Errors and Cross-validation

Wei Li

Syracuse University

Spring 2021

Errors and Validation

Problems of Least Squares Methods

Problems of Least Squares Methods

Recall linear model:

$$Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p + \epsilon$$

Simple, and has advantages:

1. Interpretability
2. Good predictive performance in many cases

There are two reasons we might be able to improve on least square fit within linear model framework:

1. Prediction Accuracy
2. Model Interpretability

Prediction Accuracy

Prediction Accuracy

Suppose the (X, Y) has a joint distribution given by $P(X, Y)$, which can be represented as

$$Y = f\left(X\right) + \epsilon$$

with $E(\epsilon|X) = 0$ and $Var(\epsilon|X) = \sigma^2$.

Denote random sample (i.i.d.) $\boldsymbol{\tau} = \left\{\left(\boldsymbol{x}_i, y_i\right), i = 1, \cdots, n\right\}$.

Using $\boldsymbol{\tau}$, we can obtain some $\cdot \mapsto \hat{f}(\cdot)$ (which depends on $\boldsymbol{\tau}$):

e.g., using linear regression, $\cdot \mapsto \hat{f}(\cdot) = \hat{\beta}^T(\cdot)$, using knn regression, $\cdot \mapsto \hat{f}(\cdot) = \frac{1}{k}\sum_{i \in N_k(\cdot)} y_i$.

Therefore, the prediction rule for Y given $X = x$ is $\hat{y} = \hat{f}(x)$.

Decomposition of MSE

Let (X_0, Y_0) be an arbitrary test point (future observation), satisfying

$$Y_0 = f\left(X_0\right) + \epsilon_0, \qquad \left(X_0, Y_0\right) \sim P(X, Y)$$

and independent of $\boldsymbol{\tau}$.

- **Mean Squared Error** (MSE) for $\hat{f}$ at $\boldsymbol{x}_0$:

$$\mathrm{MSE}\left(\boldsymbol{x}_0\right) = \mathrm{E}_{\boldsymbol{\tau}}\left[\hat{y}_0 - f\left(\boldsymbol{x}_0\right)\right]^2 = \mathrm{E}_{\boldsymbol{\tau}}\left[\hat{f}\left(\boldsymbol{x}_0\right) - f\left(\boldsymbol{x}_0\right)\right]^2$$

We can decompose the $\mathrm{MSE}\left(\boldsymbol{x}_0\right)$ as

$$\begin{aligned} \mathrm{MSE}\left(\boldsymbol{x}_0\right) &= E_{\boldsymbol{\tau}}\left[\hat{f}\left(\boldsymbol{x}_0\right) - f\left(\boldsymbol{x}_0\right)\right]^2 \\ &= \mathrm{Var}_{\boldsymbol{\tau}}\left[\hat{f}\left(\boldsymbol{x}_0\right)\right] + \mathrm{Bias}^2\left[\hat{f}\left(\boldsymbol{x}_0\right)\right] \end{aligned}$$

Expected Prediction/test Error (EPE) at $\boldsymbol{x}_0$:

$$\mathrm{EPE}\left(\boldsymbol{x}_0\right) = E_{Y_0|X_0=\boldsymbol{x}_0} E_{\boldsymbol{\tau}} \left[\hat{y}_0 - Y_0\right]^2 = E_{Y_0|X_0=\boldsymbol{x}_0} E_{\boldsymbol{\tau}} \left[\hat{f}\left(\boldsymbol{x}_0\right) - Y_0\right]^2$$

$$\mathrm{EPE}(\boldsymbol{x}_0) = \sigma^2 + \mathrm{Var}_{\boldsymbol{\tau}}\left[\hat{f}\left(\boldsymbol{x}_0\right)\right] + \mathrm{Bias}^2\left[\hat{f}\left(\boldsymbol{x}_0\right)\right]$$

Remark:

- The noise and pure stochastic nature of the error (irreducible)
- The variation from estimating our $\hat{f}$ (from limited sample size)
- The bias from estimating our model for f
  - model mis-specification error (intrinsic error from model)
  - average error from estimation

