Practical Statistical Learning

John Marden and Feng Liang

2022-09-22

Contents

Preface

These notes are based on a course in statistical learning developed by John Marden and Feng Liang at UIUC using the text **The Elements of Statistical Learning** by Hastie, Tibshirani and Friedman https://hastie.su.domains/ElemStatLearn/.

Chapter 1

Introduction

Notes [lec_W1.1_Introduction_Statistical_Learning.pdf] [lec_W1.2_kNN_vs_LinearRegression.pdf] [lec_W1.2_kNN_vs_LinearRegression_figs.pdf] [lec_W1.3_Introduction_LearningTheory.pdf]

R/Python Code: [Rcode_W1_SimulationStudy.html] [Rcode_W1_Examples_from_ESL.html] [Python_W1_SimulationStudy.html]

1.1 Introduction to Statistical Learning

What is machine learning?

> In artificial intelligence, machine learning involves some kind of machine (robot, computer) that modifies its behavior based on experience. For example, if a robot falls down every time it comes to a stairway, it will learn to avoid stairways. E-mail programs often learn to distinguish spam from regular e-mail. In statistics, machine learning uses statistical data to learn.

What is data mining?

> Looking for relationships in large data sets. Observations are "baskets" of items. The goal is to see what items are associated with other items, or which items' presence implies the presence of other items. For example, at Walmart, one may realize that people who buy socks also buy beer. Then Walmart would be smart to put some beer cases near the socks, or vice versa. Or if the government is spying on everyone's e-mails, certain words (which I better not say) found together might cause the writer to be sent to Guantanamo.

The **difference** for a statistician between supervised machine learning and regular data analysis is that in machine learning, the statistician does not care about the estimates of parameters nor hypothesis tests nor which models fit best. Rather, the focus is on finding some function that does a good job of predicting y from x. Estimating parameters, fitting models, etc., may indeed be important parts of developing the function, but they are not the objective.

1.1.1 Types of learning problems

Generally, there are two categories:

- **Supervised learning** data consists of example (y, x)'s, the training data. The machine is a function built based on the data that takes in a new x, and produces a guess of the corresponding y. It is
  - **regression** if the y's are continuous, and
  - **classification** or **categorization** if the y's are categories.
- **Unsupervised learning** is **clustering**. The data consists of example x's, and the machine is a function that groups the x's into clusters.

1.1.2 Challenge of supervised learning

1.1.3 Curse of dimensionality

1.1.4 A glimpse of learning theory

1.1.5 Bias and variance tradeoff

1.2 Least squares vs. nearest neighbors

1.2.1 Introduction to LS and kNN

1.2.2 Simulation study with R

1.2.3 Simulation study with Python

1.2.4 Compute Bayes rule

1.2.5 Discussion

Chapter 2

Linear Regression

To ease into machine learning, we start with regular linear models. There is one dependent variable, the y, and p explanatory variables, the x's. The data, or training sample, consists of n independent observations:

$$(y_1, \mathbf{x}_1), (y_2, \mathbf{x}_2), \dots, (y_n, \mathbf{x}_n).$$

For individual i, y_i is the value of the one-dimensional dependent variable, and $\mathbf{x}_i = (x_{i1}, x_{i2}, \cdots, x_{ip})^t$ is the $p \times 1$ vector of values for the explanatory variables. Generally, the y_i's are continuous, but the x_{ij}'s can be anything numerical, e.g., 0-1 indicator variables, or functions of another variable (e.g., x, x^2, x^3).

The linear model is

$$y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + e_i. \tag{2.1}$$

The β_j's are parameters, usually unknown and to be estimated. The e_i's are the errors or residuals. We will assume that

- The e_i's are independent (of each other, and of the $\mathbf{x}_i$'s);
- $\mathbb{E}[e_i] = 0$ for each i;
- $\mathrm{Var}[e_i] = \sigma_e^2$ for each i.

There is also a good chance we will assume they are normally distributed.

From STAT424 and 425 (or other courses), you know what to do now: estimate the β_j's and σ_e^2, decide which β_j's are significant, do F-tests, look for outliers and other violations of the assumptions, etc.

