正则化回归

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，增加或剔除一个自变量后，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λ=0时，损失函数退化为多元线性回归模型的损失函数；
- 当$\lambda \to \infty$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

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

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：

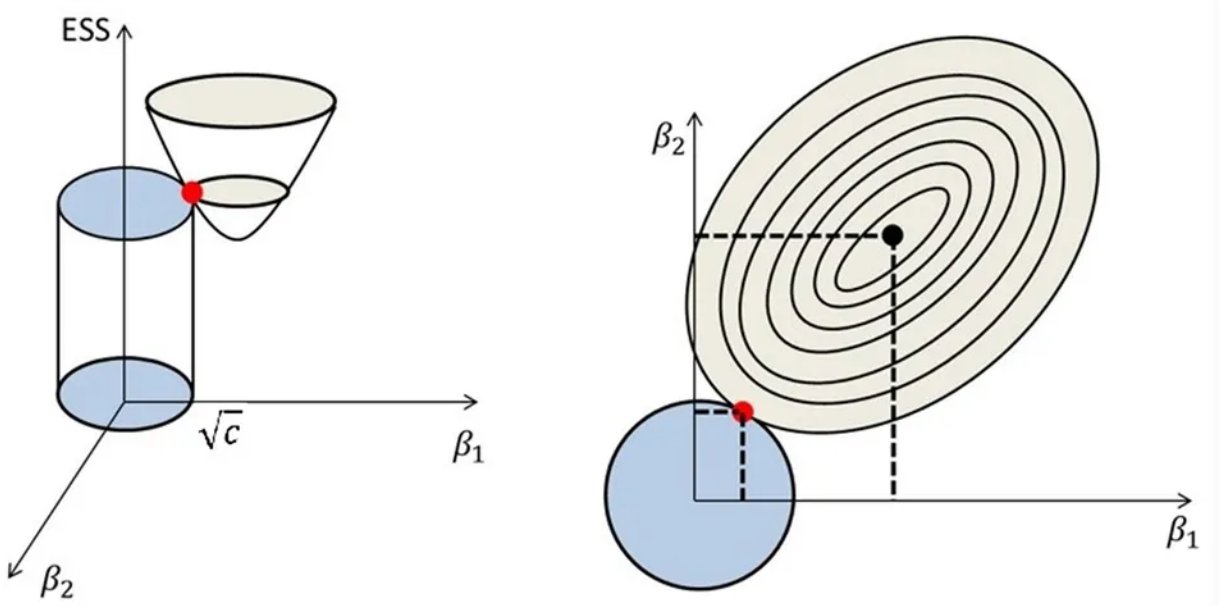


图 1： 岭回归正则项的几何意义

- 左图中椭球面代表残差平方和 $\sum_{i=1}^{n}[y_i-(\beta_0+\beta_1x_1+\beta_2x_2)]^2$

它是关于两个回归系数 β1 和 β2 的二次函数；圆柱体代表 β1^2 + β2^2 ⩽ c；

- 右图是投影到决策变量β1 和β2 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

β1^2 + β2^2 ⩽ 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：

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

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λ=0时，损失函数退化为多元线性回归模型的损失函数；
- 当$\lambda \to \infty$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 argmin_{β} 的优化问题等价于：

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

Lasso 回归的几何意义

- 以二元回归为例：

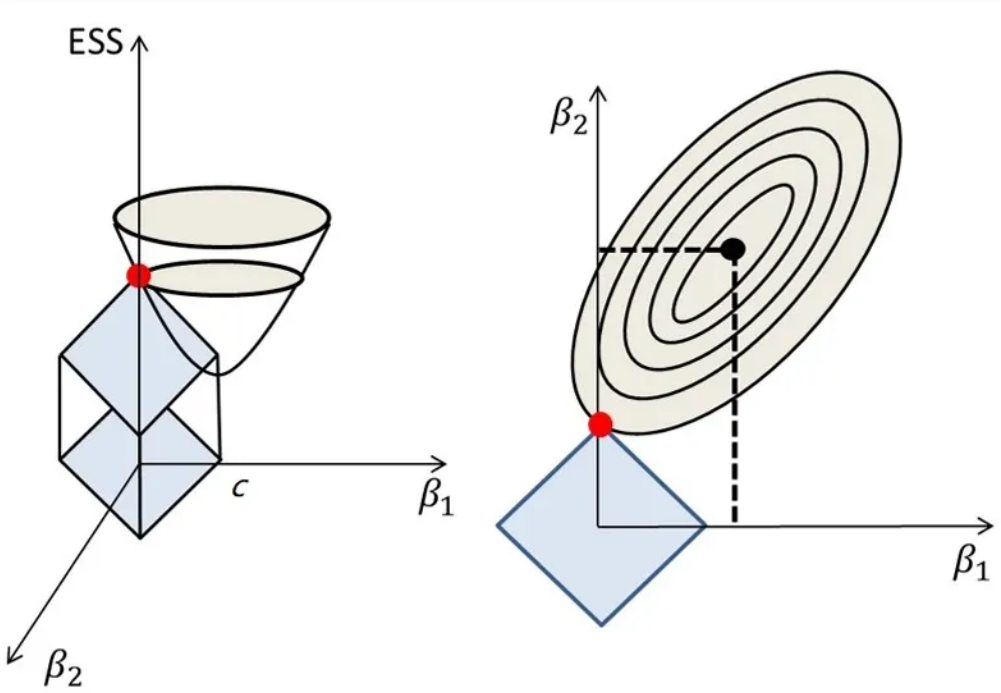


图 2: 岭回归正则项的几何意义

• 左图中椭球面代表残差平方和$\sum_{i=1}^{n}[y_i-(\beta_0+\beta_1x_1+\beta_2x_2)]^2$，

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

$|\beta_1|+|\beta_2|\leqslant c$；

• 右图是投影到决策变量$\beta1$ 和$\beta2$ 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$|\beta1|+|\beta2|\leqslant c$ 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 $\beta 2$ 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：α=0 退化为岭回归，α =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

