

正则化回归

一. 算法原理

二. 正则化回归案例

1. 岭回归预测波士顿房价

2. Lasso回归预测波士顿房价

3. 弹性网回归预测波士顿房价

多元线性回归1的最小二乘损失是让残差平方和（偏差）达到最小：

$$\mathcal{L}(\beta) = \frac{1}{2n} \|Y - X\beta\|_2^2 = \frac{1}{2n} \sum_{i=1}^n (y_i - \sum_{j=1}^p x_{ij}\beta_j)^2$$

多元线性回归通常需要在众多自变量中选择对因变量有显著性影响的自变量。

mlr3 提供了多元线性回归学习器：regr.lm

多元线性回归的缺陷：

- 伪回归：当**特征高度相关时，增加或删除一个自变量后，回归系数变化很大甚至改变符号，主要原因就是变量之间存在多重共线性**，得到回归

模型是“伪回归”。此时通过普通最小二乘法（OLS）拟合的模型会出现回归系数估计不稳定，得到误导性的统计显著性：模型看起来具有显著的解释能力，但实际上并没有捕捉到真正的影响关系。

假设 $x_1 = 2x_2$, $y = 1 + 2x_1 + 4x_2 = 1 + x_1 + 6x_2$ 系数变化没法控制。

系数随便变化，不能反映 x_1 对 y 的影响大小。

- 过拟合：多元线性回归也有过拟合的问题，即在训练集上表现良好，但在未见过的数据上表现不佳。这更容易发生在特征过多而观测较少的数据集上，可能表现为某些回归系数过大。

正则化回归是通过在最小二乘损失函数中添加一个正则项，来控制模型的复杂度，正则项通常有 $L2$ 正则化（岭回归）和 $L1$ 正则化（Lasso 回归），以及两种正则项的加权合成（弹性网模型）。

- 当存在共线性导致伪回归时， $L2$ 正则项可以通过对回归系数施加惩罚（限制回归系数的大小），使得模型更倾向于选择稳定的解，从而能够减少共线性对估计结果的影响，也就减少伪回归的发生。
- 当模型存在过拟合问题时，正则化项可以对模型参数进行约束，防止它们过度波动。 $L1$ 正则项不仅可以压缩回归系数，还可以进行特征选择，

将某些不重要的特征的系数收缩到 0，从而简化模型并提高泛化能力。

因此，正则化回归方法在解决过拟合和伪回归问题上具有很好的效果，不同的正则化方法适用于不同的问题和数据集，需要根据具体情况选择合适的正则化方法

一. 算法原理

1. 岭回归

当出现以下情况时，可能需要考虑使用岭回归：

- 自变量数量很大，甚至多于观测数，最小二乘估计可能不存在或不唯一；
- 最小二乘估计依赖于 $(X'X)^{-1}$ ，若 $(X'X)^{-1}$ 奇异或接近奇异，最小二乘估计会有问题： X 微小的改变将导致 $(X'X)^{-1}$ 巨大的改变；
- 普遍线性回归出现过拟合。

岭回归是一种改良的最小二乘法，通过放弃最小二乘法的无偏性，以损失部分信息、降低精度为代价，获得回归系数更为符合实际、更可靠的回归方法。

这种改良是通过对最小二乘损失函数添加 $L2$ 正则项来实现的：

$$\begin{aligned}\mathcal{L}(\beta, \lambda) &= \frac{1}{2n} \|Y - X\beta\|_2^2 + \lambda \cdot \frac{1}{2} \|\beta\|_2^2 \\ &= \frac{1}{2n} \sum_{i=1}^n (y_i - \sum_{j=1}^p x_{ij}\beta_j)^2 + \lambda \cdot \frac{1}{2} \sum_{j=1}^p |\beta_j|^2\end{aligned}$$

λ 控制惩罚项大小，越大，压迫回归系数变小。

其中正则项系数 λ 为非负实数，用于权衡线性回归拟合度（偏差）和回归系数惩罚力度（方差）：

- 当 $\lambda=0$ 时，损失函数退化为多元线性回归模型的损失函数；
- 当 $\lambda \rightarrow \infty$ 时，会压缩回归系数 β 使其趋于 0；
- β 为模型参数， λ 是需要调参的超参数。

λ 和 c 是相反的关系。

不同的正则化参数 λ 值，可以估计出不同的岭回归系数，通常用岭迹图展示了岭回归中的系数估计随着 λ 的变化而变化的轨迹。一般而言，当回归系数随着 λ 值的增大而趋近于稳定的点时就是所要寻找的 λ 值。

可以推导出对前式求 $\arg\min_{\beta}$ 的优化问题等价于

$$\begin{aligned} \arg \min_{\beta} \quad & \frac{1}{2n} \sum_{i=1}^n \left(y_i - \sum_{j=1}^p x_{ij} \beta_j \right)^2 \\ \text{s.t.} \quad & \sum_{j=1}^p \beta_j^2 \leq c, \quad \exists c > 0 \end{aligned}$$

利用拉格朗日对偶法，可将不等式约束问题转化为无约束问题

岭回归模型的几何意义（二元回归为例）

以二元回归为例：

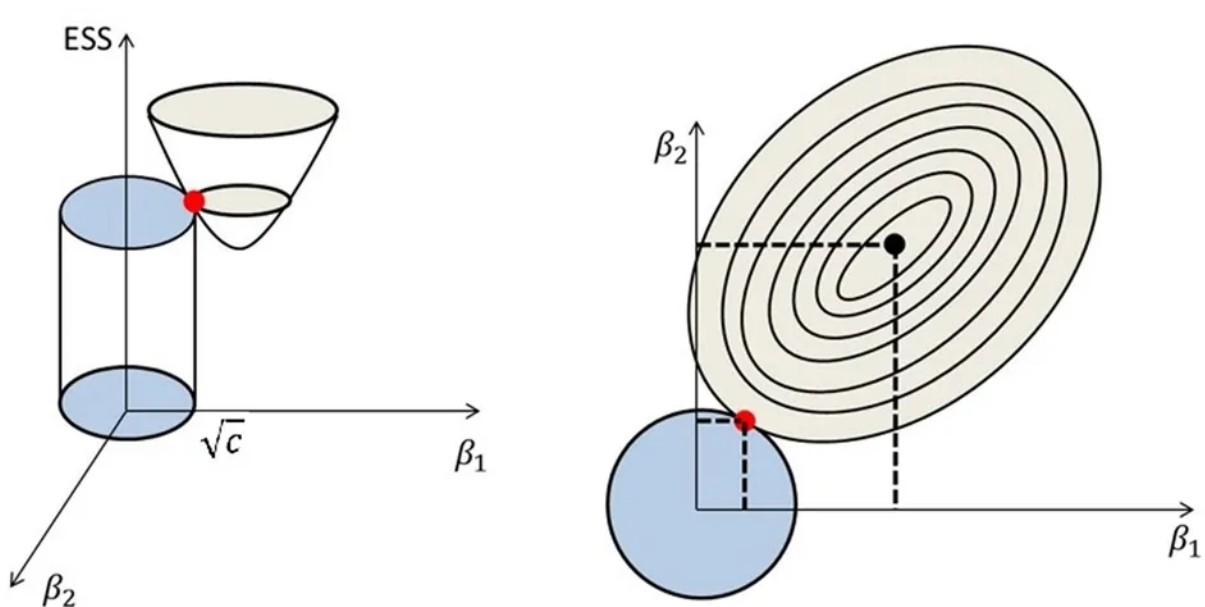


图 1: 岭回归正则项的几何意义

- 左图中椭球面代表残差平方和 $\sum_{i=1}^n [y_i - (\beta_0 + \beta_1 x_1 + \beta_2 x_2)]^2$ 它是关于两个回归系数 β_1 和 β_2 的二次函数；圆柱体代表 $\beta_1^2 + \beta_2^2 \leq c$ ；
- 右图是投影到决策变量 β_1 和 β_2 的坐标平面上；
- 椭球面的中心黑点代表模型的最小二乘解，当增加了约束条件 $\beta_1^2 + \beta_2^2 \leq c$ 时，椭球面与圆柱面构成的红色交点就是岭回归解；
- c （与 λ 存在对应）是圆柱半径的平方，让 c 变化就相当于在权衡残差平方和与惩罚项：大的回归系数可能有更小的残差平方和，“小” c 对应“大” λ ，侧重于正则项，惩罚回归系数不至于过大。

岭回归理论上的最优回归系数可以直接推导出来 (王圣元, 2020):

$$\hat{\beta}_{R_\lambda} = (\mathbf{X}'\mathbf{X} + \lambda I_p)^{-1} \mathbf{X}'y$$

在机器学习场景下，一般是用梯度下降法求解优化问题： $\min_{\beta} \mathcal{L}(\beta, \lambda)$ ，对给定的超参数 λ 。

注：若更关心岭回归的计量结果，建议使用 `lmridge` 包。

2. Lasso 回归

注意到岭回归解是出现在圆周与等高线的交点，是无论如何也到达不了两个坐标轴，也因此岭回归只能将各个自变量的回归系数压缩到近似为 0，而不能等于 0，即模型始终包含 p 个自变量不能降低模型的复杂度。

若将岭回归添加的 $L2$ 正则项，换成 $L1$ 正则项，就得到 Lasso 回归模型：

$$\begin{aligned} \mathcal{L}(\beta, \lambda) &= \frac{1}{2n} \|\mathbf{Y} - \mathbf{X}\beta\|_2^2 + \lambda \cdot \|\beta\|_1 \\ &= \frac{1}{2n} \sum_{i=1}^n (y_i - \sum_{j=1}^p x_{ij}\beta_j)^2 + \lambda \cdot \sum_{j=1}^p |\beta_j| \end{aligned}$$

