Regularized Regression

Ryan Henning
Frank Burkholder
Elliot Cohen

galvanize

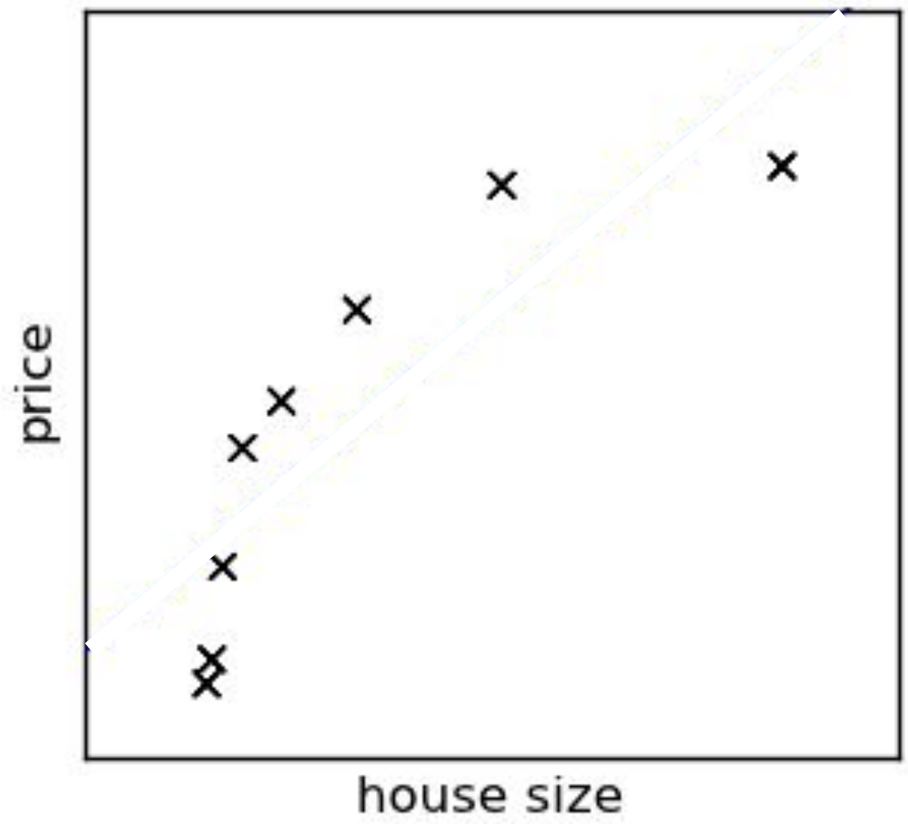
Learning Objectives:

- Explain the shortcomings of un-regularized regression
- Relate shortcomings to:
    - The Curse of Dimensionality
    - The Bias-Variance Tradeoff
- Mitigate shortcomings with:
    - Ridge Regression
    - Lasso Regression

Let's suppose we want to model housing prices as a function of house size.

Do you see a signal?

How is house price is related to house size?

