Linear Regression as Machine Learning

Machine Learning in Molecular Science

Prof. Michael Shirts

July 22nd, 2024

i-CoMSE

This session

- Simple linear regression
  - The model
  - The regression framework, which is much more general for supervised learning
- Making predictions with the model
- Multiple linear regression
- Regularization
- General least squares
- Logistic regression

The simplest machine learning: Simple linear regression (SLR)

- The regression framework (generic in supervised machine learning)
  - Propose a model
  - Define a measure of the error of the model
    - Called "score" or "loss function" in ML
  - Estimate the "best" coefficients given the score
  - Determine the error in the model

Variable definitions

$$Y \approx \beta_0 + \beta_1 X$$

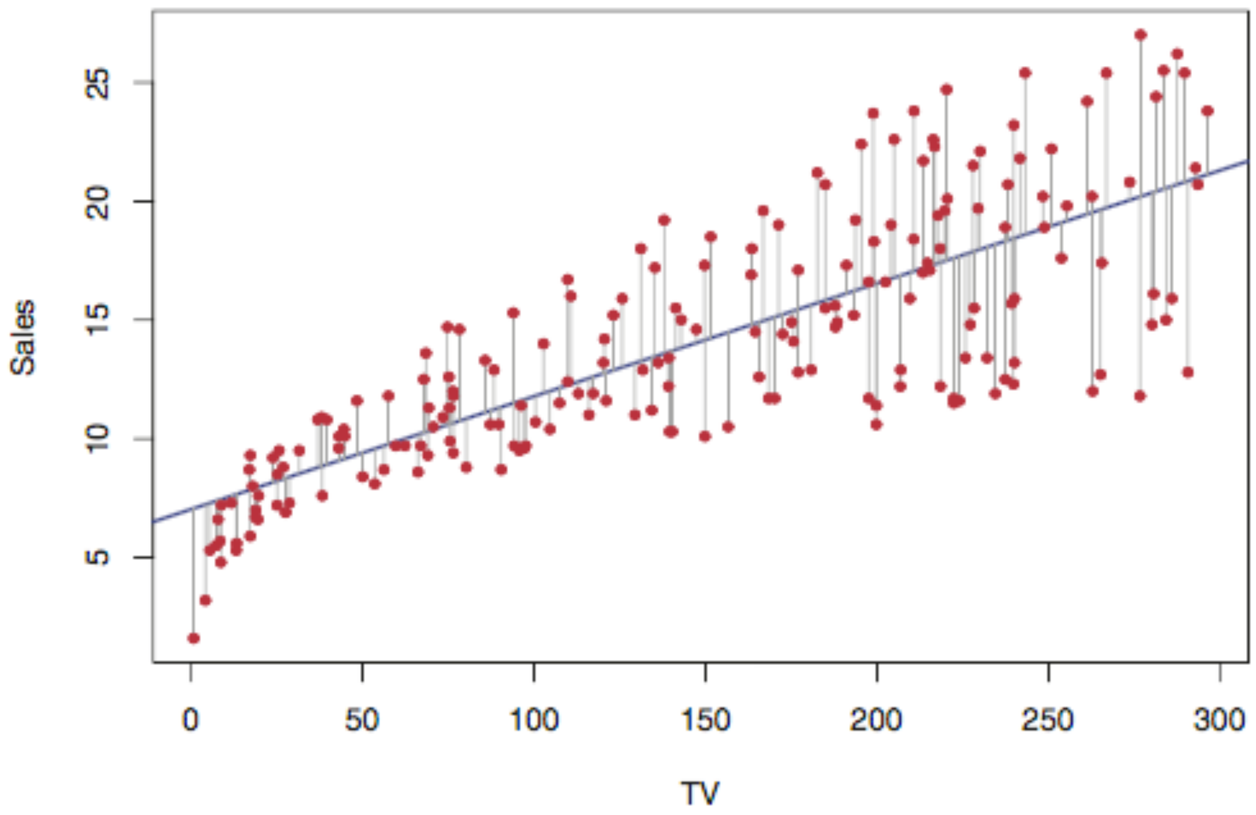
The humble SLR
(simple linear regression)
Or ordinary least squares (OLS)

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$$

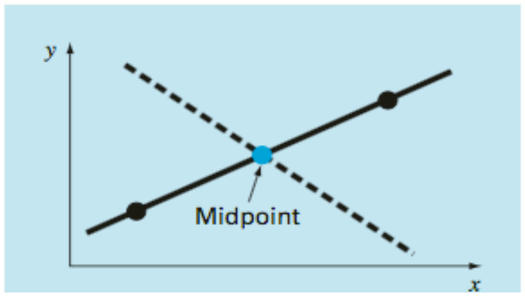
Distinguishing the estimates we make with the training data (hat) from the proposed actual coefficients/responses…

How different is the model from the truth?

- Residuals
    - The difference between the observed value of the dependent variable (y) and the predicted value (ŷ) is called the residual (e).

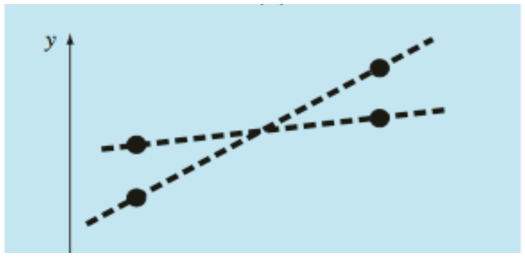


What different ways of determining the "best" model?



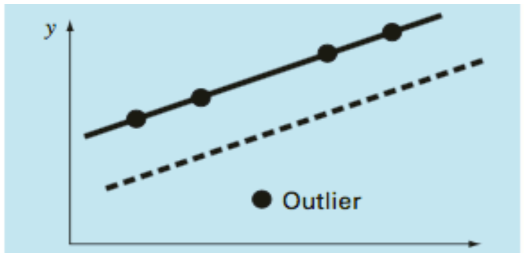
Minimize sum of residuals $\sum_i e_i$

Errors can cancel out



Minimize sum of absolute value of residuals $\sum_i |e_i|$

Multiple solutions, bad math properties



Minimize maximum absolute value residual $\text{Max } e_i$

Only depends on outliers

Residual Sum of Squares

Sum of squared residual errors has much better math properties than others,

like uniqueness and analytical error estimates.

$$\mathrm{RSS} = (y_1 - \hat{\beta}_0 - \hat{\beta}_1 x_1)^2 + (y_2 - \hat{\beta}_0 - \hat{\beta}_1 x_2)^2 + \ldots + (y_n - \hat{\beta}_0 - \hat{\beta}_1 x_n)^2$$

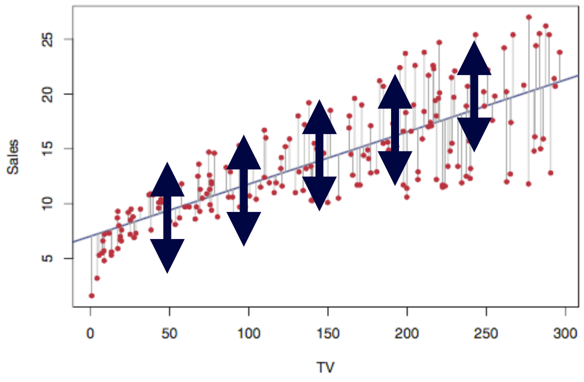
For general models:

$$\mathrm{RSS} = (y_1 - f(x_1))^2 + (y_2 - f(x_2))^2 + \ldots + (y_n - f(x_n))^2$$

Mean Square Error

Mean Square Error (MSE) is the average square residual

$$\text{MSE} = \frac{\text{RSS}}{N}$$



Root Mean Square Error (RMSE) is easier to grasp because it has the same units as the output

$$\text{RMSE} = \sqrt{\text{MSE}}$$

This is a little different than the unbiased estimator of residual standard error

$$\text{RSE} = \frac{\text{RSS}}{N-p}$$

$$\text{MSE} = \text{Variance} + \text{Bias}^2$$

The bias-variance tradeoff is directly reflected in the MSE

The method of least squares minimization

Given model

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$$

Training data

$$(x_1, y_1),\ (x_2, y_2), \ldots,\ (x_n, y_n)$$

Find the β_0 and β_1 minimizing the RSS

$$\mathrm{RSS} = (y_1 - \hat{\beta}_0 - \hat{\beta}_1 x_1)^2 + (y_2 - \hat{\beta}_0 - \hat{\beta}_1 x_2)^2 + \ldots + (y_n - \hat{\beta}_0 - \hat{\beta}_1 x_n)^2$$

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x},$$

Then we minimize the RSS

Minimizing $\mathrm{RSS} = (y_1 - \hat\beta_0 - \hat\beta_1 x_1)^2 + (y_2 - \hat\beta_0 - \hat\beta_1 x_2)^2 + \ldots + (y_n - \hat\beta_0 - \hat\beta_1 x_n)^2$

Means setting:

$$\boxed{\begin{aligned} \frac{\partial \mathrm{RSS}}{\partial \hat\beta_0} &= 0 \\ \frac{\partial \mathrm{RSS}}{\partial \hat\beta_1} &= 0 \end{aligned}}$$

Which gives:

$$(Z^T Z)B = Z^T Y$$

Fortunately, you don't have to do the math!!!