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

用as_task_regr()函数创建回归任务：

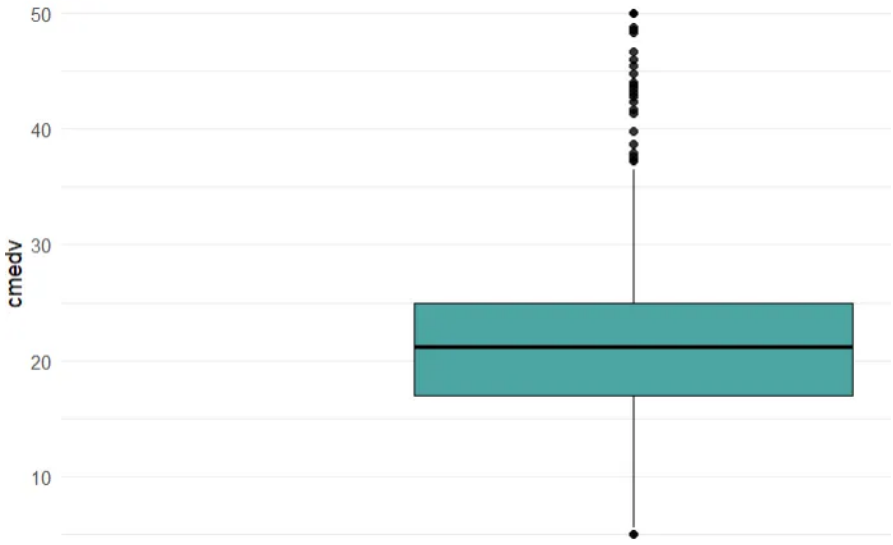
task =as_task_regr(boston,target= "cmedv")

task

```
> task
<TaskRegr:boston> (506 x 18)
* Target: cmedv
* Properties: -
* Features (17):
  - dbl (15): age, b, crim, dis, indus, lat, lon,
    lstat, nox, ptratio, rad, rm, tax, tract, zn
  - 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，所以需要做额外的数据准备：

- 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tratify=TRUE

1.岭回归预测波士顿房价

(1)选择学习器

lrn_ridge = lrn("regr.glmnet", alpha= 0) #需安装glmnet包

lrn_ridge

```
> lrn_ridge
<LearnerRegrGlmnet:regr.glmnet>: GLM with Elastic Net Regularization
* Model: -
* Parameters: alpha=0, family=gaussian
* Packages: mlr3, mlr3learners, glmnet
* Predict Types:  [response]
* Feature Types: logical, integer, numeric
* 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

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

```
 40:     type.multinomial ParamFct    NA    NA       2
 41:         upper.limits ParamUty    NA    NA     Inf
                       id    class lower upper nlevels
            default parents    value
             <list>  <list>   <list>
  1:         lambda
  2:              1                0
  3:        9.9e+35
  4:          0.999
  5: <NoDefault[0]>
  6:          1e-06
  7:          1e-08
  8:          FALSE
  9: <NoDefault[0]>
 10:            250
 11:       gaussian         gaussian
 12:          1e-05
 13:              1   relax
 14:           TRUE
 15:           TRUE
 16:          FALSE
 17: <NoDefault[0]>
 18: <NoDefault[0]>
 19: <NoDefault[0]>
 20:         100000
 21:              5
 22:            100
 23:             25
 24: <NoDefault[0]>
 25:            100
 26:
 27:          FALSE
 28: <NoDefault[0]>
 29: <NoDefault[0]>
 30:          1e-09
 31:          1e-10
 32:          FALSE
 33:           0.01
 34:           TRUE
 35:          FALSE
 36:          1e-07
 37:              0
 38: <NoDefault[0]>  family
 39: <NoDefault[0]>
 40: <NoDefault[0]>
 41: <NoDefault[0]>
            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

```R
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
```

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：

```
> at$train(task, row_ids= split$train)
INFO  [09:25:29.328] [bbotk] Starting to optimize 1 parameter(s) with '<OptimizerBatchRandomSearch>' and '<TerminatorEvals> [n_evals=10, k=0]'
INFO  [09:25:29.393] [bbotk] Evaluating 1 configuration(s)
INFO  [09:25:29.416] [mlr3] Running benchmark with 3 resampling iterations
INFO  [09:25:29.462] [mlr3] Applying learner 'regr.glmnet' on task 'boston' (iter 1/3)
INFO  [09:25:29.550] [mlr3] Applying learner 'regr.glmnet' on task 'boston' (iter 2/3)
INFO  [09:25:29.609] [mlr3] Applying learner 'regr.glmnet' on task 'boston' (iter 3/3)
INFO  [09:25:29.673] [mlr3] Finished benchmark
INFO  [09:25:29.711] [bbotk] Result of batch 1:
INFO  [09:25:29.714] [bbotk]          s regr.rmse warnings errors
INFO  [09:25:29.714] [bbotk]  0.9806368  3.937937        0      0
INFO  [09:25:29.714] [bbotk]  runtime_learners
INFO  [09:25:29.714] [bbotk]              0.15
INFO  [09:25:29.714] [bbotk]                                    uhash
INFO  [09:25:29.714] [bbotk]  19b10331-d283-4b36-99c1-dddd5a865deb
INFO  [09:25:29.724] [bbotk] Evaluating 1 configuration(s)
INFO  [09:25:29.741] [mlr3] Running benchmark with 3 resampling iterations
INFO  [09:25:29.748] [mlr3] Applying learner 'regr.glmnet' on task 'boston' (iter 1/3)
INFO  [09:25:29.807] [mlr3] Applying learner 'regr.glmnet' on task 'boston' (iter 2/3)
INFO  [09:25:29.902] [mlr3] Applying learner 'regr.glmnet' on task 'boston' (iter 3/3)
INFO  [09:25:29.956] [mlr3] Finished benchmark
INFO  [09:25:29.993] [bbotk] Result of batch 2:
INFO  [09:25:29.996] [bbotk]          s regr.rmse warnings errors runtime_learners
INFO  [09:25:29.996] [bbotk]  1.708346  4.018843        0      0
        0.16
INFO  [09:25:29.996] [bbotk]                                    uhash
INFO  [09:25:29.996] [bbotk]  42e5445a-0632-4a46-a07c-7d04c53ba404
INFO  [09:25:30.006] [bbotk] Evaluating 1 configuration(s)
INFO  [09:25:30.023] [mlr3] Running benchmark with 3 resampling iterations
INFO  [09:25:30.031] [mlr3] Applying learner 'regr.glmnet' on task 'boston' (iter 1/3)
INFO  [09:25:30.341] [mlr3] Applying learner 'regr.glmnet' on task 'boston' (iter 2/3)
```

