Model selection and validation 2: Model assessment, more cross-validation

Ryan Tibshirani
Data Mining: 36-462/36-662

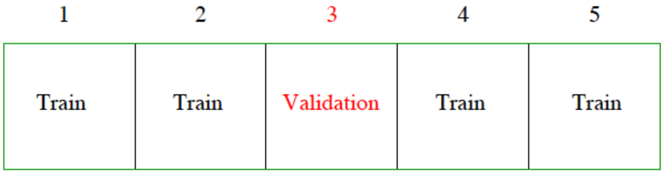
March 28 2013

Optional reading: ISL 5.1, ESL 7.10, 7.12

Reminder: cross-validation

Given training data (x_i, y_i), $i = 1, \ldots n$, we construct an estimator $\hat{f}$ of some unknown function f. Suppose that $\hat{f} = \hat{f}_\theta$ depends on a tuning parameter θ

How to choose a value of θ to optimize predictive accuracy of $\hat{f}_\theta$? Cross-validation offers one way. Basic idea is simple: divide up training data into K folds (here K is fixed, e.g., $K = 5$ or $K = 10$)

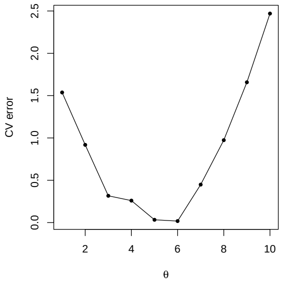


(Typically this is done at random)

Then, we hold out each fold one at a time, train on the remaining data, and predict the held out observations, for each value of the tuning parameter

I.e., for each value of the tuning parameter θ, the cross-validation error is

$$\mathrm{CV}(\theta) = \frac{1}{n} \sum_{k=1}^{K} \sum_{i \in F_k} \left(y_i - \hat{f}_\theta^{-k}(x_i) \right)^2$$



We choose the value of tuning parameter that minimizes the CV error curve,

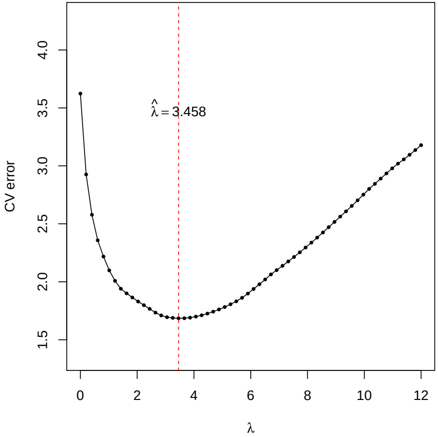
$$\hat{\theta} = \underset{\theta \in \{\theta_1, \ldots \theta_m\}}{\operatorname{argmin}} \ \mathrm{CV}(\theta)$$

Example: choosing λ for the lasso

Example from last time: $n = 50$, $p = 30$, and the true model is linear with 10 nonzero coefficients. We consider the lasso estimate

$$\hat{\beta}^{\text{lasso}} = \underset{\beta \in \mathbb{R}^p}{\text{argmin}} \, \|y - X\beta\|_2^2 + \lambda \|\beta\|_1$$

Performing 5-fold cross-validation, over 60 values of the tuning parameter between 0 and 12, we choose $\hat{\lambda} = 3.458$



What to do next?

What do we do next, after having used cross-validation to choose a value of the tuning parameter $\hat{\theta}$?

