THE UNIVERSITY OF CHICAGO

STAT 37710 / CMSC 35400 / CAAM 37710 Machine Learning

Model Selection: cross validation

Cong Ma

What we have talked about so far

- General stat framework for regression (a form of supervised learning)
- Model-free perspective of linear regression (LR): ERM with linear function class
- Model-based perspective of LR:MLE under conditional Gaussian model
- Computational algorithms for solving LR
- Regularized LR (Ridge and Lasso) and their Bayesian interpretation

But...

- How do we pick the feature mapping $\phi(x)$ in $y \approx w^\top \phi(x)$
  - E.g., how to choose the degree of polynomials?
- How to choose the regularization parameter in either ridge regression or Lasso?
- All of these require us to do model selection

Recall our ultimate goal

- Fundamental assumption: Our data set is generated ***independently and identically distributed*** (*iid*) from some unknown distribution *P*

$$(\mathbf{x}_i, y_i) \sim P(\mathbf{X}, Y)$$

- Our goal is to minimize the ***expected error (true risk)*** under *P*

$$R(h) = \int P((x), y)\ell(y; h(\mathbf{x}))d\mathbf{x}dy = \mathbb{E}_{\mathbf{x},y}\left[\ell(y; h(\mathbf{x})\right]$$

In an ideal world

- Given different hypotheses, we would just calculate

$$R(h) = \int P((x), y)\ell(y; h(\mathbf{x}))d\mathbf{x}dy = \mathbb{E}_{\mathbf{x},y}\left[\ell(y; h(\mathbf{x})\right]$$

and find the one with smallest error

- But this is far from reality: we cannot compute expected error

Fortunately, we have data---using empirical risk

- Assume our data set is generated iid from some unknown *P*
- Our goal is to minimize the ***expected error (true risk)*** under *P*

$$R(\mathbf{w}) = \int P(\mathbf{x}, y)(y - \mathbf{w}^T\mathbf{x})^2 d\mathbf{x}dy$$

$$= \mathbb{E}_{\mathbf{x},y}[(y - \mathbf{w}^T\mathbf{x})^2]$$

- Estimate the **true risk** by the **empirical risk** on a sample data set D

$$\hat{R}_D(\mathbf{w}) = \frac{1}{|D|} \sum_{(\mathbf{x},y)\in D} (y - \mathbf{w}^T\mathbf{x})^2$$

A big issue

- If we use empirical risk to evaluate, we are essentially arguing for ERM

Empirical Risk Minimization: $\hat{\mathbf{w}}_D = \underset{\mathbf{w}}{\operatorname{argmin}} \hat{R}_D(\mathbf{w})$

- Ideally, we wish to solve $\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} R(\mathbf{w})$

- But empirical risk is too optimistic

Experimental evidence

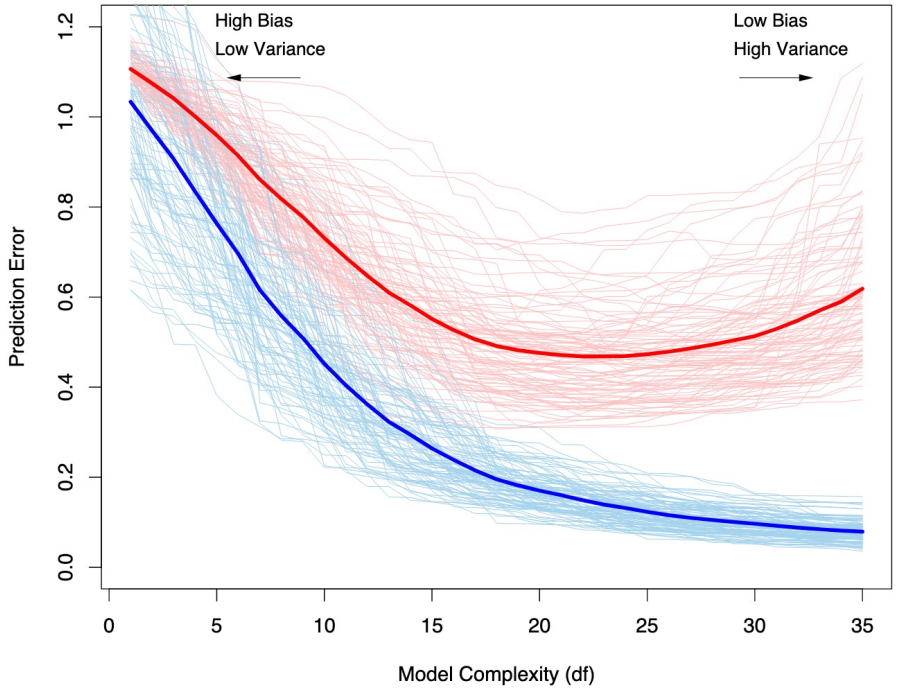

FIGURE 7.1. *Behavior of test sample and training sample error as the model complexity is varied. The light blue curves show the training error* $\overline{\text{err}}$*, while the light red curves show the conditional test error* $\text{Err}_{\mathcal{T}}$ *for* 100 *training sets of size* 50 *each,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

