elasticnet — Elastic net for prediction and model selection

Description

elasticnet selects covariates and fits linear, logistic, probit, and Poisson models using elastic net. Results from elasticnet can be used for prediction and model selection.

elasticnet saves but does not display estimated coefficients. The postestimation commands listed in [LASSO] lasso postestimation can be used to generate predictions, report coefficients, and display measures of fit.

For an introduction to lasso, see [LASSO] Lasso intro.

Quick start

Fit a linear model for $y_1$, and select covariates from $x_1$–$x_{100}$ using cross-validation (CV)

```
elasticnet linear $y_1$ $x_1$-$x_{100}$
```

As above, but specify the grid $\alpha = 0.1, 0.2, \ldots, 1$ using a numlist

```
elasticnet linear $y_1$ $x_1$-$x_{100}$, alpha(0.1(0.1)1)
```

As above, but force $x_1$ and $x_2$ to be in the model while elasticnet selects from $x_3$–$x_{100}$

```
elasticnet linear $y_1$ ($x_1$ $x_2$) $x_3$-$x_{100}$, alpha(0.1(0.1)1)
```

Fit a logistic model for binary outcome $y_2$ with grid $\alpha = 0.7, 0.8, 0.9, 1$

```
elasticnet logit $y_2$ $x_1$-$x_{100}$, alpha(0.7 0.8 0.9 1)
```

As above, and set a random-number seed for reproducibility

```
elasticnet logit $y_2$ $x_1$-$x_{100}$, alpha(0.7 0.8 0.9 1) rseed(1234)
```

As above, but fit a probit model

```
elasticnet probit $y_2$ $x_1$-$x_{100}$, alpha(0.7 0.8 0.9 1) rseed(1234)
```

Fit a Poisson model for count outcome $y_3$ with exposure time

```
elasticnet poisson $y_3$ $x_1$-$x_{100}$, alpha(0.1(0.1)1) exposure(time)
```

Calculate the CV function beyond the CV minimum to get the full coefficient paths, knots, etc.

```
elasticnet linear $y_2$ $x_1$-$x_{100}$, alpha(0.1(0.1)1) selection(cv, alllambdas)
```

Turn off the early stopping rule, and iterate over $\lambda$’s until a minimum is found or until the end of the $\lambda$ grid is reached

```
elasticnet linear $y_2$ $x_1$-$x_{100}$, alpha(0.1(0.1)1) stop(0)
```
Menu

Statistics > Lasso > Elastic net

Syntax

```
elasticnet model depvar [(alwaysvars)] othervars [if] [in] [weight] [, options]
```

`model` is one of `linear`, `logit`, `probit`, or `poisson`.

`alwaysvars` are variables that are always included in the model.

`othervars` are variables that `elasticnet` will choose to include in or exclude from the model.

<table>
<thead>
<tr>
<th>options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Model</strong></td>
<td></td>
</tr>
<tr>
<td><code>noconstant</code></td>
<td>suppress constant term</td>
</tr>
<tr>
<td><code>selection(cv [ , cv_opts ] )</code></td>
<td>select mixing parameter $\alpha^<em>$ and lasso penalty parameter $\lambda^</em>$ using CV</td>
</tr>
<tr>
<td><code>selection(none)</code></td>
<td>do not select $\alpha^<em>$ or $\lambda^</em>$</td>
</tr>
<tr>
<td><code>offset(varname_o)</code></td>
<td>include <code>varname_o</code> in model with coefficient constrained to 1</td>
</tr>
<tr>
<td><code>exposure(varname_e)</code></td>
<td>include $\ln(varname_e)$ in model with coefficient constrained to 1 (poisson models only)</td>
</tr>
<tr>
<td><strong>Optimization</strong></td>
<td></td>
</tr>
<tr>
<td><code>[no]log</code></td>
<td>display or suppress an iteration log</td>
</tr>
<tr>
<td><code>rseed(#)</code></td>
<td>set random-number seed</td>
</tr>
<tr>
<td>`alphas(numlist</td>
<td>matname)`</td>
</tr>
<tr>
<td><code>grid(#_g [, ratio(#) min(#)] )</code></td>
<td>specify the set of possible $\lambda$'s using a logarithmic grid with $#_g$ grid points</td>
</tr>
<tr>
<td><code>crossgrid(augmented)</code></td>
<td>augment the $\lambda$ grids for each $\alpha$ as necessary to produce a single $\lambda$ grid; the default</td>
</tr>
<tr>
<td><code>crossgrid(union)</code></td>
<td>use the union of the $\lambda$ grids for each $\alpha$ to produce a single $\lambda$ grid</td>
</tr>
<tr>
<td><code>crossgrid(different)</code></td>
<td>use different $\lambda$ grids for each $\alpha$</td>
</tr>
<tr>
<td><code>stop(#)</code></td>
<td>tolerance for stopping the iteration over the $\lambda$ grid early</td>
</tr>
<tr>
<td><code>cvtolerance(#)</code></td>
<td>tolerance for identification of the CV function minimum</td>
</tr>
<tr>
<td><code>tolerance(#)</code></td>
<td>convergence tolerance for coefficients based on their values</td>
</tr>
<tr>
<td><code>dtolerance(#)</code></td>
<td>convergence tolerance for coefficients based on deviance</td>
</tr>
<tr>
<td><code>penaltywt(matname)</code></td>
<td>programmer’s option for specifying a vector of weights for the coefficients in the penalty term</td>
</tr>
</tbody>
</table>
### elasticnet — Elastic net for prediction and model selection

<table>
<thead>
<tr>
<th>cv_opts</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>folds(#)</td>
<td>use # folds for CV</td>
</tr>
<tr>
<td>alllambdas</td>
<td>fit models for all λ’s in the grid or until the stop(#) tolerance is reached; by default, the CV function is calculated sequentially by λ, and estimation stops when a minimum is identified</td>
</tr>
<tr>
<td>serule</td>
<td>use the one-standard-error rule to select λ*</td>
</tr>
<tr>
<td>stopok</td>
<td>when, for a value of α, the CV function does not have an identified minimum and the stop(#) stopping criterion for λ was reached at λstop, allow λstop to be included in an (α, λ) pair that can potentially be selected as (α*, λ*); the default strict</td>
</tr>
<tr>
<td>gridminok</td>
<td>when, for a value of α, the CV function does not have an identified minimum and the stop(#) stopping criterion for λ was not reached, allow the minimum of the λ grid, λgmin, to be included in an (α, λ) pair that can potentially be selected as (α*, λ*); this option is rarely used</td>
</tr>
</tbody>
</table>

alwaysvars and othervars may contain factor variables; see [U] 11.4.3 Factor variables.

