Model Selection and Assessment

김지수 (Jisu KIM)

통계적 기계학습(Statistical Machine Learning), 2024 1st semester

The lecture note is a minor modification of the lecture notes from Prof. Yongdai Kim's "Statistical Machine Learning", and Prof Larry Wasserman and Ryan Tibshirani's "Statistical Machine Learning". Also, see Section 7 from [1].

1 Review

1.1 Basic Model for Supervised Learning

- Input(입력) / Covariate(설명 변수) : $x \in \mathbb{R}^p$, so $x = (x_1, \ldots, x_p)$.
- Output(출력) / Response(반응 변수): $y \in \mathcal{Y}$. If y is categorical, then supervised learning is "classification", and if y is continuous, then supervised learning is "regression".
- Model(모형) :

  $$y \approx f(x).$$

  If we include the error ϵ to the model, then it can be also written as

  $$y = \phi(f(x), \epsilon).$$

  For many cases, we assume additive noise, so

  $$y = f(x) + \epsilon.$$

- Assumption(가정): f belongs to a family of functions $\mathcal{M}$. This is the assumption of a model: a model can be still used when the corresponding assumption is not satisfied in your data.
- Loss function(손실 함수): $\ell(y, a)$. A loss function measures the difference between estimated and true values for an instance of data.
- Training data(학습 자료): $\mathcal{T} = \{(y_i, x_i), i = 1, \ldots, n\}$, where (y_i, x_i) is a sample from a probability distribution P_i. For many cases we assume i.i.d., or x_i's are fixed and y_i's are i.i.d..
- Goal(목적): we want to find f that minimizes the expected prediction error,

  $$f^0 = \arg\min_{f \in \mathcal{F}} \mathbb{E}_{(Y,X)\sim P} \left[\ell(Y, f(X))\right].$$

  Here, $\mathcal{F}$ can be different from $\mathcal{M}$; $\mathcal{F}$ can be smaller then $\mathcal{M}$.
- Prediction model(예측 모형): f^0 is unknown, so we estimate f^0 by $\hat{f}$ using data. For many cases we minimizes on the empirical prediction error, that is taking the expectation on the empirical distribution $P_n = \frac{1}{n}\sum_{i=1}^n \delta_{(Y_i,X_i)}$.

  $$\hat{f} = \arg\min_{f \in \mathcal{F}} \mathbb{E}_{P_n} \left[\ell(Y, f(X))\right] = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n \ell(Y_i, f(X_i)).$$

- Prediction(예측): if $\hat{f}$ is a predicted function, and x is a new input, then we predict unknown y by $\hat{f}(x)$.

1.2 Linear Regression

From the additive noise model

$$y = f(x) + \epsilon,\ f \in \mathcal{M},$$

Linear Regression Model (선형회귀모형) is that

$$\mathcal{M} = \mathcal{F} = \left\{ \beta_0 + \sum_{j=1}^{p} \beta_j x_j : \beta_j \in \mathbb{R} \right\}.$$

For estimating β, we use least squares: suppose the training data is $\{(y_i, x_{ij}) : 1 \le i \le n, 1 \le j \le p\}$. We use square loss

$$\ell(y, a) = (y - a)^2,$$

then the eimpirical loss becomes the residual sum of square (RSS) as

$$\begin{aligned} RSS(\beta) &= \sum_{i=1}^{n} (y_i - f(x_i))^2 \\ &= \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2. \end{aligned}$$

Let $\hat{\beta} = (\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_p)$ be the nimimizor of RSS, then the predicted function is

$$\hat{f}(x) = \hat{\beta}_0 + \sum_{j=1}^{p} \hat{\beta}_j x_j.$$

1.3 Overfitting and the Bias-Variance Tradeoff

In k-Nearest Neighbor, the neighbor size k determines the complexity of the prediction model. To choose k, if we know the probability distribution $(Y, X) \sim P$, then we can minimize the Expected Prediction Error (EPE, 평균예측오차 / 예측오차)

$$\mathbb{E}_{(Y,X)\sim P} \left[\ell(Y, f(X)) \right].$$

However, P is usually unknown. One way to estimate EPE is by Training Error (TE, 학습오차)

$$\frac{1}{n} \sum_{i=1}^{n} \ell(Y_i, f(X_i)),$$

where the training data $\mathcal{T} = \{(Y_i, X_i) : i = 1, \ldots, n\}$.

However, training error always underestimates prediction error, since the same dataset $\mathcal{T}$ is used for estimating the prediction model and estimating prediction error. Better estimation for the prediction error is the test error, where the prediction error is computed on the separate test data. The typical behavior of the prediction error on the test set is as in Figure 1. As the model becomes more complex, the prediction error first decreases but then increases, while test error monotonically decreases. Hence, when the model is chosen according to the training error, then the prediction error can be large: this is called overfitting. This phenomenon is explained by the bias-variance tradeoff.