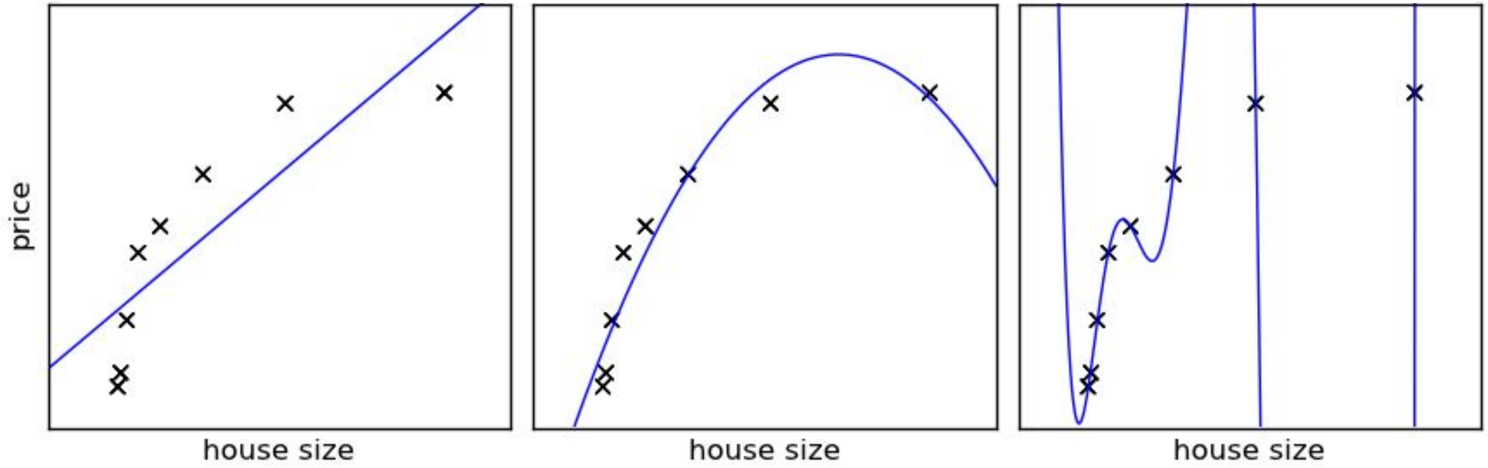


Fit a polynomial model to data

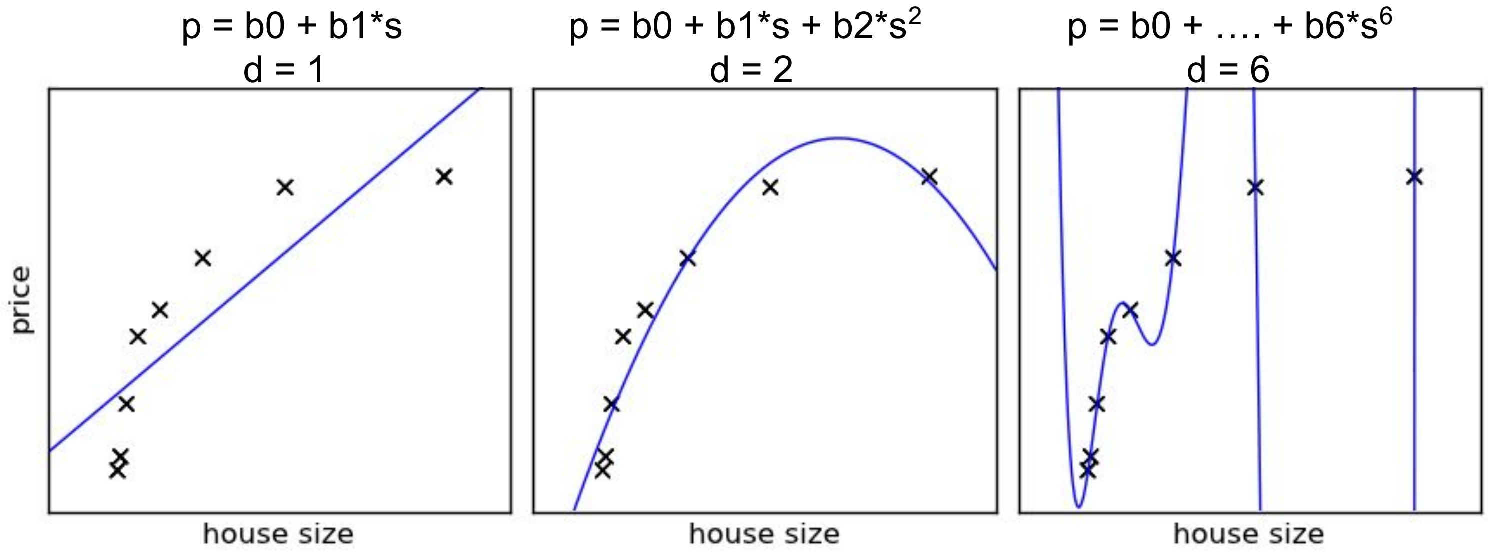
$p = b0 + b1*s$
$d = 1$

$p = b0 + b1*s + b2*s^2$
$d = 2$

$p = b0 + ... + b6*s^6$
$d = 6$

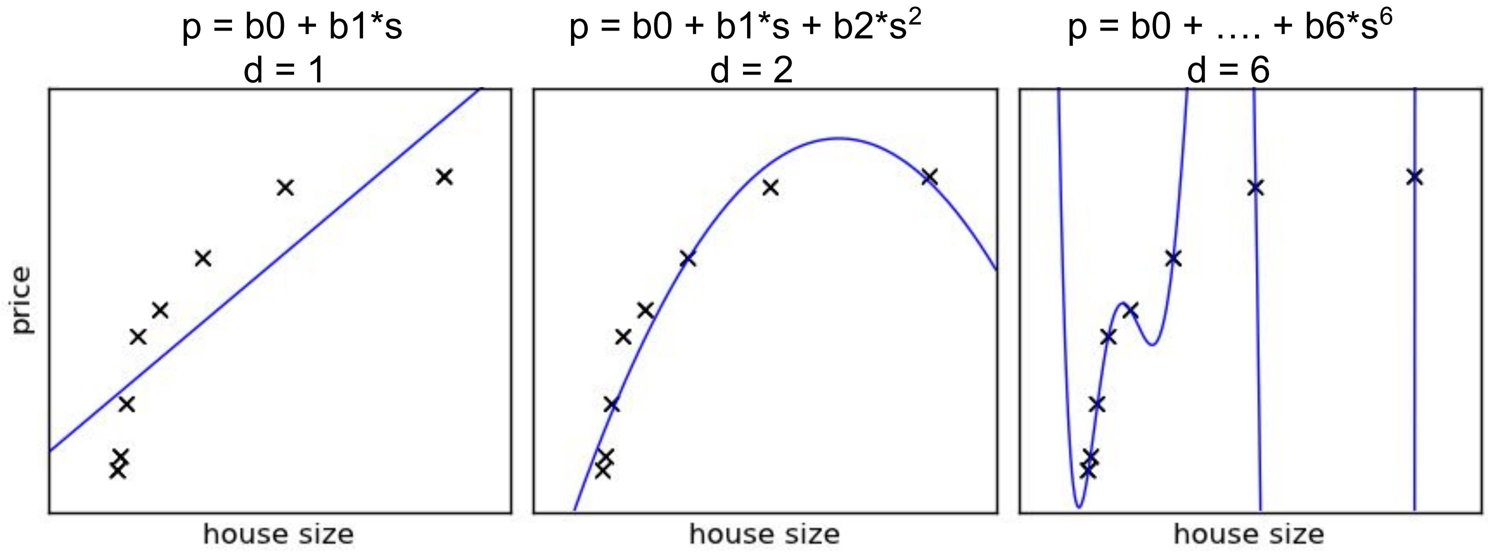


Fit a polynomial model to data



First order fit.
Underfit?
Good fit?
Overfit?

Example - let's fit this data



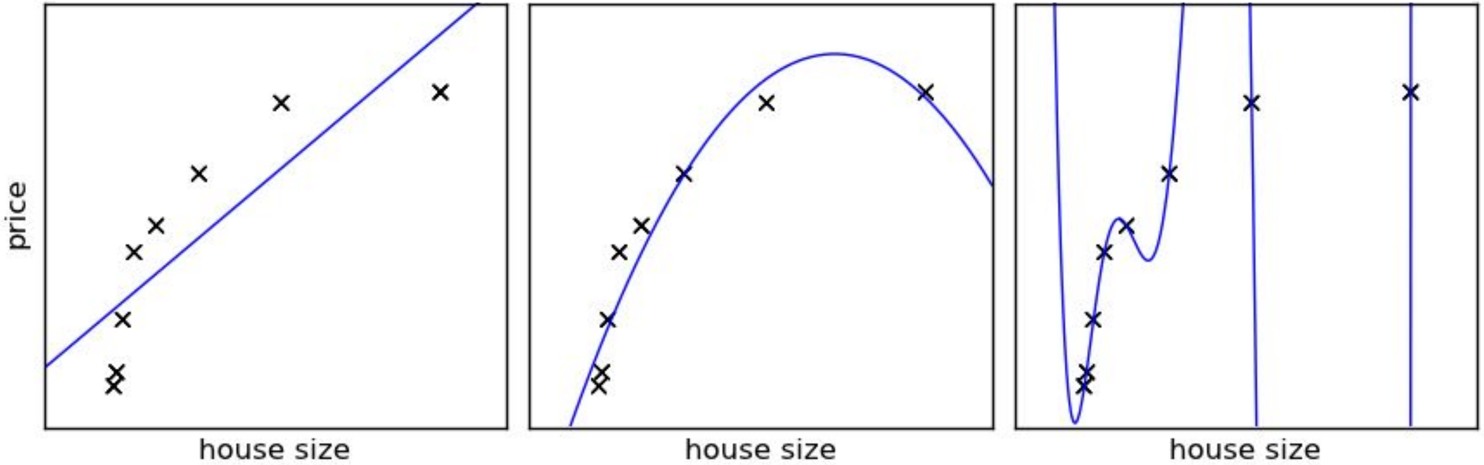
Likely underfit.
The fit doesn't fully capture the relationship between house size and price in the data.

Example - let's fit this data

$p = b0 + b1*s$
$d = 1$

$p = b0 + b1*s + b2*s^2$
$d = 2$

$p = b0 + ... + b6*s^6$
$d = 6$



Likely underfit.
The fit doesn't fully capture the relationship between house size and price in the data.

