MS&E 226: "Small" Data

Lecture 5: In-sample estimation of prediction error (v1)

Ramesh Johari

ramesh.johari@stanford.edu

Estimating prediction error

The road ahead

Thus far we have seen how we can select and evaluate predictive models using the train-validate-test methodology. This approach works well if we have “enough” data.

What if we don’t have enough data to blindly train and validate models? We have to understand the behavior of prediction error well enough to intelligently explore the space of models.

The road ahead

Starting with this lecture:

- We develop methods of evaluating models using limited data.
- We develop measures of model performance that we can use to help us effectively search for "good" models.
- We characterize exactly how prediction error behaves through the ideas of *bias* and *variance*.

A word of caution: All else being equal, more data leads to more robust model selection and evaluation! So these techniques are not "magic bullets".

Estimating prediction error

We saw how we can estimate prediction error using validation or test sets.

But what can we do if we don't have enough data to estimate test error?

In this set of notes we discuss how we can use *in-sample* estimates to measure model complexity.

Two approaches:

- Cross validation
- Model scores

Cross validation

Cross validation

Cross validation is a simple, widely used technique for estimating prediction error of a model, when data is (relatively) limited.

Basic idea follows the train-test paradigm, but with a twist:

- Train the model on a subset of the data, and test it on the remaining data
- Repeat this with *different* subsets of the data

K-fold cross validation

In detail, K-fold cross validation (CV) works as follows:

- Divide data (randomly) into K equal groups, called *folds*. Let A_k denote the set of data points $(Y_i, \mathbf{X}_i)$ placed into the k'th fold.[1]
- For $k = 1, \ldots, K$, train model on all except k'th fold. Let $\hat{f}^{-k}$ denote the resulting fitted model.
- Estimate prediction error as:

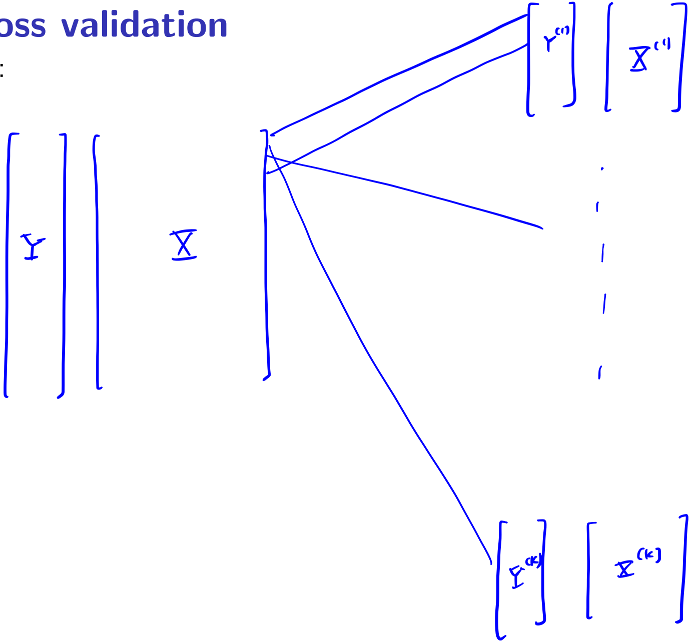
$$\mathsf{Err}_{\mathsf{CV}} = \frac{1}{K} \sum_{k=1}^{K} \left(\frac{1}{n/K} \sum_{i \in A_k} (Y_i - \hat{f}^{-k}(\mathbf{X}_i))^2 \right).$$

In words: for the k'th model, the k'th fold acts as a *validation set*. The estimated prediction error from CV $\mathsf{Err}_{\mathsf{CV}}$ is the average of the test set prediction errors of each model.

[1] For simplicity assume n/K is an integer.

K-fold cross validation

A picture:



Using CV

After running K-fold CV, what do we do?

