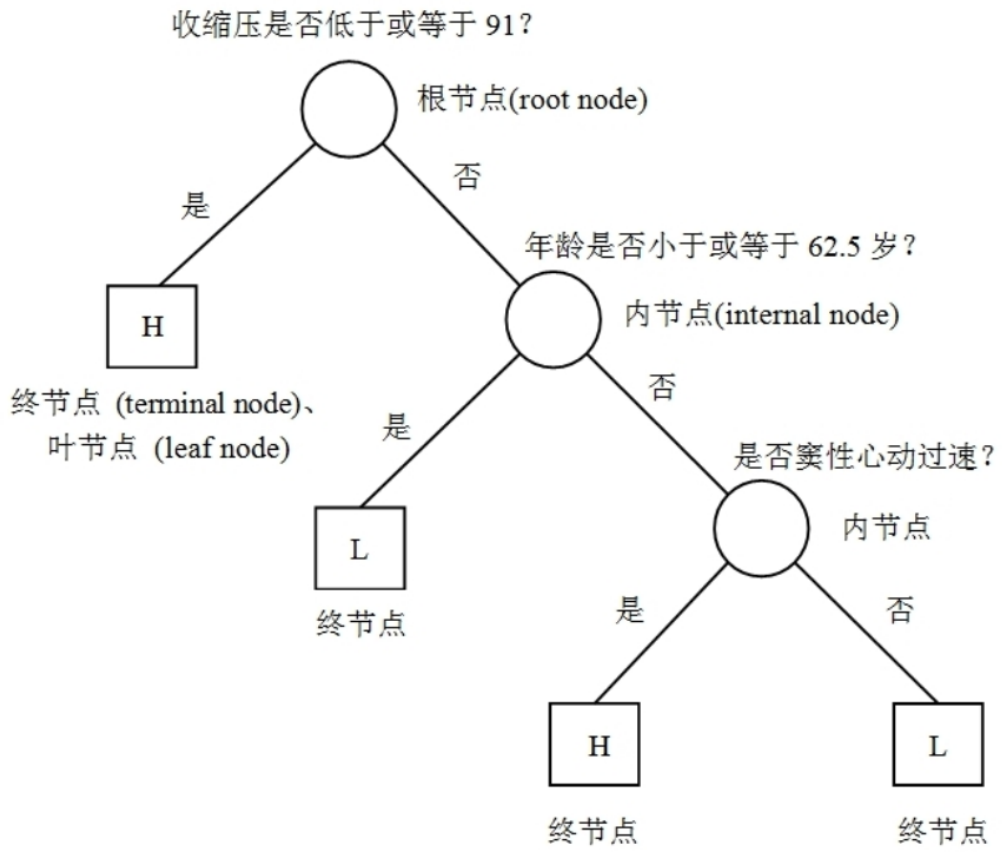
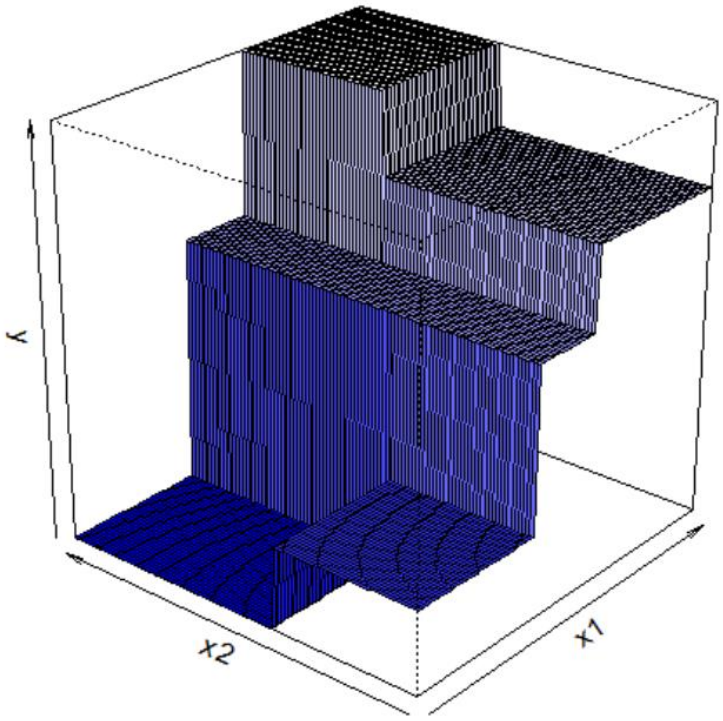


图 1 UCSD 医学中心判断高危病人的分类树

对于三维以上的特征空间，我们依然可用树状结构来表示，因为决策树每次仅使用一个变量进行分裂（splitting）决策树模型将特征空间分割为若干（超）矩形的终节点。在进行预测时，每个终节点只有一个共同的预测值。对于分类问题，此预测值为该终节点所有训练样本的最常见类别（most commonly occurring class）。对于回归问题，此预测值为该终节点所有训练样本的平均值。因此，在数学上，决策树为“分段常值函数”（piecewise constant function），参见图 10。



图表 11 作为分段常值函数的决策树

决策树成为一种灵活而有用的算法，特别是作为“基学习器”（base learmer）可广泛用于随机森林与提升法。在理论上，我们可以考虑任意形状分区。

在估计决策树模型时，面临一个选择，即何时停止节点分裂。一个较好的解决方法是，先让决策树尽情生长，记最大的树为 T_{max}，再进行“修枝”（pruning），以得到一个“子树”（subtree）T。对于任意子树 $T \subseteq T_{\max}$，定义其“复杂性”（complexity）为子树 T 的终节点数目（number of terminal nodes），记为 $|T|$。为避免过拟合，不希望决策树过于复杂，故惩罚其规模 $|T|$：

$$\min_T \underbrace{R(T)}_{cost} + \lambda \cdot \underbrace{|T|}_{complxity}$$

其中，$R(T)$ 为原来的损失函数，比如 $0-1$ 损失函数（$0-1$ loss），即在训练样本中，如果预测正确，则损失为 0，而若预测错误则损失为 1。$\lambda \geqslant 0$ 称为调节参数（tuning parameter），也称其为“复杂性参数”（complexity parameter，简记 CP）。λ 控制对决策树规模 $|T|$ 的惩罚力度，可通过交叉验证确定。这种修枝方法称为成本—复杂性修枝（cost-complexity pruning），即在成本（损失函数 $R(T)$）与复杂性（决策树规模 $|T|$）之间进行最优的权衡。

对于回归问题，其响应变量 y 可为连续变量。因此，对于回归树，可使用“最小化残差平方和”作为节点的分裂准则。这意味着，在进行节点分裂时，希望分裂后残差平方和下降最多，即两个子节点的残差平方和的总和最小。为避免过拟合，对于回归树，也要使用惩罚项来进行修枝，即最小化如下目标函数：

$$\min_T \underbrace{\sum_{m=1}^{|T|} \sum_{x_i \in R_m} \left(y_i - \hat{y}_{R_m}\right)^2}_{cost} + \lambda \cdot \underbrace{|T|}_{complexity}$$

其中，R_m 为第 m 个终节点，而 $\hat{y}_{R_m}$ 为该终节点的预测值（此终节点的样本均值）。$\sum_{x_i \in R_m} \left(y_i - \hat{y}_{R_m}\right)^2$ 为第 m 个终节点的残差平方和，然后对所有终节点 $m = 1,2,\cdots,T$ 进行加总，即为成本 $R(T)$。

随机森林

Breiman（1996）提出“装袋法”（bootstrap aggregating，简记 bagging）。即装袋法使用“自助法”（bootstrap） 来搅动数据。

首先，对训练数据进行有放回的再抽样，得到 B 个自助样本（bootstrap samples），比如 $B = 500$。其中，第 b 个自助样本可写为

$$\left\{\boldsymbol{x}_i^{*b}, y_i^{*b}\right\}_{i=1}^{n},\ b = 1, \cdots, B$$

其中，上标（superscript）的星号“*”表示这是自助样本。每个自助样本的样本容量均为 n（保持样本容量不变）。其次，根据自助样本 $\left\{\boldsymbol{x}_i^{*b}, y_i^{*b}\right\}_{b=1}^{B}$ 估计 B 棵不同的决策树，不进行修枝。记其预测结果为

$$\left\{\hat{f}^{*b}(\boldsymbol{x})\right\},\ b = 1, \cdots, B$$

最后，对于回归树，将 B 棵决策树的预测结果进行平均：

$$\hat{f}_{\text{bag}}(\boldsymbol{x}) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(\boldsymbol{x})$$

由于 bagging 将很多决策树进行平均，故可降低估计量方差，从而提高模型的预测准确率。例如，假设随机变量 $\{z_1, \cdots, z_n\}$ 为独立同分布（iid），而方差为 σ^2 ，则样本均值 $\bar{z} = \frac{1}{n}\sum_{i=1}^{n} z_i$ 的方差可缩小 n 倍:

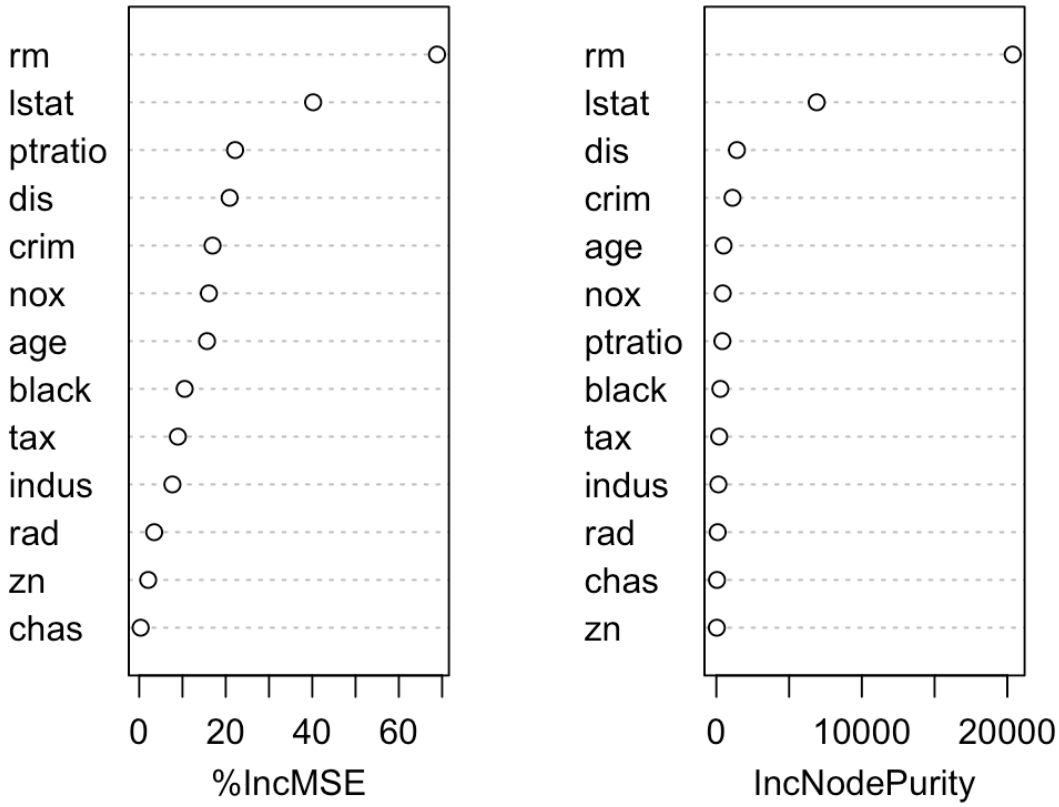
$$Var(\bar{z}) = Var\left(\frac{1}{n}\sum_{i=1}^{n} z_i\right) = \frac{1}{n^2}\sum_{i=1}^{n} Var(z_i) = \frac{\sigma^2}{n}$$

式中，由于 $\{z_1, \cdots, z_n\}$ 相互独立，式中的协方差均为 0 ，而所有 $\text{Var}(z_i) = \sigma^2$ （同分布）。另外，由于装袋法并不修枝，而让决策树尽情生长，故可降低每棵决策树的偏差（bias）；然后，通过 bagging 来控制方差。因此，在一定条件下，装袋法可以同时降低偏差与方差，因此可减小均方误差（MSE）或总误

