Carnegie Mellon University

MSE

Data Analytics for Materials Science

27-737

A.D. (Tony) Rollett, Amit Verma, R.A. LeSar (Iowa State Univ.)

Dept. Materials Sci. Eng., Carnegie Mellon University

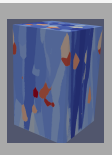
Feature Selection via Best-subset, Ridge, and Lasso regression

Lecture 7

Revised: 24th Feb. 2021

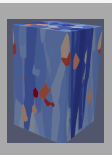
To date, we have discussed:

- linear algebra
- linear regression: prediction
- multiple linear regression: prediction
- Regular expressions



Useful sources of information (both in Canvas):

- Hastie et al. Elements of Statistical Learning. linear regression: prediction
- Wiki pages on lasso (statistics) and ridge (statistics)
- Also:
  https://www.statology.org/lasso-regression-in-r/
  https://www.statology.org/ridge-regression-in-r/



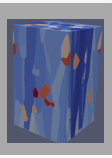
R: `glmnet` package (available on CRAN)

Matlab: `probably available …`

Mathematica: `probably available …`

There are also implementations in Python, …

Pick your favorite program and search for feature selection in the documentation.



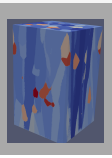
model availability

The objective of today's lecture is to acquaint you with *feature selection*, which is equivalent to *model selection* (as you find it in Hastie *et al*.).

The point is that data sets often have variables (features) that trend together i.e., are co-linear or close to it. Or a given feature does not have any predictive power and needs to be removed from the fit.

You have, fairly obviously, already done this but only by hand. There are tools to help do this automatically and provide quantification of how well it is working. Also, the by-hand approach is completely discrete: variables are either included or not. The same criticism applies to *best-subset selection* which is a variant of MLR.

We start with *ridge* regression analysis. And follow that with *lasso*. And then, time allowing, *FeaLect*.



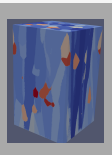
- Objective(s)

It is always possible to run an analysis on the full dataset. However, it is far better practice to split your data into two pieces, one for *training* and the other for *validation*. The choice of fractions is arbitrary and depends somewhat on the number of datapoints but a 2/3^{rds} for training and 1/3rd for validation is reasonable. Think of this as cutting the spreadsheet *horizontally* into two pieces.

A further advance is to use *cross-validation*. This is simpler than it sounds because it just means repeating the analysis, say, ten times with different choices of the split (between *training* and *validation*) and recording the accuracy in each test. This approach is also known as *k-fold validation*.

Any approach has a complexity associated with it. The simplest complexity measure is the (discrete) number of variables (features) included in a *best-subset* model. For *ridge* this is the *effective degrees of freedom* and for *lasso* this is the lambda value. This leads to consideration of the balance between complexity and the prediction error. Although the latter decreases with complexity, it usually levels out, which means that one can choose a (minimum) complexity that brings the prediction error within one standard deviation of the best case.

https://medium.com/the-owl/k-fold-cross-validation-in-keras-3ec4a3a00538



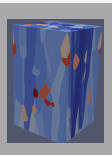
- Cross-validation, training

`cv.glmnet` applies cross-validation

In a call to `glmnet`, `alpha=0` gives you *ridge*, and `alpha=1` gives you *lasso*. You can vary alpha continuously between 0 & 1.

When using lasso, we suggest that you do not use the minimum lambda but rather "1se", i.e.,
`bestlam2 <- cv.out$lambda.1se`, as opposed to

