Lecture 6: Model Assessment and Selection

Statistical Learning and Data Mining

Xingye Qiao

Department of Mathematical Sciences
Binghamton University

E-mail: qiao@math.binghamton.edu

Read: ESLII Ch. 7

Outline

The next section would be . . . . . .

Model Performance Evaluation

How do we evaluate the performance of $\hat{f}$? Consider

- its generalization performance
- its prediction capability on an independent set of data, the so-called test data

Model performance evaluation is extremely important, as it

- measures the quality of the final model.
- allows us to compare different methods and make an optimal choice.
- guides the choice of tuning parameters in regularization methods.

Model Selection and Assessment

- **Model selection**: estimating the performance of different models and then choosing the best one.
- **Model assessment**: having chosen a final model, estimating its prediction error (generalization error) on new data.

Key to both: estimate the generalization error (test error)

Model Selection

- Selection from families of models: Ordinary linear regression, Lasso, Ridge Regression, etc.
- Within the same family of model, ...
  - Select the best subset of variables (i.e. variable selection)
  - Choose the best tuning parameter
  - Sometimes these goals coincide.
- The gold standard in regression? Mean Squared Error, where "mean" is wrt. the distribution of underlying population.
- For general problems, test error (generalization error)
- In other words, the criterion is based on how the model will work on the outside real-world data (independent of the training data).

Optimism of the Training Error Rate

- Training error: the same data is used to fit the method and to evaluate its performance.
- For many different loss funs. (squared error, 0-1, entropy, etc.)

$$\mathrm{E}(\mathrm{TestError}) = \mathrm{E}(\mathrm{TrainingError}) + \underbrace{\frac{2}{n}\sum_{i=1}^{n} \mathrm{Cov}(\hat{y}_i, y_i)}_{\text{Expected optimism}}$$

Ex. 7.4 in ESLII.

- The harder we try to fit the data, the more the training error underestimates the test error.
- **One MUST NOT rely on the training error only to select model.**

$$\mathrm{E}(\mathrm{TestError}) = \mathrm{E}(\mathrm{TrainingError}) + \underbrace{\frac{2}{n}\sum_{i=1}^{n}\mathrm{Cov}(\hat{y}_i, y_i)}_{\text{Expected optimism}}$$

The second term on the RHS is the theoretical foundation of the effective degrees of freedom, defined as

$$df = \frac{1}{\sigma^2}\sum_{i=1}^{n}\mathrm{Cov}(\hat{y}_i, y_i)$$

Therefore

$$\mathrm{TestError} \approx \mathrm{TrainingError} + \frac{2 \cdot df}{n}\hat{\sigma}^2$$

Many classical selection criteria take this general form

The next section would be ......

Akaike information criterion

$$\text{TestError} \approx \text{TrainingError} + \frac{2 \cdot df}{n}\hat{\sigma}^2$$

Using the negative log-likelihood as the loss function, one can show

$$-\frac{2}{n}\log(P_{\hat{\theta}}(\mathcal{D}_{test})) \approx -\frac{2}{n}\log(P_{\hat{\theta}}(\mathcal{D}_{training})) + \frac{2 \cdot df}{n}$$

Hence we define

$$\text{AIC} = -\frac{2}{n}\log(\text{likelihood}) + \frac{2 \cdot d}{n},$$

which, in Gaussian error linear regression setting (with σ^2 assumed known), is simplified to

$$\text{AIC} = \frac{1}{n\sigma^2} RSS_{training} + \frac{2 \cdot d}{n},$$

degrees of freedom

Here we used a trick that $df = \frac{1}{\sigma^2} \sum_{i=1}^{n} \text{Cov}(\hat{y}_i, y_i)$ is d.

- This holds
  - exactly for linear models with additive errors + squared error loss, and
  - approximately for linear models + log-likelihoods.
- But this does not hold for 0-1 loss generally.
- Recall that for a linear fitting model $\hat{\boldsymbol{y}} = \mathbf{S}\boldsymbol{y}$, we have $df = \text{Trace}(\mathbf{S})$.
  - This includes linear regression, ridge regression and smoothing spline.
- Recall that for lasso and elastic net, the number of nonzeros is an unbiased estimate of d.f.