Default weights are not allowed. iweights are allowed with both selection(cv) and selection(none). fweights are allowed when selection(none) is specified. See [U] 11.1.6 weight.

penaltywt(matname) does not appear in the dialog box.

See [U] 20 Estimation and postestimation commands for more capabilities of estimation commands.

#### Options

See [LASSO] lasso fitting for an overview of the lasso estimation procedure and a detailed description of how to set options to control it.

noconstant omits the constant term. Note, however, when there are factor variables among the othervars, elasticnet can potentially create the equivalent of the constant term by including all levels of a factor variable. This option is likely best used only when all the othervars are continuous variables and there is a conceptual reason why there should be no constant term.

selection(cv) and selection(none) specify the selection method used to select λ*. selection(cv [ , cv_opts]) is the default. It selects the (α*, λ*) that give the minimum of the CV function.

folds(#) specifies that CV with # folds be done. The default is folds(10).

alllambdas specifies that, for each α, models be fit for all λ’s in the grid or until the stop(#) tolerance is reached. By default, models are calculated sequentially from largest to smallest λ, and the CV function is calculated after each model is fit. If a minimum of the CV function is found, the computation ends at that point without evaluating additional smaller λ’s.

alllambdas computes models for these additional smaller λ’s. Because computation time is greater for smaller λ, specifying alllambdas may increase computation time manyfold. Specifying alllambdas is typically done only when a full plot of the CV function is wanted for assurance that a true minimum has been found. Regardless of whether alllambdas is specified, the selected (α*, λ*) will be the same.
elasticnet — Elastic net for prediction and model selection

serule selects $\lambda^*$ based on the “one-standard-error rule” recommended by Hastie, Tibshirani, and Wainwright (2015, 13–14) instead of the $\lambda$ that minimizes the CV function. The one-standard-error rule selects, for each $\alpha$, the largest $\lambda$ for which the CV function is within a standard error of the minimum of the CV function. Then, from among these ($\alpha, \lambda$) pairs, the one with the smallest value of the CV function is selected.

stopok, strict, and gridminok specify what to do when, for a value of $\alpha$, the CV function does not have an identified minimum at any value of $\lambda$ in the grid. A minimum is identified at $\lambda_{cv_{\min}}$ when the CV function at both larger and smaller adjacent $\lambda$’s is greater than it is at $\lambda_{cv_{\min}}$. When the CV function for a value of $\alpha$ has an identified minimum, these options all do the same thing: ($\alpha, \lambda_{cv_{\min}}$) becomes one of the ($\alpha, \lambda$) pairs that potentially can be selected as the smallest value of the CV function. In some cases, however, the CV function declines monotonically as $\lambda$ gets smaller and never rises to identify a minimum. When the CV function does not have an identified minimum, stopok and gridminok make alternative picks for $\lambda$ in the ($\alpha, \lambda$) pairs that will be assessed for the smallest value of the CV function. The option strict makes no alternative pick for $\lambda$. When stopok, strict, or gridminok is not specified, the default is stopok. With each of these options, estimation results are always left in place, and alternative ($\alpha, \lambda$) pairs can be selected and evaluated.

stopok specifies that, for a value of $\alpha$, when the CV function does not have an identified minimum and the stop(#) stopping tolerance for $\lambda$ was reached at $\lambda_{\text{stop}}$, the pair ($\alpha, \lambda_{\text{stop}}$) is picked as one of the pairs that potentially can be selected as the smallest value of the CV function. $\lambda_{\text{stop}}$ is the smallest $\lambda$ for which coefficients are estimated, and it is assumed that $\lambda_{\text{stop}}$ has a CV function value close to the true minimum for that value of $\alpha$. When no minimum is identified for a value of $\alpha$ and the stop(#) criterion is not met, an error is issued.

strict requires the CV function to have an identified minimum for each value of $\alpha$, and if not, an error is issued.

gridminok is a rarely used option that specifies that, for a value of $\alpha$, when the CV function has no identified minimum and the stop(#) stopping criterion was not met, $\lambda_{g_{\min}}$, the minimum of the $\lambda$ grid, is picked as part of a pair ($\alpha, \lambda_{g_{\min}}$) that potentially can be selected as the smallest value of the CV function.

The gridminok criterion is looser than the default stopok, which is looser than strict. With strict, the selected ($\alpha^*, \lambda^*$) pair is the minimum of the CV function chosen from the ($\alpha, \lambda_{cv_{\min}}$) pairs, where all $\lambda$’s under consideration are identified minimums. With stopok, the set of ($\alpha, \lambda$) pairs under consideration for the minimum of the CV function include identified minimums, $\lambda_{cv_{\min}}$, or values, $\lambda_{\text{stop}}$, that met the stopping criterion. With gridminok, the set of ($\alpha, \lambda$) pairs under consideration for the minimum of the CV function potentially include $\lambda_{cv_{\min}}$, $\lambda_{\text{stop}}$, or $\lambda_{g_{\min}}$.

selection(none) specifies that no ($\alpha^*, \lambda^*$) pair be selected. In this case, the elastic net is estimated for a grid of values for $\lambda$ for each $\alpha$, but no attempt is made to determine which ($\alpha, \lambda$) pair is best. The postestimation command lassoknots can be run to view a table of $\lambda$’s that define the knots (that is, the distinct sets of nonzero coefficients) for each $\alpha$. The lassoselect command can then be used to select an ($\alpha^*, \lambda^*$) pair, and lassogof can be run to evaluate the prediction performance of the selected pair.

When selection(none) is specified, the CV function is not computed. If you want to view the knot table with values of the CV function shown and then select ($\alpha^*, \lambda^*$), you must specify selection(cv). There are no suboptions for selection(none).

offset(varname$_{o}$) specifies that varname$_{o}$ be included in the model with its coefficient constrained to be 1.
exposure(varname) can be specified only for poisson models. It specifies that ln(varname) be included in the model with its coefficient constrained to be 1.

Optimization

[no]log displays or suppresses a log showing the progress of the estimation.

rseed(#) sets the random-number seed. This option can be used to reproduce results for selection(cv). (selection(none) does not use random numbers.) rseed(#) is equivalent to typing set seed # prior to running elasticnet. See [R] set seed.

alphas(numlist | matname) specifies either a numlist or a matrix containing the grid of values for \( \alpha \). The default is alphas(0.5 0.75 1). Specifying a small, nonzero value of \( \alpha \) for one of the values of alphas() will result in lengthy computation time because the optimization algorithm for a penalty that is mostly ridge regression with a little lasso mixed in is inherently inefficient. Pure ridge regression (\( \alpha = 0 \)), however, is computationally efficient.

grid(#_g [, ratio(#) min(#)]) specifies the set of possible \( \lambda \)'s using a logarithmic grid with \#_g grid points.