Here, we may do much of that, but with the goal of prediction. Suppose $(y^*, \mathbf{x}^*)$ is a new point, satisfying the same model and assumptions as above (in particular, being independent of the observed $\mathbf{x}_i$'s). Once we have the estimates of the β_j's (based on the observed data), we **predict** y^* from $\mathbf{x}^*$ by

$$\hat{y}^* = \hat{\beta}_0 + \hat{\beta}_1 x_1^* + \cdots + \hat{\beta}_p x_p^*. \tag{2.2}$$

The prediction is good if $\widehat{y}^*$ is close to y^*. We do not know y^*, but we can hope. But the key point is

> The estimates of the parameters are good if they give good predictions. We don't care if the $\widehat{\beta}_j$'s are close to the β_j's; we don't care about unbiasedness or minimum variance or significance. We just care whether we get good predictions.

2.1 Good predictions: Squared error loss and in-sample error

We want the predictions to be close to the actual (unobserved) value of the dependent variable, that is, we want $\widehat{y}^*$ close to y^*. One way to measure closeness is by using squared error:

$$(y^* - \widehat{y}^*)^2.$$

Because we do not know y^* (yet), we might look at the expected value instead:

$$E[(Y^* - \widehat{Y}^*)^2].$$

But what is that the expected value over? Certainly Y^*, but the Y_i's and $\mathbf{X}_i$'s in the sample, as well as the $\mathbf{X}^*$, could all be considered random. There is no universal answer, but for our purposes here we will assume that the all features $\mathbf{X}$ are fixed, and all the Y_i's are random, i.e.,

$$E[(Y^* - \widehat{Y}^*)^2 \mid \mathbf{X}_1 = \mathbf{x}_1, \ldots, \mathbf{X}_n = \mathbf{x}_n, \mathbf{X}^* = \mathbf{x}^*]. \tag{2.3}$$

But typically you are creating a predictor for many new x's, and likely you do not know what they will be. (You don't know what the next 1000 e-mails you get will be.) A reasonable approach is to assume the new $\mathbf{x}$'s will look much like the old ones, hence you would look at the errors for n new $\mathbf{x}_i$'s being the same as the old ones. Thus we would have n new cases, $(y_i^*, \mathbf{x}_i^*)$, but where $\mathbf{X}_i^* = \mathbf{x}_i$. The n expected errors are averaged, to obtain what is called the **in-sample error**:

$$\text{ERR}_{\text{in}} = \frac{1}{n}\sum_{i=1}^{n} E[(Y_i^* - \widehat{Y}_i^*)^2 \mid \mathbf{X}_1 = \mathbf{x}_1, \ldots, \mathbf{X}_n = \mathbf{x}_n, \mathbf{X}_i^* = \mathbf{x}_i^*]. \tag{2.4}$$

In particular situations, you may have a more precise knowledge of what the new x's would be. By all means, use those values.

We will drop the conditional part of the notation for simplicity.

2.2 Matrices and least-squares estimates

Ultimately we want to find estimates of the parameters that yield a low ERR_{in}. We'll start with the least squares estimate, then translate things to matrices. The estimates of the β_j's depends on just the training sample. The **least squares** estimate of the parameters are the β_j's that minimize the objective function

$$\text{RSS}(\beta_0, \dots, \beta_p) = \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_{i1} - \cdots - \beta_p x_{ip})^2. \tag{2.5}$$

The function is a nice convex function in the β_j's, so setting the derivatives equal to zero and solving will yield the minimum. The derivatives are

$$\frac{\partial}{\partial \beta_0} \text{RSS}(\beta_0, \dots, \beta_p) = -2 \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_{i1} - \cdots - \beta_p x_{ip}); \tag{2.6}$$

$$\frac{\partial}{\partial \beta_j} \text{RSS}(\beta_0, \dots, \beta_p) = -2 \sum_{i=1}^{n} x_{ij} (y_i - \beta_0 - \beta_1 x_{i1} - \cdots - \beta_p x_{ip}), \quad j \geq 1. \tag{2.7}$$

Write the equations in matrix form, starting with

$$\begin{pmatrix} y_1 - \beta_0 - \beta_1 x_{11} - \cdots - \beta_p x_{1p} \\ y_2 - \beta_0 - \beta_1 x_{21} - \cdots - \beta_p x_{2p} \\ \vdots \\ y_n - \beta_0 - \beta_1 x_{n1} - \cdots - \beta_p x_{np} \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} - \begin{pmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1p} \\ 1 & x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{np} \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{pmatrix} \tag{2.8}$$

which is equal to $\mathbf{y} - \mathbf{X}\beta$. The n-by-$(p+1)$ matrix $\mathbf{X}$ is the so-called design matrix.

