Cross-Validation Werks

Jean Feng*
Department of Biostatistics, University of Washington
and
Noah Simon
Department of Biostatistics, University of Washington

August 12, 2016

Abstract

In the setting of penalized regression, cross-validation is a widely used technique for tuning penalty parameters. With an oracle set of parameters, one can guarantee a particular rate of convergence of the prediction error, but it is unknown if cross-validation is able to recover the same rate. We prove that the model chosen from cross-validation will converge to the true model at the optimal rate since it converges the oracle at a near-parametric rate $(J(c + \kappa \log n)/n)^{1/2}$ where n is the number of samples and J is the number of penalty parameters. The results are counter to the common belief that increasing the number of penalty parameters drastically increase the model complexity. In fact, for nonparametric models, our error bounds allow the number of penalty parameters to increase with the number of samples while retaining the optimal rate. The proof allows cross-validation over an infinite set of penalty parameters and the lower limit of the range can decrease at any polynomial rate. For smooth regression problems, the proof only requires convexity of the loss and penalty functions; additional assumptions are required if the penalty functions are non-smooth. The proof uses techniques from entropy and an implicit differentiation trick. The simplicity of the proof may extend itself to other problems in cross-validation. Our simulation studies show that increasing the penalty parameters can substantially decrease model bias if one uses optimization algorithms that effectively minimize the validation loss.

Keywords: ...?

*Jean Feng was supported by NIH grants DP5OD019820 and T32CA206089. Noah Simon was supported by NIH grant DP5OD019820. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.

1 Introduction

What is regression. When problem is ill-posed, we use penalization. Per the usual regression setting, we have data (x, y) where x are the covariates and y is the response. The goal of model estimation is to minimize the prediction error of y, which we will presume to be the least squares loss. If the problem is ill-posed or high-dimensional ($p >> n$), one needs to balance the "bias" and "variance" of the model in order to obtain good generalization error. A popular technique is to use regularization, also known as penalization. There one introduces penalty functions into the model criterion to control model complexity and induce desired structure. The most common examples of regularization methods include the ridge penalty, lasso, and etc etc etc. Every penalty function is accompanied by a weight parameter that indicates how strongly the structure corresponding to that penalty should be enforced.

Tuning parameters are important. There is an optimal lambda, but we don't know what it is. The penalty parameters ultimately determine the fitted model, so it is important to select these parameters properly. When the penalty parameters are too small, fitted models have large variance. On the other hand, when the penalty parameters are too large, the fitted models have large bias. The oracle penalty parameters balance these two tradeoffs to guarantee fast convergence rates (Van de geer-additive models paper, Wahba-smoothing spline paper, and others?). In fact, in the case of an additive model $f = \sum f_i$, one can guarantee fast convergence rates for each of the components if the parameters are chosen appropriately – large parameters for penalties on low complexity components and small parameters for penalties on high complexity components. For appropriately chosen penalty parameters, the convergence rates of each f_i are generally as fast as in the case where the other components are known. However, these oracle values depend on unknown values, such as the complexity of the true model and the magnitude of the noise variables.

Hence we do cross-validation Since the oracle penalty parameter values are unknown, one common approach is to use K-fold cross-validation (CV). (Or explain hold-out first and then move to CV? In hold-out, one randomly splits the data into two sets. Models are trained on one partition of the data and its generalization error is estimated on the other half. The algorithm then chooses the model with the minimum estimated risk.) K-fold CV first splits the data into K partitions. Then for each penalty parameter value, we train K

models by training over $K-1$ partitions and evaluate it over the dropped partition. CV chooses the penalty parameter values that minimize the average validation loss. K-fold CV is very popular due to two main reasons: one, CV can be applied to almost any algorithm in almost any framework since it only assume that the data are independent, and two, it has been shown to be highly effective in practice. Hence, many authors have recommended using CV to tune parameters in penalized regression problems (e.g. lasso, elastic net, etc).

