Regularized Linear Regression

Ryan Henning's slides
customized by F. Burkholder

galvanize

- Shortcomings of Ordinary Linear Regression
- Curse of Dimensionality
- Ridge Regression
- Lasso Regression
- When to use each!

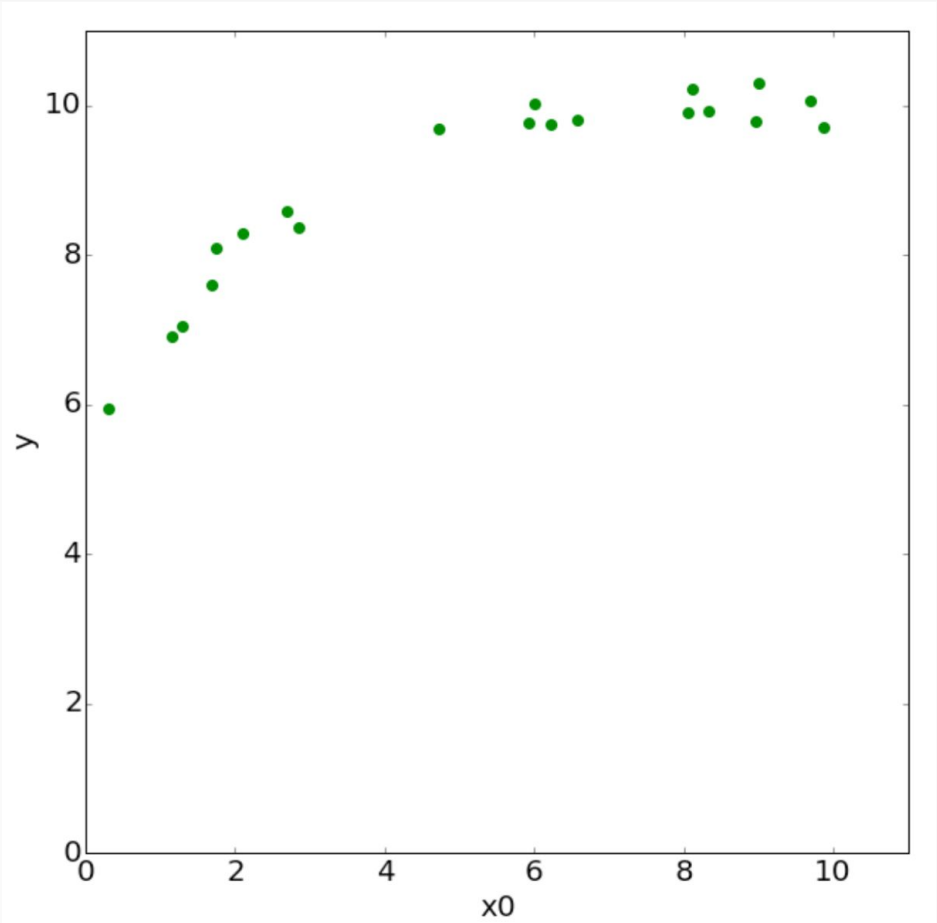
Set the scene for regularization: Linear Regression Example

