Error and Validation

Advanced Methods for Data Analysis (36-402/36-608)

Spring 2014

1 Testing and training errors

1.1 Setup and motivation

- Let's suppose we have a function for predicting Y from X, let's call it $\hat{r}$. Further suppose that $\hat{r}$ was fit on n training samples (x_i, y_i), $i = 1, \ldots n$. You can think of these as independent samples from some common distribution

  Now consider another point (X, Y), independent of our training sample, and from the same distribution. We are interested to know how well the function $\hat{r}$, constructed on the training set $(x_1, y_1), \ldots (x_n, y_n)$, can predict our new test point (X, Y). To measure this, we define the *expected test error* as

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

  where the expectation is over all that is random (training set, and test point)

- Well, why would we ever want to do this? Here are two reasons (these are not the only two, but two major ones):

  1. Performance assessment: sometimes we would just like to know how well we can predict a future observation, in absolute terms
  2. Model selection: often we will want to choose between different fitting functions; this could mean choosing between two different model classes entirely (e.g., linear regression versus some other fixed method) or choosing an underlying tuning parameter for a single method (e.g., choosing k in k-nearest neighbors). How to do this? A common way is to compare expected test errors

1.2 The bias-variance tradeoff

- The expected test error in (1), also called the prediction error, has an important property: it decomposes into informative parts

- Before we do this, let's think about a few points:

  1. Can we ever predict Y from X with zero prediction error? Likely, no. Even if for some magical reason our function $\hat{r}$ happened to capture the ideal relation underlying X, Y, i.e., the true regression function $r(X) = \mathbb{E}(Y|X)$, we would still incure some error, due to noise. We call this *irreducible error*
  2. What happens if our fitted function $\hat{r}$ belongs to a model class that is far from the true regression function r? E.g., we choose to fit a linear model in a setting where the true relationship is far from linear? We'll often refer to this as a *misspecified model*. As a result, we encounter error, what we call *estimation bias*

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

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

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

where we'll assume that ε is independent of X with mean zero and variance σ^2. Recall that (x_i, y_i), $i = 1, \ldots n$ are independent pairs with the same distribution as (X, Y), and that $\hat{r}$ is fit on this training set. We'll look at the expected test error, conditional on $X = x$ for some arbitrary input x,

$$\begin{aligned}
\mathbb{E}[\text{TestErr}(\hat{r}(x))] &= \mathbb{E}\big[\big(Y - \hat{r}(x)\big)^2 \big| X = x\big] \\
&= \mathbb{E}\big[\big(Y - r(x)\big)^2 \big| X = x\big] + \mathbb{E}\big[\big(r(x) - \hat{r}(x)\big)^2 \big| X = x\big] \\
&= \sigma^2 + \mathbb{E}\big[\big(r(x) - \hat{r}(x)\big)^2\big].
\end{aligned}$$

The first term is just a constant, σ^2, and is the *irreducible error* (sometimes referred to as the *Bayes error*). The second term can be further decomposed as

$$\begin{aligned}
\mathbb{E}\big[\big(r(x) - \hat{r}(x)\big)^2\big] &= \big(\mathbb{E}[\hat{r}(x)] - r(x)\big)^2 + \mathbb{E}\big[\big(\hat{r}(x) - \mathbb{E}[\hat{r}(x)]\big)^2\big] \\
&= \text{Bias}(\hat{r}(x))^2 + \text{Var}(\hat{r}(x)),
\end{aligned}$$

the first term being the squared *estimation bias* or simply *bias*, $\text{Bias}(\hat{r}(x)) = \mathbb{E}[\hat{r}(x)] - r(x)$, and the second term being the *estimation variance* or simply *variance*. Therefore, altogether,

$$\mathbb{E}[\text{TestErr}(\hat{r}(x))] = \sigma^2 + \text{Bias}(\hat{r}(x))^2 + \text{Var}(\hat{r}(x)), \tag{2}$$

which 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., $\mathbb{E}[\hat{r}(x)] = r(x)$, 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; i.e., in some cases, it can be worth sacrificying a little bit of bias to gain large decrease in variance, and in other cases, vice versa

- 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: relatively speaking, how do the bias and variance behave for small k, and for large k?

1.3 The optimism of training error

- What's wrong with the *expected training error*,

$$\mathbb{E}[\text{TrainErr}(\hat{r})] = \mathbb{E}\Big[\frac{1}{n}\sum_{i=1}^{n}\big(y_i - \hat{r}(x_i)\big)^2\Big], \tag{3}$$

which measures the squared error of $\hat{r}$ around the data we used to fit it? The problem is that this will be, generally speaking, far too optimistic. Why? Because we chose $\hat{r}$ 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 expected training error for our first purpose described above, performance assessment. However, if the expected training error was systematically (always) different than the expected test error by the same amount, then we could use it for our 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, examples show that this is essentially never the case. Hence the expected training error cannot really be used at all for either purpose. We'll see this quite blatantly in the R working examples

2 Estimating error in practice

2.1 A single hold-out test point

- In practice, if we fit $\hat{r}$ on samples (x_i, y_i), $i = 1, \ldots n$, then we don't typically have a way of computing the expected test error in (1). We can easily estimate the expected training error in (3), by

