Stat 849: Polynomial regression and introduction to model selection

Sündüz Keleş

Department of Statistics
Department of Biostatistics and Medical Informatics
University of Wisconsin, Madison

Polynomial regression

Useful when transformations cannot linearize the relation between the predictors and the response.
General polynomial model with one predictor:
$Y = \beta_0 + \sum_{i=1}^{k} \beta_j X^j + \epsilon$.
Parameter estimation?

Polynomial regression

Useful when transformations cannot linearize the relation between the predictors and the response.
General polynomial model with one predictor:
$Y = \beta_0 + \sum_{i=1}^{k} \beta_j X^j + \epsilon$.
Parameter estimation?
Numerical problems: Different powers of the same variable could be highly correlated, numerical values can easily get very large or very small.
To overcome this, one could normalize X:

$$X^* = \frac{2X - \max(X) - \min(X)}{\max(X) - \min(X)}.$$

A better way is to use orthogonal polynomials.

Consider the model

$$Y_i = \gamma_0 \phi_0(X_i) + \gamma_1 \phi_1(X_i) + \cdots + \gamma_k \phi_k(X_i) + \epsilon_i,$$

where $\phi_r(X_i)$ is an r-th degree polynomial in X and the polynomials are orthogonal over X:

$$\sum_{i=1}^{n} \phi_r(X_i)\phi_s(X_i) = 0, \quad \forall r, s, r \neq s.$$

We have $Y = X\gamma + \epsilon$, where

$$X = \begin{pmatrix} \phi_0(X_1) & \phi_1(X_1) & \cdots & \phi_k(X_1) \\ \phi_0(X_2) & \phi_1(X_2) & \cdots & \phi_k(X_2) \\ \cdots & \cdots & \cdots & \cdots \\ \phi_0(X_n) & \phi_1(X_n) & \cdots & \phi_k(X_n) \end{pmatrix}$$

How about $X^T X$?

$$X^T X = \begin{pmatrix} \sum_i \phi_0^2(X_i) & 0 & \cdots & 0 \\ 0 & \sum_i \phi_1^2(X_i) & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \sum_i \phi_k^2(X_i) \end{pmatrix}$$

Hence,

$$\hat{\gamma}_r = \frac{\sum_i \phi_r(X_i) Y}{\sum_i \phi_r^2(X_i)}, \quad r = 0, \cdots, k.$$

How about obtaining ϕ_r?

Generating orthogonal polynomial basis

Various ways to do so.
Hayes (1974) suggested: Normalize X so that $-1 \leq X_i \leq 1$ and

$$\phi_{r+1}(X) = 2(X - a_{r+1})\phi_r(X) - b_r\phi_{r-1}(X),$$

where $\phi_0(X) = 1$, $\phi_1(X) = 2(X - a_1)$,

$$\begin{aligned} a_{r+1} &= \frac{\sum_{i=1}^n X_i\phi_r^2(X_i)}{\sum_{i=1}^n \phi_r^2(X_i)} \\ b_r &= \frac{\sum_{i=1}^n \phi_r^2(X_i)}{\sum_{i=1}^n \phi_{r-1}^2(X_i)}, \end{aligned}$$

with $b_0 = 0$, $a_1 = \bar{X}$.
In practice, use `poly()`.

Example: 133 observations of acceleration against time for a simulated motorcycle accident (Silverman, 1985).

```
> library(MASS)
> data(mcycle)
> attach(mcycle)
> plot(times, accel)
> lm1 = lm(accel ~ poly(times, 3))
> lines(times, lm1$fitted, lty = 3, col = "red")
> lm2 = lm(accel ~ poly(times, 6))
> lines(times, lm2$fitted, lty = 5, col = "blue")
> legend(40, -100, c("degree = 3", "degree = 6"),
  lty = c(3, 5), col = c("red", "blue"))
```