Take the two summations in equations (2.7) (without the -2's) and set to 0 to get

$$\begin{pmatrix} 1 & 1 & \cdots & 1 \end{pmatrix} (\mathbf{y} - \mathbf{x}\beta) = 0; \tag{2.9}$$

$$\begin{pmatrix} x_{1j} & x_{2j} & \cdots & x_{nj} \end{pmatrix} (\mathbf{y} - \mathbf{x}\beta) = 0, \quad j \geq 1. \tag{2.10}$$

Note that the row vectors in (2.10) on the left are the $(p+1)$ columns of $\mathbf{X}$, yielding

$$\mathbf{X}^t (\mathbf{y} - \mathbf{X}\beta) = 0. \tag{2.11}$$

That equation is easily solved:

$$\mathbf{X}^t \mathbf{y} = \mathbf{X}^t \mathbf{x} \beta \quad \Rightarrow \quad \beta = (\mathbf{X}^t \mathbf{X})^{-1} \mathbf{X}^t \mathbf{y}, \tag{2.12}$$

at least if $\mathbf{X}'\mathbf{X}$ is invertible. If it is not invertible, then there will be many solutions. In practice, one can always eliminate some (appropriate) columns of $\mathbf{X}$ to obtain invertibility. Generalized inverses are available, too.

Summary. In the linear model

$$\mathbf{y} = \mathbf{X}\beta + \mathbf{e}, \tag{2.13}$$

where

$$\beta = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{pmatrix} \quad \text{and} \quad \mathbf{e} = \begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{pmatrix}, \tag{2.14}$$

the least squares estimate of β, assuming $\mathbf{X}^t\mathbf{X}$ is invertible, is

$$\widehat{\beta}_{LS} = (\mathbf{X}^t\mathbf{X})^{-1}\mathbf{X}^t\mathbf{y}. \tag{2.15}$$

2.3 Regression inference

2.4 Geometric interpretation

2.4.1 Basic concepts in vector spaces

2.4.2 LS and projection

2.5 In-sample prediction

When considering the in-sample error for the linear model, we have the same model for the training sample and the new sample:

$$\mathbf{Y} = \mathbf{X}\beta + \mathbf{e} \ \text{ and } \ \mathbf{Y}^* = \mathbf{X}\beta + \mathbf{e}^*. \tag{2.16}$$

The e_i's and e_i^*'s are independent with mean 0 and variance σ_e^2. If we use the least-squares estimate of β in the prediction, we have

$$\widehat{\mathbf{Y}}^* = \mathbf{X}\widehat{\beta}_{LS} = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y} = \mathbf{H}\mathbf{Y}, \tag{2.17}$$

where $\mathbf{H}$ is the "hat" matrix,

$$\mathbf{H} = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'. \tag{2.18}$$

Note that this matrix is symmetric and idempotent, which means that

$$\mathbf{H}^t = \mathbf{H}, \quad \mathbf{H}\mathbf{H} = \mathbf{H}. \tag{2.19}$$

The errors in prediction are the $Y_i^* - \widehat{Y}_i^*$. Before getting to the ERR_{in}, consider the mean and covariance's of these errors. First,

$$E[\mathbf{Y}] = E[\mathbf{X}\beta + \mathbf{e}] = \mathbf{X}\beta, \quad E[\mathbf{Y}^*] = E[\mathbf{X}\beta + \mathbf{e}^*] = \mathbf{X}\beta, \tag{2.20}$$

because the expected values of the e's are all 0 and we are assuming $\mathbf{X}$ is fixed, and

$$\begin{aligned} E[\widehat{\mathbf{Y}}^*] &= E[\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y}] && (2.21)\\ &= \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'E[\mathbf{Y}] && (2.22)\\ &= \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{X}\beta && (2.23)\\ &= \mathbf{X}\beta, && (2.24) \end{aligned}$$

because the $\mathbf{X}'\mathbf{X}$'s cancel. Thus,

$$\mathbb{E}[\mathbf{Y}^* - \widehat{\mathbf{Y}}^*] = \mathbf{0}_n \quad \text{(the } n \times 1 \text{ vector of 0's)}. \qquad (2.25)$$

This zero means that the errors are **unbiased**. They may be big or small, but on average right on the nose. Unbiasedness is ok, but it is really more important to be close.