Prediction error and model error

- training set: $\boldsymbol{y}, \boldsymbol{X}$
- $\hat{\boldsymbol{\beta}}$: an estimate based on training set
- new data: $\tilde{\boldsymbol{y}} = \tilde{\boldsymbol{X}}\boldsymbol{\beta} + \tilde{\boldsymbol{\eta}} \in \mathbb{R}^m$, where $\tilde{\boldsymbol{\eta}} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_m)$
- Goal: use $\hat{\boldsymbol{\beta}}$ to predict $\tilde{\boldsymbol{y}}$

One may assess the quality of the estimate based on its *prediction error* on $\tilde{\boldsymbol{y}}$, i.e.

$$
\begin{aligned}
\text{PE} &:= \mathbb{E}\left[\left\|\tilde{\boldsymbol{X}}\hat{\boldsymbol{\beta}} - \tilde{\boldsymbol{y}}\right\|^2\right] \\
&= \mathbb{E}\left[\left\|\tilde{\boldsymbol{X}}(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta})\right\|^2\right] + 2\mathbb{E}\left[(\tilde{\boldsymbol{X}}(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta}))^\top(\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{X}}\boldsymbol{\beta})\right] + \mathbb{E}\left[\left\|\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{X}}\boldsymbol{\beta}\right\|^2\right] \\
&= \underbrace{\mathbb{E}\left[\left\|\tilde{\boldsymbol{X}}(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta})\right\|^2\right]}_{:=\text{ME (model error)}} + \underbrace{m\sigma^2}_{\text{variability of data}}
\end{aligned}
$$

Empirical risk

We shall set $\tilde{\boldsymbol{X}} = \boldsymbol{X}$ (and hence $m = n$) out of simplicity

- the case where the structures of new and old data are the same

Unfortunately, we do not have access to PE (as we don't know $\boldsymbol{\beta}$)

$\Longrightarrow$ need an operational criterion for estimating PE

- One candidate: estimate PE via residual sum of squares

$$\text{RSS} := \|\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}}\|_2^2$$

$\Longrightarrow$ training error

Training error underestimates prediction error

Suppose $\boldsymbol{X}\hat{\boldsymbol{\beta}} = \boldsymbol{\Pi}\boldsymbol{y}$ for some given $\boldsymbol{\Pi}$ with $\mathrm{Tr}(\boldsymbol{\Pi}) > 0$ (e.g. LS), then

$$\mathsf{PE} = \mathbb{E}[\mathsf{RSS}] + 2\sigma^2 \mathrm{Tr}(\boldsymbol{\Pi}) > \mathbb{E}[\mathsf{RSS}] \tag{8.1}$$

Proof:

$$\begin{aligned}
\mathsf{PE} - \mathbb{E}[\mathsf{RSS}] &= \mathbb{E}\left[\|\tilde{\boldsymbol{y}} - \boldsymbol{X}\hat{\boldsymbol{\beta}}\|^2\right] - \mathbb{E}\left[\|\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}}\|^2\right] \\
&= \mathbb{E}\left[\|\tilde{\boldsymbol{y}}\|^2 + \|\boldsymbol{X}\hat{\boldsymbol{\beta}}\|^2 - 2\langle \tilde{\boldsymbol{y}}, \boldsymbol{X}\hat{\boldsymbol{\beta}}\rangle\right] \\
&\quad - \mathbb{E}\left[\|\boldsymbol{y}\|^2 + \|\boldsymbol{X}\hat{\boldsymbol{\beta}}\|^2 - 2\left\langle \boldsymbol{y}, \boldsymbol{X}\hat{\boldsymbol{\beta}}\right\rangle\right] \\
&= 2\mathbb{E}\left[\langle \boldsymbol{y} - \tilde{\boldsymbol{y}}, \boldsymbol{X}\hat{\boldsymbol{\beta}}\rangle\right] = 2\mathbb{E}\left[\langle \boldsymbol{\eta} - \tilde{\boldsymbol{\eta}}, \boldsymbol{\Pi}\boldsymbol{y}\rangle\right] \\
&= 2\mathbb{E}\left[\langle \boldsymbol{\eta}, \boldsymbol{\Pi}\boldsymbol{\eta}\rangle\right] \overset{\text{(a)}}{=} 2\mathrm{Tr}\left(\boldsymbol{\Pi}\mathbb{E}\left[\boldsymbol{\eta}\boldsymbol{\eta}^\top\right]\right) \\
&= 2\sigma^2 \mathrm{Tr}(\boldsymbol{\Pi}),
\end{aligned}$$

where (a) follows from the identity $\mathrm{Tr}(\boldsymbol{A}^\top \boldsymbol{B}) = \mathrm{Tr}(\boldsymbol{B}\boldsymbol{A}^\top)$.