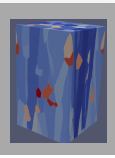
`bestlam <- cv.out$lambda.min`



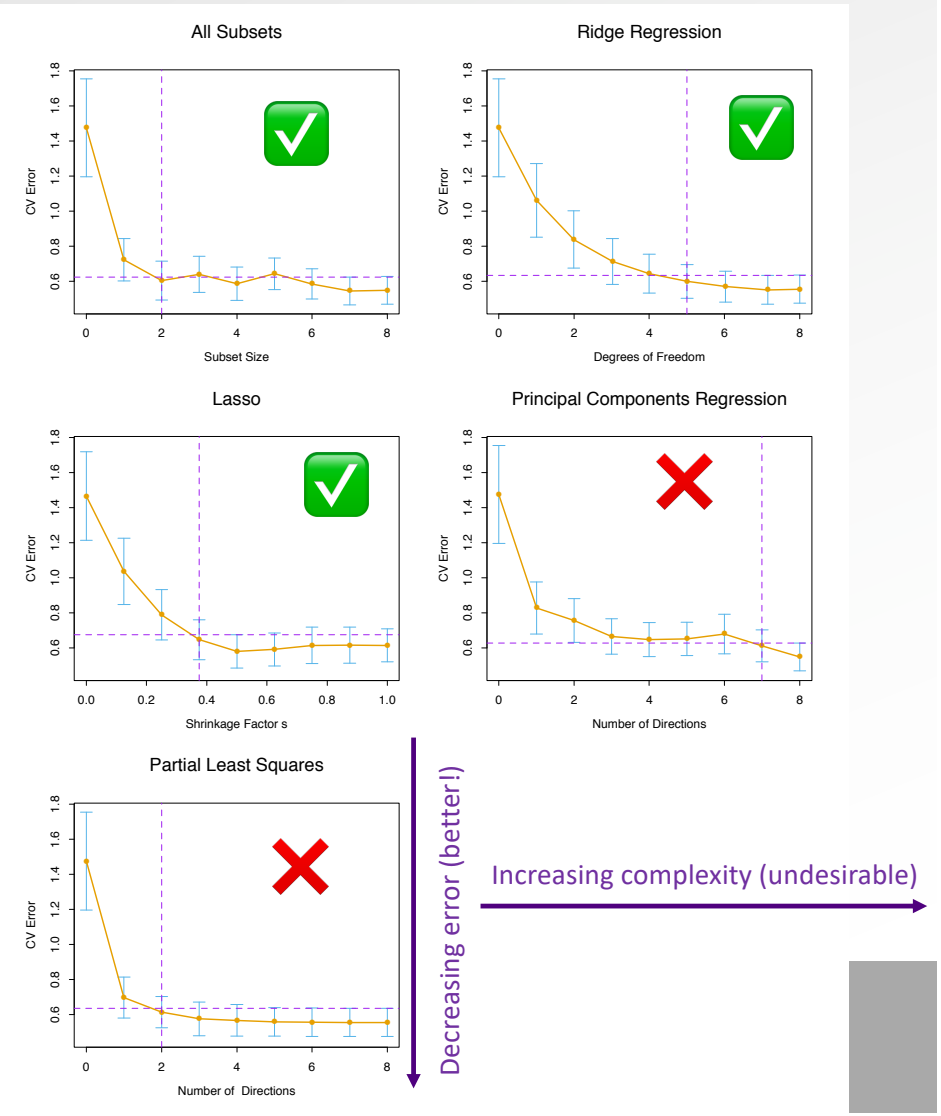
Quoting from Hastie et al.:

FIGURE 3.7. Estimated prediction error curves and their standard errors for the various selection and shrinkage methods. Each curve is plotted as a function of the corresponding complexity parameter for that method. The horizontal axis has been chosen so that the model complexity increases as we move from left to right. The estimates of prediction error and their standard errors were obtained by tenfold cross-validation; full details are given in Section 7.10. The least complex model within one standard error of the best is chosen, indicated by the purple vertical broken lines.

This is also discussed in terms of *bias* versus *variance*. The aim is always to get a combination of the two forms of error so that there is a clear choice of the trade-off at the minimum error.



- Complexity vs. Error



The best-subset regression seeks the subset of the available variables that produces the smallest residual sum of the squares. Algorithm development has extended the effective maximum out to ≈40, e.g., *leaps & bounds* by Furnival & Wilson (1974).

For a subset size *k=2*, the selected variables that minimize the error (RSS) do *not* have to include the single variable that minimizes for *k=1*, and so on and so forth.

The graph shows the example from Hastie *et al*.

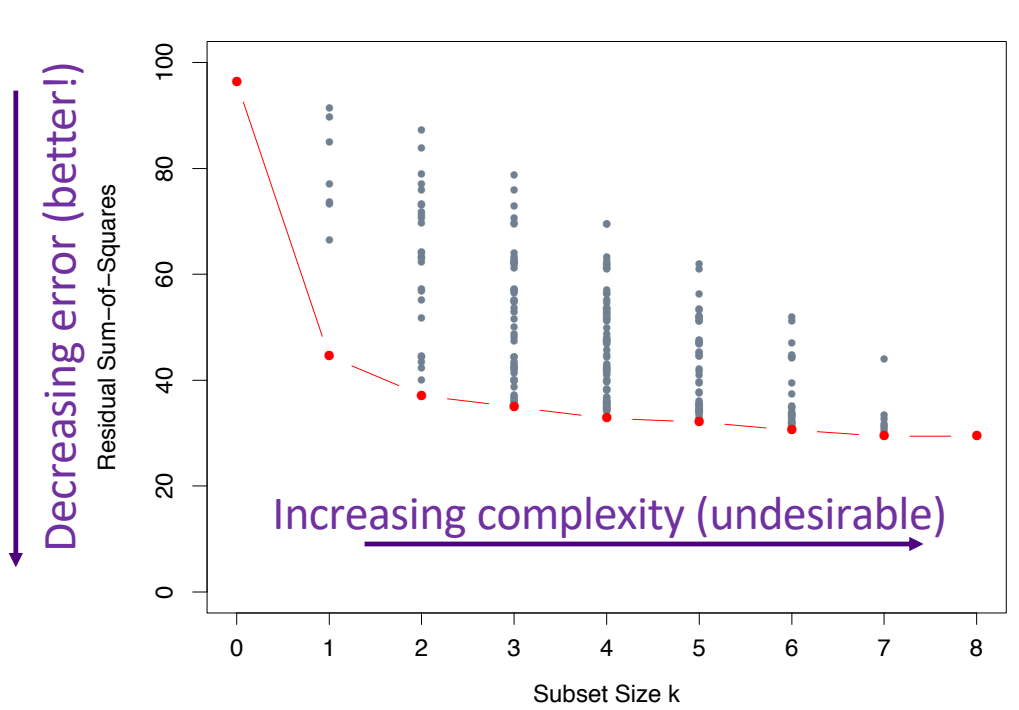
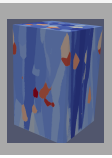