与岭回归一样，正则项系数 λ 为非负实数，用于权衡线性回归拟合度（偏差）

和回归系数惩罚力度（方差）：

- 当 $\lambda=0$ 时，损失函数退化为多元线性回归模型的损失函数；
- 当 $\lambda \rightarrow \infty$ 时，会压缩（所有）回归系数 β 使其趋于 0；

- β 为模型参数， λ 是需要调参的超参数。

不同的正则化参数 λ 值，可以估计出不同的Lasso 回归系数，通常用Lasso回归系数图展示了 Lasso 回归中的系数估计随着 λ 的变化而变化的轨迹。一般而言，当回归系数随着 λ 值的增大而趋近于稳定的点时就是所要寻找的 λ 值

可以推导出对前式求 $\operatorname{argmin}_{\beta}$ 的优化问题等价于：

$$\begin{aligned} \operatorname{argmin}_{\beta} \quad & \frac{1}{2n} \sum_{i=1}^n (y_i - \sum_{j=1}^p x_{ij} \beta_j)^2 \\ \text{s.t.} \quad & \sum_{j=1}^p |\beta_j| \leq c, \quad \exists c > 0 \end{aligned}$$

Lasso 回归的几何意义

- 以二元回归为例：

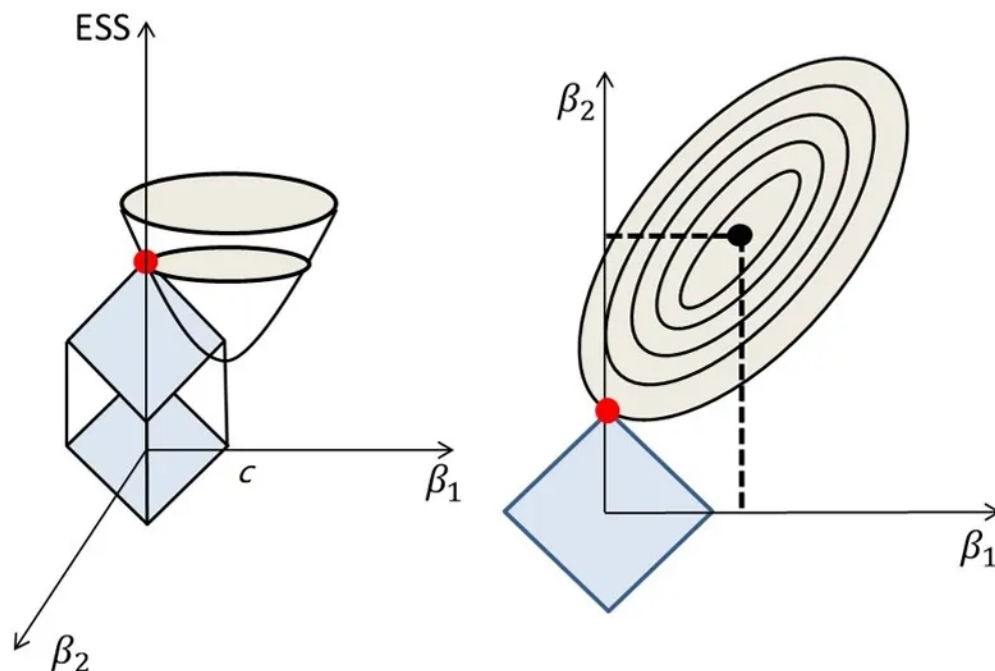


图 2: 岭回归正则项的几何意义

- 左图中椭球面代表残差平方和 $\sum_{i=1}^n [y_i - (\beta_0 + \beta_1 x_1 + \beta_2 x_2)]^2$, 它是关于两个回归系数 β_1 和 β_2 的二次函数; 四棱柱代表 $|\beta_1| + |\beta_2| \leq c$;
- 右图是投影到决策变量 β_1 和 β_2 的坐标平面上;
- 椭球面的中心黑点代表模型的最小二乘解, 当增加了约束条件 $|\beta_1| + |\beta_2| \leq c$ 时, 椭球面与棱柱面构成的红色交点就是 Lasso 回归解;
- 同样, c (与 λ 存在对应) 是棱柱“半径”, 让 c 变化就相当于在权衡残差平方和与惩罚项: 大的回归系数可能有更小的残差平方和, “小” c 对应 “大” λ , 侧重于正则项, 惩罚回归系数不至于过大

注意到，图中 Lasso 回归解是出现在正方形顶点与等高线的交点，是可以到达 β_2 坐标轴的。因此 Lasso 回归能将某些自变量的回归系数压缩到等于0，即模型不包含某些自变量，从而能起到筛选自变量降低模型复杂度的效果。

基于 Lasso 回归的特征选择法，是将对正则化系数经过交叉验证调参的Lasso 回归模型的回归系数作为特征重要性得分，基于某阈值或非零选择部分特征。

3. 弹性网回归

弹性网回归是将岭回归和 Lasso 回归结合起来，混合 $L1$ 正则项和 $L2$ 正则

项，用新参数 α 控制二者所占比重：

$$\begin{aligned}\mathcal{L}(\beta, \lambda, \alpha) &= \frac{1}{2n} \|Y - X\beta\|_2^2 + \lambda \cdot \left[(1 - \alpha) \cdot \frac{1}{2} \|\beta\|_2^2 + \alpha \|\beta\|_1 \right] \\ &= \frac{1}{2n} \sum_{i=1}^n (y_i - \sum_{j=1}^p x_{ij} \beta_j)^2 \\ &\quad + \lambda \cdot \left[(1 - \alpha) \cdot \frac{1}{2} \sum_{j=1}^p |\beta_j|^2 + \alpha \sum_{j=1}^p |\beta_j| \right]\end{aligned}$$

显然，岭回归和 Lasso 回归是弹性网回归的特例： $\alpha=0$ 退化为岭回归， $\alpha=1$ 退化为Lasso 回归。

β 为模型参数, α 和 λ 都是需要调参的超参数。

需要注意的是，与普通最小二乘回归相比，正则化回归受自变量量纲的影响很大。因此，在应用正则化回归之前，最好对自变量进行标准化（建议用： $x' = x/sd(x)$ ），以便所有的自变量都在同一量纲上，以确保所有自变量在正则化过程中被公平对待。

二. 正则化回归案例

mlr3 做正则化回归是调用的 glmnet 包实现岭回归、Lasso 回归、弹性网回归，提供了两个学习器：

- “regr.glmnet”：调用 `glmnet::glmnet()`
- “regr.cv_glmnet”：调用 `glmnet::cv.glmnet()`，带内部交叉验证对 lambda 调参

两个原始函数的基本格式为：

`glmnet(x, y, family, alpha = 1, lambda, ...)`

`cv.glmnet(x, y, family, alpha = 1, lambda, ...)`

- x为自变量数据，y为因变量数据；
- family 指定因变量的类型，默认为“gaussian”（适合一般连续变量），

还可以是“binomial”，“poisson”，“multinomial”，“cox”，“mgaussian”；

- alpha 为弹性网混合系数，0 为岭回归，1 为 Lasso 回归，介于 0 和 1 之间为弹性网回归；
- lambda 设定惩罚程度的超参数

二者区别：cv.glmnet() 已经带有交叉验证对 lambda 调参，根据预测错误率或 RMSE 最小选择最优 lambda。

下面用波士顿房价数据演示如何实现三种正则化回归模型。

boston_housing 已经是可以直接使用的 mlr3 自带任务，这里采用从数据文件读取的方式。

1. 数据准备

(1) 创建任务

```
library(tidyverse)
```

```
library(mlr3verse)
```

读取数据，同时设置 chas 和 town 列按因子类型读取：

```
boston = read_csv("data/boston_housing.csv",
```

```
col_types = cols(chas = "f", town = "f"))
```

```
glimpse(boston)
```

查看数据

```

1 # 读取数据, 同时设置 chas 和 town 列按因子类型读取:
2 > glimpse(boston)
3 Rows: 506
4 Columns: 18
5 $ cmedv    <dbl> 24.0, 21.6, 34.7, 33.4, 36.2, 28.7, 22.9...
6 $ age      <dbl> 65.2, 78.9, 61.1, 45.8, 54.2, 58.7, 66.6...
7 $ b        <dbl> 396.90, 396.90, 392.83, 394.63, 396.90, ...
8 $ chas     <fct> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0...
9 $ crim     <dbl> 0.00632, 0.02731, 0.02729, 0.03237, 0.06...
10 $ dis      <dbl> 4.0900, 4.9671, 4.9671, 6.0622, 6.0622, ...
11 $ indus    <dbl> 2.31, 7.07, 7.07, 2.18, 2.18, 2.18, 7.87...
12 $ lat      <dbl> 42.2550, 42.2875, 42.2830, 42.2930, 42.2...
13 $ lon      <dbl> -70.9550, -70.9500, -70.9360, -70.9280, ...
14 $ lstat    <dbl> 4.98, 9.14, 4.03, 2.94, 5.33, 5.21, 12.4...
15 $ nox      <dbl> 0.538, 0.469, 0.469, 0.458, 0.458, 0.458...
16 $ ptratio  <dbl> 15.3, 17.8, 17.8, 18.7, 18.7, 18.7, 15.2...
17 $ rad      <dbl> 1, 2, 2, 3, 3, 3, 5, 5, 5, 5, 5, 5, 5, 4...
18 $ rm       <dbl> 6.575, 6.421, 7.185, 6.998, 7.147, 6.430...
19 $ tax      <dbl> 296, 242, 242, 222, 222, 222, 311, 311, ...
20 $ town     <fct> Nahant, Swampscott, Swampscott, Marblehe...
21 $ tract    <dbl> 2011, 2021, 2022, 2031, 2032, 2033, 2041...
22 $ zn       <dbl> 18.0, 0.0, 0.0, 0.0, 0.0, 0.0, 12.5, 12.5...
23 >

