elasticnet — Elastic net for prediction and model selection

Description

`elasticnet` selects covariates and fits linear, logistic, probit, Poisson, and Cox proportional hazards 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 `y1`, and select covariates from `x1` to `x100` using cross-validation (CV)

```
elasticnet linear y1 x1-x100
```

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

```
elasticnet linear y1 x1-x100, alpha(0.1(0.1)1)
```

Same as above, but force `x1` and `x2` to be in the model while `elasticnet` selects `x3` to `x100`

```
elasticnet linear y1 (x1 x2) x3-x100, alpha(0.1(0.1)1)
```

Fit a logistic model for binary outcome `y2` with grid $\alpha = 0.7, 0.8, 0.9, 1$

```
elasticnet logit y2 x1-x100, alpha(0.7 0.8 0.9 1)
```

Same as above, and set a random-number seed for reproducibility

```
elasticnet logit y2 x1-x100, alpha(0.7 0.8 0.9 1) rseed(1234)
```

Fit a Poisson model for count outcome `y3` with exposure `time`

```
elasticnet poisson y3 x1-x100, 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 y1 x1-x100, alpha(0.1(0.1)1) selection(cv, alllambdas)
```

Turn off the early stopping rule, and iterate over λ's until a minimum is found or until the end of the λ grid is reached

```
elasticnet linear y1 x1-x100, alpha(0.1(0.1)1) stop(0)
```

Fit a Cox proportional hazards model for `t` with failure indicator `fail`, and select covariates from `x1` to `x100` using CV

```
stset t, failure(fail)
elasticnet cox x1-x100
```

Same as above, but select covariates by minimizing the BIC

```
elasticnet cox x1-x100, selection(bic)
```