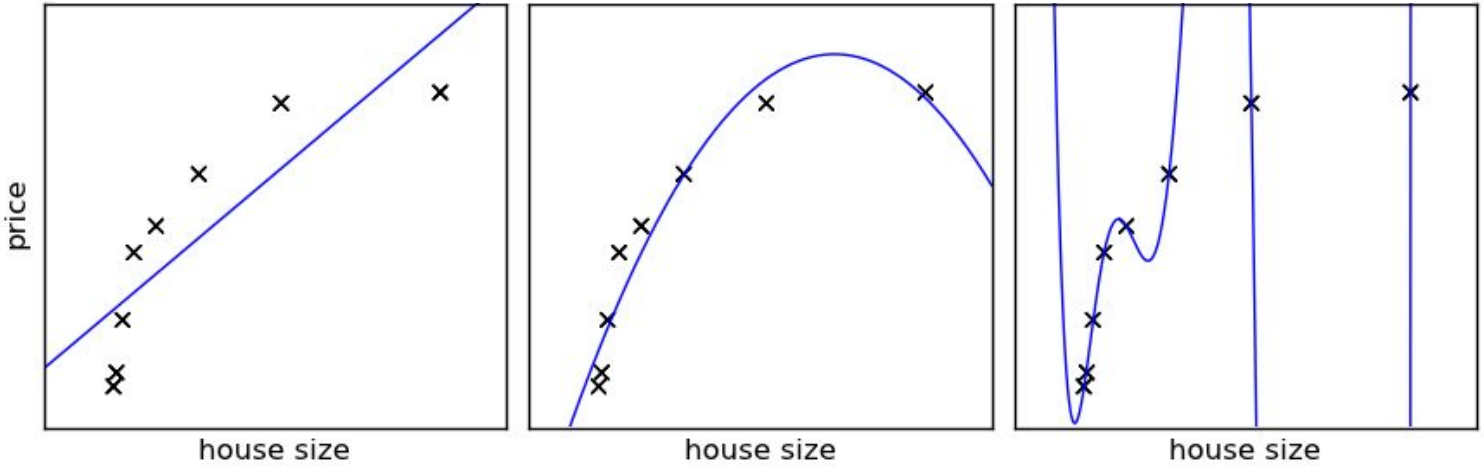
Sixth order fit.
Underfit?
Good fit?
Overfit?

Example - let's fit this data

$p = b0 + b1*s$
$d = 1$

$p = b0 + b1*s + b2*s^2$
$d = 2$

$p = b0 + ... + b6*s^6$
$d = 6$

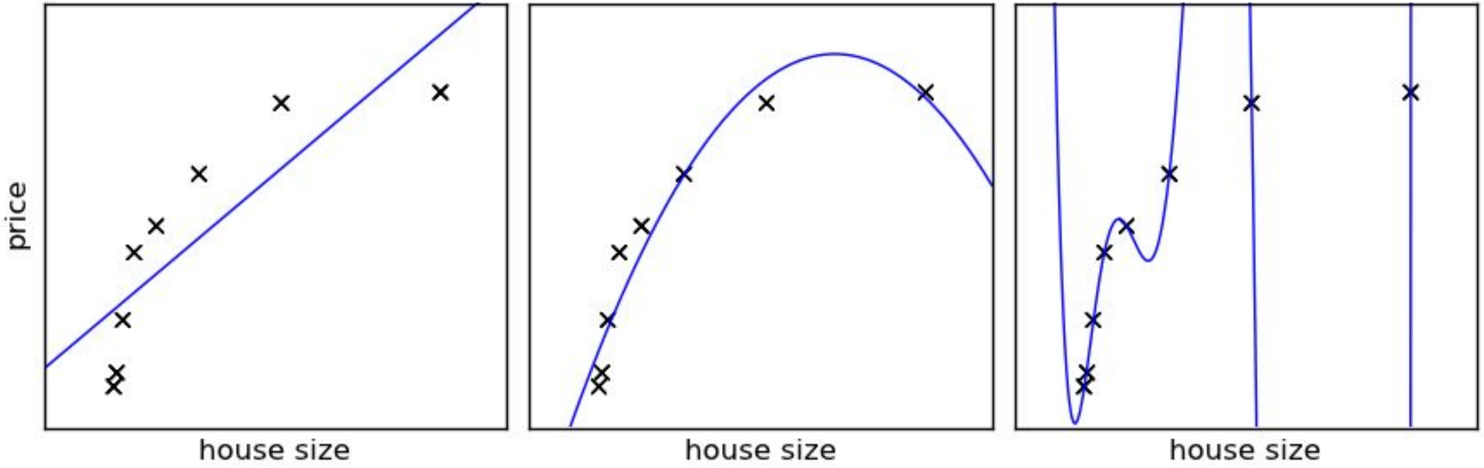


Likely underfit.
The fit doesn't fully capture the relationship between house size and price in the data.

Likely overfit.
Fits perfectly to the data, including signal and noise.

Example - let’s fit this data

$p = b0 + b1*s$, $d = 1$ | $p = b0 + b1*s + b2*s^2$, $d = 2$ | $p = b0 + ... + b6*s^6$, $d = 6$

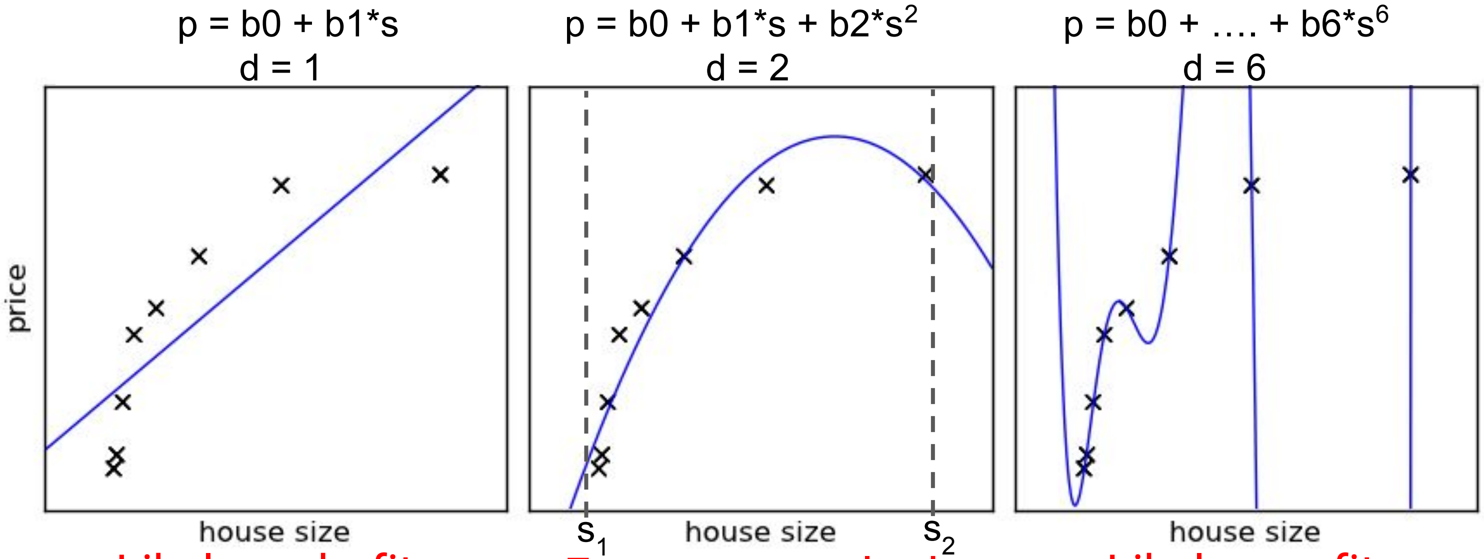


Likely underfit.
The fit doesn’t fully capture the relationship between house size and price in the data.