```

用`as_task_regr()`函数创建回归任务:

```
task = as_task_regr(boston, target = "cmedv")
```

```
task
```

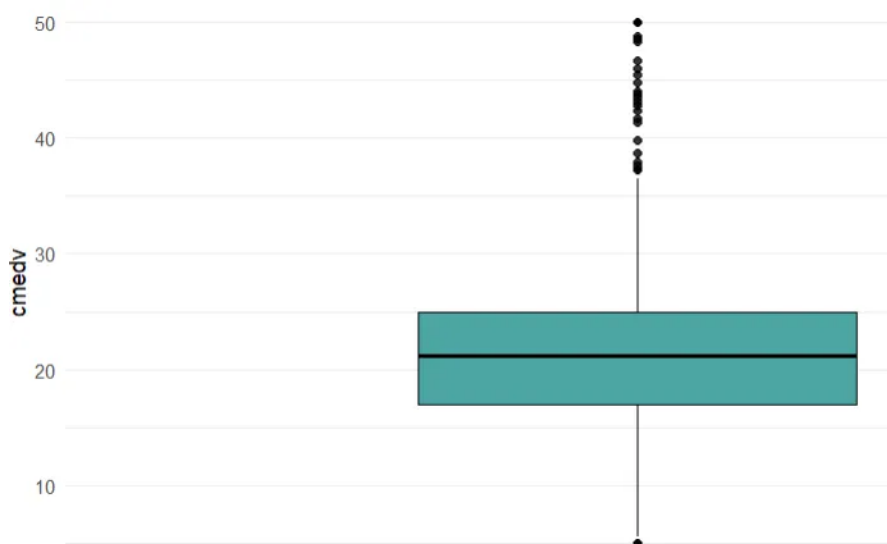
```

1 > task
2 <TaskRegr:boston> (506 x 18)
3 * Target: cmedv
4 * Properties: -
5 * Features (17):
6   - dbl (15): age, b, crim, dis, indus, lat, lon,
7     lstat, nox, ptratio, rad, rm, tax, tract, zn
8   - fct (2): chas, town

```

- 可视化任务

autoplot(task)



(2) 数据预处理

glmnet 包比较特殊，不能直接处理因子型变量，所以需要额外的数据准备：

- town 水平值过于分散，采用效应编码
- chas 是两水平因子，处理成虚拟变量

```
prep = po("encode", method = "treatment",
```

```
  affect_columns = selector_name("chas")) %>% %>%
```

```
po("encodeimpact",  
affect_columns = selector_name("town"))  
task = prep$train(task)[[1]]
```

注：glmnet() 默认 standardize = TRUE 即对自变量做标准化处理，回归系数始终按原始尺度返回。

(3) 划分训练集和测试集

做留出重抽样，80% 作为训练集，其余 20% 作为测试集。为了保持训练集、

测试集的因变量数据具有相似的分布，采用分层抽样方法：

- 用 partition() 函数对任务做划分，默认按因变量分层，返回列表包含训练集索引和测试集索引。

```
set.seed(123)
```

```
# 保证结果可重现
```

```
split = partition(task, ratio = 0.8) # 默认stratify=TRUE
```

1.岭回归预测波士顿房价

(1)选择学习器

```
lrm_ridge = lrm("regr.glmnet", alpha= 0) #需安装glmnet包
```

```
lrm_ridge
```

```
1 > lrn_ridge
2 <LearnerRegrGlmnet:regr.glmnet>: GLM with Elastic Net Regularization
3 * Model: -
4 * Parameters: alpha=0, family=gaussian
5 * Packages: mlr3, mlr3learners, glmnet
6 * Predict Types: [response]
7 * Feature Types: logical, integer, numeric
8 * Properties: weights
```

(2) 超参数调参

查看模型的所有超参数及默认值（结果略）

```
lrn_ridge$param_set
```

```

1 > lrn_ridge$param_set
2 <ParamSet(41)>
3 Key: <id>
4           id      class lower upper nlevels
5           <char> <char> <num> <num> <num>
6 1: alignment ParamFct   NA   NA      2
7 2: alpha ParamDbl     0    1    Inf
8 3: big ParamDbl    -Inf   Inf    Inf
9 4: devmax ParamDbl     0    1    Inf
10 5: dfmax ParamInt     0   Inf    Inf
11 6: eps ParamDbl     0    1    Inf
12 7: epsnr ParamDbl     0    1    Inf
13 8: exact ParamLgl    NA   NA      2
14 9: exclude ParamInt    1   Inf    Inf
15 10: exmx ParamDbl   -Inf   Inf    Inf
16 11: family ParamFct   NA   NA      2
17 12: fdev ParamDbl     0    1    Inf
18 13: gamma ParamDbl   -Inf   Inf    Inf
19 14: grouped ParamLgl   NA   NA      2
20 15: intercept ParamLgl   NA   NA      2
21 16: keep ParamLgl    NA   NA      2
22 17: lambda ParamUty   NA   NA    Inf
23 18: lambda.min.ratio ParamDbl    0    1    Inf
24 19: lower.limits ParamUty   NA   NA    Inf
25 20: maxit ParamInt     1   Inf    Inf
26 21: mnlam ParamInt     1   Inf    Inf
27 22: mxit ParamInt     1   Inf    Inf
28 23: mxitnr ParamInt     1   Inf    Inf
29 24: newoffset ParamUty   NA   NA    Inf
30 25: nlambdas ParamInt     1   Inf    Inf
31 26: offset ParamUty   NA   NA    Inf
32 27: parallel ParamLgl   NA   NA      2
33 28: penalty.factor ParamUty   NA   NA    Inf
34 29: pmax ParamInt     0   Inf    Inf
35 30: pmin ParamDbl     0    1    Inf
36 31: prec ParamDbl   -Inf   Inf    Inf
37 32: relax ParamLgl    NA   NA      2
38 33: s ParamDbl     0   Inf    Inf
39 34: standardize ParamLgl   NA   NA      2
40 35: standardize.response ParamLgl   NA   NA      2
41 36: thresh ParamDbl     0   Inf    Inf
42 37: trace.it ParamInt     0    1      2
43 38: type.gaussian ParamFct   NA   NA      2
44 39: type.logistic ParamFct   NA   NA      2

```



```

45 40:      type.multinomial ParamFct      NA      NA      2
46 41:      upper.limits ParamUty      NA      NA      Inf
47      id      class lower upper nlevels
48      default parents      value
49      <list> <list> <list>
50 1:      lambda
51 2:      1      0
52 3:      9.9e+35
53 4:      0.999
54 5: <NoDefault[0]>
55 6:      1e-06
56 7:      1e-08
57 8:      FALSE
58 9: <NoDefault[0]>
59 10:      250
60 11:      gaussian      gaussian
61 12:      1e-05
62 13:      1      relax
63 14:      TRUE
64 15:      TRUE
65 16:      FALSE
66 17: <NoDefault[0]>
67 18: <NoDefault[0]>
68 19: <NoDefault[0]>
69 20:      100000
70 21:      5
71 22:      100
72 23:      25
73 24: <NoDefault[0]>
74 25:      100
75 26:
76 27:      FALSE
77 28: <NoDefault[0]>
78 29: <NoDefault[0]>
79 30:      1e-09
80 31:      1e-10
81 32:      FALSE
82 33:      0.01
83 34:      TRUE
84 35:      FALSE
85 36:      1e-07
86 37:      0
87 38: <NoDefault[0]>      family
88 39: <NoDefault[0]>
89 40: <NoDefault[0]>
90 41: <NoDefault[0]>
91      default parents      value

```

需要对模型中的惩罚超参数 λ 做调参，对应超参数中的s。

使用自动调参器，需要设置学习器、重抽样方法、模型评估指标、搜索空间、终止条件、搜索方法

```
search_space = ps(s=p_dbl(0.001, 2))

at = auto_tuner(

tuner=tnr("random_search"),

learner= lrn_ridge,

resampling= rsmp("cv", folds= 3L), #3折交叉验证

measure= msr("regr.rmse"), #评估指标选rmse

search_space=search_space,

term_evals= 10) #计算10次终止
```

```
1 search_space = ps(s=p_dbl(0.001, 2))
2 at = auto_tuner(
3   tuner=tnr("random_search"),
4   learner= lrn_ridge,
5   resampling= rsmp("cv", folds= 3L), #3折交叉验证
6   measure= msr("regr.rmse"), #评估指标选rmse
7   search_space=search_space,
8   term_evals= 10) #计算10次终止
```

