基于 Bagging 模型的回归优化模型

摘　　要

回归问题在生活和工业中非常重要。本文将会以一个简单的数学问题出发，采用不同的回归方法进行回归并对不同的回归方法进行比较。从而探究几种常见的回归方法。

本文第一部分介绍算法的基本原理，包括最小二乘法原理，机器学习回归原理，**KNN** 回归原理，神经网络回归原理，同时介绍了 B**agging** 集成算法的原理。

本文第二部分开始使用四种回归方法对 2020 年明信片的价格分别进行回归并预测。然后使用 **Bagging** 算法求出 3 种回归算法的平均值。

本文第三部分首先将数据划分为两组。用四种回归算法和 **Bagging** 集成算法进行回归，然后计算其均方差。结果显示，使用神经网络回归的效果最佳。其均方差只有 **32.20**，因此，预测 2020 年明信片的价格为 **34.00** 美分。

关键词：**最小二乘法　KNN 回归　高阶多项式回归　剃度下降法　Bagging 模型**

目　　录

1 绪论

回归，指研究一组随机变量 $(y_1, y_2, \ldots, y_i)$ 和另一组 $(x_1, x_2, \ldots, x_k)$ 变量之间关系的统计分析方法，又称多重回归分析。通常 $y_1, y_2, \ldots, y_i$ 是因变量，$x_1, x_2, \ldots, x_k$ 是自变量。常见的回归算法包括最小二乘法回归，机器学习回归，KNN 回归，神经网络回归等。

在回归问题中，对于某些离群值或者异常值，单个回归算法会产生较大误差。集成算法是一类构建多个学习器, 通过一定策略结合来提高准确率，降低误差的模型。常见的集成算法分为 Bagging、Boosting、Stacking 三大类。本文主要使用了 Bagging 算法。

1.1 回归算法

1.1.1 最小二乘法回归

给定 n 个属性描述的自变量 $\mathbf{x} = (x_1, x_2,\ldots, x_n)$, 其中 x_i 为 $\mathbf{x}$ 在第 i 个元素上的取值。线性回归试图学得一个通过该属性的线性组合来进行预测的函数，即

$$f(\mathbf{x}) = w_1x_1 + w_2x_2 + \cdots + w_nx_n + b, \tag{1.1}$$

一般用向量形式写成

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b, \tag{1.2}$$

其中 $\mathbf{w} = (x_1, x_2,\ldots, x_n)$,$\mathbf{w}$ 和 b 确定后，模型也得以确定。

给定数据集 $D = (\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)$, 其中 $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{im}), y_i \in \mathbb{R}$. 线性模型试图学得

$$f(\mathbf{x}_i) = \mathbf{w}^T\mathbf{x}_i + b, \qquad s.t. \qquad f(\mathbf{x}_i) \approx y_i. \tag{1.3}$$

为了方便，我们令

$$\hat{w} = (\mathbf{w}, b),$$

$$\mathbf{x} = \begin{pmatrix} x_{11} & x_{12} & \ldots & x_{1d} & 1 \\ x_{21} & x_{22} & \ldots & x_{2d} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{m1} & x_{m2} & \ldots & x_{md} & 1 \end{pmatrix} = \begin{pmatrix} x_1^T & 1 \\ x_2^T & 1 \\ \vdots & \vdots \\ x_m^T & 1 \end{pmatrix},$$

$$\mathbf{y} = (y_1, y_2, \ldots, y_m),$$

其中，$\mathbf{x}$ 是一个 $m \times (d+1)$ 的矩阵，每行前 d 个元素对应集合 D 中的 d 个属性值。

为了实现 1.3 , 我们使用均方误差来衡量 $f(x_i)$ 与 y_i 之间的差别，以便确定最合适的 $\mathbf{w}$ 和 b:

$$E(f, D) = \frac{1}{m}\sum_{i=1}^{m}(f(x_i) - y_i)^2. \tag{1.4}$$

线性回归的任务转化为

$$\hat{w}^* = arg_{\hat{w}} min(\mathbf{y} - \mathbf{x}\hat{w})^T(\mathbf{y}\text{-}\mathbf{x}\hat{w}). \tag{1.5}$$

令 $E_{\hat{w}} = (\mathbf{y} - \mathbf{x}\hat{w})^T(\mathbf{y}\text{-}\mathbf{x}\hat{w})$, 对 $\hat{w}$ 求导得到

$$\frac{\partial E_{\hat{w}}}{\partial \hat{w}} = 2\mathbf{x}^T(\mathbf{x}\hat{w} - \mathbf{y}). \tag{1.6}$$