Second order fit.
Underfit?
Good fit?
Overfit?

Likely overfit.
Fits perfectly to the data, including signal and noise..

Example - let's fit this data



Likely underfit.
The fit doesn't fully capture the relationship between house size and price in the data.

From s_1 to s_2, looks good…
… but what about for house sizes $>s_2$?
Functional form flawed (see logarithmic)

Likely overfit.
The fit perfectly fits the data, including noise in the data.

Interactive Demo

http://madrury.github.io/smoothers/

In high dimensions, data is (usually) sparse

Curse of Dimensionality

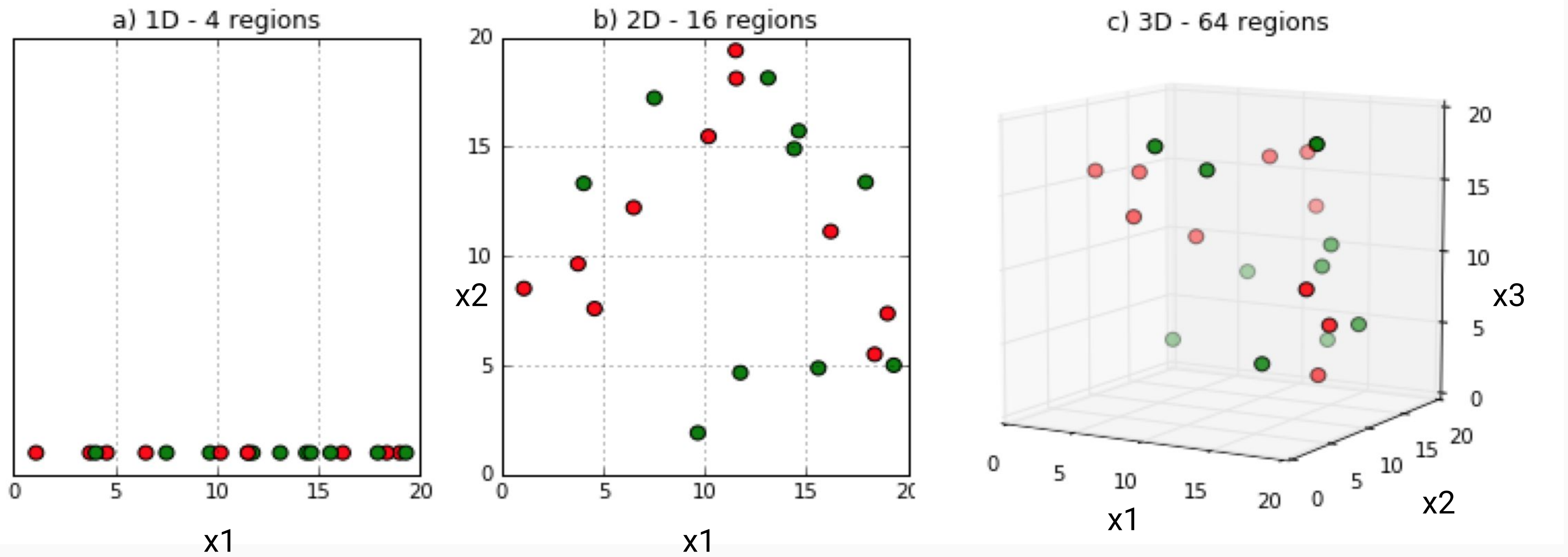
As the number of dimensions increase, the volume that data can occupy grows exponentially.

D = number of dimensions

N = number of datapoints

sampling density is proportional to $N^{\frac{1}{D}}$.

Curse of Dimensionality example



sampling density $N_1^{1/D_1} \propto N_2^{1/D_2}$

http://cleverowl.uk/2016/02/06/curse-of-dimensionality-explained/

Curse of Dimensionality: Breakout

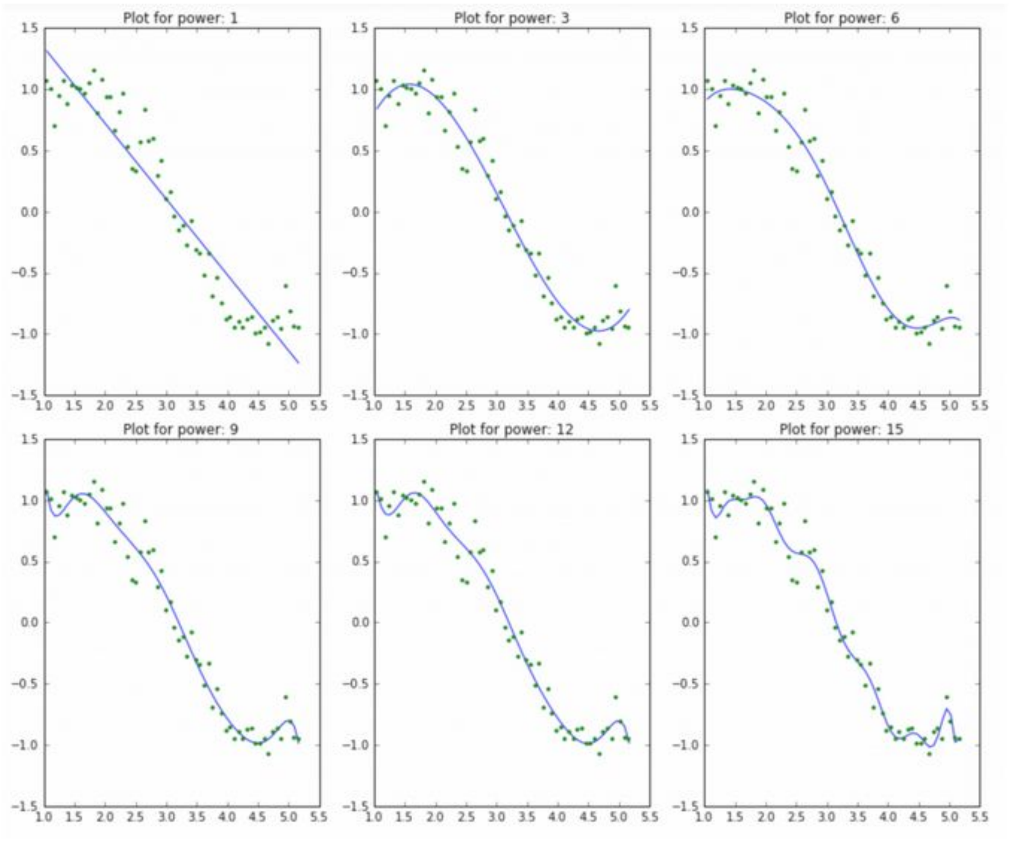
Suppose I want to model housing prices (y) as a function of square footage (x1), number of bedrooms (x2), number of bathrooms (x3), vintage (x4) and distance from downtown (x5). Suppose I have 1000 records (N).