在训练集上启动自动调参过程

```
set.seed(123)

at$train(task, row_ids= split$train)
```

训练过程：

```

1 > at$train(task, row_ids= split$train)
2 ▾ INFO [09:25:29.328] [bbotk] Starting to optimize 1 parameter(s) with '<optimizerBatchRandomSearch>' and '<TerminatorEvals> [n_evals=10, k=0]'
```

	s	regr.rmse	warnings	errors
3 ▾ INFO [09:25:29.393] [bbotk] Evaluating 1 configuration(s)				
4 ▾ INFO [09:25:29.416] [mlr3] Running benchmark with 3 resampling iterations				
5 ▾ INFO [09:25:29.462] [mlr3] Applying learner 'regr.glmnet' on task 'boston' (iter 1/3)				
6 ▾ INFO [09:25:29.550] [mlr3] Applying learner 'regr.glmnet' on task 'boston' (iter 2/3)				
7 ▾ INFO [09:25:29.609] [mlr3] Applying learner 'regr.glmnet' on task 'boston' (iter 3/3)				
8 ▾ INFO [09:25:29.673] [mlr3] Finished benchmark				
9 ▾ INFO [09:25:29.711] [bbotk] Result of batch 1:				
10 ▾ INFO [09:25:29.714] [bbotk]				
11 ▾ INFO [09:25:29.714] [bbotk]	0.9806368	3.937937	0	0
12 ▾ INFO [09:25:29.714] [bbotk] runtime_learners				
13 ▾ INFO [09:25:29.714] [bbotk]	0.15			
14 ▾ INFO [09:25:29.714] [bbotk]				uhash
15 ▾ INFO [09:25:29.714] [bbotk]	19b10331-d283-4b36-99c1-dddd5a865deb			
16 ▾ INFO [09:25:29.724] [bbotk] Evaluating 1 configuration(s)				
17 ▾ INFO [09:25:29.741] [mlr3] Running benchmark with 3 resampling iterations				
18 ▾ INFO [09:25:29.748] [mlr3] Applying learner 'regr.glmnet' on task 'boston' (iter 1/3)				
19 ▾ INFO [09:25:29.807] [mlr3] Applying learner 'regr.glmnet' on task 'boston' (iter 2/3)				
20 ▾ INFO [09:25:29.902] [mlr3] Applying learner 'regr.glmnet' on task 'boston' (iter 3/3)				
21 ▾ INFO [09:25:29.956] [mlr3] Finished benchmark				
22 ▾ INFO [09:25:29.993] [bbotk] Result of batch 2:				
23 ▾ INFO [09:25:29.996] [bbotk]				s regr.rmse warnings errors runtime_learners
24 ▾ INFO [09:25:29.996] [bbotk]	1.708346	4.018843	0	0
	0.16			
25 ▾ INFO [09:25:29.996] [bbotk]				uhash
26 ▾ INFO [09:25:29.996] [bbotk]	42e5445a-0632-4a46-a07c-7d04c53ba404			
27 ▾ INFO [09:25:30.006] [bbotk] Evaluating 1 configuration(s)				
28 ▾ INFO [09:25:30.023] [mlr3] Running benchmark with 3 resampling iterations				
29 ▾ INFO [09:25:30.031] [mlr3] Applying learner 'regr.glmnet' on task 'boston' (iter 1/3)				
30 ▾ INFO [09:25:30.341] [mlr3] Applying learner 'regr.glmnet' on task 'boston' (iter 2/3)				

```

31 INFO [09:25:30.404] [mlr3] Applying learner 'regr.glmnet' on task 'boston' (iter 3/3)
32 INFO [09:25:30.461] [mlr3] Finished benchmark
33 INFO [09:25:30.499] [bbotk] Result of batch 3:
34 INFO [09:25:30.503] [bbotk]          s regr.rmse warnings errors runtime_learners
35 INFO [09:25:30.503] [bbotk]  0.303525  3.917136          0          0
36 INFO [09:25:30.503] [bbotk]          uhash
37 INFO [09:25:30.503] [bbotk]  09f64f86-d8c3-4596-98ea-45b2fe844123
38 INFO [09:25:30.512] [bbotk] Evaluating 1 configuration(s)
39 INFO [09:25:30.528] [mlr3] Running benchmark with 3 resampling iterations
40 INFO [09:25:30.537] [mlr3] Applying learner 'regr.glmnet' on task 'boston' (iter 1/3)
41 INFO [09:25:30.594] [mlr3] Applying learner 'regr.glmnet' on task 'boston' (iter 2/3)
42 INFO [09:25:30.650] [mlr3] Applying learner 'regr.glmnet' on task 'boston' (iter 3/3)
43 INFO [09:25:30.702] [mlr3] Finished benchmark
44 INFO [09:25:30.740] [bbotk] Result of batch 4:
45 INFO [09:25:30.744] [bbotk]          s regr.rmse warnings errors runtime_learners
46 INFO [09:25:30.744] [bbotk]  1.95479  4.047632          0          0
47 INFO [09:25:30.744] [bbotk]          uhash
48 INFO [09:25:30.744] [bbotk]  035f3126-9424-4647-b7e5-5610e857db36
49 INFO [09:25:30.753] [bbotk] Evaluating 1 configuration(s)
50 INFO [09:25:30.770] [mlr3] Running benchmark with 3 resampling iterations
51 INFO [09:25:30.777] [mlr3] Applying learner 'regr.glmnet' on task 'boston' (iter 1/3)
52 INFO [09:25:30.843] [mlr3] Applying learner 'regr.glmnet' on task 'boston' (iter 2/3)
53 INFO [09:25:30.897] [mlr3] Applying learner 'regr.glmnet' on task 'boston' (iter 3/3)
54 INFO [09:25:30.951] [mlr3] Finished benchmark
55 INFO [09:25:30.987] [bbotk] Result of batch 5:
56 INFO [09:25:30.991] [bbotk]          s regr.rmse warnings errors
57 INFO [09:25:30.991] [bbotk]  0.1225037  3.917136          0          0
58 INFO [09:25:30.991] [bbotk] runtime_learners
59 INFO [09:25:30.991] [bbotk]          0.15
60 INFO [09:25:30.991] [bbotk]          uhash
61 INFO [09:25:30.991] [bbotk]  5b73e037-791f-4905-9f8e-c3c013d25dda
62 INFO [09:25:31.000] [bbotk] Evaluating 1 configuration(s)
63 INFO [09:25:31.017] [mlr3] Running benchmark with 3 resampling iterations

```

```

64 INFO [09:25:31.025] [mlr3] Applying learner 'regr.glmnet' on task 'boston' (iter 1/3)
65 INFO [09:25:31.095] [mlr3] Applying learner 'regr.glmnet' on task 'boston' (iter 2/3)
66 INFO [09:25:31.151] [mlr3] Applying learner 'regr.glmnet' on task 'boston' (iter 3/3)
67 INFO [09:25:31.205] [mlr3] Finished benchmark
68 INFO [09:25:31.245] [bbotk] Result of batch 6:
69 INFO [09:25:31.250] [bbotk]          s regr.rmse warnings errors
70 INFO [09:25:31.250] [bbotk] 0.1776382 3.917136          0          0
71 INFO [09:25:31.250] [bbotk] runtime_learners
72 INFO [09:25:31.250] [bbotk]          0.12
73 INFO [09:25:31.250] [bbotk]                                uhash
74 INFO [09:25:31.250] [bbotk] ebb85566-0b7b-4a21-9360-fad3a2be8218
75 INFO [09:25:31.260] [bbotk] Evaluating 1 configuration(s)
76 INFO [09:25:31.279] [mlr3] Running benchmark with 3 resampling iterations
77 INFO [09:25:31.288] [mlr3] Applying learner 'regr.glmnet' on task 'boston' (iter 1/3)
78 INFO [09:25:31.348] [mlr3] Applying learner 'regr.glmnet' on task 'boston' (iter 2/3)
79 INFO [09:25:31.406] [mlr3] Applying learner 'regr.glmnet' on task 'boston' (iter 3/3)
80 INFO [09:25:31.458] [mlr3] Finished benchmark
81 INFO [09:25:31.507] [bbotk] Result of batch 7:
82 INFO [09:25:31.511] [bbotk]          s regr.rmse warnings errors
83 INFO [09:25:31.511] [bbotk] 0.6756525 3.917136          0          0
84 INFO [09:25:31.511] [bbotk] runtime_learners
85 INFO [09:25:31.511] [bbotk]          0.15
86 INFO [09:25:31.511] [bbotk]                                uhash
87 INFO [09:25:31.511] [bbotk] 24aca6a7-5769-4a65-96d9-ffdd1f54abd7
88 INFO [09:25:31.519] [bbotk] Evaluating 1 configuration(s)
89 INFO [09:25:31.535] [mlr3] Running benchmark with 3 resampling iterations
90 INFO [09:25:31.543] [mlr3] Applying learner 'regr.glmnet' on task 'boston' (iter 1/3)
91 INFO [09:25:31.600] [mlr3] Applying learner 'regr.glmnet' on task 'boston' (iter 2/3)
92 INFO [09:25:31.654] [mlr3] Applying learner 'regr.glmnet' on task 'boston' (iter 3/3)
93 INFO [09:25:31.706] [mlr3] Finished benchmark
94 INFO [09:25:31.743] [bbotk] Result of batch 8:
95 INFO [09:25:31.747] [bbotk]          s regr.rmse warnings errors
96 INFO [09:25:31.747] [bbotk] 0.4752222 3.917136          0          0
97 INFO [09:25:31.747] [bbotk] runtime_learners
98 INFO [09:25:31.747] [bbotk]          0.13
99 INFO [09:25:31.747] [bbotk]                                uhash