FIGURE 3.5. *All possible subset models for the prostate cancer example. At each subset size is shown the residual sum-of-squares for each model of that size.*

Possible source of the data: https://www.rdocumentation.org/packages/MultNonParam/versions/1.2.5/topics/prostate.data



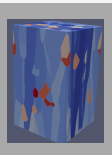
- Best-Subset selection

Lasso stands for *least absolute shrinkage and selection operator*.

It is a method that does both *variable|feature selection* and regularization. This is intended to improve accuracy of results and their interpretation.

As an instance of convergent evolution, the method was originally developed in geophysics[1]. Later, Robert Tibshirani re-discovered the technique and named it *lasso*[2].

1. Santosa, Fadil; Symes, William W. (1986). "Linear inversion of band-limited reflection seismograms". *SIAM Journal on Scientific and Statistical Computing. SIAM*. **7** (4): 1307–1330. doi:10.1137/0907087
2. Tibshirani, Robert (1996). "Regression Shrinkage and Selection via the lasso". *J. Royal Statistical Soc. Series B (methodological)*. Wiley. **58** (1): 267–88. JSTOR 2346178

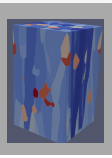


- **Lasso** regression analysis - development

Lasso came after *ridge regression* was established and in use. The latter improves prediction accuracy by shrinking the sum of the squares of the regression coefficients to be < a chosen value so as to avoid|minimize over-fitting. It does not, however, perform covariate selection, which obscures interpretation of the model.

Instead, *lasso* forces the absolute value of the RSS to be < a chosen value, thus forcing some coefficients to (essentially) zero, which is how it excludes the associated variables (features). *ridge* regression does not zero out coefficients.

Ridge and *lasso* are both variants of *multiple linear regression.*



- **Lasso** method versus **Ridge**

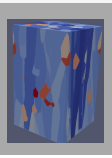
The numerical|quantitative objective of *lasso* is to minimize the same RSS as in MLR but subject to a specific constraint.

$$\min_{\beta_0,\beta}\left\{\sum_{i=1}^{N}(y_i-\beta_0-x_i^T\beta)^2\right\}\ \text{subject to}\ \sum_{j=1}^{p}|\beta_j|\le t$$

The key addition compared to MLR is the quantity t, which the analyst's choice of the parameter that determines the degree of regularization.

Set X as the covariate matrix with $X_{ij}=(x_i)_j$, and $(x_i)^T$ is the i^{th} row of X, we can re-write as:

$$\min_{\beta_0,\beta}\left\{\frac{1}{N}\|y-\beta_0 1_N-X\beta\|_2^2\right\}\ \text{subject to}\ \|\beta\|_1\le t$$



- **Lasso** technique

The notation $\|u\|_p = \left(\sum_{i=1}^{N} |u_i|^p \right)^{1/p}$

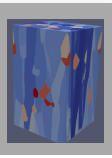
is the standard way to write the L^p norm.

(Thinking of the question of *scaling* the data) if we write the scalar mean of the datapoints as <x> and that of the responses as <y>, the resulting estimate for β_0 is <y>- <x>$^T\beta$ so that:

$$y_i - \hat{\beta}_0 - x_i^T\beta = y_i - (\bar{y} - \bar{x}^T\beta) - x_i^T\beta = (y_i - \bar{y}) - (x_i - \bar{x})^T\beta$$

This motivates the use of *auto-scaling* the data, which **glmnet** does for you. It is also normal practice to standardize the co-variates so that the solution does not depend on the scales associated with the measured variables.

$$\left(\sum_{i=1}^{N} x_i^2 = 1 \right)$$



In fact, the equations to this point are almost the same as for *ridge regression*!
Writing the same equation with a *complexity parameter* λ that controls the amount of shrinkage (of the coefficients), we get:

$$\hat{\beta}^{\text{ridge}} = \underset{\beta}{\text{argmin}} \left\{ \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij} \beta_j \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\}.$$

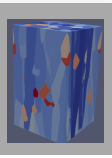
λ is limited to $\lambda \geq 0$. Larger values shrink coefficients down towards zero.

Re-writing in a slightly different way:

$$\hat{\beta}^{\text{ridge}} = \underset{\beta}{\text{argmin}} \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij} \beta_j \right)^2,$$

$$\text{subject to } \sum_{j=1}^{p} \beta_j^2 \leq t,$$

The *t* and λ variables are equivalent to each other.