令 1.6等于零可以求得 $\hat{w}$ 最优解的闭式解。特殊地，当集合域 D 中只有一个属性时，可以求得

$$w = \frac{\sum_{i=1}^{m} y_i(x_i - \bar{x})}{\sum_{i=1}^{m} x_i^2 - \frac{1}{m}(\sum_{i=1}^{m} x_i)^2}, \tag{1.7}$$

$$b = \frac{1}{m}\sum_{i=1}^{m}(y_i - wx_i), \tag{1.8}$$

其中 $\bar{x} = \frac{1}{m}\sum_{i=1}^{m} x_i$ 为 x 的均值.

现在对一个给定的数据点集用最小二乘法准则拟合 $y = Ax^n$ 形式的曲线，n 为固定数，研究模型 $f(x) = ax^n$ 的最小二乘估计，应用该准则要求极小化

$$s = \sum_{i=1}^{m}[y_i - f(x_i)]^2 = \sum_{i=1}^{m}[y_i - ax_i^n]^2$$

. 最优化的必要条件是 $\frac{dS}{da} = -2\sum_{i=1}^{m} x_i^n[y_i - ax_i^n] = 0$, 从方程可以解出 a，得

$$a = \frac{\sum x_i^n y_i}{\sum x_i^{2n}}$$

1.1.2 机器学习回归

在 1.1.1 中为了实现 1.3, 我们用均方误差 1.4来衡量拟合的曲线与实际数据的误差. 当我们使 1.4取得最小值时, 也就确定了最优解. 神经网络使用梯度下降算法确定最优解. 为了求出 1.5, 神经网络模型的计算步骤如下:

Algorithm 1 机器学习回归问题

1: 随机初始化一组值 w_0

2: 求解 $\frac{\partial E_{\hat{w}}}{\partial \hat{w}} = 2\mathbf{x}^T(\mathbf{x}\hat{w} - \mathbf{y})$

3: $w_1 = w_0 - \alpha \frac{\partial E_{\hat{w}}}{\partial \hat{w}}$, 其中 α 为常数, 也称学习速率.

4: $w_0 = w_1$, 重复 3, 直到 $w_1 < 10^{-6}$

5: 输出结果 w_1

其中, 特别地, 当集合域 D 中只有一个属性时:

Algorithm 2 只有一个属性时机器学习解决回归问题

1: 随机初始化一组值 w_0, b_0

2: 求解 $\frac{\partial E_{\hat{w}}}{\partial \hat{w}}, \frac{\partial E_{\hat{w}}}{\partial b}$

3: $w_1 = w_0 - \alpha \frac{\partial E_{\hat{w}}}{\partial \hat{w}}, b_1 = b_0 - \alpha \frac{\partial E_{\hat{w}}}{\partial b}$ 其中 α 为常数, 也称学习速率.

4: $w_0 = w_1, b_0 = b_1$, 重复 3, 直到 $w_1 < 10^{-6}$

5: 输出结果 w_1, w_0

机器学习能够解决的不仅只有线性回归问题，当知道目标函数后，即可使用剃度下降法求得目标函数的极值，从而求解各种回归问题。

1.1.3 KNN 回归

KNN 算法不仅可以用来聚类，还可以用来回归。KNN 是一个简单的算法。简单的说就是取 x 距离最近的 K 个点，求它们的平均值作为预测值。如图 1.1是一个 K=3 时用 KNN 回归的一个例子,KNN 算法选取了最近的 3 个样本并取它们的平均值作为预测值。

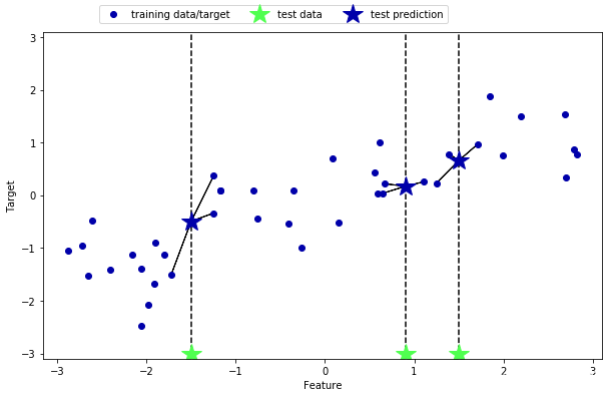

图 1.1　K=3 时 KNN 回归的一个例子

使用 KNN 回归的算法流程图如下：

Algorithm 3 KNN 回归

Input:

N 个训练样本的集合 $A[n]$;

人为设定的近邻个数 k;

给定的需要的预测的变量 x:

Output:

预测的值 y;

选择 $A[1]$ 至 $A[k]$ 作为 x 的初始近邻；

计算初始近邻与测试样本 x 间的欧氏距离 $d(x, A[i]), i = 1, 2, ...k$；

按 $d(x, A[i])$ 从小到大排序;

计算最远样本与 x 间的距离 D，即 $max\{d(x, A[j])|j = 1, 2...k\}$;

$for(i = k + 1; i < n + 1; i + +)$ {

计算 $A[i]$ 与 x 间的距离 $d(x, A[i])$;

$if(d(x, A[i]) < D)$

{

用 $A[i]$ 代替最远样本;

}

按照 d(x, A[i]) 从小到大排序；

计算最远样本与 x 间的距离 D，即 $max\{d(x, A[j])|j = 1, ...i\}$；

}

计算 $y = \frac{1}{k}\sum_{i=1}^{k} d_i$

return y;

1.1.4 神经网络回归

神经网络（Artificial Neural Networks，简写为 ANNs）也简称为神经网络（NNs）或称作连接模型（Connection Model），它是一种模仿动物神经网络行为特征，进行分布式并行信息处理的算法数学模型。