```

```

100 INFO [09:25:31.747] [bbotk] 1e24c8b6-b8b4-46bb-bb7e-02b7f4aec32f
101 INFO [09:25:31.756] [bbotk] Evaluating 1 configuration(s)
102 INFO [09:25:31.772] [mlr3] Running benchmark with 3 resampling iteration
s
103 INFO [09:25:31.780] [mlr3] Applying learner 'regr.glmnet' on task 'bosto
n' (iter 1/3)
104 INFO [09:25:31.844] [mlr3] Applying learner 'regr.glmnet' on task 'bosto
n' (iter 2/3)
105 INFO [09:25:31.898] [mlr3] Applying learner 'regr.glmnet' on task 'bosto
n' (iter 3/3)
106 INFO [09:25:31.951] [mlr3] Finished benchmark
107 INFO [09:25:31.989] [bbotk] Result of batch 9:
108 INFO [09:25:31.992] [bbotk] s regr.rmse warnings errors
109 INFO [09:25:31.992] [bbotk] 0.6376707 3.917136 0 0
110 INFO [09:25:31.992] [bbotk] runtime_learners
111 INFO [09:25:31.992] [bbotk] 0.15
112 INFO [09:25:31.992] [bbotk] uhash
113 INFO [09:25:31.992] [bbotk] 4d903261-eb23-4bf0-86f1-e59cbacad0d9
114 INFO [09:25:32.002] [bbotk] Evaluating 1 configuration(s)
115 INFO [09:25:32.018] [mlr3] Running benchmark with 3 resampling iteration
s
116 INFO [09:25:32.026] [mlr3] Applying learner 'regr.glmnet' on task 'bosto
n' (iter 1/3)
117 INFO [09:25:32.082] [mlr3] Applying learner 'regr.glmnet' on task 'bosto
n' (iter 2/3)
118 INFO [09:25:32.137] [mlr3] Applying learner 'regr.glmnet' on task 'bosto
n' (iter 3/3)
119 INFO [09:25:32.199] [mlr3] Finished benchmark
120 INFO [09:25:32.237] [bbotk] Result of batch 10:
121 INFO [09:25:32.241] [bbotk] s regr.rmse warnings errors
122 INFO [09:25:32.241] [bbotk] 0.2934178 3.917136 0 0
123 INFO [09:25:32.241] [bbotk] runtime_learners
124 INFO [09:25:32.241] [bbotk] 0.13
125 INFO [09:25:32.241] [bbotk] uhash
126 INFO [09:25:32.241] [bbotk] 3b6eb25f-f7a4-4905-ad98-2c1a51043531
127 INFO [09:25:32.265] [bbotk] Finished optimizing after 10 evaluation(s)
128 INFO [09:25:32.267] [bbotk] Result:
129 INFO [09:25:32.271] [bbotk] s learner_param_vals x_domain regr.
rmse
130 INFO [09:25:32.271] [bbotk] <num> <list> <list> <
num>
131 INFO [09:25:32.271] [bbotk] 0.303525 <list[3]> <list[1]> 3.91
7136

```

最初出现过下面的错误提示：(估计是代码问题)

```
> #在训练集上启动自动调参过程
> set.seed(123)
> at$train(task, row_ids= split$train)
错误: Type 'regr' of <TaskRegr:boston> does not match type
'classif' of <AutoTuner:classif.svm.tuned>
> #查看最优超参数
> at$tuning_result
```

查看最优超参数

at\$tuning_result

```
1 > #查看最优超参数
2 > at$tuning_result
3           s learner_param_vals  x_domain regr.rmse
4           <num>                <list>    <list>    <num>
5 1: 0.303525                    <list[3]> <list[1]>  3.917136
```

老师课件结果是下面：

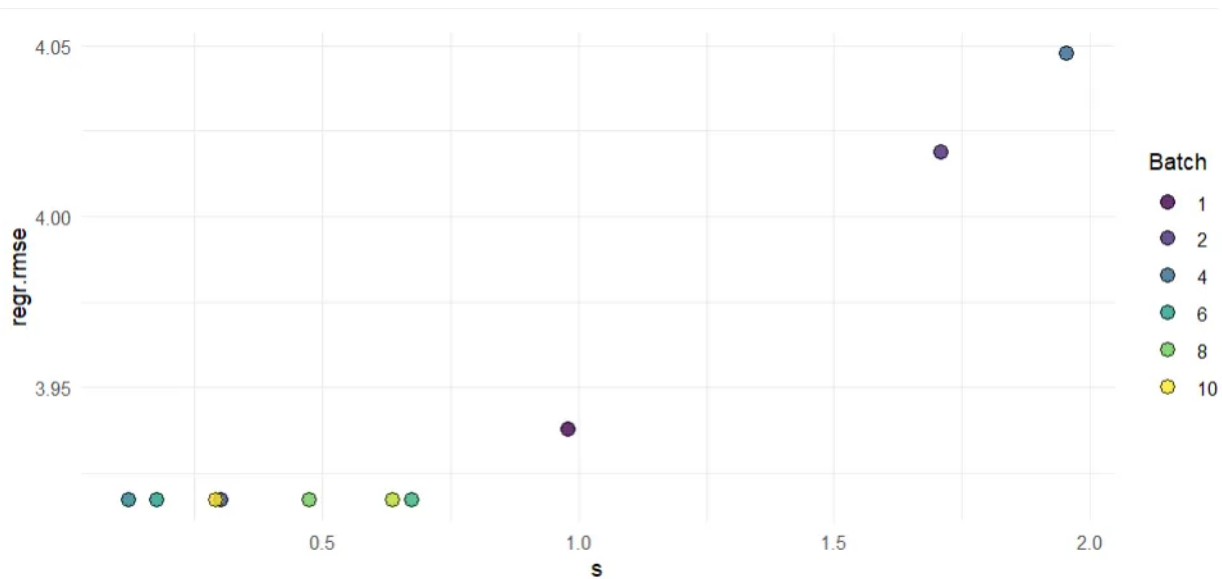
查看最优超参数

at\$tuning_result

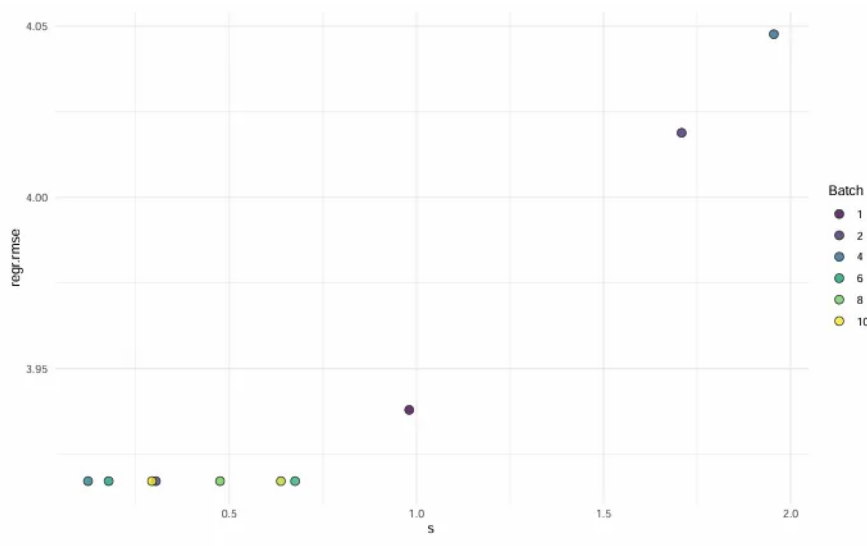
```
#>           s learner_param_vals  x_domain regr.rmse
#>    <num>                <list>    <list>    <num>
#> 1: 0.304                    <list[3]> <list[1]>    3.92
```

可视化超参数变化对模型性能的影响

autoplot(at\$tuning_instance)



下面是老师的结果：



(3) 训练模型

自动调参器已经使用超参数调参得到最优超参数在整个训练集上重新训练，取出该模型结果，可以跟直接用 `glmnet()` 训练模型得到的模型结果一样使用。

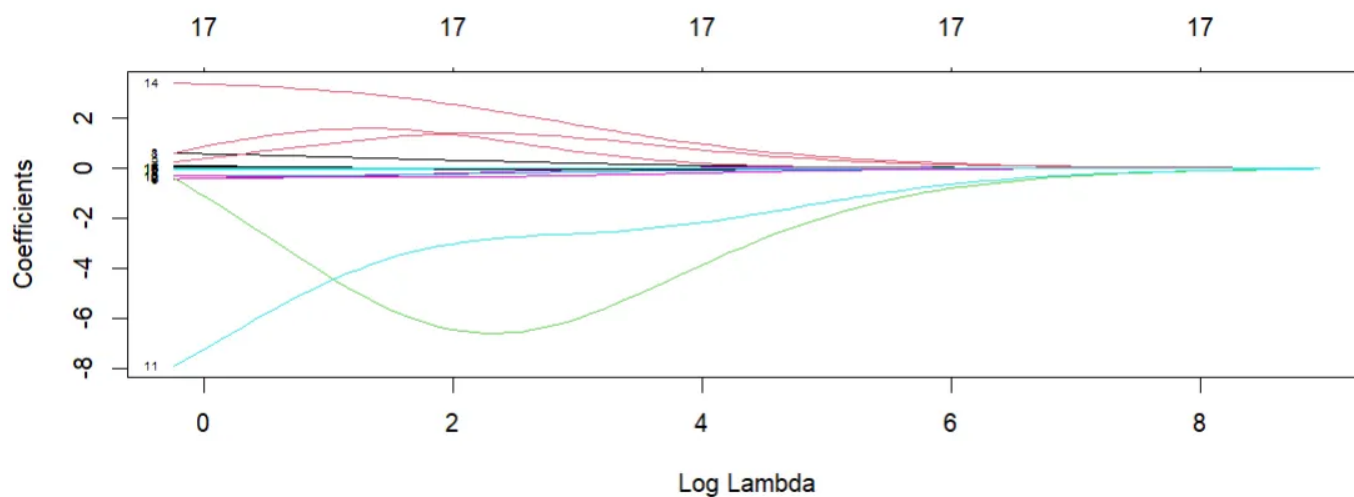
```
model = at$learner$model
```

提取岭回归模型系数，绘制岭迹图：

```
coef(model, s = at$tuning_result$s)
```

```
1 > coef(model, s = at$tuning_result$s)
2 18 x 1 sparse Matrix of class "dgCMatrix"
3      s1
4 (Intercept) -3.989849e+01
5 town        5.986355e-01
6 chas        2.698817e-01
7 age        -1.057666e-02
8 b           7.005990e-03
9 crim       -4.766126e-02
10 dis       -4.307300e-01
11 indus      1.065570e-01
12 lat        6.193917e-01
13 lon       -4.004903e-01
14 lstat     -3.036641e-01
15 nox       -7.908075e+00
16 ptratio   -3.197294e-01
17 rad        3.153456e-02
18 rm        3.400158e+00
19 tax        1.584127e-04
20 tract     -3.292376e-04
21 zn        -2.869692e-03
22 >
```

```
plot(model, xvar = "lambda", label = TRUE)
```



