INDR 450/550

Spring 2022

Lecture 16: Model Shrinkage,
Non-linear regressions

April 6, 2022

Fikri Karaesmen

Announcements

- Class Exercise at the end of lecture today. If you are participating online, please upload your document under Course Contents/Class Exercises
- Lab 5 material (on KNN regression) and a short video are available
- Exam scheduled.

- The first five labs were uploaded. Please follow them.

Predictive Analytics

- Remaining topics (to complete at the latest the week after the spring break)
  - Validation ✓
  - Model selection / regularization ✓
  - Non-linear regressions, generalized additive models
  - Tree-based methods (after the break)

Regularization: Reminder

- We talked about alternative formulations of the least squares regression problems that enable model reduction (reducing the number of parameters).
- Ridge regression: penalizes sums of squares of coefficients
- Lasso regression: penalizes sums of absolute values of coefficients
- There’s also an integer optimization formulation.

Regularization: Constrained Optimization formulation

- For instance, one can easily eliminate correlated predictors using an additional constraint.
- Assume that predictor i and k have correlation (in absolute value) above a threshold. We can then add the constraint:

$$z_i + z_k \leq 1$$

- And we can repeat this for all pairwise correlated predictors.

This part is based on *Machine Learning under a Modern Optimization Lens* by Bertsimas and Dunn

Regularization: Constrained Optimization formulation

- One can easily control functional forms.
- For instance, we might be tempted to try t, $t^{4/3}$, $t^{5/3}$ and t^2 as predictors.
- But for robustness, we might prefer to use at most of the four predictors:

$$z_1 + z_2 + z_3 + z_4 \leq 1$$

This part is based on *Machine Learning under a Modern Optimization Lens* by Bertsimas and Dunn

Regularization: Constrained Optimization formulation

- Using more complicated and non-linear constraints:
  - Multi-collinearity can be handled
  - We can specify that only statistically significant predictors are used
- This constrained optimization framework is called holistic regression (Berstimas and Dunn, 2019).

This part is based on *Machine Learning under a Modern Optimization Lens* by Bertsimas and Dunn

Regularization: Dimension Reduction

- An alternative approach is to reduce the dimension of the problem by projecting the data to a lower-dimensional space.
- Principal Component Analysis is the tool for such projections.
- Principal Component Regression is the estimation counterpart.

Regularization: High Dimensional Data

- The approaches we discussed for model shrinkage are always useful but especially crucial for high dimensional problems when the number of predictors may be larger than the sample size.
- This routinely happens today as we seek for better estimators and can access mote data and run large regressions.
- With high dimensional data, a regression will always be a perfect fit on the training set.
- But this is entirely due to overfitting and the perfect fit will perform poorly on the test data.

Regularization: High Dimensional Data

- Increasing the number of predictors leads to an MSE of zero on the training set but has terrible MSE performance on the test set.