It may be an obvious point, but worth being clear: we now fit our estimator to the entire training set (x_i, y_i), $i = 1, \ldots n$, using the tuning parameter value $\hat{\theta}$

E.g., in the last lasso example, we resolve the lasso problem

$$\hat{\beta}^{\text{lasso}} = \underset{\beta \in \mathbb{R}^p}{\operatorname{argmin}} \, \|y - X\beta\|_2^2 + \lambda \|\beta\|_1$$

on all of the training data, with $\hat{\lambda} = 3.458$

We can then use this estimator $\hat{\beta}^{\text{lasso}}$ to make future predictions

Reminder: standard errors

Recall that we can compute standard errors for the CV error curve at each tuning parameter value θ. First define, for $k = 1, \dots K$:

$$\mathrm{CV}_k(\theta) = \frac{1}{n_k} \sum_{i \in F_k} \big(y_i - \hat{f}_\theta^{-k}(x_i)\big)^2$$

where n_k is the number of points in the kth fold

Then we compute the sample standard deviation of $\mathrm{CV}_1(\theta), \dots \mathrm{CV}_K(\theta)$,

$$\mathrm{SD}(\theta) = \sqrt{\mathrm{var}\big(\mathrm{CV}_1(\theta), \dots \mathrm{CV}_K(\theta)\big)}$$

Finally we use

$$\mathrm{SE}(\theta) = \mathrm{SD}(\theta)/\sqrt{K}$$

for the standard error of $\mathrm{CV}(\theta)$

Reminder: one standard error rule

Recall that the one standard error rule is an alternative way of choosing θ from the CV curve. We start with the usual estimate

$$\hat{\theta} = \underset{\theta \in \{\theta_1, \ldots \theta_m\}}{\operatorname{argmin}} \ \mathrm{CV}(\theta)$$

and we move θ in the direction of increasing regularization until it ceases to be true that

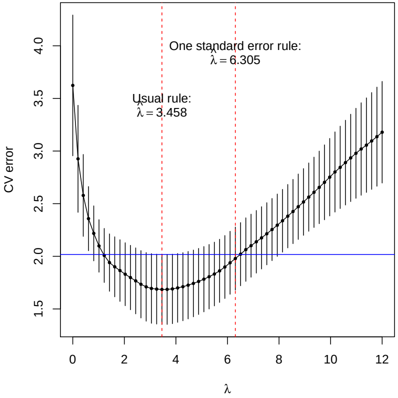
$$\mathrm{CV}(\theta) \leq \mathrm{CV}(\hat{\theta}) + \mathrm{SE}(\hat{\theta})$$

In words, we take the simplest (most regularized) model whose error is within one standard error of the minimal error

Example: choosing λ for the lasso

In the lasso criterion, larger λ means more regularization

For our last example, applying the one standard error rule has us increase the tuning parameter choice from $\hat{\lambda} = 3.458$ all the way up until $\hat{\lambda} = 6.305$



When fitting on the whole training data, this is a difference between a model with 19 and 16 nonzero coefficients

(Remember the true model had 10)

Prediction versus interpretation

Remember: cross-validation error estimates prediction error at any fixed value of the tuning parameter, and so by using it, we are implicitly assuming that achieving minimal prediction error is our goal

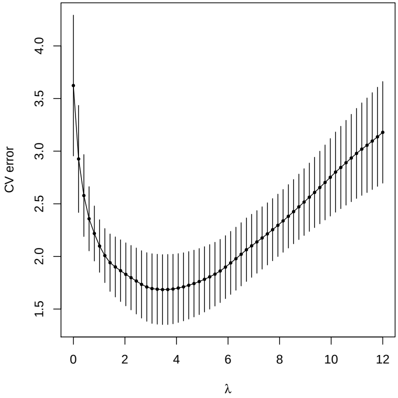
Choosing λ for the goal of recovering the true model, for the sake of interpretation, is somewhat of a different task. The value of the parameter that achieves the smallest cross-validation error often corresponds to not enough regularization for these purposes. But the one standard error rule is a step in the right direction

There are other (often more complicated) schemes for selecting a value of the tuning parameter for the purposes of true model recovery

Model assessment

Instead of just using the CV error curves to pick a value of the tuning parameter, suppose that we wanted to use cross-validation to actually estimate the prediction error values themselves

I.e., suppose that we actually cared about the values of the CV error curve, rather than just the location of its minimum