\#_g is the number of grid points for \( \lambda \). The default is \#_g = 100. The grid is logarithmic with the \( i \)th grid point (\( i = 1, \ldots, n = \#_g \)) given by \( \ln \lambda_i = [(i - 1)/(n - 1)] \ln r + \ln \lambda_{g_{\text{max}}} \), where \( \lambda_{g_{\text{max}}} = \lambda_1 \) is the maximum, \( \lambda_{g_{\text{min}}} = \lambda_n = \text{min}(#) \) is the minimum, and \( r = \lambda_{g_{\text{min}}}/\lambda_{g_{\text{max}}} = \text{ratio}(#) \) is the ratio of the minimum to the maximum.

ratio(#) specifies \( \lambda_{g_{\text{min}}}/\lambda_{g_{\text{max}}} \). The maximum of the grid, \( \lambda_{g_{\text{max}}} \), is set to the smallest \( \lambda \) for which all the coefficients in the lasso are estimated to be zero (except the coefficients of the alwaysvars). \( \lambda_{g_{\text{min}}} \) is then set based on ratio(#). When \( p < N \), where \( p \) is the total number of othervars and alwaysvars (not including the constant term) and \( N \) is the number of observations, the default value of ratio(#) is 1e−4. When \( p \geq N \), the default is 1e−2.

min(#) sets \( \lambda_{g_{\text{min}}} \). By default, \( \lambda_{g_{\text{min}}} \) is based on ratio(#) and \( \lambda_{g_{\text{max}}} \), which is computed from the data.

crossgrid(augmented), crossgrid(union), and crossgrid(different) specify the type of two-dimensional grid used for (\( \alpha, \lambda \)). crossgrid(augmented) and crossgrid(union) produce a grid that is the product of two one-dimensional grids. That is, the \( \lambda \) grid is the same for every value of \( \alpha \). crossgrid(different) uses different \( \lambda \) grids for different values of \( \alpha \).

crossgrid(augmented), the default grid, is formed by an augmentation algorithm. First, a suitable \( \lambda \) grid for each \( \alpha \) is computed. Then, nonoverlapping segments of these grids are formed and combined into a single \( \lambda \) grid.

crossgrid(union) specifies that the union of \( \lambda \) grids across each value of \( \alpha \) be used. That is, a \( \lambda \) grid for each \( \alpha \) is computed, and then they are combined by simply putting all the \( \lambda \) values into one grid that is used for each \( \alpha \). This produces a fine grid that can cause the computation to take a long time without significant gain in most cases.

crossgrid(different) specifies that different \( \lambda \) grids be used for each value of \( \alpha \). This option is rarely used. Using different \( \lambda \) grids for different values of \( \alpha \) complicates the interpretation of the CV selection method. When the \( \lambda \) grid is not the same for every value of \( \alpha \), comparisons are based on parameter intervals that are not on the same scale.

stop(#) specifies a tolerance that is the stopping criterion for the \( \lambda \) iterations. The default is 1e−5. Estimation starts with the maximum grid value, \( \lambda_{g_{\text{max}}} \), and iterates toward the minimum grid value, \( \lambda_{g_{\text{min}}} \). When the relative difference in the deviance produced by two adjacent \( \lambda \) grid values is less than stop(#), the iteration stops and no smaller \( \lambda \)'s are evaluated. The value of \( \lambda \) that meets this tolerance is denoted by \( \lambda_{\text{stop}} \). Typically, this stopping criterion is met before the iteration reaches \( \lambda_{g_{\text{min}}} \).
Setting `stop(#)` to a larger value means that iterations are stopped earlier at a larger $\lambda_{\text{stop}}$. To produce coefficient estimates for all values of the $\lambda$ grid, you can specify `stop(0)`. Note, however, that computations for small $\lambda$’s can be extremely time consuming. In terms of time, when you use `selection(cv)`, the optimal value of `stop(#)` is the largest value that allows estimates for just enough $\lambda$’s to be computed to identify the minimum of the CV function. When setting `stop(#)` to larger values, be aware of the consequences of the default $\lambda^*$ selection procedure given by the default stopok. You may want to override the stopok behavior by using strict.

cvtolerance(#) is a rarely used option that changes the tolerance for identifying the minimum CV function. For linear models, a minimum is declared identified when the CV function rises above a nominal minimum for at least three smaller $\lambda$’s with a relative difference in the CV function greater than #. For nonlinear models, at least five smaller $\lambda$’s are required. The default is $1e^{-3}$. Setting # to a bigger value makes a stricter criterion for identifying a minimum and brings more assurance that a declared minimum is a true minimum, but it also means that models may need to be fit for additional smaller $\lambda$, and this can be time consuming. See Methods and formulas for [LASSO] lasso for more information about this tolerance and the other tolerances.

tolerance(#) is a rarely used option that specifies the convergence tolerance for the coefficients. Convergence is achieved when the relative change in each coefficient is less than this tolerance. The default is `tolerance(1e-7)`.

dtolerance(#) is a rarely used option that changes the convergence criterion for the coefficients. When dtolerance(#) is specified, the convergence criterion is based on the change in deviance instead of the change in the values of coefficient estimates. Convergence is declared when the relative change in the deviance is less than #. More accurate coefficient estimates are typically achieved by not specifying this option and instead using the default `tolerance(1e-7)` criterion or specifying a smaller value for tolerance(#).

The following option is available with elasticnet but is not shown in the dialog box:

penaltywt(matname) is a programmer’s option for specifying a vector of weights for the coefficients in the penalty term. The contribution of each coefficient to the lasso penalty term is multiplied by its corresponding weight. Weights must be nonnegative. By default, each coefficient’s penalty weight is 1.

Remarks and examples

Elastic net, originally proposed by Zou and Hastie (2005), extends lasso to have a penalty term that is a mixture of the absolute-value penalty used by lasso and the squared penalty used by ridge regression. Coefficient estimates from elastic net are more robust to the presence of highly correlated covariates than are lasso solutions.

For the linear model, the penalized objective function for elastic net is

$$Q = \frac{1}{2N} \sum_{i=1}^{N} (y_i - \beta_0 - x_i \beta^T)^2 + \lambda \sum_{j=1}^{p} \left( \frac{1 - \alpha}{2} \beta_j^2 + \alpha |\beta_j| \right)$$

where $\beta$ is the $p$-dimensional vector of coefficients on covariates $x$. The estimated $\beta$ are those that minimize $Q$ for given values of $\alpha$ and $\lambda$.