可见，随着 λ 的增大，有越来越多的指标回归系数压迫到接近0。

或者直接用 `autoplot()` 绘图：

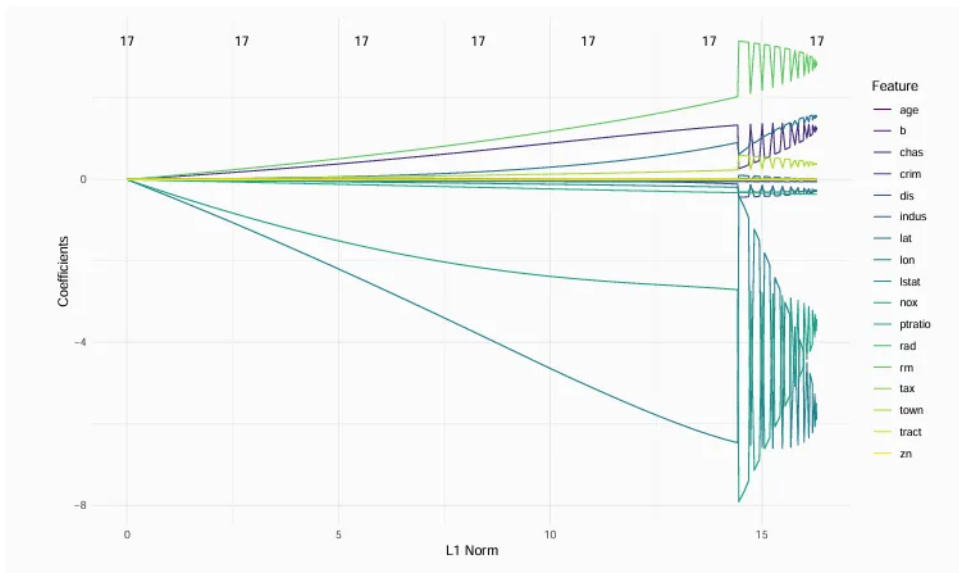
```
autoplot(at$learner, type = "ggfortify")
```

```
> #或者直接用 autoplot() 绘图:
```

```
> autoplot(at$learner, type = "ggfortify")
```

错误：The following packages could not be loaded:
d: ggfortify

需要单独安装这个包。



(4) 模型预测

在测试集上做预测：

```
pred_ridge = at$predict(task, row_ids = split$test)
```

```
pred_ridge
```

```

1 > pred_ridge
2 <PredictionRegr> for 101 observations:
3   row_ids truth  response
4         2  21.6 27.725306
5         3  34.7 32.025778
6        30  21.0 19.536576
7  ---
8       491   8.1  9.011048
9       497  19.7 16.066158
10      500  17.5 18.663841

```

(5) 模型评估

计算测试集上的泛化性能：均方根误差与 R 方

```
pred_ridge$score(msrs(c("regr.rmse", "regr.rsq")))
```

```

1 > pred_ridge$score(msrs(c("regr.rmse", "regr.rsq")))
2 regr.rmse regr.rsq
3 3.2135292 0.8728367

```

(6) 预测新数据

```
newdata = task$data()[1:5, -1] # 作为新数据
```

```
at$predict_newdata(newdata)
```

```

1 > at$predict_newdata(newdata)
2 <PredictionRegr> for 5 observations:
3   row_ids truth response
4         1   NA 27.20859
5         2   NA 27.72531
6         3   NA 32.02578
7         4   NA 33.49367
8         5   NA 33.20021

```

2. Lasso回归预测波士顿房价

跟岭回归几乎一样的代码，只需要选择学习器时设置alpha=1。

(1)选择学习器

`lrn_lasso = lrn("regr.glmnet", alpha= 1) #需安装glmnet包`

`lrn_lasso`

```

1 > lrn_lasso
2 <LearnerRegrGlmnet:regr.glmnet>: GLM with Elastic Net Regularization
3 * Model: -
4 * Parameters: alpha=1, family=gaussian
5 * Packages: mlr3, mlr3learners, glmnet
6 * Predict Types: [response]
7 * Feature Types: logical, integer,
8   numeric
9 * Properties: weights
10 >

```

(2) 超参数调参

查看模型的所有超参数及默认值

lrn_lasso\$param_set

```
1 > lrn_lasso$param_set
2 <ParamSet(41)>
3 Key: <id>
4           id      class lower
5     <char>  <char> <num>
6 1: alignment ParamFct   NA
7 2:   alpha ParamDbL    0
8 3:    big ParamDbL -Inf
9 4: devmax ParamDbL    0
10 5: dfmax ParamInt    0
11 6:   eps ParamDbL    0
12 7: epsnr ParamDbL    0
13 8:  exact ParamLgl   NA
14 9: exclude ParamInt    1
15 . . .
```

需要对模型中的惩罚超参数 λ 做调参，对应超参数中的s。使用自动调参器，需要设置学习器、重抽样方法、模型评估指标、搜索空间、终止条件、搜索方法

```
1 search_space = ps(s=p_dbl(0.001, 2))
2
3 at = auto_tuner(
4   tuner=tnr("random_search"),
5   learner= lrn_lasso,
6   resampling= rsmpl("cv", folds= 3L), #3折交叉验证
7   measure= msr("regr.rmse"), #评估指标选rmse
8   search_space=search_space,
9   term_evals= 10) #计算10次终止
```

在训练集上启动自动调参过程

```
set.seed(123)
```

```
at$train(task, row_ids= split$train)
```

过程略。

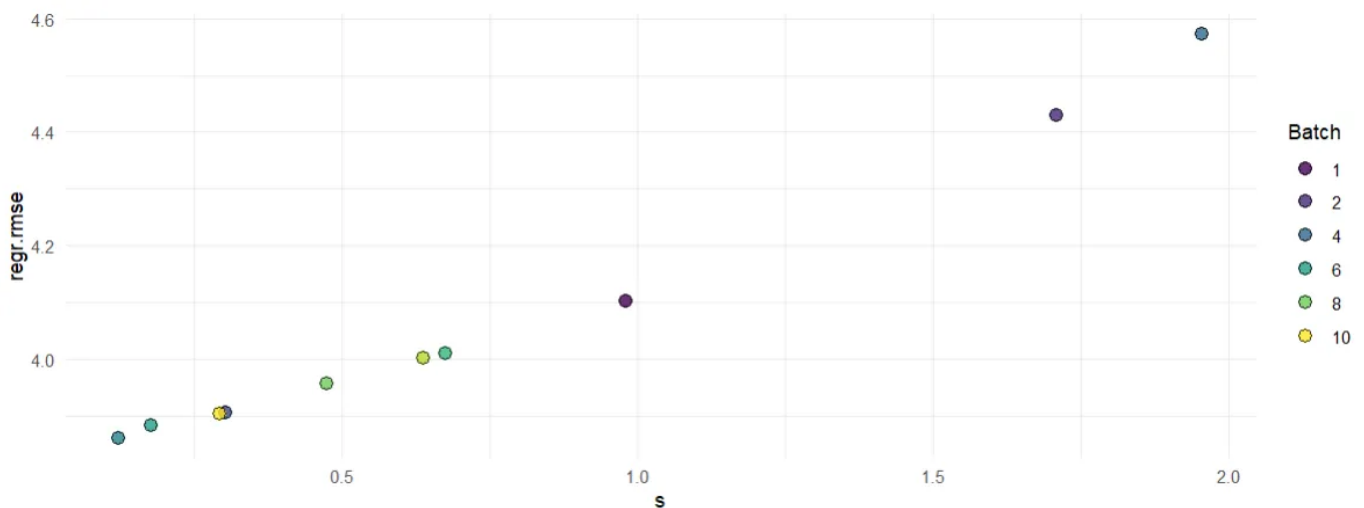
查看最优超参数

```
at$tuning_result
```

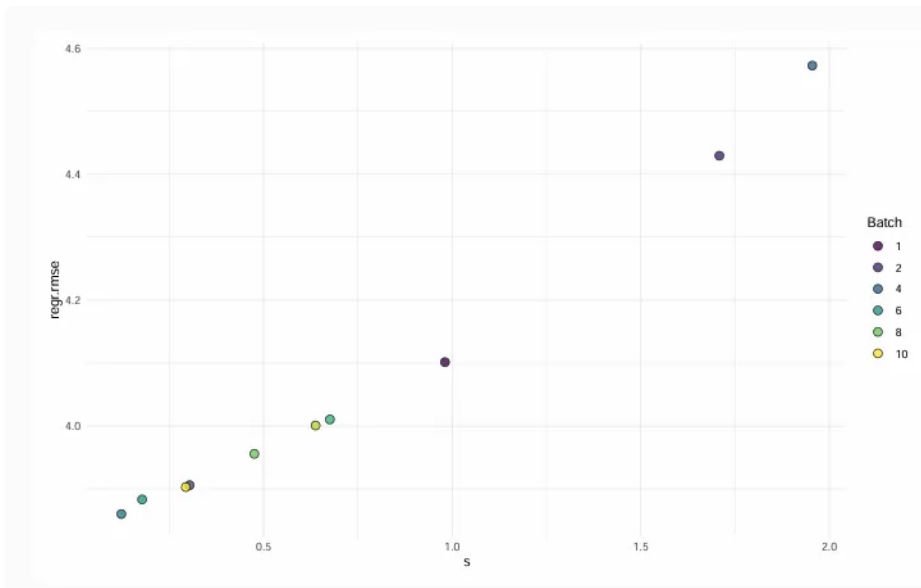
```
1 > #查看最优超参数
2 > at$tuning_result
3           s learner_param_vals x_domain regr.rmse
4           <num>           <list>   <list>   <num>
5 1: 0.1225037           <list[3]> <list[1]> 3.860208
```