差。Breiman（1996）使用若干标准的机器学习数据集，发现与单棵决策树相比，bagging 可显著降低测试误差。

显然，在使用装袋法时，由于所用到的算法完全相同，而每棵决策树所用自助样本都来自同样的原始数据，故必然存在比较高相关性。如果想进一步提高 bagging 的预测性能，则必须降低这些决策树之间的相关性。问题是，如何使决策树之间更不相关（decorrelate）？具体来说，bagging 决策树之间相关性强，也是因为使用相同特征变量 $\boldsymbol{x}$。为此，Breiman（2001b）大胆地提出了随机森林（Random Forest）：在 bagging 的基础上（依然使用自助样本），在决策树的每个节点进行分裂时，仅仅随机选取部分变量（比如 m 个变量）作为候选的分裂变量。比如，在一个节点随机选择 m 个变量作为候选的分裂变量（其余 $p-m$ 个变量则不用），而在下一节点，再次随机选择（可能不相同的）m 个变量作为候选的分裂变量，以此类推；然后对随机森林中的每棵决策树都如此操作。

与单棵决策树相比，随机森林包含很多决策树（比如 500 棵决策树的平均），故无法像单棵决策树那样进行解释。对于随机森林，应如何度量“变量重要性”（Variable Importance）？由于在每次节点分裂，仅使用一个变量，因此容易区分每个变量的贡献，考察该分裂变量使得残差平方和（或者基尼指数）下降多少。具体来说，对于每个变量，在随机森林的每棵决策树，可度量由该变量所导致的分裂准则函数的下降幅度。然后，针对此下降幅度，对每棵决策树进行平均，即为对该变量重要性的度量。将每个特征变量的重要性依次排列画图，即为变量重要性图（Variable Importance Plot）。



图表 12 变量重要性图（Variable Importance Plot）

我们依然感兴趣于每个变量对于 y 的边际效应（marginal effects）。比如，对于特征向量 $\boldsymbol{x}=(x_1,x_2,\cdots,x_p)'$，假设 $y=f(\boldsymbol{x})$，但函数 $f(\cdot)$ 无解析表达式（比如随机森林）。不失一般性，考虑第 1 个特征变量 x_1 对 y 的边际效应：

$$\frac{\partial y}{\partial x_1}=\frac{\partial f(x_1,x_2,\cdots,x_p)}{\partial x_1}$$

在上式中，边际效应 $\frac{\partial y}{\partial x_1}$ 依赖于其他变量 $(x_2,\cdots,x_p)$ 取值，一般来说不是常数（除非是线性模型）。为此，考虑在函数 $y=f(x_1,x_2,\cdots,x_p)$ 中，将其他变量 $(x_2,\cdots,x_p)$ 对于 y 的影响通过积分平均掉：

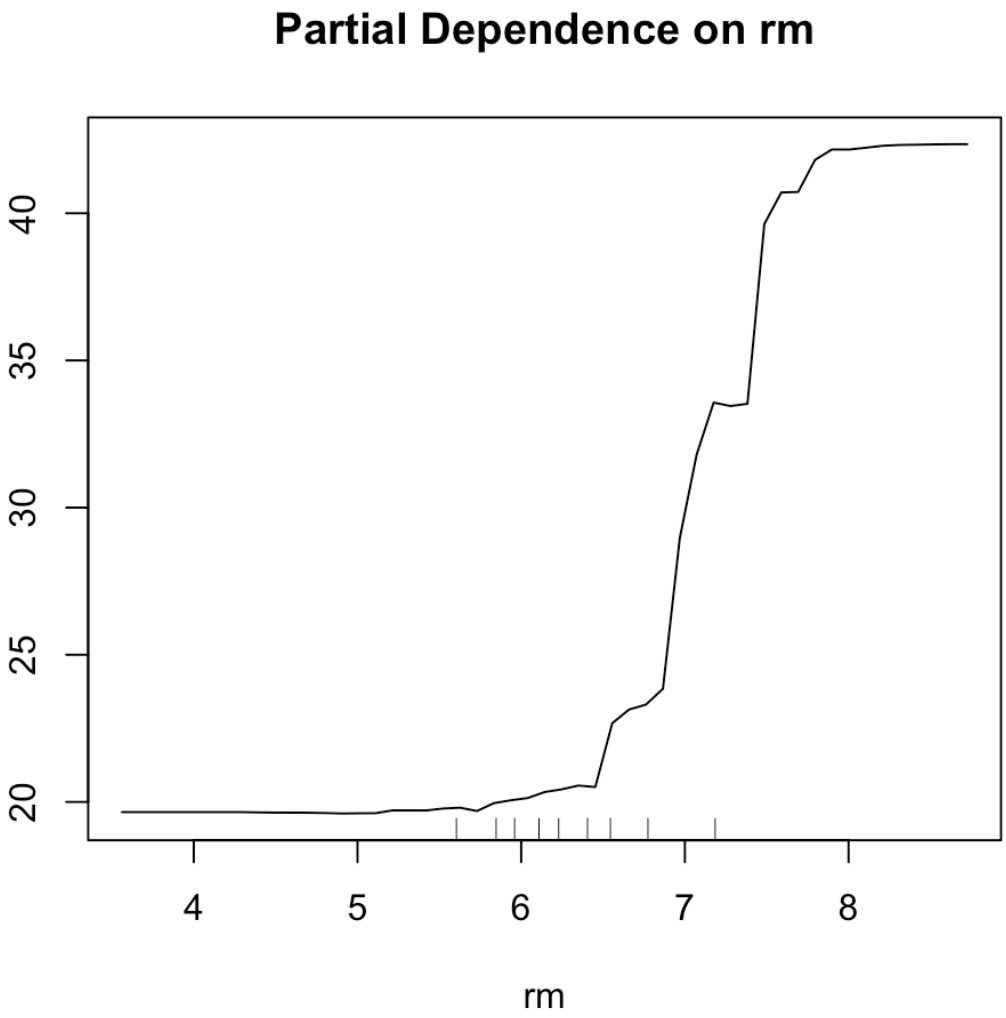
$$\phi(x_1)\equiv E_{x_2,\cdots,x_p}f(x_1,x_2,\cdots,x_p)$$

其中，期望算子 $E_{x_2,\cdots,x_p}(\cdot)$ 对变量 $(x_2,\cdots,x_p)$ 求期望，故在式（11）右边已将 $(x_2,\cdots,x_p)$ 积分积掉，因此所得结果 $\phi(x_1)$ 只是 x_1 的函数。当然，由于 $f(\cdot)$ 无解析表达式，一般很难直接计算此期望。为此，使用统计学的常用估计方法，以样本均值替代总体均值 $E_{x_2,\cdots,x_p}(\cdot)$ 可得：

$$\hat{\phi}(x_1) \equiv \frac{1}{n}\sum_{i=1}^{n} f(x_1, x_{i2}, \cdots, x_{ip})$$

在式中，任意给定 x_1 均可计算 $\hat{\phi}(x_1)$，并可画出 $\left(x_1, \hat{\phi}(x_1)\right)$ 的图像，称为偏依赖图（Partial Dependence Plot）。不难看出，偏依赖图适用于任何 $f(\cdot)$ 无解析表达式的黑箱（black box）方法。进一步，也可同时考虑多个变量对于 y 的偏影响。比如，响应变量 y 对于 (x_1, x_2) 的偏依赖可定义为

$$\hat{\phi}(x_1, x_2) \equiv \frac{1}{n}\sum_{i=1}^{n} f(x_1, x_2, x_{i3}, \cdots, x_{ip})$$



图表 13 偏依赖图（Partial Dependence Plot）

自适应提升

最早的提升法为 Freund and Schapire（1996，1997）提出的自适应提升法（Adaptive Boosting，简记 AdaBoost），仅适用于分类问题。对于分类问题，考虑依次种 M 棵树 $G_1(x),\cdots,G_M(x)$。对于第 m 棵树中错误分类的观测值，则在其之后的第 $m+1$ 棵树加大其权重，以此类推。具体来说，可通过以下两种方法来加大错分观测值的权重：

i. 权重更新（reweighting）。在定义基尼指数或信息熵的不纯度函数，以及在计算终节点预测值时，均考虑不同观测值的权重

ii. 再抽样（resampling）。在种每棵决策树时，都使用从“加权的分布”（weighted distribution）中再抽样得到的数据。当然，这需要先将每轮权重标准化，使得权重之和为 1 。

二分类问题的 AdaBoost 算法可分为以下 3 步；而其中第 2 步又分为 5 小步，针对所种决策树 $m = 1, \cdots, M$ 进行 for 循环：

1. 初始化每个观测值的权重 $\boldsymbol{w_i = \frac{1}{n}, i = 1, \cdots, n}$。这意味着，刚开始所有观测值的权重均相同，等于 $\frac{\mathbf{1}}{\boldsymbol{n}}$ （$\boldsymbol{n}$ 为样本容量）。

2. 对于决策树 $\boldsymbol{m = 1, \cdots, M}$，进行以下操作。

(1) 使用观测值 $\boldsymbol{i}$ 的当前权重 $\boldsymbol{w_i}$，估计第 $\boldsymbol{m}$ 棵决策树 $\boldsymbol{G_m(x)}$

(2) 根据当前权重 $\boldsymbol{w_i}$，计算第 $\boldsymbol{m}$ 棵决策树的错分率 $\boldsymbol{err_m}$：

$$err_m \equiv \frac{\sum_{i=1}^{n} w_i I\left(y_i \neq G_m(\boldsymbol{x}_i)\right)}{\sum_{i=1}^{n} w_i}$$

其中，分母为所有观测值的权重之和，而分子为错分观测值 $\left(y_i \neq G_m(x_i)\right)$ 的权重之和。

(3) 计算正确分类的对数几率（log odds）$\boldsymbol{\alpha_m}$ ，也即正确分类的概率 $\boldsymbol{1 - err_m}$ 除以错误分类的概率 $\boldsymbol{err_m}$，再取对数:

$$\alpha_m \equiv ln\left(\frac{1 - err_m}{err_m}\right)$$

(4) 更新观测值的权重（详见下文解释）：

$$w_i \leftarrow w_i \cdot exp\{\alpha_m \cdot I[y_i \neq G_m(\boldsymbol{x}_i)]\}$$

(5) 将所有权重标准化，保证权重之和为 $\mathbf{1}$，即

$$w_i \leftarrow \frac{w_i}{\sum_{j=1}^{n} w_j}$$

3. 将每棵决策树预测结果 $\boldsymbol{G_m(x)}$，以对数几率 $\boldsymbol{\alpha_m}$ 为权重，通过加权多数票（weighted majority vote） 的方式组合在一起，输出最终预测结果：

$$G(\boldsymbol{x}) = sign\left[\sum_{m=1}^{M} \alpha_m G_m(\boldsymbol{x})\right]$$

其中， sign(·) 为“符号函数”（sign function），即

$$sign(z) = \begin{cases} 1 & if \quad z \geqslant 0 \\ -1 & if \quad z < 0 \end{cases}$$

在 AdaBoost 算法的第 2 步中，进一步考察第（4）步观测值权重的变化。假定“弱分类器”（weak classifier）的错分率至少比随机猜测更低，则几率 $\frac{1-err_m}{err_m} > 1$，故对数几率 $\alpha_m = ln\left(\frac{1-err_m}{err_m}\right) > 0$ 。权重更新公式可以写为

$$w_i \leftarrow w_i \cdot exp\{\alpha_m \cdot I[y_i \neq G_m(\boldsymbol{x}_i)]\} = \begin{cases} w_i \cdot \left(\frac{1-err_m}{err_m}\right) & if\, y_i \neq G_m(\boldsymbol{x}_i) \\ w_i & if\, y_i = G_m(\boldsymbol{x}_i) \end{cases}$$

自从 Freund and Schapire（1996）提出 AdaBoost 算法后，即取得了很好的预测效果。但学界一直不明白为何 AdaBoost 的预测效果如此好。直至 Friedman et al.（2000），才给出 AdaBoost 的统计解释。关键结论是，AdaBoost 算法可以等价地视为前向分段加法模型（forward stagewise additive modeling），并使用了指数损失函数（exponential loss function）。

