Cross Validated

Feature selection & model with glmnet on Methylation data (p>>N)

Asked 6 years, 2 months ago Active 2 years, 7 months ago Viewed 20k times

29

22

I would like to use GLM and Elastic Net to select those relevant features + build a linear regression model (i.e., both prediction and understanding, so it would be better to be left with relatively few parameters). The output is continuous. It's 20000 genes per 50 cases. I've been reading about the `glmnet` package, but I'm not 100% sure about the steps to follow:

1. Perform CV to choose lambda:
   `cv <- cv.glmnet(x,y,alpha=0.5)`
   (Q1) given the input data, would you choose a different alpha value?
   (Q2) do I need to do something else before build the model?
2. Fit the model:
   `model=glmnet(x,y,type.gaussian="covariance",lambda=cv$lambda.min)`
   (Q3) anything better than "covariance"?
   (Q4) If lambda was chosen by CV, why does this step need `nlambda=` ?
   (Q5) is it better to use `lambda.min` or `lambda.1se` ?
3. Obtain the coefficients, to see which parameters have fallen out ("."):
   `predict(model, type="coefficients")`

   In the help page there are many `predict` methods (e.g., `predict.fishnet` , `predict.glmnet` , `predict.lognet` , etc). But any "plain" predict as I saw on an example.
   (Q6) Should I use `predict` or `predict.glmnet` or other?

Despite what I've read about regularization methods, I'm quite new in R and in these statistical packages, so it's difficult to be sure if I'm adapting my problem to the code. Any suggestions will be welcomed.

UPDATE
Based on "As previously noted, an object of class train contains an element called `finalModel` , which is the fitted model with the tuning parameter values selected by resampling. This object can be used in the traditional way to generate predictions for new samples, using that model's predict function."

Using `caret` to tune both alpha and lambda:

```
trc = trainControl(method=cv, number=10)
fitM = train(x, y, trControl = trC, method="glmnet")
```

Does `fitM` replace previous step 2? If so, how to specify the glmnet options (`type.gaussian="naive",lambda=cv$lambda.min/1se`) now?
And the following `predict` step, can I replace `model` to `fitM` ?

If I do

does it make sense at all or am I incorrectly mixing both package vocabulary?

r regularization glmnet elastic-net underdetermined

edited Sep 18 '13 at 15:29

asked Sep 17 '13 at 13:34
PGreen
515 1 5 11

1 Answer

42

Part 1

In the elastic net two types of constraints on the parameters are employed

1. Lasso constraints (i.e. on the size of the absolute values of β_j)
2. Ridge constraints (i.e. on the size of the squared values of β_j)

α controls the relative weighting of the two types. The Lasso constraints allow for the selection/removal of variables in the model. The ridge constraints can cope with collinear variables. Which you put more weight upon will depend on the data properties; lots of correlated variables may need both constraints, a few correlated variables might suggest more emphasis on ridge constraints.

One way to solve this is to treat α as a tuning parameter alongside λ and use the values that give the lowest CV error, in the same way that you are tuning over λ at the moment with `cv.glmnet`.

The R package caret can build models using the **glmnet** package and should be set up to tune over both parameters α and λ.

Part 2

Q3

Yes, in this case where $m \gg n$ (number of variables $\gg$ number of observations), the help page for `?glmnet` suggests to use

```
type.gaussian = "naive"
```

Instead of storing all the inner-products computed along the way, which can be inefficient with a large number of variables or when $m \gg n$, the `"naive"` option will loop over n each time it is required to computer inner products.

If you had not specified this argument, `glmnet` would have chosen `"naive"` anyway as $m > 500$, but it is better to specify this explicitly incase the defaults and options change later in the package and you are running code at a future date.

Q4

then having a modest set of values of λ over the interval results in a much nicer set of paths. The computational burden of doing the entire path relative to one specific λ is not that great, the result of a lot of effort to develop algorithms to do this job correctly. I would just leave `nlambda` at the default, unless it makes an appreciable difference in compute time.

Q5

This is a question about parsimony. The `lambda.min` option refers to value of λ at the lowest CV error. The error at this value of λ is the average of the errors over the k folds and hence this *estimate* of the error is uncertain. The `lambda.1se` represents the value of λ in the search that was simpler than the best model (`lambda.min`), but which has error within **1** standard error of the best model. In other words, using the value of `lambda.1se` as the selected value for λ results in a model that is slightly simpler than the best model but which cannot be distinguished from the best model in terms of error given the uncertainty in the k-fold CV estimate of the error of the best model.

The choice is yours:

1. The best model that may be too complex of slightly overfitted: `lambda.min`
2. The simplest model that has comparable error to the best model given the uncertainty: `lambda.1se`

Part 3

This is a simple one and is something you'll come across a lot with R. You use the `predict()` function 99.9% of the time. R will arrange for the use of the correct function for the object supplied as the first argument.

More technically, `predict` is a generic function, which has *methods* (versions of the function) for objects of different types (technically known as *classes*). The object created by `glmnet` has a particular class (or classes) depending on what type of model is actually fitted. **glmnet** (the package) provides methods for the `predict` function for these different types of objects. R knows about these methods and will choose the appropriate one based on the class of the object supplied.

edited Mar 16 '17 at 16:02
Community ♦
1

answered Sep 17 '13 at 15:58
Reinstate Monica - G. Simpson
29.3k 4 81 125

2 GREAT anwser! I've read now about caret. Not 100% sure about the relationship between caret and glmnet package, so I did an update to my question to clarify the 'merge' of these to packages or either the switch from glmnet to caret. – PGreen Sep 18 '13 at 11:24

1 @PGreen: `caret` is an R wrapper package which wraps function interfaces from 100+ ML packages to be more consistent and adds CV, gridsearch, modifies insane parameter defaults etc. It isn't without its quirks but it's pretty good and widely used. – smci Feb 1 '17 at 3:00

I just want to add that for alpha tuning you can use `cva.glmnet(..)` instead of just `cv.glmnet(...)` and tune alpha and lambda at the same time. Then you can run `minlossplot(cva.fit)` to see which alpha gives the best result. This is both part of glmnetUtils – Espen Riskedal Nov 9 '18 at 10:21