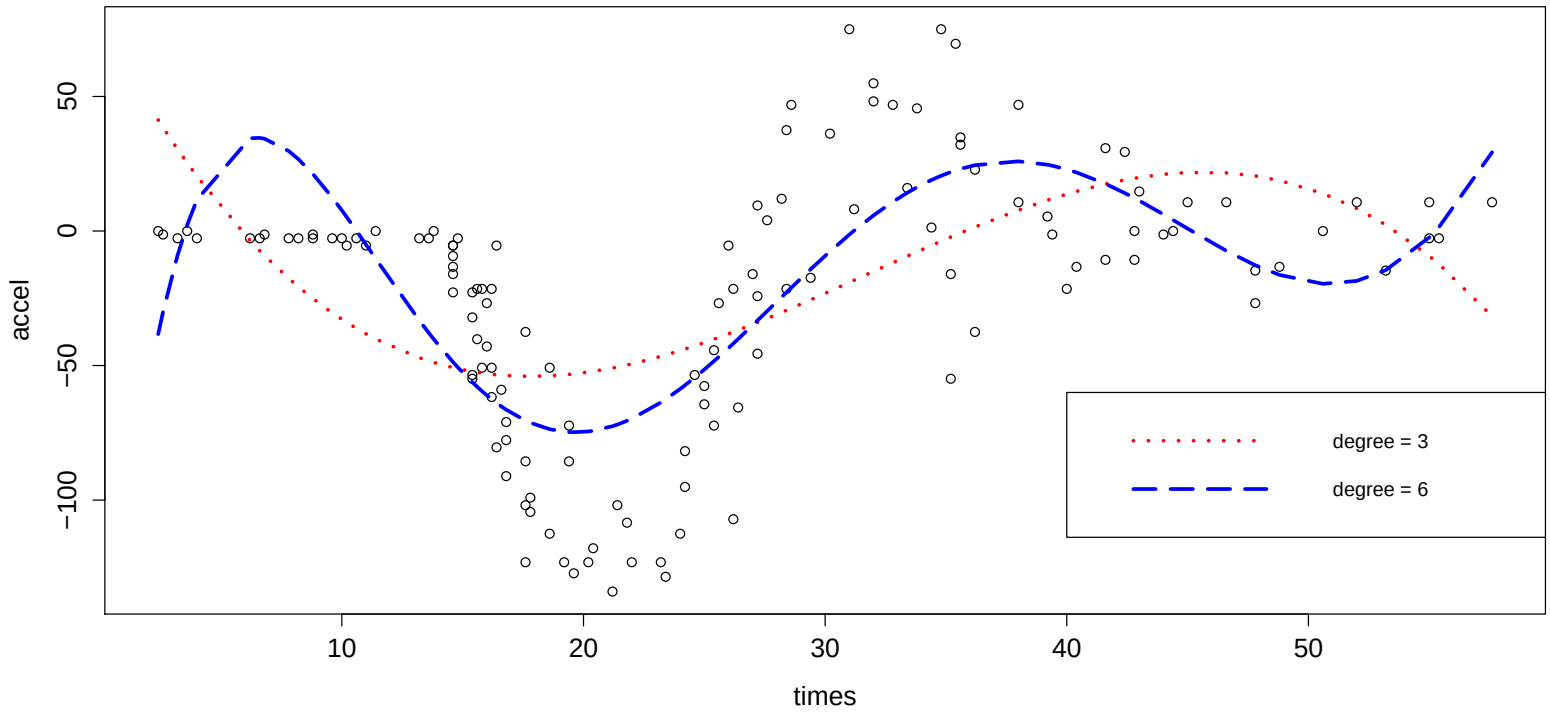

Figure: Polynomial regression.

```
> cbind(times, times^2, times^3)[1:7, ]
      times
 [1,]   2.4  5.76  13.824
 [2,]   2.6  6.76  17.576
 [3,]   3.2 10.24  32.768
 [4,]   3.6 12.96  46.656
 [5,]   4.0 16.00  64.000
 [6,]   6.2 38.44 238.328
 [7,]   6.6 43.56 287.496
> model.matrix(lm1)[1:7, ]
   (Intercept) poly(times, 3)1 poly(times, 3)2 poly(times, 3)3
1            1    -0.150978025     0.228360989   -0.2572341691
2            1    -0.149652433     0.223634171   -0.2473124239
3            1    -0.145675655     0.209670791   -0.2186500006
4            1    -0.143024469     0.200542765   -0.2004466931
5            1    -0.140373284     0.191559455   -0.1829542651
6            1    -0.125791765     0.144738044   -0.0989511020
7            1    -0.123140580     0.136695387   -0.0858056922
```