What is the sampling density of my model?

Now suppose a colleague suggests I look at all pairwise interaction terms as well.

How many new features do I have?
What is the new sampling density?
How many new observations would I need to maintain the original sampling density?

Consider the following polynomial models



https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-ridge-lasso-regression-python/

Here are the coefficients of the models on the previous slide.
Notice how the magnitude of the coefficients increases as complexity increases

	rss	intercept	coef_x_1	coef_x_2	coef_x_3	coef_x_4	coef_x_5	coef_x_6	coef_x_7	coef_x_8	coef_x_9	coef_x_10	coef_x_11	c
model_pow_1	3.3	2	-0.62	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
model_pow_2	3.3	1.9	-0.58	-0.006	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
model_pow_3	1.1	-1.1	3	-1.3	0.14	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
model_pow_4	1.1	-0.27	1.7	-0.53	-0.036	0.014	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
model_pow_5	1	3	-5.1	4.7	-1.9	0.33	-0.021	NaN	NaN	NaN	NaN	NaN	NaN	N
model_pow_6	0.99	-2.8	9.5	-9.7	5.2	-1.6	0.23	-0.014	NaN	NaN	NaN	NaN	NaN	N
model_pow_7	0.93	19	-56	69	-45	17	-3.5	0.4	-0.019	NaN	NaN	NaN	NaN	N
model_pow_8	0.92	43	-1.4e+02	1.8e+02	-1.3e+02	58	-15	2.4	-0.21	0.0077	NaN	NaN	NaN	N
model_pow_9	0.87	1.7e+02	-6.1e+02	9.6e+02	-8.5e+02	4.6e+02	-1.6e+02	37	-5.2	0.42	-0.015	NaN	NaN	N
model_pow_10	0.87	1.4e+02	-4.9e+02	7.3e+02	-6e+02	2.9e+02	-87	15	-0.81	-0.14	0.026	-0.0013	NaN	N
model_pow_11	0.87	-75	5.1e+02	-1.3e+03	1.9e+03	-1.6e+03	9.1e+02	-3.5e+02	91	-16	1.8	-0.12	0.0034	N
model_pow_12	0.87	-3.4e+02	1.9e+03	-4.4e+03	6e+03	-5.2e+03	3.1e+03	-1.3e+03	3.8e+02	-80	12	-1.1	0.062	-
model_pow_13	0.86	3.2e+03	-1.8e+04	4.5e+04	-6.7e+04	6.6e+04	-4.6e+04	2.3e+04	-8.5e+03	2.3e+03	-4.5e+02	62	-5.7	0
model_pow_14	0.79	2.4e+04	-1.4e+05	3.8e+05	-6.1e+05	6.6e+05	-5e+05	2.8e+05	-1.2e+05	3.7e+04	-8.5e+03	1.5e+03	-1.8e+02	1
model_pow_15	0.7	-3.6e+04	2.4e+05	-7.5e+05	1.4e+06	-1.7e+06	1.5e+06	-1e+06	5e+05	-1.9e+05	5.4e+04	-1.2e+04	1.9e+03	-

https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-ridge-lasso-regression-python/

Summary of shortcomings of un-regularized regression

As model complexity increases…

(1) magnitude of model coefficients tends to *increase*

(2) stability of model coefficients (particularly those corresponding to correlated features) tends to *decrease* for even small perturbations in sample data

(3) sampling density *decreases* exponentially for a given sample size N.

(4) generalizability to unseen data *decreases*.

Linear Regression (review)

We model the world as:

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

We estimate the model parameters by minimizing:

$$\sum_{i=1}^{N} (y_i - \hat{\beta}_0 - \sum_{j=1}^{p} x_{ij} \hat{\beta}_j)^2$$

Linear Regression with (L2) Regularization

We model the world as:

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

(same as before)

We estimate the model parameters by minimizing:

(the "regularization" parameter)

$$\sum_{i=1}^{N}(y_i - \hat{\beta}_0 - \sum_{j=1}^{p} x_{ij}\hat{\beta}_j)^2 + \lambda \underline{\sum_{i=1}^{p} \hat{\beta}_i^2}$$

(new term - shrinkage)