发现 AdaBoost 算法的数学本质后，即开启了改进与推广 AdaBoost 之门。比如，AdaBoost 算法使用指数损失函数，那么在前向分段加法模型中，使用其他损失函数，即可得到不同的算法。最初的 AdaBoost 算法仅适用于分类问题，这是因为它使用指数损失函数。而对于回归问题，很自然地可以考虑使用“误差平方”（squared loss）的损失函数：

$$L[y, f(\boldsymbol{x})] = [y - f(\boldsymbol{x})]^2$$

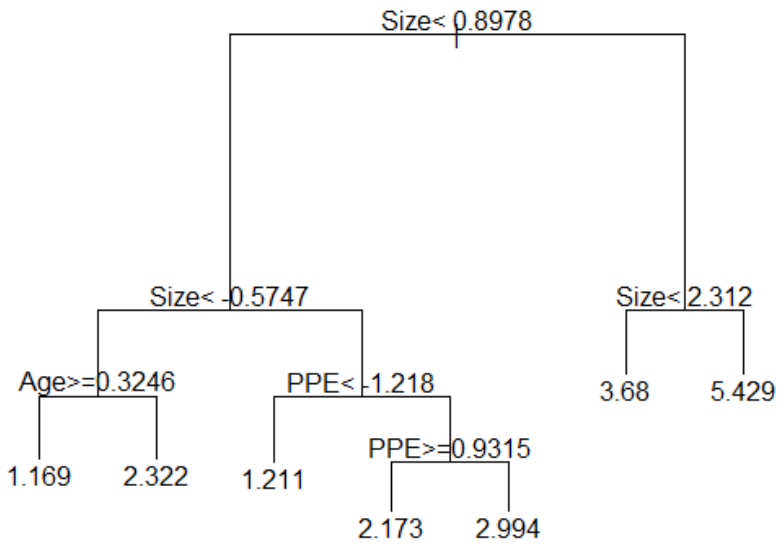
将前向分段加法模型代入此损失函数可得：

$$\begin{aligned} \min_{\beta, G} \; L[y_i, f_{m-1}(\boldsymbol{x}_i) + \boldsymbol{\beta} \cdot \boldsymbol{G}(\boldsymbol{x}_i; \boldsymbol{\gamma})] &= [y_i - f_{m-1}(\boldsymbol{x}_i) - \boldsymbol{\beta} \cdot \boldsymbol{G}(\boldsymbol{x}_i; \boldsymbol{\gamma})]^2 \\ &= \left[r_i^{(m)} - \boldsymbol{\beta} \cdot \boldsymbol{G}(\boldsymbol{x}_i; \boldsymbol{\gamma})\right]^2 \end{aligned}$$

其中，$r_i^{(m)} \equiv y_i - f_{m-1}(\boldsymbol{x}_i)$ 为当前阶段模型的回归残差（residual）。这意味着，只要以当前残差 $r_i^{(m)}$ 为响应变量，对特征向量 $\boldsymbol{x}_i$ 进行回归即可。这种方法也称为回归问题的提升法，或“2 范数提升法”（L_2 boosting）。在以残差 $r_i^{(m)}$ 为响应变量对特征向量 $\boldsymbol{x}_i$ 进行“回归”时，既可进行线性回归，也可使用回归树。如果使用回归树，则称为回归提升树 （boosted regression tree）。

分类树回归估计结果

```
data$Digit <- scale(data$Digit)
data$Size <- scale(data$Size)
data$Lev <- scale(data$Lev)
data$PPE <- scale(data$PPE)
data$ROA <- scale(data$ROA)
data$Cash <- scale(data$Cash)
data$SOE <- scale(data$SOE)
data$Boardsize <- scale(data$Boardsize)
data$Indboard <- scale(data$Indboard)
data$BM <- scale(data$BM)
data$Dual <- scale(data$Dual)
data$Age <- scale(data$Age)
data$Id <- scale(data$Id)
data$Year <- scale(data$Year)
install.packages("rpart")
library(rpart)
formula <- Innovation ~ Digit + Size + Lev + PPE + ROA + Cash + SOE + Boardsize
+ Indboard + BM + Dual + Age + Id + Year
model <- rpart(formula, data = data, method = "anova")#分类树回归
plot(model,margin=0.1)
text(model)
```



图表 14 分类树回归估计结果

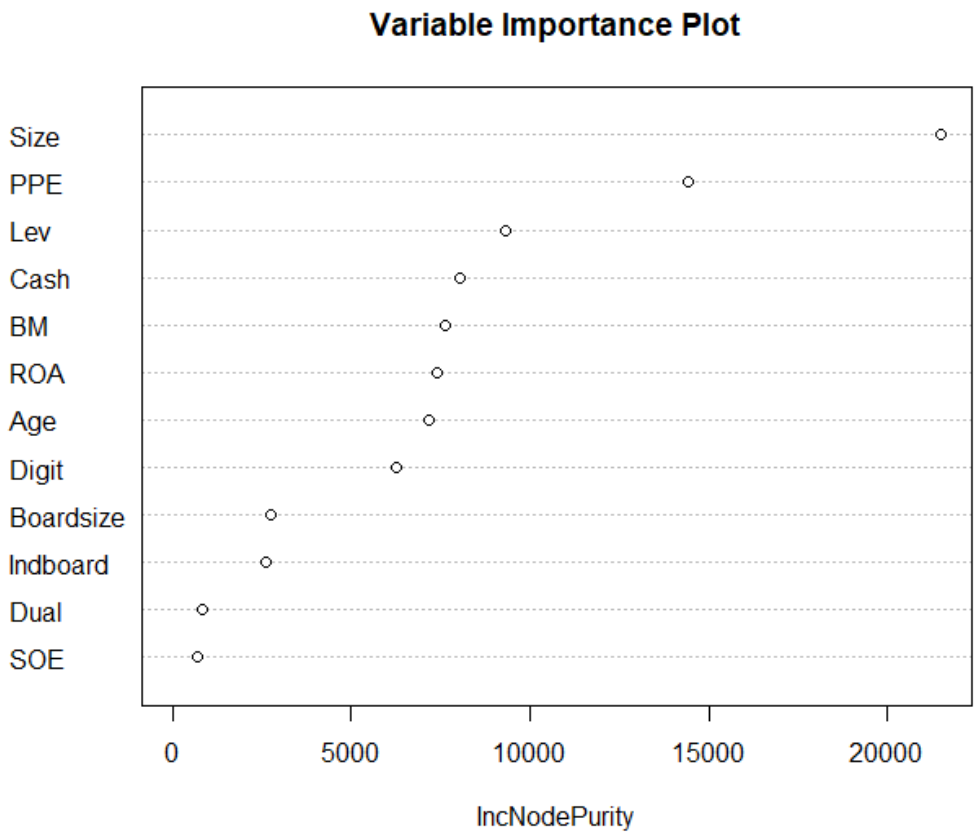
随机森林回归估计结果

```
install.packages("randomForest")
library(randomForest)
formula <- Innovation ~ Digit + Size + Lev + PPE + ROA + Cash + SOE + Boardsize
+ Indboard + BM + Dual + Age + Id + Year
model <- randomForest(formula, data = data, ntree = 50) #  随机森林回归估计
print(model)
```

```
Call:
 randomForest(formula = formula, data = data, ntree = 50)
               Type of random forest: regression
                     Number of trees: 50
No. of variables tried at each split: 4

          Mean of squared residuals: 1.779478
                    % Var explained: 41.76
```

partialPlot(model, pred.data = data,x.var =)



图表 15 随机森林回归估计结果

自适应提升法回归估计结果

```
install.packages("gbm")
library(gbm)#AdaBoost 回归
formula <- Innovation ~ Digit + Size + Lev + PPE + ROA + Cash + SOE + Boardsize
+ Indboard + BM + Dual + Age + Id + Year
set.seed(123)
model <- gbm(formula,
                    data = data,
                    distribution = "gaussian",
                    shrinkage = 0.01,
                    interaction.depth = 3,
```

```
        n.trees = 50,
        n.minobsinnode = 10,
        cv.folds = 5)
summary(model)
```

```
> summary(model)
                  var   rel.inf
Size             Size 77.273018
Age               Age 11.645551
PPE               PPE  8.803018
Year             Year  2.278412
Digit           Digit  0.000000
Lev               Lev  0.000000
ROA               ROA  0.000000
Cash             Cash  0.000000
SOE               SOE  0.000000
Boardsize Boardsize  0.000000
Indboard   Indboard  0.000000
BM                BM  0.000000
Dual            Dual  0.000000
Id                Id  0.000000
```

图表 16 自适应提升法回归估计结果

研究 3：使用非参数方法对模型参数估计进行优化

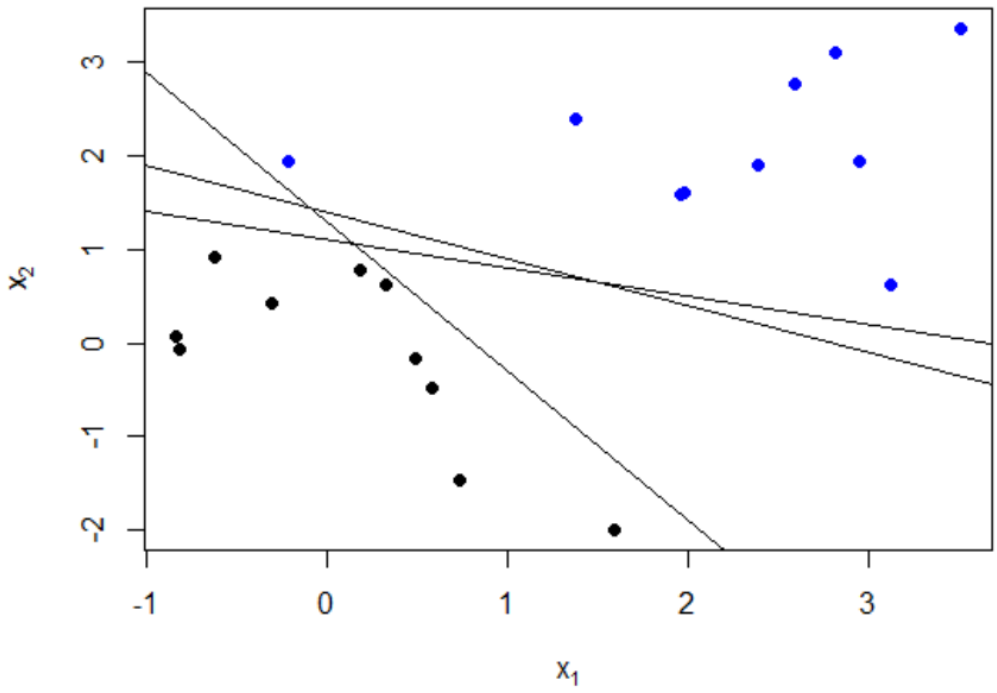
向量机

假设有 p 个特征变量，则分离超平面 L 的方程可写为

$$\beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p = \beta_0 + \boldsymbol{\beta}'\boldsymbol{x} = 0$$

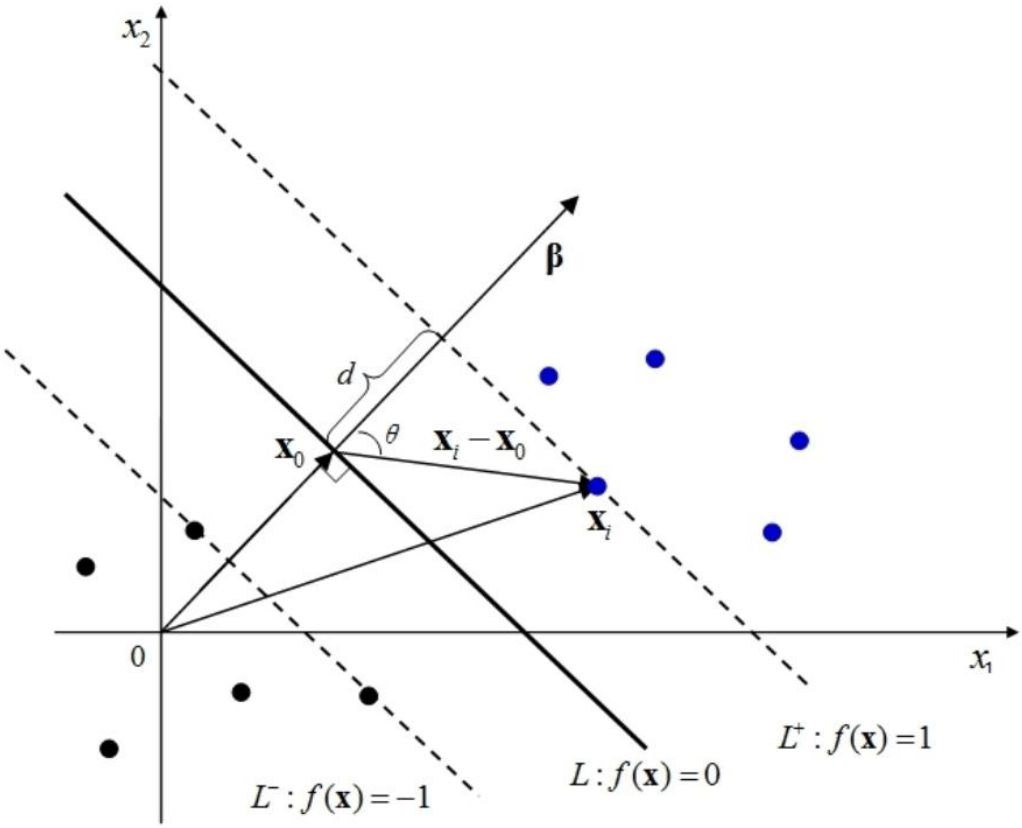
显然，此分离超平面 L 将 p 维特征空间（feature space） 一分为二。可见如果 $\beta_0 + \boldsymbol{\beta}'\boldsymbol{x} > 0$，则观测值 $\boldsymbol{x}$ 落于超平面 L 的一边。反之，如果 $\beta_0 + \boldsymbol{\beta}'\boldsymbol{x} < 0$，则观测值 $\boldsymbol{x}$ 落于超平面 L 的另一边。进一步，$|\boldsymbol{\beta}_0 + \boldsymbol{\beta}'\boldsymbol{x}|$（绝对值）的大