Is the design matrix orthogonal?

```
> X = model.matrix(lm1)
> t(X[, 1])  %*% X[, 2]
             [,1]
[1,] 1.110223e-15
> t(X[, 1])  %*% X[, 3]
             [,1]
[1,] 1.665335e-16
> t(X[, 1])  %*% X[, 4]
             [,1]
[1,] 6.661338e-16
> t(X[, 2])  %*% X[, 3]
              [,1]
[1,] -5.691384e-17
> t(X[, 2])  %*% X[, 4]
             [,1]
[1,] 1.996287e-17
```

`poly()` function generates orthogonal polynomials which represent a basis for polynomial regression. Orthogonal polynomials are not uniquely defined.

Comparing fits from different order polynomial regressions

```
> anova(lm1, lm2)
Analysis of Variance Table

Model 1: accel ~ poly(times, 3)
Model 2: accel ~ poly(times, 6)
  Res.Df    RSS  Df Sum of Sq      F    Pr(>F)
1    129 206424
2    126 138921   3     67503 20.408 7.645e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Model/variable selection

Why select?

Model/variable selection

Why select?

- In practice, we never know the underlying true variables that are associated with the response of our interest (e.g., true model is unknown to us). Typically, we will be given p variables and we would like to find out which of these we should include in the regression model. Many investigators like a small model, i.e., small number of variables because of its simplicity to explain and understand.

Model/variable selection

Why select?

- In practice, we never know the underlying true variables that are associated with the response of our interest (e.g., true model is unknown to us). Typically, we will be given p variables and we would like to find out which of these we should include in the regression model. Many investigators like a small model, i.e., small number of variables because of its simplicity to explain and understand.
- Often, we might have $p > n$ variables. Can we fit a linear regression model using all these p variables?

Model/variable selection

Why select?

- In practice, we never know the underlying true variables that are associated with the response of our interest (e.g., true model is unknown to us). Typically, we will be given p variables and we would like to find out which of these we should include in the regression model. Many investigators like a small model, i.e., small number of variables because of its simplicity to explain and understand.
- Often, we might have $p > n$ variables. Can we fit a linear regression model using all these p variables?
- We might have variables that are very correlated. We will see that multicollinearity can be a severe problem.

Model/variable selection

Why select?

- In practice, we never know the underlying true variables that are associated with the response of our interest (e.g., true model is unknown to us). Typically, we will be given p variables and we would like to find out which of these we should include in the regression model. Many investigators like a small model, i.e., small number of variables because of its simplicity to explain and understand.
- Often, we might have $p > n$ variables. Can we fit a linear regression model using all these p variables?
- We might have variables that are very correlated. We will see that multicollinearity can be a severe problem.

Model/variable selection

Why select?

- In practice, we never know the underlying true variables that are associated with the response of our interest (e.g., true model is unknown to us). Typically, we will be given p variables and we would like to find out which of these we should include in the regression model. Many investigators like a small model, i.e., small number of variables because of its simplicity to explain and understand.
- Often, we might have $p > n$ variables. Can we fit a linear regression model using all these p variables?
- We might have variables that are very correlated. We will see that multicollinearity can be a severe problem.

The emphasis in the next couple of lectures is model/variable selection.

Multicollinearity: why it might be a problem and what can we do?

Possible remedies:

- Subset selection.

- Shrinkage estimators: Ridge and Lasso (regularization); Principal components regression (PCR), and Partial least squares regression (PLS) (methods using derived input directions).

Multicollinearity: why it might be a problem and what can we do?

Possible remedies:
- Subset selection.
- Shrinkage estimators: Ridge and Lasso (regularization); Principal components regression (PCR), and Partial least squares regression (PLS) (methods using derived input directions).

Overall idea: we will give up some bias (in estimates of β) to gain in variance. Recall that mean squared error of an estimator = $\text{Bias}^2 + \text{Variance}$.

Multicollinearity: why it might be a problem and what can we do?

Possible remedies:
- Subset selection.
- Shrinkage estimators: Ridge and Lasso (regularization); Principal components regression (PCR), and Partial least squares regression (PLS) (methods using derived input directions).

Overall idea: we will give up some bias (in estimates of β) to gain in variance. Recall that mean squared error of an estimator = $\text{Bias}^2 + \text{Variance}$.

First, let's investigate what happens with collinearity with simple examples.

