MS&E 226: Fundamentals of Data Science

Lecture 4: Cross validation

Ramesh Johari
rjohari@stanford.edu

Estimating generalization error

Remember the overall goal in prediction:

We want to be able to pick models that have low *generalization error*, i.e., that make good predictions on average on new samples from the population.

In this lecture, we study *cross validation*, a widely used method for estimation of generalization error.

Train-test

Suppose you have a *learning algorithm* that takes as input a training data set and produces as output a fitted model $\hat{f}$.

Recall, you should think of this as code that produces, e.g., a fitted linear regression, lasso regression, ridge regression, etc.

Example: The R call `fm = lm(data = input, formula = Y ~ 1 + X1 + X2)` takes as input the data frame `input`, and produces as output a fitted model `fm` (i.e., the linear regression coefficients), using the two covariates `X1` and `X2` to predict the outcome.

Train-test

We could evaluate this learning algorithm using the train-test paradigm:

1. Split the dataset into a training set (say 80%) and test set (say 20%).
2. Train using the given learning algorithm on the training set to produce a *fitted model* $\hat{f}$, and evaluate $\hat{f}$ on the test set (as in last lecture).

Note no validation step here, since we only have one learning algorithm (we are not "picking a winner").

Why stop there?

But why stop there? We could do this training-test split multiple times with the same data!

That's the essential idea of *cross validation*:

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

It allows us to use our data more efficiently to evaluate a learning algorithm.

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 (fit a model) using learning algorithm on all except k'th fold. Let $\hat{f}^{-k}$ denote the resulting fitted model.
- Estimate prediction error as:

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

In words: for the k'th model, the k'th fold acts as a *test set*. The estimated prediction error from CV Err_{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 $\mathrm{Err}_{\mathrm{CV}}$ should be a good estimate of the generalization error of $\hat{f}$.[2]

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

Impacts of the choice of K

First impact: Overestimation of generalization error for small K.

- Ultimately we train a model $\hat{f}$ on the whole dataset, and $\mathrm{Err}_{\mathrm{CV}}$ is supposed to estimate the generalization error of this model.
- But for any K that is much smaller than n, the models in CV are trained on much less data than the entire data set, and so each $\hat{f}^{-k}$ should have higher generalization error than $\hat{f}$ on average.
- This means if K is small, $\mathrm{Err}_{\mathrm{CV}}$ tends to overestimate the true generalization error.

Impacts of the choice of K

Second impact: Sensitivity to the data for small K.

- For small K, because each of our models $\hat{f}^{-k}$ are being trained on less data, they are also much more *sensitive* to that data.
- This shows up as a greater amount of *variability* in their generalization performance.
- That in turn means an estimate Err_{CV} that is less precise as an estimate of generalization error.

Impacts of the choice of K

Third impact: Sensitivity to the data for large K, especially with small data sets.

- If K is large, then all the models $\hat{f}^{-k}$ are going to be very similar to each other, because they are trained on very similar data sets.
- In this case, the estimate $\mathrm{Err}_{\mathrm{CV}}$ will be highly sensitive to the realization of the data set.

Choosing K in practice

These impacts are highly context-specific, and their interaction with each other is not well understood overall.

In practice, the choice of K is typically driven by computational considerations; $K = 5$ to 10 is common.

$K = n$ is called *leave-one-out* (LOO) cross validation; this is typically computationally prohibitive, but has an efficient implementation available for linear regression (see appendix).

CV for model selection

Cross validation can also be used for selecting a winner among multiple learning algorithms:

1. Use cross validation to get an estimate of generalization error for *each* learning algorithm.
2. Pick the learning algorithm with the best performance (the “winner”).
3. Apply that learning algorithm on the entire data set to obtain a fitted model $\hat{f}$.

(This use of cross validation replaces the “validation” step in the train-validate-test paradigm.)

Model selection: 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 a small subset 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 the generalization error of your chosen model?

A hypothetical example (continued)

No – You already used the data to choose your 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, Err_{CV} will typically *underestimate* the prediction performance of your chosen model.[3]

***Moral*: To get trustworthy results, any model selection must be carried out without the holdout data included!**

[3]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 for linear regression 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 Err_{CV}. Can be used more generally with other model fitting methods besides `lm`.

Specific packages (e.g., `glmnet`) often include their own CV routines; make sure you understand what they are doing!

An alternative to CV: 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. (Such scores can then be used to select among competing alternative fitted models.)

Examples of model scores include C_p, AIC (the Akaike information criterion), and BIC (the Bayesian information criterion).

These scores and their underlying theory are discussed in detail in the optional lecture notes on “Model Scores” and “Model Selection Using Model Scores”.

Comparing train-test with CV

Comparing train-test with CV

Train-test vs. CV

If we are given a data set $\mathbf{X}, \mathbf{Y}$, we now have two ways to estimate generalization error:

- Train then test
- Cross validation

To make things simple, suppose we are comparing an 80%-20% train-test split to 5-fold CV. How do we choose between them?

Which $\hat{f}$?

Suppose the model of interest is *the fitted model* $\hat{f}$ *obtained by training on all the data*.

This is often the case in production systems, e.g., recommendation systems, etc.; even if train-test separation is used, in the end the model implemented in production will be the model trained on all the data.

In this case, note that 5-fold CV implements the same thing as the 80%-20% train-test split, but just does it 5 times. This is always going to yield a more precise estimate of generalization error than just train-test.

Therefore, if the goal is to estimate the generalization error of a model that is fit on all the data, then CV will always provide the best estimate. The only restriction in this case to whether CV is used or not is computational: it may be prohibitive to run CV if the modeling strategy is quite complex.

Which $\hat{f}$?

On the other hand, suppose it is known in advance that the model of interest is one fit on only 80% of the data, and the 20% test set must be held out for model evaluation.

This is the case, e.g., in Kaggle competitions: Kaggle always maintains a held out test set that is not available to data scientists as they build their predictive models.

In this case, the test error on the 20% test set has the advantage that it will be an unbiased estimate of the true generalization error (as compared to 5-fold CV run on the training dataset).

Appendix: Leave-one-out CV for linear regression [$*$]

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:*[4]

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

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

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