Regression and Prediction

Brenton Kenkel — PSCI 8357
April 14, 2016

Today we are taking a bit of a left turn. Our goal, instead of testing hypotheses or estimating causal effects, will simply be to predict the response as a function of covariates.

The Problem of Overfitting

Imagine the following *prediction problem*. We are told to build a model of the conditional expectation function $E[Y_i \mid x_i]$ using the data available to us. After we build the model, we will be presented with a random draw of the covariates x_i from the population, and we will use the model to predict the corresponding value of Y_i. This will be new, previously unobserved data—not part of the data we used to build the model. The closer our prediction is to the actual value, the greater our reward will be. How should we select a model to maximize our reward?

The prediction problem is related to, but distinct from, what we have done in most of this course. Up to now, our goal has been to estimate and test hypotheses about parameters of the regression function. We have usually relied on the linear model,

$$E[Y_i \mid x_i] = x_i^\top \beta,$$

and formulated hypotheses in terms of β. Getting β right, in terms of unbiasedness and consistency, has been a primary concern. In the prediction problem, however, we only care about getting β right insofar as it helps us get the prediction right.

Even if we restrict ourselves to OLS, there are many ways to build a predictive model from a given set of data. We might include all the covariates we have, or only a subset of them. We might include interactions and higher-order terms, or only the linear components. How do we select among the possible models?

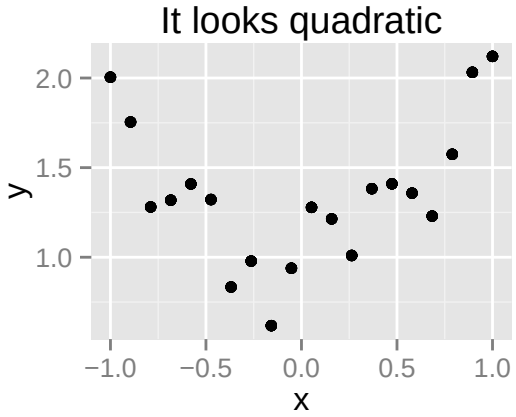
It is tempting to look at how well the model fits the data we used to estimate it. We should resist the temptation. Think about the sample data as consisting

of both signal and noise:

$$Y_i = f(x_i) + \epsilon_i = \text{signal} + \text{noise}.$$

In statistical modeling, including predictive modeling, our goal is to extract the signal. We don't want to generalize from features of the data that are due to chance or sampling variation. This means we don't want a model that fits the sample data perfectly. The in-sample fit can only be perfect if we're treating the noise in our data as if it were signal.

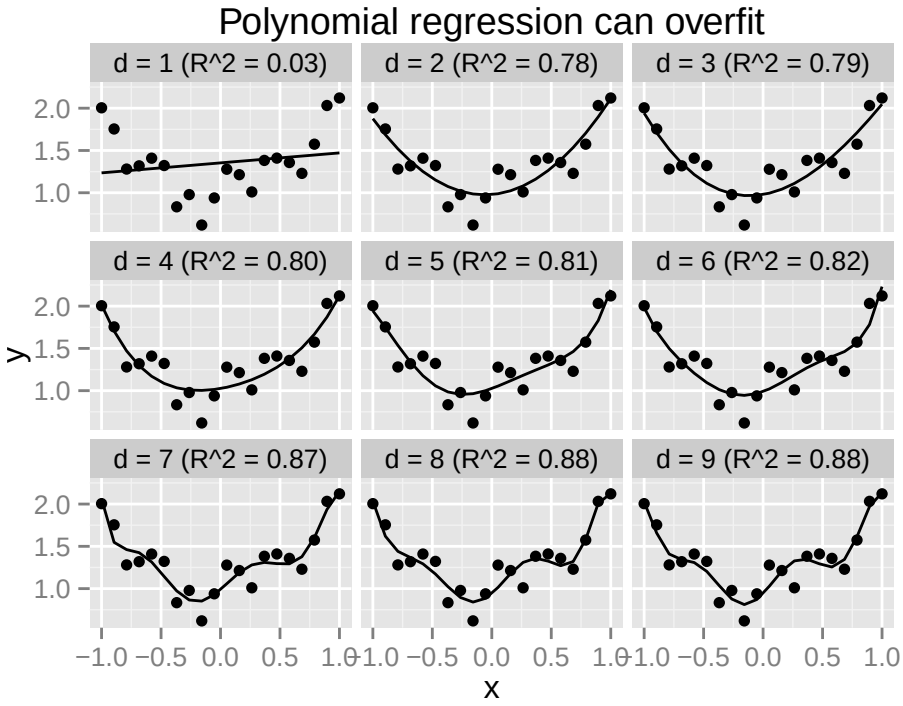
As an example, imagine we have the data plotted below.