- 可视化超参数变化对模型性能的影响

```
autoplot(at$tuning_instance)
```



下面是老师课件结果。



(3) 训练模型

自动调参器已经使用超参数调参得到最优超参数在整个训练集上重新训练，取出该模型结果，可以跟直接用 `glmnet()` 训练模型得到的模型结果一样使用。

```
model = at$learner$model
```

- 提取Lasso 回归模型系数：

```
coef(model, s = at$tuning_result$s)
```



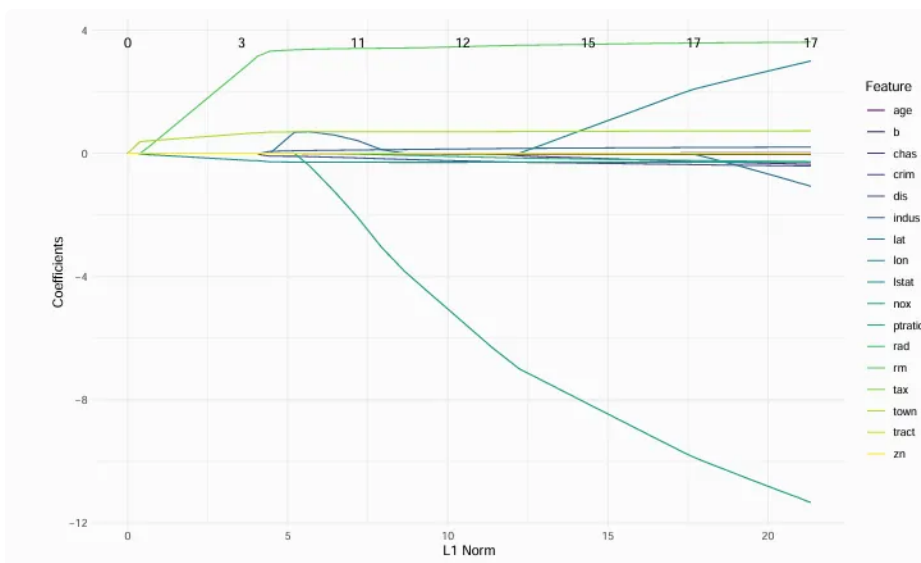
```

1 > coef(model, s = at$tuning_result$s)
2 18 x 1 sparse Matrix of class "dgCMatrix"
3
4 (Intercept) 3.8889524733
5 town        0.7018007401
6 chas        .
7 age         .
8 b           0.0058758279
9 crim        -0.0141305145
10 dis         -0.1901648290
11 indus       0.1193572201
12 lat         0.0431100663
13 lon         .
14 lstat       -0.2935041550
15 nox         -3.5447355812
16 ptratio     -0.0565460351
17 rad         .
18 rm          3.4148549528
19 tax         .
20 tract       -0.0002025079
21 zn          .
22 >

```

- 绘制Lasso 迹图

`autoplot(at$learner, type = "ggfortify")`



从右往左看，随着 $L1$ 范数的减小（总回归系数减小， λ 增大），有越来越多的指标回归系数压迫到 0。

(4) 模型预测

在测试集上做预测：

```
pred_lasso = at$predict(task, row_ids = split$test)
```

```
pred_lasso
```

```
▼ R |
1 > pred_lasso
2 <PredictionRegr> for 101 observations:
3   row_ids truth  response
4         2  21.6 28.053602
5         3  34.7 32.138046
6        30  21.0 19.777358
7 ----
8       491   8.1  8.775198
9       497  19.7 15.839494
10      500  17.5 18.293259
```

(5) 模型评估

计算测试集上的泛化性能：均方根误差与 R 方

```
pred_lasso$score(msrs(c("regr.rmse", "regr.rsq")))
```

```

1 > #计算测试集上的泛化性能：均方根误差与 R 方
2 > pred_lasso$score(msrs(c("regr.rmse", "regr.rsq")))
3 regr.rmse regr.rsq
4 3.2247900 0.8719439

```

(6)预测新数据

`newdata = task$data()[1:5, -1]` #作为新数据

`at$predict_newdata(newdata)`

```

1 > at$predict_newdata(newdata)
2 <PredictionRegr> for 5 observations:
3   row_ids truth response
4         1   NA 26.38444
5         2   NA 28.05360
6         3   NA 32.13805
7         4   NA 34.26463
8         5   NA 34.08480
9 >

```

3.弹性网回归预测波士顿房价

跟岭回归、Lasso回归几乎一样的代码，只是选择学习器时不设置alpha参数，以及调参时多调参一个alpha。

(1)选择学习器

`lrm_glmnet = lrm("regr.glmnet")` #需安装glmnet包

`lrm_glmnet`

```
1 > lrn_glmnet
2 <LearnerRegrGlmnet:regr.glmnet>: GLM with Elastic Net Regularization
3 * Model: -
4 * Parameters: family=gaussian
5 * Packages: mlr3, mlr3learners, glmnet
6 * Predict Types: [response]
7 * Feature Types: logical, integer,
8   numeric
9 * Properties: weights
```

(2) 超参数调参

查看模型的所有超参数及默认值

```
lrn_glmnet$param_set
```

```

1 > lrn_glmnet$param_set
2 <ParamSet(41)>
3 Key: <id>
4           id      class lower
5           <char> <char> <num>
6 1: alignment ParamFct   NA
7 2: alpha ParamDbl    0
8 3: big ParamDbl  -Inf
9 4: devmax ParamDbl    0
10 5: dfmax ParamInt    0
11 6: eps ParamDbl    0
12 7: epsnr ParamDbl    0
13 8: exact ParamLgl   NA
14 9: exclude ParamInt  1
15 10: exmx ParamDbl  -Inf
16 11: family ParamFct  NA
17 12: fdev ParamDbl    0
18 13: gamma ParamDbl  -Inf
19 14: grouped ParamLgl  NA
20 15: intercept ParamLgl NA
21 16: keep ParamLgl   NA
22 17: lambda ParamUty  NA
23 18: lambda.min.ratio ParamDbl 0
24 19: lower.limits ParamUty  NA
25 20: maxit ParamInt  1
26 21: mnlam ParamInt  1
27 22: mxit ParamInt  1
28 23: mxitnr ParamInt  1
29 24: newoffset ParamUty  NA
30 25: nlambda ParamInt  1
31 26: offset ParamUty  NA

```

需要对模型中的权重 α 和惩罚超参数 λ 做调参，对应超参数中的s。

使用自动调参器，需要设置学习器、重抽样方法、模型评估指标、搜索空间、终止条件、搜索方法

```

1 search_space = ps(alpha= p_dbl(0, 1),
2                   s=p_dbl(0.001, 2))
3
4 at = auto_tuner(
5   tuner=tnr("random_search"),
6   learner= lrn_glmnet,
7   resampling= rsmpl("cv", folds= 3), #3折交叉验证
8   measure= msr("regr.rmse"), #评估指标选rmse
9   search_space=search_space,
10  term_evals= 10) #计算10次终止
11
12

```

在训练集上启动自动调参过程

set.seed(123)

at\$train(task, row_ids= split\$train)

查看最优超参数

at\$tuning_result

```

1 > #查看最优超参数
2
3 > at$tuning_result
4       alpha      s learner_param_vals
5       <num>    <num>                <list>
6 1: 0.3184946 0.3487936                <list[3]>
7       x_domain regr.rmse
8       <list>    <num>
9 1: <list[2]>    3.876519
10 >

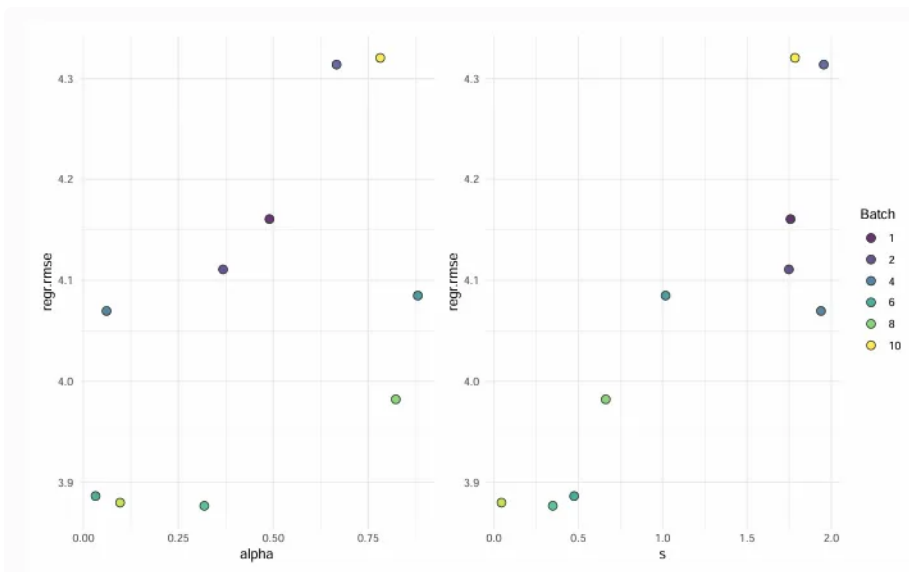
```

查看最优超参数

```
at$tuning_result
#>      alpha      s learner_param_vals x_domain regr.rmse
#>      <num> <num>                <list>   <list>      <num>
#> 1: 0.318 0.349                <list[3]> <list[2]>      3.88
```