```
INFO  [09:25:30.404] [mlr3] Applying learner 'regr.glmnet' on task 'boston' (iter 3/3)
INFO  [09:25:30.461] [mlr3] Finished benchmark
INFO  [09:25:30.499] [bbotk] Result of batch 3:
INFO  [09:25:30.503] [bbotk]         s regr.rmse warnings errors runtime_learners
INFO  [09:25:30.503] [bbotk]  0.303525  3.917136        0      0       0.1
INFO  [09:25:30.503] [bbotk]                                    uhash
INFO  [09:25:30.503] [bbotk]  09f64f86-d8c3-4596-98ea-45b2fe844123
INFO  [09:25:30.512] [bbotk] Evaluating 1 configuration(s)
INFO  [09:25:30.528] [mlr3] Running benchmark with 3 resampling iterations
INFO  [09:25:30.537] [mlr3] Applying learner 'regr.glmnet' on task 'boston' (iter 1/3)
INFO  [09:25:30.594] [mlr3] Applying learner 'regr.glmnet' on task 'boston' (iter 2/3)
INFO  [09:25:30.650] [mlr3] Applying learner 'regr.glmnet' on task 'boston' (iter 3/3)
INFO  [09:25:30.702] [mlr3] Finished benchmark
INFO  [09:25:30.740] [bbotk] Result of batch 4:
INFO  [09:25:30.744] [bbotk]        s regr.rmse warnings errors runtime_learners
INFO  [09:25:30.744] [bbotk]  1.95479  4.047632        0      0      0.13
INFO  [09:25:30.744] [bbotk]                                    uhash
INFO  [09:25:30.744] [bbotk]  035f3126-9424-4647-b7e5-5610e857db36
INFO  [09:25:30.753] [bbotk] Evaluating 1 configuration(s)
INFO  [09:25:30.770] [mlr3] Running benchmark with 3 resampling iterations
INFO  [09:25:30.777] [mlr3] Applying learner 'regr.glmnet' on task 'boston' (iter 1/3)
INFO  [09:25:30.843] [mlr3] Applying learner 'regr.glmnet' on task 'boston' (iter 2/3)
INFO  [09:25:30.897] [mlr3] Applying learner 'regr.glmnet' on task 'boston' (iter 3/3)
INFO  [09:25:30.951] [mlr3] Finished benchmark
INFO  [09:25:30.987] [bbotk] Result of batch 5:
INFO  [09:25:30.991] [bbotk]          s regr.rmse warnings errors
INFO  [09:25:30.991] [bbotk]  0.1225037  3.917136        0      0
INFO  [09:25:30.991] [bbotk]  runtime_learners
INFO  [09:25:30.991] [bbotk]              0.15
INFO  [09:25:30.991] [bbotk]                                    uhash
INFO  [09:25:30.991] [bbotk]  5b73e037-791f-4905-9f8e-c3c013d25dda
INFO  [09:25:31.000] [bbotk] Evaluating 1 configuration(s)
INFO  [09:25:31.017] [mlr3] Running benchmark with 3 resampling iterations
```