2 Introduction

- Suppose that we observe (X, Y) from some unknown joint distribution, and we want to predict Y from X. Over all functions f, the expected prediction error, measured in terms of squared loss,

  $$\mathbb{E}\big[\big(Y - f(X)\big)^2\big]$$

  is minimized at $f(x) = \mathbb{E}(Y|X = x)$. This is called the true regression function associated with the pair (X, Y).

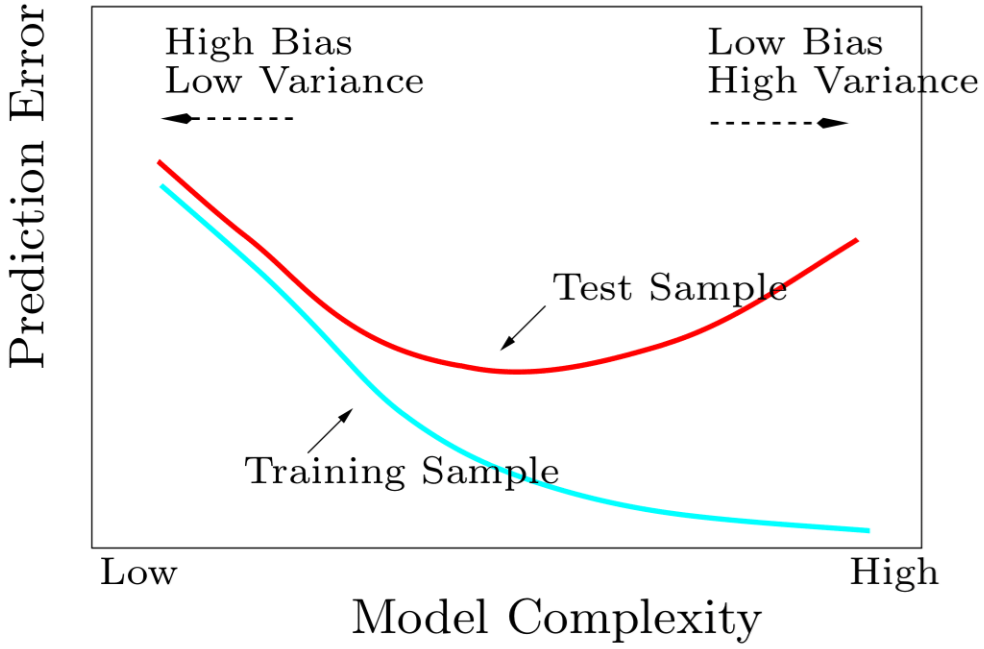

Figure 1: Test and training error as a function of model complexity. Figure 2.11 from [1].

- Now suppose that we observe training samples $(x_1, y_1), \ldots (x_n, y_n)$ i.i.d. from the same joint distribution as (X, Y). We use these training samples to construct a prediction function $\hat{f}$; this mimics $f(x) = \mathbb{E}(Y|X = x)$, which is optimal but unknowable in practice. The expected prediction error, or expected test error, of $\hat{f}$ is

$$\mathbb{E}\big[\big(Y - \hat{f}(X)\big)^2\big], \tag{1}$$

  where the expectation is over all that is random, namely, the training set (x_i, y_i), $i = 1, \ldots n$ and the test point (X, Y)

- Why would we be interested in (1)? Here are two reasons:

  1. Model assessment: sometimes we want to know how well we can predict a future observation, in absolute terms
  2. Model selection: often we want to choose between different prediction functions; this might mean choosing between two different model classes entirely, or choosing an underlying tuning parameter for a single method. In principle, we could do this by comparing expected test errors

- The expected test error in (1) has an important property: it decomposes into informative parts. But before we show this, let's think about a few points:

  – Can we ever predict Y from X with zero error? Generally, no. Even the true regression function f cannot generically do this, and will incur some error due to noise. We call this *irreducible error*

  – What happens if our fitted function $\hat{f}$ belongs to a model class that is far from the true regression function f? E.g., we choose to fit a linear model in a setting where the true relationship is far from linear? As a result, we encounter error, what we call *estimation bias*

  – What happens if our fitted (random) function $\hat{f}$ is itself quite variable? In other words, over different copies of the training set, we end up constructing substantially different functions $\hat{f}$? This is another source of error, that we'll call *estimation variance*

We'll talk about the following methods: Cross-validation, AIC, BIC.

Perhaps the only slightly counter-intuitive fact you need to remember is that when there is a true model, methods like cross-validation can fail to find it.