Which is solved by:

$$B = (Z^T Z)^{-1} Z^T Y$$

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \ldots \\ y_n \end{bmatrix} \quad B = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} \quad Z = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \\ \ldots & \ldots \\ 1 & x_n \end{bmatrix}$$

Which is the same thing as:

$$\hat\beta_1 = \frac{\sum_{i=1}^n (x_i - \bar x)(y_i - \bar y)}{\sum_{i=1}^n (x_i - \bar x)^2}$$

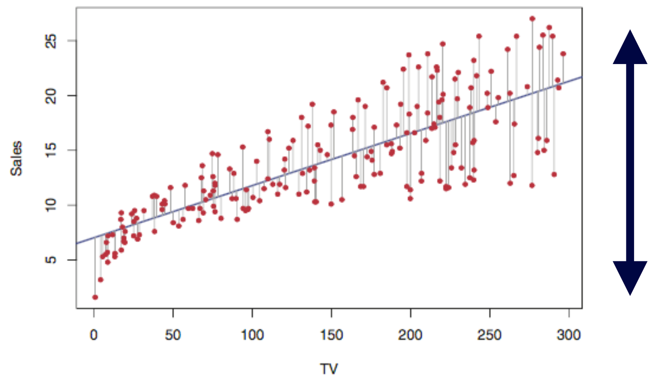
$$\hat\beta_0 = \bar y - \hat\beta_1 \bar x,$$

Other ways to measure accuracy of the model – how are we doing overall?

ESS = TSS - RSS ("explained sum of squares")

R² statistic $$R^2 = \frac{TSS - RSS}{TSS} = \frac{ESS}{TSS}$$

A scale-invariant measure (ranges between [0,1]) that explains ”the proportion of the variability of Y that is explained by X”



Differences with simple least squares and the rest of ML

- There's a LOT of useful statistics that you can get out of SLR/ORL
  - Estimates in uncertainties in parameters
  - Hypothesis testing about whether the parameters are statistically different than zero, and therefore relevant
  - Check the "statsmodel" package for more
- Most of them are used nor not possible to use for ML, so we won't cover them here.
  - If you have millions of parameters, uncertainties are not that useful, and hypothesis testing becomes messy
- For deep learning, you don't even TRY to get to the true minimum, since it's almost certainly overfit

To the notebook!

Multiple Linear Regression

- Concept: independently assess the variation in Y with different values of X:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon,$$

- As with SLR, the coefficients are determined by setting the analytical partial derivatives to zero and solving the resultant p+1 linear equations

We again minimize the RSS

$$\frac{\partial \mathrm{RSS}}{\partial \vec{\beta}} = 0$$

Which gives:

$$(Z^T Z)B = Z^T Y$$

Which is solved by:

$$B = (Z^T Z)^{-1} Z^T Y$$

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \ldots \\ y_n \end{bmatrix} \quad B = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_0 \\ \ldots \\ \beta_p \end{bmatrix} \quad Z = \begin{bmatrix} 1 & x_{11} & x_{21} & \ldots & x_{p1} \\ 1 & x_{12} & x_{22} & \ldots & x_{p2} \\ 1 & x_{13} & x_{23} & \ldots & x_{p3} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 1 & x_{1n} & x_{2n} & \ldots & x_{pn} \end{bmatrix}$$

More numerically robust to solve this using np.linalg.solv($Z^T Z$,$Z^T Y$)

Visualizing multiple linear regression

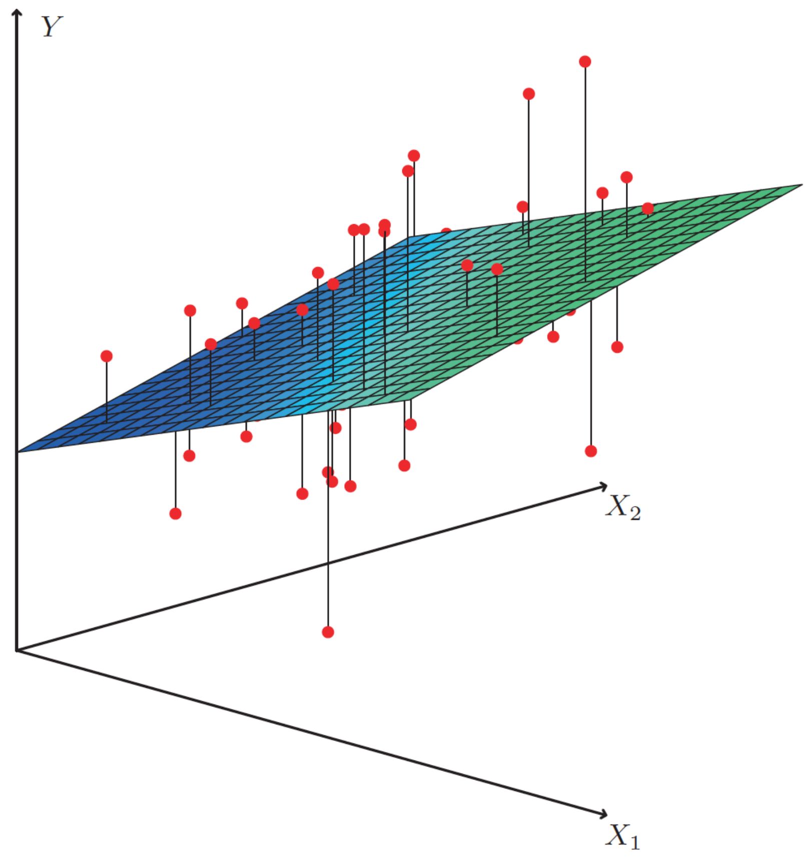


FIGURE 3.4. *In a three-dimensional setting, with two predictors and one response, the least squares regression line becomes a plane. The plane is chosen to minimize the sum of the squared vertical distances between each observation (shown in red) and the plane.*

Big picture concepts

- Tempting to simply add one term for each feature in X and see how good of a fit we obtain, but that doesn’t give us much inference
- There is a huge risk of overfitting with using a lot of parameters!
- We can use a new different type of hypothesis test to find out if any of the parameters are significant
  - Not generally useful in ML
- We can use some algorithms (selection algorithms) to try and reduce the number of parameters
  - Again, not as generally useful in ML

To the notebook!

General Linear Models

$$Y = \beta_0 + \beta_1 f_1(x_1) + \beta_2 f_2(x_2) + \cdots \beta_p f_p(x_p)$$

Works for any set of functions *f*; can be different for each x_i

$$Y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$$

Can be polynomials

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_{1,2} x_1 x_2$$

Can have linear terms of multiple variables

$$Y = \beta_0 + \beta_1 f_1(x_1) + \beta_2 f_2(x_2) + \beta_{1,2} f_{1,2}(x_1, x_2)$$

Can have functions of multiple variables

Can still implement in LinearRegression by using functions of X as predictors

"Clicker" question

Which of the following is a truly nonlinear model?