Data: 20 rows x 10 features

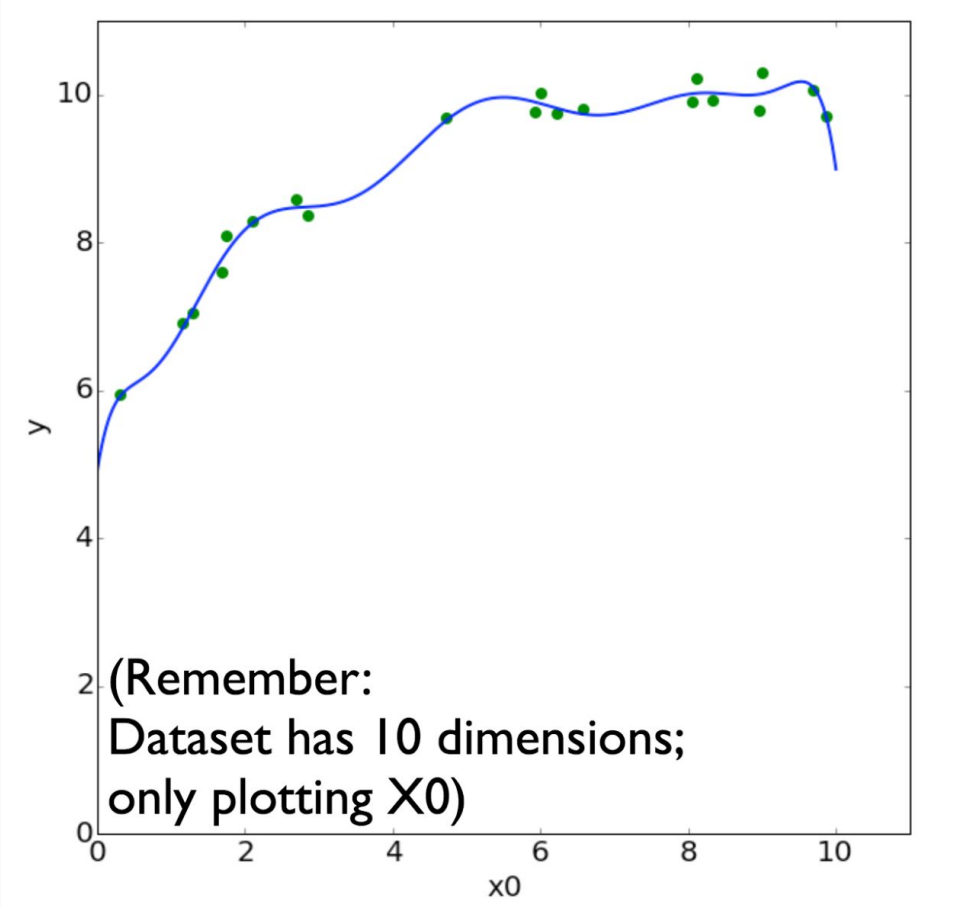
Predict: *y*

y	x0	x1	x2	x3	...
9.92	8.33	69.39	578.0	4815.	...
8.58	2.69	7.26	19.54	52.64	...
8.07	1.75	3.06	5.35	9.36	...
8.29	2.11	4.46	9.41	19.86	...
...	...	...	...	...	...

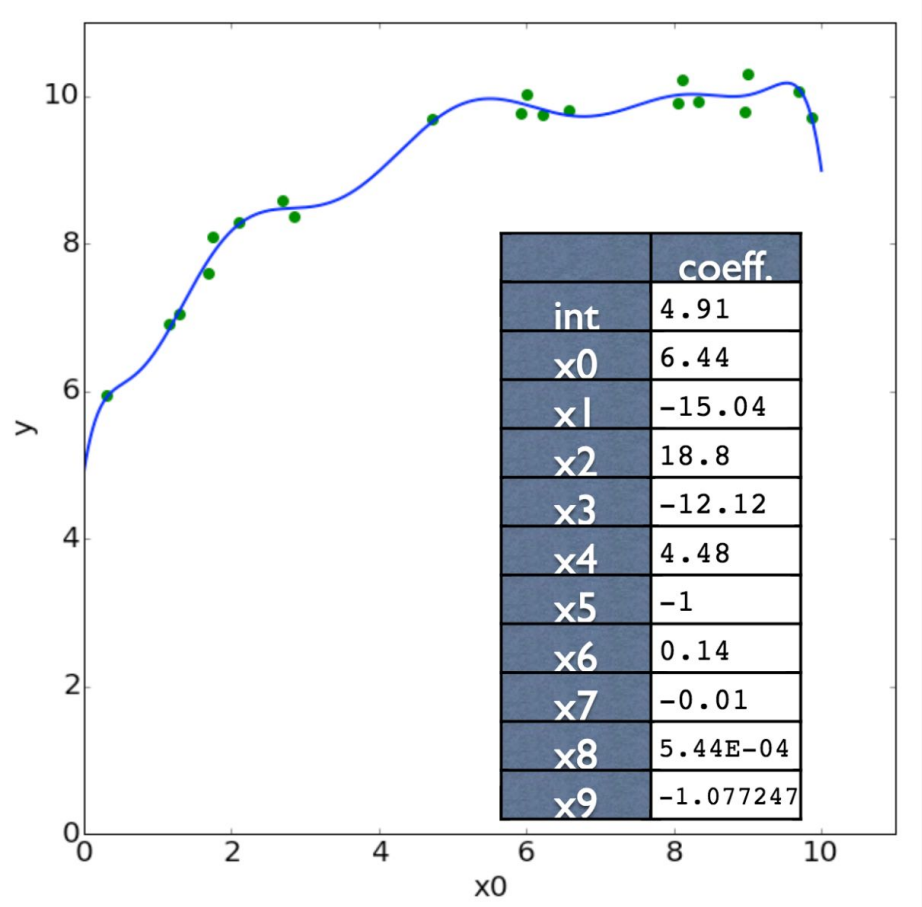
Linear Regression Example (x0 vs y)