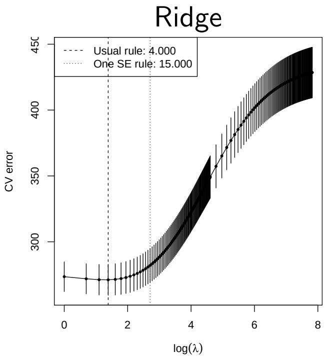


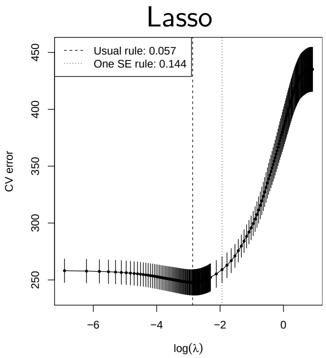
This is called model assessment (as opposed to model selection)

Why would we want to do this? Multiple reasons:

- Report estimate for prediction error to collaborator
- Compare estimated prediction error to known prediction errors of other estimators
- Answering the question: is my estimator $\hat{f}$ working at all?
- Choosing between different estimators (each possibly having their own tuning parameters)

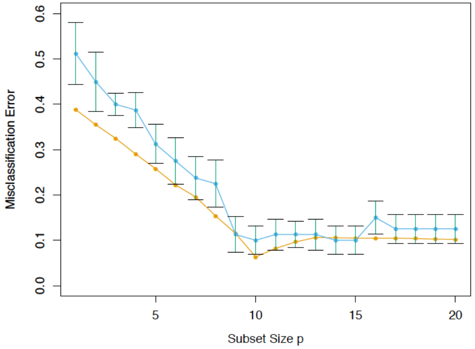
E.g., from Homework 4:





Generally speaking, cross-validation error produces good estimates of prediction error

An example from ESL page 244:



Prediction error is in orange, cross-validation error is in blue (and one standard errors bars are drawn)

Choice of K

Essentially, the larger the choice of K, the more iterations needed, and so the more computation. So taking $K = n$ (i.e., leave-one-out cross-validation) is going to be a lot more costly than $K = 5$

Aside from computation, the choice of K affects the quality of our cross-validation error estimates for model assessment. Consider:

- $K = 2$: split-sample cross-validation. Our CV error estimates are going to be biased upwards, because we are only training on half the data each time
- $K = n$: leave-one-out cross-validation. Our CV estimates

  $$\mathrm{CV}(\theta) = \frac{1}{n}\sum_{i=1}^{n}\big(y_i - \hat{f}_\theta^{-i}(x_i)\big)^2$$

  are going to have high variance because we're averaging a bunch of (positively) correlated quantities

The bias-variance tradeoff, again!

Choosing $K = 5$ or $K = 10$ seems to generally be a good tradeoff

- In each iteration we train on a fraction of (about) $(K-1)/K$ the total training set, so this reduces the bias
- There is less overlap between the training sets across iterations, so the terms in

$$\mathrm{CV}(\theta) = \frac{1}{n} \sum_{k=1}^{K} \sum_{i \in F_k} \big(y_i - \hat{f}_\theta^{-k}(x_i)\big)^2$$

  are not as correlated, and the error estimate has a smaller variance

The choices $K = 5$ or $K = 10$ are pretty much the standards, and people believe that these give good estimates of prediction error, but there is not really any theory supporting this

Leave-one-out shortcut

Recall that leave-one-out cross-validation is given by

$$\mathrm{CV}(\theta) = \frac{1}{n}\sum_{i=1}^{n}\big(y_i - \hat{f}_\theta^{-i}(x_i)\big)^2$$

where $\hat{f}_\theta^{-i}$ is the estimator fit to all but the ith training pair (x_i, y_i)

Suppose that our tuning parameter is $\theta = \lambda$ and $\hat{f}_\lambda$ is the ridge regression estimator

$$\hat{f}_\lambda(x_i) = x_i^T\hat{\beta}^{\mathrm{ridge}} = x_i^T(X^TX + \lambda I)^{-1}X^Ty$$