As with lasso, $p$ can be greater than the sample size $N$. When $\alpha = 1$, elastic net reduces to lasso. When $\alpha = 0$, elastic net reduces to ridge regression.
When $\alpha > 0$, elastic net, like lasso, produces sparse solutions in which many of the coefficient estimates are exactly zero. When $\alpha = 0$, that is, ridge regression, all coefficients are nonzero, although typically many are small.

Ridge regression has long been used as a method to keep highly collinear variables in a regression model used for prediction. The ordinary least-squares (OLS) estimator becomes increasingly unstable as the correlation among the covariates grows. OLS produces wild coefficient estimates on highly correlated covariates that cancel each other out in terms of fit. The ridge regression penalty removes this instability and produces point estimates that can be used for prediction in this case.

None of the ridge regression estimates are exactly zero because the squared penalty induces a smooth tradeoff around 0 instead of the kinked-corner tradeoff induced by lasso. By mixing the two penalties, elastic net retains the sparse-solution property of lasso, but it is less variable than the lasso in the presence of highly collinear variables. The coefficient paths of elastic-net solutions are also smoother over $\lambda$ than are lasso solutions because of the added ridge-regression component.

To fit a model with elasticnet, you specify a set of candidate $\alpha$’s and a grid of $\lambda$ values. CV is performed on the combined set of $(\alpha, \lambda)$ values, and the $(\alpha^*, \lambda^*)$ pair that minimizes the value of the CV function is selected.

This procedure follows the convention of Hastie, Tibshirani, and Wainwright (2015), which is to specify a few values for $\alpha$ and a finer grid for $\lambda$. The idea is that only a few points in the space between ridge regression and lasso are worth reviewing, but a finer grid over $\lambda$ is needed to trace out the paths of which coefficients are not zero.

The default candidate values of $\alpha$ are 0.5, 0.75, and 1. Typically, you would use the default first and then set $\alpha$ using the alpha(numlist) option to get lower and upper bounds on $\alpha^*$. Models for small, nonzero values of $\alpha$ take more time to estimate than $\alpha = 0$ and larger values of $\alpha$. This is because the algorithm for fitting a model that is mostly ridge regression with a little lasso mixed in is inherently inefficient. Pure ridge or mostly lasso models are faster.

The $\lambda$ grid is set automatically, and the default settings are typically sufficient to determine $\lambda^*$. The default grid can be changed using the grid() option. See [LASSO] lasso fitting for a detailed description of the CV selection process and how to set options to control it.

**Example 1: Elastic net and data that are not highly correlated**

We will fit an elastic-net model using the example dataset from [LASSO] lasso examples. It has stored variable lists created by vl. See [D] vl for a complete description of the vl system and how to use it to manage large variable lists.
After we load the dataset, we type `vl rebuild` to make the saved variable lists active again.

```
. use https://www.stata-press.com/data/r16/fakesurvey_vl
(Fictitious Survey Data with vl)
. vl rebuild
Rebuilding vl macros ...
```

<table>
<thead>
<tr>
<th>Macro</th>
<th># Vars</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>System</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$vldummy</td>
<td>98</td>
<td>0/1 variables</td>
</tr>
<tr>
<td>$vlcategorical</td>
<td>16</td>
<td>categorical variables</td>
</tr>
<tr>
<td>$vlcontinuous</td>
<td>29</td>
<td>continuous variables</td>
</tr>
<tr>
<td>$vluncertain</td>
<td>16</td>
<td>perhaps continuous, perhaps categorical variables</td>
</tr>
<tr>
<td>$vlother</td>
<td>12</td>
<td>all missing or constant variables</td>
</tr>
<tr>
<td>User</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$demographics</td>
<td>4</td>
<td>variables</td>
</tr>
<tr>
<td>$factors</td>
<td>110</td>
<td>variables</td>
</tr>
<tr>
<td>$idemographics</td>
<td></td>
<td>factor-variable list</td>
</tr>
<tr>
<td>$ifactors</td>
<td></td>
<td>factor-variable list</td>
</tr>
</tbody>
</table>

We have four user-defined variable lists, `demographics`, `factors`, `idemographics`, and `ifactors`. The variable lists `idemographics` and `ifactors` contain factor-variable versions of the categorical variables in `demographics` and `factors`. That is, a variable `q3` in `demographics` is `i.q3` in `idemographics`. See the examples in `[LASSO] lasso examples` to see how we created these variable lists.

We are going to use `idemographics` and `ifactors` along with the system-defined variable list `vlcontinuous` as arguments to `elasticnet`. Together they contain the potential variables we want to specify. Variable lists are actually global macros, and when we use them as arguments in commands, we put a `$` in front of them.

We also set the random-number seed using the `rseed()` option so we can reproduce our results.

```
. elasticnet linear q104 $idemographics $ifactors $vlcontinuous, rseed(1234)
alpha 1 of 3: alpha = 1
10-fold cross-validation with 109 lambdas ...
Grid value 1: lambda = 1.818102 no. of nonzero coef. = 0
Folds: 1...5....10 CVF = 18.34476
(output omitted)
Grid value 37: lambda = .0737359 no. of nonzero coef. = 80
Folds: 1...5....10 CVF = 11.92887
... cross-validation complete ... minimum found
alpha 2 of 3: alpha = 0.75
10-fold cross-validation with 109 lambdas ...
Grid value 1: lambda = 1.818102 no. of nonzero coef. = 0
Folds: 1...5....10 CVF = 18.34476
(output omitted)
```
elasticnet — Elastic net for prediction and model selection

Grid value 34: lambda = .0974746 no. of nonzero coef. = 126
Folds: 1...5....10 CVF = 11.95437
... cross-validation complete ... minimum found
alpha 3 of 3: alpha = 0.5
10-fold cross-validation with 109 lambdas ...
Grid value 1: lambda = 1.818102 no. of nonzero coef. = 0
Folds: 1...5....10 CVF = 18.33643
(output omitted)
Grid value 31: lambda = .1288556 no. of nonzero coef. = 139
Folds: 1...5....10 CVF = 12.0549
... cross-validation complete ... minimum found
Elastic net linear model
No. of obs = 914
No. of covariates = 277
Selection: Cross-validation
No. of CV folds = 10