Menu

Statistics > Lasso > Elastic net

Syntax

For linear, logit, probit, and Poisson models

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

For Cox models

```
elasticnet cox [ (alwaysvars) ] othervars [ if ] [ in ] [ , 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.

options	Description
Model	
* `noconstant`	suppress constant term
`selection(cv [ , cv_opts ])`	select mixing parameter α^* and lasso penalty parameter λ^* using CV
`selection(bic [ , bic_opts ])`	select mixing parameter α^* and lasso penalty parameter λ^* using BIC
`selection(none)`	do not select α^* or λ^*
`offset(`*varname*$_o$`)`	include *varname*$_o$ in model with coefficient constrained to 1
`exposure(`*varname*$_e$`)`	include ln(*varname*$_e$) in model with coefficient constrained to 1 (`poisson` model only)
* `cluster(`*clustvar*`)`	specify cluster variable *clustvar*
Optimization	
`[no]log`	display or suppress an iteration log
`rseed(#)`	set random-number seed
`alphas(`*numlist* | *matname*`)`	specify the α grid with *numlist* or a matrix
`grid(`$\#_g$ `[ , ratio(#) min(#) ])`	specify the set of possible λ's using a logarithmic grid with $\#_g$ grid points
`crossgrid(augmented)`	augment the λ grids for each α as necessary to produce a single λ grid; the default
`crossgrid(union)`	use the union of the λ grids for each α to produce a single λ grid
`crossgrid(different)`	use different λ grids for each α
`stop(#)`	tolerance for stopping the iteration over the λ grid early
`cvtolerance(#)`	tolerance for identification of the CV function minimum
`bictolerance(#)`	tolerance for identification of the BIC function minimum
`tolerance(#)`	convergence tolerance for coefficients based on their values
`dtolerance(#)`	convergence tolerance for coefficients based on deviance
`penaltywt(`*matname*`)`	programmer's option for specifying a vector of weights for the coefficients in the penalty term

cv_opts	Description
`folds(#)`	use # folds for CV
`alllambdas`	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
`serule`	use the one-standard-error rule to select λ^*
`stopok`	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 λ_{stop}, allow λ_{stop} to be included in an (α, λ) pair that can potentially be selected as (α^*, λ^*); the default
`strict`	requires the CV function to have an identified minimum for every value of α; this is a stricter alternative to the default `stopok`
`gridminok`	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, λ_{gmin}, to be included in an (α, λ) pair that can potentially be selected as (α^*, λ^*); this is a looser alternative to the default `stopok` and is rarely used

bic_opts	Description
`alllambdas`	fit models for all λ's in the grid or until the `stop(#)` tolerance is reached; by default, the BIC function is calculated sequentially by λ, and estimation stops when a minimum is identified
`stopok`	when, for a value of α, the BIC function does not have an identified minimum and the `stop(#)` stopping criterion for λ was reached at λ_{stop}, allow λ_{stop} to be included in an (α, λ) pair that can potentially be selected as (α^*, λ^*); the default
`strict`	requires the BIC function to have an identified minimum for every value of α; this is a stricter alternative to the default `stopok`
`gridminok`	when, for a value of α, the BIC function does not have an identified minimum and the `stop(#)` stopping criterion for λ was not reached, allow the minimum of the λ grid, λ_{gmin}, to be included in an (α, λ) pair that can potentially be selected as (α^*, λ^*); this is a looser alternative to the default `stopok` and is rarely used
`postselection`	use postselection coefficients to compute BIC

*`noconstant` and `cluster()` are not allowed with `elasticnet cox`.

You must `stset` your data before using `elasticnet cox`; see [ST] **stset**.

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

`collect` is allowed; see [U] **11.1.10 Prefix commands**.

Default weights are not allowed. `iweights` are allowed with all *sel_method* options. `fweights` are allowed when `selection(plugin)`, `selection(bic)`, or `selection(none)` is specified. See [U] **11.1.6 weight**. For `elasticnet cox`, weights must be specified when you `stset` your data.

`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.

Model

`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. This option is not allowed with `cox`.

`selection(cv)`, `selection(bic)`, 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.

`serule` selects λ^* based on the "one-standard-error rule" recommended by Hastie, Tibshirani, and Wainwright (2015, 13–14) instead of the λ that minimizes the CV function. The one-standard-error rule selects, for each α, the largest λ for which the CV function is within a standard error of the minimum of the CV function. Then, from among these (α, λ) 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 α, the CV function does not have an identified minimum at any value of λ in the grid. A minimum is identified at λ_{cvmin} when the CV function at both larger and smaller adjacent λ's is greater than it is at λ_{cvmin}. When the CV function for a value of α has an identified minimum, these options all do the same thing: $(\alpha, \lambda_{\text{cvmin}})$ becomes one of the (α, λ) pairs that potentially can be selected as the smallest value of the CV function. In some cases, however, the CV function declines monotonically as λ 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 λ in the (α, λ) pairs that will be assessed for the smallest value of the CV function. The option `strict` makes no alternative pick for λ. You may specify only one of `stopok`, `strict`, or `gridminok`; `stopok` is the default if you do not specify one. With each of these options, estimation results are always left in place, and alternative (α, λ) pairs can be selected and evaluated.