Note:

$$\left[E_{\boldsymbol{\tau}}\left(\hat{f}\left(\boldsymbol{x}_0\right) - f\left(\boldsymbol{x}_0\right)\right)\right]^2 = \left[E_{\boldsymbol{\tau}}\left(\hat{f}\left(\boldsymbol{x}_0\right) - f^*\left(\boldsymbol{x}_0\right)\right) + f^*\left(\boldsymbol{x}_0\right) - f\left(\boldsymbol{x}_0\right)\right]^2$$

where $f^* = \arg\min_{\tilde{f}\in\mathcal{F}} E(f(X) - \tilde{f}(X))^2$.

Bias-Variance Tradeoff

Bias-Variance Tradeoff

The goal is to achieve small EPE is to minimize both the variance and bias by using some good estimator for f (model).

It turns out, there is a **bias-variance tradeoff**.

Consider the k-nearest-neighbor regression:

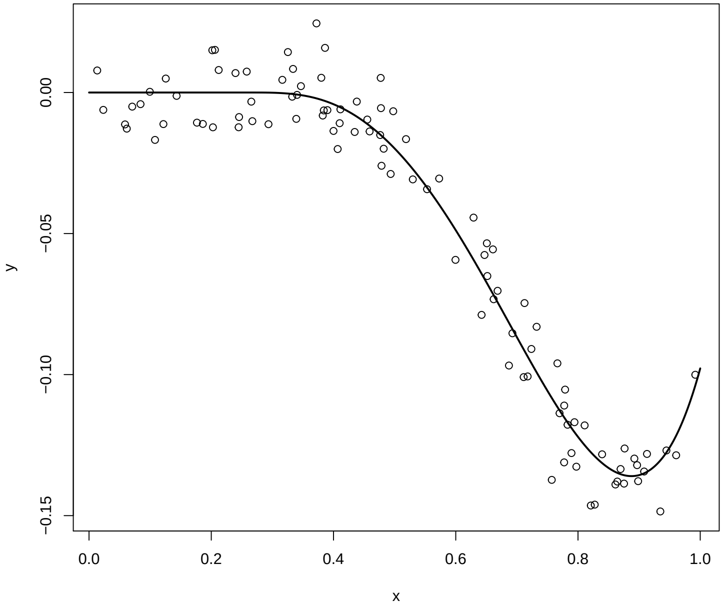
$$\hat{f}(x) = \frac{1}{k} \sum_{i \in N_k(x)} y_i$$

where $N_k(x)$ gives the k nearest neighbors of x.

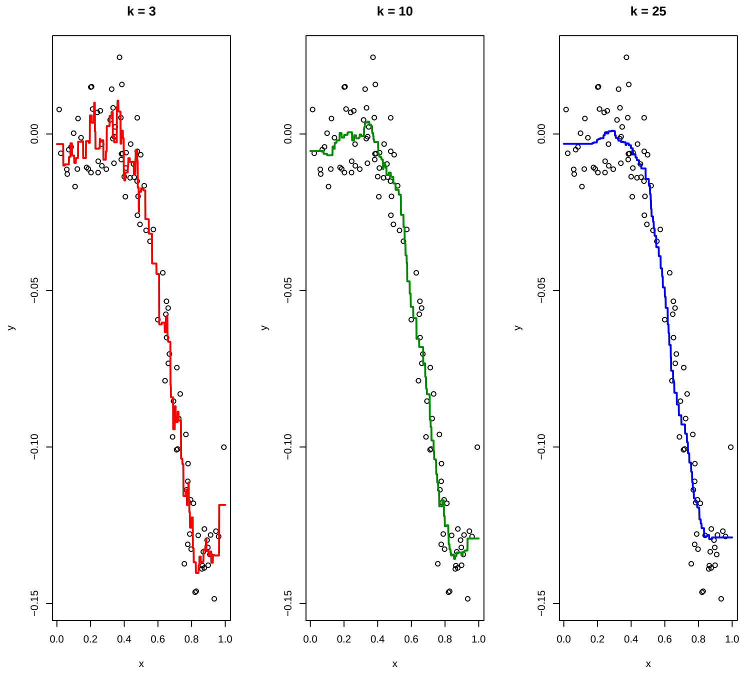
To use this in practice, we're going to need to choose k. What are the tradeoffs at either end?