神经网络在理论上能够拟合任意函数，包括非线性函数，是当下最流行的数学建模方法之一。

1959 年两个生物科学家发现青蛙的神经元接受多个输入，输入包括青蛙的多个器官的输入，只有单输入的和到达一个阈值，才会有输出（青蛙接受的刺激比较大时才会有反应。

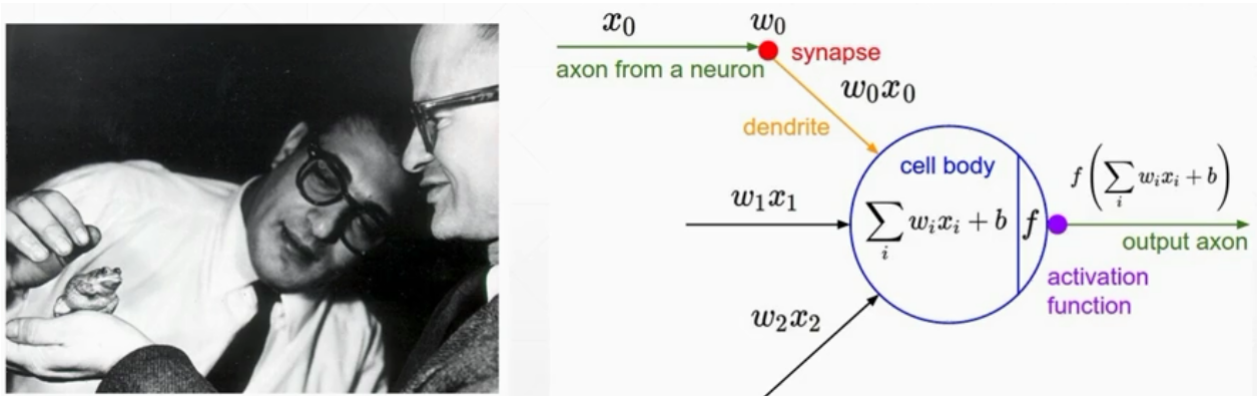

图 1.2　科学家研究青蛙的神经元原理

于是计算机科学家仿照生物神经元的原理和结构，提出了感知器：

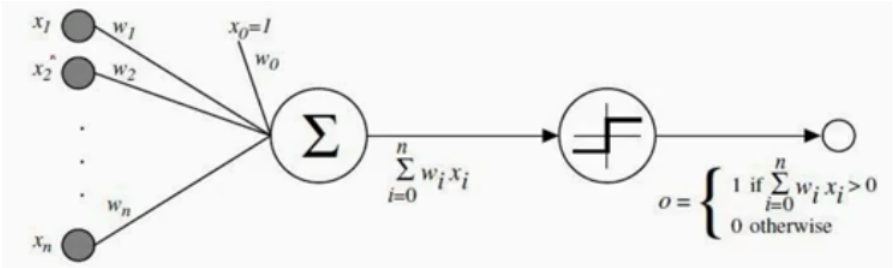

图 1.3　感知器

经过不断完善和发展，形成了神经网络：

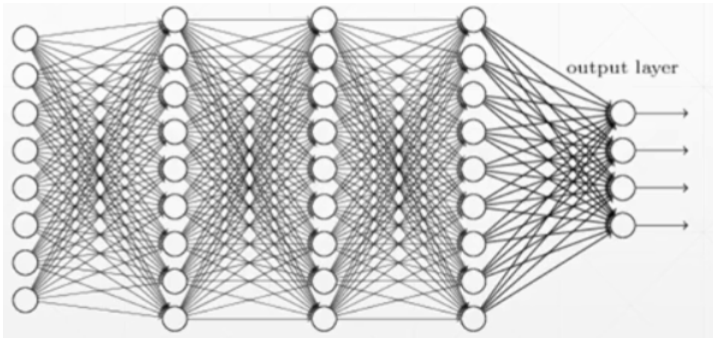

图 1.4　神经网络

神经网络使用剃度下降法求解最优值，剃度下降法是一类求解函数最小值的计算机算法。假设希望求解目标函数 $f(x) = f(x_1, \ldots, x_n)$ 的最小值，可以从一个初始点

$x^{(0)} = (x_1^{(0)}, \ldots, x_n^{(0)})$ 开始，基于学习率 $\alpha > 0$ 构建一个迭代过程:

$$x_1^{i+1} = x_1^i + \alpha \frac{\partial f}{\partial x_1}(x^{(i)})$$

$$\ldots$$

$$x_1^{i+1} = x_1^i + \alpha \frac{\partial f}{\partial x_1}(x^{(i)})$$

其中 $x^{(i)} = (x_1^{(i)}, \ldots, x_n^{(n)}), i >= 0$, 一旦达到收敛条件的话，迭代就结束了。图 1.5是用剃度下降法求解 $y = x^2$ 的最小值的过程，其中的点代表每次迭代后的值。

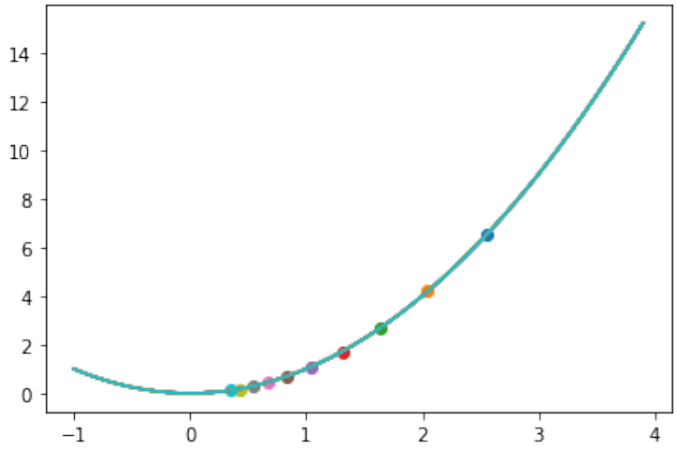

图 1.5　剃度下降法图解

1.2　Bagging 算法

Bagging 算法（英语：Bootstrap aggregating，引导聚集算法），又称装袋算法，是机器学习领域的一种团体学习算法。最初由 Leo Breiman 于 1996 年提出。Bagging 算法可与其他分类、回归算法结合，提高其准确率、稳定性的同时，通过降低结果的方差，避免过拟合的发生。Bagging 算法描述如下所示.

输入： 训练集 $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
基学习算法 $\mathfrak{L}$;
训练轮数 T.

过程：

1: **for** $t = 1, 2, \ldots, T$ **do**
2: $\quad h_t = \mathfrak{L}(D, \mathcal{D}_{bs})$
3: **end for**

输出： $H(\boldsymbol{x}) = \arg\max_{y \in \mathcal{Y}} \sum_{t=1}^{T} \mathbb{I}(h_t(\boldsymbol{x}) = y)$

图 1.6　Bagging 算法

Bagging 算法就是将多个分类或者回归器同时训练，最后取平均值作为最终结果。其原理如图 1.7所示，其中输入为自变量，即需要输入的训练数据，$Function1-4$ 为 4 个不同的分类器或者回归器，y_i 为它们的输出结果，Bagging 模型最终求其平均作为最终结果。