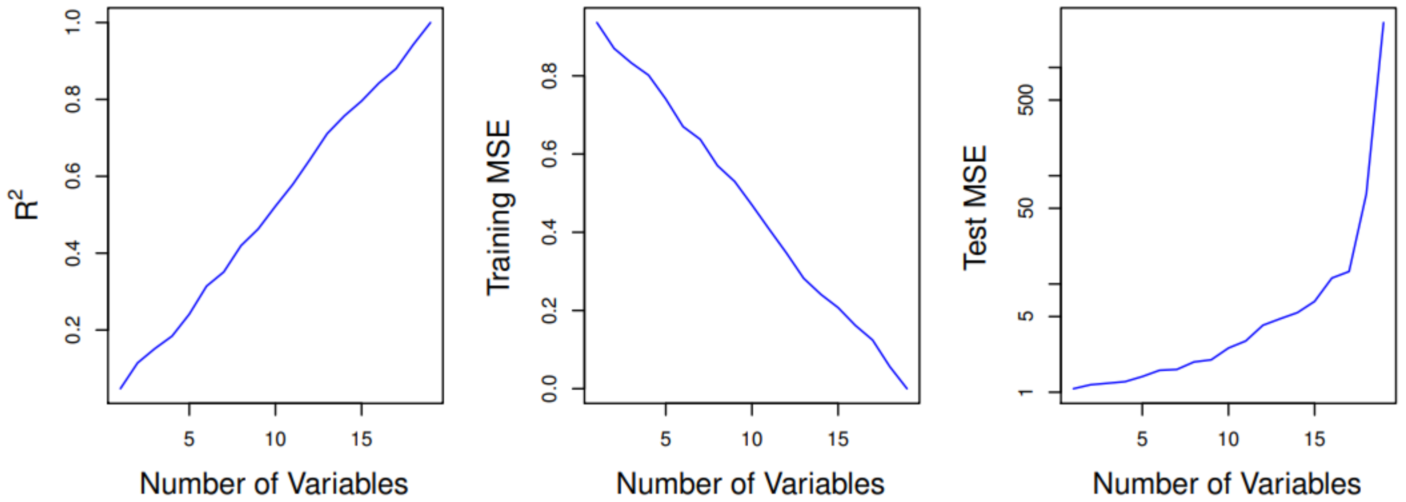
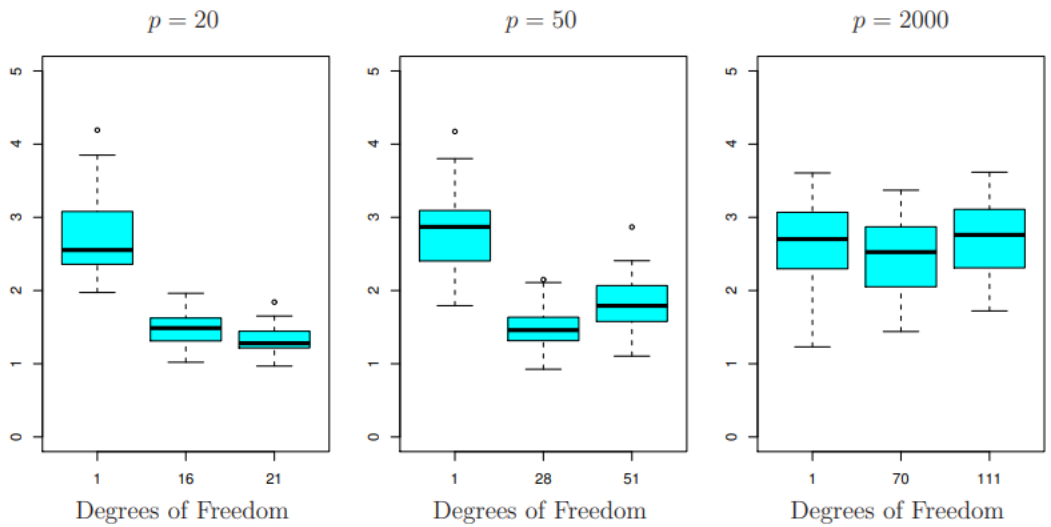


FIGURE 6.23. *On a simulated example with $n = 20$ training observations, features that are completely unrelated to the outcome are added to the model.* Left: *The R^2 increases to 1 as more features are included.* Center: *The training set MSE decreases to 0 as more features are included.* Right: *The test set MSE increases as more features are included.*

Regularization: High Dimensional Data

- Note that measures such as adjusted R^2, AIC etc. fail with high dimensional data because the MSE is close to zero.
- This is why tools like ridge and lasso are crucial.
- We should also be aware that despite lasso and ridge extracting the best reduced model is hard when the number of predictors is much larger than the sample size.