Next, the covariance matrices:

$$\text{Cov}[\mathbf{Y}] = \text{Cov}[\mathbf{X}\beta + \mathbf{e}] = \text{Cov}[\mathbf{e}] = \sigma_e^2\mathbf{I}_n \quad \text{(the } n \times n \text{ identity matrix)}, \qquad (2.26)$$

because the e_i's are independent, hence have zero covariance, and all have variance σ_e^2. Similarly,

$$\text{Cov}[\mathbf{Y}^*] = \sigma_e^2\mathbf{I}_n. \qquad (2.27)$$

Less similar,

$$\begin{aligned} \text{Cov}[\widehat{\mathbf{Y}}^*] &= \text{Cov}[\mathbf{X}(\mathbf{X}^t\mathbf{X})^{-1}\mathbf{X}^t\mathbf{Y}] && (2.28)\\ &= \mathbf{X}(\mathbf{X}^t\mathbf{X})^{-1}\mathbf{X}^t\text{Cov}[\mathbf{Y}]\mathbf{X}(\mathbf{X}^t\mathbf{X})^{-1}\mathbf{X}^t && (2.29)\\ &= \mathbf{X}(\mathbf{X}^t\mathbf{X})^{-1}\mathbf{X}^t\sigma_e^2\mathbf{I}_n\mathbf{X}(\mathbf{X}^t\mathbf{X})^{-1}\mathbf{X}^t && (2.30)\\ &= \sigma_e^2\mathbf{X}(\mathbf{X}^t\mathbf{X})^{-1}\mathbf{X}^t\mathbf{X}(\mathbf{X}^t\mathbf{X})^{-1}\mathbf{X}^t && (2.31)\\ &= \sigma_e^2\mathbf{X}(\mathbf{X}^t\mathbf{X})^{-1}\mathbf{X}^t && (2.32)\\ &= \sigma_e^2\mathbf{H}, && (2.33) \end{aligned}$$

the second line following from (2.18).

Finally, for the errors, note that $\mathbf{Y}^*$ and $\widehat{\mathbf{Y}}^*$ are independent, because the latter depends on the training sample alone. Hence,

$$\begin{aligned} \text{Cov}[\mathbf{Y}^* - \widehat{\mathbf{Y}}^*] &= \text{Cov}[\mathbf{Y}^*] + Cov[\widehat{\mathbf{Y}}^*] \quad \text{(notice the +)} && (2.34)\\ &= \sigma_e^2\mathbf{I}_n + \sigma_e^2\mathbf{H} && (2.35)\\ &= \sigma_e^2(\mathbf{I}_n + \mathbf{H}). && (2.36) \end{aligned}$$

Now,

$$\begin{aligned} n \cdot \text{ERR}_{\text{in}} &= \mathbb{E}[\|\mathbf{Y}^* - \widehat{\mathbf{Y}}^*\|^2] && (2.37)\\ &= \|\mathbb{E}\mathbf{Y}^* - \mathbb{E}\widehat{\mathbf{Y}}^*\|^2 + \text{tr}(\text{Cov}[\mathbf{Y}^* - \widehat{\mathbf{Y}}^*]) && (2.38)\\ &= \text{tr}(\sigma_e^2(\mathbf{I}_n + \mathbf{H})) && (2.39)\\ &= \sigma_e^2(n + \text{tr}(\mathbf{H})). && (2.40) \end{aligned}$$

The third line follows from (2.36) and (2.25), and the second from the following result: for any random vector $\mathbf{Z}$

$$\begin{aligned}
\mathbb{E}[\|\mathbf{Z}\|^2] &= \mathbb{E}[Z_1^2 + \cdots + Z_m^2] && (2.41)\\
&= \mathbb{E}[Z_1^2] + \cdots + \mathbb{E}[Z_m^2] && (2.42)\\
&= \mathbb{E}[Z_1]^2 + \mathrm{Var}[Z_1]^2 + \cdots + \mathbb{E}[Z_m]^2 + \mathrm{Var}[Z_m]^2 && (2.43)\\
&= \|\mathbb{E}\mathbf{Z}\|^2 + \mathrm{tr}(\mathrm{Cov}[\mathbf{Z}]), && (2.44)
\end{aligned}$$

where the trace of a matrix is the sum of the diagonals, which in the case of a covariance matrix are the variances.