Then it turns out that

$$\frac{1}{n}\sum_{i=1}^{n}\big(y_i - \hat{f}_\lambda^{-i}(x_i)\big)^2 = \frac{1}{n}\sum_{i=1}^{n}\left[\frac{y_i - \hat{f}_\lambda(x_i)}{1 - S_{ii}}\right]^2$$

where $S = X(X^TX + \lambda I)^{-1}X^T$. This realization provides a huge computational savings!

Consider a linear fitting method $\hat{f}$, i.e., this is an estimator such that $\hat{y} = (\hat{f}(x_1), \dots \hat{f}(x_n))$ is given by

$$\hat{y} = Sy$$

for some matrix S. It turns out (Homework 5) that for many such estimators, we have the leave-one-out shortcut

$$\frac{1}{n}\sum_{i=1}^{n}\big(y_i - \hat{f}^{-i}(x_i)\big)^2 = \frac{1}{n}\sum_{i=1}^{n}\left[\frac{y_i - \hat{f}(x_i)}{1 - S_{ii}}\right]^2$$

For linear fitting methods, generalized cross-validation is defined as

$$\frac{1}{n}\sum_{i=1}^{n}\left[\frac{y_i - \hat{f}(x_i)}{1 - \mathrm{trace}(S)/n}\right]^2$$

This is a computationally efficient alternative to cross-validation (and it is used whether or not the leave-one-out shortcut holds)

Cross-validation for structured problems

In choosing the K folds at random, we are assuming, more or less, that the training pairs (x_i, y_i), $i = 1, \dots n$ are exchangeable

This is definitely not true in problems in which there is structure in the training inputs $x_1, \dots x_n$

E.g., suppose that the predictor measurements $x_1, \dots x_n$

- are taken across timepoints
- correspond to locations on a line
- correspond to pixel locations in an image
- are spatially related to one another

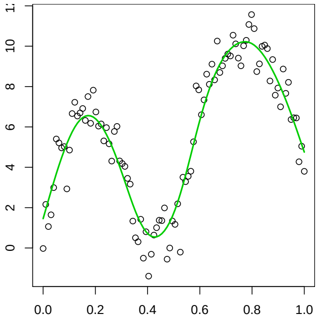
Cross-validation for such structured problems is usually tricky, and must be done with care

Example: cross-validation for smoothing splines

Here $x_i \in \mathbb{R}$, real-valued inputs. Recall that the smoothing spline estimator is

$$\hat{f}_\lambda = \underset{f}{\operatorname{argmin}} \sum_{i=1}^n \big(y_i - f(x_i)\big)^2 + \lambda \cdot \int \big(f''(x)\big)^2 dx$$

where λ is the tuning parameter



Here the training pairs are not really exchangeable, because our estimator $\hat{f}$ smooths across inputs x_i

So, e.g., it would be bad to have all $x_i \geq 0.8$ in one fold

One idea is to create folds that respect ordering: sort the inputs $x_1, \dots x_n$, and for K-fold cross-validation, take

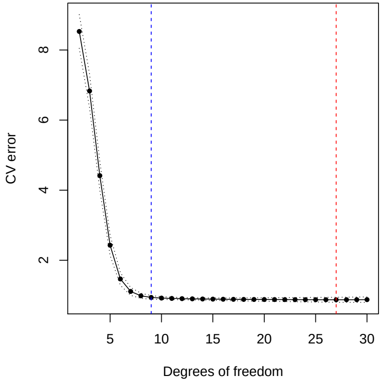
- Fold 1: points $1, K+1, 2K+1, \dots$
- Fold 2: points $2, K+2, 2K+2, \dots$
- $\dots$
- Fold K: points $K, 2K, 3K, \dots$