小可用于度量观测值 $\boldsymbol{x}$ 到超平面 L 距离远近。如果两类数据之间存在分离超平面，则称数据为线性可分（linearly separable）。然而，在线性可分的情况下，分离超平面一般并不唯一，因为总可以稍微移动超平面，而依然将两类数据分离，具体参见图 10。



图表 17 分离超平面

针对分离超平面不唯一的问题，一种解决方法是使得分离超平面离两类数据尽量远。具体来说，希望在两类数据之间有一条“隔离带”，而且这条隔离带要越宽越好。这就是所谓最大间隔分类器（maximal margin classifier），俗称为最宽街道法（widest street approach），即在两类数据之间建一条最宽的街道。



图表 18 观测数据到分离超平面的符号距离

记法向量 $\boldsymbol{\beta}$ 方向与超平面 L 的交点为 $\boldsymbol{x}_0$，则从观测值 $\boldsymbol{x}_i$ 到超平面 L 的最短（垂直）距离为：

$$\begin{aligned} d(\boldsymbol{x}_i, L) &= \|\boldsymbol{x}_i - \boldsymbol{x}_0\| \cos\theta \\ &= \|\boldsymbol{x}_i - \boldsymbol{x}_0\| \cdot \frac{\boldsymbol{\beta}'(\boldsymbol{x}_i - \boldsymbol{x}_0)}{\|\boldsymbol{x}_i - \boldsymbol{x}_0\| \, \|\boldsymbol{\beta}\|} \\ &= \frac{\boldsymbol{\beta}'(\boldsymbol{x}_i - \boldsymbol{x}_0)}{\|\boldsymbol{\beta}\|} \\ &= \underbrace{\frac{\boldsymbol{\beta}'\boldsymbol{x}_i - \boldsymbol{\beta}'\boldsymbol{x}_0}{\|\boldsymbol{\beta}\|}}_{d(\boldsymbol{x}_i, L) = \frac{\beta_0 + \boldsymbol{\beta}'\boldsymbol{x}_i}{\|\boldsymbol{\beta}\|} = \frac{f(\boldsymbol{x}_i)}{\|\boldsymbol{\beta}\|}} \end{aligned}$$

不失一般性，由于 $f(\boldsymbol{x}) = \beta_0 + \boldsymbol{\beta}'\boldsymbol{x}$ 为线性函数，因此，可以通过线性变换（比如，乘以某常数 c），使得对于所有在正间隔 L^+ 上样本点都有 $f(\boldsymbol{x}_i) = 1$；而且对于所有在负间隔 L^- 上样本点都有 $f(\boldsymbol{x}_i) = -1$。考虑正间隔 L^+ 上

的某样本点 x^*，则 x^* 到分离超平面 L 的距离的两倍，即为正间隔 L^+ 与负间隔 L^- 之间的“最大间隔”（maximal margin）：

$$2d(\boldsymbol{x}^*, L) = 2\frac{f(\boldsymbol{x}^*)}{\| \boldsymbol{\beta} \|} = \frac{2}{\| \boldsymbol{\beta} \|}$$

其中，$f(\boldsymbol{x}^*) = 1$，因为 $\boldsymbol{x}^*$ 位于正间隔 L^+上。式（8）为我们想要最大化的目标函数，而约束条件则是所有样本点都能正确分类。在完全正确分类情况下，对于所有“$y_i = 1$”的正样例，都有 $f(\boldsymbol{x}_i) \geqslant 1$，故 $y_i f(\boldsymbol{x}_i) \geqslant 1$；反之，对于所有 “$y_i = -1$”的负样例，都有 $f(\boldsymbol{x}_i) \leqslant -1$，故依然 $y_i f(\boldsymbol{x}_i) \geqslant 1$ 。因此，无论是正样例，还是负样例，约束条件都是 $y_i f(\boldsymbol{x}_i) \geqslant 1$ 。这正是令 $y_i \in \{-1,1\}$ 的方便之处。

求解最大间隔的超平面的约束极值问题为

$$\begin{aligned} \max_{\boldsymbol{\beta}, \beta_0} \quad & \frac{2}{\| \boldsymbol{\beta} \|} \\ \text{s. t.} \quad & y_i f(\boldsymbol{x}_i) \geqslant 1, i = 1, \cdots, n \end{aligned}$$

最大化 $\frac{2}{\|\boldsymbol{\beta}\|}$ 等价于最小化 $\| \boldsymbol{\beta} \|$ ，而后者又等价于最小化 $\frac{1}{2}\| \boldsymbol{\beta} \|^2 = \frac{1}{2}\boldsymbol{\beta}'\boldsymbol{\beta}$ 。将“$f(\boldsymbol{x}_i) = \boldsymbol{\beta}_0 + \boldsymbol{\beta}'\boldsymbol{x}_i$”代入约束条件，则最优化问题可写为

$$\begin{aligned} \min_{\boldsymbol{\beta}, \beta_0} \quad & \frac{1}{2}\boldsymbol{\beta}'\boldsymbol{\beta} \\ \text{s.t.} \quad & y_i(\beta_0 + \boldsymbol{\beta}'\boldsymbol{x}_i) \geqslant 1, i = 1, \cdots, n \end{aligned}$$

由于目标函数 $\frac{1}{2}\boldsymbol{\beta}'\boldsymbol{\beta} = \frac{1}{2}\left(\boldsymbol{\beta}_1^2 + \cdots + \boldsymbol{\beta}_p^2\right)$ 为二次型，约束条件 $y_i(\boldsymbol{\beta}_0 + \boldsymbol{\beta}'\boldsymbol{x}_i) \geqslant 1$ 为线性不等式约束，故为“凸二次规划”（convex quadratic programming）问题。为求解此问题，引入“原问题”（primal problem）的拉格朗日乘子函数 L_P：

$$\begin{aligned} \min_{\boldsymbol{\beta}, \beta_0, \boldsymbol{\alpha}} L_P(\boldsymbol{\beta}, \beta_0, \boldsymbol{\alpha}) \quad &= \frac{1}{2}\boldsymbol{\beta}'\boldsymbol{\beta} - \sum_{i=1}^{n} \alpha_i [y_i(\beta_0 + \boldsymbol{\beta}'\boldsymbol{x}_i) - 1] \\ &= \frac{1}{2}\boldsymbol{\beta}'\boldsymbol{\beta} - \beta_0 \sum_{i=1}^{n} \alpha_i y_i - \boldsymbol{\beta}' \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x}_i + \sum_{i=1}^{n} \alpha_i \end{aligned}$$

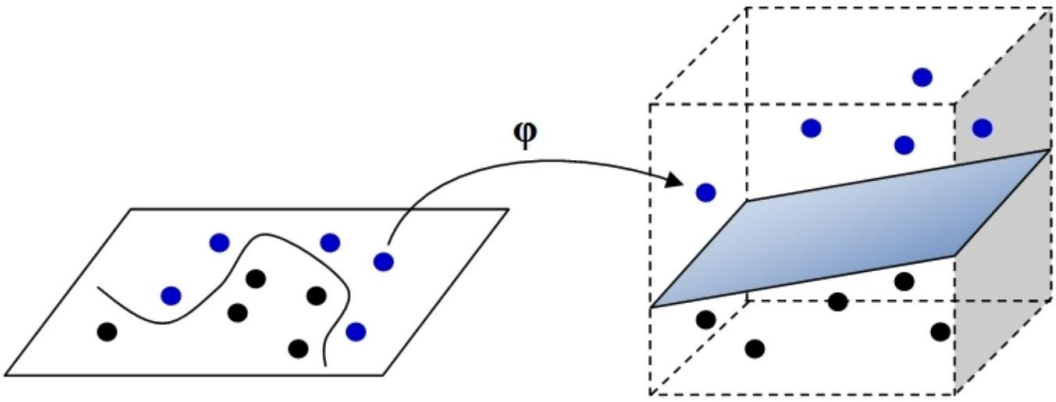
其中，$\boldsymbol{\alpha} \equiv (\alpha_1 \cdots \alpha_n)'$ 为对应于约束条件 $y_i(\beta_0 + \boldsymbol{\beta}'\boldsymbol{x}_i) \geqslant 1$ 的 n 个拉格朗日乘子。可以证明，原问题的最优解为拉格朗日乘子函数（11）的 “鞍点”，故单独从拉格朗日乘子 $\boldsymbol{\alpha}$ 来看，则为最大化问题。

将一阶条件代回拉格朗日函数，可得到“对偶问题”（dual problem）L_D：

$$\begin{aligned}
\max_{\boldsymbol{\alpha}} L_D = & \frac{1}{2}\underbrace{\left(\sum_{i=1}^{n}\alpha_i y_i \boldsymbol{x}_i\right)'}_{=\hat{\boldsymbol{\beta}}'}\underbrace{\left(\sum_{i=1}^{n}\alpha_i y_i \boldsymbol{x}_i\right)}_{=\hat{\boldsymbol{\beta}}} - \beta_0 \underbrace{\sum_{i=1}^{n}\alpha_i y_i}_{=0} \\
& - \underbrace{\left(\sum_{i=1}^{n}\alpha_i y_i \boldsymbol{x}_i\right)'}_{=\hat{\boldsymbol{\beta}}'}\underbrace{\left(\sum_{i=1}^{n}\alpha_i y_i \boldsymbol{x}_i\right)}_{=\hat{\boldsymbol{\beta}}} + \sum_{i=1}^{n}\alpha_i \\
= & \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\left(\sum_{i=1}^{n}\alpha_i y_i \boldsymbol{x}_i\right)'\left(\sum_{i=1}^{n}\alpha_i y_i \boldsymbol{x}_i\right) \\
= & \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\left(\sum_{i=1}^{n}\alpha_i y_i \boldsymbol{x}_i'\right)\left(\sum_{j=1}^{n}\alpha_j y_j \boldsymbol{x}_j\right) \\
= & \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \boldsymbol{x}_i'\boldsymbol{x}_j \\
= & \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle
\end{aligned}$$

其中，$\boldsymbol{x}_i'\boldsymbol{x}_j$ 为 $\boldsymbol{x}_i$ 与 $\boldsymbol{x}_j$ 的内积，可记为 $\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$。这意味着，特征向量 $\{\boldsymbol{x}_i\}_{i=1}^n$ 仅通过相互之间内积的方式而影响最优解。这为第在支持向量机中使用“核技巧”（kernel trick）提供了方便。最大化问题是关于拉格朗日乘子 $\boldsymbol{\alpha}$ 的二次（型）规划问题。求解此对偶问题，并将所得解 $\hat{\boldsymbol{\alpha}} = (\hat{\alpha}_1 \cdots \hat{\alpha}_n)'$ 代回最优 $\boldsymbol{\beta}$ 表达式可得，$\hat{\boldsymbol{\beta}} = \sum_{i=1}^{n}\hat{\alpha}_i y_i \boldsymbol{x}_i$。对于截距项 β_0，可通过支持向量来求解。

更一般地，对于决策边界非线性数据，考虑对特征向量 $\boldsymbol{x}_i$ 进行变换，比如将 $\boldsymbol{x}_i$ 变换为 $\boldsymbol{\varphi}(\boldsymbol{x}_i)$；其中，$\boldsymbol{\varphi}(\boldsymbol{x}_i)$ 为多维函数（维度可以高于 $\boldsymbol{x}_i$），甚至可以无限维函数。这意味着，可将训练样本 $\{\boldsymbol{x}_i, y_i\}_{i=1}^n$ 变换为 $\{\boldsymbol{\varphi}(\boldsymbol{x}_i), y_i\}_{i=1}^n$，目的是希望在 $\boldsymbol{\varphi}(\boldsymbol{x}_i)$ 的特征空间（feature space）中，可以得到线性可分的情形，参见图 12。但难点在于，对于高维数据，一般不知道变换 $\boldsymbol{\varphi}(\cdot)$ 的具体形式。