Past work in CV - and their drawbacks? Given the popularity of CV, there has been a lot of work. Arlot (cite) presents a very comprehensive review of CV, hence we will only review CV in the context of model estimation. Most of the research seems to occur at the two ends of the spectrum – super general or super specific. **General approaches:** Van Der Laan (2003, 2004) establishes finite and asymptotic sample oracle inequalities for a general CV framework and proves the asymptotic optimality of CV as long as the ratio between the size of the folds and the total dataset stays away from zero. Mitchell uses entropy methods to prove oracle inequalities for CV using entropy methods, but when applied to the Lasso, it requires assumptions on the design matrix that are often unrealistic. **Specific (penalized) regression approaches:** Gyorfi (2002) provides a finite sample inequality for holdout for least squares. Wegkamp (2003) proved an oracle inequality for a penalized least squares holdout procedure (our inequality bound has faster convergence). In the realm of penalized regression models, Golub, Heath and Wahba showed that generalized CV chooses a good ridge parameter and Chetverikov gives convergence rates for the CV-retrained lasso. We were unable to find theoretical proofs for convergence rates for cross-validated models fit using multiple penalty parameters.

Previous work don't address infinite number of penalty parameters, but it's a relevant problem! All of the previously mentioned CV proofs assume that one searches over a finite number of penalty parameters. Previously this was a reasonable assumption since we only had to tune one or two penalty parameters and the common approach was to use grid search. However, the machine learning literature (and our previous paper) are increasingly interested in tuning multiple hyperparameters. Grid search is clearly too slow in this case, so the approach here is to treat tuning hyperparameters as a continuous optimization approach (Bengio, Foo, Feng, MacLaurin, Snoek). Given that the set of hyperparameters to test are

not predefined and are instead defined at each iteration, the question is now what happens when there are an infinite number of penalty parameters.

Our paper proves CV works for penalized regression with mulitple penalty parmaeters even after testing an infinite number of penalty parameters. Here are the important contributions. Our paper addresses CV in the penalized regression setting and provides a finite sample upper bound for the prediction error of a model selected by CV. We find that this CV-selected model converges towards the oracle model at a near-parametric rate. Hence for most nonparametric problems, the convergence rate of the CV model to the true model is dominated by convergence rate of oracle. This inequality bound also provides insight into how to improve model estimation. The error bounds suggest that one way to reduce generalization error is to increase the number of penalty parameters with the sample size, contrary to popular belief. However, it also gives an upper bound for the number of penalty parameters to tune. Unlike many of the previous methods, we allow CV to be performed over an infinite (possibly uncountable) number of penalty parameters and (?) allow the lower bound on the penalty parameters to decrease at any polynomial rate with respect to the number of samples.

Proof techniques? Do people want to read about this? Also proof assumptions? The proof is based on entropy methods (sara's book) and an implicit differentiation trick (bengio, Foo, Feng). We assume that the loss and penalty functions are convex and smooth. If the penalty is non-smooth but smooth almost everywhere, such as in the Lasso, further assumptions are needed on the behavior of the the directional derivatives and the local optimality space. The assumptions here are minimal. **Important assumption that we dont address: we can't find minimum of nonconvex problems...** The problem is not convex in the penalty parameters, so finding the minimizer of the validation loss is very difficult. So how? Will we just assume this is true, but tel you that minimizing is a good thing? (What if we don't find the minimizer of the validation error or the training error. Does this proof fail? Does this method fail? Is this a minimal assumption.)

Paper Organization Section 1 provides the theorem. Section 2 applies our technique to various problems. Section 3 provides simulation studies. Section 4 is a discussion and bids you farewell. Section 5 gives the hairy proof details.

2 Main Results

2.0.1 Problem Setup

Per the usual regression framework, we observe response y_i and p predictors x_i. Suppose y_i is generated from the true model g^* from model class $\mathcal{G}$

$$y_i = g^*(x_i) + \epsilon_i \tag{1}$$

where ϵ_i are independent sub-Gaussian random variables. Given n observations, we estimate the model by K-fold cross-validation.