```
INFO  [09:25:31.025] [mlr3] Applying learner 'regr.glmnet' on task 'boston' (iter 1/3)
INFO  [09:25:31.095] [mlr3] Applying learner 'regr.glmnet' on task 'boston' (iter 2/3)
INFO  [09:25:31.151] [mlr3] Applying learner 'regr.glmnet' on task 'boston' (iter 3/3)
INFO  [09:25:31.205] [mlr3] Finished benchmark
INFO  [09:25:31.245] [bbotk] Result of batch 6:
INFO  [09:25:31.250] [bbotk]          s regr.rmse warnings errors
INFO  [09:25:31.250] [bbotk]  0.1776382  3.917136        0      0
INFO  [09:25:31.250] [bbotk]  runtime_learners
INFO  [09:25:31.250] [bbotk]              0.12
INFO  [09:25:31.250] [bbotk]                                   uhash
INFO  [09:25:31.250] [bbotk]  ebb85566-0b7b-4a21-9360-fad3a2be8218
INFO  [09:25:31.260] [bbotk] Evaluating 1 configuration(s)
INFO  [09:25:31.279] [mlr3] Running benchmark with 3 resampling iterations
INFO  [09:25:31.288] [mlr3] Applying learner 'regr.glmnet' on task 'boston' (iter 1/3)
INFO  [09:25:31.348] [mlr3] Applying learner 'regr.glmnet' on task 'boston' (iter 2/3)
INFO  [09:25:31.406] [mlr3] Applying learner 'regr.glmnet' on task 'boston' (iter 3/3)
INFO  [09:25:31.458] [mlr3] Finished benchmark
INFO  [09:25:31.507] [bbotk] Result of batch 7:
INFO  [09:25:31.511] [bbotk]          s regr.rmse warnings errors
INFO  [09:25:31.511] [bbotk]  0.6756525  3.917136        0      0
INFO  [09:25:31.511] [bbotk]  runtime_learners
INFO  [09:25:31.511] [bbotk]              0.15
INFO  [09:25:31.511] [bbotk]                                   uhash
INFO  [09:25:31.511] [bbotk]  24aca6a7-5769-4a65-96d9-ffdd1f54abd7
INFO  [09:25:31.519] [bbotk] Evaluating 1 configuration(s)
INFO  [09:25:31.535] [mlr3] Running benchmark with 3 resampling iterations
INFO  [09:25:31.543] [mlr3] Applying learner 'regr.glmnet' on task 'boston' (iter 1/3)
INFO  [09:25:31.600] [mlr3] Applying learner 'regr.glmnet' on task 'boston' (iter 2/3)
INFO  [09:25:31.654] [mlr3] Applying learner 'regr.glmnet' on task 'boston' (iter 3/3)
INFO  [09:25:31.706] [mlr3] Finished benchmark
INFO  [09:25:31.743] [bbotk] Result of batch 8:
INFO  [09:25:31.747] [bbotk]          s regr.rmse warnings errors
INFO  [09:25:31.747] [bbotk]  0.4752222  3.917136        0      0
INFO  [09:25:31.747] [bbotk]  runtime_learners
INFO  [09:25:31.747] [bbotk]              0.13
INFO  [09:25:31.747] [bbotk]                                   uhash
```

```
INFO  [09:25:31.747] [bbotk]  1e24c8b6-b8b4-46bb-bb7e-02b7f4aec32f
INFO  [09:25:31.756] [bbotk] Evaluating 1 configuration(s)
INFO  [09:25:31.772] [mlr3] Running benchmark with 3 resampling iterations
INFO  [09:25:31.780] [mlr3] Applying learner 'regr.glmnet' on task 'boston' (iter 1/3)
INFO  [09:25:31.844] [mlr3] Applying learner 'regr.glmnet' on task 'boston' (iter 2/3)
INFO  [09:25:31.898] [mlr3] Applying learner 'regr.glmnet' on task 'boston' (iter 3/3)
INFO  [09:25:31.951] [mlr3] Finished benchmark
INFO  [09:25:31.989] [bbotk] Result of batch 9:
INFO  [09:25:31.992] [bbotk]          s regr.rmse warnings errors
INFO  [09:25:31.992] [bbotk]  0.6376707  3.917136        0      0
INFO  [09:25:31.992] [bbotk]  runtime_learners
INFO  [09:25:31.992] [bbotk]              0.15
INFO  [09:25:31.992] [bbotk]                                    uhash
INFO  [09:25:31.992] [bbotk]  4d903261-eb23-4bf0-86f1-e59cbacad0d9
INFO  [09:25:32.002] [bbotk] Evaluating 1 configuration(s)
INFO  [09:25:32.018] [mlr3] Running benchmark with 3 resampling iterations
INFO  [09:25:32.026] [mlr3] Applying learner 'regr.glmnet' on task 'boston' (iter 1/3)
INFO  [09:25:32.082] [mlr3] Applying learner 'regr.glmnet' on task 'boston' (iter 2/3)
INFO  [09:25:32.137] [mlr3] Applying learner 'regr.glmnet' on task 'boston' (iter 3/3)
INFO  [09:25:32.199] [mlr3] Finished benchmark
INFO  [09:25:32.237] [bbotk] Result of batch 10:
INFO  [09:25:32.241] [bbotk]          s regr.rmse warnings errors
INFO  [09:25:32.241] [bbotk]  0.2934178  3.917136        0      0
INFO  [09:25:32.241] [bbotk]  runtime_learners
INFO  [09:25:32.241] [bbotk]              0.13
INFO  [09:25:32.241] [bbotk]                                    uhash
INFO  [09:25:32.241] [bbotk]  3b6eb25f-f7a4-4905-ad98-2c1a51043531
INFO  [09:25:32.265] [bbotk] Finished optimizing after 10 evaluation(s)
INFO  [09:25:32.267] [bbotk] Result:
INFO  [09:25:32.271] [bbotk]         s learner_param_vals  x_domain regr.rmse
INFO  [09:25:32.271] [bbotk]     <num>             <list>    <list>     <num>
INFO  [09:25:32.271] [bbotk]  0.303525          <list[3]> <list[1]>  3.917136
```

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
```