For the trace, recall that $\mathbf{X}$ is $n \times (p+1)$, so that

$$\begin{aligned}
\mathrm{tr}(\mathbf{H}) &= \mathrm{tr}(\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}') && (2.45)\\
&= \mathrm{tr}((\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{X}) && (2.46)\\
&= \mathrm{tr}(\mathbf{I}_{p+1}) = p + 1. && (2.47)
\end{aligned}$$

Putting that answer in (2.40) we obtain

$$\mathrm{ERR}_{\mathrm{in}} = \sigma_e^2 + \sigma_e^2 \frac{p+1}{n}. \qquad (2.48)$$

This expected in-sample error is a simple function of three quantities. We will use it as a benchmark. The goal in the rest of this section will be to find, if possible, predictors that have lower in-sample error.

There's not much we can do about σ_e^2, since it is the inherent variance of the observations. Taking a bigger training sample will decrease the error, as one would expect. The one part we can work with is the p, that is, try to reduce p by eliminating some of the explanatory variables. Will that strategy work? It is the subject of the next chapter.

2.6 Practical issues

2.6.1 Using R

2.6.2 Interpret LS coefficients

2.6.3 Handle categorical variables

2.6.4 Outliers and rank deficiency

Chapter 3

Variable Selection and Regularization

Notes [lec_W3_VariableSelection.pdf] [lec_W3_PCR.pdf] [lec_W3_More_on_Ridge_Lasso.pdf]
R/Python Code [Rcode_W3_VarSel_SubsetSelection] [Rcode_W3_VarSel_RidgeLasso]

3.1 Subset selection

3.2 Ridge regression

3.3 Lasso regression

Chapter 4

Regression Trees

Chapter 5

Nonlinear Regression

Chapter 2 assumed that the mean of the dependent variables was a linear function of the explanatory variables. In this chapter we will consider non-linear functions. We start with just one x-variable, and consider the model

$$Y_i = f(x_i) + e_i, \quad i = 1, \dots, n, \tag{5.1}$$

where the x_i's are given, and the e_i's are independent with mean zero and variances σ_e^2. A linear model would have $f(x_i) = \beta_0 + \beta_1 x_i$. Here, we are not constraining f to be linear, or even any parametric function. Basically, f can be any function as long as it is sufficiently **smooth**. Exactly what we mean by smooth will be detailed later.

From a prediction point of view, the goal is to find an estimated function of f, $\widehat{f}$, so that new y's can be predicted from new x's by $\widehat{f}(x)$. Related but not identical goals include

- **Curve-fitting**: fit a smooth curve to the data in order to have a good summary of the data; find a curve so that the graph "looks nice";
- **Interpolation**: Estimate y for values of x not among the observed, but in the same range as the observed;
- **Extrapolation**: Estimate y for values of x outside the range of the observed x's, a somewhat dangerous activity.

This chapter deals with **nonparametric** functions f, which strictly speaking means that we are not assuming a particular form of the function based on a finite number of parameters. Examples of parametric nonlinear functions:

$$f(x) = \alpha\ e^{\beta x} \ \text{ and } \ f(x) = \sin(\alpha + \beta\ x + \gamma x^2). \tag{5.2}$$

Such models can be fit with least squares much as the linear models, although the derivatives are not simple linear functions of the parameters, and Newton-Raphson or some other numerical method is needed.

The approach we take to estimating f in the nonparametric model is to use some sort of **basis expansion** of the functions on $\mathbb{R}$. That is, we have an infinite set of known functions, $h_1(x), h_2(x), \dots$, and estimate f using a linear combination of a subset of the functions, e.g.,

$$\widehat{f}(x) = \widehat{\beta}_0 + \widehat{\beta}_1 h_1(x) + \cdots + \widehat{\beta}_m h_m(x). \tag{5.3}$$

We are **not** assuming that f is a finite linear combination of the h_j's, hence will always have a biased estimator of f. Usually we *do* assume that f can be arbitrarily well approximated by such a linear combination, that is, there is a sequence $\beta_0, \beta_1, \beta_2, \dots$, such that

$$f(x) = \beta_0 + \lim_{m\rightarrow\infty} \sum_{i=1}^{m} \beta_j h_j(x) \tag{5.4}$$

uniformly, at least for x in a finite range.