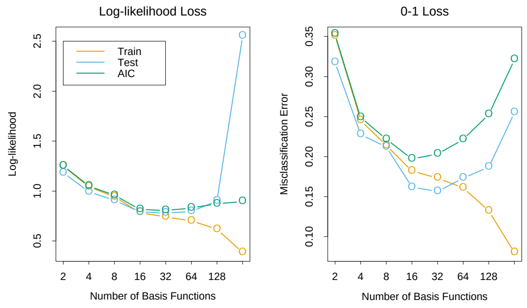

FIGURE 7.4. *AIC used for model selection for the phoneme recognition example of Section 5.2.3. The logistic regression coefficient function* $\beta(f) = \sum_{m=1}^{M} h_m(f)\theta_m$ *is modeled as an expansion in* M *spline basis functions. In the left panel we see the AIC statistic used to estimate* $\mathrm{Err}_{\mathrm{in}}$ *using log-likelihood loss. Included is an estimate of* Err *based on an independent test sample. It does well except for the extremely over-parametrized case* ($M = 256$ *parameters for* $N = 1000$ *observations). In the right panel the same is done for 0–1 loss. Although the AIC formula does not strictly apply here, it does a reasonable job in this case.*

Bayesian information criterion

$$\mathrm{AIC} = -\frac{2}{n}\log(\text{likelihood}) + \frac{2 \cdot d}{n},$$
$$\mathrm{BIC} = -2\log(\text{likelihood}) + \log(n) \cdot d$$

For Gaussian linear regression model with squared error loss and σ^2 assumed known,

$$\mathrm{AIC} = \frac{1}{n\sigma^2} RSS_{training} + \frac{2 \cdot d}{n},$$
$$\mathrm{BIC} = n[\frac{1}{n\sigma^2} RSS_{training} + \frac{\log(n) \cdot d}{n}],$$

Drawback of BIC

- BIC is motivated by the Bayesian approach to model selection
- Each potential model was assigned a prior probability.
- Key in BIC: all the models have the same prior probability (uniformly distributed.)
- This may not be reasonable with large p.
- E.g. Out of 10 variables, there are more models with 5 variables (252) than models with only 3 variables (45).
- If p is large, it is more like to select a large model (hence, unfair for small models).

Extended BIC

- The prior for selecting any size-d model is $\tau(d)^{\xi}$, where $\tau(d)$ is the number of size-d models (which is p choose d), and $\xi < 1$
- The chance for select a particular size-d model, given the size is d, is $1/\tau(d)$.
- Overall, prior is $\tau(d)^{\xi-1} = 1/\tau(d)^{1-\xi} := 1/\tau(d)^{\gamma}$
- Extended BIC:

$$\text{BIC}_{\gamma} = -2\log(\text{likelihood}) + \log(n)\cdot d + 2\gamma\log(\tau(d))$$

- Not only penalize the size, but also the number of peers models which have the same size.

Are these criteria useful in ML?

- Yes and No.
- Must use **log-likelihood**, or the formula is not exact.
- d must be a reasonable measure of model complexity. Otherwise, use **effective degrees of freedom**
- Effective degrees of freedom not always available.
- If there are too many possible models, computationally expensive to calculate AIC/BIC/eBIC for each of them.
- Validation may be an alternative way to evaluate the model.

The next section would be . . . . . .

Validation Set

If we have a lot of data (data-rich situation): randomly divide the dataset into three parts: **a training set, a validation set, and a test set.**

- training set is used to fit the models;
- validation set (sometimes called *tuning set*) is used to estimate prediction error for the purpose of model selection
  - The error on the validation set is viewed as an estimate of the prediction error of every model under inspection.
- test set is used for a final estimate of the generalization error of the final chosen model.

The three errors

- **Training error**: too optimistic. The algorithm/procedure either minimizes it directly, or minimizes one of its surrogates.
- **Validation/tuning error**: try to estimate the generalization error, for the purpose of selecting models. The model with the smallest tuning error is selected.
- **Test error**: after the final model is chosen, assess the performance. This error is unbiased to the true generalization error. If test data size $\to \infty$, test error converges to the true generalization error.

Since validation step is still part of the model fitting/selection process, validation error of the final chosen model is still too optimistic as estimate of generalization error. Ideally, the test set should be kept in a vault, and be brought out only at the end of the data analysis. Why?

Split the data

Train
Validation
Test

- Personal favor: 1/3 training, 1/3 validation, 1/3 test, or
- Just enough for training, the same size for validation, the rest for test.
- ESLII: 1/2 training, 1/4 validation, 1/4 test.
- My reasoning: the validation set should mimic just how the training data would look like. It provides a second opinion as to how the model work (in addition to the training error.)

What's catch?

- Need lots of data,
- ... which we often cannot afford or we do not have. Usually we are in a data-deficit situation.

Cross-Validation

- A cheap alternative: K-fold cross-validated error.
- Pseudo-code: for each λ value

```
divide train data into $K$ parts
for i = 1 to $K$ do
    train model using all but the $i$th part, and then
    compute error using the $i$th part
end for
compute total error over the $K$ runs
```

As each time the (temporary) training data is similar to the full training data, the resulting classifier may be viewed as close to the classifier that we actually care about. However, the (temporary) test data is not used in training, a good property we desire.

Typically choose the λ (or the model) with the smallest CV error.