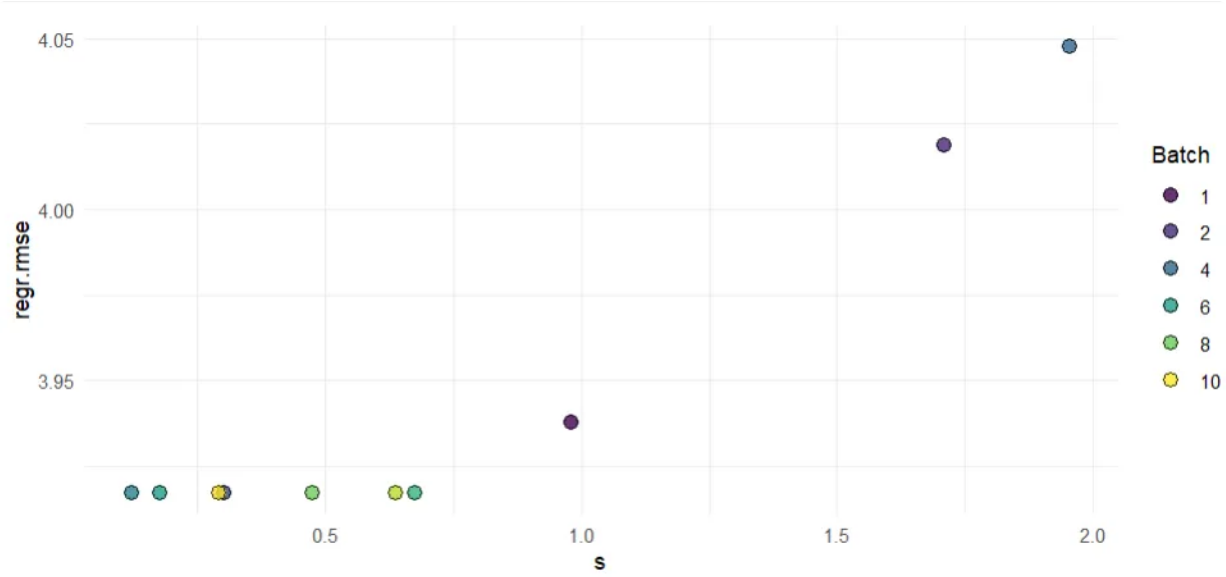
老师课件结果是下面：

查看最优超参数

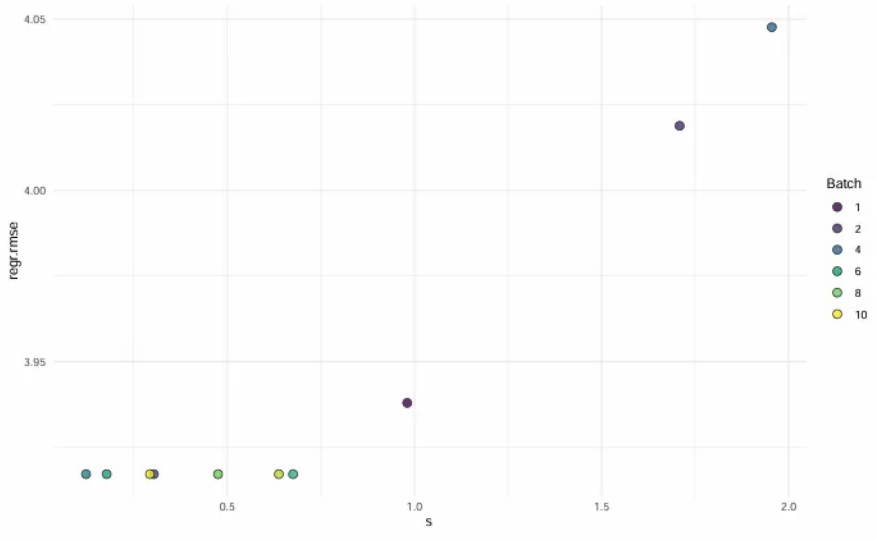
```
at$tuning_result
#>         s learner_param_vals  x_domain regr.rmse
#>     <num>             <list>    <list>      <num>
#> 1: 0.304          <list[3]> <list[1]>       3.92
```

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。

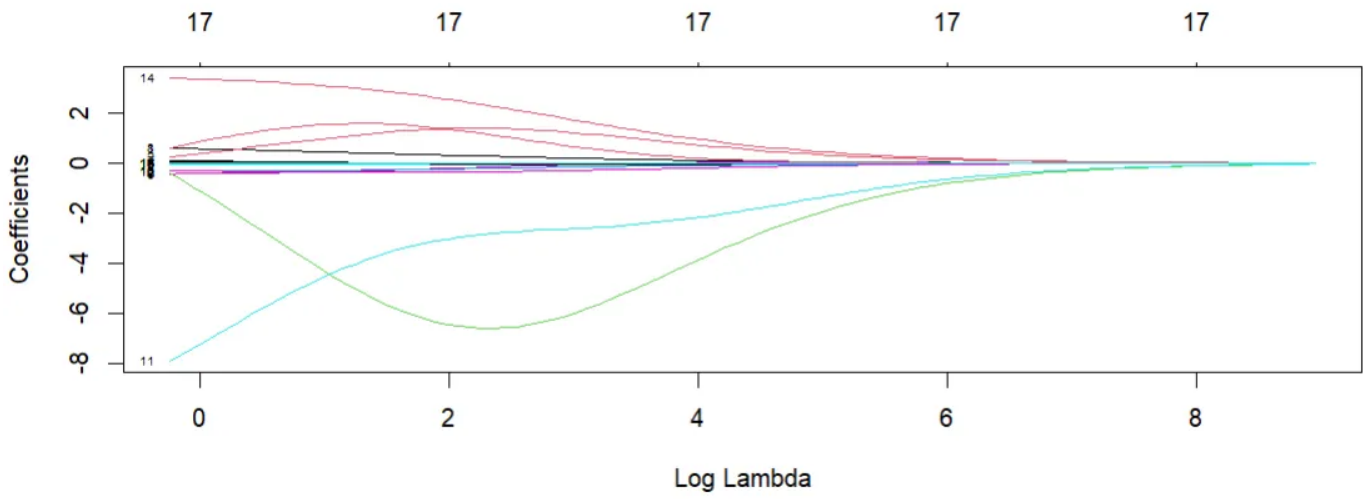
model = at$learner$model

提取岭回归模型系数，绘制岭迹图：

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

```
> coef(model, s = at$tuning_result$s)
18 x 1 sparse Matrix of class "dgCMatrix"
                        s1
(Intercept) -3.989849e+01
town         5.986355e-01
chas         2.698817e-01
age         -1.057666e-02
b            7.005990e-03
crim        -4.766126e-02
dis         -4.307300e-01
indus        1.065570e-01
lat          6.193917e-01
lon         -4.004903e-01
lstat       -3.036641e-01
nox         -7.908075e+00
ptratio     -3.197294e-01
rad          3.153456e-02
rm           3.400158e+00
tax          1.584127e-04
tract       -3.292376e-04
zn          -2.869692e-03
>
```