A. $Y = a + b X^2 + c X^4$

B. $Y = a X_1 + b X_2 + c X_3 X_4$

C. $Y = a X + \ln X^b$

D. $Y = a X + b \ln(X - c)$

E. $Y = a X_1 + b(1 - X_2)^2$

General Linear Models: We again minimize the RSS

$$Y = \beta_0 + \beta_1 f_1(x_1) + \beta_2 f_2(x_2) + \cdots \beta_p f_p(x_p)$$

$$\mathrm{RSS} = \sum_i (Y_i - \hat{Y}_i)^2$$

$$\frac{\partial \mathrm{RSS}}{\partial \vec{\beta}} = 0$$

Which gives:

$$(Z^T Z)B = Z^T Y$$

Which is solved by:

$$B = (Z^T Z)^{-1} Z^T Y$$

$$Y = \begin{array}{|c|} \hline y_1 \\ \hline y_2 \\ \hline y_3 \\ \hline \ldots \\ \hline y_n \\ \hline \end{array} \qquad B = \begin{array}{|c|} \hline \beta_0 \\ \hline \beta_1 \\ \hline \beta_0 \\ \hline \ldots \\ \hline \beta_p \\ \hline \end{array} \qquad Z = \begin{array}{|c|c|c|c|c|} \hline 1 & f_1(x_{11}) & f_2(x_{21}) & \ldots & f_p(x_{p1}) \\ \hline 1 & f_1(x_{12}) & f_2(x_{22}) & \ldots & f_p(x_{p2}) \\ \hline 1 & f_1(x_{13} & f_2(x_{23}) & \ldots & f_p(x_{p3}) \\ \hline \ldots & \ldots & \ldots & \ldots & \ldots \\ \hline 1 & f_1(x_{1n} & f_2(x_{2n}) & \ldots & f_p(x_{pn}) \\ \hline \end{array}$$

Works for any set of functions *f*; can be different function for each x_i

How do I decide to use a more complicated linear model?

- Careful selection based on statistical analysis about model freedom (out of scope for today)
- Because the physics or underlying data structure say that those terms <u>should</u> have a given functional form
  - Heat capacities are known to be polynomial in T
  - Many quantities are naturally logarithmic/exponential/product/inverse
- Requires domain expertise!

What other kinds of basis functions can you use?

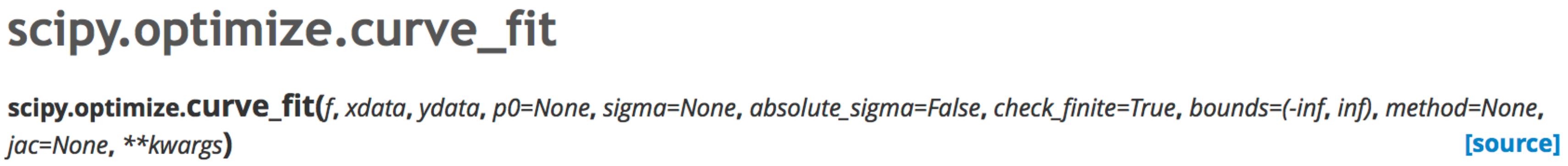
- Polynomials
- Periodic functions if dealing with fundamentally periodic behavior
  - Ocean temperature weather is a periodic function of day and year!
- Something else that comes from physics or other information
- Piecewise functions
  - Step functions
  - Splines

Fully nonlinear models

- Where to go when you have a true nonlinear model to fit/train?
- Write a function for the RSS or other loss function

$$\mathrm{RSS} = \sum_i \left(Y_i - f(\overrightarrow{X}, \vec{p}) \right)^2$$

- Minimize it! Lots of tools available



scipy.optimize.curve_fit

```
scipy.optimize.curve_fit(f, xdata, ydata, p0=None, sigma=None, absolute_sigma=False, check_finite=True, bounds=(-inf, inf), method=None, jac=None, **kwargs)
```
[source]

- Most learning kits have different optmization methods built in
- There's still a formula for errors in parameters, but it gets messy.

Types of nonlinear models

- K-nearest neighbors, decision tree, or support vector regression or classification
- <u>All</u> types of neural nets
- A key point is that they require numerical solution.
  - This means the strategies to solve them require a lot of fiddling with and testing
  - Can give very different results from trial to trial
  - We will spend time with this in the context of neural nets

Go to the notebook!

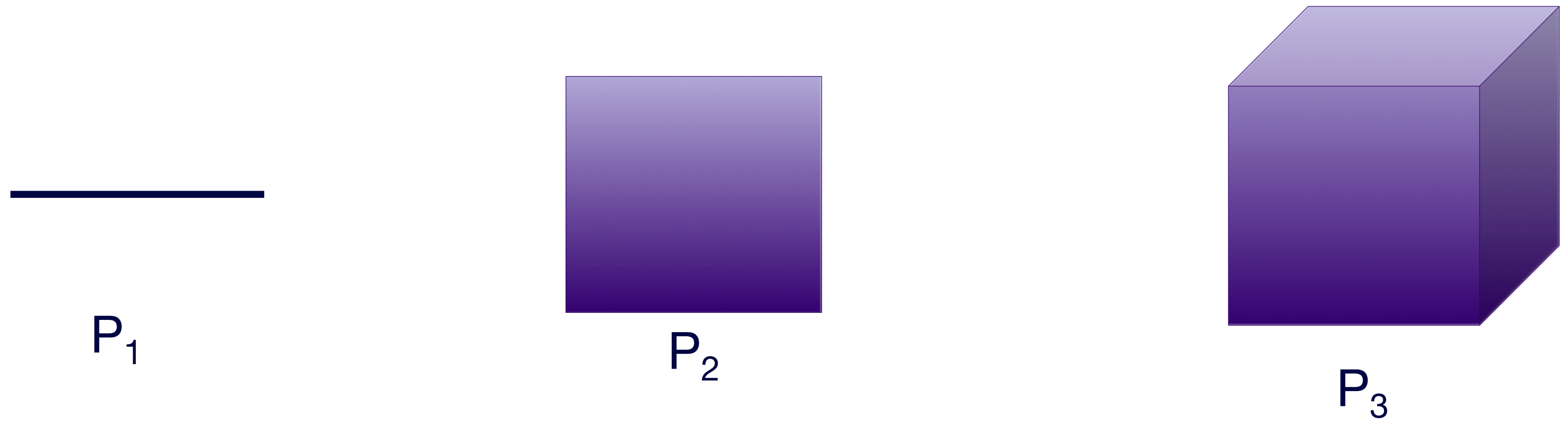
When good models go bad

- Increasing parameters can be seductive, especially if you “feel” like you have a large data set:
  - BUT in high dimension your training data may severely undersample the space of parameters
- Supervised learning models are powerful:
  - If you have a lot of (relevant) data you can often get a model that is predictive

The curse of dimensionality

Consider a situation in which you need n=10 data points in order to cover response range (in Y) for each X_i.

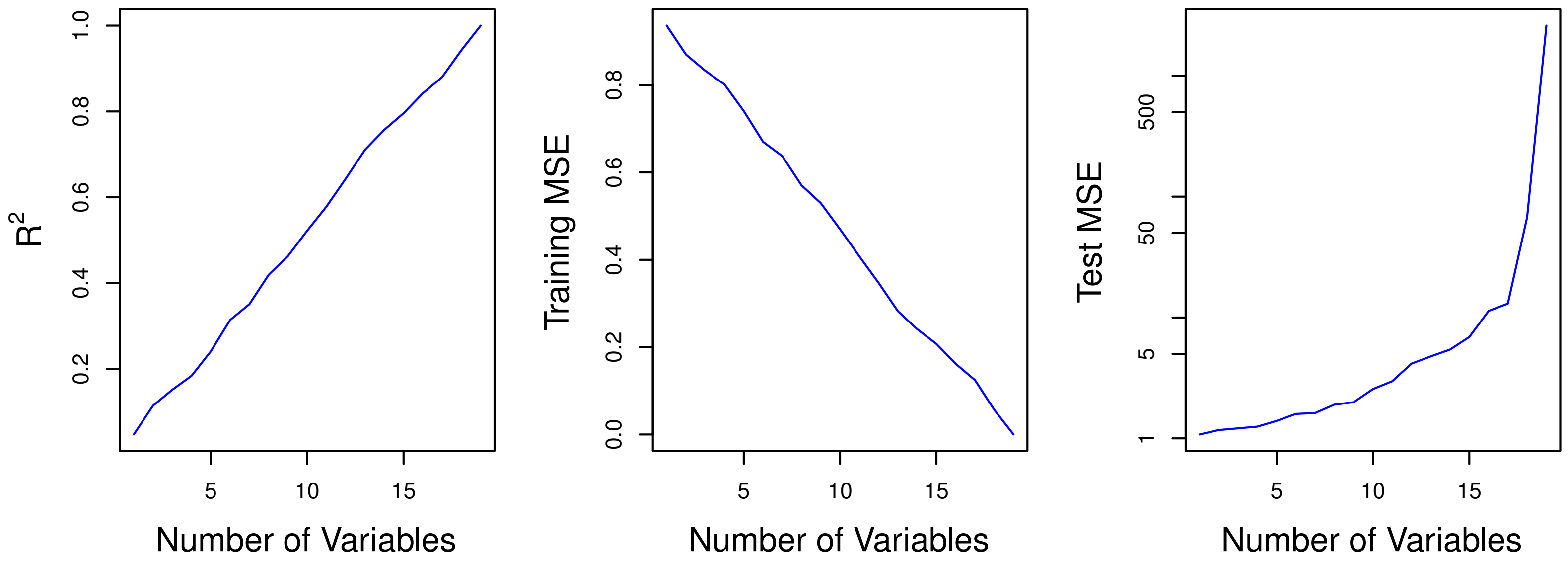
For P=1, n=10, P=2, need n=100, P=3, need n=1000…