<table>
<thead>
<tr>
<th>alpha ID</th>
<th>Description</th>
<th>lambda</th>
<th>no. of nonzero coef.</th>
<th>R-squared</th>
<th>Out-of-sample prediction error</th>
<th>CV mean prediction error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.000</td>
<td>first lambda 1.818102</td>
<td>0</td>
<td>0.0016</td>
<td>18.34476</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>first lambda</td>
<td>1.818102</td>
<td>0</td>
<td>0.0016</td>
<td>18.34476</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>lambda before .1174085</td>
<td>58</td>
<td>0.3543</td>
<td>11.82553</td>
<td></td>
<td></td>
</tr>
<tr>
<td>* 33</td>
<td>selected lambda .1069782</td>
<td>64</td>
<td>0.3547</td>
<td>11.81814</td>
<td></td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>lambda after .0974746</td>
<td>66</td>
<td>0.3545</td>
<td>11.8222</td>
<td></td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>last lambda   .0737359</td>
<td>80</td>
<td>0.3487</td>
<td>11.92887</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.750</td>
<td>first lambda 1.818102</td>
<td>0</td>
<td>0.0016</td>
<td>18.34476</td>
<td></td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>first lambda</td>
<td>1.818102</td>
<td>0</td>
<td>0.0016</td>
<td>18.34476</td>
<td></td>
</tr>
<tr>
<td>71</td>
<td>last lambda   .0974746</td>
<td>126</td>
<td>0.3473</td>
<td>11.95437</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.500</td>
<td>first lambda 1.818102</td>
<td>0</td>
<td>0.0012</td>
<td>18.33643</td>
<td></td>
<td></td>
</tr>
<tr>
<td>72</td>
<td>first lambda</td>
<td>1.818102</td>
<td>0</td>
<td>0.0012</td>
<td>18.33643</td>
<td></td>
</tr>
<tr>
<td>102</td>
<td>last lambda   .1288556</td>
<td>139</td>
<td>0.3418</td>
<td>12.0549</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* alpha and lambda selected by cross-validation.

CV selected $\alpha^* = 1$, that is, the results from an ordinary lasso.

All models we fit using elastic net on these data selected $\alpha^* = 1$. The data are not correlated enough to need elastic net.

Example 2: Elastic net and data that are highly correlated

The dataset in example 1, fakesurvey_vl, contained data we created in a simulation. We did our simulation again setting the correlation parameters to much higher values, up to $\rho = 0.95$, and we created two groups of highly correlated variables, with correlations between variables from different groups much lower. We saved these data in a new dataset named fakesurvey2_vl. Elastic net was proposed not just for highly correlated variables but especially for groups of highly correlated variables.
We load the new dataset and run \texttt{vl rebuild}.

\begin{verbatim}
  . use https://www.stata-press.com/data/r16/fakesurvey2_vl, clear
  (Fictitious Survey Data with vl)
  . vl rebuild
  Rebuilding vl macros ...  
\end{verbatim}

(output omitted)

In anticipation of elastic net showing interesting results this time, we randomly split our data into two samples of equal sizes. One we will fit models on, and the other we will use to test their predictions. We use \texttt{splitsample} to generate a variable indicating the samples.

\begin{verbatim}
  . set seed 1234
  . splitsample, generate(sample) nsplit(2)
  . label define svalues 1 "Training" 2 "Testing"
  . label values sample svalues
\end{verbatim}

We fit an elastic-net model using the default \(\alpha\)'s.

\begin{verbatim}
  . elasticnet linear q104 $idemographics $ifactors $vlcontinuous
     > if sample == 1, rseed(1234)
  alpha 1 of 3: alpha = 1
  (output omitted)
  10-fold cross-validation with 109 lambdas ...  
  Grid value 1:   lambda = 6.323778   no. of nonzero coef. = 0 
  Folds: 1...5....10   CVF = 26.82324 
  (output omitted)
  Grid value 42:   lambda = .161071   no. of nonzero coef. = 29 
  Folds: 1...5....10   CVF = 15.12964 
  ... cross-validation complete ... minimum found 
  alpha 2 of 3: alpha = 0.75 
  (output omitted)
  10-fold cross-validation with 109 lambdas ...  
  Grid value 1:   lambda = 6.323778   no. of nonzero coef. = 0 
  Folds: 1...5....10   CVF = 26.82324 
  (output omitted)
  Grid value 40:   lambda = .1940106   no. of nonzero coef. = 52 
  Folds: 1...5....10   CVF = 15.07523 
  ... cross-validation complete ... minimum found 
  alpha 3 of 3: alpha = 0.5 
  (output omitted)
  10-fold cross-validation with 109 lambdas ...  
  Grid value 1:   lambda = 6.323778   no. of nonzero coef. = 0 
  Folds: 1...5....10   CVF = 26.78722 
  (output omitted)
\end{verbatim}
Grid value 46: lambda = .11102 no. of nonzero coef. = 115  
Folds: 1...5....10 CVF = 14.90808  
... cross-validation complete ... minimum found

Elastic net linear model  
No. of obs = 449  
No. of covariates = 275  
Selection: Cross-validation  
No. of CV folds = 10

<table>
<thead>
<tr>
<th>alpha</th>
<th>ID</th>
<th>Description</th>
<th>lambda</th>
<th>no. of nonzero coef.</th>
<th>Out-of-sample R-squared</th>
<th>CV mean prediction error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.000</td>
<td>1</td>
<td>first lambda</td>
<td>6.323778</td>
<td>0</td>
<td>0.0036</td>
<td>26.82324</td>
</tr>
<tr>
<td></td>
<td>42</td>
<td>last lambda</td>
<td>.161071</td>
<td>29</td>
<td>0.4339</td>
<td>15.12964</td>
</tr>
<tr>
<td>0.750</td>
<td>43</td>
<td>first lambda</td>
<td>6.323778</td>
<td>0</td>
<td>0.0022</td>
<td>26.78722</td>
</tr>
<tr>
<td></td>
<td>82</td>
<td>last lambda</td>
<td>.1940106</td>
<td>52</td>
<td>0.4360</td>
<td>15.07523</td>
</tr>
<tr>
<td>0.500</td>
<td>83</td>
<td>first lambda</td>
<td>6.323778</td>
<td>0</td>
<td>0.0022</td>
<td>26.78722</td>
</tr>
<tr>
<td></td>
<td>124</td>
<td>lambda before</td>
<td>.161071</td>
<td>87</td>
<td>0.4473</td>
<td>14.77189</td>
</tr>
<tr>
<td></td>
<td>125</td>
<td>selected lambda</td>
<td>.1467619</td>
<td>92</td>
<td>0.4476</td>
<td>14.76569</td>
</tr>
<tr>
<td></td>
<td>126</td>
<td>lambda after</td>
<td>.133724</td>
<td>96</td>
<td>0.4468</td>
<td>14.78648</td>
</tr>
<tr>
<td></td>
<td>128</td>
<td>last lambda</td>
<td>.11102</td>
<td>115</td>
<td>0.4422</td>
<td>14.90808</td>
</tr>
</tbody>
</table>

* alpha and lambda selected by cross-validation.