The basic take-aways are:

1. If your goal is prediction, you have a reasonable sample-size and you have a reasonable computation budget use cross-validation.
2. If your goal is prediction, but you either have too small a sample or you have a very low computational budget, you should consider using AIC.
3. If your goal is selecting the true model you should use BIC.

3 Statistical Prediction and the Bias-Variance Tradeoff, revisited

- More so than just these intuitive descriptions, the expected test error mathematically decomposes into a sum of three corresponding parts. Begin by writing the model

$$Y = f(X) + \varepsilon,$$

where ε has mean zero, variance σ^2, and is independent of X. Note that the independence condition is the actual (nontrivial) assumption. Recall that (x_i, y_i), $i = 1, \ldots n$ are independent of each other and of (X, Y), all with the same distribution. We'll look at the expected test error, conditional on $X = x$ for some arbitrary input x. It follows that

$$\mathbb{E}\big[\big(Y - \hat{f}(x)\big)^2 \big| X = x\big] = \sigma^2 + \underbrace{\mathbb{E}\big[\big(f(x) - \hat{f}(x)\big)^2\big]}_{\mathrm{Risk}(\hat{f}(x))}.$$

The first term σ^2 is the *irreducible error*, or sometimes referred to as the *Bayes error*, and the second term is called the risk, or mean squared error (MSE). The risk further decomposes into two parts, so that

$$\mathbb{E}\big[\big(Y - \hat{f}(x)\big)^2 \big| X = x\big] = \sigma^2 + \underbrace{\big(f(x) - \mathbb{E}[\hat{f}(x)]\big)^2}_{\mathrm{Bias}^2(\hat{f}(x))} + \underbrace{\mathbb{E}\big[\big(\hat{f}(x) - \mathbb{E}[\hat{f}(x)]\big)^2\big]}_{\mathrm{Var}(\hat{f}(x))}, \tag{2}$$

the latter terms being the squared *estimation bias* or simply *bias*, and the *estimation variance* or simply *variance*, respectively. The decomposition (2) is called the *bias-variance decomposition* or *bias-variance tradeoff*

- From the bias-variance tradeoff (2), we can see that even if our prediction is unbiased, i.e., $\mathbb{E}[\hat{f}(x)] = f(x)$, we can still incur a large error if it is highly variable. Meanwhile, even when our prediction is stable and not variable, we can incur a large error if it is badly biased. There is a tradeoff here, but it need it is really never be one-to-one; e.g., in some cases, it can be worth sacrificying a little bit of bias to gain large decrease in variance
- Typical trend: underfitting means high bias and low variance, overfitting means low bias but high variance. E.g., think about k in k-nearest-neighbors regression: recall that, trained on (x_i, y_i), $i = 1, \ldots n$, the k-nearest-neighbors regression fit at an input point x is

$$\hat{f}(x) = \frac{1}{k} \sum_{i \in \mathcal{N}_k(x)} y_i, \tag{3}$$

where $\mathcal{N}_k(x)$ denotes the indices of the k training inputs closest to x. How do the bias and variance behave for small k, and for large k?

Let's consider the input training points $x_1, \ldots x_n$ all fixed (nonrandom) for simplicity. Then for the k-nearest-neighbors fit $\hat{f}$,

$$\mathbb{E}\big[\big(Y - \hat{f}(X)\big)^2 \big| X = x\big] = \sigma^2 + \Big(f(x) - \frac{1}{k} \sum_{i \in \mathcal{N}_k(x)} f(x_i)\Big)^2 + \frac{\sigma^2}{k}.$$

Hence we can explicitly see that as k increases, the variance decreases, and the bias (likely) increases. Note that, for fixed $x_1, \ldots x_n$, the k-nearest-neighbors fit $\hat{f}(x)$ in (3), at any input x, is just a linear function of $y_1, \ldots y_n$. This is an example of what is called a linear smoother, which we'll discuss more later in Section 9

- Finally, unconditionally over X, the bias-variance decomposition reads

$$\mathbb{E}\big[\big(Y - \hat{f}(X)\big)^2\big] = \sigma^2 + \int \mathrm{Bias}^2(\hat{f}(x))\, P_X(dx) + \int \mathrm{Var}(\hat{f}(x))\, P_X(dx),$$

where P_X is the distribution of X (this follows from taking an expectation over X in (2)). In words, the expected test error is the Bayes error + average (squared) bias + average variance

4 Training error and optimism

- Now that we have convinced ourselves that the expected test error (1) is a useful quantity, how do we estimate it? Clearly if we had a test set $(x'_1, y'_1), \ldots (x'_m, y'_m)$, independent of the training set and from the same distribution, then the observed average test error

$$\frac{1}{m} \sum_{i=1}^{m} \big(y'_i - \hat{f}(x'_i)\big)^2$$

would serve as an unbiased estimate for $\mathbb{E}[(Y - \hat{f}(X))^2]$. But we are often not this fortunate to have test data