```
x1 = c(2, 8, 6, 10)
x2 = c(6, 9, 8, 10)
y  = c(23, 83, 63, 103)
#Obtain the design matrix
x = cbind(1, x1, x2)
> qr(x)$rank #QR decomposition
[1] 2
xx = t(x)%*%x
> xx
        x1  x2
     4  26  33
 x1 26 204 232
 x2 33 232 281
#R will not report an estimated coefficient for all predictors
> lm(y~x1+x2)
Call: lm(formula = y ~ x1 + x2)
Coefficients:
(Intercept)           x1           x2
          3           10           NA
#Though it will give us the fitted values of Y
> lm1 = lm(y~x1+x2)
> lm1$fitted
 1   2   3   4
23  83  63 103
```

```
#Finding generalized inverse using  SVD
svd1 = svd(xx, nu = 2, nv = 2)
ginv1 = svd1$v%*%diag(1/svd1$d[1:2])%*%t(svd1$u)
             [,1]         [,2]         [,3]
[1,]  0.005247813 -0.01974733  0.01636540
[2,] -0.019747328  0.07628377 -0.06059475
[3,]  0.016365403 -0.06059475  0.05152964

#check whether this satisfies the g-inverse property.
> xx%*%ginv1%*%xx

        x1  x2
     4  26  33
 x1 26 204 232
 x2 33 232 281
```

```
#Another way of obtaining a generalized inverse.
lxx = xx[1:2, 1:2]
a = solve(lxx)
a = cbind(a, 0)
a = rbind(a, 0)
ginv2 = a
> ginv2
                          x1
    1.4571429 -0.18571429 0
x1 -0.1857143  0.02857143 0
    0.0000000  0.00000000 0
#check whether this satisfies the g-inverse property.
> xx%*%ginv2%*%xx

         x1  x2
     4   26  33
 x1 26 204 232
 x2 33 232 281
```

```
hatbeta1 = ginv1%*%t(x)%*%y
hatbeta2 = ginv2%*%t(x)%*%y
> hatbeta1

             [,1]
 [1,] -0.8095238
 [2,]  9.6190476
 [3,]  0.7619048
> hatbeta2

    [,1]
       3
x1    10
       0
```

```
#Let's check the predictions from two fits:
> x%*%hatbeta1

      [,1]
 [1,]   23
 [2,]   83
 [3,]   63
 [4,]  103
> x%*%hatbeta2

      [,1]
 [1,]   23
 [2,]   83
 [3,]   63
 [4,]  103
#This confirms that x*hatbeta  is the same for both set of
parameter estimates.
```

```
#Other coefficient  estimates computed using
#other generalized inverses...
hatbeta3 = matrix(c(-87, 1, 18), nrow = 3)
hatbeta4 = matrix(c(-7, 9, 2), nrow = 3)
> x%*%hatbeta3

      [,1]
 [1,]   23
 [2,]   83
 [3,]   63
 [4,]  103
> x%*%hatbeta4

      [,1]
 [1,]   23
 [2,]   83
 [3,]   63
 [4,]  103
```

Notes

```
> x1
[1]  2  8  6 10
> x2
[1]  6  9  8 10
```

Notes

```
> x1
[1]  2  8  6 10
> x2
[1]  6  9  8 10
```

- We have $x2 = 5 + 0.5 * x1$, perfect correlation (perfect multicollinearity) between the two predictors!

Notes

```
> x1
[1]  2  8  6 10
> x2
[1]  6  9  8 10
```

- We have $x2 = 5 + 0.5 * x1$, perfect correlation (perfect multicollinearity) between the two predictors!
- So, what is the problem ? Can you say which predictor is more important?

Notes

```
> x1
[1]  2  8  6 10
> x2
[1]  6  9  8 10
```

- We have $x2 = 5 + 0.5 * x1$, perfect correlation (perfect multicollinearity) between the two predictors!
- So, what is the problem ? Can you say which predictor is more important?
- What happens when we get a new observation (x_1 and x_2), and we want to predict Y for this observation? E.g. $x_1 = 6$, $x_2 = -3$.