Regularization: High Dimensional Data



n=100

FIGURE 6.24. *The lasso was performed with $n = 100$ observations and three values of p, the number of features. Of the p features, 20 were associated with the response. The boxplots show the test MSEs that result using three different values of the tuning parameter λ in (6.7). For ease of interpretation, rather than reporting λ, the* degrees of freedom *are reported; for the lasso this turns out to be simply the number of estimated non-zero coefficients. When $p = 20$, the lowest test MSE was obtained with the smallest amount of regularization. When $p = 50$, the lowest test MSE was achieved when there is a substantial amount of regularization. When $p = 2{,}000$ the lasso performed poorly regardless of the amount of regularization, due to the fact that only 20 of the 2,000 features truly are associated with the outcome.*

Non-linear regressions

- We already used non-linear transformations of data to see if we could get better fits.
- Let’s formalize some classes of non-linear representations that lead to useful regressions.

Non-linear regressions: polynomial basis

- **Basis Functions**: Recall that, we attempted to strengthen the below single variable regression:

$$y_t = \beta_0 + \beta_1 t + \epsilon_t$$

  by the following extended form:

$$y_t = \beta_0 + \beta_1 t + \beta_2 t^2 + \beta_3 t^3 + \epsilon_t$$

- The extended form would maybe explain the non-linearities in the relationship.

- Polynomial functions are examples of a basis. Typically, we don't go beyond the cubic term x^3.

Non-linear regressions: basis functions

- **Basis Functions**: Consider a family of transformations for the predictor X: $b_1(X)$, $b_2(X)$,...,$b_n(X)$.
- Polynomials are an example where we have $b_i(X) = X^i$.
- We then consider the following regression using the basis variables:

$$y_t = \beta_0 + \beta_1 b_1(x_t) + \beta_2 b_2(x_t) + ... + \beta_n b_n(x_t) + \epsilon_t$$

- All tools from ordinary least squares regression are available under such transformations.

Non-linear regressions: polynomial basis

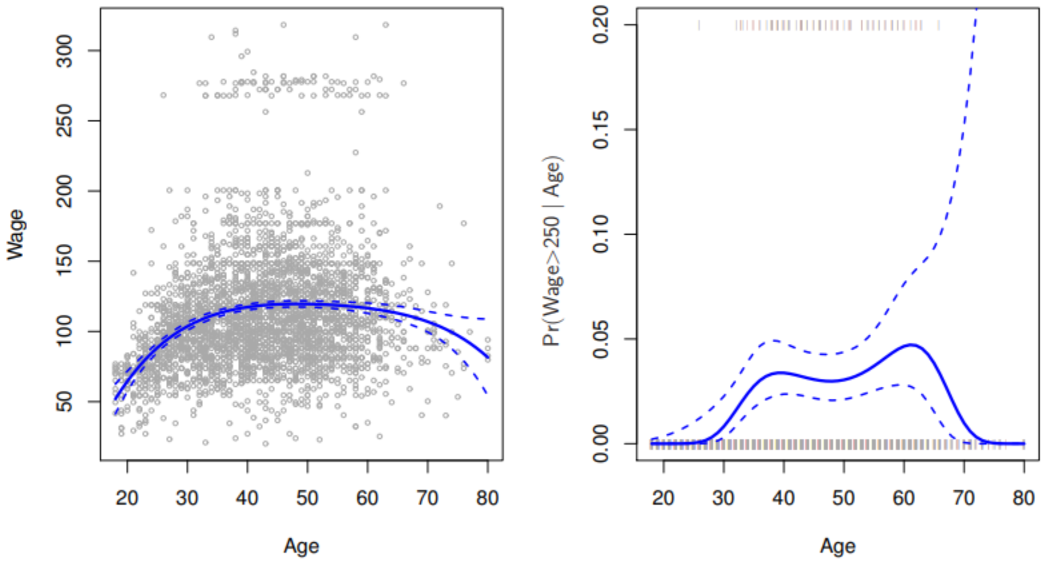


FIGURE 7.1. *The* Wage *data.* Left: *The solid blue curve is a degree-4 polynomial of* wage *(in thousands of dollars) as a function of* age*, fit by least squares. The dashed curves indicate an estimated 95 % confidence interval.* Right: *We model the binary event* wage>250 *using logistic regression, again with a degree-4 polynomial. The fitted posterior probability of* wage *exceeding* \$250,000 *is shown in blue, along with an estimated 95 % confidence interval.*