Ridge Regression

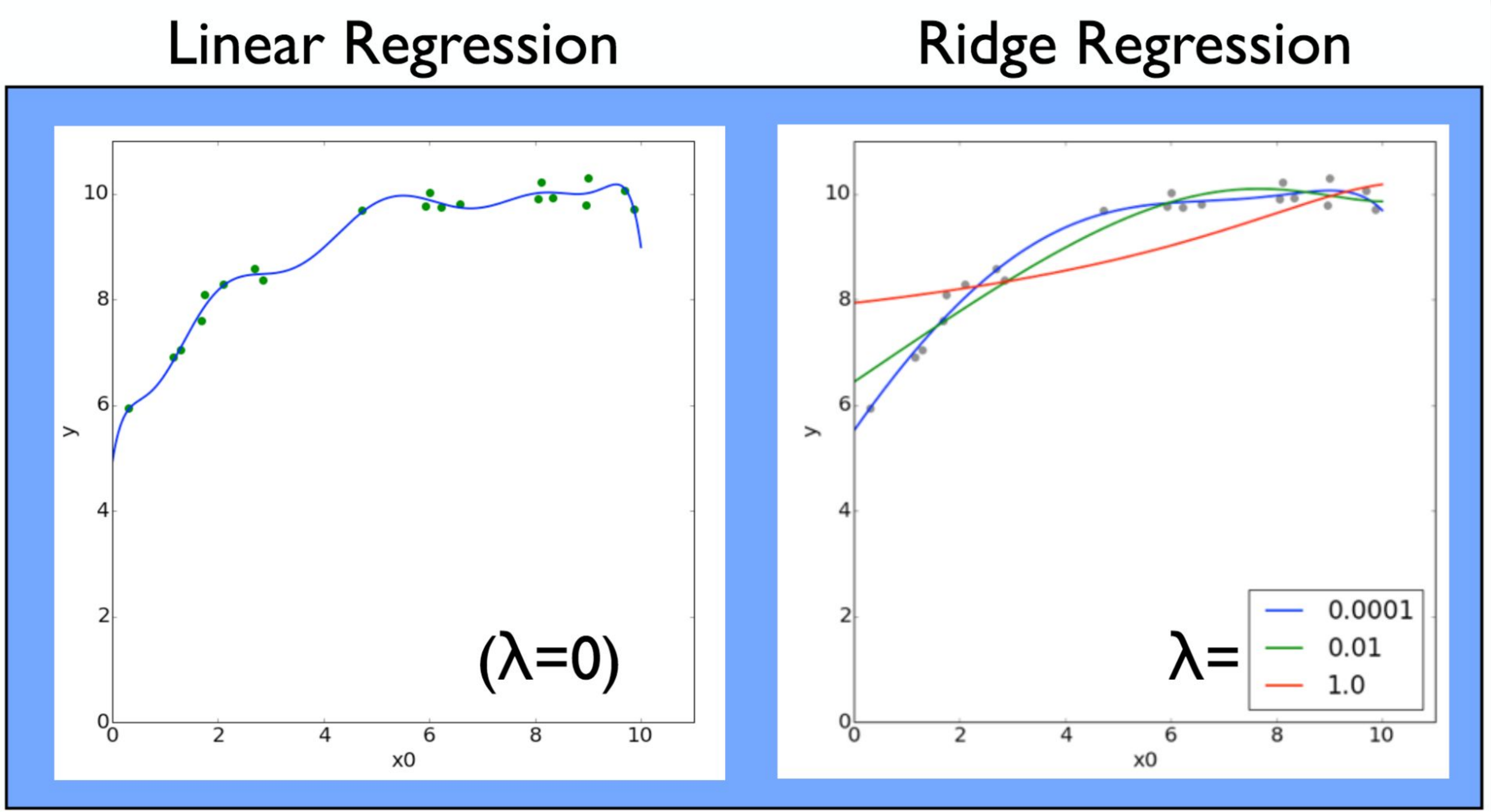
$$\sum_{i=1}^{N}(y_i - \hat{\beta}_0 - \sum_{j=1}^{p} x_{ij}\hat{\beta}_j)^2 + \lambda\sum_{i=1}^{p}\hat{\beta}_i^2$$

Notice, we do not penalize B_0.

Changing the hyperparameter lambda changes the amount that large coefficients are penalized.

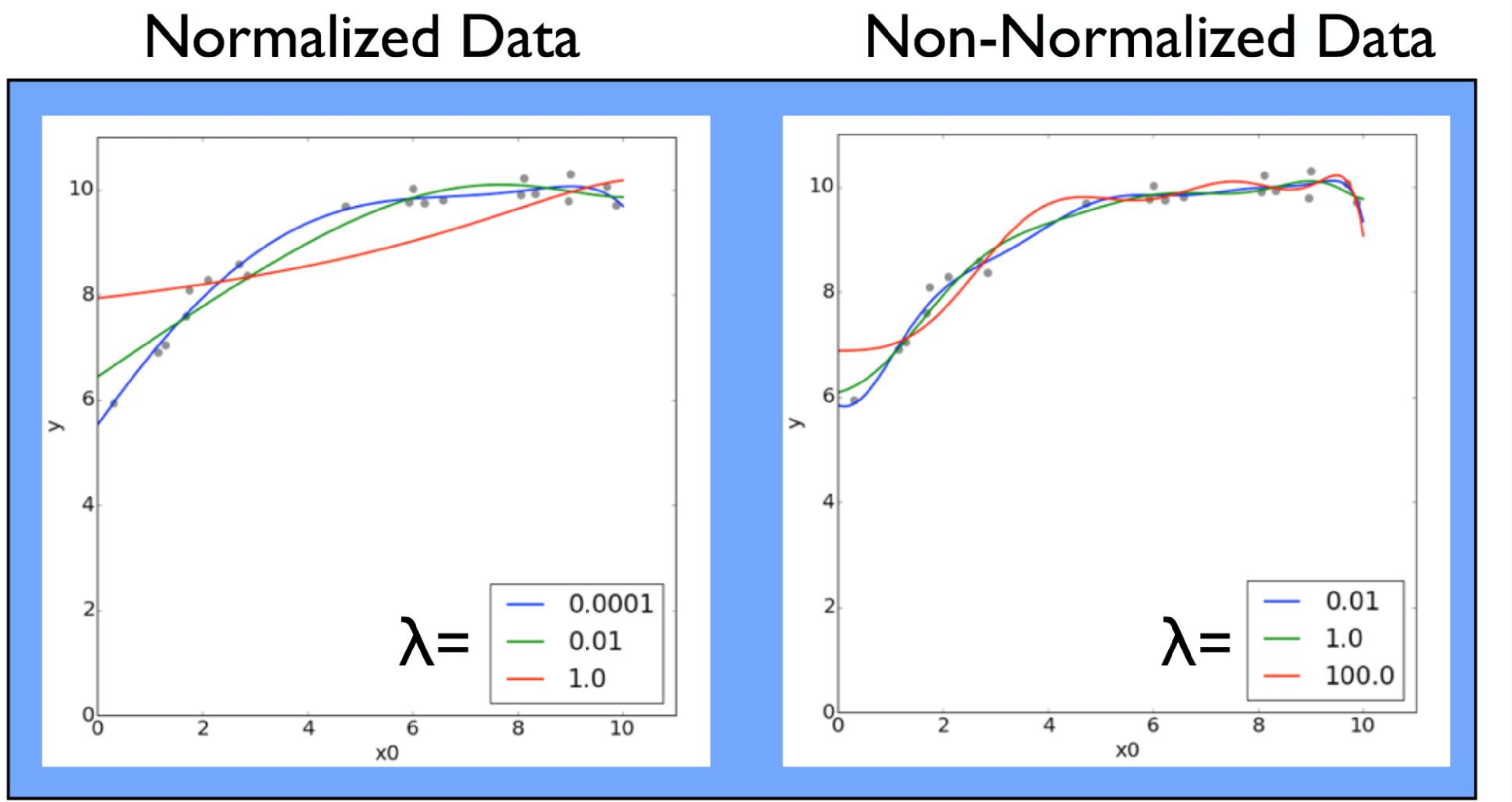
Increasing lambda increases the model's bias and decreases its variance. ← this is cool!

Ridge Regression



Ridge Regression - Importance of standardized data

$$X_s = \frac{X - \bar{X}}{Std(X)}$$



Single value for λ assumes features are on the same scale!!