Our ability to capture significant fractions of the predictor space collapses after a few dimensions

VERY EASY TO OVERFIT MODELS WITH MANY PARAMETERS

- If there are many parameters
- There are many coefficients in the model
- You can easily overfit
- Cannot use R^2 or MSE of training set
- Must use MSE of validation set or other measure.



Shown in graph: data set R^2, training MSE and test MSE as a function of adding variables unrelated to output

Ridge regression

Ridge replaces the RSS term: $\mathrm{RSS} = \sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2$

with a new minimizer that includes a so-called shrinkage or regularization penalty:

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

- The adjustable parameter λ, trades the vanilla RSS with a penalty for nonzero coefficients.
- As λ increases to infinity, the minimized error drives all of the coefficients to zero

Regularization: big picture concepts

- **Why would do we want to do this?**
  - Recall that models with <u>more parameters</u> will better estimate the training set, reducing the <u>training error.</u>
  - However, adding more parameters <u>increases the testing error.</u>
  - The variance of response Y is increased via the bias-variance tradeoff
  - Reducing the <u>magnitude of the coefficients</u> allows the most important variables to account for most of the fit
- Many ML regression methods use this, not just linear regression

Ridge in practice

Unlike RSS minimizers, which are scale equivariant, the predictor data <u>must be standardized</u> in regularization methods:

Standard Least Squares

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

Ridge regression

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

$$\tilde{x}_{ij}=\frac{x_{ij}}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_{ij}-\overline{x}_j)^2}} \longleftarrow$$

<u>standardized data</u>

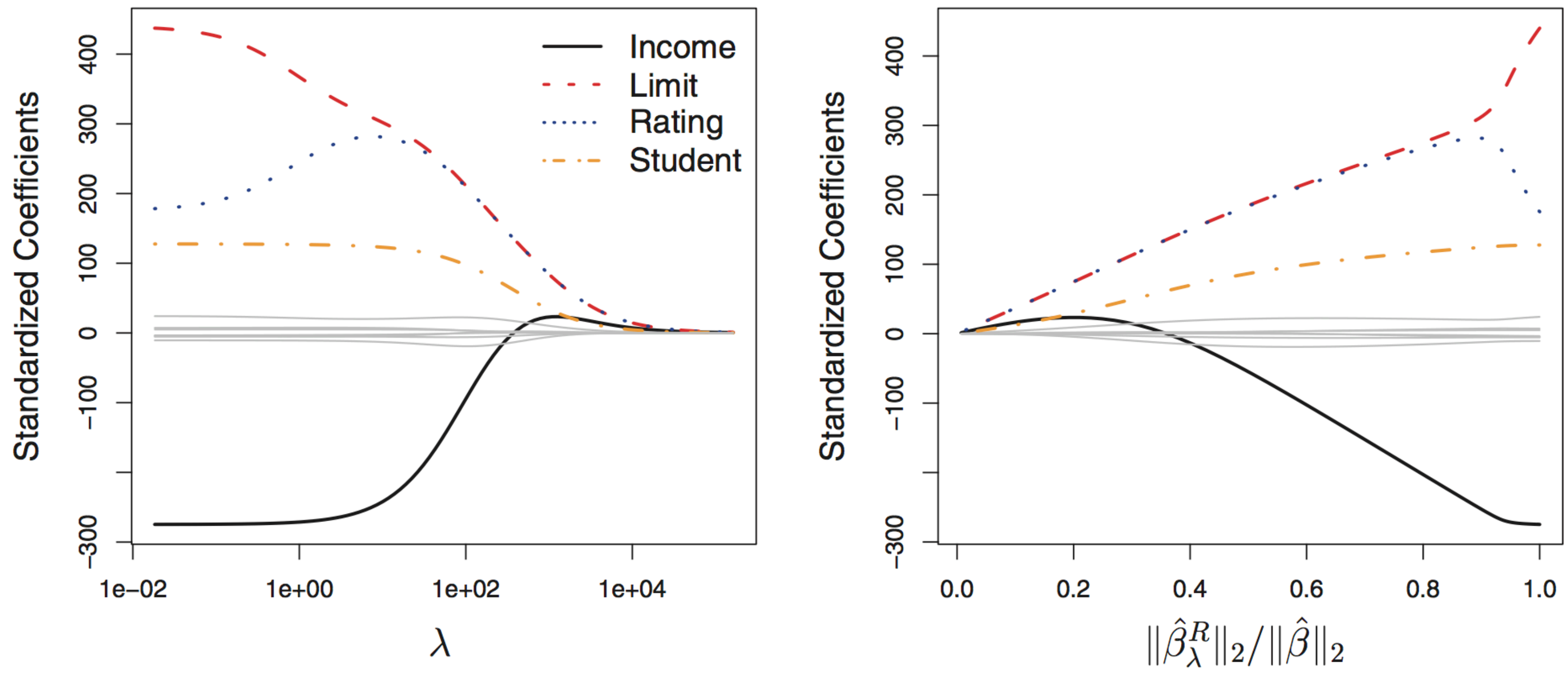
Ridge regression process

- Fit a series of ridge regression models
  - Across a wide range of λ and
  - Track the coefficient values and test set error (or estimate) as λ is changed
- Determine the model w/smallest validation error
- Then estimate the true test error w/new data that was not used in the training
- In practice: use cross-validation!

Hyperparameters are choices that you make that are not the parameters of the model

- What should the value of λ be for a ridge regression?
- What should the value of K be in a K-neighbors algorithm?
- How do you choose them?