Linear Regression Example (x0 vs y, model over all features)

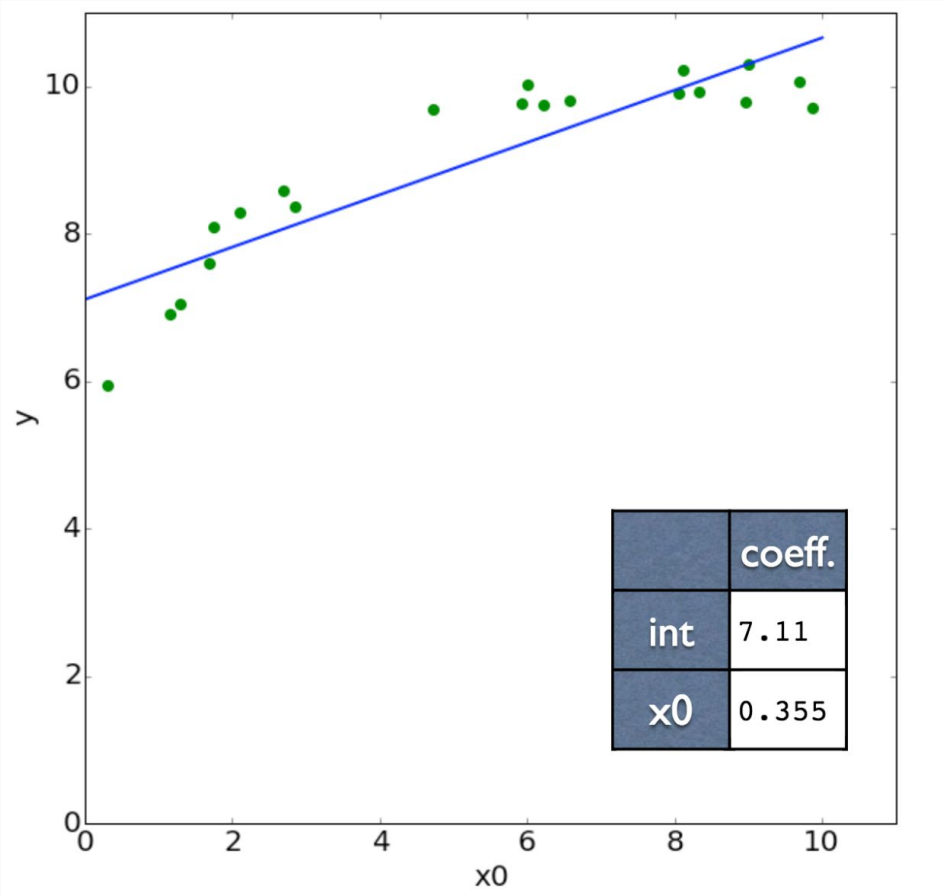


Linear Regression Example (x0 vs y, model over all features)



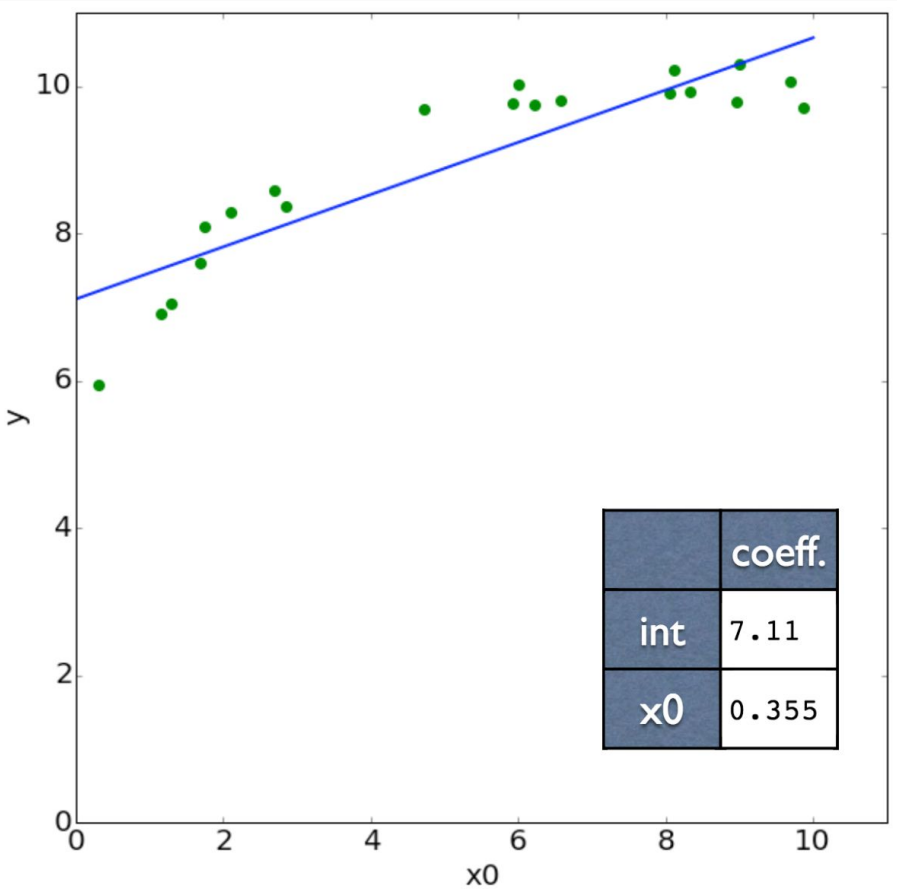
	coeff.
int	4.91
x0	6.44
x1	-15.04
x2	18.8
x3	-12.12
x4	4.48
x5	-1
x6	0.14
x7	-0.01
x8	5.44E-04
x9	-1.077247