```
#Example 1:
newx = c(1, 6, -3)
> newx%*%hatbeta1

          [,1]
 [1,] 54.61905
> newx%*%hatbeta2

       [,1]
 [1,]   63
> newx%*%hatbeta3
     [,1]
[1,] -135
> newx%*%hatbeta4
      [,1]
[1,]   41
#4 different predictions are obtained!!
```

```
#Example 2:
newx = c(1, 7, 8.5) #note that 5 + 0.5 * 7 = 8.5
> newx%*%hatbeta1

       [,1]
 [1,]   73
> newx%*%hatbeta2

       [,1]
 [1,]   73
> newx%*%hatbeta3
     [,1]
[1,]   73
> newx%*%hatbeta4
     [,1]
[1,]   73
```

If the new observations are following the same collinearity pattern, the predictions are not affected, i.e., predictions for new observations are unique for different estimates of regression coefficients.

Almost perfect multicollinearity

```
n = 100
p = 2
set.seed(1)
#I am not showing you how I generated
#predictors X1 and X2
Y = 1.3+ 0.5 * X[, 1] + 0.05 * X[, 2] + rnorm(n, 0, 1)
```

Now, lets regress Y on X_1 and X_2.

```
> summary(lm(Y~ X))
 Call: lm(formula = Y ~ X)
 Residuals:
       Min         1Q    Median         3Q        Max
 -2.943591 -0.436453  0.002018  0.636917  2.639407
 Coefficients:
             Estimate Std. Error t value Pr(>|t|)
 (Intercept)   1.3254     0.1052  12.599   <2e-16 ***
          X1   0.8963     0.7770   1.154    0.252
          X2  -0.3290     0.7761  -0.424    0.673
 Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
 Residual standard error: 1.043 on 97 degrees of freedom
 Multiple R-Squared: 0.1988,     Adjusted R-squared: 0.1823
 F-statistic: 12.03 on 2 and 97 DF,  p-value: 2.144e-05
```

Notice anything strange?

```
> cor(X[, 1], X[, 2])
[1] 0.988647
```

The paradoxical result is due to the fact that X_1 and X_2 are highly correlated, they essentially convey the same information regarding Y.

```
> summary(lm(y~ X[, 1]))
Call: lm(formula = y ~ X[, 1])
Residuals:
     Min       1Q   Median       3Q      Max
-2.91800 -0.46847 -0.04045  0.64358  2.64128
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   1.3271     0.1047  12.679  < 2e-16 ***
     X[, 1]   0.5707     0.1163   4.908 3.66e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Residual standard error: 1.039 on 98 degrees of freedom
Multiple R-Squared: 0.1973,     Adjusted R-squared: 0.1891
F-statistic: 24.09 on 1 and 98 DF,  p-value: 3.663e-06
```

How about the fitted values?

```
> lm1
 Call: lm(formula = y ~ X)
 (Intercept)            X1            X2
      1.3254        0.8963       -0.3290
 > lm2
 Call: lm(formula = y ~ X[, 1])
 (Intercept)        X[, 1]
     1.3271        0.5707
 > lm3
 Call: lm(formula = y ~ X[, 2])
 (Intercept)        X[, 2]
     1.3323        0.5561
 > cor(lm1$fitted, lm2$fitted)
 [1] 0.9962594
 > cor(lm1$fitted, lm3$fitted)
 [1] 0.9719647
 > cor(lm2$fitted, lm3$fitted)
 [1] 0.988647
```

Summary

- Perfect multicollinearity (rank deficient design matrix X, singularity) affects interpretability, causes unreliable predictions for future observations.
- In case of high collinearity (not perfect but in the range of $\sim$ 0.90), we won't be able to understand how various predictors impact Y, e.g., individual p-values might be misleading, confidence intervals will be wide and adding or deleting a single data point might change the coefficients dramatically (unstable coefficient estimates).
- We might be interested in only a subset of the predictors when we have thousands of them.
- Prediction accuracy might be improved by sacrificing a bit of bias in exchange for reducing the variance.
- It is often easier to interpret a simple model than a complex one.