How parameters change with change in shrinkage parameter

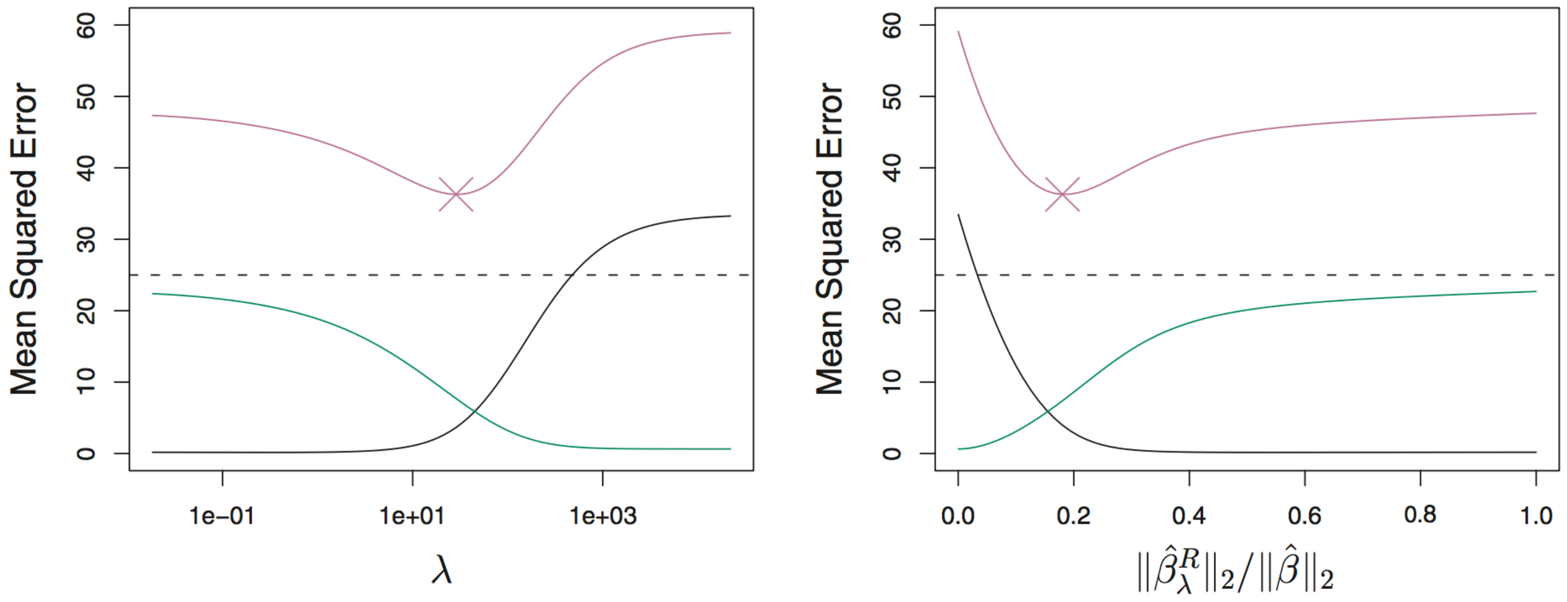


$\hat{\beta}$ = vector of least squares coefficient estimates

$\|\beta\|_2 = \sqrt{\sum_{j=1}^{p} \beta_j{}^2}$ = L_2 norm (distance of β from 0) = size of parameters

$\|\hat{\beta}_\lambda^R\|_2/\|\hat{\beta}\|_2,$ = amount coefficients are smaller

Reducing flexibility in a controlled way can minimize MSE



Bias^2 = black
Variance = green
MSE = purple

- When λ = 0, Ridge = linear regression
  - Variance is high, but there is no bias
- As λ increases, the flexibility of the regression decreases
  - Leads to decreased <u>variance</u> of prediction
  - But, increased bias, since the parameters are restrained

LASSO regression

- **L**east **A**bsolute **S**hrinkage and **S**election **O**perator
- Ridge regression does not set any of the coefficients exactly to zero but can shrink all of them
  - Final model still includes all p predictors
- The LASSO regression was developed and , inspired by ridge,
  - To provide the possibility that some of the coefficients can take a value of zero
- Like ridge, the LASSO operator is

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

Measuring size of parameters

$$\lambda \sum_{j=1}^{p} |\beta_j|$$

Size via L_1 norm

$$\lambda \sum_{j=1}^{p} \beta_j^2$$

Size via L_2 norm

An extra bonus in LASSO: deselecting variables that don't contribute to the fit

The key difference is the penalty due to nonzero coefficients.

LASSO (L1)

$$\underset{\beta}{\text{minimize}} \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \right\} \quad \text{subject to} \quad \sum_{j=1}^{p} |\beta_j| \leq s$$

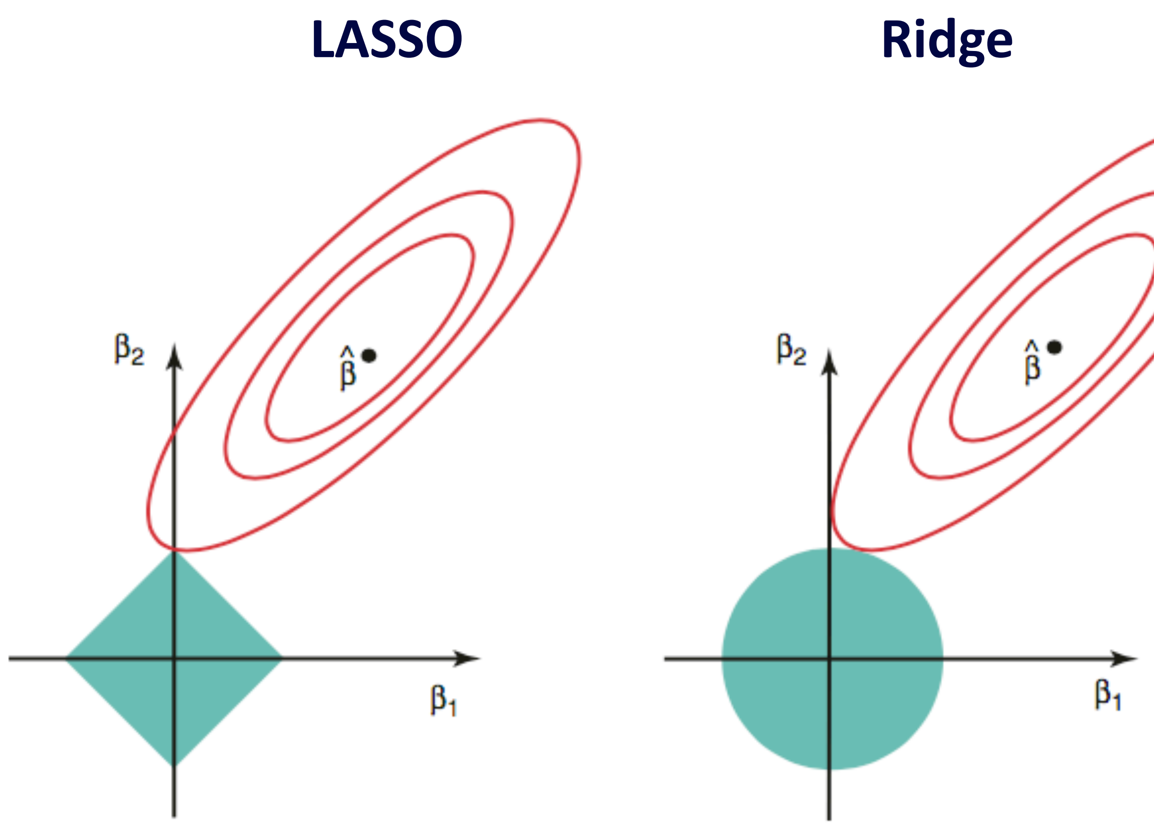
and

Ridge (L2)

$$\underset{\beta}{\text{minimize}} \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \right\} \quad \text{subject to} \quad \sum_{j=1}^{p} \beta_j^2 \leq s,$$

A mathematical result of LASSO is that some of the β values can be zero at the minimum error

Why can LASSO coefficients become zero?



Red = lines of constant RSS

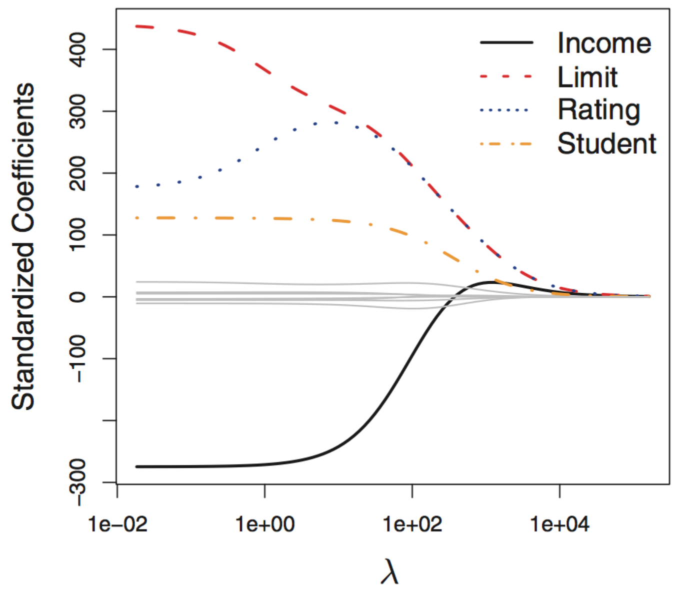
Blue = constraint zone assuming a fixed value of s