. estimates store elasticnet

Wonderful! It selected $\alpha^* = 0.5$. We should not stop here, however. There may be smaller values of $\alpha$ that give lower minimums of the CV function. If the number of observations and number of potential variables are not too large, you could specify the option \texttt{alpha(0(0.1)1)} the first time you run \texttt{elasticnet}. However, if we did this, the command would take much longer to run than the default. It will be especially slow for $\alpha = 0.1$ as we mentioned earlier.

. elasticnet linear q104 $\text{idemographics}$ $\text{ifactors}$ $\text{vlcontinuous}$  
> if sample == 1, rseed(1234) alpha(0.1 0.2 0.3)  
alpha 1 of 3: alpha = .3  
(output omitted)  
10-fold cross-validation with 113 lambdas ...  
Grid value 1: lambda = 31.61889 no. of nonzero coef. = 0  
Folds: 1...5....10 CVF = 26.82324  
(output omitted)  
Grid value 59: lambda = .160193 no. of nonzero coef. = 122  
Folds: 1...5....10 CVF = 14.84229  
... cross-validation complete ... minimum found  
alpha 2 of 3: alpha = .2  
(output omitted)  
10-fold cross-validation with 113 lambdas ...  
Grid value 1: lambda = 31.61889 no. of nonzero coef. = 0  
Folds: 1...5....10 CVF = 26.82324  
(output omitted)  
Grid value 56: lambda = .2117657 no. of nonzero coef. = 137  
Folds: 1...5....10 CVF = 14.81594  
... cross-validation complete ... minimum found  
alpha 3 of 3: alpha = .1  
(output omitted)
10-fold cross-validation with 113 lambdas ...

Grid value 1: lambda = 31.61889 no. of nonzero coef. = 0
Folds: 1...5...10 CVF = 26.81813

(output omitted)

Grid value 51: lambda = .3371909 no. of nonzero coef. = 162
Folds: 1...5...10 CVF = 14.81783
... cross-validation complete ... minimum found

Elastic net linear model
No. of obs = 449
No. of covariates = 275
Selection: Cross-validation
No. of CV folds = 10

<table>
<thead>
<tr>
<th>alpha</th>
<th>ID</th>
<th>Description</th>
<th>lambda</th>
<th>no. of nonzero coef.</th>
<th>R-squared</th>
<th>CV mean prediction error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.300</td>
<td>1</td>
<td>first lambda</td>
<td>31.61889</td>
<td>0</td>
<td>0.0036</td>
<td>26.82324</td>
</tr>
<tr>
<td></td>
<td>59</td>
<td>last lambda</td>
<td>0.160193</td>
<td>122</td>
<td>0.4447</td>
<td>14.84229</td>
</tr>
</tbody>
</table>

| 0.200 | 60 | first lambda      | 31.61889 | 0                     | 0.0036    | 26.82324                 |
| 110   |    | lambda before     | 0.3371909| 108                   | 0.4512    | 14.66875                 |
| * 111 |    | selected lambda   | 0.3072358| 108                   | 0.4514    | 14.66358                 |
| 112   |    | lambda after      | 0.2799418| 125                   | 0.4509    | 14.67566                 |
| 115   |    | last lambda       | 0.2117657| 137                   | 0.4457    | 14.81594                 |

| 0.100 | 116| first lambda      | 31.61889 | 0                     | 0.0034    | 26.81813                 |
| 166   |    | last lambda       | 0.3371909| 162                   | 0.4456    | 14.81783                 |

* alpha and lambda selected by cross-validation.
.
.estimates store elasticnet

The selected $\alpha^*$ is 0.2. This value is better, according to CV, than $\alpha = 0.1$ or $\alpha = 0.3$.
.
.cvplot

We can plot the CV function for the selected $\alpha^* = 0.2$. 
The CV function looks quite flat around the selected $\lambda^*$. We could assess alternative $\lambda$ (and alternative $\alpha$) using `lassoknots`. We run `lassoknots` with options requesting the number of nonzero coefficients be shown (nonzero), along with the CV function (cvmpe) and estimates of the out-of-sample $R^2$ (osr2).

```
. lassoknots, display(nonzero cvmpe osr2)
```

<table>
<thead>
<tr>
<th>alpha</th>
<th>ID</th>
<th>lambda</th>
<th>No. of nonzero coef.</th>
<th>CV mean pred. error</th>
<th>Out-of-sample R-squared</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.300</td>
<td>15</td>
<td>9.603319</td>
<td>4</td>
<td>26.42296</td>
<td>0.0114</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(output omitted)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>54</td>
<td>0.2550726</td>
<td>92</td>
<td>14.67746</td>
<td>0.4509</td>
</tr>
<tr>
<td></td>
<td>55</td>
<td>0.2324126</td>
<td>98</td>
<td>14.66803</td>
<td>0.4512</td>
</tr>
<tr>
<td></td>
<td>56</td>
<td>0.2117657</td>
<td>105</td>
<td>14.67652</td>
<td>0.4509</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(output omitted)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>59</td>
<td>0.160193</td>
<td>122</td>
<td>14.84229</td>
<td>0.4447</td>
</tr>
<tr>
<td>0.200</td>
<td>69</td>
<td>14.40498</td>
<td>4</td>
<td>26.54791</td>
<td>0.0067</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(output omitted)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>110</td>
<td>0.3371909</td>
<td>108</td>
<td>14.66875</td>
<td>0.4512</td>
</tr>
<tr>
<td>* 111</td>
<td></td>
<td>0.3072358</td>
<td>118</td>
<td>14.66358</td>
<td>0.4514</td>
</tr>
<tr>
<td></td>
<td>112</td>
<td>0.2799418</td>
<td>125</td>
<td>14.67566</td>
<td>0.4509</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(output omitted)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>115</td>
<td>0.2117657</td>
<td>137</td>
<td>14.81594</td>
<td>0.4457</td>
</tr>
<tr>
<td>0.100</td>
<td>117</td>
<td>28.80996</td>
<td>4</td>
<td>26.67947</td>
<td>0.0018</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(output omitted)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>161</td>
<td>0.5369033</td>
<td>143</td>
<td>14.76586</td>
<td>0.4476</td>
</tr>
<tr>
<td></td>
<td>162</td>
<td>0.4892063</td>
<td>148</td>
<td>14.75827</td>
<td>0.4478</td>
</tr>
<tr>
<td></td>
<td>162</td>
<td>0.4892063</td>
<td>148</td>
<td>14.75827</td>
<td>0.4478</td>
</tr>
<tr>
<td></td>
<td>163</td>
<td>0.4457466</td>
<td>152</td>
<td>14.76197</td>
<td>0.4477</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(output omitted)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>166</td>
<td>0.3371909</td>
<td>162</td>
<td>14.81783</td>
<td>0.4456</td>
</tr>
</tbody>
</table>

* alpha and lambda selected by cross-validation.