From *An Introduction to Statistical Learning*, James, Witten, Hastie, Thibshirani

Non-linear regressions: step functions

Step functions divide the range of data into intervals:

In greater detail, we create cutpoints $c_1, c_2, \ldots, c_K$ in the range of X, and then construct $K+1$ new variables

$$\begin{aligned}
C_0(X) &= I(X < c_1),\\
C_1(X) &= I(c_1 \leq X < c_2),\\
C_2(X) &= I(c_2 \leq X < c_3),\\
&\ \ \vdots\\
C_{K-1}(X) &= I(c_{K-1} \leq X < c_K),\\
C_K(X) &= I(c_K \leq X),
\end{aligned} \tag{7.4}$$

where $I(\cdot)$ is an *indicator function* that returns a 1 if the condition is true,

From *An Introduction to Statistical Learning*, James, Witten, Hastie, Thibshirani

Non-linear regressions: step functions

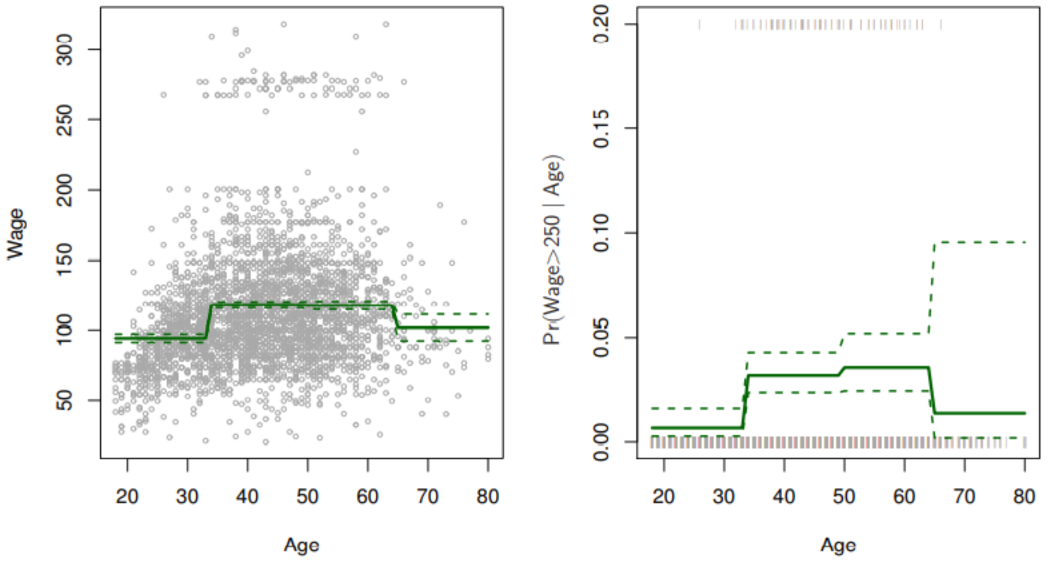


FIGURE 7.2. *The* Wage *data.* Left: *The solid curve displays the fitted value from a least squares regression of* wage *(in thousands of dollars) using step functions of* age*. The dashed curves indicate an estimated 95 % confidence interval.* Right: *We model the binary event* wage>250 *using logistic regression, again using step functions of* age*. The fitted posterior probability of* wage *exceeding* $250,000 *is shown, along with an estimated 95 % confidence interval.*

From *An Introduction to Statistical Learning*, James, Witten, Hastie, Thibshirani

Non-linear regressions: knots

- There are many other useful general basis functions. A particularly useful one is basis functions involving knots.
- Here is an example of a knot at t_1:

$$(t - t_1)^+ \equiv \max(t - t_1, 0)$$

- Now consider extending the basic model by a knot at the point t_1:

$$y_t = \beta_0 + \beta_1 t + \beta_2 (t - t_1)^+ + \epsilon_t$$

- Note that the knot has the effect of changing the slope of the regression curve at the point t_1. We are now fitting a partially linear function instead of a linear one.