`stopok` specifies that, for a value of α, when the CV function does not have an identified minimum and the `stop(`#`)` stopping tolerance for λ was reached at λ_{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. λ_{stop} is the smallest λ for which coefficients are estimated, and it is assumed that λ_{stop} has a CV function value close to the true minimum for that value of α. When no minimum is identified for a value of α 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 α, and if not, an error is issued.

`gridminok` is a rarely used option that specifies that, for a value of α, when the CV function has no identified minimum and the `stop(#)` stopping criterion was not met, λ_{gmin}, the minimum of the λ grid, is picked as part of a pair $(\alpha, \lambda_{\text{gmin}})$ 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 (α^*, λ^*) pair is the minimum of the CV function chosen from the $(\alpha, \lambda_{\text{cvmin}})$ pairs, where all λ's under consideration are identified minimums. With `stopok`, the set of (α, λ) pairs under consideration for the minimum of the CV function include identified minimums, λ_{cvmin}, or values, λ_{stop}, that met the stopping criterion. With `gridminok`, the set of (α, λ) pairs under consideration for the minimum of the CV function potentially include λ_{cvmin}, λ_{stop}, or λ_{gmin}.

`selection(bic` [`,` *bic_opts*]`)` selects (α^*, λ^*) by using the Bayesian information criterion (BIC) function. It selects the (α^*, λ^*) with the minimum BIC function value.

bic_opts are `alllambdas`, `stopok`, `strict`, `gridminok`, and `postselection`.

`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 BIC function is calculated after each model is fit. If a minimum of the BIC 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 BIC 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.

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

`stopok` specifies that, for a value of α, when the BIC function does not have an identified minimum and the `stop(#)` stopping tolerance for λ was reached at λ_{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 BIC function. λ_{stop} is the smallest λ for which coefficients are estimated, and it is assumed that λ_{stop} has a BIC function value close to the true minimum for that value of α. When no minimum is identified for a value of α and the `stop(#)` criterion is not met, an error is issued.

`strict` requires the BIC function to have an identified minimum for each value of α, and if not, an error is issued.

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

The `gridminok` criterion is looser than the default `stopok`, which is looser than `strict`. With `strict`, the selected (α^*, λ^*) pair is the minimum of the BIC function chosen from the $(\alpha, \lambda_{\text{bicmin}})$ pairs, where all λ's under consideration are identified minimums. With `stopok`, the set of (α, λ) pairs under consideration for the minimum of the BIC function include identified minimums, λ_{bicmin}, or values, λ_{stop}, that met the stopping criterion. With `gridminok`, the set of (α, λ) pairs under consideration for the minimum of the BIC function potentially include λ_{bicmin}, λ_{stop}, or λ_{gmin}.

`postselection` specifies to use the postselection coefficients to compute the BIC function. By default, the penalized coefficients are used.

`selection(none)` does not select an (α^*, λ^*) pair. In this case, the elastic net is estimated for a grid of values for λ for each α, but no attempt is made to determine which (α, λ) pair is best. The postestimation command `lassoknots` can be run to view a table of λ's that define the knots (that is, the distinct sets of nonzero coefficients) for each α. The `lassoselect` command can then be used to select an (α^*, λ^*) pair, and `lassogof` can be run to evaluate the prediction performance of the selected pair.

When `selection(none)` is specified, neither the CV function nor the BIC function is computed. If you want to view the knot table with values of the CV function shown and then select (α^*, λ^*), you must specify `selection(cv)`. Similarly, if you want to view the knot table with values of the BIC function shown, you must specify `selection(bic)`. 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*$_e$`)` can be specified only for the `poisson` model. It specifies that ln(*varname*$_e$) be included in the model with its coefficient constrained to be 1.

`cluster(`*clustvar*`)` specifies the cluster variable *clustvar*. Specifying a cluster variable will affect how the log-likelihood function is computed and the sample split in cross-validation. The log-likelihood function is computed as the sum of the log likelihood at the cluster levels. If option `selection(cv)` is specified, the cross-validation sample is split by the clusters defined by *clustvar*. That is, the subsample in each fold is drawn on the cluster level. Therefore, all observations in a cluster are kept together in the same subsample. This option is not allowed with `elasticnet cox`.

