07 - Model Selection

ST 697 | Fall 2017
University of Alabama

Model Complexity

Models can be of varying complexity:

- number of parameters (polynomials, interactions)
- penalty λ or constraint (t) for regularized models
- neighborhood size
- number of trees, tree depth
- etc. (we will cover more models later in course)

Highly adaptable model families can accommodate complex relationships, but easily overemphasize patterns that are not reproducible (e.g. noise or statistical fluctuation)

Goal is to be flexible (complex) enough to find reproducible (true) structure, but not overfit

Tuning parameters

Tuning parameters are the “control knobs" that are not directly estimated from data. It may be better to refer to these as *complexity parameters*.

These usually are connected to the flexibility/complexity of a model. E.g., the λ in lasso and ridge are tuning parameters

- Given the value of the tuning parameter, we often can estimate the other parameters from the data (e.g., $\hat{\beta}$)

Usual Set-up

Optimization function: $J(\theta, \omega) = \ell(\theta) + Pen(\omega, \theta)$

- θ are model parameters
- ω are tuning parameters

Example: Elastic Net Regression

$$\ell(\theta) = \frac{1}{2} \sum_{i=1}^{n} (y_i - x_i^{\mathsf{T}} \theta)^2$$

$$Pen(\omega, \theta) = \lambda \left(\frac{(1-\alpha)}{2} \sum_{j=1}^{p} \theta_j^2 + \alpha \sum_{j=1}^{p} |\theta_j| \right)$$

- $\omega = (\lambda, \alpha)$ (usual notation of $\theta = \beta$)

Usual Set-up

Optimization function: $J(\theta, \omega) = \ell(\theta) + Pen(\omega, \theta)$

- θ are model parameters
- ω are tuning parameters

Estimate model parameters *given* the tuning parameter(s) ω:

$$\hat{\theta}(\omega) = \arg\min_{\theta} J(\theta, \omega)$$

But how to pick the optimal model complexity, ω?

Predictive Performance

Because our focus is on predictive performance (not interpretation or inference), the optimal tuning parameter(s) is the value(s) that minimizes the expected prediction error (PE):

$$\mathsf{EPE}(\omega) = \mathbb{E}[\mathsf{PE}(\omega)] = \mathbb{E}_{\tilde{X},\tilde{Y}}[\ell(\tilde{Y}, \hat{m}_\omega(\tilde{X}))]$$

where $\ell()$ is the loss function (e.g. RSS), and $\tilde{X}, \tilde{Y}$ are drawn from the same distribution (hopefully) as the sample data. $\hat{m}_\omega(x) = m_\omega(x; \hat{\theta}(\mathcal{D}))$ is the prediction function with a given tuning parameter ω

Ideally, we want to pick tuning parameter(s) ω to minimize EPE

$$\omega^{opt} = \arg\min_\omega \ \mathsf{EPE}(\omega)$$

Training Error

There are a few ways to estimate EPE. A *not* very good way is to use the training error (TE):

$$\text{TE}(\omega, \mathcal{D}) = \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, \hat{m}_\omega(x_i))$$

where $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$ is the training data.

And use the ω that minimizes the training error:

$$\omega^* = \arg\min_{\omega} \ \text{TE}(\omega, \mathcal{D})$$

- Which will always overfit (as long as f is flexible enough)
- Why?

Training Error vs. Testing Error (ESL Fig 7.1)