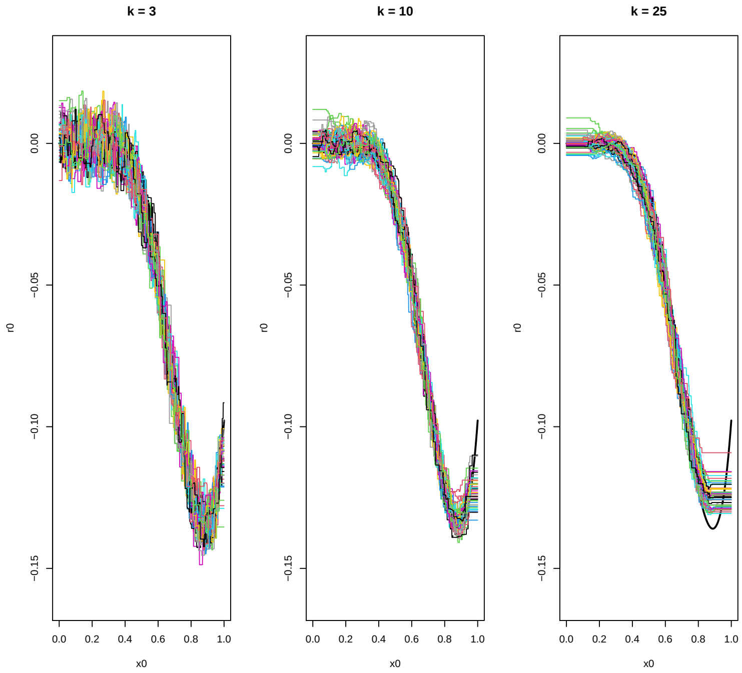
For small k, we risk picking up noisy features of the sample that don't really have to do with the true regression function $f(x)$, called **overfitting**; for large k, we may miss important details, due to averaging over too many y_i values, called **underfitting**.

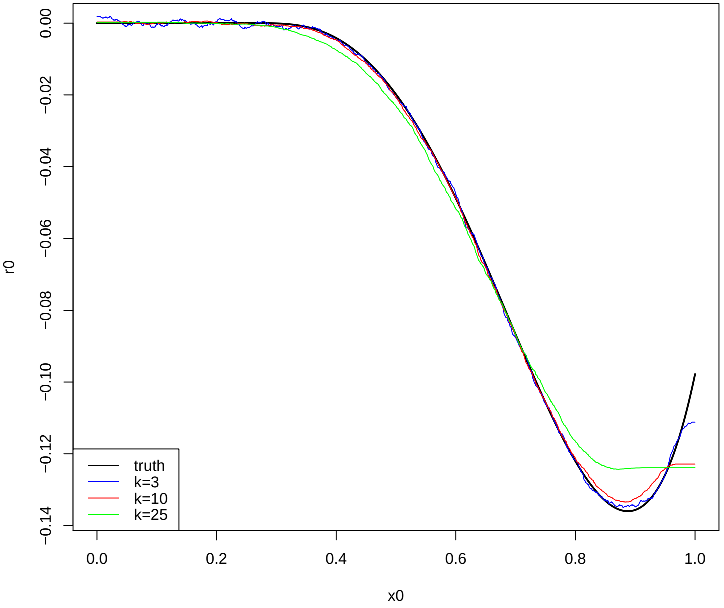
Underfitting means high bias and low variance, overfitting means low bias but high variance.