When we examine the output from `lassoknots`, we see that the CV function appears rather flat along $\lambda$ from the minimum and also across $\alpha$. 

\[\]
Example 3: Ridge regression

Let's continue with the previous example and fit a ridge regression. We do this by specifying \( \text{alpha}(0) \).

```
elasticnet linear q104 $idemographics $ifactors $vlcontinuous
> if sample == 1, rseed(1234) alpha(0)
(output omitted)
```

Evaluating up to 100 lambdas in grid ...

Grid value 1: lambda = 3.16e+08 no. of nonzero coef. = 275

Grid value 2: lambda = 2880.996 no. of nonzero coef. = 275

(output omitted)

Grid value 99: lambda = .3470169 no. of nonzero coef. = 275
Grid value 100: lambda = .3161889 no. of nonzero coef. = 275

10-fold cross-validation with 100 lambdas ...

Fold 1 of 10: 10....20....30....40....50....60....70....80....90....100
(output omitted)

Fold 10 of 10: 10....20....30....40....50....60....70....80....90....100
... cross-validation complete

Elastic net linear model

<table>
<thead>
<tr>
<th>No. of obs</th>
<th>No. of covariates</th>
</tr>
</thead>
<tbody>
<tr>
<td>449</td>
<td>275</td>
</tr>
</tbody>
</table>

Selection: Cross-validation

<table>
<thead>
<tr>
<th>alpha</th>
<th>Description</th>
<th>lambda</th>
<th>no. of nonzero coef.</th>
<th>R-squared</th>
<th>CV mean prediction error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000</td>
<td>first lambda</td>
<td>3161.889</td>
<td>275</td>
<td>0.0036</td>
<td>26.82323</td>
</tr>
<tr>
<td></td>
<td>lambda before</td>
<td>.9655953</td>
<td>275</td>
<td>0.4387</td>
<td>15.00168</td>
</tr>
<tr>
<td>* 89</td>
<td>selected lambda</td>
<td>.8798144</td>
<td>275</td>
<td>0.4388</td>
<td>14.99956</td>
</tr>
<tr>
<td>90</td>
<td>lambda after</td>
<td>.8016542</td>
<td>275</td>
<td>0.4386</td>
<td>15.00425</td>
</tr>
<tr>
<td>100</td>
<td>last lambda</td>
<td>.3161889</td>
<td>275</td>
<td>0.4198</td>
<td>15.50644</td>
</tr>
</tbody>
</table>

* alpha and lambda selected by cross-validation.

```
estimates store ridge
```

In this implementation, ridge regression selects \( \lambda^* \) using CV. We can plot the CV function.

```
cvpplot
```

In this implementation, ridge regression selects \( \lambda^* \) using CV. We can plot the CV function.
Example 4: Comparing elastic net, ridge regression, and lasso

We fit elastic net and ridge on half of the sample in the previous examples so we could evaluate the prediction on the other half of the sample.

Let’s continue with the data from example 2 and example 3 and fit a lasso.

```stata
. lasso linear q104 $idemographics $ifactors $vlcontinuous > if sample == 1, rseed(1234)
no. of obs = 449
note: 1.q14 dropped because of collinearity with another variable
note: 1.q136 dropped because of collinearity with another variable
10-fold cross-validation with 100 lambdas ...
Grid value 1: lambda = 3.161889 no. of nonzero coef. = 0
(output omitted)
Grid value 33: lambda = .161071 no. of nonzero coef. = 29
Folds: 1...5....10 CVF = 15.12964
... cross-validation complete ... minimum found
Lasso linear model No. of obs = 449
No. of covariates = 275
Selection: Cross-validation No. of CV folds = 10
ID Description lambda coef. R-squared error
1 first lambda 3.161889 0 0.0020 26.67513
28 lambda before .2564706 18 0.4348 15.10566
* 29 selected lambda .2336864 21 0.4358 15.07917
30 lambda after .2129264 21 0.4355 15.08812
33 last lambda .161071 29 0.4339 15.12964
* lambda selected by cross-validation.
. estimates store lasso
```

We stored the results of the earlier elastic net and ridge in memory using estimates store. We did the same for the lasso results. Now we can compare out-of-sample prediction using lassogof.

```stata
. lassogof elasticnet ridge lasso, over(sample)
Penalized coefficients
```

<table>
<thead>
<tr>
<th>Name</th>
<th>sample</th>
<th>MSE</th>
<th>R-squared</th>
<th>Obs</th>
</tr>
</thead>
<tbody>
<tr>
<td>elasticnet</td>
<td>Training</td>
<td>11.70471</td>
<td>0.5520</td>
<td>480</td>
</tr>
<tr>
<td></td>
<td>Testing</td>
<td>14.60949</td>
<td>0.4967</td>
<td>501</td>
</tr>
<tr>
<td>ridge</td>
<td>Training</td>
<td>11.82482</td>
<td>0.5576</td>
<td>449</td>
</tr>
<tr>
<td></td>
<td>Testing</td>
<td>14.88123</td>
<td>0.4809</td>
<td>476</td>
</tr>
<tr>
<td>lasso</td>
<td>Training</td>
<td>13.41709</td>
<td>0.4823</td>
<td>506</td>
</tr>
<tr>
<td></td>
<td>Testing</td>
<td>14.91674</td>
<td>0.4867</td>
<td>513</td>
</tr>
</tbody>
</table>

Elastic net did better out of sample based on the mean squared error and $R^2$ than ridge and lasso.

Note that the numbers of observations for both the training and testing samples were slightly different for each of the models. splitsample split the sample exactly in half with 529 observations in each half sample. The sample sizes across the models differ because the different models contain different sets of selected variables; hence, the pattern of missing values is different. If you want to
make the half samples exactly equal after missing values are dropped, an optional varlist containing
the dependent variable and all the potential variables can be used with \texttt{splitsample} to omit any
missing values in these variables. See \texttt{[D] splitsample}.