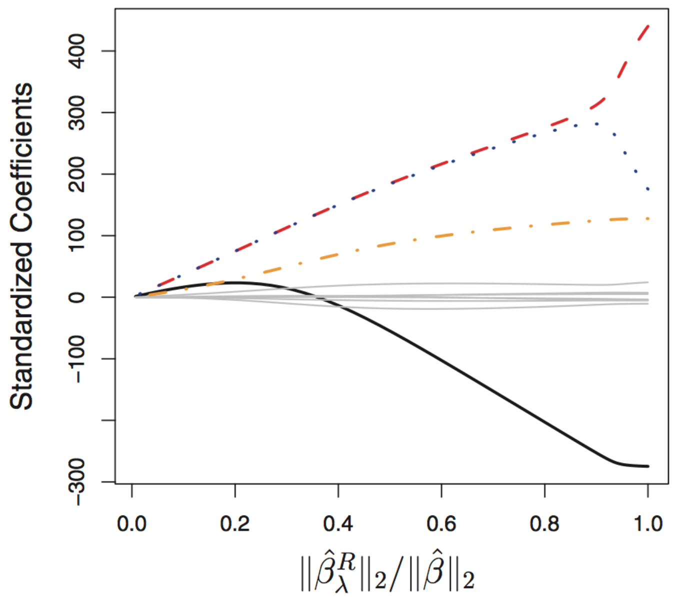
FIGURE 6.7. *Contours of the error and constraint functions for the lasso* (left) *and ridge regression* (right). *The solid blue areas are the constraint regions, $|\beta_1| + |\beta_2| \leq s$ and $\beta_1^2 + \beta_2^2 \leq s$, while the red ellipses are the contours of the RSS.*

Axes (where one β is zero) are further from the origin in LASSO than the other points

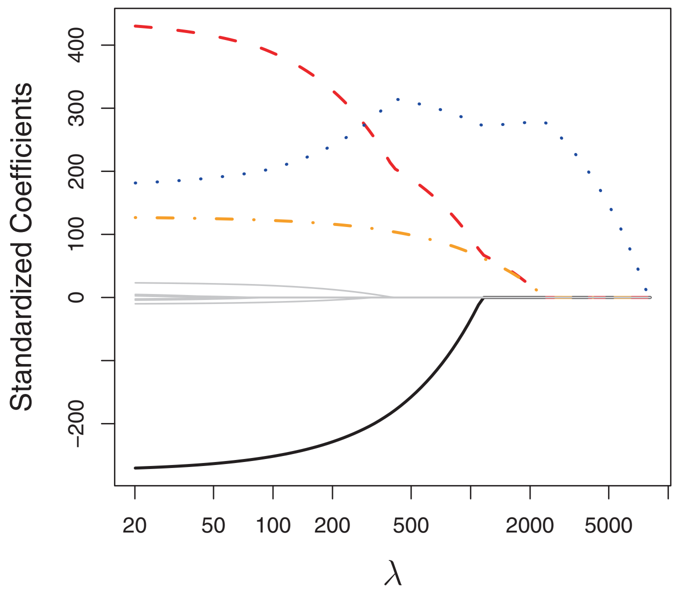
Ridge vs LASSO coefficient size

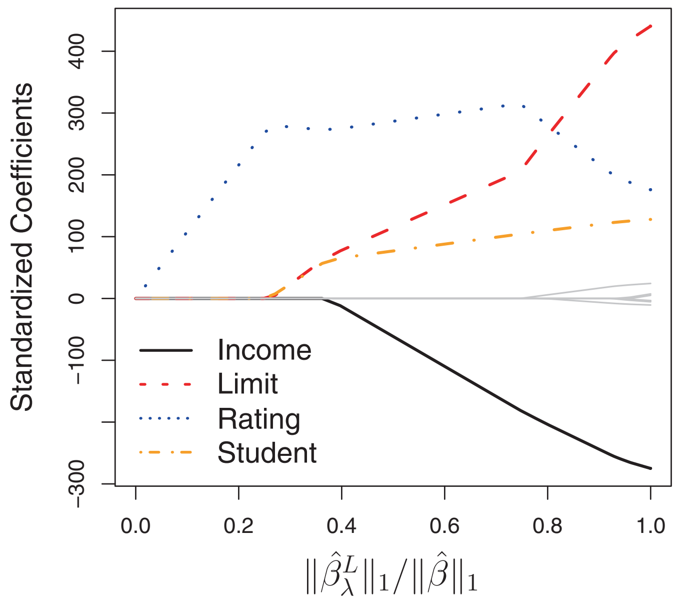
Ridge





LASSO



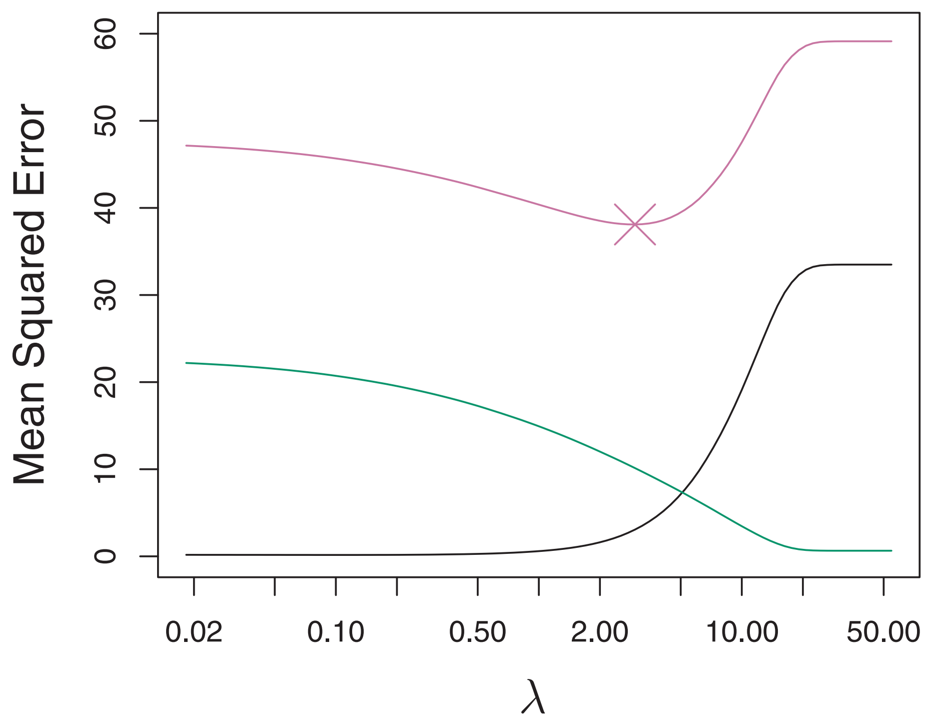


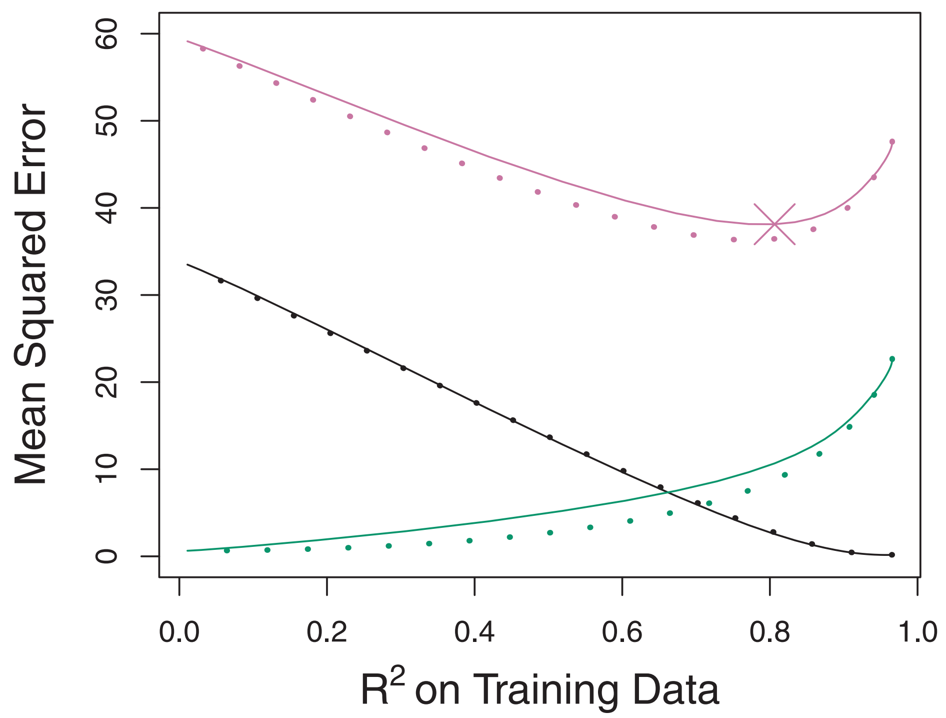
LASSO vs. Ridge

- If ALL variables predictive, LASSO and ridge are similar, with ridge slightly better