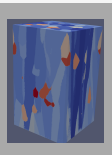
Continuing on with the autoscaled data (2 slides back), we write the equation in matrix form with the RSS as a function of the complexity parameter. $\boldsymbol{X}$ is the data matrix with N rows and p columns (for auto-scaled data).

$$\mathrm{RSS}(\lambda) = (\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta) + \lambda\beta^T\beta,$$

Now we can write a solution for the ridge problem as follows, $\mathbf{I}$ is the identity matrix:

$$\hat{\beta}^{ridge} = (\mathbf{X^T X} + \lambda\mathbf{I})^{-1}\mathbf{X^T y}$$

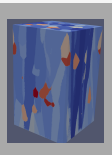
Inserting the penalty term as quadratic in beta, the *ridge regression solution* is a linear function of y. It also adds a constant to each leading diagonal term in $\mathbf{X}^{\mathrm{T}}\mathbf{X}$; this avoids nonsingular matrices, i.e., copes with collinearity.



- Ridge regression solution

This graph (from Hastie et al.) gives us an idea of how ridge regression works. As the effective degrees of freedom decrease (from right to left), the coefficients (none of them ever getting to zero except at crossing points) vary continuously in different proportions.

Note the use of the *effective degrees of freedom* for the horizontal axis. Ordinary MLR corresponds to the situation on the right.



- Ridge example

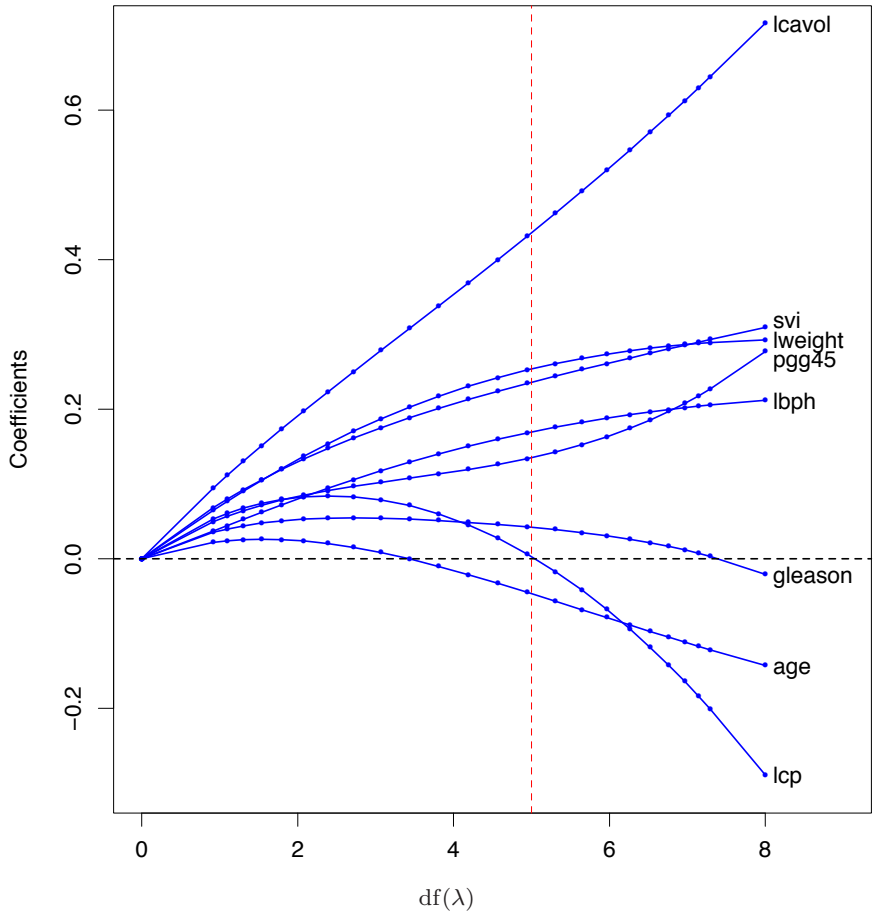


FIGURE 3.8. *Profiles of ridge coefficients for the prostate cancer example, as the tuning parameter* λ *is varied. Coefficients are plotted versus* $\mathrm{df}(\lambda)$*, the effective degrees of freedom. A vertical line is drawn at* $\mathrm{df} = 5.0$*, the value chosen by cross-validation.*

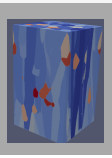
There is a convenient variable for use with ridge regression called the *effective degrees of freedom, df*. This is a decreasing function of the regularization factor λ, i.e., as you increase λ, the effective degrees of freedom decreases (down to zero). So, we start with the available *p* variables and decrease from there. The *effective degrees of freedom* is calculated as follows. One has to first *diagonalize* the matrix X, which can be done with *singular value decomposition* (SVD). The latter is a widely used technique of numerical analysis (not just statistics) that rotates the data into a combination of *diagonal matrix* and a *rotation matrix*. The values on the diagonal are d_j, j=1,p. Any zero entry means that the matrix is singular (co-linearity!).