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



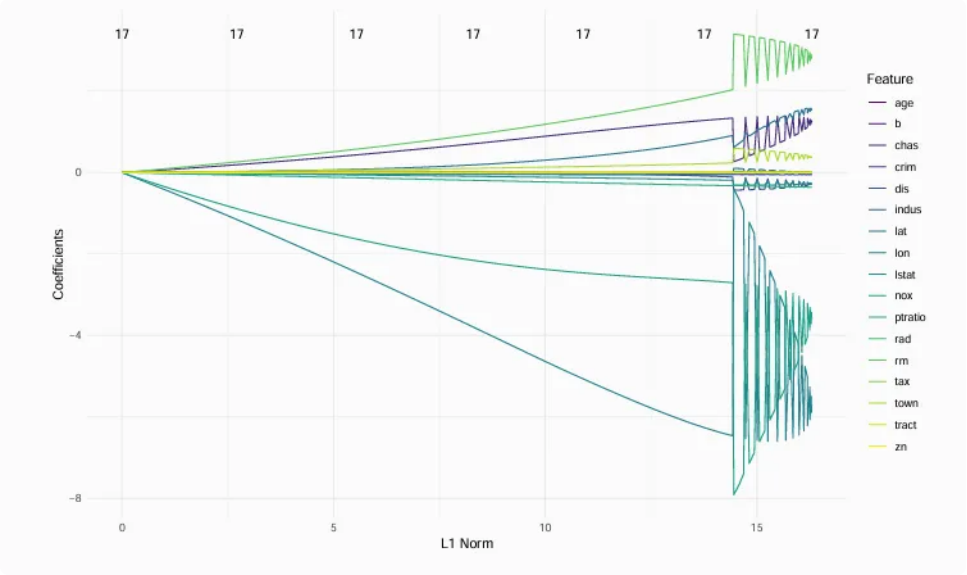
可见，随着 λ的增大，有越来越多的指标回归系数压迫到接近0。

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 loade
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

```
> pred_ridge
<PredictionRegr> for 101 observations:
    row_ids truth  response
          2  21.6 27.725306
          3  34.7 32.025778
         30  21.0 19.536576
---
        491   8.1  9.011048
        497  19.7 16.066158
        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")))

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

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

```R
> lrn_lasso
<LearnerRegrGlmnet:regr.glmnet>: GLM with Elastic Net Regularization
* Model: -
* Parameters: alpha=1, family=gaussian
* Packages: mlr3, mlr3learners, glmnet
* Predict Types:  [response]
* Feature Types: logical, integer,
  numeric
* Properties: weights
>
```

(2) 超参数调参

查看模型的所有超参数及默认值

lrn_lasso$param_set

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

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_lasso,
  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
```