- 可视化超参数变化对模型性能的影响

autoplot(at\$tuning_instance)



(3) 训练模型

自动调参器已经使用超参数调参得到最优超参数在整个训练集上重新训练，取出该模型结果，可以跟直接用 `glmnet()` 训练模型得到的模型结果一样使用。

```
model = at$learner$model
```

- 提取弹性网回归模型系数：

```
coef(model, s = at$tuning_result$s)
```

```
1 > coef(model, s = at$tuning_result$s)
2 18 x 1 sparse Matrix of class "dgCMatrix"
3               s1
4 (Intercept)  7.7857179956
5 town         0.6647284626
6 chas         .
7 age         -0.0008238749
8 b            0.0059202762
9 crim        -0.0194763388
10 dis         -0.2421870297
11 indus       0.1000830418
12 lat         .
13 lon         .
14 lstat       -0.2979543835
15 nox         -3.9314394768
16 ptratio     -0.1168453668
17 rad         .
18 rm         3.3885128895
19 tax         .
20 tract       -0.0002220278
21 zn         .
22 >
```

下面是老师的结果：

- 提取弹性网回归模型系数：

```
coef(model, s = at$tuning_result$s)
#> 18 x 1 sparse Matrix of class "dgCMatrix"
#>           s1
#> (Intercept)  7.785718
#> town        0.664728
#> chas        .
#> age        -0.000824
#> b          0.005920
#> crim       -0.019476
#> dis       -0.242187
```

可视化学习器：

```
autoplot(at$learner, type = "ggfortify")
```

从右往左看，随着 $L1$ 范数的减小（总回归系数减小， λ 增大），有越来越多的指标回归系数压迫到（接近）0。

(4) 模型预测

在测试集上做预测：

```
pred_glmnet = at$predict(task, row_ids = split$test)
```

```
pred_glmnet
```

```

1 > pred_glmnet
2 <PredictionRegr> for 101 observations:
3   row_ids truth  response
4         2  21.6 27.972836
5         3  34.7 32.074555
6        30  21.0 19.885501
7  ---
8       491   8.1  8.769547
9       497  19.7 15.965446
10      500  17.5 18.462521

```

(5) 模型评估

计算测试集上的泛化性能：均方根误差与 R 方

```
pred_glmnet$score(msrs(c("regr.rmse", "regr.rsq")))
```

```

1 > pred_glmnet$score(msrs(c("regr.rmse", "regr.rsq")))
2 regr.rmse regr.rsq
3  3.2301054 0.8715214

```

(6) 预测新数据

```
newdata = task$data()[1:5, -1] # 作为新数据
```

```
at$predict_newdata(newdata)
```

```
1 > at$predict_newdata(newdata)
2 <PredictionRegr> for 5 observations:
3   row_ids truth response
4         1   NA 26.74641
5         2   NA 27.97284
6         3   NA 32.07456
7         4   NA 34.03909
8         5   NA 33.83745
```

regr.glmnet 学习器也支持正则化的广义线性模型，比如正则化 Logistic 回归，设置超参数 `family = "binomial"` 即可。

另外，给模型增加正则项，对模型的复杂度施加一定的惩罚，不止适用于多元线性回归，还能适用于很多机器学习模型，比如决策树、支持向量机等。

```

1  #(1) 创建任务
2  library(tidyverse)
3  library(mlr3verse)
4
5  boston = read_csv("D:/24暑期-R机器学习班/data/boston_housing.csv",
6                  col_types = cols(chas = "f", town = "f"))
7  # 读取数据, 同时设置 chas 和 town 列按因子类型读取:
8  glimpse(boston)
9
10 #用as_task_regr()函数创建回归任务:
11 task = as_task_regr(boston, target = "cmedv")
12 task
13
14 autoplot(task) # 可视化任务
15
16 prep = po("encode", method = "treatment",
17          affect_columns = selector_name("chas")) %>%
18   po("encodeimpact",
19     affect_columns = selector_name("town"))
20
21 # town 水平值过于分散, 采用效应编码
22 # chas 是两水平因子, 处理成虚拟变量
23
24 task = prep$train(task)[[1]]
25
26 set.seed(123)
27 # 保证结果可重现
28 split = partition(task, ratio = 0.8) # 默认 stratify=TRUE
29 #用 partition() 函数对任务做划分, 默认按因变量分层, 返回列表包含训练集索引和测试集索引。
30 #(1)选择学习器
31 lrn_ridge = lrn("regr.glmnet", alpha = 0) #需安装glmnet包
32 lrn_ridge
33
34 #查看模型的所有超参数及默认值 (结果略)
35 lrn_ridge$param_set
36
37
38
39
40 search_space = ps(s = p_dbl(0.001, 2))
41 at = auto_tuner(
42   tuner = tnr("random_search"),
43   learner = lrn_ridge,

```

```

44     resampling= rsmpl("cv", folds= 3L), #3折交叉验证
45     measure= msr("regr.rmse"), #评估指标选rmse
46     search_space=search_space,
47     term_evals= 10) #计算10次终止
48
49 #在训练集上启动自动调参过程
50 set.seed(123)
51 at$train(task, row_ids= split$train)
52 #查看最优超参数
53 at$tuning_result
54
55
56 #可视化超参数变化对模型性能的影响
57 autoplot(at$tuning_instance)
58
59 #训练模型
60 model = at$learner$model
61 #提取岭回归模型系数，绘制岭迹图：
62 coef(model, s = at$tuning_result$s)
63 plot(model, xvar = "lambda", label = TRUE)
64
65
66 #或者直接用 autoplot() 绘图：
67 autoplot(at$learner, type = "ggfortify")
68
69 #在测试集上做预测：
70 pred_ridge = at$predict(task, row_ids = split$test)
71 pred_ridge
72
73
74 #模型评估
75 #计算测试集上的泛化性能：均方根误差与 R 方
76 pred_ridge$score(msrs(c("regr.rmse", "regr.rsq")))
77
78 #6) 预测新数据
79 newdata =task$data()[1:5,-1] #作为新数据
80 at$predict_newdata(newdata)
81
82
83
84
85 #Lasso回归预测波士顿房价
86
87 # (1) 选择学习器
88 lrn_lasso = lrn("regr.glmnet", alpha= 1) #需安装glmnet包
89 lrn_lasso
90

```

```

91  #(2) 超参数调参
92  #查看模型的所有超参数及默认值
93  lrn_lasso$param_set
94
95
96  search_space = ps(s=p_dbl(0.001, 2))
97
98  at = auto_tuner(
99    tuner=tnr("random_search"),
100    learner= lrn_lasso,
101    resampling= rsmpl("cv", folds= 3L), #3折交叉验证
102    measure= msr("regr.rmse"), #评估指标选rmse
103    search_space=search_space,
104    term_evals= 10) #计算10次终止
105
106  #在训练集上启动自动调参过程
107  set.seed(123)
108  at$train(task, row_ids= split$train)
109  #查看最优超参数
110  at$tuning_result
111  #可视化超参数变化对模型性能的影响
112  autoplot(at$tuning_instance)
113
114  #训练模型
115  model = at$learner$model
116  #提取Lasso 回归模型系数:
117  coef(model, s = at$tuning_result$s)
118
119
120  #在测试集上做预测:
121  pred_lasso = at$predict(task, row_ids = split$test)
122  pred_lasso
123
124  #模型评估
125  #计算测试集上的泛化性能: 均方根误差与 R 方
126  pred_lasso$score(msrs(c("regr.rmse", "regr.rsq")))
127
128
129  # (6) 预测新数据
130  newdata =task$data()[1:5,-1] #作为新数据
131  at$predict_newdata(newdata)
132
133
134  #弹性网回归预测波士顿房价
135
136  lrn_glmnet= lrn("regr.glmnet")#需安装glmnet包
137  lrn_glmnet

```

```

138
139   lrn_glmnet$param_set
140
141   search_space = ps(alpha= p_dbl(0, 1),
142                     s=p_dbl(0.001, 2))
143
144   at = auto_tuner(
145     tuner=tnr("random_search"),
146     learner= lrn_glmnet,
147     resampling= rsmpl("cv", folds= 3), #3折交叉验证
148     measure= msr("regr.rmse"), #评估指标选rmse
149     search_space=search_space,
150     term_evals= 10) #计算10次终止
151
152
153   #在训练集上启动自动调参过程
154   set.seed(123)
155   at$train(task, row_ids= split$train)
156   #查看最优超参数
157   at$tuning_result
158   # 可视化超参数变化对模型性能的影响
159   autoplot(at$tuning_instance)
160
161   #训练模型
162   model = at$learner$model
163   # 提取弹性网回归模型系数:
164   coef(model, s = at$tuning_result$s)
165   #可视化学习器:
166   autoplot(at$learner, type = "ggfortify")
167
168   #在测试集上做预测:
169   pred_glmnet = at$predict(task, row_ids = split$test)
170   pred_glmnet
171   #(5) 模型评估
172   #计算测试集上的泛化性能: 均方根误差与 R 方
173   pred_glmnet$score(msrs(c("regr.rmse", "regr.rsq")))
174
175
176   #(6) 预测新数据
177   newdata =task$data()[1:5,-1] #作为新数据
178   at$predict_newdata(newdata)

```

本文主要参考 (Bernd Bischl and Lang, 2024), (刘顺祥, 2018).

Bernd Bischl, Raphael Sonabend, L. K. and Lang, M., editors

(2024). Applied Machine Learning Using mlr3 in R. CRC Press.

刘顺祥 (2018). 从零开始学 Python—数据分析与挖掘. 清华大学出版社, 北京.

王圣元 (2020). 快乐机器学习. 电子工业出版社.