Before we conclude that elastic net won out over ridge and lasso, we must point out that we
were not fair to lasso. Theory states that for the lasso linear model, postselection coefficients provide
slightly better predictions. See \texttt{predict} in \texttt{[LASSO] lasso postestimation}.

We run \texttt{lassogof} again for the lasso results, this time specifying that postselection coefficients
be used.

\begin{verbatim}
. lassogof lasso, over(sample) postselection
Postselection coefficients

<table>
<thead>
<tr>
<th>Name</th>
<th>sample</th>
<th>MSE</th>
<th>R-squared</th>
<th>Obs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Training</td>
<td>13.14487</td>
<td>0.4928</td>
<td>506</td>
</tr>
<tr>
<td></td>
<td>Testing</td>
<td>14.62903</td>
<td>0.4966</td>
<td>513</td>
</tr>
</tbody>
</table>
\end{verbatim}

We declare a tie with elastic net!

Postselection coefficients should not be used with \texttt{elasticnet} and, in particular, with ridge
regression. Ridge works by shrinking the coefficient estimates, and these are the estimates that should
be used for prediction. Because postselection coefficients are OLS regression coefficients for the
selected coefficients and because ridge always selects all variables, postselection coefficients after
ridge are OLS regression coefficients for all potential variables, which clearly we do not want to use
for prediction.

\textbf{Stored results}

\texttt{elasticnet} stores the following in \texttt{e()}: 

- Scalars
  - \texttt{e(N)}: number of observations
  - \texttt{e(k_allvars)}: number of potential variables
  - \texttt{e(k_nonzero_sel)}: number of nonzero coefficients for selected model
  - \texttt{e(k_nonzero_cv)}: number of nonzero coefficients at CV mean function minimum
  - \texttt{e(k_nonzero_serule)}: number of nonzero coefficients for one-standard-error rule
  - \texttt{e(k_nonzero_min)}: minimum number of nonzero coefficients among estimated \( \lambda \)’s
  - \texttt{e(k_nonzero_max)}: maximum number of nonzero coefficients among estimated \( \lambda \)’s
  - \texttt{e(alpha_sel)}: value of selected \( \alpha^* \)
  - \texttt{e(alpha_cv)}: value of \( \alpha \) at CV mean function minimum
  - \texttt{e(lambda_sel)}: value of selected \( \lambda^* \)
  - \texttt{e(lambda_gmin)}: value of \( \lambda \) at grid minimum
  - \texttt{e(lambda_gmax)}: value of \( \lambda \) at grid maximum
  - \texttt{e(lambda_last)}: value of last \( \lambda \) computed
  - \texttt{e(lambda_cv)}: value of \( \lambda \) at CV mean function minimum
  - \texttt{e(lambda_serule)}: value of \( \lambda \) for one-standard-error rule
  - \texttt{e(ID_sel)}: ID of selected \( \lambda^* \)
  - \texttt{e(ID_cv)}: ID of \( \lambda \) at CV mean function minimum
  - \texttt{e(ID_serule)}: ID of \( \lambda \) for one-standard-error rule
  - \texttt{e(cvm_min)}: minimum CV mean function value
  - \texttt{e(cvm_serule)}: CV mean function value at one-standard-error rule
  - \texttt{e(devratio_min)}: minimum deviance ratio
  - \texttt{e(devratio_max)}: maximum deviance ratio
  - \texttt{e(L1_min)}: minimum value of \( \ell_1 \)-norm of penalized unstandardized coefficients
Methods and formulas

The methods and formulas for elastic net are given in *Methods and formulas* in [LASSO] lasso. Here we provide the methods and formulas for ridge regression, which is a special case of elastic net.

Unlike lasso and elastic net, ridge regression has a differentiable objective function, and there is a closed-form solution to the problem of minimizing the objective function. The solutions for ridge regression with nonlinear models are obtained by iteratively reweighted least squares.

The estimates of a generalized linear model (GLM) ridge regression model are obtained by minimizing

\[ \ell_r(\beta) = \frac{1}{N} \sum_{i=1}^{N} w_i f(y_i, \beta_0 + x_i \beta') + \frac{\lambda}{2} \sum_{j=1}^{p} \kappa_j \beta_j^2 \]

where \( w_i \) are the observation-level weights, \( y_i \) is the outcome, \( \beta_0 \) is a constant term, \( x_i \) is the \( p \)-dimensional vector of covariates, \( \beta \) is the \( p \)-dimensional vector of coefficients, \( f(y_i, \beta_0 + x_i \beta') \) is the contribution of the \( i \)th observation to the unpenalized objective function, \( \lambda \) is the penalty parameter, and \( \kappa_j \) are coefficient-level weights. By default, all the coefficient-level weights are 1.

When the model is linear,

\[ f(y_i, \beta_0 + x_i \beta') = \frac{1}{2} (y_i - \beta_0 - x_i \beta')^2 \]
When the model is logit,

\[ f(y_i, \beta_0 + x_i \beta') = -y_i(\beta_0 + x_i \beta') + \ln\{1 + \exp(\beta_0 + x_i \beta')\} \]

When the model is poisson,

\[ f(y_i, \beta_0 + x_i \beta') = -y_i(\beta_0 + x_i \beta') + \exp(\beta_0 + x_i \beta') \]

When the model is probit,

\[ f(y_i, \beta_0 + x_i \beta') = -y_i \ln\{\Phi(\beta_0 + x_i \beta')\} - (1 - y_i) \ln\{1 - \Phi(\beta_0 + x_i \beta')\} \]

For the linear model, the point estimates are given by

\[ (\hat{\beta}_0, \hat{\beta})' = \left( \sum_{i=1}^{N} w_i \tilde{x}_i' \tilde{x}_i + \lambda \tilde{I} \right)^{-1} \sum_{i=1}^{N} w_i y_i \tilde{x}_i' \]

where \( \tilde{x}_i = (1, x_i) \) and \( \tilde{I} \) is a diagonal matrix with the coefficient-level weights \( 0, \kappa_1, \ldots, \kappa_p \) on the diagonal.

For the nonlinear model, the optimization problem is solved using iteratively reweighted least squares. See Segerstedt (1992) and Nyquist (1991) for details of the iteratively reweighted least-squares algorithm for the GLM ridge-regression estimator.

References


Also see

[lasso] lasso postestimation — Postestimation tools for lasso for prediction
[lasso] lasso — Lasso for prediction and model selection
[sqrtlasso] sqrtlasso — Square-root lasso for prediction and model selection
[logit] logit — Logistic regression, reporting coefficients
[poisson] poisson — Poisson regression
[probit] probit — Probit regression
[regress] regress — Linear regression
[20 Estimation and postestimation commands]