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。

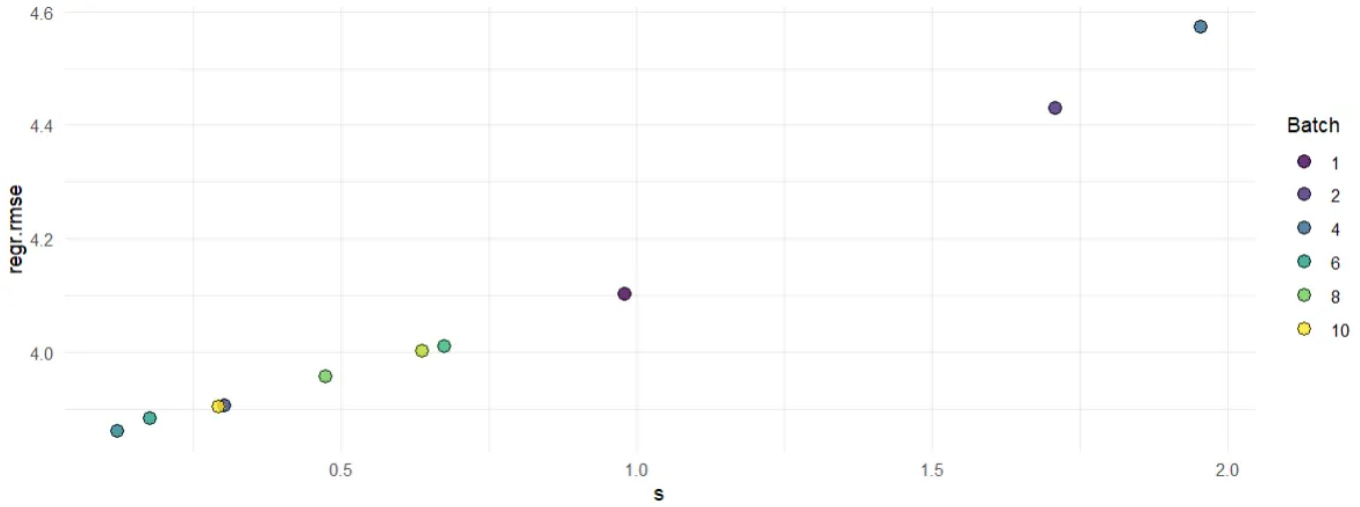
查看最优超参数

at$tuning_result

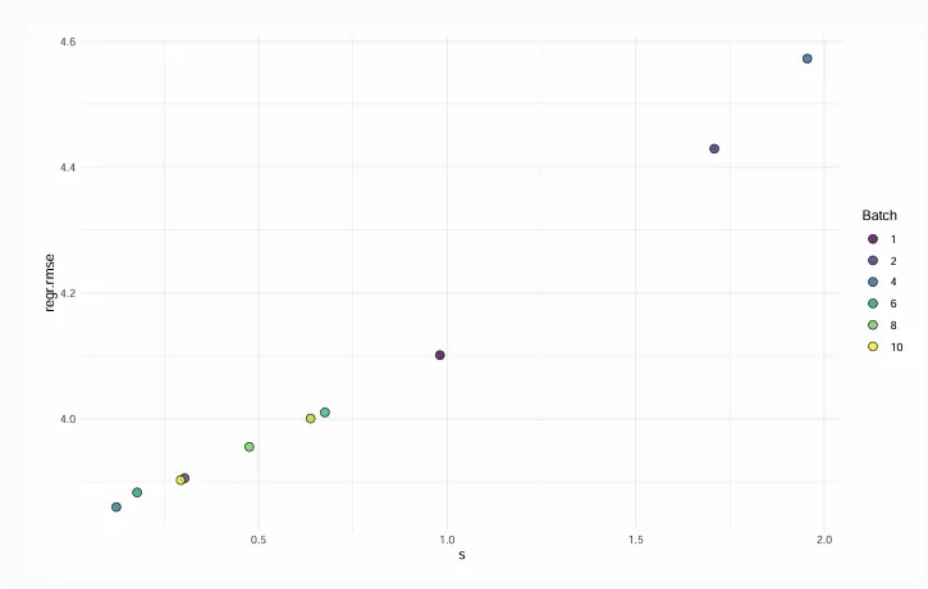
```
> #查看最优超参数
> at$tuning_result
            s learner_param_vals  x_domain regr.rmse
        <num>             <list>    <list>     <num>
1: 0.1225037          <list[3]> <list[1]>  3.860208
```

- 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

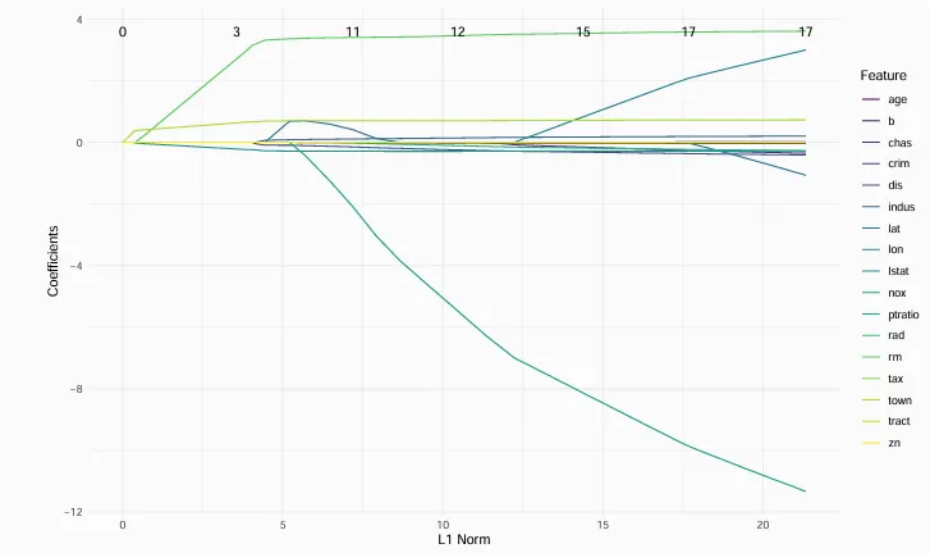
- 提取Lasso 回归模型系数：

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

```
> coef(model, s = at$tuning_result$s)
18 x 1 sparse Matrix of class "dgCMatrix"
                        s1
(Intercept)  3.8889524733
town         0.7018007401
chas         .
age          .
b            0.0058758279
crim        -0.0141305145
dis         -0.1901648290
indus        0.1193572201
lat          0.0431100663
lon          .
lstat       -0.2935041550
nox         -3.5447355812
ptratio     -0.0565460351
rad          .
rm           3.4148549528
tax          .
tract       -0.0002025079
zn           .
>
```

- 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
> pred_lasso
<PredictionRegr> for 101 observations:
    row_ids truth  response
          2  21.6 28.053602
          3  34.7 32.138046
         30  21.0 19.777358
---
        491   8.1  8.775198
        497  19.7 15.839494
        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")))

```R
> #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")))
regr.rmse  regr.rsq
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)

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

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

lrn_glmnet= lrn("regr.glmnet")#需安装glmnet包

lrn_glmnet

```
> lrn_glmnet
<LearnerRegrGlmnet:regr.glmnet>: GLM with Elastic Net Regularization
* Model: -
* Parameters: family=gaussian
* Packages: mlr3, mlr3learners, glmnet
* Predict Types:  [response]
* Feature Types: logical, integer,
  numeric
* 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

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

需要对模型中的权重alpha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

```R
search_space = ps(alpha= p_dbl(0, 1),
                  s=p_dbl(0.001, 2))

at = auto_tuner(
  tuner=tnr("random_search"),
  learner= lrn_glmnet,
  resampling= rsmp("cv", folds= 3), #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