- What's wrong with the average training error

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

This measures the squared error of $\hat{f}$ around the data we used to fit it—the problem is that this will be, generally speaking, far too optimistic. Why? Because we chose $\hat{f}$ so that it fit well to the training data (x_i, y_i), $i = 1, \ldots n$, of course its error is going to look too small

- This rules out using the training error for the first purpose described above, model assessment. But if the expected training error was systematically (always) different than the expected test error by the same amount, then we could use it for the second purpose, model selection. E.g., if it the training error curve had the right shape with k (but just not the right height), then we could still use it to choose k in k-nearest-neighbors regression. Unforunately, this is essentially never the case; in almost all real examples, the training error curve has both the wrong magnitude and the wrong shape. (What is the 1-nearest-neighbor training error?)

- The expected test and training errors can be linked in a useful way, if we again assume that fixed inputs, for simplicity. That is, let $(x_1, y'_1), \ldots (x_n, y'_n)$ denote a test set, independent of and having the same distribution as as $(x_1, y_1), \ldots (x_n, y_n)$. The inputs $x_1, \ldots x_n$ are fixed in both sets, and only the observations y_i, y'_i, $i = 1, \ldots n$ are random, drawn according to

$$\begin{aligned} y_i &= f(x_i) + \epsilon_i, \quad i = 1, \ldots n, \\ y'_i &= f(x_i) + \epsilon'_i, \quad i = 1, \ldots n, \end{aligned}$$

all errors ϵ_i, ϵ'_i, $i = 1, \ldots n$ i.i.d. with mean zero and variance σ^2

In vector notation, write $y, y' \in \mathbb{R}^n$ for the training and test observations, and write $\hat{f} = (\hat{f}(x_1), \ldots \hat{f}(x_n)) \in \mathbb{R}^n$ for the training fitted values. Then the expected test error is

$$\frac{1}{n} \mathbb{E} \|y' - \hat{f}\|_2^2,$$

where the expectation is taken over y, y'. Now write $f = (f(x_1), \ldots f(x_n)) \in \mathbb{R}^n$, and decompose

$$\begin{aligned} \frac{1}{n} \mathbb{E} \|y' - \hat{f}\|_2^2 &= \frac{1}{n} \mathbb{E} \|y' - f + f - \hat{f}\|_2^2 \\ &= \sigma^2 + \frac{1}{n} \mathbb{E} \|f - \hat{f}\|_2^2 \\ &= 2\sigma^2 + \frac{1}{n} \mathbb{E} \|y - \hat{f}\|_2^2 + \frac{2}{n} \mathbb{E} (f - y)^T (y - \hat{f}) \\ &= \frac{1}{n} \mathbb{E} \|y - \hat{f}\|_2^2 + \frac{2}{n} \mathrm{tr}\big(\mathrm{Cov}(y, \hat{f})\big). \end{aligned}$$

In other words, the *optimism*—difference in expected test and expected training errors—can be expressed as

$$\frac{1}{n} \mathbb{E} \|y' - \hat{f}\|_2^2 - \frac{1}{n} \mathbb{E} \|y - \hat{f}\|_2^2 = \frac{2}{n} \mathrm{tr}\big(\mathrm{Cov}(y, \hat{f})\big) = \frac{2}{n} \sum_{i=1}^{n} \mathrm{Cov}\big(y_i, \hat{f}(x_i)\big). \tag{4}$$

So the higher the correlation between y_i and its fitted value $\hat{f}(x_i)$, the greater the optimism. We'll see that in many cases we can either evaluate or estimate the right-hand side above, which can be extremely useful

5 Degrees of freedom and unbiased risk estimation

- For $\hat{f} = (\hat{f}(x_1), \ldots \hat{f}(x_n)) \in \mathbb{R}^n$ and the same setup as before—that is, $x_1, \ldots x_n$ considered fixed, and $\hat{f}$ trained on $(x_1, y_1), \ldots (x_n, y_n)$—the *degrees of freedom* of $\hat{f}$ is defined as

$$\mathrm{df}(\hat{f}) = \frac{1}{\sigma^2} \sum_{i=1}^{n} \mathrm{Cov}\big(y_i, \hat{f}(x_i)\big). \tag{5}$$

  Hence the optimism in (4) can be more succintly written as

$$\frac{1}{n}\mathbb{E}\|y' - \hat{f}\|_2^2 - \frac{1}{n}\mathbb{E}\|y - \hat{f}\|_2^2 = \frac{2\sigma^2}{n}\mathrm{df}(\hat{f}). \tag{6}$$

  Intuitively, we can think of $\mathrm{df}(\hat{f})$ as the effective number of parameters used by the fit $\hat{f}$, a measure of complexity of $\hat{f}$. A few simple examples that support this intuition:

  - If $\hat{f} = (y_1, \ldots y_n)$, then $\mathrm{df}(\hat{f}) = n$
  - If $\hat{f}(x_i) = \bar{y} = \frac{1}{n}\sum_{j=1}^{n} y_j$ for each $i = 1, \ldots n$, then $\mathrm{df}(\hat{f}) = 1$
  - If $\hat{f}(x_i) = \frac{1}{k}\sum_{j \in \mathcal{N}_k(x)} y_j$ for each $i = 1, \ldots n$, then $\mathrm{df}(\hat{f}) = n/k$

- Suppose that we can compute the degrees of freedom of $\hat{f}$, or even more broadly, compute an unbiased estimate $\widehat{\mathrm{df}}(\hat{f})$, with the property

$$\mathbb{E}[\widehat{\mathrm{df}}(\hat{f})] = \mathrm{df}(\hat{f}).$$

  Then following from (6), the quantity

$$\hat{T} = \frac{1}{n}\|y - \hat{f}\|_2^2 + \frac{2\sigma^2}{n}\widehat{\mathrm{df}}(\hat{f})$$

  serves as unbiased estimate for the expected test error of $\hat{f}$, i.e.,

$$\mathbb{E}(\hat{T}) = \frac{1}{n}\mathbb{E}\|y' - \hat{f}\|_2^2$$

- Often times you will encounter this principle referred to in terms of unbiased risk estimation, since

$$\hat{R} = \hat{T} - \sigma^2 = \frac{1}{n}\|y - \hat{f}\|_2^2 + \frac{2\sigma^2}{n}\widehat{\mathrm{df}}(\hat{f}) - \sigma^2$$

  serves as an unbiased estimate for the risk,

$$\mathbb{E}(\hat{R}) = \frac{1}{n}\mathbb{E}\|f - \hat{f}\|_2^2$$

  We'll see how to explicitly compute degrees of freedom for linear smoothers, in Section 9. This is a suprisingly broad class. Later in the course, we'll see how to use Stein's formula to compute unbiased estimates of degrees of freedom in an even more general setting, too

- Note: all of the above assumes that σ^2 is known, which is not realisic. In practice, we'd have to estimate it

- An example of how to use this for model selection: suppose that we were considering fitted models $\hat{f}_\theta$ indexed by a parameter $\theta \in \Theta$; e.g., this could represent the number of neighbors k in k-nearest-neighbors regression. Suppose also that had access to $\widehat{\mathrm{df}}(\hat{f}_\theta)$ at each value of θ. Then we could choose the parameter by minimizing our test error estimate,

$$\hat{\theta} = \mathrm{argmin}_{\theta \in \Theta} \; \frac{1}{n}\|y - \hat{f}_\theta\|_2^2 + \frac{2\sigma^2}{n}\widehat{\mathrm{df}}(\hat{f}_\theta).$$

  An important point to be aware of: while $\hat{T}(\hat{f}_\theta) = \frac{1}{n}\|y - \hat{f}_\theta\|_2^2 + \frac{2\sigma^2}{n}\widehat{\mathrm{df}}(\hat{f}_\theta)$ gives an unbiased estimate for the expected test error at any fixed θ, we should anticipate that $\hat{T}(\hat{f}_{\hat{\theta}})$ is itself downward biased (optimistic) as an estimate for the expected test error at $\hat{\theta}$. Why?

6 Cross-validation

- Cross-validation (CV) is quite a general tool for estimating the expected test error (1), that makes minimal assumptions—i.e., it doesn't assume that $Y = f(X) + \varepsilon$ with ε independent of X, it doesn't assume that the training inputs $x_1, \ldots x_n$ are fixed, all it really assumes is that the training samples $(x_1, y_1), \ldots (x_n, y_n)$ are i.i.d.

  We split up our training set into K divisions or folds, for some number K; usually this is done randomly. Write these as $F_1, \ldots F_K$, so $F_1 \cup \ldots \cup F_K = \{1, \ldots n\}$. Now for each $k = 1, \ldots K$, we fit our prediction function on all points but those in the kth fold, denoted $\hat{f}^{-(k)}$, and evaluate squared errors on the points in the kth fold,

  $$\mathrm{CV}_k(\hat{f}^{-(k)}) = \frac{1}{n_k} \sum_{i \in F_k} \big(y_i - \hat{f}^{-(k)}(x_i)\big)^2.$$

  Here n_k denotes the number of points in the kth fold, $n_k = |F_k|$. We average these fold-based errors to yield an estimate of the expected test error,

  $$\mathrm{CV}(\hat{f}) = \frac{1}{K} \sum_{k=1}^{K} \mathrm{CV}_k(\hat{f}^{-(k)}). \tag{7}$$

  This is called K-fold cross-validation; the special case when $K = n$ is referred to as leave-one-out cross-validation

- What is the difference between choosing say $K = 5$ (a common choice) versus $K = n$?