Explain cross-validation. Tuning the penalty parameter values by K-fold cross-validation is essentially an optimization problem over the penalty parameter space. The procedure begins by randomly splitting the dataset into K partitions denoted D_{-k} for $k = 1, ..., K$. Then for a given penalty parameter value, one fits k models that minimize the penalized training criterion. The "goodness" of the penalty parameter values is then evaluated by the average validation loss. We can formulate this as a joint optimization problem, where the training step is the inner optimization problem and the tuning step is the outer optimization problem.

$$\hat{\lambda} = \arg\min_{\lambda \in \Lambda} \frac{1}{2} \sum_{k=1}^{K} \|y - \hat{g}_\lambda(\cdot|D_{-k})\|_k^2 \tag{2}$$

$$\hat{g}(\lambda|D_{-k}) = \arg\min_{g \in \mathcal{G}} \frac{1}{2}\|y - g\|_{-k}^2 + \sum_{j=1}^{J} \lambda_j P_j^{v_j}(g) \tag{3}$$

The final model produced by K-fold cross-validation is one that is trained over the entire dataset

$$\hat{g}(\hat{\lambda}|D) = \arg\min_{g \in \mathcal{G}} \frac{1}{2}\|y - g\|_D^2 + \sum_{j=1}^{J} \hat{\lambda}_j P_j^{v_j}(g) \tag{4}$$

Present the Cross-validation Theorem (and assumptions) We have the following finite-sample size result.

Present the most interesting consequences of these results

- The proofs shows that retraining the model over all the data is similar to just taking an average of the k trained models. The rates of convergence are the same for both models.

- It doesn't cost much to add penalty parameters. Most people believed that adding more parameters increased the model complexity drastically, but it doesn't. One does not overfit even when tuning over an infinite number of possible penalty parameter values.

- Intuitively, this makes sense since we are technically tuning over a finite-dimensional parameter space. One would hope that when tuning the penalty parameters, we would be attaining a parametric rate. This is not quite a parametric rate since there is a $\log n$ term, but it is very close.

Comparisons to other Cross-validation bounds

- Are our rates are the same - great we match Van Der Laan

- Are we letting lambda go to zero faster than other approaches - we allow faster shrinkage of lambda compared to chet and same lambda shrink rate as chaterjee.

- We assume fewer things?

Explain the main proof ideas

The proof is pretty general. It utilizes empirical process theory and we generalize some standard results. One could probably use this as a standard recipe to proving other cross-validation results.

The proof relies on a couple of key ideas. First, instead of addressing the original problem, we consider a slightly perturbed version by adding a small ridge penalty. For this perturbed problem, we find that the functions $\hat{g}_{\hat{\lambda}}(\cdot|D_{-k})$ are actually "Lipschitz" in λ. Next, it shows that the rate of convergence the model from cross-validation is bounded by the rate of convergence of models $\hat{g}_{\hat{\lambda}}(\cdot|D_{-k})$.

Explain and prove the Lipschitz property (maybe not even the entropy result
For smooth and many nonsmooth penalties, one can show that $\hat{g}_{\hat{\lambda}}(\cdot|D_{-k})$ is "Lipschitz" in λ. The most important implication is that this gives us the covering number/entropy for the model class, which in turn allows us to bound the empirical process. We prove the Lipschitz property below. For nonsmooth penalties, one needs more assumptions - refer to the appendix.

Corollary: Entropy Result for g_λ Perhaps people might be interested. Not sure.

Empirical Process theory tools, generalized? I'm not sure if this is worth putting here at all. Should we just hide it in the appendix?

3 Examples

If the reader is not convinced by the applicability of our results, we could show them a list of examples. Lasso. Sobolev penalty.

4 Simulations

5 Discussion

6 The actual proof + Lemmas