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(bic)` and `selection(none)` do 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 α. The default is `alphas(0.5 0.75 1)`. Specifying a small, nonzero value of α 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 λ's using a logarithmic grid with $\#_g$ grid points.

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

`ratio(#)` specifies $\lambda_{\text{gmin}}/\lambda_{\text{gmax}}$. The maximum of the grid, λ_{gmax}, is set to the smallest λ for which all the coefficients in the lasso are estimated to be zero (except the coefficients of the *alwaysvars*). λ_{gmin} 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 λ_{gmin}. By default, λ_{gmin} is based on `ratio(#)` and λ_{gmax}, which is computed from the data.

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

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

`crossgrid(union)` specifies that the union of λ grids across each value of α be used. That is, a λ grid for each α is computed, and then they are combined by simply putting all the λ values into one grid that is used for each α. 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 λ grids be used for each value of α. This option is rarely used. Using different λ grids for different values of α complicates the interpretation of the CV selection method. When the λ grid is not the same for every value of α, comparisons are based on parameter intervals that are not on the same scale.

`stop(#)` specifies a tolerance that is the stopping criterion for the λ iterations. The default is 1e−5. Estimation starts with the maximum grid value, λ_{gmax}, and iterates toward the minimum grid value, λ_{gmin}. When the relative difference in the deviance produced by two adjacent λ grid values is less than `stop(#)`, the iteration stops and no smaller λ's are evaluated. The value of λ that meets this tolerance is denoted by λ_{stop}. Typically, this stopping criterion is met before the iteration reaches λ_{gmin}.

Setting `stop(#)` to a larger value means that iterations are stopped earlier at a larger λ_{stop}. To produce coefficient estimates for all values of the λ grid, you can specify `stop(0)`. Note, however, that computations for small λ's can be extremely time consuming. In terms of time, when you use `selection(cv)` or `selection(bic)`, the optimal value of `stop(#)` is the largest value that allows estimates for just enough λ's to be computed to identify the minimum of the CV or BIC function. When setting `stop(#)` to larger values, be aware of the consequences of the default λ^* 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 identified when the CV function rises above a nominal minimum for at least three smaller λ's with a relative difference in the CV function greater than #. For nonlinear models, at least five smaller λ'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 λ, which can be time consuming. See *Methods and formulas* for [LASSO] **lasso** for more information about this tolerance and the other tolerances.

`bictolerance(#)` is a rarely used option that changes the tolerance for identifying the minimum BIC function. A minimum is identified when the BIC function rises above a nominal minimum for at least two smaller λ's with a relative difference in the BIC function greater than #. The default is 1e−2. 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 λ, which can be time consuming. See *Methods and formulas* in [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

$$Q = \frac{1}{2N}\sum_{i=1}^{N}(y_i - \beta_0 - \mathbf{x}_i\boldsymbol{\beta}')^2 + \lambda\sum_{j=1}^{p}\left(\frac{1-\alpha}{2}\beta_j^2 + \alpha\,|\beta_j|\right)$$

where $\boldsymbol{\beta}$ is the p-dimensional vector of coefficients on covariates $\mathbf{x}$. The estimated $\boldsymbol{\beta}$ are those that minimize Q for given values of α and λ.

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 λ than are lasso solutions because of the added ridge-regression component.

To fit a model with `elasticnet`, you specify a set of candidate α's and a grid of λ values. CV is performed on the combined set of (α, λ) values, and the (α^*, λ^*) 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 α and a finer grid for λ. The idea is that only a few points in the space between ridge regression and lasso are worth reviewing, but a finer grid over λ is needed to trace out the paths of which coefficients are not zero.

The default candidate values of α are 0.5, 0.75, and 1. Typically, you would use the default first and then set α using the `alpha(`*numlist*`)` option to get lower and upper bounds on α^*. Models for small, nonzero values of α take more time to estimate than $\alpha = 0$ and larger values of α. 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 λ grid is set automatically, and the default settings are typically sufficient to determine λ^*. 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/r19/fakesurvey_vl
(Fictitious survey data with vl)
. vl rebuild
Rebuilding vl macros ...
```