0.00
−0.05
−0.10
−0.15
y
0.0
0.2
0.4
0.6
0.8
1.0
x


k = 3
k = 10
k = 25
y
x
0.00
−0.05
−0.10
−0.15
0.0
0.2
0.4
0.6
0.8
1.0


k = 3
k = 10
k = 25
r0
x0
0.00
−0.05
−0.10
−0.15
0.0
0.2
0.4
0.6
0.8
1.0


0.00
−0.02
−0.04
−0.06
−0.08
−0.10
−0.12
−0.14
r0
0.0
0.2
0.4
0.6
0.8
1.0
x0
truth
k=3
k=10
k=25

(Overall) expected test error:

$$\text{EPE} = \text{Err} = E_{(Y_0, X_0, \boldsymbol{\tau})}\left[\left(Y_0 - \hat{f}(X_0)\right)^2\right]$$

Of course, one practical question is how to measure this test error? A good estimate is the **expected test error**

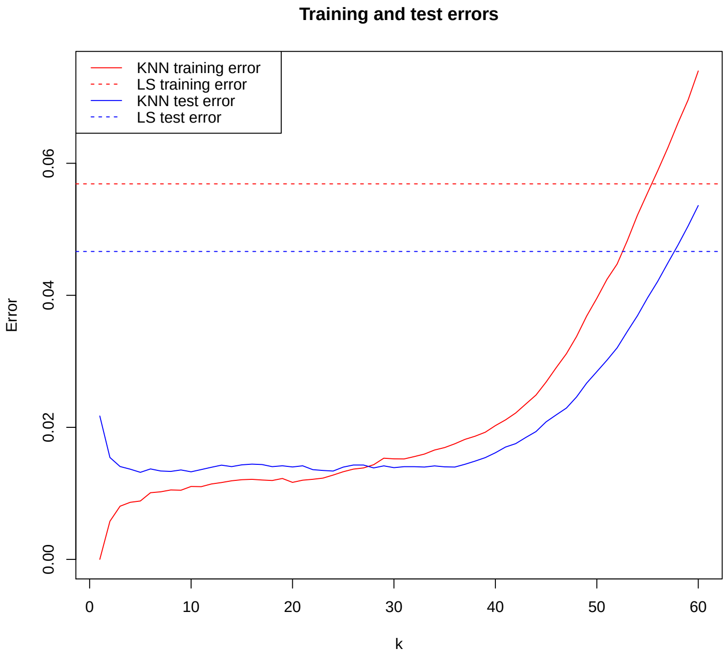
$$E\left[\frac{1}{n'}\sum_{i=1}^{n'}\left(Y_{0i} - \hat{f}(X_{0i})\right)^2\right], \text{ where } (X_{0i}, Y_{0i})_{i=1}^{n'} \text{ is the new sample}$$

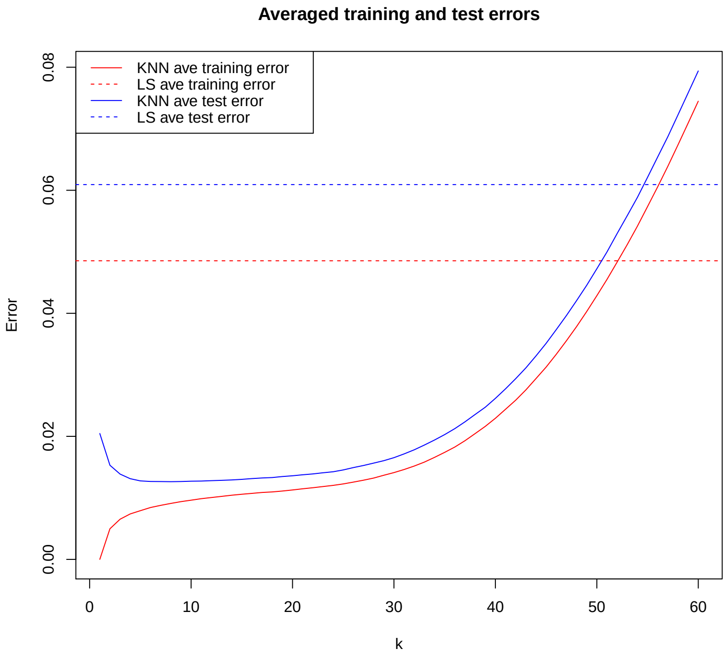
The expectation here is taken over all that is random (both training and test samples)