$$\begin{aligned} \mathrm{df}(\lambda) &= \mathrm{tr}[\mathbf{X}(\mathbf{X}^T\mathbf{X}+\lambda\mathbf{I})^{-1}\mathbf{X}^T], \\ &= \mathrm{tr}(\mathbf{H}_\lambda) \\ &= \sum_{j=1}^{p} \frac{d_j^2}{d_j^2+\lambda}. \end{aligned}$$

Note. The SVD of an auto-scaled **N**×**p** data matrix **X** is

$$\mathbf{X} = \mathbf{UDV}^{\mathrm{T}}$$

U and **V** are **N**×**p** and **p**×**p** orthogonal matrices, respectively



- Effective degrees of freedom, *df*

Did you notice the difference between the two approaches|methods?!

The answer lies in which norm is used to obtain a solution.

Ridge uses L^2

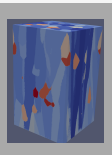
whereas Lasso uses L^1.

Which do you think might do better with large outlier values?

Ridge does a proportional decrease in the coefficients. Lasso, by contrast, translates each coefficients so that different variable reach zero at different points.

For lasso, we commonly plot versus lambda (decreases from the max to min from right to left) or, more intuitively versus the shrinkage factor s, which varies between $s=1$ (OLS) down to $s=0$ (next slide).

$$s = \frac{t}{\sum_j^p |\beta_j|}$$



- Ridge versus Lasso – what is the difference?!

At the largest value of *s*, the result is the same as OLS, i.e., all coefficients are included. It is now very obvious that as you decrease *s,* i.e., shrink, certain variables hit zero before others. Also note how you can observe switch-overs in the variables. Hastie *et al.*

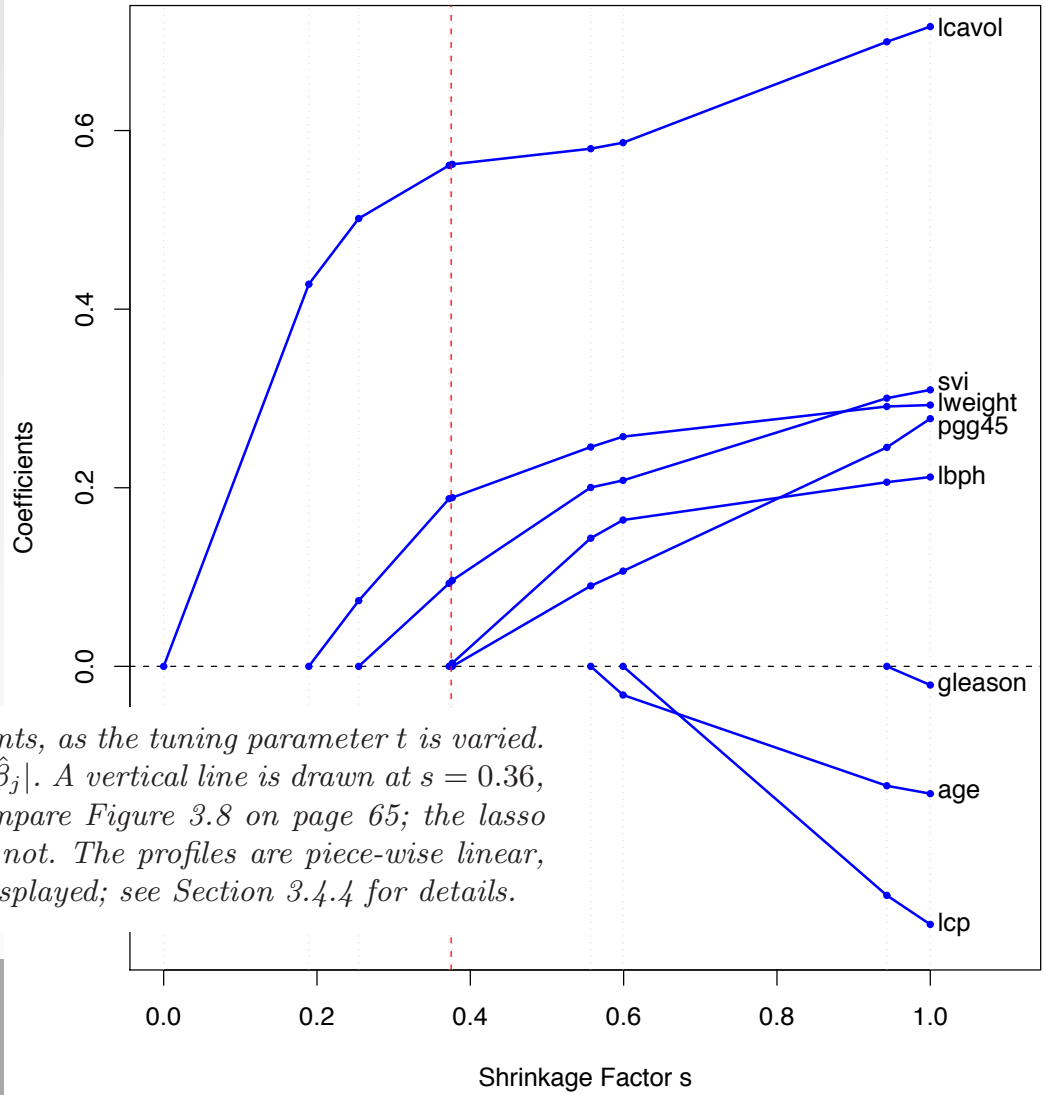
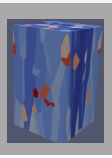