  - When $K = 5$, the function $\hat{f}^{-(k)}$ in each fold k is fit on about $4/5 \cdot n$ samples, and so we are looking at the errors incurred by a procedure that is trained on less data than the full $\hat{f}$ in (1). Therefore the mean of the CV estimate (7) could be off. When $K = n$, this is not really an issue, since each $f^{-(k)}$ is trained on $n - 1$ samples
  - When $K = n$, the CV estimate (7) is an average of n extremely correlated quantities; this is because each $\hat{f}^{-(k)}$ and $\hat{f}^{-(\ell)}$ are fit on $n - 2$ common training points. Hence the CV estimate will likely have very high variance. When $K = 5$, the CV estimate will have lower variance, since it the average of quantities that are less correlated, as the fits $\hat{f}^{-(k)}$, $k = 1, \ldots 5$ do not share as much overlapping training data

  This is tradeoff (the bias-variance tradeoff, in fact!). Usually, a choice like $K = 5$ or $K = 10$ is more common in practice than $K = n$, but this is probably an issue of debate

- For K-fold CV, it's can be helpful to assign a notion of variability to the CV error estimate. We argue that

  $$\mathrm{Var}\big(\mathrm{CV}(\hat{f})\big) = \mathrm{Var}\Big(\frac{1}{K} \sum_{k=1}^{K} \mathrm{CV}_k(\hat{f}^{-(k)})\Big) \approx \frac{1}{K} \mathrm{Var}(\mathrm{CV}_1(\hat{f}^{-(1)})). \tag{8}$$

  Why is this an approximation? This would hold exactly if $\mathrm{CV}_1(\hat{f}^{-(1)}), \ldots \mathrm{CV}_K(\hat{f}^{-(K)})$ were i.i.d., but they're not. This approximation is valid for small K (e.g., $K = 5$ or 10) but not really for big K (e.g., $K = n$), because then the quantities $\mathrm{CV}_1(\hat{f}^{-(1)}), \ldots \mathrm{CV}_K(\hat{f}^{-(K)})$ are highly correlated

- For small K (e.g., $K = 5$ or 10), we can leverage (8) to get an estimate of the variance of the CV error estimate. We just use the sample variance appropriately, so that

  $$\frac{1}{K} \sum_{k=1}^{K} \Big(\mathrm{CV}_k(\hat{f}^{-(k)}) - \mathrm{CV}(\hat{f})\Big)^2$$

  is our estimate of the variance of $\mathrm{CV}(\hat{f})$

- We can use this variance estimate to draw approximate standard deviation bands around a CV error curve, and this leads to model selection heuristics like the *one-standard-error rule*. That is, the usual rule for selecting a parameter θ in a family of fitted models $\hat{f}_\theta$, $\theta \in \Theta$ would be to minimize the CV error:

  $$\hat{\theta} = \mathrm{argmin}_{\theta \in \Theta}\ \mathrm{CV}(\hat{f}_\theta).$$

The one-standard-error rule, instead, chooses the simplest model that is within one standard error (i.e., one standard deviation) of the minimum CV error. How to measure simplicity? Well, degrees of freedom is one way to do so! Often, the parametrization $\hat{f}_\theta$ will be such that $\mathrm{df}(\hat{f}_\theta)$ behaves monotonically with θ; e.g., in k-nearest-neighbors, the degrees of freedom increases as k increases (in fact, it equals k). When $\mathrm{df}(\hat{f}_\theta)$ is monotone increasing with θ, the one-standard-error rule can be written as

$$\tilde{\theta} = \min\Big\{\theta \in \Theta : \mathrm{CV}(\hat{f}_\theta) \leq \mathrm{CV}(\hat{f}_{\hat{\theta}}) + \mathrm{SE}(\hat{f}_{\hat{\theta}})\Big\},$$

where $\mathrm{SE}(\hat{f}_\theta)$ denotes our estimate of the standard deviation of $\mathrm{CV}(\hat{f}_\theta)$, as explained above. Just to be perfectly clear, this is

$$\mathrm{SE}(\hat{f}_\theta) = \frac{1}{\sqrt{K}}\bigg[\sum_{k=1}^{K}\Big(\mathrm{CV}_k(\hat{f}_\theta^{-(k)}) - \mathrm{CV}(\hat{f}_\theta)\Big)^2\bigg]^{1/2}$$