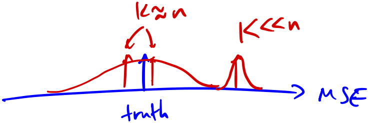
- We then build a model from *all* the training data. Call this $\hat{f}$.
- The idea is that Err_{CV} should be a good estimate of Err, the generalization error of $\hat{f}$.[2]

So with that in mind, how to choose K?

- If $K = \overset{n}{\cancel{N}}$, the resulting method is called *leave-one-out* (LOO) cross validation.
- If $K = 1$, then there is no cross validation at all.
- In practice, in part due to computational considerations, often use $K = 5$ to 10.

[2] Recall generalization error is the expected prediction error of $\hat{f}$ on new samples.

How to choose K?



There are two separate questions: how well Err_{CV} approximates the true error Err; and how sensitive the estimated error is to the training data itself.[3]

First: *How well does* Err_{CV} *approximate* Err*?*

- When $K = \overset{n}{\cancel{N}}$, the training set for each $\hat{f}^{-k}$ is nearly the entire training data.
  Therefore Err_{CV} will be nearly unbiased as an estimate of Err.
- When $K \ll \overset{n}{\cancel{N}}$, since the models use much less data than the entire training set, each model $\hat{f}^{-k}$ has higher generalization error; therefore Err_{CV} will tend to *overestimate* Err.

[3] We will later interpret these ideas in terms of concepts known as *bias* and *variance*, respectively.

How to choose K?

Second:: *How much does* Err_{CV} *vary if the training data is changed?*

- When $K = \overset{n}{\cancel{N}}$, because the training sets are very similar across all the models $\hat{f}^{-k}$, they will tend to have strong positive correlation in their predictions; in other words, the estimated Err_{CV} is very sensitive to the training data.
- When $K \ll \overset{n}{\cancel{N}}$, the models $\hat{f}^{-k}$ are less correlated with each other, so Err_{CV} is less sensitive to the training data.[4]

The overall effect is highly context specific, and choosing K remains more art than science in practice.

[4] On the other hand, note that each model is trained on significantly less data, which can also make the estimate Err_{CV} sensitive to the training data.

Leave-one-out CV and linear regression [$*$]

Leave-one-out CV is particularly straightforward for linear models fitted by OLS: there is no need to refit the model at all. This is a useful computational trick for linear models.

Theorem

Given training data $\mathbf{X}$ *and* $\mathbf{Y}$, *let* $\mathbf{H} = \mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top$ *be the* hat matrix, *and let* $\hat{\mathbf{Y}} = \mathbf{H}\mathbf{Y}$ *be the fitted values under OLS with the full training data.*

Then for leave-one-out cross validation:[5]

$$\text{Err}_{\text{LOOCV}} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{Y_i - \hat{Y}_i}{1 - H_{ii}}\right)^2 .$$

Interpretation: Observations with H_{ii} close to 1 are very "influential" in the fit, and therefore have a big effect on generalization error.

[5] It can be shown that $H_{ii} < 1$ for all i.

LOO CV and OLS: Proof sketch [$*$]

- Let $\hat{f}^{-i}$ be the fitted model from OLS when observation i is left out.
- Define $Z_j = Y_j$ if $j \neq i$, and $Z_i = \hat{f}^{-i}(\mathbf{X}_i)$.
- Show that OLS with training data $\mathbf{X}$ and $\mathbf{Z}$ has $\hat{f}^{-i}$ as solution.
- Therefore $\hat{f}^{-i}(\mathbf{X}_i) = (\mathbf{HZ})_i$.
- Now use the fact that:

$$(\mathbf{HZ})_i = \sum_j H_{ij} Z_j = (\mathbf{HY})_i - H_{ii} Y_i + H_{ii} \hat{f}^{-i}(\mathbf{X}_i).$$

A hypothetical example

- You are given a large dataset with many covariates. You carry out a variety of visualizations and explorations to conclude that you only want to use p of the covariates.
- You then use cross validation to pick the best model using these covariates.
- Question: is $\mathrm{Err}_{\mathrm{CV}}$ a good estimate of Err in this case?

A hypothetical example (continued)

No – You already used the data to choose your p covariates!