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

  but as explained above, this is not at all a good surrogate for test error

- Looking back at (1), we can see that we need a test point; without access to any more data samples, what can we do? One idea is to fit our prediction function on the first $n-1$ training samples (x_i, y_i), $i = 1, \ldots n-1$, call it $\hat{r}^{(-n)}$, and then treat the last point (x_n, y_n) as a test sample. Therefore we have expected test error estimate

$$\big(y_n - \hat{r}^{-(n)}(x_n)\big)^2.$$

  Let's consider two questions:

  1. Does this estimate have the right expectation? Not quite (why?) but it probably won't be far off at all
  2. Is this estimate variable? Probably yes, since we're only using a single sample point to evaluate error

2.2 Leave-one-out cross-validation

- A natural extension of the above: hold-out each point (x_i, y_i) from our training set in turn, fit $\hat{r}^{-(i)}$ on all points except this one (i.e., (x_j, y_j), $j \neq i$), record the squared error on (x_i, y_i), and average the results. This yields the test error estimate

$$\mathrm{CVErr}(\hat{r}) = \frac{1}{n} \sum_{i=1}^{n} \big(y_i - \hat{r}^{-(i)}(x_i)\big)^2.$$

  This method is called *leave-one-out cross-validation* (sometimes acronymized LOOCV).

- Compared to using a single hold-out test point, we've increased the computational burden (by a factor of n!), but with the advantage that now the variance of our error estimate is greatly reduced

  (We haven't really changed the expectation of our test error estimate and should still have $\mathbb{E}[\mathrm{CVErr}(\hat{r})] \approx \mathbb{E}[\mathrm{TestErr}(\hat{r})]$)

2.3 K-fold cross-validation

- There is yet another way to carve things up. We split up our training set into K divisions or *folds*, for some number K. Usually this division 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, call it $\hat{r}^{-(k)}$, and evaluate error on the points in the kth fold,

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

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

$$\mathrm{CVErr}(\hat{r}) = \frac{1}{K} \sum_{k=1}^{K} \mathrm{CV}_k(\hat{r}^{-(k)}).$$

  This is called *K-fold cross validation*, and note that leave-one-out cross-validation is a special case of this corresponding to $K = n$

- Another highly common choice (other than $K = n$) is to choose $K = 5$ or $K = 10$. What does this do?

  1. The mean of our estimate is now a little bit farther off, i.e., $\mathbb{E}[\mathrm{CVErr}(\hat{r})]$ is probably a little farther from $\mathbb{E}[\mathrm{TestErr}(\hat{r})]$ (why?), but still for large enough n this shouldn't be a problem
  2. On the plus side, we've actually managed to further decrease the variance of our error estimate. Now $\mathrm{CVErr}(\hat{r})$ is the average of quantities that are less correlated (than they were with $K = n$), because the fits $\hat{r}^{-(k)}$, $k = 1, \ldots K$ are not dependent on as much overlapping data. Note that the variance of the sum of highly correlated quantities is larger than that with midly correlated quantities

2.4 Cross-validation standard errors

- For K-fold cross-validation, it's very helpful to assign a quantitative notion of variability to the cross-validation error estimate. We argue that

$$\mathrm{Var}(\mathrm{CVErr}(\hat{r})) = \mathrm{Var}\Big(\frac{1}{K} \sum_{k=1}^{K} \mathrm{CV}_k(\hat{r}^{-(k)})\Big) \approx \frac{1}{K} \mathrm{Var}(\mathrm{CV}_1(\hat{r}^{-(1)})). \tag{4}$$

  Why is this an approximation? This would hold exactly if $\mathrm{CV}_1(\hat{r}^{-(1)}), \ldots \mathrm{CV}_K(\hat{r}^{-(K)})$ 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 $\mathrm{CV}_1(\hat{r}^{-(1)}), \ldots \mathrm{CV}_K(\hat{r}^{-(K)})$ are highly correlated

- For small K (e.g., $K = 5$ or 10), we can leverage (4) to get an estimate of the variance of the cross-validation error estimate. We use

$$\frac{1}{K} \mathrm{var}\big\{\mathrm{CV}_1(\hat{r}^{-(1)}), \ldots \mathrm{CV}_K(\hat{r}^{-(K)})\big\},$$

  (where $\mathrm{var}(\cdot)$ denotes the sample variance operator) and hence

$$\frac{1}{\sqrt{K}} \mathrm{sd}\big\{\mathrm{CV}_1(\hat{r}^{-(1)}), \ldots \mathrm{CV}_K(\hat{r}^{-(K)})\big\},$$

  for the standard deviation or *standard error* of the cross-validation error estimate (where sd denotes the sample standard deviation)

2.5 Model selection in practice

- To choose between models in practice, we simply compute cross-validated errors for each, and then take the model with the minimum cross-validated error
- For tuning parameter selection (as in k in k-nearest neighbors), write $\hat{r}_\theta$ to denote that our fitting mechanism depends on some parameter θ. Then for a range of parameter values $\theta_1, \ldots \theta_m$ of interest, we compute

  $$\text{CVErr}(\hat{r}_{\theta_1}), \ldots \text{CVErr}(\hat{r}_{\theta_m}),$$

  and choose the value of minimizing the cross-validation error curve (a curve over θ),

  $$\hat{\theta} = \underset{\theta \in \{\theta_1, \ldots \theta_m\}}{\text{argmin}} \ \text{CVErr}(\hat{r}_\theta)$$

- For close calls, pay attention to standard errors! More on this later ...