FIGURE 3.10. *Profiles of lasso coefficients, as the tuning parameter t is varied. Coefficients are plotted versus* $s = t/\sum_1^p |\hat{\beta}_j|$. *A vertical line is drawn at* $s = 0.36$, *the value chosen by cross-validation. Compare Figure 3.8 on page 65; the lasso profiles hit zero, while those for ridge do not. The profiles are piece-wise linear, and so are computed only at the points displayed; see Section 3.4.4 for details.*



- Lasso example

Previously, we discussed complexity vs error: we can also analyze this trade-off in terms of bias versus variance

$$\text{EPE}\left(Y, \hat{f}(x)\right) = \mathbb{E}_{Y|X,\mathcal{D}}\left[\left(Y - \hat{f}(X)\right)^2 \mid X = x\right] = \underbrace{\mathbb{E}_{\mathcal{D}}\left[\left(f(x) - \hat{f}(x)\right)^2\right]}_{\text{reducible error}} + \underbrace{\mathbb{V}_{Y|X}\left[Y \mid X = x\right]}_{\text{irreducible error}}$$

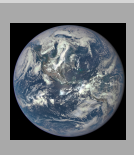
EPE: Expected Prediction Error
D: Sampled Data

Error due to sampled data

Variance (ε)

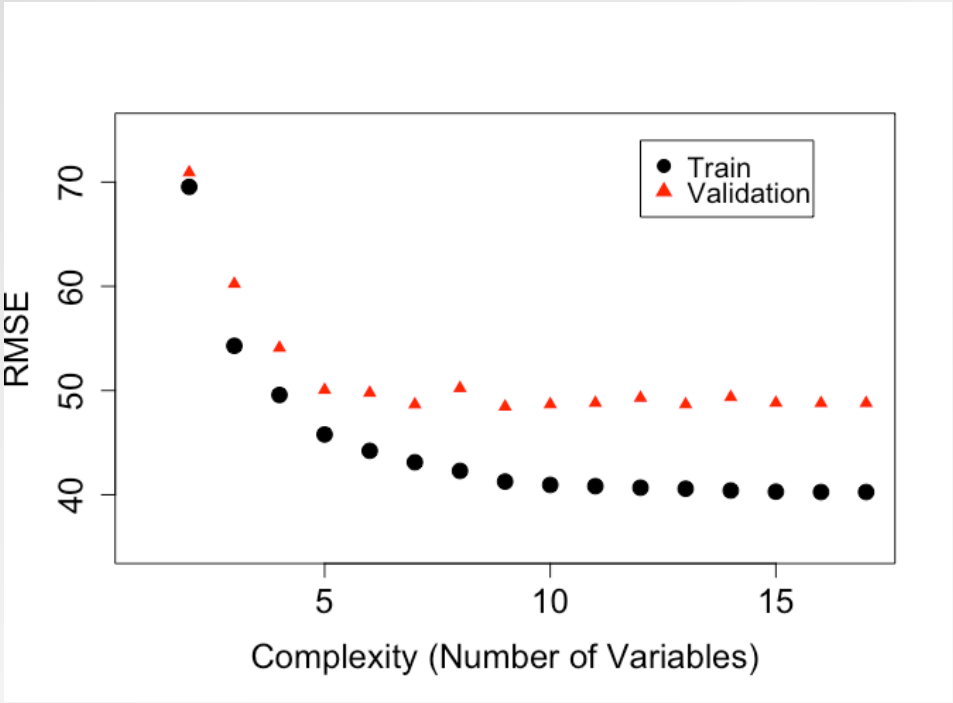
$$\text{MSE}\left(f(x), \hat{f}(x)\right) = \mathbb{E}_{\mathcal{D}}\left[\left(f(x) - \hat{f}(x)\right)^2\right] = \underbrace{\left(f(x) - \mathbb{E}\left[\hat{f}(x)\right]\right)^2}_{\text{bias}^2\left(\hat{f}(x)\right)} + \underbrace{\mathbb{E}\left[\left(\hat{f}(x) - \mathbb{E}\left[\hat{f}(x)\right]\right)^2\right]}_{\text{var}\left(\hat{f}(x)\right)}$$

Bias-variance trade-off



- Bias-Variance Trade-off

Biased: simple models; not all needed variables
Variance: complex models; too many variables



Linear Models have a high-bias problem!