	Macro's contents	
Macro	# Vars	Description
System		
$vldummy	98	0/1 variables
$vlcategorical	16	categorical variables
$vlcontinuous	29	continuous variables
$vluncertain	16	perhaps continuous, perhaps categorical variables
$vlother	12	all missing or constant variables
User		
$demographics	4	variables
$factors	110	variables
$idemographics		factor-variable list
$ifactors		factor-variable list

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 [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)
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
```

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

```
* 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 `vl rebuild`.

```
. use https://www.stata-press.com/data/r19/fakesurvey2_vl, clear
(Fictitious survey data with vl)
. vl rebuild
Rebuilding vl macros ...
```

(*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 `splitsample` to generate a variable indicating the samples.

```
. set seed 1234
. splitsample, generate(sample) nsplit(2)
. label define svalues 1 "Training" 2 "Testing"
. label values sample svalues
```

We fit an elastic-net model using the default α's.

```
. 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)
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
```

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

```
* 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 α 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 `alpha(0(0.1)1)` the first time you run `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 $idemographics $ifactors $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
```

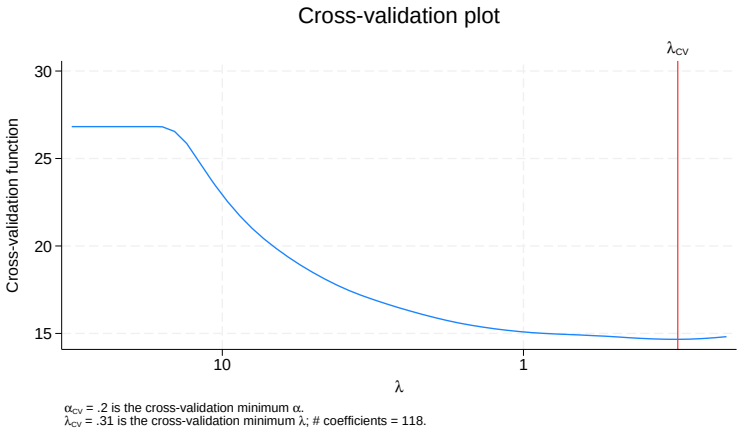
alpha	ID	Description	lambda	No. of nonzero coef.	Out-of-sample R-squared	CV mean prediction error
0.300						
	1	first lambda	31.61889	0	-0.0036	26.82324
	59	last lambda	.160193	122	0.4447	14.84229
0.200						
	60	first lambda	31.61889	0	-0.0036	26.82324
	110	lambda before	.3371909	108	0.4512	14.66875
*	111	selected lambda	.3072358	118	0.4514	14.66358
	112	lambda after	.2799418	125	0.4509	14.67566
	115	last lambda	.2117657	137	0.4457	14.81594
0.100						
	116	first lambda	31.61889	0	-0.0034	26.81813
	166	last lambda	.3371909	162	0.4456	14.81783

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

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

We can plot the CV function for the selected $\alpha^* = 0.2$.

```
. cvplot
```



The CV function looks quite flat around the selected λ^*. We could assess alternative λ (and alternative α) 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)
```

alpha		ID	lambda	No. of nonzero coef.	CV mean pred. error	Out-of-sample R-squared
0.300						
		15	9.603319	4	26.42296	0.0114
	(*output omitted*)					
		54	.2550726	92	14.67746	0.4509
		55	.2324126	98	14.66803	0.4512
		56	.2117657	105	14.67652	0.4509
	(*output omitted*)					
		59	.160193	122	14.84229	0.4447
0.200						
		69	14.40498	4	26.54791	0.0067
	(*output omitted*)					
		110	.3371909	108	14.66875	0.4512
	*	111	.3072358	118	14.66358	0.4514
		112	.2799418	125	14.67566	0.4509
	(*output omitted*)					
		115	.2117657	137	14.81594	0.4457

