Lecture 2: Linear methods for prediction

Alan Lee

Department of Statistics
STATS 784 Lecture 2

July 28, 2017

Outline

Today's agenda

In this lecture begin our discussion of predictive analytics. We discuss linear methods for prediction, although many of the concepts covered will be applicable to other methods. Some of the ideas should be familiar from other courses, particularly STATS 330/762. Today we will cover

- Linear models
- Least squares
- Prediction error

We will use the California housing data as a running example.

See *The Elements of Statistical Learning*, Ch3, *Introduction to Statistical Learning* Ch 3.

Basic prediction setup

y : The output (response variable, target), the quantity to be predicted, assumed numeric for today's lecture;

$x_1, \ldots, x_k$: The inputs (explanatory variables, features) the variables used for prediction;

$\hat{f}$: the predictor, the mathematical function used to combine the inputs to produce the prediction.

We assume that there is a relationship of the form

$$y = f(x_1, \ldots, x_k) + \text{error}$$

that connects the inputs to the outputs, but f is unknown. The errors are assumed to be unpredictable. We select our predictor $\hat{f}$ from some class of predictors. In today's lecture we assume this class is the class of *linear functions*.

Linear predictors

This is the simplest prediction method, but is often surprisingly competitive with the more sophisticated methods we will start to discuss next week. Linear predictors are linear functions of the form

$$f(x_1, \ldots, x_k) = b_0 + b_1 x_1 + \cdots + b_k x_k$$

so that the predicted target is a linear combination of the features.

We need to choose the best predictor in this class. we do this by choosing the b's to minimize the sum of squared errors

$$(y - b_0 - b_1 x_1 - \cdots - b_k x_k)^2.$$

Choosing the b's

Suppose we have rectangular set of data where columns correspond to variables and rows to cases, which we can represent by

$$(x_{i1}, \ldots, x_{ik}, y_i), \ i = 1, 2, \ldots, n$$

where the subscript i indexes the cases, $x_{i1}, \ldots, x_{ik}$ are the values of the inputs for the ith case, and y_i is the output for the ith case. Our problem is to minimize the sum of squared errors

$$\sum_{i=1}^{n} (y_i - b_0 - b_1 x_{i1} - \cdots - b_k x_{ik})^2$$

as a function of the coefficients $b_0, b_1, \ldots, b_k$. This is a standard mathematical problem and very robust computer algorithms exist for solving it. See e.g. STATS 310 and STATS 760 for more details.

Computing the predictor

If $\hat{b}_0$, $\hat{b}_1$,..., $\hat{b}_k$ are the minimizing values, then the predictor for new data $x_1, \ldots, x_k$ is

$$\hat{y} = \hat{b}_0 + \hat{b}_1 x_1 + \cdots + \hat{b}_k x_k.$$

The $\hat{b}$'s (and many other things) are computed by the R function `lm`. A generic function call (for $k = 4$) is

```
my.model <- lm(y~x1+x2+x3+x4, data=data.df)
```

where the data is in a data frame `data.df`, containing inputs `x1`, `x2`, `x3`, `x4` and output `y`. The coefficients $\hat{b}_0$,$\hat{b}_1$,...,$\hat{b}_k$ are computed by

```
coefficients(my.model)
```

The California data

The California dataset contains information on 20,640 California "block groups". We will use 10,000 for this example, holding the rest back for future use. The variables are

medval	:	The median value of houses in the block group (the output);
medinc	:	The median income of residents of the block group;
medage	:	The median age of residents of the block group;
rooms	:	The total number of rooms in houses in the block group;
bedrooms	:	The total number of bedrooms in houses in the block group;
pop	:	The population of the block group;
house	:	The number of households in the block group;
lat	:	The latitude of the block group;
long	:	The longitude of the block group.

The 10,000 block groups are in the data frame `california.df`.

Fitting the linear model

We want to develop a predictor to predict the log of the median value. To fit the model (using log(medval) as the output and the other variables as inputs), and calculate the coefficients, type

```
california <- lm(log(medval)~ medinc + medage + rooms +
      bedrooms + pop + house + lat + long, data = california.df)
```

```
> coefficients(california)
   (Intercept)         medinc         medage          rooms       bedrooms
-1.139141e+01  1.777315e-01  3.554765e-03 -3.022942e-05  4.675074e-04
          pop          house            lat           long
-1.947238e-04  3.077543e-04 -2.762629e-01 -2.716643e-01
```