The covariates were chosen because they looked favorable on the training data; this makes it more likely that they will lead to low cross validation error.

Thus in this approach, $\mathrm{Err}_{\mathrm{CV}}$ will typically be *lower* than true generalization error Err.[6]

MORAL: To get unbiased results, any model selection must be carried out without the holdout data included!

[6] Analogous to our discussion of validation and test sets in the train-validate-test approach.

Cross validation in R

In R, cross validation can be carried out using the `cvTools` package.

```
> library(cvTools)
> cv.folds = cvFolds(n, K)
> cv.out = cvFit(lm, formula =  ...,
              folds = cv.folds, cost = mspe)
```

When done, `cv.out$cv` contains $\mathrm{Err}_{\mathrm{CV}}$. Can be used more generally with other model fitting methods besides `lm`.

Model scores

Model scores

A different approach to in-sample estimation of prediction error uses the following approach:

- Choose a model, and fit it using the data.
- Compute a *model score* that uses the sample itself to estimate the prediction error of the model.

By necessity, this approach works only for certain model classes; we show how model scores are developed for linear regression.

Training error

The first idea for estimating prediction error of a fitted model might be to look at the sum of squared error in-sample:

$$\mathsf{Err_{tr}} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{f}(\mathbf{X}_i))^2 = \frac{1}{n}\sum_{i=1}^{n}\hat{r}_i^2.$$

This is called the *training error*; it is the same as $1/n\times$ *sum of squared residuals* we studied earlier.

Training error vs. prediction error

$\begin{bmatrix} \mathbf{Y} \end{bmatrix} \quad \begin{bmatrix} \mathbf{X} \end{bmatrix}$

Of course, we should expect that training error is *too optimistic* relative to the error on a new test set: after all, the model was specifically tuned to do well on the training data.

To formalize this, we can compare $\mathsf{Err}_{\mathsf{tr}}$ to $\mathsf{Err}_{\mathsf{in}}$, the *in-sample prediction error*: → TERRIBLE TERMINOLOGY !!

$$\mathsf{Err}_{\mathsf{in}} = \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}[(Y - \hat{f}(\vec{X}))^2 | \mathbf{X}, \mathbf{Y}, \vec{X} = \mathbf{X}_i].$$

This is the prediction error if we received new samples of Y corresponding to each covariate vector in our existing data.[7]

[7] The name is confusing: "in-sample" means that it is prediction error on the covariate vectors $\mathbf{X}$ already in the training data; but note that this measure is the expected prediction error on *new* outcomes for each of these covariate vectors.

In-sample prediction error

Interpreting in-sample prediction error:

$$\mathsf{Err}_{\mathsf{in}} = \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}[(Y-\hat{f}(\vec{X}))^2|\mathbf{X},\mathbf{Y},\vec{X}=\mathbf{X}_i].$$

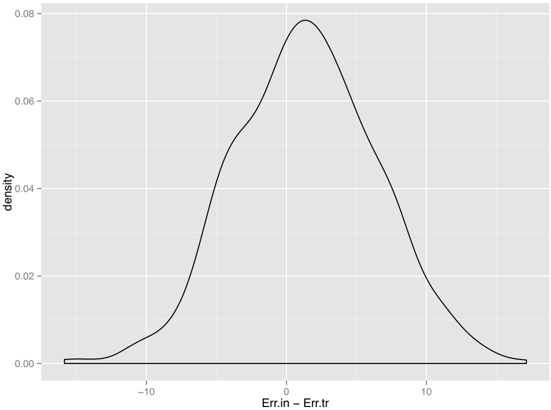
Training error vs. test error

Let's first check how these behave relative to each other.

- Generate 100 $X_1, X_2 \sim N(0, 1)$, i.i.d.
- Let $Y_i = 1 + X_{i1} + 2X_{i2} + \varepsilon_i$, where $\varepsilon_i \sim N(0, 5)$, i.i.d.
- Fit a model $\hat{f}$ using OLS, and the formula `Y ~ 1 + X1 + X2`.
- Compute training error of the model. ← mean of squared residuals.
- Generate another 100 *test samples* of Y corresponding to each row of $\mathbf{X}$, using the same population model.
- Compute in-sample prediction error of the fitted model on the test set.
- Repeat this process 500 times, and create a plot of the results.