Bias² = black
Variance = green
MSE = purple

LASSO - solid
Ridge - dotted



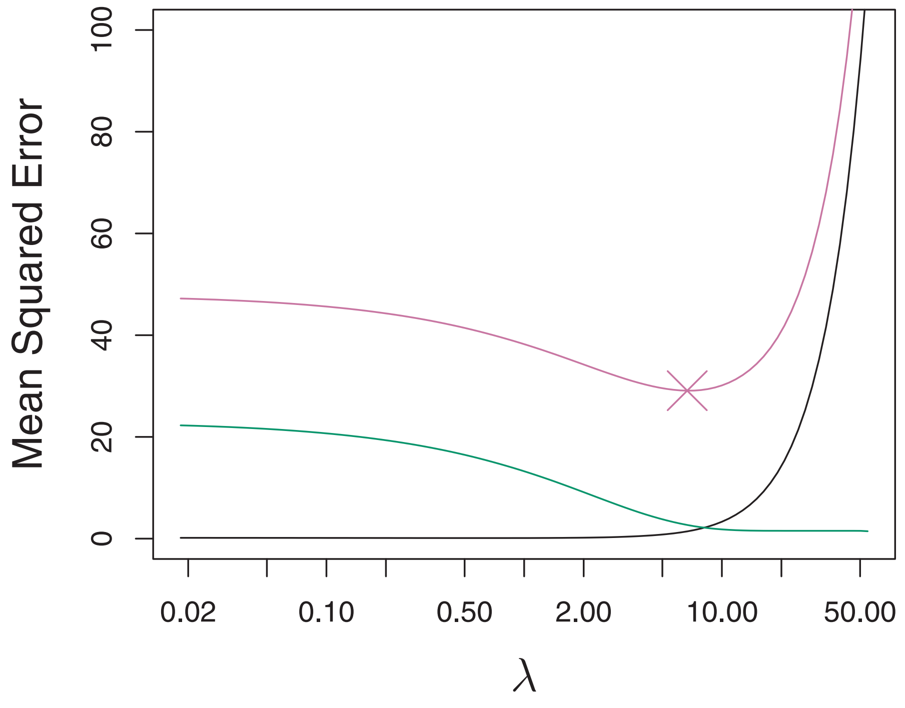


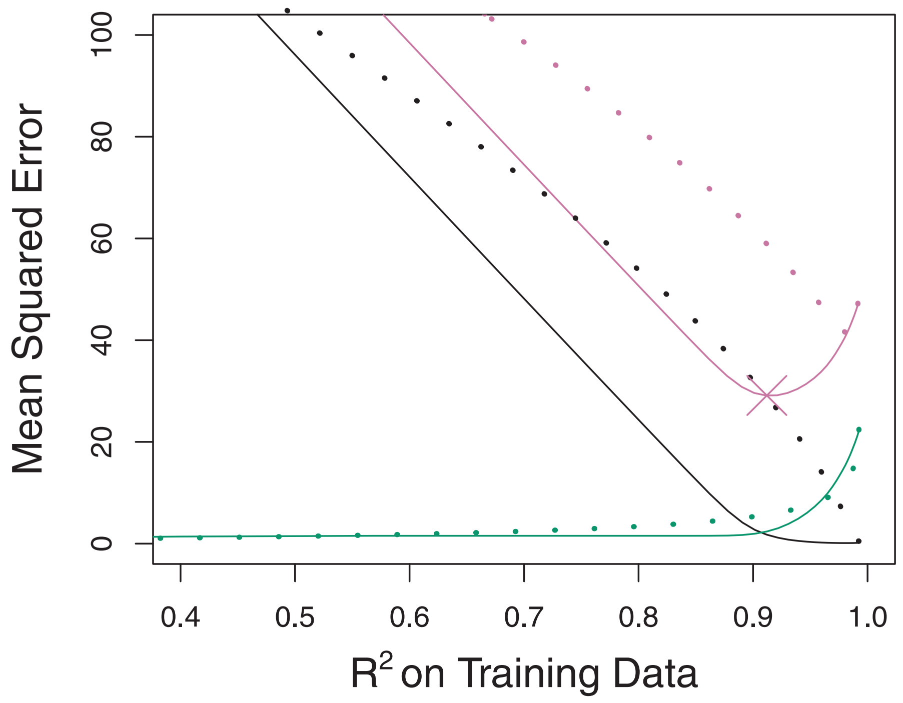
LASSO vs. Ridge

- If only a <u>few</u> variables are predictive, LASSO has real advantages

Bias² = black
Variance = green
MSE = purple

LASSO - solid
Ridge - dotted





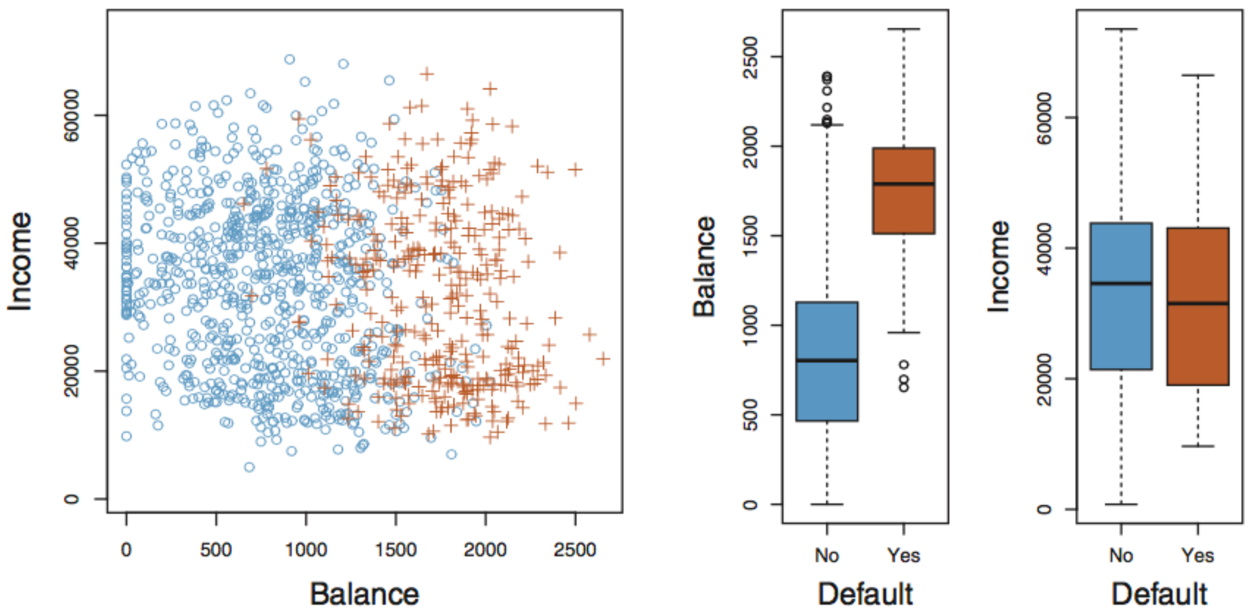
Directly relevant with deep learning

- L1 and L2 regularization **frequently used** in many ML loss functions
- Prevents overfitting when there are many, many parameters

Go to the notebook!