图表 19 特征变换

根据上述推导，支持向量机的估计结果仅依赖于 $\langle \boldsymbol{\varphi}(\boldsymbol{x}_i), \boldsymbol{\varphi}(\boldsymbol{x}_j) \rangle$，即 $\boldsymbol{\varphi}(\boldsymbol{x}_i)$ 与 $\boldsymbol{\varphi}(\boldsymbol{x}_j)$ 的内积，而不必知道 $\boldsymbol{\varphi}(\cdot)$。为此，将此内积定义为如下核函数（kernel function）：

$$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) \equiv \langle \boldsymbol{\varphi}(\boldsymbol{x}_i), \boldsymbol{\varphi}(\boldsymbol{x}_j) \rangle = \boldsymbol{\varphi}(\boldsymbol{x}_i)' \boldsymbol{\varphi}(\boldsymbol{x}_j)$$

因此，我们只要直接指定核函数 $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$ 的具体形式即可；而无须预先知道 $\varphi(\cdot)$，再来计算 $\langle \boldsymbol{\varphi}(\boldsymbol{x}_i), \boldsymbol{\varphi}(\boldsymbol{x}_j) \rangle$（此内积可能不易计算，尤其在高维空间中）。这种方法称为核技巧（kernel trick）。显然，$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$ 关于 x_i 与 x_j 是对称的，即 $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \kappa(\boldsymbol{x}_j, \boldsymbol{x}_i)$。

对于线性不可分的数据，可以放松对于约束条件的要求，即只要求分离超平面将大多数观测值正确分离，而允许少量错误分类（或落入间隔之内）观测值。为此，可引入“松弛变量”（slack variable） $\xi_i \geqslant 0$，而将约束条件变为 $y_i(\boldsymbol{\beta}_0 + \boldsymbol{\beta}' \boldsymbol{x}_i) \geqslant 1 - \xi_i$；但对所有观测值松弛变量之和 $\sum_{i=1}^{n} \xi_i$ 进行惩罚。此最小化问题可写为

$$\min_{\boldsymbol{\beta}, \beta_0 \cdot \xi_i} \frac{1}{2} \boldsymbol{\beta}' \boldsymbol{\beta} + C \sum_{i=1}^{n} \xi_i$$
$$\text{s. t. } y_i(\beta_0 + \boldsymbol{\beta}' \boldsymbol{x}_i) \geqslant 1 - \xi_i, \xi_i \geqslant 0, \forall i$$

其中，$C \geqslant 0$ 为调节变量（C 表示 Cost），用来惩罚过大的松驰变量总和（太多错误）。由于存在松驰变量 $\xi_i \geqslant 0$，故允许 $\boldsymbol{x}_i$ 落在间隔的错误一边（wrong side of the margin），甚至分离超平面的错误一边（wrong side of the separating hyperplane），因此称为软间隔分类器（soft margin classifier），或支持向量分类器 （support vector classifier）。特别地，如果 $0 < \xi_i < 1$，则 x_i 落在间隔错误一边 （即间隔之内），但依然在超平面的正确一边。如果 $\xi_i = 1$，则 x_i 正好落在超平面上。而如果 $\xi_i > 1$，则 $\boldsymbol{x}_i$ 落在分离超平面的错误一边。对于软间隔分类器，所有在间隔上、间隔内、分类错误的样本点都是支持向量，因为它们都对最优解有影响。对于软间隔分类器的求解，依然可使用拉格朗日函数，但须引入松驰变量 ξ_i 的拉格朗日乘子 μ_i：

$$\min_{\boldsymbol{\beta},\beta_0,\boldsymbol{\alpha},\boldsymbol{\mu}} L_p(\boldsymbol{\beta},\beta_0,\boldsymbol{\alpha},\boldsymbol{\mu}) = \frac{1}{2}\boldsymbol{\beta}'\boldsymbol{\beta} + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i[y_i(\beta_0 + \boldsymbol{\beta}'\boldsymbol{x}_i) - 1 + \xi_i] - \sum_{i=1}^{n}\mu_i\xi_i$$

此问题求解方法与最优解的形式，均类似于硬间隔分类器，在此不再赘述；推导细节可参考周志华（2016）。

对于支持向量机，可以证明，它使用的替代损失函数为如下合页损失函数（hinge loss function）：

$$\ell_{hinge}(z_i) = \begin{cases} 0 & \text{当 } z_i \geqslant 1 \\ 1 - z_i & \text{当 } z_i < 1 \end{cases}$$

其中，如果裕度 $z_i \geqslant 1$（分类正确，且在间隔之外），则损失为 0（不进行惩罚）；反之，如果 $z_i < 1$（在间隔之内，甚至分类错误），则其损失为 $1 - z_i$（斜率为 -1，故惩罚力度为 1 对 1）。由于此函数的形状类似于门框的合页，故称其为“合页损失函数”。更简洁地，可将分段函数写为统一的表达式：

$$\ell_{hinge}(z_i) = max(0, 1 - z_i)$$

将合页损失函数代入目标函数，并记松弛变量 $\xi_i = 1 - z_i$，可得

$$\min_{\boldsymbol{\beta},\beta_0} \frac{1}{2}\boldsymbol{\beta}'\boldsymbol{\beta} + C\sum_{i=1}^{n} \ell_{\text{hinge}}(z_i) = \frac{1}{2}\boldsymbol{\beta}'\boldsymbol{\beta} + C\sum_{i=1}^{n} \max(0,1-z_i)$$
$$= \frac{1}{2}\boldsymbol{\beta}'\boldsymbol{\beta} + C\sum_{i=1}^{n} (1-z_i)I(1-z_i \geqslant 0)$$
$$= \frac{1}{2}\boldsymbol{\beta}'\boldsymbol{\beta} + C\sum_{i=1}^{n} \xi_i I(\xi_i \geqslant 0)$$

不难看出，该式与软间隔分类器最优化问题等价。尽管合页损失函数依然不光滑（在 $z_i = 1$ 处有尖点），但至少是连续的凸函数，其数学性质优于 $0-1$ 损失函数。而光滑的替代损失函数则包括“指数损失函数”（exponential loss function，用于 AdaBoost 算法），以及“逻辑损失函数”（logistic loss function，用于二分类问题的梯度提升法）。

支持向量回归（support vector regression，简记 SVR）的基本思想是，将支持向量机的合页损失函数移植到回归问题。记回归函数（超平面） 为 $f(\boldsymbol{x}) = \beta_0 + \boldsymbol{x}'\boldsymbol{\beta}$，并以此函数预测连续型响应变量 y。SVR 的目标函数为：

$$\min_{\boldsymbol{\beta},\boldsymbol{\beta}_0} \frac{1}{2}\boldsymbol{\beta}'\boldsymbol{\beta} + C\sum_{i=1}^{n} \ell_{\varepsilon}[y_i - f(\boldsymbol{x}_i)]$$

其中，$C > 0$ 为正则化参数（regularization parameter），$z_i \equiv y_i - f(\boldsymbol{x}_i)$ 为残差（residual）；而 $\ell_{\varepsilon}(\cdot)$ 为 ε-不敏感损失函数（ε-insensitive loss function）：

$$\ell_{\varepsilon}(z_i) = \begin{cases} 0 & \text{当 } |z_i| \leqslant \varepsilon \\ |z_i| - \varepsilon & \text{当 } |z_i| > \varepsilon \end{cases}$$

其中，$\varepsilon > 0$ 也是调节参数。

如果将 $\frac{1}{2}\boldsymbol{\beta}'\boldsymbol{\beta}$ 视为惩罚项，则 SVR 的目标函数类似于岭回归：

$$\min_{\boldsymbol{\beta},\beta_0} C\sum_{i=1}^{n} \ell_{e}[y_i - f(\boldsymbol{x}_i)] + \frac{1}{2}\boldsymbol{\beta}'\boldsymbol{\beta}$$

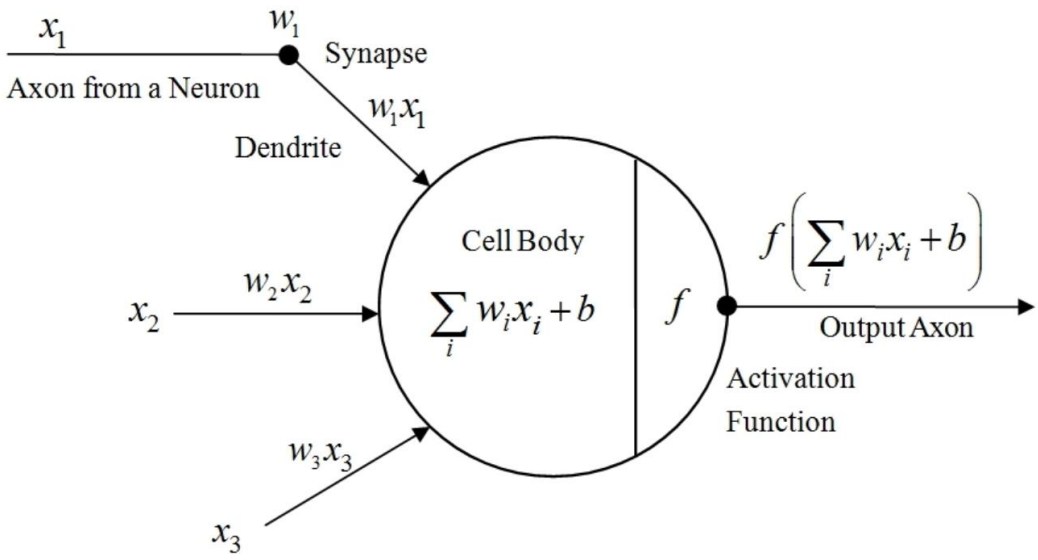
岭回归的区别在于，岭回归以平方损失函数替代 ε-不敏感损失函数 $\ell_s(\cdot)$。由于使用 ε-不敏感损失函数，这种支持向量回归也称为 ε-回归（ε-regression）。

感知机

对于二分类问题，考虑使用分离超平面“$b+\boldsymbol{w}'\boldsymbol{x}=0$” 进行分类，而响应变量 $y\in\{1,-1\}$。如果 $b+w'\boldsymbol{x}>0$ 则预测 $y=1$。反之，如果 $b+w'\boldsymbol{x}<0$ 则预测 $y=-1$（$b+\ \boldsymbol{w}'\boldsymbol{x}=0$ 可随意预测）。显然，正确分类要求 $y_i(b+\boldsymbol{w}'\boldsymbol{x}_i)>0$ 。反之如果 $y_i(b+\boldsymbol{w}'\boldsymbol{x}_i)<0$，则为错误分类。从某个初始值 $(\boldsymbol{w}_0,b_0)$ 出发，感知机希望通过调整参数 $(\boldsymbol{w},b)$ ，来使得模型的错误分类最少。具体来说，感知机目标函数为最小化所有分类错误观测值的“错误程度”之和（即“负裕度”（negative margin）$-y_if(x_i)$）：

$$\min_{\boldsymbol{w},b}L(\boldsymbol{w},b)=-\sum_{i\in M}y_i(b+\boldsymbol{w}'\boldsymbol{x}_i)$$

其中， M 为所有错误分类（misclassified）个体下标的集合。



图表 20 人工神经网络的神经元示意图

为简单起见，假定 M 不变（如果 M 有变，在迭代过程中更新即可），则此目标函数梯度向量为

$$\frac{\partial L(\boldsymbol{w},b)}{\partial \boldsymbol{w}}=-\sum_{i\in M}y_i\boldsymbol{x}_i$$

$$\frac{\partial L(\boldsymbol{w},b)}{\partial b}=-\sum_{i\in M}y_i$$

使用梯度下降法，沿着负梯度方向更新，则参数的更新规则为

$$\boldsymbol{w}\leftarrow\boldsymbol{w}+\eta\sum_{i\in M}y_i\boldsymbol{x}_i$$

$$b\leftarrow b+\eta\sum_{i\in M}y_i$$

其中，η 为“学习率”（learning rate），也称“步长”（step size）。通过迭代，可使损失函数 $L(\boldsymbol{w},b)$ 不断减小，直到变为 0 为止。