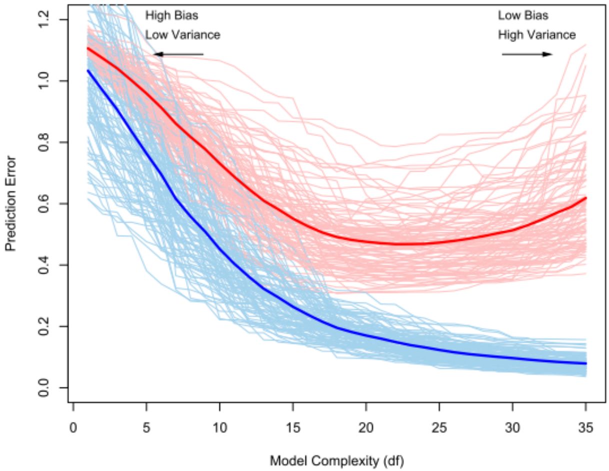


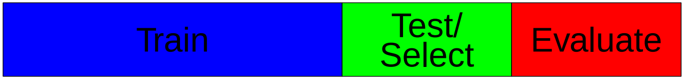
FIGURE 7.1. *Behavior of test sample and training sample error as the model complexity is varied. The light blue curves show the training error* $\overline{\text{err}}$*, while the light red curves show the conditional test error* $\text{Err}_{\mathcal{T}}$ *for 100 training sets of size 50 each, as the model complexity is increased. The solid curves show the expected test error* Err *and the expected training error* $\text{E}[\overline{\text{err}}]$.

Model Selection and Assessment

- **Model selection:** estimating the performance of different models in order to choose the best one.
- **Model assessment:** having chosen a final model, estimating its prediction error on new data (i.e., estimating EPE for final model).

Train/Validate/Test

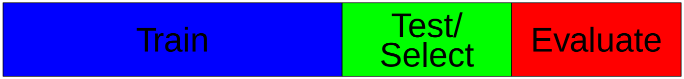
If it were possible to have lots of data (all from the same distribution), then we would split up the data into three pieces:



- Train: Estimate model parameters θ (for given tuning ω)
- Test/Select: Choose optimal tuning parameters ω (i.e., model selection step)
- Evaluate: Estimate EPE (i.e., final model assessment)

Train/Validate/Test

If it were possible to have lots of data (all from the same distribution), then we would split up the data into three pieces:



- Train: Estimate model parameters θ (for given tuning ω)
- Test/Select: Choose optimal tuning parameters ω (i.e., model selection step)
- Evaluate: Estimate EPE (i.e., final model assessment)

1. What estimate suffers where there is not enough data in each group?
2. Is the performance of a model on the test data reflective of its performance on the evaluation data? When/Why do we need evaluation data?

Note on definitions

The ESL text (pg. 222) (and many others) use the following notation: training, validation, test.

Whereas I replaced Validate with Test and Test with Evaluate.

We usually talk about performance on a test set, and that is what we do in phase 2. The last step, Evaluation, is to get a true sense of how well the *final model* will do on new data (since the test set performance will be over-optimistic).

- But this historical nomenclature is where *cross-validation* comes from

Model Selection

The goal of model selection is to pick the model that will provide the best *predictive performance* on test data (i.e, model with smallest EPE).

There are three main approaches to estimating the predictive performance (EPE) of a model:

1. Predict on hold-out test data
2. Make a mathematical adjustment to the training error that better estimates the test error
3. resampling methods (cross-validation, bootstrap)

Hold out set

Set aside some data in a test set $\mathcal{D}_{\text{test}}$ and consider:

$$\text{PE}(\omega, \mathcal{D}_{\text{test}}) = \frac{1}{m} \sum_{i=1}^{m} \ell(\tilde{y}_i, \hat{m}_\omega(\tilde{x}_i))$$

where $(\tilde{x}_i, \tilde{y}_i) \in \mathcal{D}_{\text{test}}$ for $i = 1, 2, \ldots, m$ and
$\hat{m}_\omega(\tilde{x}_i) = m_\omega(\tilde{x}_i; \hat{\theta}(\mathcal{D}_{\text{train}}))$

But the more data used in the test set, the less can be used for estimating model parameters.

Adjustments to Training Error

Another approach is to use all the data for training, but adjust the training error to account for potential over-fitting

- The adjustment is usually a function of model complexity (e.g., edf)
  - This will not work great for lasso, since edf is not continuous in λ

Examples:

- AIC/BIC
- Mallow's Cp
- Adjusted R^2

AIC/BIC

Akaike information criterion (AIC)

$$AIC(\mathcal{M}) = -2\log L(\mathcal{M}) + 2d(\mathcal{M})$$

Bayesian information criterion (BIC)

$$BIC(\mathcal{M}) = AIC(\mathcal{M}) + d(\mathcal{M})\left(\log(n) - 2\right)$$

where $L(\mathcal{M})$ is the likelihood of model $\mathcal{M}$ (requires distributional assumption) and $d(\mathcal{M})$ is the effective degrees of freedom (effective number of model parameters)

Cross-Validation

Performance Evaluation

An obvious way to assess how well a model will perform on test data is to evaluate it on test data.