LASSO Regression
(Linear Regression w/ LASSO (L1) Regularization)

We model the world as:

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

(same as before)

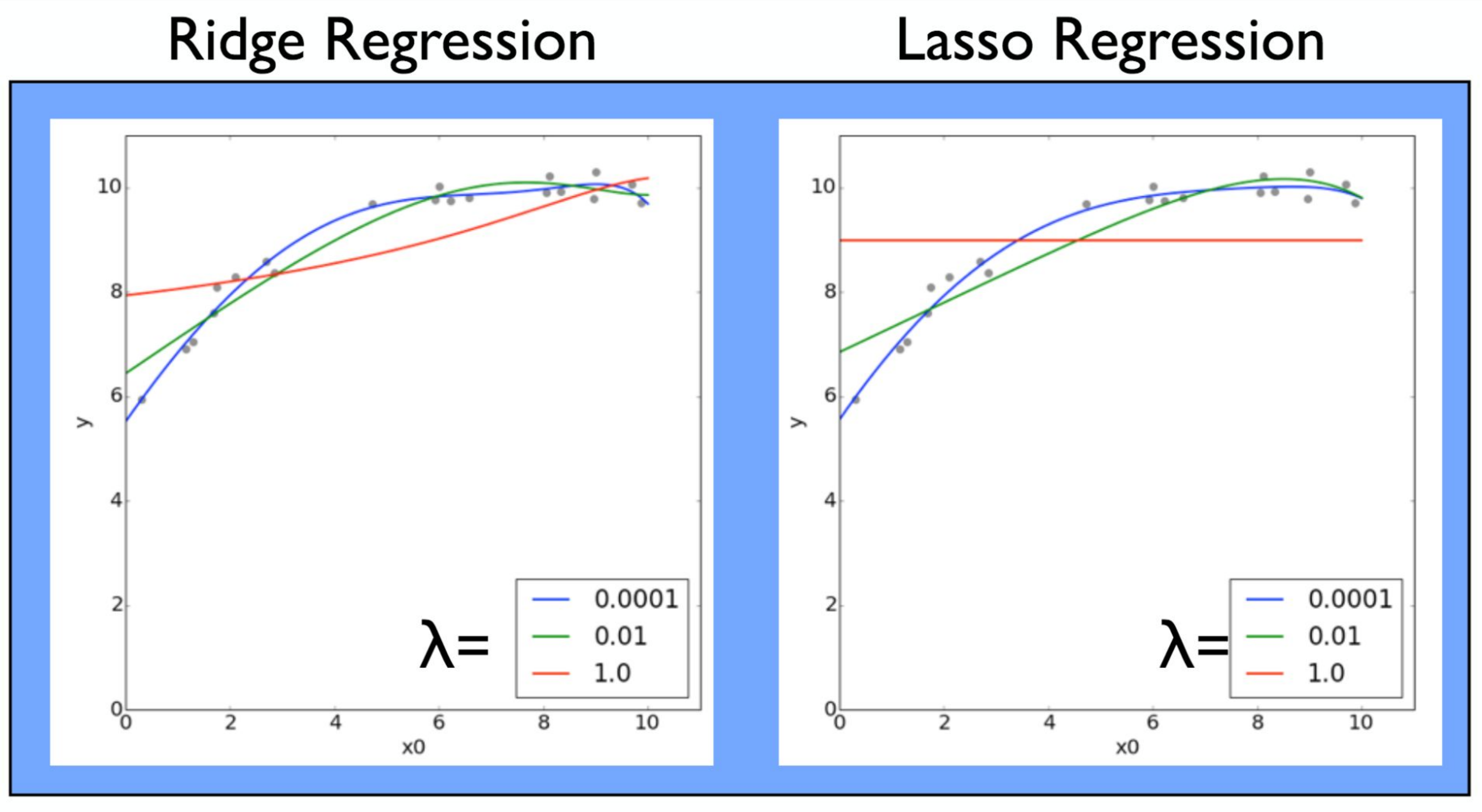
We estimate the model parameters to minimizing:

(the “regularization” parameter)

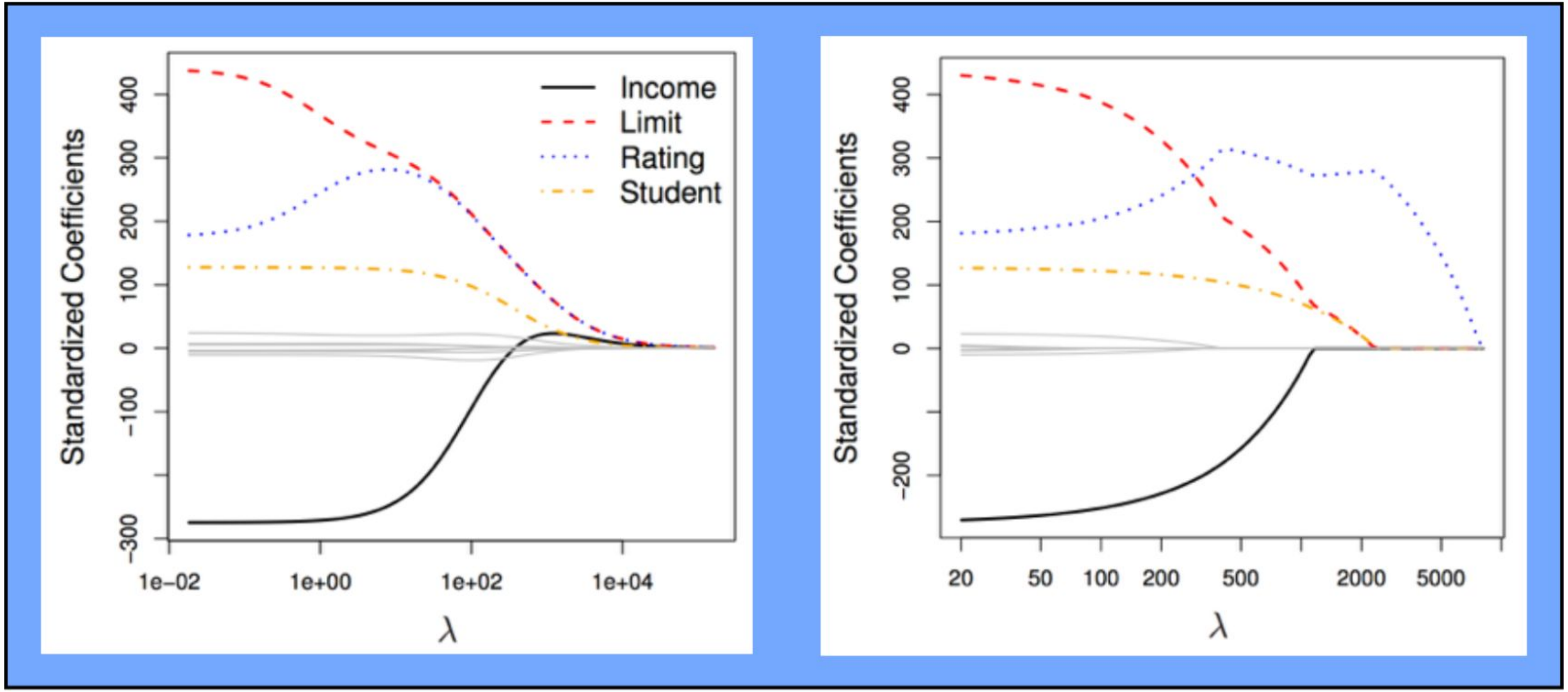
$$\sum_{i=1}^{N}(y_i - \hat{\beta}_0 - \sum_{j=1}^{p} x_{ij}\hat{\beta}_j)^2 + \lambda \sum_{i=1}^{p} |\hat{\beta}_i|$$