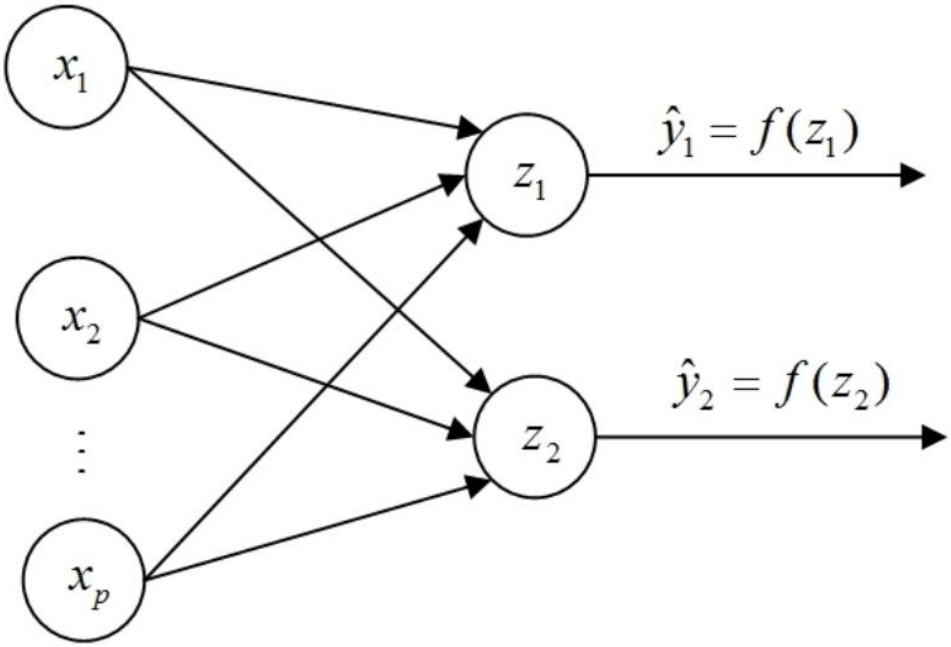
感知机算法的直观解释为，当一个样本点被错误分类，即出现在分离超平面“$b+\boldsymbol{w}'\boldsymbol{x}=0$”的错误一侧时，则调整参数 $(\boldsymbol{w},b)$，使得分离超平面向该误分类点一侧移动，以减少此误分类点与超平面的距离，直至正确分类为止。

可以证明，对于线性可分的数据，则感知机一定会收敛，参见李航（2019）。这表明，只要给予足够的数据，感知机具备学得参数 $(\boldsymbol{w},b)$ 的能力，仿佛拥有 “感知”世界的能力（比如，自动将事物分类），故名“感知机”。然而，从不同的初始值出发，一般会得到不同的分离超平面，无法得到唯一解。另一方面，如果数据为线性不可分，则感知机的算法不会收敛。感知机更严重的缺陷是，它的决策边界依然为线性函数。

神经网络

只要引入多层神经元，经过两个及以上的非线性激活函数迭代之后，即可得到非线性决策边界。在此，非线性的激活函数是关键；因为如果使用线性的激活函数，则无论叠加或嵌套多少次（相当于微积分的复合函数），所得结果一定还是线性函数。

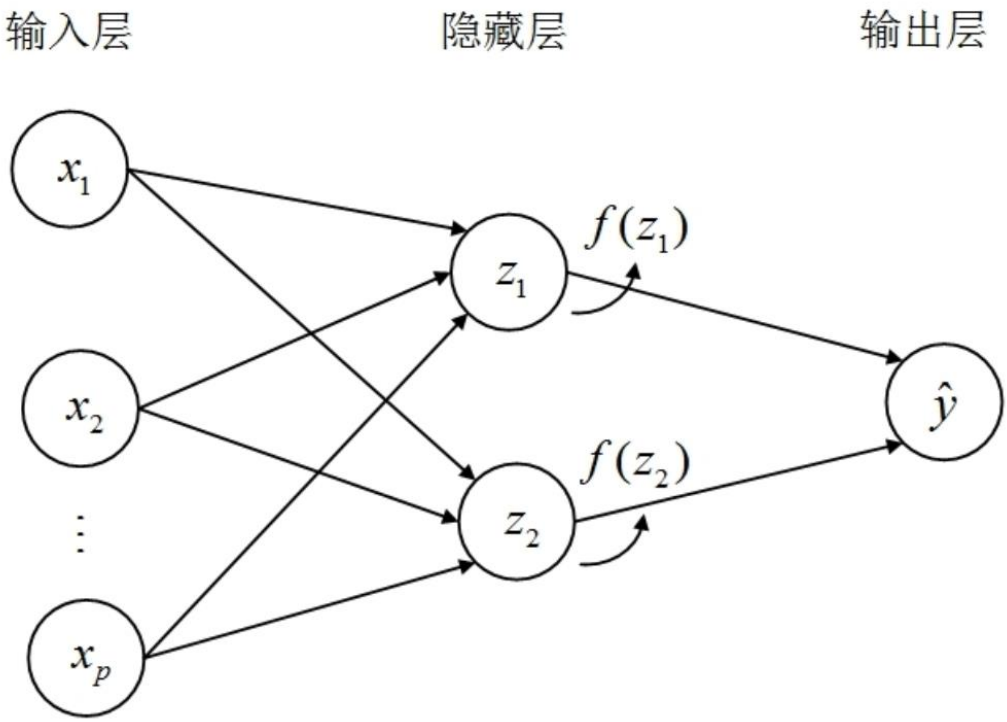
首先, 考虑具有多个输出结果（multi-output）的感知机，如图所示。



图表 21 多个输出结果的感知机

图中，共有两个输出（响应）变量，$\hat{y}_1$ 与 $\hat{y}_2$ 。其中，$z_1 \equiv b_1 + \sum_{i=1}^{p} w_{i1} x_i$ 与 $z_2 \equiv b_2 + \sum_{i=1}^{p} w_{i2} x_i$ ，均为在施加激活函数之前的加总值；而 $f(\cdot)$ 为激活函数。

其次，图中的多个输出结果可重新作为输入变量，经过加权求和后，再次施以激活函数。



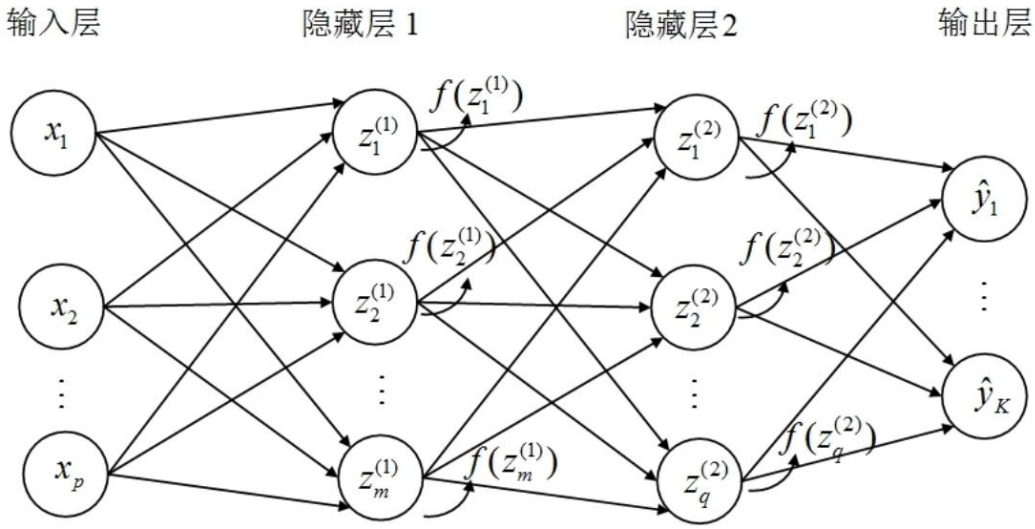
图表 22 多层感知机

最终输出结果为：

$$\hat{y} = f\left[b^{(2)} + w_1^{(2)} f(z_1) + w_2^{(2)} f(z_2)\right]$$

即对 $f(z_1)$ 与 $f(z_2)$ 再次加权求和，然后再施加激活函数 $f(\cdot)$ 。显然，该函数所对应的决策边界为非线性的。图中，最左边为输入层 （input layer），中间为隐藏层（hidden layer），而最右边为输出层（output layer）。之所以将中间层称为“隐藏层”，因为该层的计算在算法内部进行，从外面并不可见。

当然，隐藏层可以有更多的神经元，而不仅是图 14 中的两个隐藏神经元; 而输出层也可有多个输出结果。在概念上，只要包含一个隐藏层，即为多层感知机（Multilayer Perceptron，简记 MLP），也就是“神经网络模型”。更一般地，神经网络模型可以有多个隐藏层。



图表 23 双隐藏层神经网络

如图所示，即为基本的神经网络结构。这种标准神经网络，称为前馈神经网络（feed forward neural network），因为输入从左向右不断前馈；也称为全连接神经网络（fully connected neural network），因为相邻层的所有神经元都相互连接。当然，针对特殊数据类型可能还需要一些特别的网络结构，比如，卷积神经网络（适用于图像识别）、循环神经网络（适用于自然语言等时间序列）等。另外，如果神经网络的隐藏层很多，则称为深度神经网络（deep neural networks），简称深度学习（deep learning）。

Cybenko （1988）与 Hornik, Stinchcombe and White（1989）使用“泛函分析”（functional analysis），证明了神经网络的“通用近似定理”（universal approximation theorem）。其主要结论为，包含单一隐藏层的前馈神经网络模型，只要其神经元数目足够多，则可以以任意精度逼近任何一个在有界闭集上定义的连续函数。实际上，包含单隐藏层的前馈神经网络所代表的函数可写为

$$G(\boldsymbol{x}) = \sum_{i=1}^{m} \alpha_i f(\boldsymbol{w}_i' \boldsymbol{x} + b_i)$$

其中，$(\boldsymbol{w}_i, b_i)$ 为第 i 个神经元的权重与偏置参数，$f(\cdot)$ 为激活函数，α_i 为连接隐藏层与输出层的参数，而 m 为神经元的数目。通用近似定理表明，形如 $G(\boldsymbol{x})$ 的函数在定义于有界闭集上的连续函数的集合中是“稠密的”（dense），

这意味着对于任意有界闭集上的连续函数，都可以找到形如 $G(\boldsymbol{x})$ 的函数（也即单隐层的前馈神经网络），使二者的距离任意接近。

在文献中，通用近似定理的激活函数可采取不同形式的非线性函数，既可以包括非常数（nonconstant）、有界（bounded）且单调递增的连续函数（例如 S 形函数、双曲正切函数），也包括无界（unbounded）且单调递增的连续函数（例如 ReLU），甚至允许不连续函数（例如阶梯函数）。在某种意义上，通用近似定理可表明，神经网络可作为“万能”函数来使用。然而，通用近似定理只是说明，对于任意有界闭集上的连续函数，都存在与它非常接近的单隐层前馈神经网络；但没有未给出找到此神经网络的方法，也不知道究竟需要多少个神经元，才能达到既定的接近程度 ε_0 当然, 在实际应用中，一般并不知道真实函数 $g(\boldsymbol{x})$，而我们更关心神经网络 $G(x)$ 的泛化能力。由于神经网络的强大拟合能力，反而容易在训练集上过拟合，故需要避免过拟合，以降低测试误差。

神经网络的损失函数的一般形式可写为

$$\boldsymbol{W}^* = \underset{\boldsymbol{W}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} L[y_i, G(\boldsymbol{x}_i; \boldsymbol{W})]$$

其中，参数矩阵 $\boldsymbol{W}$（也可排为参数向量）包含神经网络模型的所有参数（包括偏置），其每一列对应于神经网络每一层的参数；W^* 为 $\boldsymbol{W}$ 的最优值，$G(\boldsymbol{x}_i; W)$ 为神经网络对观测值 $\boldsymbol{x}_i$ 所作的预测（即 $\hat{y}_i$），而 $L(y_i, \hat{y}_i)$ 为损失函数。式中，整个样本的损失函数为每个观测值的损失 $L[y_i, G(\boldsymbol{x}_i; \boldsymbol{W})]$ 的平均值。对于响应变量为连续的回归问题，一般使用平方损失函数（squared loss function） 最小化训练集的均方误差：

$$\boldsymbol{W}^* = \underset{\boldsymbol{W}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} [y_i - G(\boldsymbol{x}_i; \boldsymbol{W})]^2$$

此外，在训练神经网络之前，一般建议将全部特征变量归一化（normalization，即令最小值变为 0 而最大值变为 1） 或标准化（standardization，即均值变为 0 而标准差变为 1）。这是因为，如果特征变量