```
0.100
        117 | 28.80996          4   26.67947      0.0018
  (output omitted)
        161 | .5369033        143   14.76586      0.4476
        162 | .4892063        148   14.75827      0.4478
        162 | .4892063        148   14.75827      0.4478
        163 | .4457466        152   14.76197      0.4477
  (output omitted)
        166 | .3371909        162   14.81783      0.4456
------------------------------------------------------------------------------
* alpha and lambda selected by cross-validation.
```

When we examine the output from `lassoknots`, we see that the CV function appears rather flat along λ from the minimum and also across α.

◁

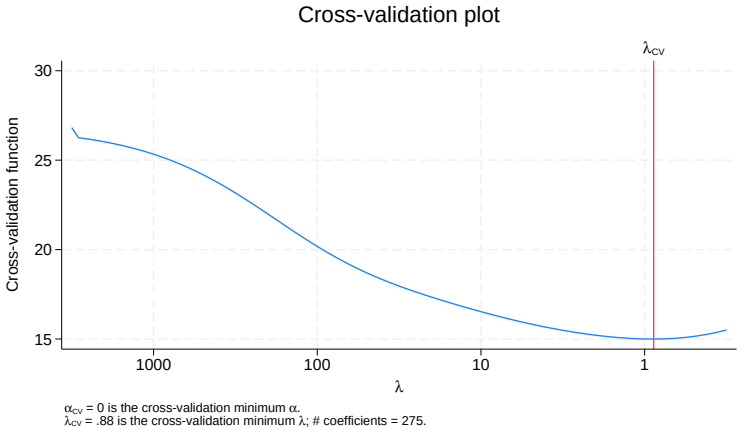
▷ Example 3: Ridge regression

Let's continue with the previous example and fit a ridge regression. We do this by specifying `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                      No. of obs        =        449
                                              No. of covariates =        275
Selection: Cross-validation                   No. of CV folds   =         10
------------------------------------------------------------------------------
                                                    No. of      Out-of-   CV mean
                                                    nonzero       sample prediction
alpha       ID     Description      lambda          coef.   R-squared      error
------------------------------------------------------------------------------
0.000
             1     first lambda    3161.889            275     -0.0036   26.82323
            88    lambda before    .9655953            275      0.4387   15.00168
          * 89  selected lambda    .8798144            275      0.4388   14.99956
            90     lambda after    .8016542            275      0.4386   15.00425
           100      last lambda    .3161889            275      0.4198   15.50644
------------------------------------------------------------------------------
* alpha and lambda selected by cross-validation.
. estimates store ridge
```

In this implementation, ridge regression selects λ^* using CV. We can plot the CV function.

```
. cvplot
```



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.

```
. lasso linear q104 $idemographics $ifactors $vlcontinuous
> if sample == 1, rseed(1234)
note: 1.q14 omitted because of collinearity with another variable.
note: 1.q136 omitted 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	No. of nonzero coef.	Out-of-sample R-squared	CV mean prediction 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`.

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

Name	sample	MSE	R-squared	Obs
elasticnet				
	Training	11.70471	0.5520	480
	Testing	14.60949	0.4967	501
ridge				
	Training	11.82482	0.5576	449
	Testing	14.88123	0.4809	476
lasso				
	Training	13.41709	0.4823	506
	Testing	14.91674	0.4867	513

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 `splitsample` to omit any missing values in these variables. See [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 `predict` in [LASSO] **lasso postestimation**.

We run `lassogof` again for the lasso results, this time specifying that postselection coefficients be used.