A simple example

Consider a simple classifier for large-p-small-n data:

1. Starting with 5000 predictors and 50 samples, find the 100 predictors having the largest correlation with the class labels
2. Conduct LDA using only these 100 genes. Reach a classifier.

How do we estimate the test set performance of this classifier?

Calculate the cross-validation error using the 100 selected genes after step 1?

This is WRONG!

- In step 1, the procedure has already "seen" the response of ALL the training data, and made use ALL of them to select the genes.
- **This is a form of training** and must be counted in the validation process.
- Although CV is conducted, we cannot claim that "the (temporary) test data is not used in training"
- We have seen this error made in at least 4 high profile microarray papers
- WRONG: Apply cross-validation in step 2 only.
- CORRECT: Apply cross-validation to both steps 1 and 2. That is, for each run of the CV, re-select the 500 genes without using $1/K$ of the data that is reserved for testing.

Min vs. One standard error rule.

- CV error is an estimate of the GE.
- It is random and the estimate may not be accurate.
- The CV error curve may be too flat near the optimality.
- The model with the smallest CV error may not be the one with the smallest GE!
- A conservative approach: among models with CV errors within 1 standard error away from the smallest CV error, choose the simplest one.

$$\text{standard error}(\lambda) = s.d.(err_1(\lambda), \ldots, err_K(\lambda))/\sqrt{K}$$

- Idea: their underlying GE are indistinguishable. Then the simpler the better.

More motive for 1-standard-error rule

The goal of recovering the true model, for the sake of interpretation, is somewhat of a different task than the goal of having a model with best predictive power. The value of the parameter that achieves the smallest cross-validation error often corresponds to not enough regularization. But the 1-standard-error rule is a step in the right direction.

Leave-One-Out Cross Validation

- The N-fold cross validation, where N is the sample size, is called Leave-One-Out Cross Validation.

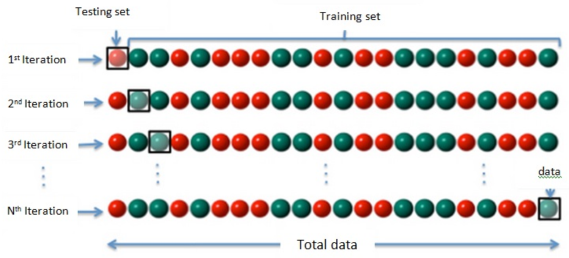

Figure: For ith iteration, testing misclassification rate p_i is computed, average of those p_i is the estimate of the total probability of misclassification $P(\hat{\phi}(\boldsymbol{X}) \neq Y)$.

- The LOOCV error is NOT random with respect to data splitting. WHY?

Choice of K

- Small K:
  - the post-split training data is too small comparable to the actual training data, hence the CV error is biased upward as an estimate of the GE.
- Large K:
  - The post-split training data between runs are too similar to each other. Cannot reduce variance by averaging! Large variance.
- Bias-variance trade-off again!

How many folds are needed?

- $K = 5$ or 10 seems to achieve a good trade-off. Industry standard. Not really any theory.
- K-fold CV ($K \ll N$) is different from N-fold CV in that:
  - For given data, the N-fold CV is deterministic, since permutation of the data does not change the CV value.
  - For given data, the K-fold CV depends on how the data are split into folds. An observation may be judged by a different set of classifiers due to a different way of splitting.
- If time is not an issue, try repeat the K-fold many times with random splitting in order to reduce the variance (but does not work for very large K or $K = n$.)

Leave-one-out shortcut

Suppose we have a linear fitting model,

$$\hat{\boldsymbol{y}} = \mathbf{S}(\lambda)\boldsymbol{y}.$$

It turns out that for many such models, the LOOCV is

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

This cute relation is already very nice. But we can find more convenient approximation.

Generalized cross-validation

GCV is an approximation to the LOOCV above.

$$GCV(\lambda) = \frac{1}{n}\sum_{i=1}^{n}\left[\frac{y_i - \hat{f}(x_i)}{1 - \text{Trace}(\mathbf{S})/n}\right]^2$$

Recall that Trace($\mathbf{S}$) is the effective degrees of freedom.

Fun fact: GCV is connected to AIC. Use the approximation $1/(1+x)^2 \approx 1 + 2x$, we can see that

$$GCV \approx \frac{1}{n}\sum_{i=1}^{n}\left[y_i - \hat{f}(x_i)\right]^2 (1 + \frac{2df}{n})$$

Application Examples

For regression problems, apply CV to

- shrinkage methods: select λ
- best subset selection: select the best model
- nonparametric methods: select the smoothing parameter

For classification methods, apply CV to

- kNN: select k
- SVM: select λ

After the optimal model/parameter is selected, re-fit using the full data to get performance model.