取值范围差别较大，会影响神经网络的权重参数，不利于神经网络训练。对于回归问题，若对特征变量进行了归一化处理，则所有特征变量 $x \in [0,1]$，此时建议也将响应变量作归一化处理，以便于模型的训练与预测。另外，在选择参数矩阵 $\boldsymbol{W}$ 的初始值 $\boldsymbol{W}_0$ 时，一般并不会将其所有元素都设为相同的取值（比如，都设为 0 或 1 ），而通常从标准正态分布 $N(0,1)$ 或取值介于 $[-0.7,0.7]$ 的均匀分布中随机抽样，这样有利于不同神经元之间的分化，避免趋同。

支持向量机回归估计结果

```
install.packages("e1071")
library(e1071)
model <- svm(Innovation ~ Digit + Size + Lev + PPE + ROA + Cash + SOE +
Boardsize + Indboard + BM + Dual + Age + Id + Year,
                    data = data, type = "eps-regression")
print(model)
```

```
Call:
svm(formula = Innovation ~ Digit + Size + Lev + PPE + ROA + Cash + SOE +
    Boardsize + Indboard + BM + Dual + Age + Id + Year, data = data, type = "eps-regressio
n")


Parameters:
   SVM-Type:  eps-regression
 SVM-Kernel:  radial
       cost:  1
      gamma:  0.07142857
    epsilon:  0.1


Number of Support Vectors:  27064
```

图表 24 支持向量机回归估计结果

人工神经网络回归估计结果

```
install.packages("neuralnet")
library(neuralnet)
formula <- Innovation ~ Digit + Size + Lev + PPE + ROA + Cash + SOE + Boardsize
+ Indboard + BM + Dual + Age + Id + Year
model <- neuralnet(formula, data = data, hidden=3)#人工神经网络回归
```

plot(model)

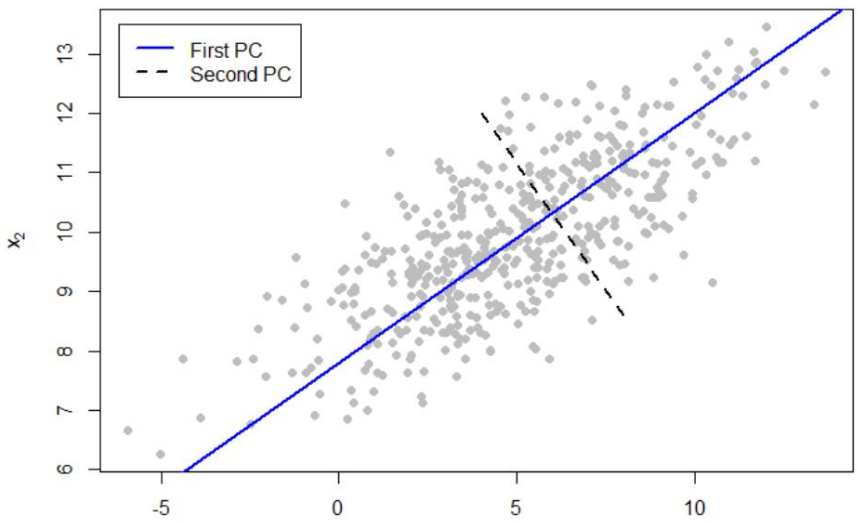
研究 4：使用非监督学习方法对模型聚类进行优化

主成分分析

给定 p 维随机向量 $\boldsymbol{x} \equiv (x_1 x_2 \cdots x_p)'$，要找到 $\boldsymbol{x}$ 的一个线性组合 $\boldsymbol{a}'\boldsymbol{x}$，使它包含 $(x_1 x_2 \cdots x_p')$尽可能多的“信息”。进一步，希望找到以下 p 个线性组合，即主成分（principal component）：

$$z_1 = \boldsymbol{a}_1'\boldsymbol{x}, z_2 = \boldsymbol{a}_2'\boldsymbol{x}, \cdots, z_p = \boldsymbol{a}_p'\boldsymbol{x}$$

其中，第 1 个主成分 z_1 包含信息最多，第 2 个主成分 z_2 包含信息第二多，以此类推；而且这些主成分 $\{z_1, z_2, \cdots, z_p\}$ 之间均不相关。相应地，将组合系数 $\boldsymbol{a}_k \equiv (a_{k1} \cdots a_{kp})'$ 称为第 k 个主成分 z_k 的主成分载荷向量（principal component loading vector），其每个分量反映原变量 $(x_1, \cdots, x_p)$ 对于 z_k 的不同影响（即 loading）。一种方法是，希望数据沿着线性组合 $z_1 = \boldsymbol{a}_1'\boldsymbol{x}$ 的直线方向具有最大的变动幅度；换言之，此线性组合的方差 $Var(\boldsymbol{a}_1'\boldsymbol{x})$ 最大。



图表 25 主成分分析示意图

记随机向量 $\boldsymbol{x}$ 的协方差矩阵为 $\boldsymbol{\Sigma}$ 。根据夹心估计量（sandwich estimator） 的法则，对于任意 p 维常数向量 $\boldsymbol{a}$，则线性组合 $\boldsymbol{a}'\boldsymbol{x}$ 的方差为 $Var(\boldsymbol{a}'\boldsymbol{x}) = \boldsymbol{a}'\boldsymbol{\Sigma}\boldsymbol{a}$。显然，如果向量 $\boldsymbol{a}$ 的长度越大，则 $Var(\boldsymbol{a}'\boldsymbol{x})$ 越大。为此，将线性组合系数 $\boldsymbol{a}$ 标准化，要求其长度为 1，即 $\boldsymbol{a}'\boldsymbol{a} = 1$。由此可得以下约束极值问题：

$$\max_{\mathbf{a}} Var(\boldsymbol{a}'\boldsymbol{x}) = \boldsymbol{a}'\boldsymbol{\Sigma}\boldsymbol{a}$$
$$\text{s. t. } \boldsymbol{a}'\boldsymbol{a} = 1$$

即在满足约束条件 $\boldsymbol{a}'\boldsymbol{a} = 1$ 的情况下，最大化线性组合 $\boldsymbol{a}'\boldsymbol{x}$ 的方差 $Var(\boldsymbol{a}'\boldsymbol{x})$。此约束极值问题的拉格朗日函数为

$$\max_{a,\lambda} L(\boldsymbol{a}, \lambda) = \boldsymbol{a}'\boldsymbol{\Sigma}\boldsymbol{a} - \lambda(\boldsymbol{a}'\boldsymbol{a} - 1)$$

对 $\boldsymbol{a}$ 进行向量微分，可得一阶条件：

$$\frac{\partial L}{\partial \boldsymbol{a}} = \mathbf{2}\boldsymbol{\Sigma}\boldsymbol{a} - 2\lambda\boldsymbol{a} = 0$$

经移项整理可得：

$$\boldsymbol{\Sigma}\boldsymbol{a} = \lambda\boldsymbol{a}$$

根据线性代数知识，最优解 $\boldsymbol{\lambda}$ 与 $\boldsymbol{a}$ 分别为协方差矩阵 $\boldsymbol{\Sigma}$ 的特征值（eigenvalue) 与特征向量（eigenvector）。

将一阶条件两边同乘 $\boldsymbol{a}'$，即可得到目标函数的表达式：

$$Var(\boldsymbol{a}'\boldsymbol{x}) = \boldsymbol{a}'\boldsymbol{\Sigma}\boldsymbol{a} = \lambda \underbrace{\boldsymbol{a}'\boldsymbol{a}}_{=\mathbf{1}} = \lambda$$

因此，对于第 1 个主成分 $z_1 = \boldsymbol{a}_1'\boldsymbol{x}$，应选择最大的特征值（记为 λ_1），使得 $Var(z_1) = \lambda_1$；而最优的 $\boldsymbol{a}_1$ 为 λ_1 相应的特征向量。进一步，由于协方差矩阵 $\boldsymbol{\Sigma}$ 必然半正定（positive semidefinite)，故可将其所有特征值按照从大到小排列：

$$\lambda_1 \geqslant \lambda_2 \geqslant \cdots \geqslant \lambda_p \geqslant 0$$

由此可得到所有的主成分：

$$z_1 = \boldsymbol{a}_1'\boldsymbol{x}, Var(z_1) = \lambda_1$$
$$z_2 = \boldsymbol{a}_2'\boldsymbol{x}, Var(z_2) = \lambda_2$$

其中，$\boldsymbol{a}_2$ 为特征值 λ_2 的特征向量，以此类推。可知，各主成分方差依次递减。我们还希望不同的主成分之间还没有相关性（即正交）。容易验证，对于任意 $i \neq j$，主成分 z_i 与主成分 z_j 的协方差为 0：

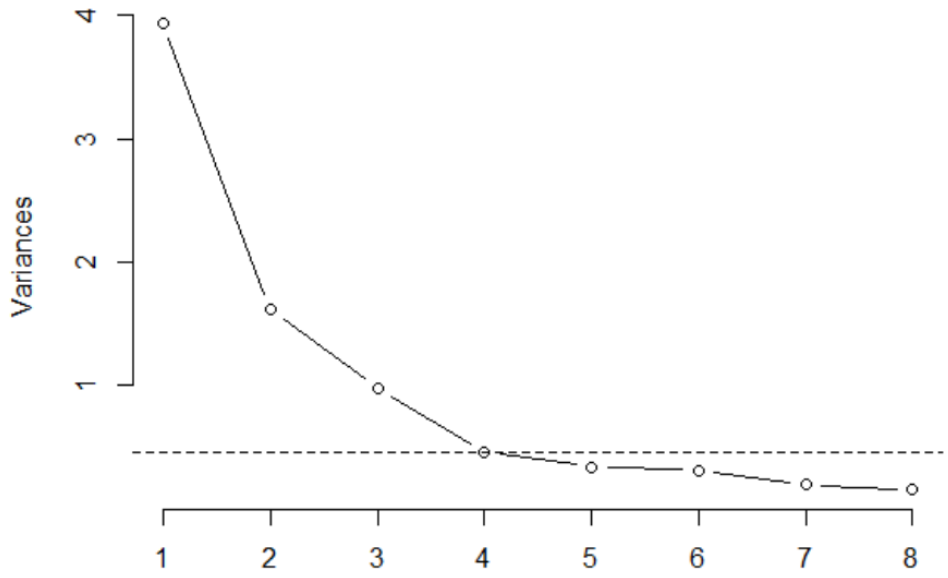
$$\begin{aligned}
Cov(z_i, z_j) &= Cov(\boldsymbol{a}_i'\boldsymbol{x}, \boldsymbol{a}_j'\boldsymbol{x}) \\
&= E[\boldsymbol{a}_i'\boldsymbol{x} - \boldsymbol{a}_i'E(\boldsymbol{x})]\left[\boldsymbol{a}_j'\boldsymbol{x} - \boldsymbol{a}_j'E(\boldsymbol{x})\right]' \\
&= E\{\boldsymbol{a}_i'[\boldsymbol{x} - E(\boldsymbol{x})][\boldsymbol{x} - E(\boldsymbol{x})]'\boldsymbol{a}_j\} \\
&= a_i'E[\boldsymbol{x} - E(\boldsymbol{x})][\boldsymbol{x} - E(\boldsymbol{x})]'\boldsymbol{a}_j \\
&= \boldsymbol{a}_i' Var(\boldsymbol{x})\, \boldsymbol{a}_j \\
&= \boldsymbol{a}_i'\boldsymbol{\Sigma}\boldsymbol{a}_j \\
&= \underbrace{\boldsymbol{a}_i'\lambda_j\boldsymbol{a}_j}_{\text{一阶条件}} \\
&= \underbrace{\lambda_j \underbrace{\boldsymbol{a}_i'\boldsymbol{a}_j}_{=0} = 0}_{\text{不同特征值的特征向量正交}}
\end{aligned}$$

其中，由于 a_i 与 a_j 分别为不同特征值 λ_i 与 λ_j 的相应特征向量，故根据线性代数知识可知，二者正交，即 $\boldsymbol{a}_i'\boldsymbol{a}_j = 0$。因此，所有主成分之间均不相关。

可将各分量方差之和 $\sum_{i=1}^{p} Var(\boldsymbol{x_i})$ 分解如下：

$$\begin{aligned}\sum_{i=1}^{p} Var(x_i) &= trace(\boldsymbol{\Sigma}) \\ &= trace(\boldsymbol{P\Lambda P'}) \\ &= \underbrace{trace\left(\boldsymbol{\Lambda}\underbrace{\boldsymbol{PP'}}_{=I}\right)}_{\text{迹乘法可交换次序且 } \boldsymbol{P} \text{ 为正交矩阵}} \\ &= trace(\boldsymbol{\Lambda}) \\ &= \sum_{i=1}^{p} \lambda_i = \sum_{i=1}^{p} Var(z_i)\end{aligned}$$