6.1 Cross-validation with KL divergence

For comparing CV, AIC and BIC, we use KL divergence instead of ℓ_2 loss. KL divergence between two densities p and q is defined as

$$KL(p,q) = \int p(x)\log\left(\frac{p(x)}{q(x)}\right)dx.$$

Suppose we find the mle $\hat{\theta}_j$ for model $\mathcal{M}_j$. Let p be the true density of Y and $p_j = p(\cdot;\hat{\theta}_j)$ be the estimated density of Y from model $\mathcal{M}_j$. The idea is to choose j which minimizes $K(p,p_j)$, i.e.,

$$\hat{j} = \mathrm{argmin}_j KL(p,p_j).$$

Notice that minimizing $K(p,p_j)$ is the same as maximizing

$$R_j = \int p(x)\log p_j(x)dx.$$

For K-fold cross validation, we can estimate R_j from calculating from all points but those in the kth fold and averaging them as

$$\mathrm{CV}(\hat{f}) = \frac{1}{K}\sum_{k=1}^{K}\frac{1}{n_k}\sum_{i\in F_k}\log p(x_i;\hat{\theta}_j^{-(k)}).$$

7 AIC

Suppose we don't want to use data-splitting because of lack of data or lack of computational resources. We could try to estimate R_j by

$$\hat{R}_j = \frac{1}{n}\sum_i \log p(x_i;\hat{\theta}_j) = (1/n)\ell_j(\hat{\theta}_j).$$

But, as we discussed above, this is very biased. And the more parameters there are, ther more biased this will be. Akaike proved that the bias is approximately equal to d_j/n where d_j is the dimension of the model $\mathcal{M}_j$. So a bias-adjusted estimator is

$$\hat{R}_j = (1/n)[\ell_j(\hat{\theta}_j) - d_j].$$

For a variety of historical reasons, we will multiple by $2n$ which does not affect which model is the maximizer. This leads to the criterion

$$\mathrm{AIC}_j = 2\ell_j(\hat{\theta}_j) - 2d_j.$$

We choose j to maximize AIC_j. It is best to think of this as an approximation to CV.

8 BIC

The BIC crierion is

$$\text{BIC}_j = 2\ell_j(\hat{\theta}_j) - d_j \log n.$$

This is basically AIC but with a harsher penalty. This approach was proposed by Gideon Schwartz based on the following Bayesian argument. We place prior probabilities $\nu_1, \ldots, \nu_k$ for each model. Then we put priors $p_j(\theta_j)$ for the parameters in model $\mathcal{M}_j$. By Bayes' theorem

$$p(\mathcal{M}_j|X_1, \ldots, X_n) = \frac{p(X_1, \ldots, X_n|\mathcal{M}_j)\nu_j}{\sum_s p(X_1, \ldots, X_n|\mathcal{M}_s)\nu_s} = \frac{\nu_j \int \mathcal{L}_j(\theta_j)d\theta_j}{\sum_s \nu_s \int \mathcal{L}_s(\theta_s)d\theta_s}.$$

Schwartz showed that

$$\log p(\mathcal{M}_j|X_1, \ldots, X_n) \approx \text{BIC}_j.$$

Suppose that the true density is in one of the models. Let $\mathcal{M}_j$ be the smallest model that contains p. Then it can be shown that $\hat{j} \xrightarrow{P} j$ where $\hat{j}$ is the model chosen by BIC.

9 Linear smoothers

- Consider a prediction function $\hat{f}$ trained on (x_i, y_i), $i = 1, \ldots n$, that can be written as

  $$\hat{f}(x) = \sum_{i=1}^{n} w_i(x) y_i = w(x)^T y.$$

  Here the weights $w_i(x)$, $i = 1, \ldots n$ are allowed to depend on the query point x and $x_1, \ldots x_n$. With $x_1, \ldots x_n$ considered fixed, this just a linear function of $y_1, \ldots y_n$, and therefore such a function $\hat{f}$ is called a *linear smoother*

- We saw that the k-nearest-neighbor regression fit is a linear smoother, but so are many other interesting examples. We'll see a lot more in the the nonparametric regression lectures, but e.g., both linear regression and ridge regression are examples of linear smoothers. In these cases, each $x_i \in \mathbb{R}^p$, $i = 1, \ldots n$, and we write $X \in \mathbb{R}^{n \times p}$ as the matrix whose ith row is x_i. Then for linear regression of y on X,

  $$\hat{f}(x) = x^T \hat{\beta} = x^T (X^T X)^{-1} X^T y,$$

  and for ridge regression of y on X, with tuning parameter $\lambda \geq 0$,

  $$\hat{f}(x) = x^T \hat{\beta}_\lambda = x^T (X^T X + \lambda I)^{-1} X^T y$$

- A linear smoother produces fitted values $\hat{\mu} = (\hat{f}(x_1), \ldots \hat{f}(x_n))$ of the form

  $$\hat{\mu} = Sy$$

  for a matrix $S \in \mathbb{R}^{n \times n}$, whose ith row is $w_i(x)$. Just to emphasize, the matrix S can depend on $x_1, \ldots x_n$, and also on external quantities like a tuning parameter, but it cannot depend on $y = (y_1, \ldots y_n)$. E.g., for linear regression, $S = X(X^T X)^{-1} X^T$, and for ridge regression, $S = X(X^T X + \lambda I)^{-1} X^T$