From *An Introduction to Statistical Learning*, James, Witten, Hastie, Thibshirani

Non-linear regressions: knots

- Knots constitute useful basis functions. For instance, we can then try

$$y_t = \beta_0 + \beta_1 t + \beta_2(t - t_1)^+ + \beta_3(t - t_2)^+ + \beta_4(t - t_3)^+ + \epsilon_t$$

  where $t_1 < t_2 < t_3$.
- Using three knots, we are now allowing the slope to change at three different points.

From *An Introduction to Statistical Learning*, James, Witten, Hastie, Thibshirani

Some trials on the Google Share Price Data

- 253 days of data. First 150 days for training and the rest for test.
- I experimented with some polynomial terms and first degree knots at t_1=20, t_2=40, t_3=60 etc.

Out[6]:

Day	Price	t	t^2	t^3	k1	k2	k3	k4	k5	k6	k7	k8	k9
1	2064.879883	1	1.000000	1	0	0	0	0	0	0	0	0	0
2	2070.860107	2	1.414214	4	0	0	0	0	0	0	0	0	0
3	2095.169922	3	1.732051	9	0	0	0	0	0	0	0	0	0
4	2031.359985	4	2.000000	16	0	0	0	0	0	0	0	0	0
5	2036.859985	5	2.236068	25	0	0	0	0	0	0	0	0	0

In [7]:

```
dftest=df[150:]
dftest.head()
```

Out[7]:

Day	Price	t	t^2	t^3	k1	k2	k3	k4	k5	k6	k7
151	2852.659912	151	12.288206	22801	131	111	91	71	51	31	11
152	2830.020020	152	12.328828	23104	132	112	92	72	52	32	12
153	2723.679932	153	12.369317	23409	133	113	93	73	53	33	13
154	2690.419922	154	12.409674	23716	134	114	94	74	54	34	14
155	2665.310059	155	12.449900	24025	135	115	95	75	55	35	15

Some trials on the Google Share Price Data

OLS Regression Results

Dep. Variable:	Price	**R-squared:**	0.983
Model:	OLS	**Adj. R-squared:**	0.982
Method:	Least Squares	**F-statistic:**	800.6
Date:	Mon, 04 Apr 2022	**Prob (F-statistic):**	1.42e-117
Time:	17:53:43	**Log-Likelihood:**	-750.61
No. Observations:	150	**AIC:**	1523.
Df Residuals:	139	**BIC:**	1556.
Df Model:	10		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	2082.2044	16.292	127.807	0.000	2049.993	2114.416
t	-1.6274	1.920	-0.848	0.398	-5.423	2.169
t ^ 2	-3.0551	3.403	-0.898	0.371	-9.783	3.673
t ^ 3	2.0053	3.060	0.655	0.513	-4.045	8.055
k1	17.6413	1.854	9.516	0.000	13.976	21.307
k2	-14.7021	1.604	-9.164	0.000	-17.874	-11.530
k3	8.5620	1.585	5.401	0.000	5.428	11.696
k4	-3.2221	1.584	-2.034	0.044	-6.354	-0.090
k5	1.5443	1.590	0.971	0.333	-1.600	4.688
k6	0.2036	1.673	0.122	0.903	-3.104	3.512
k7	-18.7806	3.317	-5.661	0.000	-25.340	-12.221

Many of the knots are statistically significant!

RMSE (train) = 36.06

But RMSE (test) = 710.58!