Making predictions

In the example, we held out 10,640 observations. We will use these data to make some predictions. The new data are in the data frame `newCalifornia.df`. To predict the log median income for these 10,640 block groups, we type

```
> predictions <- predict(california, newdata = newCalifornia.df)
> head(predictions)
       1        2        3        4        5        6
12.98445 12.34761 12.20270 12.17361 12.08313 12.09673
> log(newCalifornia.df$medval[1:6])
[1] 13.02276 12.47266 12.54789 12.27139 12.16160 11.93492
```

Prediction error

How well have we predicted the house prices? In other words, what sort of error do we expect? One measure of this error is the **expected squared error**, the average squared error over a large number of future predictions. If we have some new data (a "test set") where we know the actual output (as we did for the California data) , we can calculate the average squared error directly. For the California data we get

```
> actuals <- log(newCalifornia.df$medval)
> mean((predictions - actuals)^2)
[1]  0.1181294
```

Thus, our estimate of the expected squared error is about 0.1181. Note that this is measured on the log scale. This estimate is known as the **test set estimate** or the **out-of-sample** estimate.

Why not use the original data?

Could we use the training set (the data used to fit the model) to estimate the expected squared error? This is called the **training error** estimate or the **apparent error**. To do this, we could type

```
> mean((predict(california) - log(california.df$medval))^2)
[1]0.1131505
```

The apparent error estimate is a bit smaller, which is typical: in most cases it **underestimates** the expected squared error.

Prediction and models: generalities

Suppose we have a target y and features x, related by an (unknown) function f, plus some random noise, so that

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

The function f is arbitrary, we make no assumptions about it. We have training data $Z = (x_i, y_i), i = 1, \ldots, n$, which we use to fit some model and construct a predictor $\hat{f}_Z$ of f. At this stage we will assume that $\hat{f}_Z$ is a linear function, obtained by fitting a linear model as previously discussed.

Future y's with features x will be predicted by $\hat{f}_Z(x)$. The Z subscript indicates the training data and emphasizes that $\hat{f}_Z$ is a function of the training data. **Note that this assumes that the future data are generated by the same mechanism as the training data.**

Prediction error: theory

The **conditional prediction error** (that is assuming a fixed training set Z and squared error loss) is

$$PE(Z) = E[(Y_0 - \hat{f}_Z(X_0))^2]].$$

Here, the expectation is with respect to a single future observation $Z_0 = (X_0, Y_0)$, so we are averaging the squared error over all future observations. If we have a test set, the test error estimate is an unbiased estimator of the conditional PE. Note that this assumes that new observation comes from the same distribution as the training data.

If we average the conditional prediction error over all possible training sets, we get the **unconditional prediction error**

$$PE = E_Z[PE(Z)].$$

Prediction error: more theory

A third type of prediction error is when we want to predict at some fixed value of the features, say x_0. If the corresponding target is Y_0, we may be interested in the error when predicting Y_0, but averaged over all training sets: This is

$$PE = E_Z[PE(Z)]$$

where now $PE(Z)$ is $E[(Y_0 - \hat{f}_Z(x_0))^2]]$ i.e. the features in the test observation are now fixed. In this case we can split up the error in a nice way:

$$\left[Y_0 - \hat{f}_Z(x_0)\right] = [Y_0 - E[Y_0]] + \left[E(Y_0) - E[\hat{f}_Z(x_0)]\right] + \left[E[\hat{f}_Z(x_0)] - \hat{f}_Z(x_0)\right].$$

Prediction error: still more theory

Since these three components are uncorrelated, we get (squaring and taking expectations over both Y_0 and Z)

$$\begin{aligned} E\left[(Y_0 - \hat{f}_Z(x_0))^2\right] &= E\left[(Y_0 - E[Y_0])^2\right] + \left[(E(Y_0) - E[\hat{f}_Z(x_0)])^2\right] + \mathrm{Var}(\hat{f}_Z(x_0)) \\ &= \sigma^2 + \mathrm{BIAS}^2 + \mathrm{Variance}. \end{aligned}$$

The quantity σ^2 is the essential variability of the target (at features x_0) and doesn't depend on the prediction formula we select. Sometimes called the "irreducible error".

The other two terms do depend on the prediction formula, and can be made smaller by a good choice of model. Typically, models that are too simple will have large bias and small variance, while models that are too complex will have small bias and large variance. This is called the Bias/Variance tradeoff.