Training error vs. test error

Results:



Mean of $\text{Err}_{\text{in}} - \text{Err}_{\text{tr}} = 1.42$; i.e., training error is underestimating in-sample prediction error.

Training error vs. test error

If we could somehow *correct* $\mathsf{Err}_{\mathsf{tr}}$ to behave more like $\mathsf{Err}_{\mathsf{in}}$, we would have a way to estimate prediction error on new data (at least, for covariates $\mathbf{X}_i$ we have already seen).

Here is a key result towards that correction.[8]

Theorem

$$\mathbb{E}[\mathsf{Err}_{\mathsf{in}}|\mathbf{X}] = \mathbb{E}[\mathsf{Err}_{\mathsf{tr}}|\mathbf{X}] + \frac{2}{n}\sum_{i=1}^{n} \overbrace{\mathrm{Cov}(\hat{f}(\mathbf{X}_i), Y_i|\mathbf{X})}^{\text{the randomness is in } \mathbf{Y} \text{ (part of our training data)}}.$$

In particular, if $\mathrm{Cov}(\hat{f}(\mathbf{X}_i), Y_i|\mathbf{X}) > 0$, then training error underestimates test error.

[8] This result holds more generally for other measures of prediction error, e.g., 0-1 loss in binary classification.

Training error vs. test error: Proof [$*$]

Proof: If we expand the definitions of $\mathsf{Err}_{\mathsf{tr}}$ and $\mathsf{Err}_{\mathsf{in}}$, we get:

$$\begin{aligned}\mathsf{Err}_{\mathsf{in}} - \mathsf{Err}_{\mathsf{tr}} = \frac{1}{n}\sum_{i=1}^{n} \Big(&\mathbb{E}[Y^2|\vec{X} = \mathbf{X}_i] - Y_i^2 \\ &- 2(\mathbb{E}[Y|\vec{X} = \mathbf{X}_i] - Y_i)\hat{f}(\mathbf{X}_i)\Big)\end{aligned}$$

Now take expectations over $\mathbf{Y}$. Note that:

$$\mathbb{E}[Y^2|\mathbf{X}, \vec{X} = \mathbf{X}_i] = \mathbb{E}[Y_i^2|\mathbf{X}],$$

since both are the expectation of the square of a random outcome with associated covariate $\mathbf{X}_i$. So we have:

$$\mathbb{E}[\mathsf{Err}_{\mathsf{in}} - \mathsf{Err}_{\mathsf{tr}}|\mathbf{X}] = -\frac{2}{n}\sum_{i=1}^{n} \mathbb{E}\Big[(\mathbb{E}[Y|\vec{X} = \mathbf{X}_i] - Y_i)\hat{f}(\mathbf{X}_i)\Big|\mathbf{X}\Big].$$

Training error vs. test error: Proof [$*$]

Proof (continued): Also note that $\mathbb{E}[Y|\vec{X} = \mathbf{X}_i] = \mathbb{E}[Y_i|\mathbf{X}]$, for the same reason. Finally, since:

$$\mathbb{E}[Y_i - \mathbb{E}[Y_i|\mathbf{X}]|\mathbf{X}] = 0,$$

we get:

$$\begin{aligned}\mathbb{E}[\mathsf{Err}_{\mathsf{in}} - \mathsf{Err}_{\mathsf{tr}}|\mathbf{X}] = \frac{2}{n}\sum_{i=1}^{n}\Big(&\mathbb{E}\Big[(Y_i - \mathbb{E}[Y|\vec{X} = \mathbf{X}_i])\hat{f}(\mathbf{X}_i)\Big|\mathbf{X}\Big] \\ &- \mathbb{E}[Y_i - \mathbb{E}[Y_i|\mathbf{X}]|\mathbf{X}]\mathbb{E}[\hat{f}(\mathbf{X}_i)\big|\mathbf{X}]\Big),\end{aligned}$$

which reduces to $(2/n)\sum_{i=1}^{n}\mathrm{Cov}(\hat{f}(\mathbf{X}_i), Y_i|\mathbf{X})$, as desired.