Some trials on the Google Share Price Data

- To improve the error on the test set, I tried a lasso regression

```
In [275]: lasso.set_params(alpha=alphas[13])
          lasso.fit(x_train, y_train)
          mse_test=mean_squared_error(y_test, lasso.predict(x_test))
          mse_train=mean_squared_error(y_train, lasso.predict(x_train))
          mses.append(mse_test)
          msetrains.append(mse_train)
          print('RMSE Train =', np.sqrt(mse_train))
          print('RMSE Test =', np.sqrt(mse_test))
          lasso.coef_
```

```
RMSE Train = 48.53826598386097
RMSE Test = 110.96263681715224

Out[275]: array([ 0.        , 19.10604872,  0.        ,  5.15889502,  0.        ,
                  0.        ,  0.        ,  0.        ,  0.        , -6.1961299 ])
```

The final model uses one of the polynomial terms and two knots.

RMSE (test) = 110.96

Non-linear regressions: knots / cubic splines

- Approximations of functions using knots fall in the class of spline approximations.
- Note that when we use a knot $(t - t_1)^+$, we are allowing a slope change at t_1. This results in a change of slope at t_1 and therefore a non-smooth fit.
- To have a smooth curve, we need continuity at the knot and also the continuity of the first and second derivatives.
- The basis that gives such smooth functions consists of the following knots:

$$((t - t_1)^+)^3 = \begin{cases} (t - t_1)^3 & \text{if } t > t_1 \\ 0 & \text{otherwise} \end{cases}$$

From *An Introduction to Statistical Learning*, James, Witten, Hastie, Thibshirani

Non-linear regressions: knots / cubic splines

- Approximations of functions using knots fall in the class of spline approximations.
- Note that when we use a knot $(t - t_1)^+$, we are allowing a slope change at t_1. This results in a change of slope at t_1 and therefore a non-smooth fit.
- To have a smooth curve, we need continuity at the knot and also the continuity of the first and second derivatives.
- The basis that gives such smooth functions consists of the following knots:

$$((t - t_1)^+)^3 = \begin{cases} (t - t_1)^3 & \text{if } t > t_1 \\ 0 & \text{otherwise} \end{cases}$$

From *An Introduction to Statistical Learning*, James, Witten, Hastie, Thibshirani

Non-linear regressions: knots / cubic splines

- Here's an example of a cubic spline basis with 2 knots:

$$\begin{aligned} y_t &= \beta_0 + \beta_1 t + \beta_2 t^2 + \beta_3 t^3 \\ &\quad + \beta_4 \left((t - t_1)^+\right)^3 + \beta_5 \left((t - t_2)^+\right)^3 + \epsilon_t \end{aligned}$$

- Note that if we have K knots, then we have $K + 4$ predictors in the model.
- The resulting fit be a smooth curve.

From *An Introduction to Statistical Learning*, James, Witten, Hastie, Thibshirani