```
. lassogof lasso, over(sample) postselection
Postselection coefficients
```

Name	sample	MSE	R-squared	Obs
lasso				
	Training	13.14487	0.4928	506
	Testing	14.62903	0.4966	513

We declare a tie with elastic net!

Postselection coefficients should not be used with `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.

◁

Stored results

elasticnet stores the following in e():

Scalars

e(N)	number of observations
e(N_clust)	number of clusters
e(k_allvars)	number of potential variables
e(k_nonzero_sel)	number of nonzero coefficients for selected model
e(k_nonzero_cv)	number of nonzero coefficients at CV mean function minimum
e(k_nonzero_serule)	number of nonzero coefficients for one-standard-error rule
e(k_nonzero_min)	minimum number of nonzero coefficients among estimated λ's
e(k_nonzero_max)	maximum number of nonzero coefficients among estimated λ's
e(k_nonzero_bic)	number of nonzero coefficients at BIC function minimum
e(alpha_sel)	value of selected α^*
e(alpha_cv)	value of α at CV mean function minimum
e(lambda_sel)	value of selected λ^*
e(lambda_gmin)	value of λ at grid minimum
e(lambda_gmax)	value of λ at grid maximum
e(lambda_last)	value of last λ computed
e(lambda_cv)	value of λ at CV mean function minimum
e(lambda_serule)	value of λ for one-standard-error rule
e(lambda_bic)	value of λ at BIC function minimum
e(ID_sel)	ID of selected λ^*
e(ID_cv)	ID of λ at CV mean function minimum
e(ID_serule)	ID of λ for one-standard-error rule
e(ID_bic)	ID of λ at BIC function minimum
e(cvm_min)	minimum CV mean function value
e(cvm_serule)	CV mean function value at one-standard-error rule
e(devratio_min)	minimum deviance ratio
e(devratio_max)	maximum deviance ratio
e(L1_min)	minimum value of ℓ_1-norm of penalized unstandardized coefficients
e(L1_max)	maximum value of ℓ_1-norm of penalized unstandardized coefficients
e(L2_min)	minimum value of ℓ_2-norm of penalized unstandardized coefficients
e(L2_max)	maximum value of ℓ_2-norm of penalized unstandardized coefficients
e(ll_sel)	log-likelihood value of selected model
e(n_lambda)	number of λ's
e(n_fold)	number of CV folds
e(stop)	stopping rule tolerance

Macros

e(cmd)	elasticnet
e(cmdline)	command as typed
e(depvar)	name of dependent variable
e(allvars)	names of all potential variables
e(allvars_sel)	names of all selected variables
e(alwaysvars)	names of always-included variables
e(othervars_sel)	names of other selected variables
e(post_sel_vars)	all variables needed for postelastic net
e(clustvar)	name of cluster variable
e(lasso_selection)	selection method
e(sel_criterion)	criterion used to select λ^*
e(crossgrid)	type of two-dimensional grid
e(model)	linear, logit, probit, poisson, or cox
e(title)	title in estimation output
e(rngstate)	random-number state used
e(properties)	b
e(predict)	program used to implement predict
e(marginsnotok)	predictions disallowed by margins

Matrices

`e(b)`	penalized unstandardized coefficient vector
`e(b_standardized)`	penalized standardized coefficient vector
`e(b_postselection)`	postselection coefficient vector

Functions

`e(sample)`	marks estimation sample

In addition to the above, the following is stored in `r()`:

Matrices

`r(table)`	matrix containing the coefficients with their standard errors, test statistics, p-values, and confidence intervals

Note that results stored in `r()` are updated when the command is replayed and will be replaced when any r-class command is run after the estimation command.

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

$$Q_L = \sum_{i=1}^{N} \widetilde{w}_i f(y_i, \beta_0 + \mathbf{x}_i\boldsymbol{\beta}') + \frac{\lambda}{2}\sum_{j=1}^{p} \kappa_j \beta_j^2$$