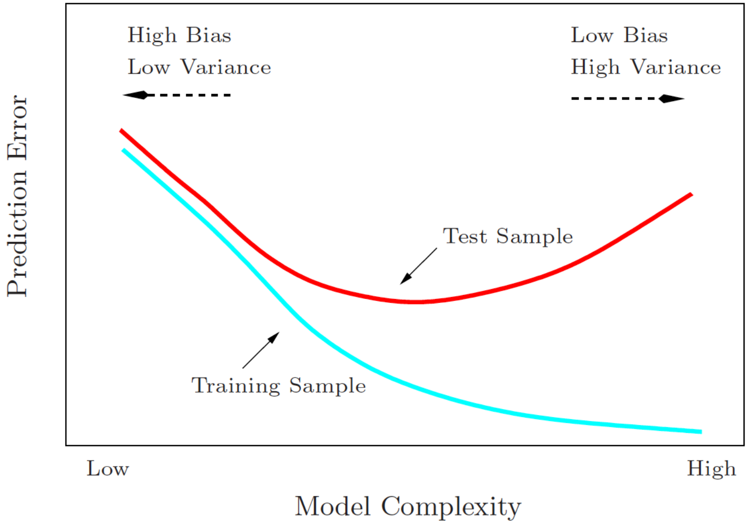
Another obvious candidate **expected training error** is

$$E\left[\frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{f}(X_i)\right)^2\right].$$

Unfortunately this is not a good estimate of test error.


Training and test errors
KNN training error
LS training error
KNN test error
LS test error
Error
0.00
0.02
0.04
0.06
k
0
10
20
30
40
50
60


Averaged training and test errors
KNN ave training error
LS ave training error
KNN ave test error
LS ave test error
Error
0.00
0.02
0.04
0.06
0.08
k
0
10
20
30
40
50
60


High Bias
Low Variance
Low Bias
High Variance
Prediction Error
Test Sample
Training Sample
Low
High
Model Complexity

ESL: Fig 2.11

Double Descent

In the deep learning literature, it has been observed a phenomenon about the test curve–called “double descent”.

It is observed and conjectured that after the model complexity surpasses a certain threshold (“interpolating or overparametrized regime”), the variance from estimating over-parametrized models may start to decrease significantly, which may then lower the test curve. This is mostly an empirical finding, and a rigorous theoretical explanation is not yet available.

For most of the models we discuss here, we are in the “classical regime” where we do not attempt to interpolate the data nor attempt to overparametrize the model. The benefit for doing this is not clear in view of the double descent phenomenon.

ref. Belkin et al. PNAS 2019.

Estimation of Test Errors

Validation-set approach

Without tuning:

- Randomly divide the available set of samples into two parts: a training set and a validation or hold-out set.
- The model is fit on the training set, and the fitted model is used to predict the responses for the observations in the validation set.

With tuning: Randomly divide the dataset into three parts:

- training set: to fit the models
- validation (tuning) set: to estimate the prediction error for model selection
- test set: to assess the generalization error of the final chosen model

The typical proportions are respectively: $50\%, 25\%, 25\%$.

Cross-validation

Hold-out each point (y_i, x_i) from our training set in turn, fit $\hat{f}^{-(i)}$ on all points except this one (i.e., $(y_j, x_j), j \neq i$), record the squared error on (x_i, y_i) and average the results. This yields the so-called **leave-one-out cross validation** (LOOCV) estimate of test error

$$CV(\hat{f}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}^{-(i)}(x_i))^2$$

where $\hat{f}^{-(i)}$ is fitted using all training data except the i-th observation.

More generally, we can split up our training set into K divisions or folds, for some number K: **K-fold validation**.