We want to build a predictive model of Y_i given our single covariate X_i. The relationship appears to be nonlinear, so we want to fit a polynomial model of the form

$$Y_i = \beta_0 + \sum_{k=1}^{d} \beta_k X_i^k + \epsilon_i.$$

Do we want a linear model? A quadratic model? Something even higher-order? Let's look at the results of fitting dth-order polynomials for $d = 1, \ldots, 9$.



The higher the dimension, the greater the in-sample fit, as measured by R^2. But a more holistic look at these plots, not to mention common sense, suggests that a 9th-degree polynomial might not be the best modeling choice. The high-degree models are essentially connecting the dots. If we were to uncover a new observation from the population, we would probably expect it to fall closer to the parabolic curve from the quadratic model than the overfit curve from the highest-dimension model. How can we formalize this intuition and select the best predictive model?

Cross-Validation

Ideally, we would have a constant inflow of new data that we could use to build and verify predictive models. As new data came in, we would run our models on it and keep the ones with the best predictive performance. Unfortunately, for most of us, new data is expensive to collect or only comes out infrequently.[1]

[1] Though if you're ever at a conference and wondering why so many graduate students are writing papers that use Twitter data for predictive modeling…

How can we use our sample data to estimate the prediction error, or generalizability, of our models?

Cross-validation lets us estimate the prediction error of a model without collecting new data. To calculate the cross-validation estimate of the prediction error of the OLS estimator $\hat{\beta}$:

1. For each observation $i = 1, \ldots, N$:
    1. Regress Y on $\mathbf{X}$ using every observation except the i'th. Let $\hat{\beta}^{(-i)}$ denote the resulting estimate.
    2. Calculate the predicted value for the i'th observation using the model fit using the rest of the data: $\hat{Y}_i = x_i^\top \hat{\beta}^{(-i)}$.
2. Calculate the mean squared error of these "out of sample" predictions:

$$\hat{e}_{\text{CV}} = \frac{1}{N} \sum_{i=1}^{N} (Y_i - \hat{Y}_i)^2.$$

If we have a variety of candidate models and our goal is prediction, we should prefer models with lower cross-validation error.

Returning to our polynomial regression example, here is the cross-validation error of each estimator.

```
##       dim cv_error rank
##  [1,]   1   0.1903    6
##  [2,]   2   0.0433    1
##  [3,]   3   0.0457    2
##  [4,]   4   0.0461    3
##  [5,]   5   0.0662    4
##  [6,]   6   0.1603    5
##  [7,]   7   0.2322    7
##  [8,]   8   0.3271    8
##  [9,]   9   1.6516    9
```

As I've described it, cross-validation entails re-running each candidate model N times. If N is huge, or there are lots of candidate models, or the model takes forever to run, we have a problem. Luckily, for these cases we have *K-fold cross-validation*:

1. Randomly assign each observation $i = 1, \ldots, N$ to a fold $k_i \in \{1, \ldots, K\}$.

2. For each fold $k = 1, \ldots, K$: Regress Y on $\mathbf{X}$ using every observation not in the k'th fold. Let $\hat{\beta}^{(-k)}$ denote the resulting estimate.
3. For each observation $i = 1, \ldots, N$: Calculate the predicted value using the model fit when i's fold was excluded: $\hat{Y}_i = x_i^\top \hat{\beta}^{(-k_i)}$.
4. Calculate the mean squared error of the out-of-fold predictions:

$$\hat{e}_{\text{CV}} = \frac{1}{N} \sum_{i=1}^{N} (Y_i - \hat{Y}_i)^2.$$

Typical choices are $K = 5$ or $K = 10$. What we looked at before was the special case of $K = N$, also known as *leave-one-out cross-validation*.

Ridge Regression and the LASSO

There are more algorithms for predictive modeling than you could learn in a lifetime. Hastie, Tibshirani, and Friedman (2009) provide an overview. For our purposes today, we will focus on two predictive algorithms that generalize OLS: ridge regression (Hoerl and Kennard 1970) and the least absolute shrinkage and selection operator, or LASSO (Tibshirani 1996). The typical use case for both of these is that the number of covariates is "large" relative to the sample size, in which case the OLS estimator either does not exist ($p > N$) or is too highly variable to make reliable predictions.