Example: polynomial regression

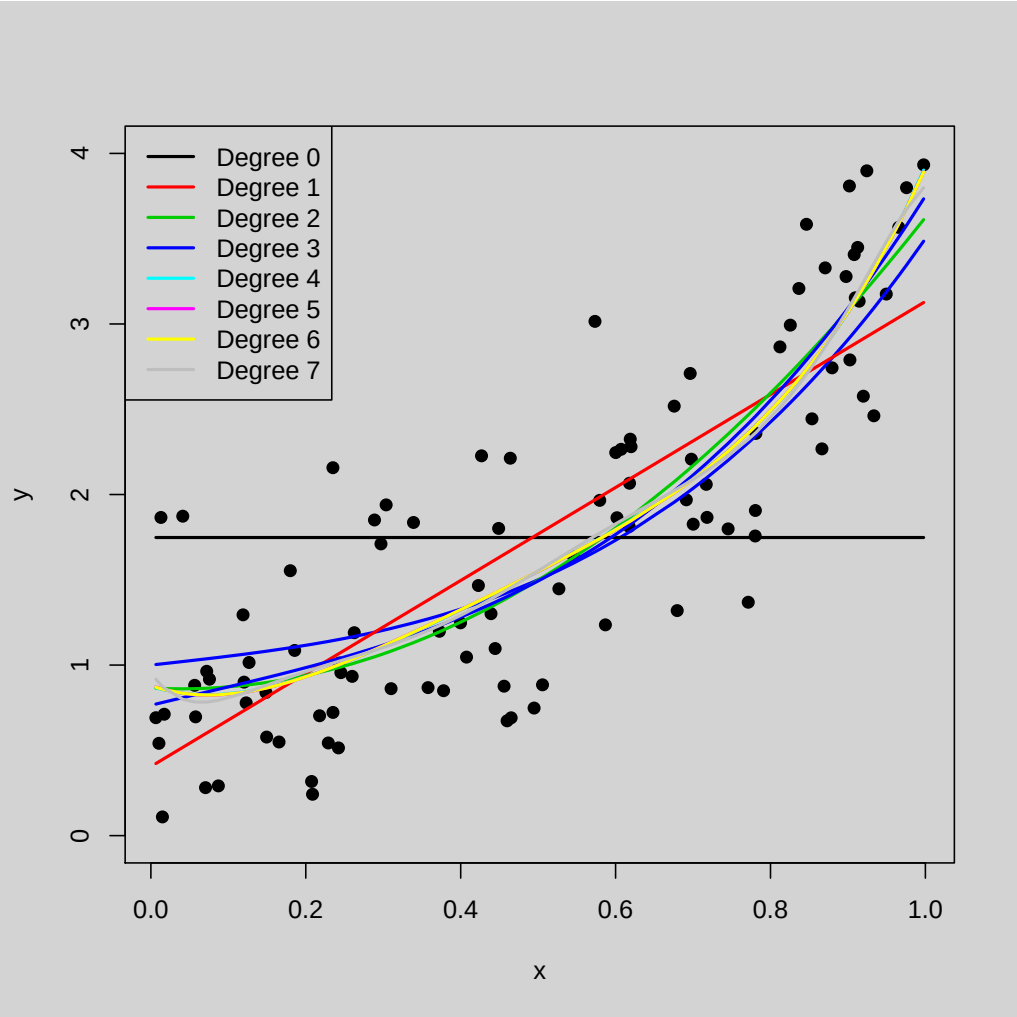
Suppose we have some data (100 data points) which follow an (unknown) cubic model

$$Y = 1 + 0.5x + 2x^3 + N(0, 1)$$

where the x's have a uniform distribution on $[0, 1]$. We want to make a prediction of the value of the target Y_0 when $x_0 = 0.7$.