- Let the K parts be $C_1, C_2, \dots C_K$, where C_k denotes the indices of the observations in part k. There are n_k observations in part k : if n is a multiple of K, then $n_k = n/K$
- Compute

$$CV(\hat{f}) = \frac{1}{K} \sum_{k=1}^{K} \mathrm{CV}_k$$

  where $\mathrm{CV}_k := \mathrm{CV}_k(\hat{f}^{-(k)}) = \sum_{i \in C_k} (y_i - \hat{f}^{-(k)}(x_i))^2 / n_k$, and $\hat{f}^{-(k)}(x_i)$ is the fit for observation i, obtained from the data with part k removed.

Common choice of K: $K = 5$ or $K = 10$, or $K = n$.

Remark:

If each training set only uses a part of the sample, it tends to overestimate test error. Compared with validation approach and general K-fold CV, LOOCV has less bias, thus tend to not to overestimate that test error rate. LOOCV will give estimates of the test error with smallest bias.

However, LOOCV has higher variance than does K-fold CV ($K < n$), because the n training sets are so similar to one another.

Note also LOOCV does not involve random splitting.

CV standard error

$$\mathrm{Var}(\mathrm{CV}(\hat{f})) = \mathrm{Var}\left(\frac{1}{K}\sum_{k=1}^{K}\mathrm{CV}_k\right) \approx \frac{1}{K}\,\mathrm{Var}\left(\mathrm{CV}_1\right)$$

This approximation is valid for small K(e.g., , $K = 5$ or 10) but not really for big K (e.g., $K = n$).

For small K(e.g., $K = 5$ or 10), we estimate of the variance of the cross-validation error estimate:

$$\frac{1}{K}\,\mathrm{S}\left\{\mathrm{CV}_1, \ldots \mathrm{CV}_K\right\}$$

One standard erorr rule

Suppose $\hat{f} = \hat{f}_\alpha$ depends on some tuning parameter α.

We find the best parameter $\hat{\alpha}$ by minimizing $CV(\hat{f}_\alpha)$.

Let $\alpha \in \{\alpha_1, \ldots, \alpha_m\}$, within a set of candidate values for α. Then we have the cross validation error

$$CV(\hat{f}_\alpha) = \frac{1}{K}\sum_{k=1}^{K} CV_k(\hat{f}_\alpha^{-(k)})$$

- The standard error of $CV(\hat{f}_\alpha)$ is calculated as

$$SE(\alpha) = \sqrt{\mathrm{Var}(CV(\hat{f}_\alpha))}$$

$$\hat{\alpha} = \arg \min_{\alpha \in \{\alpha_1, \ldots, \alpha_m\}} CV(\hat{f}_\alpha)$$

The **one-standard error rule** is to choose the most parsimonious model with error no more than one standard error above the best error. That is, to choose the α among $\{\alpha : CV(\hat{f}_\alpha) \leq CV(\hat{f}_{\hat{\alpha}}) + SE(\hat{\alpha})\}$ that gives the most parsimonious model (less complex model).

The implication for the linear model

If *true* relationship is approximately linear, then Ordinary Least Squares has low bias. But for variance:

- When n $\gg p$, OLS tends to have low variance, and hence perform well on test observations.
- When $n < p$, then there is no longer OLS estimate, and the variance of these estimates are infinite. May be able to increase bias slightly, but decrease variance substantially via
  - regularization/shrinkage
  - features selection
  - dimension reduction
- When n is not much larger than p, then the least squares fit can have high variance and may result in over fitting and poor prediction on future observations.

The **key message**: It is possible to trade a little bias with the large reduction in variance, thus achieving higher prediction accuracy.

Model Interpretability

Model Interpretability

Even it is true that $Y_i = \beta^T X_i + \epsilon_i$ where $E\left[\epsilon_i \mid X_i\right] = 0$ and $\operatorname{Var}\left(\epsilon_i \mid X_i\right) = \sigma^2$,

- When we have a large number of variables X in the model there will generally be many that have little or no effect on Y
- Leaving these variables in the model makes it harder to see the "big picture", i.e., the effect of the "important variables"
- The model would be easier to interpret by removing (i.e. setting the coefficients to zero) the unimportant variables. This can be accomplished via variable selection, or feature selection.