(absolute value instead of squared)

galvanize


Ridge Regression
Lasso Regression
y
x0
0
2
4
6
8
10
λ=
0.0001
0.01
1.0

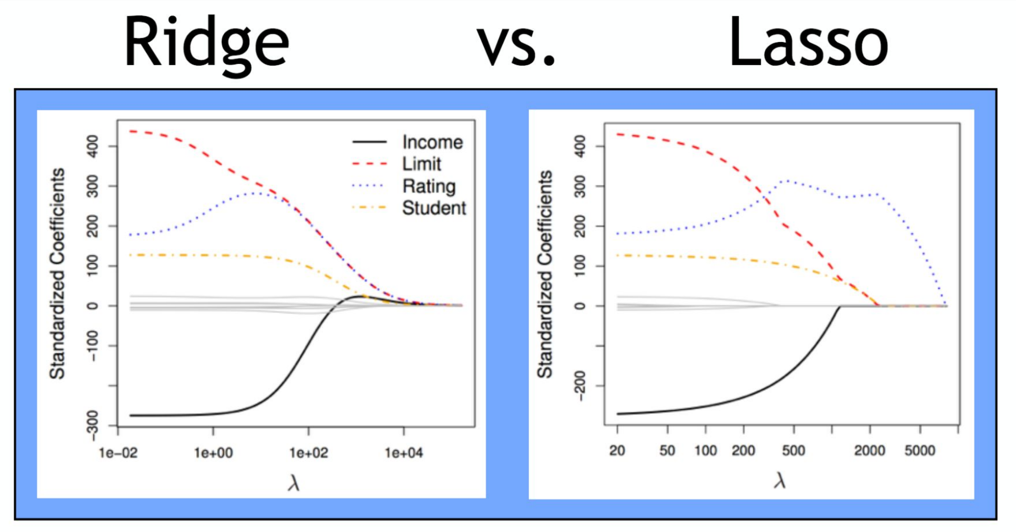
Ridge vs. Lasso



Ridge vs LASSO

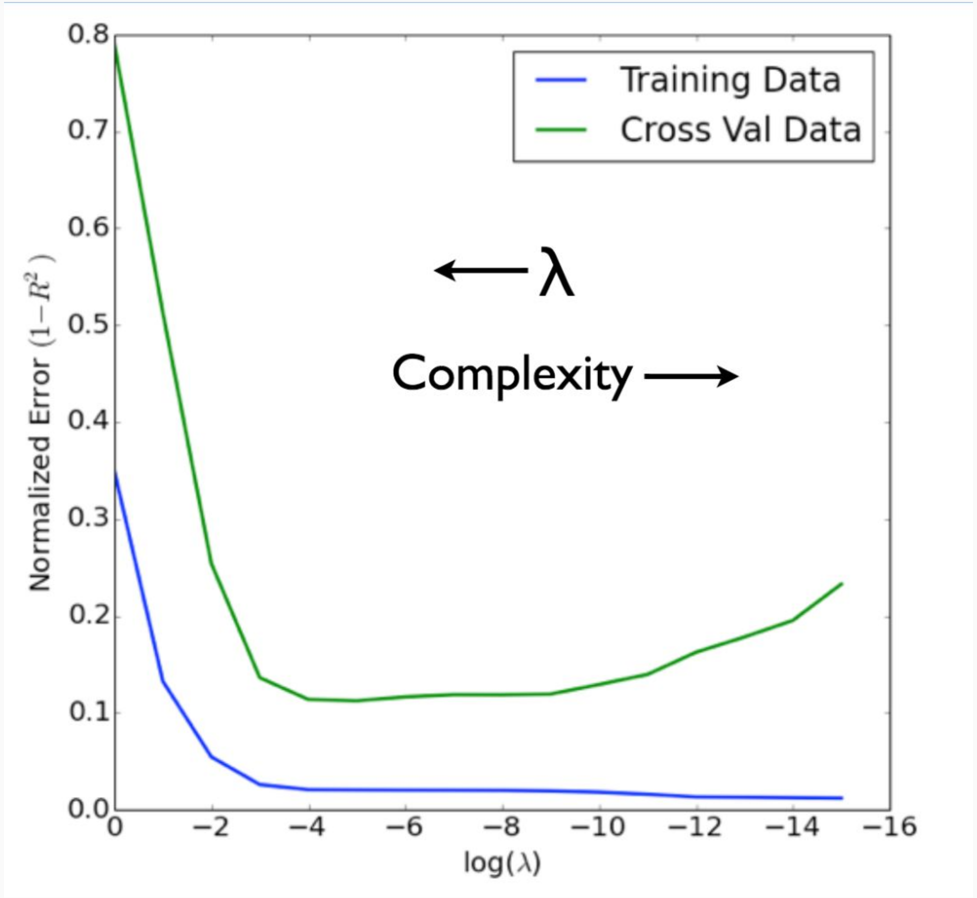
- Ridge forces parameters to be small + Ridge is computationally easier (faster!) because it's differentiable
- Lasso tends to set coefficients exactly equal to zero
  - This is useful as a sort-of "automatic feature selection" mechanism,
  - leads to "sparse" models, and
  - serves a similar purpose to stepwise features selection

Which is better depends on your dataset!



How might we select (tune) lambda??

Chose lambda via Cross-Validation!



scikit-learn

Classes:

- sklearn.linear_model.**LinearRegression**(...)
- sklearn.linear_model.**Ridge**(alpha=my_alpha, …)
- sklearn.linear_model.**Lasso**(alpha=my_alpha, …)
- sklearn.linear_model.**ElasticNet**(alpha=my_alpha, l1_ratio = !!!!, …) Wow!

(In sklearn alpha = lambda)

All have these methods:

- fit(X, y)
- predict(X)
- score(X, y)

Remember:

1) Use regularization!
    a) Helps prevent overfitting
    b) Helps with collinearity
    c) Gives you a knob to adjust bias/variance trade-off
2) Don’t forget to standardize your data!
    a) Column-by-column, de-mean and divide by the standard deviation
3) Lambdas control the size (L1 & L2) and existence (L1) of feature coefficients.
    a) Large lambdas mean more regularization (fewer/smaller coefficients) and *less* model complexity.
4) You can have it all! (ElasticNet)