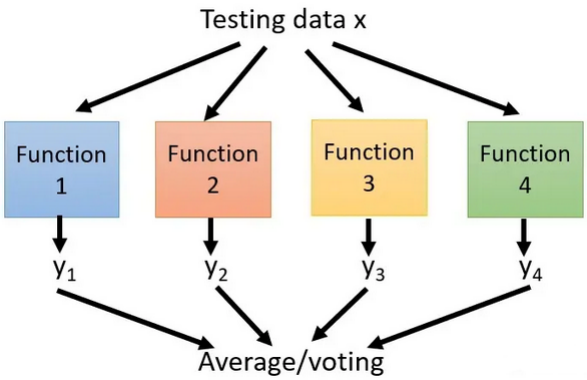

图 1.7　Bagging 算法图解

1.3　问题重述

图 1.8 是 1898 年到 2012 年明信片价格的变化，据此完成下列任务:

Year	Postcard cost in cents
1898	1
1952	2
1958	3
1963	4
1968	5
1971	6
1974	8
1975	7
1975	9
1978	10
1981	12
1981	13
1985	14
1988	15
1991	19
1995	20
1999	20
2001	20
2001	21
2001	23
2006	24
2007	26
2008	27
2009	28
2011	29
2012	32

图 1.8　明信片价格变化

(一)

- 使用最小二乘法建立曲线回归模型，预测 2020 年明信片的价格.
- 使用机器学习的方法求解高阶多项式模型并预测 2020 年明信片的价格.
- 建立 KNN 回归模型并预测 2020 年明信片的价格.
- 建立神经网络回归模型并预测 2020 年明信片的价格.
- 在上述四种模型的基础上建立 Bagging 算法模型并预测 2020 年明信片的价格.

(二)

- 比较以上 5 种回归模型的结果并选出最合适的模型，给出最终结果.

2 问题求解

2.1 数据预处理

我们将数据输入为 csv 文件，并对年份进行了处理：

$$x_i = x_i - x_1 + 1$$

以便更好的求解模型。

2.2 最小二乘法拟合 ax^n

假定需要拟合的曲线为 $y = ax^2$, 研究 $f(x) = ax^2$ 的最小二乘估计，应用准则要求极小化

$$S = \sum_{i=1}^{m}[y_i - f(x_i)]^2 = \sum_{i=1}^{m}[y_i - ax_i^2]^2$$

最优化的必要条件是导数 dS/da 等于 0：

$$\frac{dS}{da} = -2\sum_{i=1}^{m} x_i^n[y_i - ax_i^2] = 0$$

从方程中解出 a，得

$$a = \frac{\sum x_i^2 y_i}{\sum x_i^4} = 0.001981$$

因此，回归方程为

$$y = 0.001981 * x^2$$

, 当 $x = 2020 - 1898 + 1 = 123$ 时，

$$y = 29.97$$

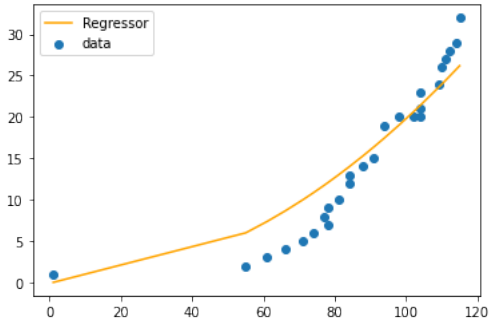

图 2.1　拟合结果

2.3　剃度下降法拟合高阶多项式模型

假设需要拟合的曲线为

$$y = a_0 + a_1 x + a_2 x^2 + \cdots + a_{10} x^{10}$$

损失函数为

$$loss = \sum (y(x_i) - y_i)^2$$

初始化一组解

$$a_0 = a_1 = \cdots = a_{10} = 1$$

求解

$$grad_{a_j} = \frac{dloss}{da_j} = 2 \sum (y(x_i) - y_i) \sum x_i^j \tag{2.1}$$

求解

$$a_i = a_i - \alpha * grad_{a_i} \tag{2.2}$$

, 其中 $\alpha = 0.001, i = 1, 2, \ldots, 10$. 迭代 2.1 与 2.2，直到 $|grad_{a_i}|$<=0.001. 通过计算机程序可求得:

$$y = -1.639136814e-15 * x^{10} + 1.059936718e-12 * x^9 - 0.0000000002960388319 * x^8$$
$$+0.00000004652752046 * x^7 - 0.000004472787601 * x^6 + 0.0002650337924 * x^5$$
$$-0.00900509148 * x^4 + 0.126851489 * x^3 + 1.240266286 * x^2$$
$$-47.08797047 * x + 46.72959737.$$