An advantage to using estimates as in (5.3) is that the estimated function is a linear one, linear in the h_j's, though not in the x. But with x_i's fixed, we are in the same estimation bailiwick as the linear models in Chapter 2, hence ideas such as subset selection, ridge, lasso, and estimating prediction errors carry over reasonably easily.

In the next few sections, we consider possible sets of h_j's, including polynomials and local polynomials such as cubic splines.

5.1 Polynomials

The estimate of f is a polynomial in x, where the challenge is to figure out the degree. In raw form, we have

$$h_1(x) = x,\ h_2(x) = x^2,\ h_3(x) = x^3, \dots. \tag{5.5}$$

(The Weierstrass Approximation Theorem guarantees that (5.4) holds.) The m^{th} degree polynomial fit is then

$$\widehat{f}(x) = \widehat{\beta}_0 + \widehat{\beta}_1 x + \widehat{\beta}_2 x^2 + \cdots + \widehat{\beta}_m x^m. \tag{5.6}$$

It is straightforward to find the estimates $\widehat{\beta}_j$ using the techniques from Chapter 2, where here

$$\mathbf{X} = \begin{pmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^m \\ 1 & x_2 & x_2^2 & \cdots & x_2^m \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_N & x_N^2 & \cdots & x_N^m \end{pmatrix}. \tag{5.7}$$

Technically, one could perform a regular subset regression procedure, but generally one considers only fits allowing the first m coefficients to be nonzero, and requiring the rest to be zero. Thus the only fits considered are

$$\begin{array}{rcll}
\widehat{f}(x_i) & = & \widehat{\beta}_0 & \text{(constant)} \\
\widehat{f}(x_i) & = & \widehat{\beta}_0 + \widehat{\beta}_1 x_i & \text{(linear)} \\
\widehat{f}(x_i) & = & \widehat{\beta}_0 + \widehat{\beta}_1 x_i + \widehat{\beta}_2 x_i^2 & \text{(quadratic)} \\
 & \vdots & & \\
\widehat{f}(x_i) & = & \widehat{\beta}_0 + \widehat{\beta}_1 x_i + \widehat{\beta}_2 x_i^2 + \cdots + \widehat{\beta}_m x_i^m & (m^{th} \text{ degree}) \\
 & \vdots & &
\end{array} \qquad (5.8)$$

We will use the birthrate data to illustrate polynomial fits. The x's are the years from 1917 to 2003, and the y's are the births per 10,000 women aged twenty-three in the U.S.[1]

Figure **??** contains the fits of several polynomials, from a cubic to an 80^{th}-degree polynomial. It looks like the $m = 3$ and $m = 5$ fits are poor, $m = 20$ to 40 are reasonable, and $m = 80$ is overfitting, i.e., the curve is too jagged.

5.1.1 Model selection

5.1.2 Using R

The $\mathbf{X}$ matrix in (5.7) is not the one to use for computations. For any high-degree polynomial, we will end up with huge numbers (e.g., $87^{16} \approx 10^{31}$) and small numbers in the matrix, which leads to numerical inaccuracies. A better matrix uses *orthogonal polynomials*, or in fact orthonormal ones. Thus we want the columns of the $\mathbf{X}$ matrix, except for the intercept column $\mathbf{1}_n$, to have mean 0 and norm 1, but also to have them orthogonal to each other in such a way that the first m columns still yield the m^{th}-degree polynomials. To illustrate, without going into much detail, we use the Gram-Schmidt algorithm for $\mathbf{x} = (1, 2, 3, 4, 5)'$

[1] The data up through 1975 can be found in the Data and Story Library at http://lib.stat.cmu.edu/DASL/Datafiles/Birthrates.html. See Velleman, P. F. and Hoaglin, D. C. (1981). *Applications, Basics, and Computing of Exploratory Data Analysis.* Belmont. CA: Wadsworth, Inc. The original data is from P.K. Whelpton and A. A. Campbell, "Fertility Tables for Birth Charts of American Women," Vital Statistics Special Reports 51, no. 1. (Washington D.C.:Government Printing Office, 1960, years 1917-1957) and National Center for Health Statistics, Vital Statistics of the United States Vol. 1, Natality (Washington D.C.:Government Printing Office, yearly, 1958-1975). The data from 1976 to 2003 are actually rates for women aged 20-24, found in the National Vital Statistics Reports Volume 54, Number 2 September 8, 2005, *Births: Final Data for 2003*; http://www.cdc.gov/nchs/data/nvsr/nvsr54/nvsr54/_02.pdf.