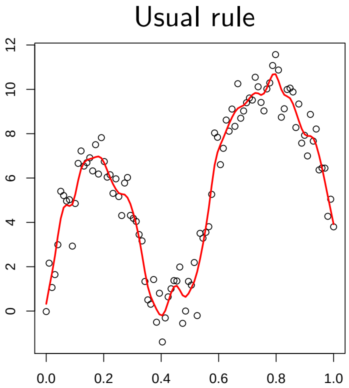
E.g., for 4-fold cross-validation:

ⓧ ① ② ③ ④ ① ② ③ ④ ① ② ⓧ

(We might also leave off the first and last points from any fold, and always include them in the training sets in each iteration of cross-validation)

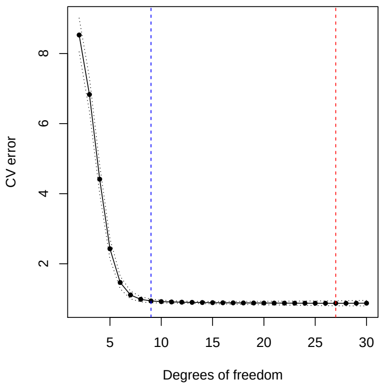
For our running smoothing spline example, we performed 5-fold cross-validation, in this structured way

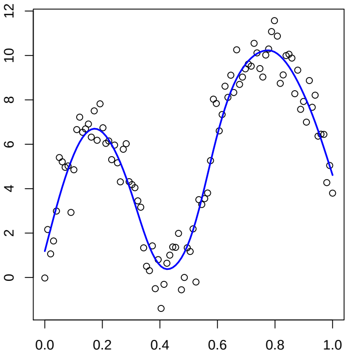




The usual rule selects a model with 27 degrees of freedom

One standard error rule





The one standard error rule selects a model with 9 degrees of freedom

Cross-validation is a general tool

So far we've looked at cross-validation for estimation under squared error loss, but it applies much more broadly than this

For an arbitrary loss $\ell(y_i, \hat{f}(x_i))$, the cross-validation estimate of prediction error under ℓ is

$$\frac{1}{n} \sum_{k=1}^{K} \sum_{i \in F_k} \ell\big(y_i, \hat{f}^{-k}(x_i)\big)$$

E.g., for classification, each $y_i \in \{0, 1\}$, and we might want to use the 0-1 loss

$$\ell\big(y_i, \hat{f}(x_i)\big) = \begin{cases} 0 & y_i = \hat{f}(x_i) \\ 1 & y_i \neq \hat{f}(x_i) \end{cases}$$

Cross-validation now gives us an estimate of misclassification error for a new observation. Usually in cross-validation for classification we try to balance the folds

Cross-validation alternatives

Cross-validation is a highly popular tool, but it's certainly not the only way to choose tuning parameters. There's also:

- The bootstrap
- Information criteria like AIC, BIC, RIC
- SURE (Stein's Unbiased Risk Estimate)
- SRM (Structural Risk Minimization)
- Stability-based methods
- Theoretically-guided choices (problem specific)

Chapter 7 of ESL provides a good review of most of these methods

Recap: model assessment, more cross-validation

Model assessment is the goal of estimating the prediction error of (typically a fixed) model. This is a different goal than model selection; e.g., if the cross-validation error curve has the same general shape as does the prediction error curve, we can do well at model selection, but unless it matches the values of the prediction error curve closely, we won't do well with model assessment

The bias-variance tradeoff is present in the choice of K for K-fold cross-validation. Typically $K = 5$ or $K = 10$ is a favorable choice

For problems in which the input points have structure, choosing the folds randomly may be a bad idea. Here cross-validation must be performed with care. E.g., for smoothing splines, we can form (nonrandom) folds based on the order of the inputs

There are any alternatives to cross-validation (but perhaps none as simple and broadly applicable)

Next time: classification

Now we only need to predict 0 or 1 … does this make the problem easier?