当 $x = 123$ 时，$y = 13.78$

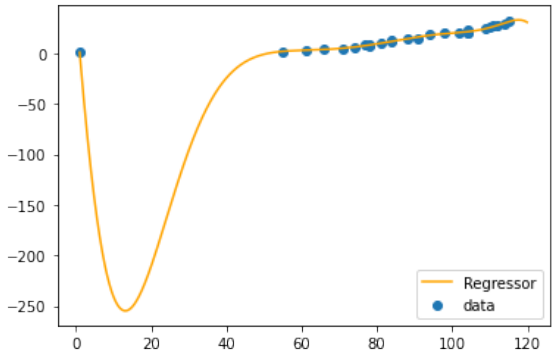

图 2.2　拟合结果

2.4　KNN 回归

我们使用 KNN 进行回归，得出以下结果：

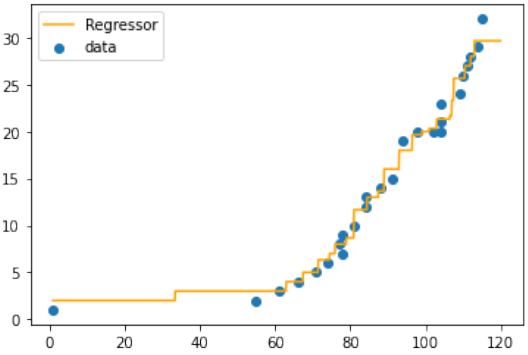

图 2.3　拟合结果

当 $x = 123$ 时,

$$y = 29.67$$

2.5　神经网络回归

我们建立下面的神经网络，并进行训练.

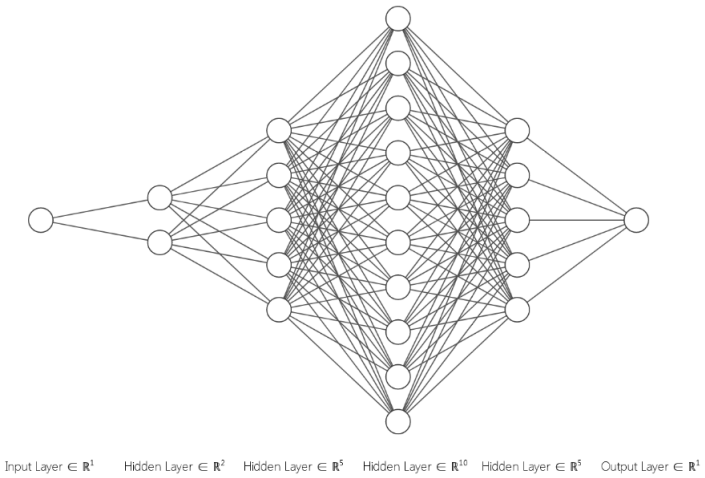

图 2.4　回归模型

其中，学习速率 $\alpha = 0.001$, 最终拟合结果如图 2.5所示，当 $x = 123$ 时，

$$y = 34.00$$

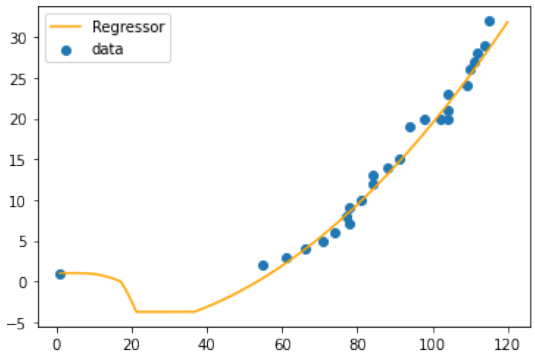

图 2.5　拟合结果

2.6　Bagging

最后，我们使用 Bagging 算法对上面的四种结果取平均，其中，注意到，高阶多项式模型的误差比较大，因此 Bagging 算法选用其他三种回归算法。算法流程图如下。