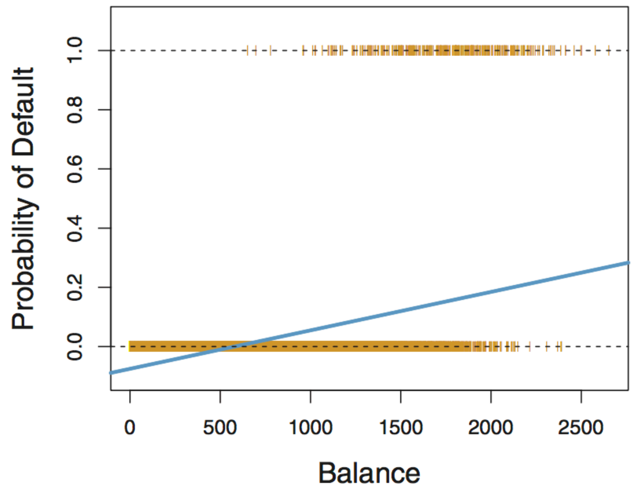
Classification with the logistic model

- Consider the data below:
  - We are trying to make a guess as to whether someone will default on a loan on the basis of their bank account balance and income level
  - We have a two choice classification: will they default or not?



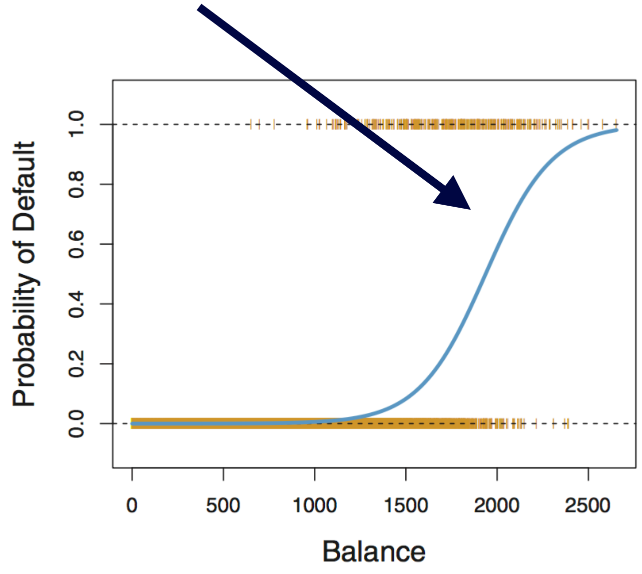
Why logistic regression?

- Let's say we trained a linear model such that the output was either 1 or 0.
- Blue line is what our predictive might look like



- We would like something more like this, with a probability of default given for each input

$$\Pr(\texttt{default} = \texttt{Yes}|\texttt{balance})$$



The logistic function

$$p(X) = \beta_0 + \beta_1 X.$$

simple linear model

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}}.$$

logistic equation

$$\frac{p(X)}{1 - p(X)} = e^{\beta_0+\beta_1 X}.$$

Solve for $e^{\beta_0+\beta_1 X}$

Can anyone define “odds”?

If your probability of success is 1 in 5 (0.2), your odds of success are 1 in 4: p(x)/(1-p(x)) = 0.2 / (1-0.2) = 0.25

The logistic function

$$\frac{p(X)}{1 - p(X)} = e^{\beta_0 + \beta_1 X}.$$

$$\log \left(\frac{p(X)}{1 - p(X)} \right) = \beta_0 + \beta_1 X.$$

The left side of the equation is log of the odds
Called the "logit",
Equation elates the change in log of odds of success w/change in X

The logit is a linear function

Probability goes from 0 to 1
Log probability goes from -∞ to 1
Logit (log odds) goes from -∞ to ∞

Multiple ways to solve it

- Use linear regression to solve: $\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X.$
    - Predict $\log\left(\frac{p(X)}{1-p(X)}\right)$. If > 0, yes, if < 0. No.
- Use nonlinear regression of $p(X) = \frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}}.$
    - Predict p(x). If $>$ 0.5, yes, if < 0.5, no.

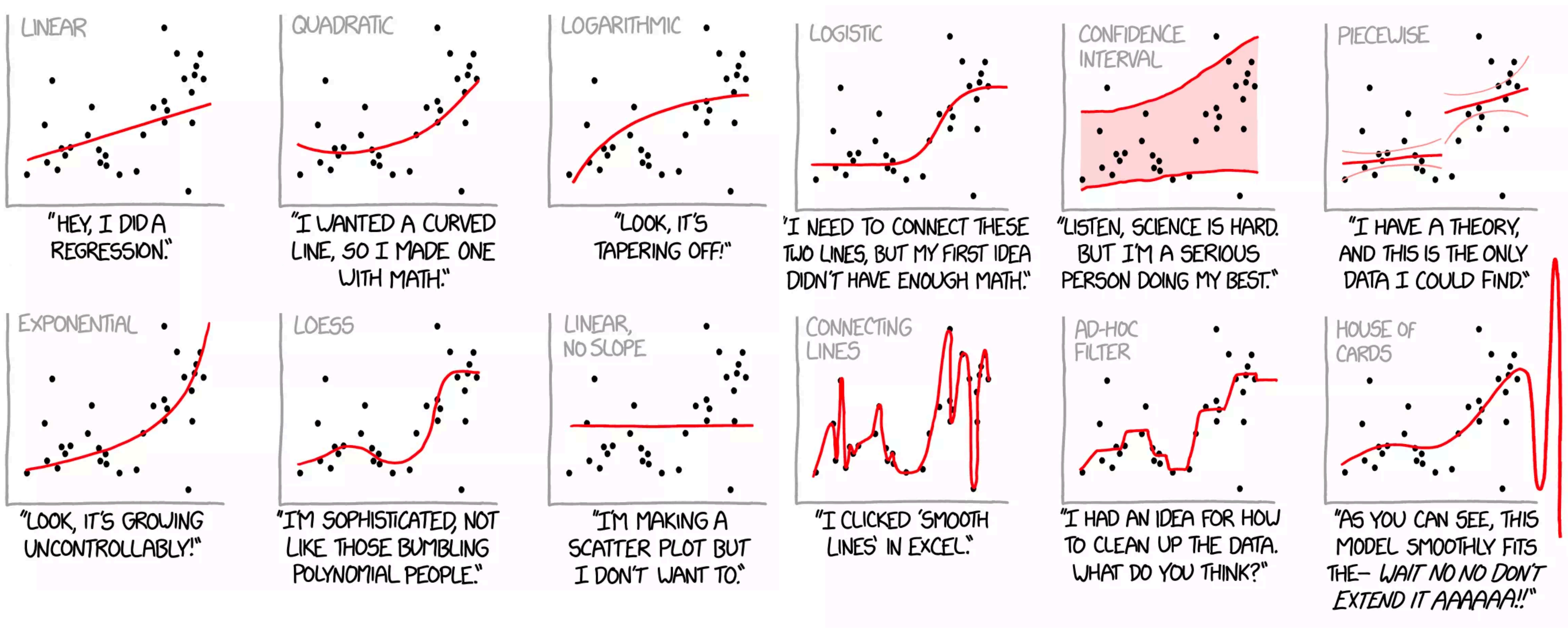
Go to the notebook!

Multiple logistic regression

- Extension is very straightforward!

$$\ln\left(\frac{P(X)}{1-P(X)}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p$$

Curve Fitting Methods and the Messages They Send


LINEAR
"HEY, I DID A REGRESSION."
QUADRATIC
"I WANTED A CURVED LINE, SO I MADE ONE WITH MATH."
LOGARITHMIC
"LOOK, IT'S TAPERING OFF!"
LOGISTIC
"I NEED TO CONNECT THESE TWO LINES, BUT MY FIRST IDEA DIDN'T HAVE ENOUGH MATH."
CONFIDENCE INTERVAL
"LISTEN, SCIENCE IS HARD. BUT I'M A SERIOUS PERSON DOING MY BEST."
PIECEWISE
"I HAVE A THEORY, AND THIS IS THE ONLY DATA I COULD FIND."
EXPONENTIAL
"LOOK, IT'S GROWING UNCONTROLLABLY!"
LOESS
"I'M SOPHISTICATED, NOT LIKE THOSE BUMBLING POLYNOMIAL PEOPLE."
LINEAR, NO SLOPE
"I'M MAKING A SCATTER PLOT BUT I DON'T WANT TO."
CONNECTING LINES
"I CLICKED 'SMOOTH LINES' IN EXCEL."
AD-HOC FILTER
"I HAD AN IDEA FOR HOW TO CLEAN UP THE DATA. WHAT DO YOU THINK?"
HOUSE OF CARDS
"AS YOU CAN SEE, THIS MODEL SMOOTHLY FITS THE- WAIT NO NO DON'T EXTEND IT AAAAAA!!"

https://xkcd.com/2048/