Remember that OLS solves the least-squares problem

$$\min_{\beta \in \mathbb{R}^p} \left\{ \sum_{i=1}^{N} (Y_i - x_i^\top \beta)^2 \right\}.$$

Ridge regression modifies the problem to include a penalty for the magnitude of the coefficients:

$$\min_{\beta \in \mathbb{R}^p} \left\{ \sum_{i=1}^{N} (Y_i - x_i^\top \beta)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}.$$

If $\lambda = 0$, ridge regression is identical to OLS. As $\lambda \to \infty$, the ridge regression coefficients go to zero. In between, ridge regression "shrinks" the coefficients toward zero.

We usually scale the covariates to have variance one before running ridge regression, so that the penalty is "fair". Similarly, we either exclude the intercept from the penalty or center the covariates and response to have mean zero so that the intercept can be dropped. The R implementation we will discuss does this for you automatically.

The trick to ridge regression is selecting the penalty parameter λ. This is a Goldilocks problem: our predictions will be too variable if λ is too low and too biased if λ is too high. We can use cross-validation to approximate the best value for prediction:

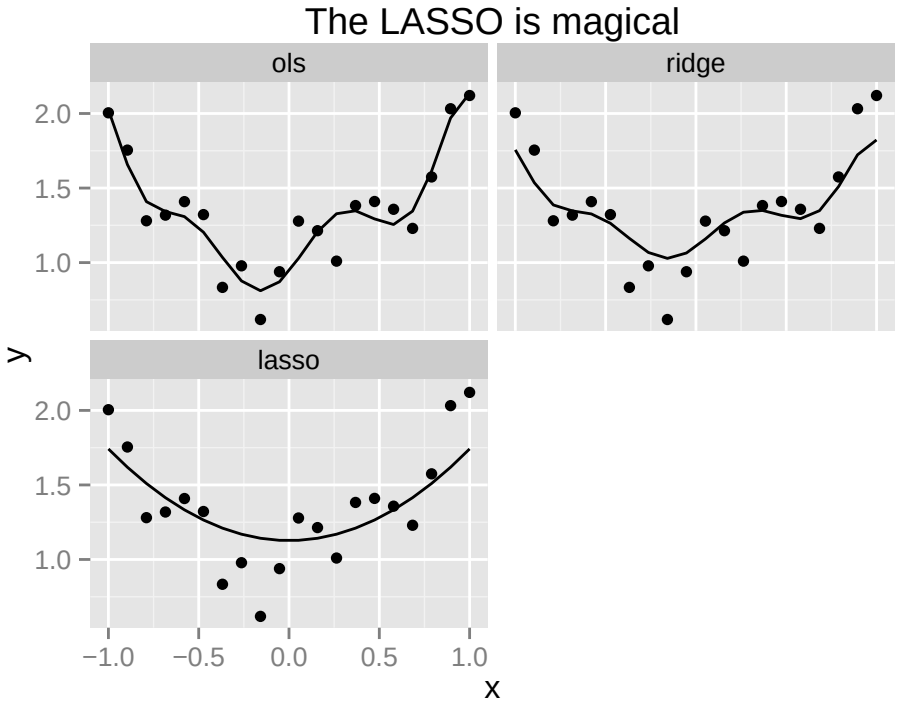
1. Set up a grid of M values $\lambda_1, \ldots, \lambda_M$.
2. For each $m = 1, \ldots, M$: use leave-one-out or K-fold cross-validation to estimate the prediction error of ridge regression with penalty λ_m.
3. Find the value of λ_m with the lowest $\hat{e}_{\text{CV}}(\lambda_m)$. Call it λ_{CV}.
4. Run ridge regression on the full sample using penalty parameter λ_{CV}.

The LASSO solves a similar problem to ridge regression:

$$\min_{\beta \in \mathbb{R}^p} \left\{ \sum_{i=1}^{N} (Y_i - x_i^\top \beta)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\}.$$

As with ridge regression, we typically center and scale the variables and use cross-validation to select λ.