根据 Bagging 算法，可以求得

$$y = \frac{1}{3}(34.00 + 29.67 + 29.97) = 31.21$$

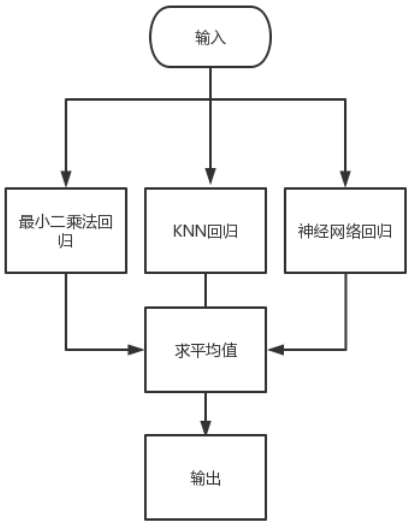

图 2.6　Bagging 算法

3　模型比较

3.1　数据预处理

为了能够更好的比较模型的准确度，我们从数据中抽取一半的数据用于建立模型，剩下的一半数据用于判断模型的准确度。数据分布如下：

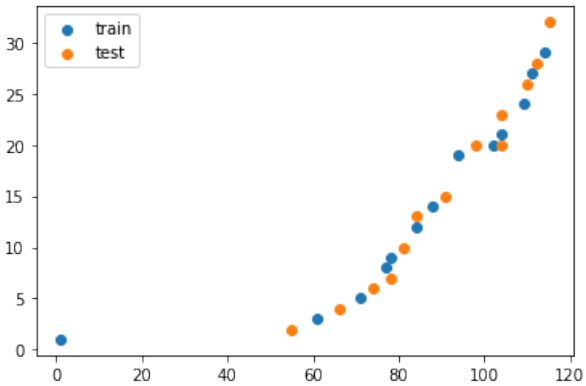

图 3.1　数据划分

3.2 最小二乘法回归

假定需要拟合的曲线为 $y = ax^2$, 研究 $f(x) = ax^2$ 的最小二乘估计，应用准则要求极小化

$$S = \sum_{i=1}^{m}[y_i - f(x_i)]^2 = \sum_{i=1}^{m}[y_i - ax_i^2]^2$$

最优化的必要条件是导数 dS/da 等于 0：

$$\frac{dS}{da} = -2\sum_{i=1}^{m} x_i^n[y_i - ax_i^2] = 0$$

从方程中解出 a，得

$$a = \frac{\sum x_i^2 y_i}{\sum x_i^4} = 0.001956$$

因此，回归方程为

$$y = 0.001981 * x^2$$

.

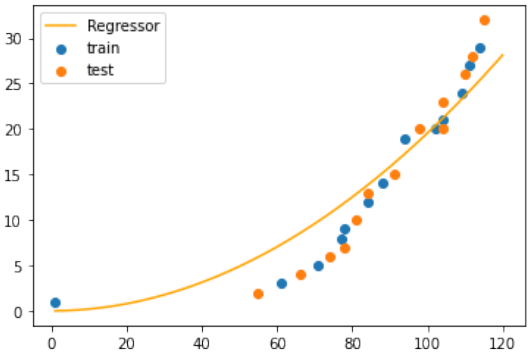

图 3.2 拟合结果

测试数据的均方误差为

$$E = \frac{1}{m}\sum_{i=1}^{m}(f(x_i) - y_i)^2 = 4395.53.$$

3.3 剃度下降法拟合高阶多项式模型

假设需要拟合的曲线为

$$y = a_0 + a_1 x + a_2 x^2 + \cdots + a_{10} x^{10}$$

损失函数为

$$loss = \sum(y(x_i) - y_i)^2$$

初始化一组解

$$a_0 = a_1 = \cdots = a_{10} = 1$$

求解

$$grad_{a_j} = \frac{dloss}{da_j} = 2\sum(y(x_i) - y_i)\sum x_i^j \tag{3.1}$$

求解

$$a_i = a_i - \alpha * grad_{a_i} \tag{3.2}$$

, 其中 $\alpha = 0.001, i = 1, 2, \ldots, 10$. 迭代 2.1 与 2.2，直到 $|grad_{a_i}|$<=0.001. 通过计算机程序可求得:

$$3.975590632e-13 * x^10 - 0.0000000003290771782 * x^9$$
$$+0.0000001204184953 * x^8 - 0.00002556825203 * x^7$$
$$+0.003471943435 * x^6 - 0.3127761976 * x^5 + 18.70628778 * x^4$$
$$-717.4565232 * x^3 + 16097.80278 * x^2 - 165200.0207 * x + 149802.2775$$