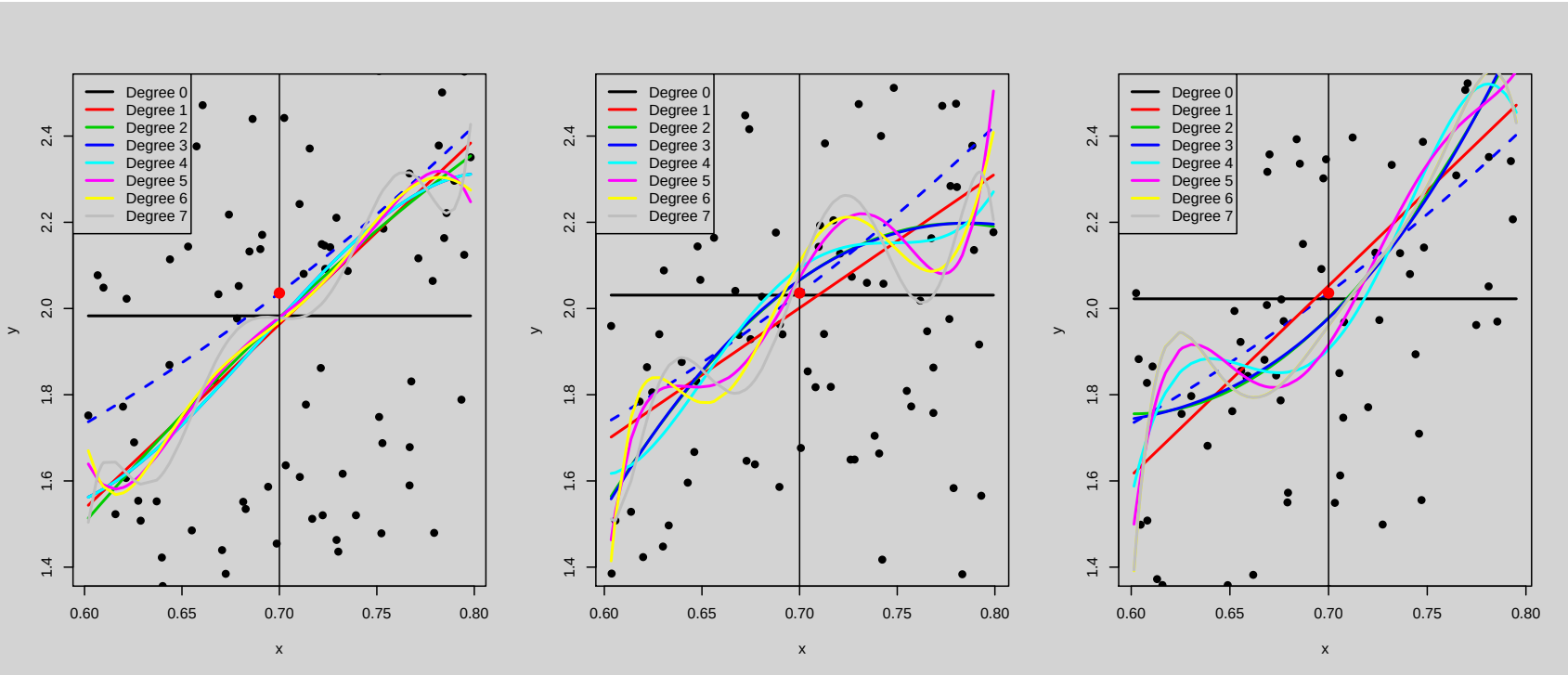
We could consider predictors based on fitting polynomials of degree $0, 1, 2 \ldots, 7$. In the figure on the next slide, we have plotted 100 data points, and the results of fitting these 8 polynomials.

The plot


Degree 0
Degree 1
Degree 2
Degree 3
Degree 4
Degree 5
Degree 6
Degree 7
y
x
0
1
2
3
4
0.0
0.2
0.4
0.6
0.8
1.0