The theorem's condition

What does $\mathrm{Cov}(\hat{f}(\mathbf{X}_i), Y_i|\mathbf{X}) > 0$ mean?

In practice, for any "reasonable" modeling procedure, we should expect our predictions to be positively correlated with our outcome.

Example: Linear regression

Assume a linear population model $Y = \vec{X}\boldsymbol{\beta} + \varepsilon$, where $\mathbb{E}[\varepsilon|\vec{X}] = 0$, $\mathrm{Var}(\varepsilon) = \sigma^2$, and errors are uncorrelated.

Suppose we use a subset S of the covariates and fit a linear regression model by OLS. Then:

$$\sum_{i=1}^{n} \mathrm{Cov}(\hat{f}(\mathbf{X}_i), Y_i|\mathbf{X}) = |S|\sigma^2.$$

In other words, in this setting we have:

$$\mathbb{E}[\mathsf{Err}_{\mathsf{in}}|\mathbf{X}] = \mathbb{E}[\mathsf{Err}_{\mathsf{tr}}|\mathbf{X}] + \frac{2|S|}{n}\sigma^2.$$

A model score for linear regression

The last result suggests how we might estimate in-sample prediction error for linear regression:

- Estimate σ^2 using the sample standard deviation of the residuals on the full fitted model, i.e., with $S = \{1, \dots, p\}$; call this $\hat{\sigma}^2$.[9]
- For a given model using a set of covariates S, compute:

$$C_p = \mathsf{Err}_{\mathsf{tr}} + \frac{2|S|}{n}\hat{\sigma}^2.$$

This is called *Mallow's C_p statistic*. It is an estimate of the prediction error.

[9] Informally, the reason to use the full fitted model is that this should provide the best estimate of σ^2.

A model score for linear regression

$$C_p = \mathsf{Err}_{\mathsf{tr}} + \frac{2|S|}{n}\hat{\sigma}^2.$$

How to interpret this?

- The first term measures fit to the existing data.
- The second term is a penalty for *model complexity*.

So the C_p statistic balances underfitting and overfitting the data; for this reason it is sometimes called a *model complexity score*.

(We will later provide conceptual foundations for this tradeoff in terms of *bias* and *variance*.)

AIC, BIC

Other model scores:

- *Akaike information criterion* (AIC). In the linear population model with *normal* ε, this is equivalent to:

$$\frac{n}{\hat{\sigma}^2}\left(\mathsf{Err}_{\mathsf{tr}} + \frac{2|S|}{n}\hat{\sigma}^2\right).$$

- *Bayesian information criterion* (BIC). In the linear population model with normal ε, this is equivalent to:

$$\frac{n}{\hat{\sigma}^2}\left(\mathsf{Err}_{\mathsf{tr}} + \frac{|S|\ln n}{n}\hat{\sigma}^2\right).$$

Both are more general, and derived from a *likelihood* approach. (More on that later.)

AIC, BIC

Note that:

- AIC is the same (up to scaling) as C_p in the linear population model with normal ε.
- BIC penalizes model complexity more heavily than AIC.

AIC, BIC in software [$*$]

In practice, there can be significant differences between the actual values of C_p, AIC, and BIC depending on software; but these don't affect model selection.

- The estimate of sample variance $\hat{\sigma}^2$ for C_p will usually be computed using the full fitted model (i.e., with all p covariates), while the estimate of sample variance for AIC and BIC will usually be computed using just the fitted model being evaluated (i.e., with just $|S|$ covariates). This typically has no substantive effect on model selection.
- In addition, sometimes AIC and BIC are reported as the *negation* of the expressions on the previous slide, so that larger values are better; or without the scaling coefficient in front. Again, none of these changes affect model selection.