```

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

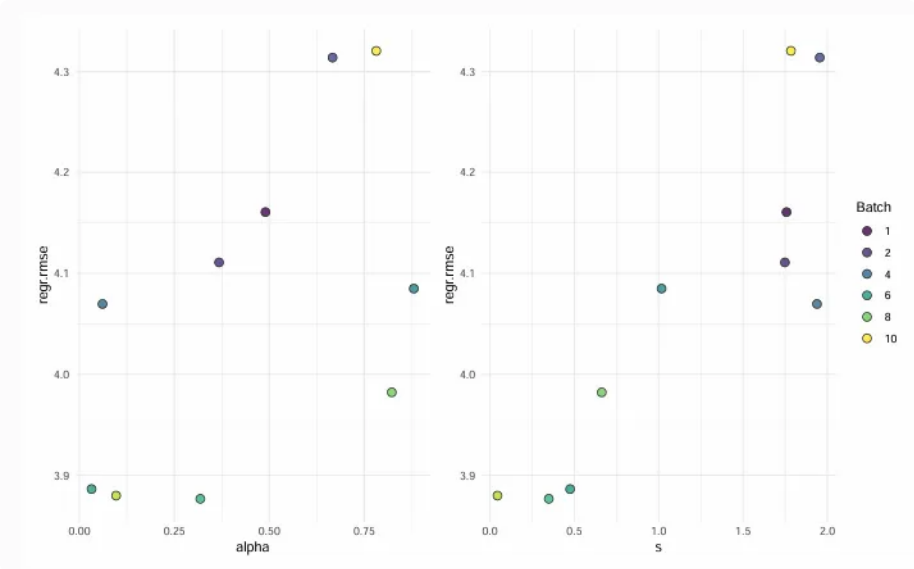
```R
> #查看最优超参数

> at$tuning_result
       alpha         s learner_param_vals
       <num>     <num>             <list>
1: 0.3184946 0.3487936          <list[3]>
     x_domain regr.rmse
       <list>     <num>
1: <list[2]>  3.876519
>
```

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)

```
> coef(model, s = at$tuning_result$s)
18 x 1 sparse Matrix of class "dgCMatrix"
                       s1
(Intercept)  7.7857179956
town         0.6647284626
chas         .
age         -0.0008238749
b            0.0059202762
crim        -0.0194763388
dis         -0.2421870297
indus        0.1000830418
lat          .
lon          .
lstat       -0.2979543835
nox         -3.9314394768
ptratio     -0.1168453668
rad          .
rm           3.3885128895
tax          .
tract       -0.0002220278
zn           .
>
```

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

```
> pred_glmnet
<PredictionRegr> for 101 observations:
    row_ids truth  response
          2  21.6 27.972836
          3  34.7 32.074555
         30  21.0 19.885501
---
        491   8.1  8.769547
        497  19.7 15.965446
        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")))

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

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

```R
#(1) 创建任务
library(tidyverse)
library(mlr3verse)

boston = read_csv("D:/24暑期-R机器学习班/data/boston_housing.csv",
                  col_types = cols(chas = "f", town = "f"))
# 读取数据，同时设置 chas 和 town 列按因子类型读取：
glimpse(boston)

#用as_task_regr()函数创建回归任务：
task =as_task_regr(boston,target= "cmedv")
task

autoplot(task) # 可视化任务

prep = po("encode", method = "treatment",
          affect_columns = selector_name("chas")) %>>%
  po("encodeimpact",
     affect_columns = selector_name("town"))

# 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

task = prep$train(task)[[1]]

set.seed(123)
# 保证结果可重现
split = partition(task, ratio = 0.8) # 默认 stratify=TRUE
#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。
#(1)选择学习器
lrn_ridge = lrn("regr.glmnet", alpha= 0) #需安装glmnet包
lrn_ridge

#查看模型的所有超参数及默认值（结果略）
lrn_ridge$param_set



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

#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


#可视化超参数变化对模型性能的影响
autoplot(at$tuning_instance)

#训练模型
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)
plot(model, xvar = "lambda", label = TRUE)


#或者直接用 autoplot() 绘图：
autoplot(at$learner, type = "ggfortify")

#在测试集上做预测：
pred_ridge = at$predict(task, row_ids = split$test)
pred_ridge


#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")))

#6)预测新数据
newdata =task$data()[1:5,-1] #作为新数据
at$predict_newdata(newdata)




#Lasso回归预测波士顿房价

#(1)选择学习器
lrn_lasso = lrn("regr.glmnet", alpha= 1) #需安装glmnet包
lrn_lasso

```

```r
#(2) 超参数调参
#查看模型的所有超参数及默认值
lrn_lasso$param_set


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

at = auto_tuner(
  tuner=tnr("random_search"),
  learner= lrn_lasso,
  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

#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$tuning_result
#可视化超参数变化对模型性能的影响
autoplot(at$tuning_instance)

#训练模型
model = at$learner$model
#提取Lasso 回归模型系数:
coef(model, s = at$tuning_result$s)


#在测试集上做预测:
pred_lasso = at$predict(task, row_ids = split$test)
pred_lasso

#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")))


#(6)预测新数据
newdata =task$data()[1:5,-1] #作为新数据
at$predict_newdata(newdata)


#弹性网回归预测波士顿房价

lrn_glmnet= lrn("regr.glmnet")#需安装glmnet包
lrn_glmnet
```

```r

lrn_glmnet$param_set

search_space = ps(alpha= p_dbl(0, 1),
                  s=p_dbl(0.001, 2))

at = auto_tuner(
  tuner=tnr("random_search"),
  learner= lrn_glmnet,
  resampling= rsmp("cv", folds= 3), #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


#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$tuning_result
# 可视化超参数变化对模型性能的影响
autoplot(at$tuning_instance)

#训练模型
model = at$learner$model
# 提取弹性网回归模型系数:
coef(model, s = at$tuning_result$s)
#可视化学习器:
autoplot(at$learner, type = "ggfortify")

#在测试集上做预测:
pred_glmnet = at$predict(task, row_ids = split$test)
pred_glmnet
#(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")))


#(6)预测新数据
newdata =task$data()[1:5,-1] #作为新数据
at$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一数据分析与挖掘. 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.