More realistic evaluation: using separate test data

- Want to avoid underestimating the prediction error
- Idea: Use **separate test set** from the same distribution P
- Obtain training and test data D_{train} and D_{test}
- Optimize ***w*** on training set $\hat{\mathbf{w}}_{D_{\text{train}}} = \arg\min_{\mathbf{w}} \hat{R}_{\text{train}}(\mathbf{w})$
- Evaluate on test set $\hat{R}_{\text{test}}(\hat{\mathbf{w}}) = \frac{1}{|D_{\text{test}}|} \sum_{(\mathbf{x},y)\in D_{\text{test}}} \left(y - \hat{\mathbf{w}}^\top \mathbf{x}\right)^2$
- Then $\mathbb{E}_{D_{\text{train}},D_{\text{test}}}\left[\hat{R}_{\text{test}}(\hat{\mathbf{w}}_{D_{\text{train}}})\right] = \mathbb{E}_{D_{\text{train}}}\left[R(\hat{\mathbf{w}}_{D_{\text{train}}})\right]$

First attempt: Evaluation for model selection

- Obtain training and test data $D_{\text{train}}, D_{\text{test}}$
- Fit each candidate model (e.g., degree *m* of polynomial)

$$\hat{\mathbf{w}}_m = \underset{\mathbf{w}:\text{degree}(\mathbf{w}) \leq m}{\text{argmin}} \hat{R}_{\text{train}}(\mathbf{w})$$

- Pick one that does best on test set: $\hat{m} = \underset{m}{\text{argmin}}\, \hat{R}_{\text{test}}(\hat{\mathbf{w}}_m)$

- *Do you see a problem?*

Overfitting to *test* set

- Test error is itself random! Variance usually increases for more complex models
- Optimizing for *single* test set creates bias

Solution: Pick multiple test sets!

- **Key idea**: Instead of using a single test set, use **multiple test sets** and average to decrease variance!
- **Dilemma**:
  Any data I use for testing I can't use for training

- Using multiple independent test sets is expensive and wasteful

Cross validation

- For each candidate model *m* (e.g., polynomial degree) repeat the following procedure for i = 1:k
  - Split the same data set into training and validation set

$$D = D_{\text{train}}^{(i)} \uplus D_{\text{val}}^{(i)}$$

  - Train model $\hat{\mathbf{w}}_{i,m} = \arg\min_{\mathbf{w}} \hat{R}_{\text{train}}^{(i)}(\mathbf{w})$
  - Estimate error $\hat{R}_m^{(i)} = \hat{R}_{\text{val}}^{(i)}(\hat{\mathbf{w}}_i)$
- Select model

$$\hat{m} = \operatorname{argmin}_m \frac{1}{k} \sum_{i=1}^{k} \hat{R}_m^{(i)}$$

How should we do the splitting?

- Randomly (Monte Carlo cross-validation)
  - Pick training set of given size uniformly at random Validate on remaining points
  - Estimate prediction error by averaging the validation error over multiple random trials

- k-fold cross-validation
  - Partition the data into *k* “folds”
  - Train on *(k-1)* folds, evaluating on remaining fold
  - Estimate prediction error by averaging the validation error obtained while varying the validation fold

Accuracy of cross-validation

- Cross-validation error estimate is very nearly unbiased for large enough *k*
- How large should we pick k?
  - Too small
    - Risk of overfitting to test set
    - Using too little data for training
    - risk of underfitting to training set
  - Too large
    - In general, better performance! k=n is perfectly fine (called leave-one-out cross-validation, LOOCV)
    - Higher computational complexity
- In practice, k=5 or k=10 is often used and works well

Best practice for evaluating supervised learning

- Split data set into training and test set
- Never look at test set when fitting the model.

  For example, use *k*-fold cross-validation on training set
- Report final accuracy on test set

  (but never optimize on test set)!

- **Caveat**: This only works if the data is i.i.d.

References & acknowledgement

- K. Murphy (2021). “Probabilistic Machine Learning: An Introduction”
  - Ch 4.5.4, 4.5.5, “Regularization”

- Hastie et al. (2021). “The Elements of Statistical Learning”
  - Ch 18.3.4, “Feature Selection”
  - Ch 18.4 “Linear Classifiers with L1 Regularization”

- Virgil Pavlu, “Feature Selection, Sparsity, Regression Regularization”
  - http://www.ccs.neu.edu/home/vip/teach/MLcourse/5_features_dimensions/lecture_notes/feature_selection/features_selection.pdf

- A. Krause, “Introduction to Machine Learning” (ETH Zurich, 2019)