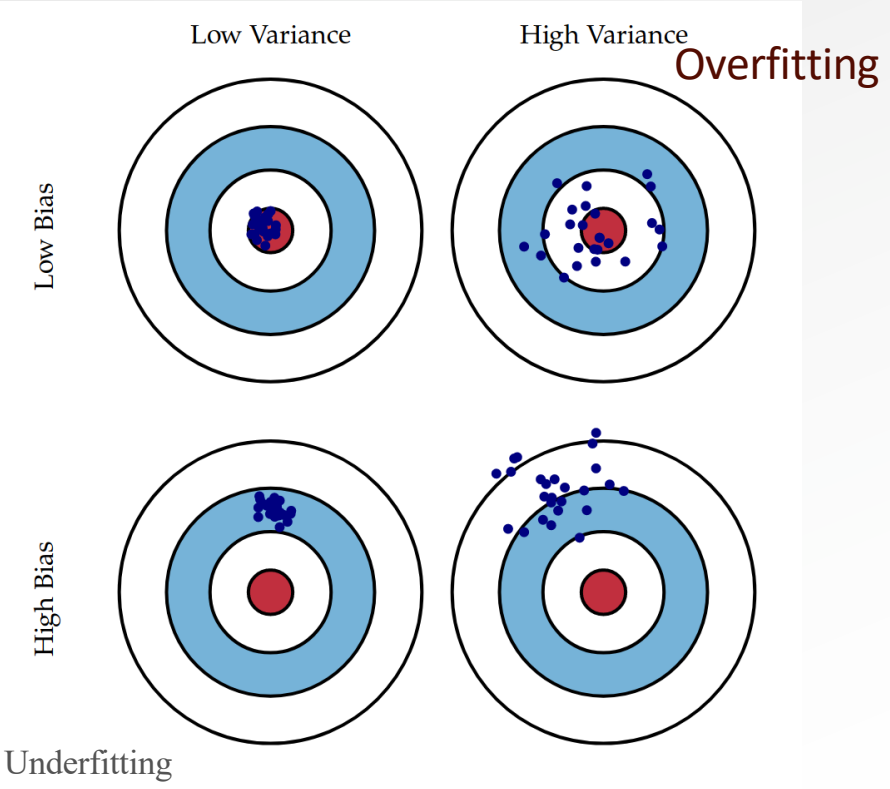
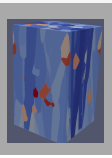
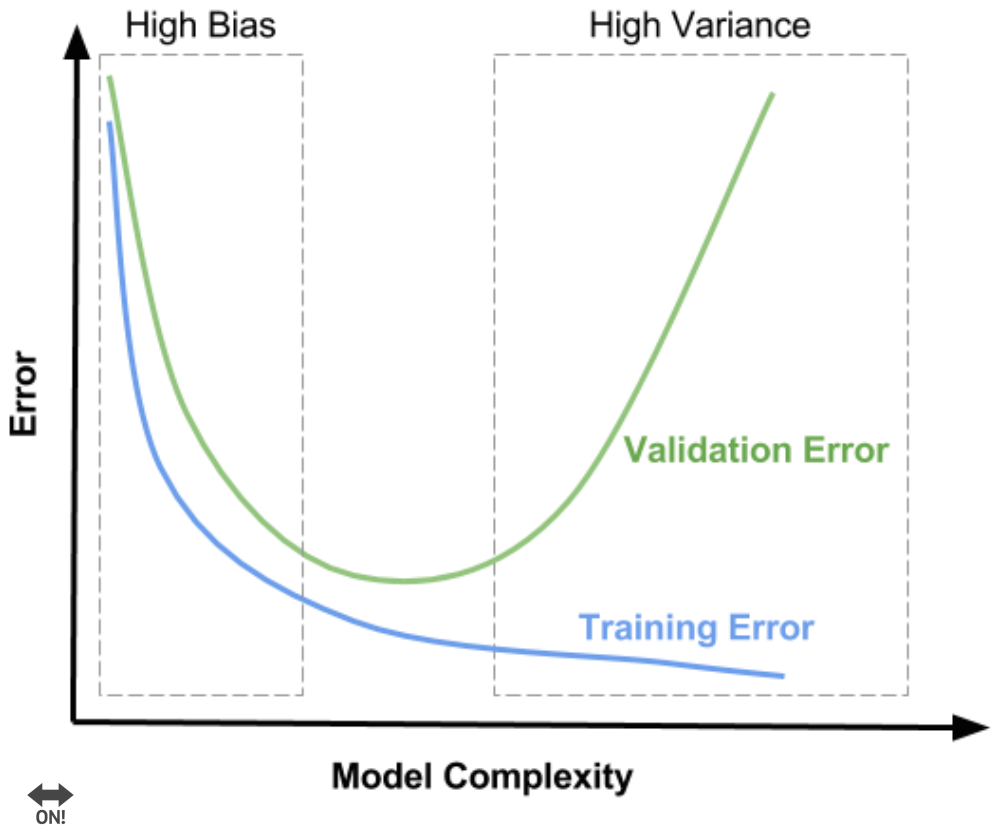


Fig. 1 Graphical illustration of bias and variance.