where N is the number of observations; $\widetilde{w}_i$ is the normalized observation-level weight; $f(\cdot)$ is the likelihood contribution for the `regress`, `logit`, `probit`, or `poisson` model; β_0 is the intercept; $\mathbf{x}_i$ is the $1 \times p$ vector of covariates; $\boldsymbol{\beta}$ is the $1 \times p$ vector of coefficients; λ is the ridge penalty parameter, which must be greater than or equal to 0; and κ_j are coefficient-level weights (which by default are all 1).

The normalized weights $\widetilde{w}_i$ sum to 1. That is,

$$\widetilde{w}_i = \frac{w_i}{\sum_{i=1}^{N} w_i}$$

where w_i is the original observation-level weight. If weights are not specified with `elasticnet`, $w_i = 1$ and $\widetilde{w}_i = 1/N$.

The penalized objective function of the ridge regression for the `cox` model is

$$Q_L = -\sum_{j=1}^{N_f} \sum_{i \in D_j} \widetilde{w}_i \left[\mathbf{x}_i\boldsymbol{\beta}' - \log\left\{ \sum_{\ell \in R_j} \widetilde{w}_\ell \exp(\mathbf{x}_\ell\boldsymbol{\beta}') \right\} \right] + \frac{\lambda}{2}\sum_{j=1}^{p} \kappa_j \beta_j^2$$

where j indexes the ordered failure times $t_{(j)}$, $j = 1, \ldots, N_f$; D_j is the set of observations that fail at $t_{(j)}$; and R_j is the set of observations k that are at risk at time $t_{(j)}$ (that is, all k such that $t_{0k} < t_{(j)} \leq t_k$, and t_{0k} is the entry time for the kth observation).

When the model is `linear`,

$$f(y_i, \beta_0 + \mathbf{x}_i\boldsymbol{\beta}') = \frac{1}{2}(y_i - \beta_0 - \mathbf{x}_i\boldsymbol{\beta}')^2$$

When the model is `logit`,

$$f(y_i, \beta_0 + \mathbf{x}_i\boldsymbol{\beta}') = -y_i(\beta_0 + \mathbf{x}_i\boldsymbol{\beta}') + \ln\{1 + \exp(\beta_0 + \mathbf{x}_i\boldsymbol{\beta}')\}$$

When the model is `probit`,

$$f(y_i, \beta_0 + \mathbf{x}_i\boldsymbol{\beta}') = -y_i \ln\{\Phi(\beta_0 + \mathbf{x}_i\boldsymbol{\beta}')\} - (1 - y_i)\ln\{1 - \Phi(\beta_0 + \mathbf{x}_i\boldsymbol{\beta}')\}$$

When the model is `poisson`,

$$f(y_i, \beta_0 + \mathbf{x}_i\boldsymbol{\beta}') = -y_i(\beta_0 + \mathbf{x}_i\boldsymbol{\beta}') + \exp(\beta_0 + \mathbf{x}_i\boldsymbol{\beta}')$$

For the `linear` model, the point estimates are given by

$$(\hat{\beta}_0, \widehat{\boldsymbol{\beta}})' = \left(\sum_{i=1}^{N} \widetilde{w}_i \tilde{\mathbf{x}}_i' \tilde{\mathbf{x}}_i + \lambda\tilde{\mathbf{I}}\right)^{-1} \sum_{i=1}^{N} \widetilde{w}_i y_i \tilde{\mathbf{x}}_i'$$

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

For the nonlinear models, 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

Hastie, T. J., R. J. Tibshirani, and M. Wainwright. 2015. *Statistical Learning with Sparsity: The Lasso and Generalizations*. Boca Raton, FL: CRC Press. https://doi.org/10.1201/b18401.

Nyquist, H. 1991. Restricted estimation of generalized linear models. *Journal of the Royal Statistical Society*, C ser., 40: 133–141. https://doi.org/10.2307/2347912.

Segerstedt, B. 1992. On ordinary ridge regression in generalized linear models. *Communications in Statistics—Theory and Methods* 21: 2227–2246. https://doi.org/10.1080/03610929208830909.

Zou, H., and T. J. Hastie. 2005. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society*, B ser., 67: 301–320. https://doi.org/10.1111/j.1467-9868.2005.00503.x.

Also see

[LASSO] **lasso postestimation** — Postestimation tools for lasso for prediction

[LASSO] **lasso** — Lasso for prediction and model selection

[LASSO] **Lasso intro** — Introduction to lasso

[LASSO] **sqrtlasso** — Square-root lasso for prediction and model selection

[R] **logit** — Logistic regression, reporting coefficients

[R] **poisson** — Poisson regression

[R] **probit** — Probit regression

[R] **regress** — Linear regression

[ST] **stset** — Declare data to be survival-time data

[U] 20 Estimation and postestimation commands