Comparisons

Simulation: Comparing C_p, AIC, BIC, CV

Repeat the following steps 10 times:

- For $1 \leq i \leq 100$, generate $X_i \sim$ uniform$[-3, 3]$.
- For $1 \leq i \leq 100$, generate Y_i as:

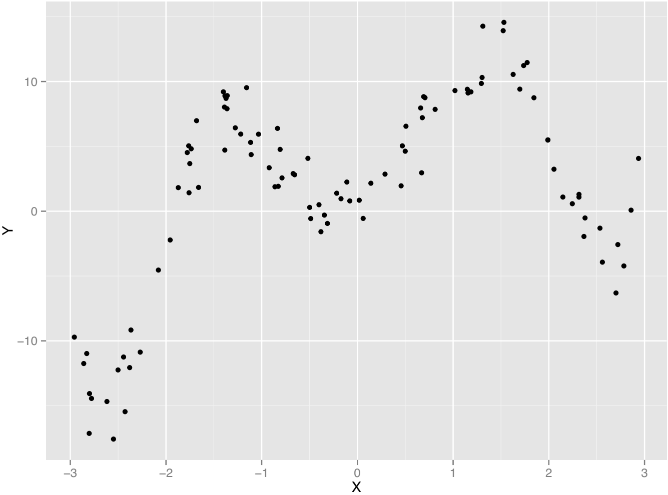
$$Y_i = \alpha_1 X_i + \alpha_2 X_i^2 - \alpha_3 X_i^3 + \alpha_4 X_i^4 - \alpha_5 X_i^5 + \alpha_6 X_i^6 + \varepsilon_i,$$

  where $\varepsilon_i \sim$ uniform$[-3, 3]$.
- For $p = 1, \ldots, 20$, we evaluate the model `Y ~ 0 + X + I(X^2) + ... + I(X^p)` using C_p, BIC, and 10-fold cross validation.[10]

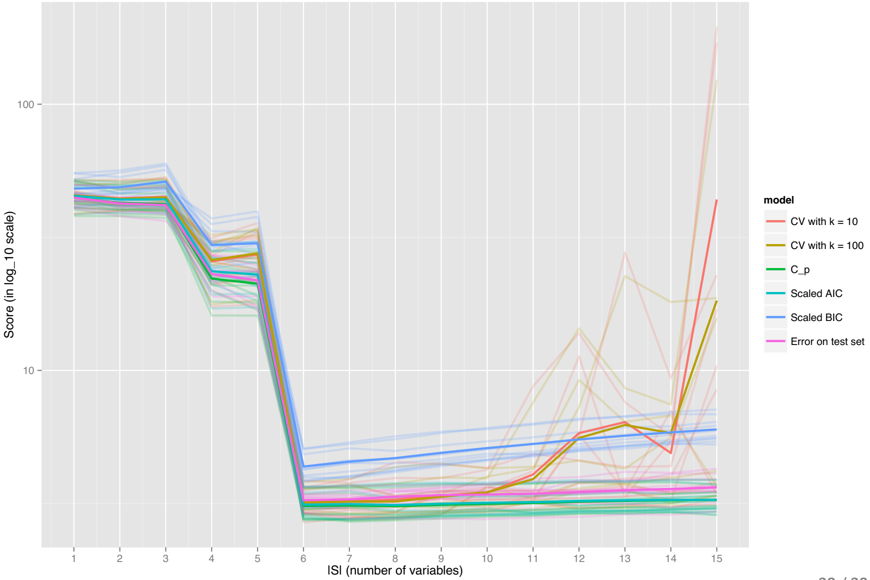
How do these methods compare?

[10] We leave out AIC since it is exactly a scaled version of C_p.

Simulation: Visualizing the data



Simulation: Comparing C_p, AIC, BIC, CV



Score (in log_10 scale)
100
10
|S| (number of variables)
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
model
CV with k = 10
CV with k = 100
C_p
Scaled AIC
Scaled BIC
Error on test set