显然，各主成分的方差逐渐下降，即 $Var(z_1) \geqslant Var(z_2) \geqslant \cdots \geqslant Var(z_p)$ 。将 $[k, Var(z_k)], k = 1, \cdots, p$ 画图，即可得到所谓陡坡图（scree plot）



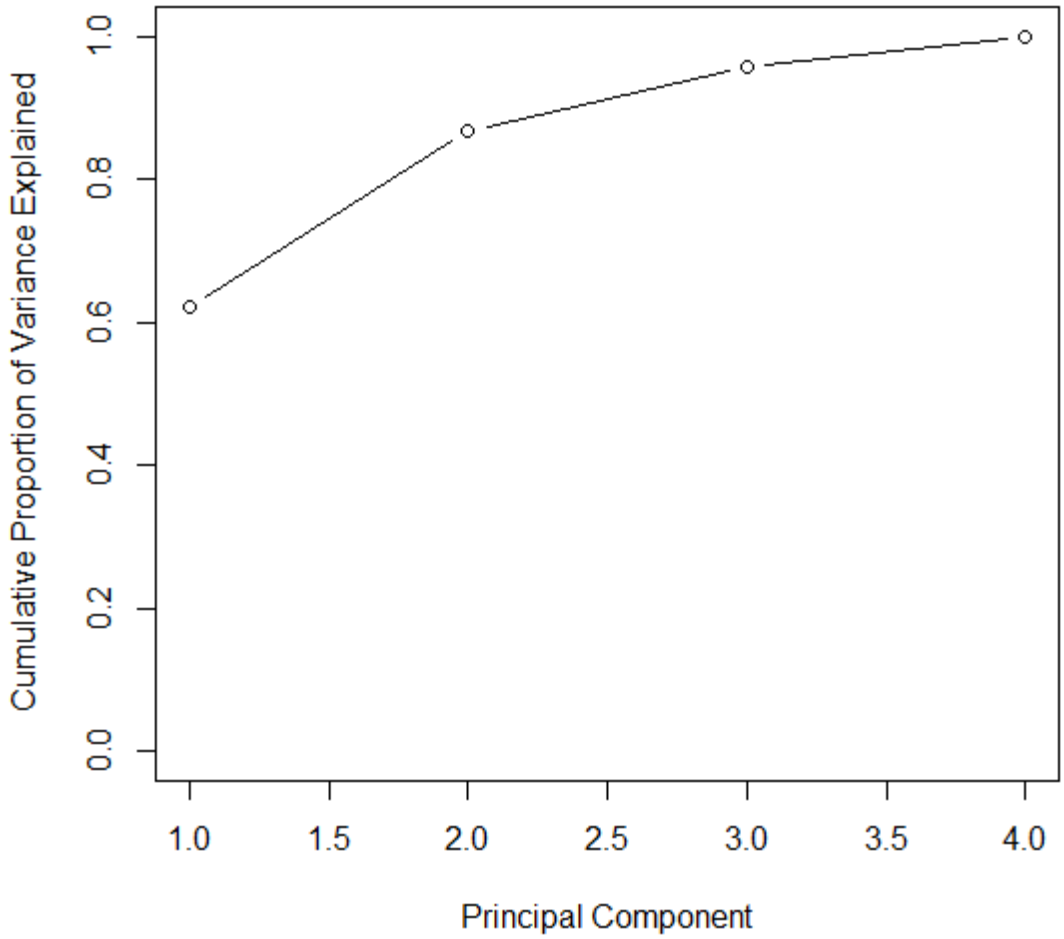
图表 26 陡坡图（scree plot）

更一般地，第 k 个主成分 z_k 对 x 总方差的贡献比例为

$$PVE_k \equiv \frac{Var(z_k)}{\sum_{i=1}^{p} Var(z_i)} = \frac{\lambda_k}{\lambda_1 + \cdots + \lambda_p}$$

其中，PVE_k 为第 k 个主成分 z_k 所能解释的方差比例（proportion of variance explained，简记 PVE）。由此可得每个主成分对方差的解释比例 $\{PVE_1, \cdots, PVE_p\}$ 。更直观地，可将 $(k, PVE_k), k = 1, \cdots, p$ 画图，所得结果类似

于陡坡图（可视为标准化陡坡图）。还可以将累积 PVE（Cumulative PVE）画图，以考察前若干个主成分对于总方差的累积解释比例。



图表 27 累积 PVE 图（Cumulative PVE）

聚类分析

K 均值聚类（K-means clustering）起源于波兰数学家 Steinhaus（1956），而成型于美国统计学家 McQueen（1967）。它是一种自上而下（top-down）的聚类方法，须预先确定样本中的聚类数目，即 K 的具体取值，比如根据经验或试错。假设数据可分为 K 类（K 已知）。将观测值下标 $\{1,\cdots,n\}$ 划分（partition）为 K 个不相交的集合 $\{C_1,\cdots,C_K\}$。其中，$C_k \cap C_{k'} = \emptyset(\forall k \neq k')$，而且 $C_1 \cup C_2 \cup \cdots \cup C_K = \{1,\cdots,n\}$（所有聚类的并集为整个样本）。因此，每个观测值都应有唯一的类别归属。具体来说，$i \in C_k$ 意味着第 i 个观测值 $\boldsymbol{x}_i$ 属于第 k 个聚类。我们希望每个聚类的“组内变动”（within-

cluster variation）越小越好。因此，记聚类 k 的均值或中心位置（cluster centriod）为

$$\boldsymbol{c}_k \equiv \frac{1}{|C_k|}\sum_{i\in C_k}\boldsymbol{x}_i \tag{1}$$

其中，$|C_k|$ 表示聚类 k 共有几个观测值。显然，样本中一共有 K 个中心位置（均值），故此算法称为“K 均值聚类”（K-means clustering）。对于聚类 k 中的某观测值 $\boldsymbol{x}_i(i \in C_k)$，称其到聚类中心位置的离差 $(\boldsymbol{x}_i - \boldsymbol{c}_k)$ 为“误差”（error）。将聚类 k 中所有误差的平方和加总，即得到聚类 k 的误差平方和（sum of squared errors，简记 SSE）：

$$SSE_k \equiv \sum_{i\in C_k}\|\boldsymbol{x}_i - \boldsymbol{c}_k\|^2 \tag{2}$$

其中，$\|x_i - c_k\|$ 为欧氏距离，即各分量平方和开根号。

将所有聚类的误差平方和加总，即可得到全样本的误差平方和：

$$SSE \equiv \sum_{k=1}^{K}\sum_{i\in C_k}\|\boldsymbol{x}_i - \boldsymbol{c}_k\|^2 \tag{3}$$

其中，SSE 也称为“组内平方总和”（total within sum of squares）。进一步地，我们的目标是，寻找对于样本下标集 $\{1,\cdots,n\}$ 的一个划分 $\{C_1,\cdots,C_K\}$，来使得全样本的误差平方和最小化：

$$\min_{c_1,\cdots,c_K} SSE = \sum_{k=1}^{K}\sum_{i\in C_k}\|\boldsymbol{x}_i - \boldsymbol{c}_k\|^2 \tag{4}$$

K 均值算法的具体步骤如下：

1. 随机选择每个聚类的中心位置（初始化）

2. 循环执行以下两步，直至收敛

(1) 将每个观测值重新分配到离它最近的聚类（分配）

(2) 更新每个聚类的中心位置（更新）。

显然，在每次循环迭代时，全样本的误差平方和 SSE 肯定下降；这是因为在每步迭代时，所有观测值都被分配到更近的聚类。如果 SSE 不再下降，则达到一个局部最小解。在以上第 1 步初始化时，也可以先将所有观测值随机分到 K 个聚类，再进行迭代。然而，由于 K 均值算法仅能找到局部最小值（未必是全局最小值），故其聚类结果依赖于初始的聚类中心位置（或聚类分布）。如果使用不同的初始聚类中心位置，则可能得到不同的局部最小解。在实践中，一般会尝试多个不同的初始聚类中心位置（设置不同的随机数种子），然后选择最佳结果，最小化全样本 SSE。这里我们用模拟数据来演示 K 均值算法的收敛过程。

K 均值聚类需要预先指定聚类数目 K ，但 K 可能不好确定。另一种聚类方式为分层聚类（hierarchical clustering）。分层聚类是一种自下而上（bottom-up）的逐次聚类方法 (agglomerative clustering)。它无须预先设定聚类数目，而从个体层面开始寻找最近的邻居，然后层层往上，可形成“树状图”（dendrogram），最后才选择聚类数目。总结起来，分层聚类算法的步骤如下：

1. 以每个观测值作为一个聚类

2. 找到距离最近的两个聚类，然后合并（merge）

3. 重复第 2 步，直至所有观测值都在同一聚类中

如果使用基于相关系数的距离指标，则两个观测值 $\boldsymbol{x_i} = \left(x_{i1}, \cdots, x_{ip}\right)'$ 与 $\boldsymbol{x_j} = \left(x_{j1}, \cdots, x_{jp}\right)'$ 之间的相关系数越大，其距离就越近：

$$dist\left(\boldsymbol{x_i}, \boldsymbol{x_j}\right) = \mathbf{1} - corr\left(\boldsymbol{x_i}, \boldsymbol{x_j}\right) \tag{7}$$

其中，$corr\left(\boldsymbol{x_i}, \boldsymbol{x_j}\right)$ 为观测值 $\boldsymbol{x_i}, \boldsymbol{x_j}$ 的相关系数。如果 $corr\left(\boldsymbol{x_i}, \boldsymbol{x_j}\right) = 1$，则二者的距离为 0。反之，如果 $corr\left(\boldsymbol{x_i}, \boldsymbol{x_j}\right) = -1$，则达到最远距离 2。显然，即使两个观测值的相关系数很接近于 1，二者的欧氏距离也可能很大。比如 $\boldsymbol{x_j} = 10\boldsymbol{x_i}$，此时二者的相关系数为 1，但在特征空间的欧氏距离却较远。

主成分分析估计结果

```
install.packages("pls")
```

```
library(pls)

pca_result <- princomp(data[, c("Digit", "Size", "Lev", "PPE", "ROA", "Cash",
"SOE", "Boardsize", "Indboard", "BM", "Dual", "Age","Id","Year")],
                       cor = TRUE)

summary(pca_result)
```

```
Importance of components:
                          Comp.1    Comp.2    Comp.3     Comp.4     Comp.5     Comp.6
Standard deviation     1.5667665 1.2992542 1.2281453 1.17190814 1.02814330 1.00162123
Proportion of Variance 0.1753398 0.1205758 0.1077386 0.09809776 0.07550562 0.07166036
Cumulative Proportion  0.1753398 0.2959156 0.4036543 0.50175202 0.57725764 0.64891800
                          Comp.7    Comp.8     Comp.9    Comp.10    Comp.11    Comp.12
Standard deviation     0.9583691 0.9005051 0.85088820 0.83034908 0.78687169 0.68441421
Proportion of Variance 0.0656051 0.0579221 0.05171505 0.04924854 0.04422622 0.03345877
Cumulative Proportion  0.7145231 0.7724452 0.82416026 0.87340880 0.91763502 0.95109379
                          Comp.13    Comp.14
Standard deviation     0.62264454 0.54497769
Proportion of Variance 0.02769187 0.02121433
Cumulative Proportion  0.97878567 1.00000000
```

图表 28 主成分分析结果

```
pca_scores <- scale(pca_result$scores)

pcr_model <- pcr(Innovation ~ ., data = cbind(data, pca_scores),
                 validation = "CV")#主成分分析回归

summary(pcr_model)
```

分层聚类分析估计结果

```
install.packages("NbClust")
library(NbClust)
nb <- NbClust(data, distance = "euclidean", min.nc = 2, max.nc = 10, method = "kmeans")
best_clusters <- nb$Best.nc[["Number_clusters"]][1]
clusters <- cutree(hc, k = best_clusters)
your_data$cluster <- clusters
library(MASS)
models <- list()
```

```
for (i in 1:best_clusters) {
  cluster_data <- your_data[clusters == i, ]
  formula <- as.formula(paste("Innovation ~ Digit + Size + Lev + PPE + ROA +
Cash + SOE + Boardsize + Indboard + BM + Dual + Age"))
  model <- stepAIC(lm(formula, data = cluster_data), direction = "both")
  models[[i]] <- model
}#分层聚类回归
summary(models)
```