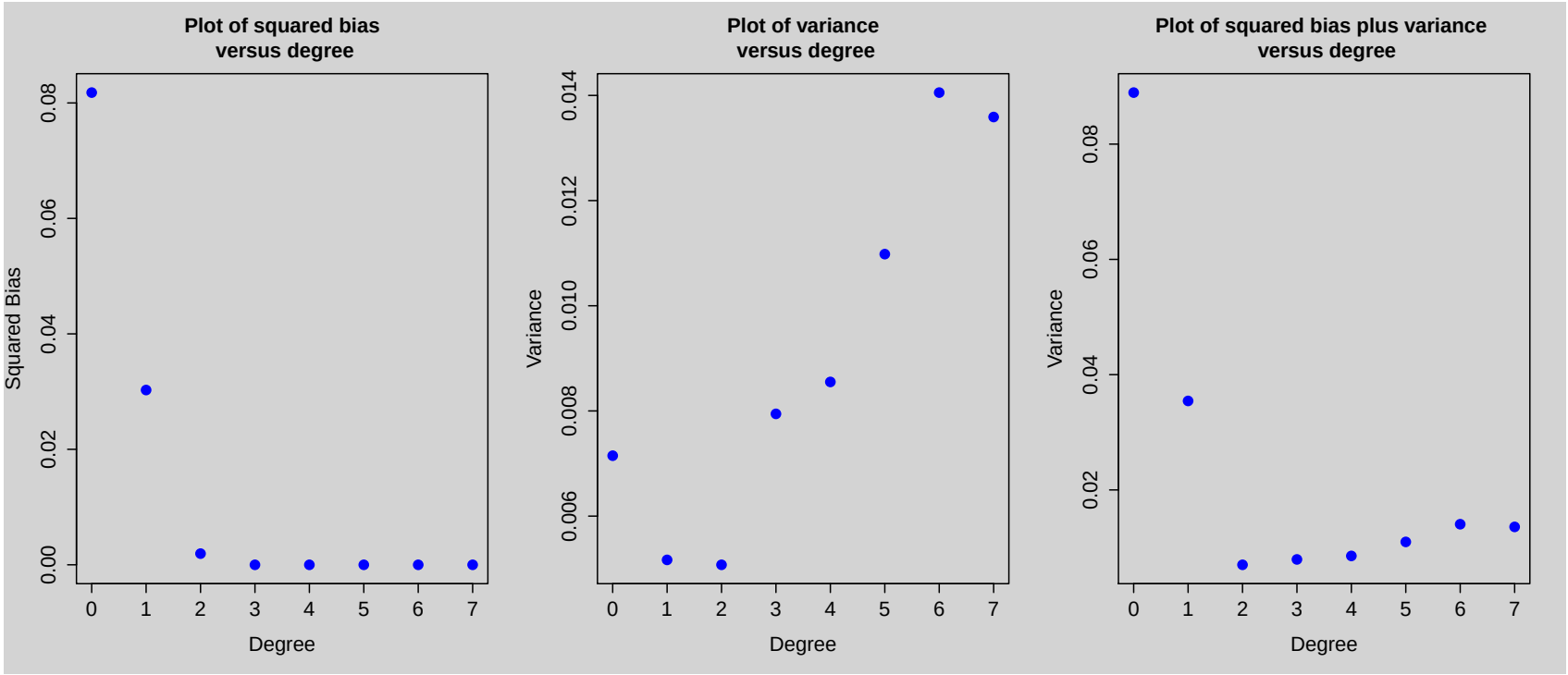
Bias and variance

Here we plot three realisations of 100 data points with fitted lines over the range (0.6, 0.8):



Blue dashed line is correct model.

Bias and variance (Cont)

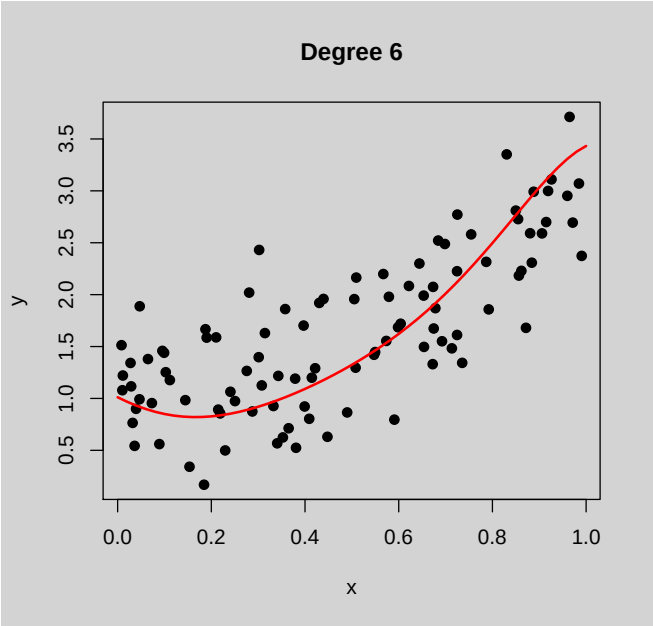


Points to note:

- If the model is too simple (i.e. degree 0 or 1) the variance is less but the bias is big,
- If the model is too complex (i.e. degree 5, 6, 7) the bias is less but the variance is big,
- Interestingly, the model with the smallest PE was degree 2 (not degree 3) The 2-degree model has a sufficiently small bias to offset the larger variance of the 3-degree model.

Apparent error and test error

Lets revisit a 6 degree polynomial (shown as a red solid line).



The apparent error estimate for this data set is 0.208, but the estimate of out-of sample error (based on a test set of 100) is 0.234. The true conditional PE (averaging over all test sets of size 100) is about 0.268.

Note that the estimate of out-of sample error is not very good here: it is quite variable.

References

1. Hastie, T., Tibshirani, R.J. and Friedman, J. (2009). *The Elements of Statistical Learning, 2nd Ed*. Springer.
2. James, G., Witten, D., Hastie. and Tibshirani, R.J. (2013). *An Introduction to Statistical Learning*. Springer.
3. Kuhn, M. and Johnson, K. (2013). *Applied Predictive Modelling*. Springer.
4. Larose, D and Larose C. (2014). *Discovering knowledge in data : an introduction to data mining*, 2nd Ed. Wiley.
5. Witten, I., Frank E., and Hall, M. (2011). *Data mining : practical machine learning tools and techniques*, 3rd Ed. Morgan Kaufmann.