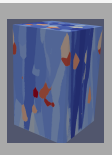


- Bias-Variance Trade-off

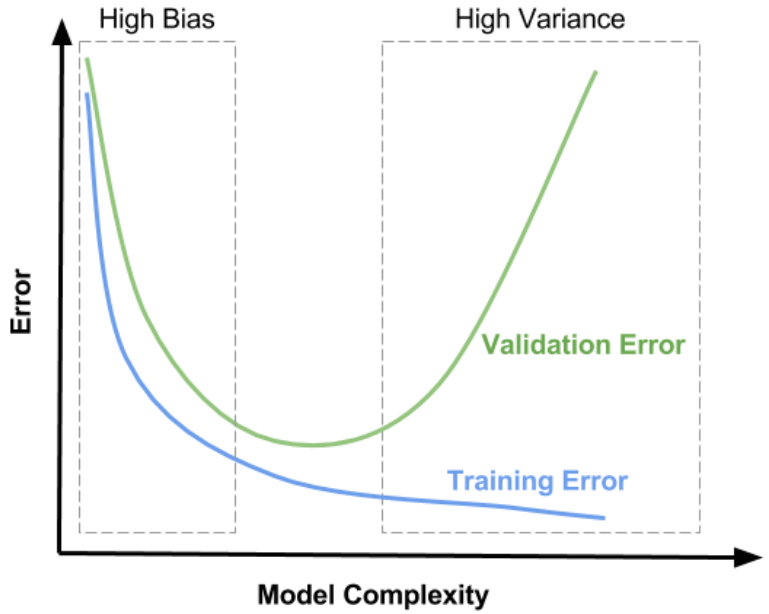
First figure in Scott Fortmann-Roe's
Understanding the Bias-Variance Tradeoff

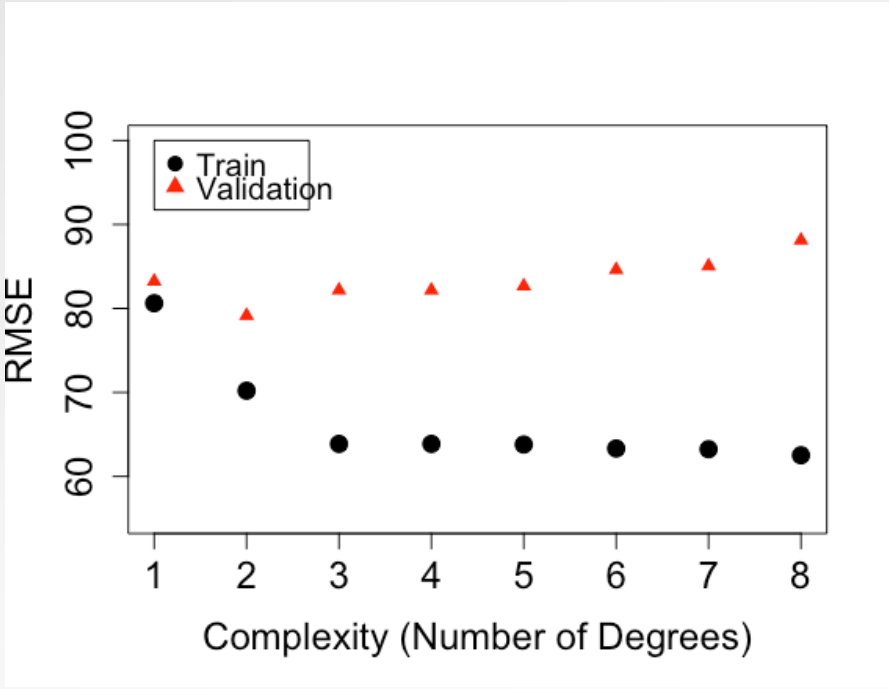


https://www.learnopencv.com/bias-variance-tradeoff-in-machine-learning/



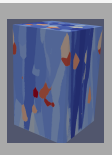
- Bias-Variance Trade-off





High-variance problems have a low training error and a very high validation error

https://www.learnopencv.com/bias-variance-tradeoff-in-machine-learning/



- Bias-Variance Trade-off

- Subset Selection: Identify a subset of predictors

$$\text{RSS} = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2$$

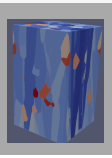
- Regularization: all predictors, however estimated coefficients are shrunken towards zero (*ridge*) / exactly zero (*lasso*)
  - *Ridge*: λ is the tuning parameter
  - Leads to low variance
  - Disadvantage: all predictors

$$\sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 = \text{RSS} + \lambda \sum_{j=1}^{p} \beta_j^2$$

  - *Lasso*: performs variable selection

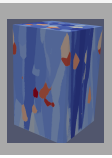
$$\sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| = \text{RSS} + \lambda \sum_{j=1}^{p} |\beta_j|$$

shrinkage penalty



• Ridge / Lasso Regularization

QUESTIONS?



Homework will contain questions on *regular expressions* and on *feature selection.*