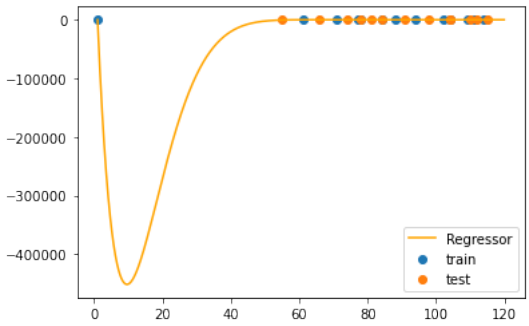

图 3.3　拟合结果

测试数据的均方误差为

$$E = \frac{1}{m}\sum_{i=1}^{m}(f(x_i) - y_i)^2 = 177174.24.$$

3.4　KNN 回归

我们使用 KNN 进行回归，得出以下结果:

测试数据的均方误差为

$$E = \frac{1}{m}\sum_{i=1}^{m}(f(x_i) - y_i)^2 = 25636.22.$$

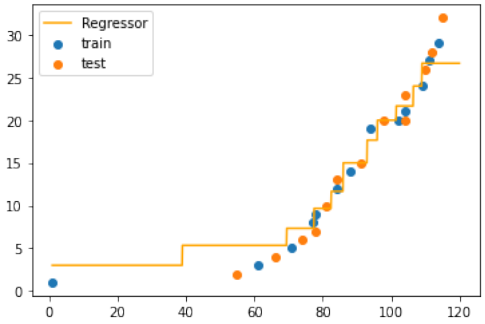

图 3.4　拟合结果

3.5　神经网络回归

我们采用第二节的神经网络 2.4 模型进行训练. 其中，学习速率 $\alpha = 0.001$, 最终拟合结果如图 2.5所示，当 $x = 123$ 时，

$$y = 34.00$$

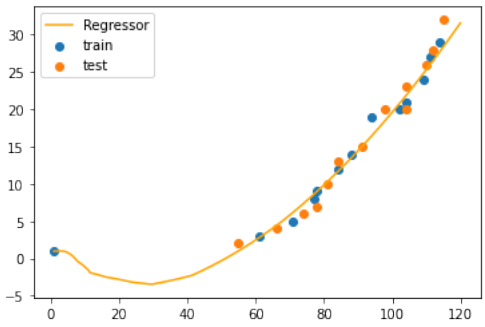

图 3.5　拟合结果

测试数据的均方误差为

$$E = \frac{1}{m}\sum_{i=1}^{m}(f(x_i) - y_i)^2 = 31.2013.$$

3.6　Bagging

最后，我们使用 Bagging 算法对上面的四种结果取平均，其中，注意到，高阶多项式模型的误差比较大，因此 Bagging 算法选用其他三种回归算法。

测试数据的均方误差为:

$$E = \frac{1}{m}\sum_{i=1}^{m}(f(x_i) - y_i)^2 = 24614.14$$

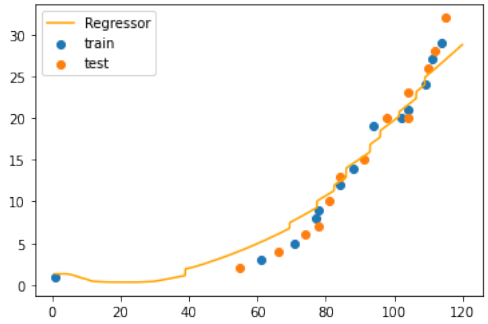

图 3.6　拟合结果

3.7　最终结果

表 3.1　不同回归方法的比较

回归方法	2020 年预测结果	均方差
最小二乘法曲线回归	29.97	2723090.92
高阶多项式回归	13.78	177174.24
KNN 回归	29.67	25636.22
神经网络回归	34.00	31.2013
Bagging 模型	31.21	24614.14

综上所述，选用神经网络回归效果最好，测试数据的均方误差为

$$E = 31.2013.$$

，2020 年明信片价格预测结果为

$$y = 34.00$$

4　总结

从模型的结果来看，神经网络的拟合效果是最好的。其中 KNN 回归原理虽然简单，但是效果良好；而高阶多项式回归在数据集两侧的误差则特别大。Bagging 模型的拟合效果虽然不如神经网络，但优于其他回归方法，具有提高准确率，降低过拟合的作用。

参考文献