Whereas ridge regression shrinks all of the coefficients toward zero, the LASSO pushes a subset of them all the way to zero. In other words, the LASSO performs automatic model selection for us. The main downside of the LASSO relative to ridge regression is computational; it solves a harder optimization problem and thus takes longer to run. The software has gotten good enough that this doesn't matter. Of course, in any particular application, you may fit both ridge and LASSO estimators and pick whichever has the lowest cross-validation error. If you care about interpretability or parsimony—and who doesn't?—you may prefer the LASSO even if its cross-validation error is slightly higher.




```
##                ols   ridge lasso
## (Intercept)  1.353  1.3533 1.353
## poly(x, 9)1  0.320  0.1921 0.000
## poly(x, 9)2  1.490  0.8931 0.902
## poly(x, 9)3 -0.147 -0.0883 0.000
## poly(x, 9)4  0.183  0.1094 0.000
## poly(x, 9)5  0.203  0.1215 0.000
## poly(x, 9)6  0.139  0.0836 0.000
## poly(x, 9)7 -0.381 -0.2285 0.000
## poly(x, 9)8 -0.181 -0.1086 0.000
## poly(x, 9)9  0.109  0.0655 0.000
```

Standard Errors

Ridge regression and the LASSO are estimators, which means they have sampling distributions and associated standard errors. Unfortunately, their standard errors are not nearly as easy to estimate from data as those of OLS are.

There is no nice formula to plug in.[2] What do we do, then, to quantify our uncertainty about ridge or LASSO estimates?

1. **Just don't do it.** Usually, though not always, our interest in standard errors stems from our interest in hypothesis testing. The best model for prediction is probably not the best one for testing hypotheses. For the latter task, it is safer to stick with an estimator like OLS whose statistical properties (particularly small-sample properties) are well understood.

   Predictive models serve a different purpose than models intended for hypothesis testing. You may not be testing a hypothesis if your goal is to predict Y_i well as a function of x_i. To take an example from my own research, Rob Carroll and I have a working paper (Carroll and Kenkel 2016) in which we measure material military power by finding the function of military capabilities that best predicts militarized dispute outcomes. Since our task is purely predictive, we don't worry about calculating measures of uncertainty or performing hypothesis tests—we have no hypotheses to test! But we do worry a lot about making sure we're not overfitting, and we carefully use cross-validation to measure the out-of-sample predictive accuracy of our model.

2. **Use computationally intensive methods.** Sometimes you might use a predictive model even if your main goal is to test a hypothesis. For example, you might have almost as many covariates as observations and worry that OLS will have gigantic standard errors. In situations like this, you might use a predictive model like the LASSO as a means to a non-predictive end.

   Since there are no canned formulas available, you will need to use computationally intensive methods to approximate the standard errors of estimators like cross-validated ridge regression or LASSO. These methods typically involve repeatedly resampling from your data and recalculating the estimator on each resample. The simplest method that I know of that works for model selection problems is Efron (2014).

[2] For ridge regression, formulas for the standard errors are available for cases when λ is chosen *a priori*. In this case, ridge regression is a linear estimator. In most real-world applications, however, you'll be selecting λ via cross-validation or some other data-driven method.

Appendix: Implementation

To illustrate cross-validation, ridge regression, and the LASSO, we will return to our old friend, the occupational prestige dataset.

```
library("car")
data(Prestige)
head(Prestige)
```

```
##                     education income women prestige census type
## gov.administrators       13.1  12351 11.16     68.8   1113 prof
## general.managers         12.3  25879  4.02     69.1   1130 prof
## accountants              12.8   9271 15.70     63.4   1171 prof
## purchasing.officers      11.4   8865  9.11     56.8   1175 prof
## chemists                 14.6   8403 11.68     73.5   2111 prof
## physicists               15.6  11030  5.13     77.6   2113 prof
```

To make our lives easier, we will remove all missing observations from the dataset. (Next week, we'll talk about better ways to deal with missing data.)

```
sum(is.na(Prestige))
```

```
## [1] 4
```

```
Prestige <- na.omit(Prestige)
```

First, we'll use OLS to estimate the conditional expectation function, then we'll cross-validate to estimate its prediction error. To make things interesting, we'll estimate a three-way interactive model: occupational prestige as a function of the interaction of education, income, and percentage of women.

```
fit_ols <- lm(prestige ~ education * income * women,
              data = Prestige)
summary(fit_ols)
```

```
##
## Call:
## lm(formula = prestige ~ education * income * women, data = Prestige)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -17.769  -5.057   0.398   4.983  14.864
```

```
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)            -2.74e+01   9.95e+00   -2.76   0.0070
## education               5.89e+00   8.67e-01    6.79  1.2e-09
## income                  4.27e-03   1.39e-03    3.07   0.0028
## women                   1.19e-01   3.06e-01    0.39   0.6992
## education:income       -2.37e-04   1.04e-04   -2.29   0.0242
## education:women        -1.88e-02   2.59e-02   -0.73   0.4700
## income:women            1.85e-05   7.87e-05    0.23   0.8148
## education:income:women  4.76e-07   5.85e-06    0.08   0.9353
##
## Residual standard error: 7.31 on 90 degrees of freedom
## Multiple R-squared:  0.83,   Adjusted R-squared:  0.817
## F-statistic: 62.9 on 7 and 90 DF,  p-value: <2e-16
```