and $m = 2$. Start with the raw columns,

$$\mathbf{1}_5 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}, \mathbf{x}_{[1]} = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{pmatrix}, \quad \text{and} \quad \mathbf{x}_{[2]} = \begin{pmatrix} 1 \\ 4 \\ 9 \\ 16 \\ 25 \end{pmatrix}. \tag{5.9}$$

Let the first one as is, but subtract the means (3 and 11) from each of the other two:

$$\mathbf{x}_{[1]}^{(2)} = \begin{pmatrix} -2 \\ -1 \\ 0 \\ 1 \\ 2 \end{pmatrix}, \quad \text{and} \quad \mathbf{x}_{[2]}^{(2)} = \begin{pmatrix} -10 \\ -7 \\ -2 \\ 5 \\ 14 \end{pmatrix}. \tag{5.10}$$

Now leave the first two alone, and make the third orthogonal to the second by applying the main Gram-Schmidt step,

$$\mathbf{u} \rightarrow \mathbf{u} - \frac{\mathbf{u}^t\mathbf{v}}{\mathbf{v}^t\mathbf{v}} \cdot \mathbf{v}, \tag{5.11}$$

with $\mathbf{v} = \mathbf{x}_{[1]}^{(2)}$ and $\mathbf{u} = \mathbf{x}_{[2]}^{(2)}$:

$$\mathbf{x}_{[2]}^{[3]} = \begin{pmatrix} -10 \\ -7 \\ -2 \\ 5 \\ 14 \end{pmatrix} - \frac{60}{10} \begin{pmatrix} -2 \\ -1 \\ 0 \\ 1 \\ 2 \end{pmatrix} = \begin{pmatrix} 2 \\ -1 \\ -2 \\ -1 \\ 2 \end{pmatrix}. \tag{5.12}$$

To complete the picture, divide the last two x's by their respective norms, to get

$$\mathbf{1}_5 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}, \mathbf{x}_{[1]}^{Norm} = \frac{1}{\sqrt{10}} \begin{pmatrix} -2 \\ -1 \\ 0 \\ 1 \\ 2 \end{pmatrix}, \quad \text{and} \quad \mathbf{x}_{[2]}^{Norm} = \frac{1}{\sqrt{14}} \begin{pmatrix} -10 \\ -7 \\ -2 \\ 5 \\ 14 \end{pmatrix}. \tag{5.13}$$

You can check that indeed these three vectors are orthogonal, and the last two orthonormal.

For a large n and m, you would continue, at step k orthogonalizing the current $(k+1)^{st}, \ldots, (m+1)^{st}$ vector to the current k^{th} vector. Once you have these vectors, then the fitting is easy, because the $\mathbf{X}_m$ for the m^{th} degree polynomial (leaving out the $\mathbf{1}_n$) just uses the first m vectors, and $\mathbf{X}_m^t\mathbf{X}_m = \mathbf{I}_m$, so that the estimates of beta are just $\mathbf{X}_m^t\mathbf{y}$, and the $\mathbf{H}_m = \mathbf{X}_m\mathbf{X}_m^t$. Using the saturated model, i.e., $(n-1)^{st}$-degree, we can get all the coefficients at once,

$$\widehat{\beta} = \mathbf{X}_{n-1}^t\mathbf{y}, \quad (\widehat{\beta}_0 = \overline{y}). \tag{5.14}$$

Then the coefficients for the m^{th} order fit are the first m elements of $\widehat{\beta}$. Also, the residual sum of squares equals the sum of squares of the left-out coefficients:

$$\mathrm{RSS}_m = \sum_{j=m+1}^{n-1} \widehat{\beta}_j^2, \tag{5.15}$$

from which it is easy to find the residual variances, $\overline{err}^m$'s, and estimated prediction errors.

Unfortunately, polynomials are not very good for extrapolation. Using the two polynomial fits, we have the following extrapolations.

5.2 Regression Splines

5.3 Smoothing Splines

5.4 Sinces and cosines

5.5 A glimpse of wavelets

Chapter 6

Clustering Analysis

Chapter 7

Latent Structure Models

Chapter 8

More on Clustering

Chapter 9

Distriminant Analysis

Chapter 10

Logistic Regression

Chapter 11

Support Vector Machines

Chapter 12

Classification Trees and Boosting

Chapter 13

Recommender System

Chapter 14

Introduction to Deep Learning