- We can split our data into a **training** and **test** set
- But how to decide how much data into each set?
    - Too little in training and poor parameter estimates
    - Too little in test and poor performance estimates and model selection

Cross-Validation

Cross-validation is a way to use more of the data for **both** training and testing

- Randomly divide the set of observations into K groups, or folds, of approximately equal size.
- The first fold is treated as a validation set, and the model is fit on the remaining $K-1$ folds. And predictions are made on the hold-out set.
- The performance on each fold is combined to get a more accurate assessment of model performance on future data.

3-fold cross-validation

	Fold 1	Fold 2	Fold 3
Iter 1	Train	Train	Test
Iter 2	Train	Test	Train
Iter 3	Test	Train	Train

Cross-Validation Algorithm

1. Split data into K folds (of roughly equal size)
   - $\mathcal{F}_1, \dots, \mathcal{F}_K$
2. For $k = 1, \dots, K$ and for all models (e.g., for all ω) :
   - 2.1 Use the data in $\mathcal{D} \setminus \mathcal{F}_k$ to estimate the model $\hat{m}_\omega^{-k}$
   - 2.2 Predict for data in $\mathcal{F}_k$ and calculate the average loss

$$V_k(\omega) = \frac{1}{n_k} \sum_{i \in \mathcal{F}_k} \ell(y_i, \hat{m}_\omega^{-k}(x_i))$$

   where n_k is the number of observations in fold k

3. Choose tuning parameters (model selection) that minimize cross-validation loss

$$\hat{\omega} = \arg\min_\omega \text{CV}(\omega)$$

   where $\text{CV}(\omega) = \frac{1}{n} \sum_{k=1}^K n_k V_k(\omega)$

4. Refit all data using $\hat{\omega}$ to get the final model ($\hat{\theta}$). Or combine prediction from all folds.

1 SE Rule

The standard error of the average cross-validation error can be estimated by:

$$\text{SE}(\omega) = \frac{\text{standard deviation of } \{V_k\}}{\sqrt{K}}$$

One Standard Error Rule suggests that instead of using $\hat{\omega} = \arg\min_{\omega} \text{CV}(\omega)$, we should use the least complex model that is within one standard error of $\text{CV}(\hat{\omega})$

Choice of K

- $K = 5, 10, n$ are common choices
- $K = n$ is leave-one-out (LOOCV)
    - closed form solution for models that are linear in y
- Note: around $(K-1)/K$ of the data is used for training and $1/K$ for testing
- if K is too small, then not enough training data and poor cv error estimate
- if K is too large, then the m_ω^{-k} are correlated and variance is not reduced
    - in the sense of training data is similar across folds

Balanced Data Splitting for Cross-Validation

- The most basic approach is to use random sampling to assign observations to folds
- But this can be problematic if there are outliers or classes with small frequency
  - Especially a problem for categorical data. Some levels are not included in training set.
  - So how to predict when they show up in test set?
- Stratified sampling can be used to ensure similar distributions in each fold
  - e.g., equal number of observations in each fold from same quartile of y
- Or based on predictor values: clustering the training data and then assigning into folds such that an equal number of observations from each cluster are in each fold

Different Model Families

Most of the discussion has been on finding the optimal complexity/tuning parameter for a given *model family*.

But cross-validation can be used to compare across model families.

Consider some options and their corresponding complexity/tuning parameters:

- linear regression with stepwise variable selection
  - number of parameters p, or acceptance/rejection criteria
- elastic net (includes lasso and ridge)
  - α and λ
- k-nearest neighbor
  - k

Use CV to pick best-of-the-best