- It turns out that a lot of specific things can be said about linear smoothers. The degrees of freedom of a linear smoother is easily computed:

  $$\text{df}(\hat{f}) = \frac{1}{\sigma^2} \text{tr}\big(\text{Cov}(y, \hat{y})\big) = \text{tr}(S),$$

  with $\text{tr}(S)$ denoting the trace of the smoother matrix S. For linear regression, this turns out to be $\text{tr}(X(X^T X)^{-1} X^T) = p$, and for ridge regression, this is $\text{tr}(X(X^T X + \lambda I)^{-1} X^T) < p$ when $\lambda > 0$.

9.1 Cross-validation

K-fold cross-validation can be used to estimate the prediction error and choose tuning parameters.

For linear smoothers $\hat{\mu} = (\hat{f}(x_1), \ldots \hat{f}(x_n)) = SY$, leave-one-out cross-validation can be particularly appealing because in many cases we have the seemingly magical reduction

$$\mathrm{CV}(\hat{f}) = \frac{1}{n}\sum_{i=1}^{n}\big(Y_i - \hat{f}^{-(i)}(X_i)\big)^2 = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{Y_i - \hat{f}(X_i)}{1 - S_{ii}}\right)^2, \tag{9}$$

where $\hat{f}^{-(i)}$ denotes the estimated regression function that was trained on all but the ith pair (X_i, Y_i). This leads to a big computational savings since it shows us that, to compute leave-one-out cross-validation error, we don't have to actually ever compute $\hat{f}^{-(i)}$, $i = 1, \ldots, n$.

Why does (9) hold, and for which linear smoothers $\hat{\mu} = Sy$? Just rearranging (9) perhaps demystifies this seemingly magical relationship and helps to answer these questions. Suppose we knew that $\hat{f}$ had the property

$$\hat{f}^{-(i)}(X_i) = \frac{1}{1 - S_{ii}}\big(\hat{f}(X_i) - S_{ii}Y_i\big). \tag{10}$$

That is, to obtain the estimate at X_i under the function $\hat{f}^{-(i)}$ fit on all but (X_i, Y_i), we take the sum of the linear weights (from our original fitted function $\hat{f}$) across all but the ith point, $\hat{f}(X_i) - S_{ii}Y_i = \sum_{i \neq j} S_{ij}Y_j$, and then renormalize so that these weights sum to 1.

This is not an unreasonable property; e.g., we can immediately convince ourselves that it holds for kernel smoothing. A little calculation shows that it also holds for smoothing splines (using the Sherman-Morrison update formula).

From the special property (10), it is easy to show the leave-one-out formula (9). We have

$$Y_i - \hat{f}^{-(i)}(X_i) = Y_i - \frac{1}{1 - S_{ii}}\big(\hat{f}(X_i) - S_{ii}Y_i\big) = \frac{Y_i - \hat{f}(X_i)}{1 - S_{ii}},$$

and then squaring both sides and summing over n gives (9).

Finally, *generalized cross-validation* is a small twist on the right-hand side in (9) that gives an approximation to leave-one-out cross-validation error. It is defined as by replacing the appearances of diagonal terms S_{ii} with the average diagonal term $\mathrm{tr}(S)/n$,

$$\mathrm{GCV}(\hat{f}) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{Y_i - \hat{f}(X_i)}{1 - \mathrm{tr}(S)/n}\right)^2 = (1 - \nu/n)^{-2}\hat{R}$$

where ν is the effective degrees of freedom and $\hat{R}$ is the training error. This can be of computational advantage in some cases where $\mathrm{tr}(S)$ is easier to compute than individual elements S_{ii}.

10 What to do?

The derivations of AIC and BIC depend on many technical assumptions about the model. CV does not depend on any model assumptions. For this reason, CV is the better choice if it is feasible.

A difficult problem that we have not considered is how to account for model selection when doing inference. This is a complicated topic. The simplest think, if we have lots of data, is to keep some hold out data. After model selection, we use the hold out data which is not affected by the model selection process.

Another issue that we have not considered is interpretability. Getting good predictions is not the only goal. We might be willing to sacrice a bit of prediction accuracy to have a more interpretable model. This is an area of active research.

References

[1] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning.* Springer Series in Statistics. Springer, New York, second edition, 2009. Data mining, inference, and prediction.