To cross-validate, we will use a loop. In each iteration of the loop, we'll fit the model to all but the i'th row of the data. Then we'll compare the prediction for the i'th response to the actual value.

```
library("foreach")

cv_ols <- foreach (i = 1:nrow(Prestige), .combine = "c") %do% {
    dat_i <- Prestige[-i, ]
    fit_i <- lm(prestige ~ education * income * women,
                data = dat_i)
    pred_i <- predict(fit_i, newdata = Prestige[i, ])

    (pred_i - Prestige$prestige[i])^2
}

mean(cv_ols)
```

```
## [1] 62.7
```

What this means is that the average squared distance between prediction and reality is 62.7. To make this more interpretable, we can take the square root:

```
sqrt(mean(cv_ols))
```

```
## [1] 7.92
```

On average, our out-of-sample prediction of occupational prestige using this model will be off by about 7.9.

To run ridge regression and the LASSO with λ selected via cross-validation, we will use the excellent **glmnet** package.

```
library("glmnet")
```

We will rely on the function `cv.glmnet()`, which automatically sets up a grid of values of λ to cross-validate over and selects the best one. Unfortunately, this function does not use the "formula notation" `y ~ x1 + x2 + ...` familiar to us from `lm()` and friends. Instead, it takes the matrix `x` of covariates and the response vector `y` as individual arguments. So we will need to construct these ourselves. The `model.matrix()` function lets us translate a formula into a matrix. We add a `-1` to the formula we used before, since the **glmnet** functions will automatically add an intercept for us.

```
X <- model.matrix(prestige ~ education * income * women - 1,
                  data = Prestige)
colnames(X)
```

```
## [1] "education"              "income"
## [3] "women"                  "education:income"
## [5] "education:women"        "income:women"
## [7] "education:income:women"
```

```
Y <- Prestige$prestige
```

Now we can use `cv.glmnet()` to perform K-fold cross-validation. For leave-one-out cross-validation, we would set `nfolds` to equal the number of rows in the dataset. In the interest of computation time, we will use $K = 5$.

The `alpha` argument controls whether we use ridge regression or the LASSO. For ridge regression, set `alpha = 0`; for the LASSO, set `alpha = 1` (the default).[3]

```
fit_ridge <- cv.glmnet(x = X, y = Y, alpha = 0, nfolds = 5)
fit_lasso <- cv.glmnet(x = X, y = Y, alpha = 1, nfolds = 5)
```

[3] Values between 0 and 1 allow for a mix of the ridge and LASSO penalties, known as the *elastic net*.

We can use the `coef()` method to recover the coefficients associated with the λ that has the lowest cross-validation error.

```
coef_ols <- coef(fit_ols)
coef_ridge <- coef(fit_ridge, s = "lambda.min")
coef_lasso <- coef(fit_lasso, s = "lambda.min")

cbind(coef_ols,
      coef_ridge,
      coef_lasso)
```

```
## 8 x 3 sparse Matrix of class "dgCMatrix"
##                          coef_ols         1        1
## (Intercept)            -2.74e+01  2.00e+00 -5.60412
## education               5.89e+00  3.36e+00  4.16046
## income                  4.27e-03  8.42e-04  0.00116
## women                   1.19e-01 -6.10e-02  .
## education:income       -2.37e-04  3.15e-05  .
## education:women        -1.88e-02  1.38e-04  .
## income:women            1.85e-05  6.29e-06  .
## education:income:women  4.76e-07  8.94e-07  .
```

References

Carroll, Robert J., and Brenton Kenkel. 2016. "Capability Ratios Predict Nothing." http://doe-scores.com.

Efron, Bradley. 2014. "Estimation and Accuracy After Model Selection." *Journal of the American Statistical Association* 109 (507): 991–1007.

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman. 2009. *The Elements of Statistical Learning*. 2nd ed. New York: Springer.

Hoerl, Arthur E, and Robert W Kennard. 1970. "Ridge Regression: Biased Estimation for Nonorthogonal Problems." *Technometrics* 12 (1): 55.

Tibshirani, Robert. 1996. "Regression Shrinkage and Selection via the Lasso." *Journal of the Royal Statistical Society. Series B (Methodological)* 58 (1): 267–88.