Non-linear regressions: knots / cubic splines

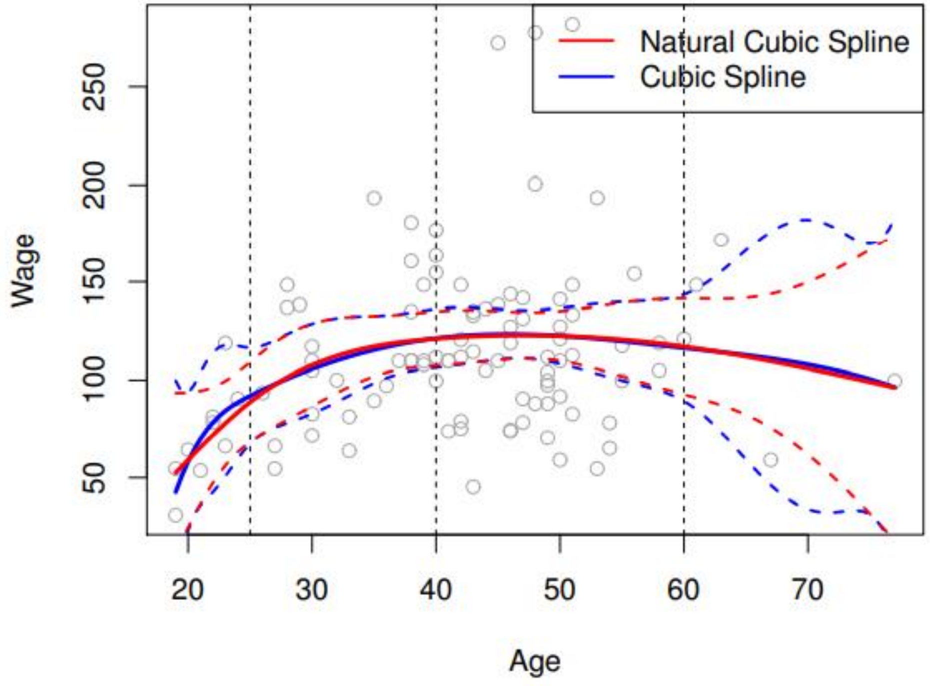


FIGURE 7.4. *A cubic spline and a natural cubic spline, with three knots, fit to a subset of the* `Wage` *data. The dashed lines denote the knot locations.*

From *An Introduction to Statistical Learning*, James, Witten, Hastie, Thibshirani

Non-linear regressions: knots / cubic splines

To determine the location and the number of knots: try several ones and choose by cross-validation.

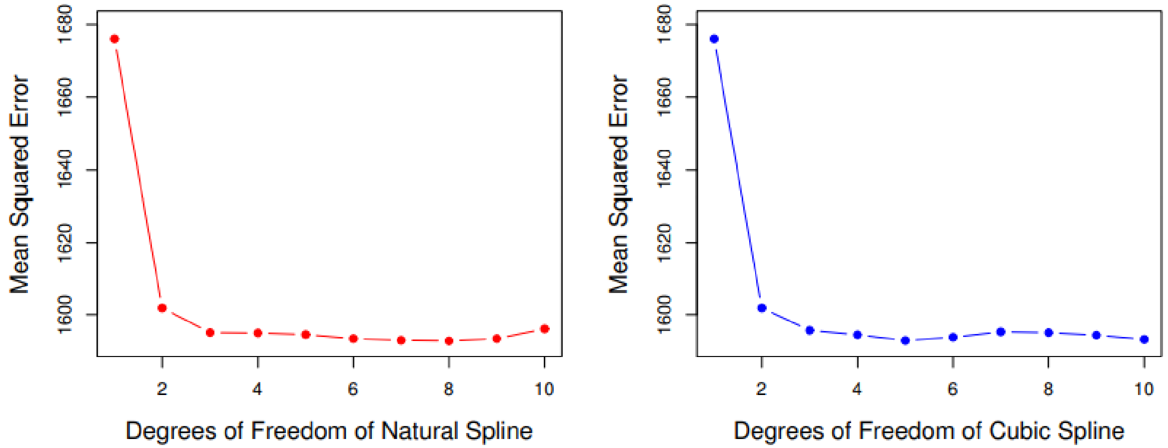


FIGURE 7.6. *Ten-fold cross-validated mean squared errors for selecting the degrees of freedom when fitting splines to the* `Wage` *data. The response is* `wage` *and the predictor* `age`. Left: *A natural cubic spline.* Right: *A cubic spline.*

From *An Introduction to Statistical Learning*, James, Witten, Hastie, Thibshirani

Some trials on the Google Share Price Data

- I also a tried a cubic spline with three knots at 25,40 and 60.

```
In [35]: fit1 = sm.GLM(y_train, t_3knots).fit()
         fit1.params

Out[35]: array([2066.14112978,    5.07698694,  -79.74134178,  304.27959612,
                 266.51573955,  877.49354265,  781.23132479])

In [36]: pred1 = fit1.predict()

In [45]: np.sqrt(np.mean(np.square(pred1-y_train)))

Out[45]: 40.530954127347194

In [43]: np.mean(np.abs((pred1-y_train)/y_train))

Out[43]: 0.013367668355775858
```

RMSE (train) = 40.53