Linear Regression Example (x0 vs y, model over only x0 features)

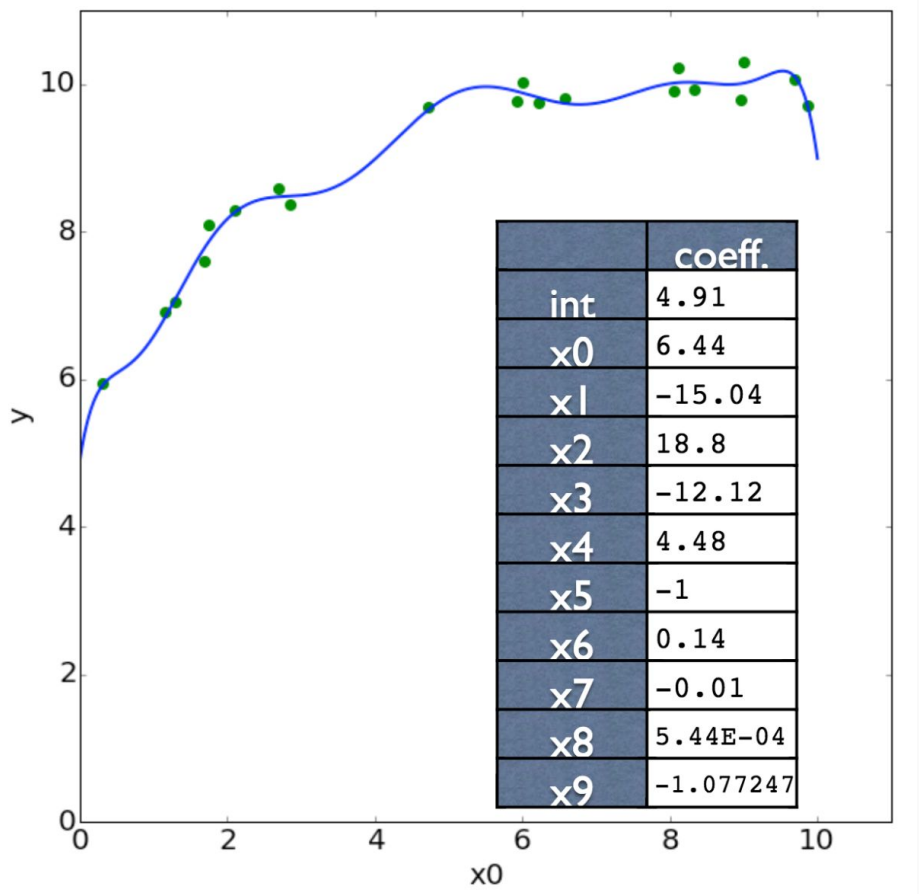


More dimensions makes our model more complex.

galvanize



	coeff.
int	7.11
x0	0.355



	coeff.
int	4.91
x0	6.44
x1	-15.04
x2	18.8
x3	-12.12
x4	4.48
x5	-1
x6	0.14
x7	-0.01
x8	5.44E-04
x9	-1.077247

And what's worse, in high dimensions, data is (usually) sparse

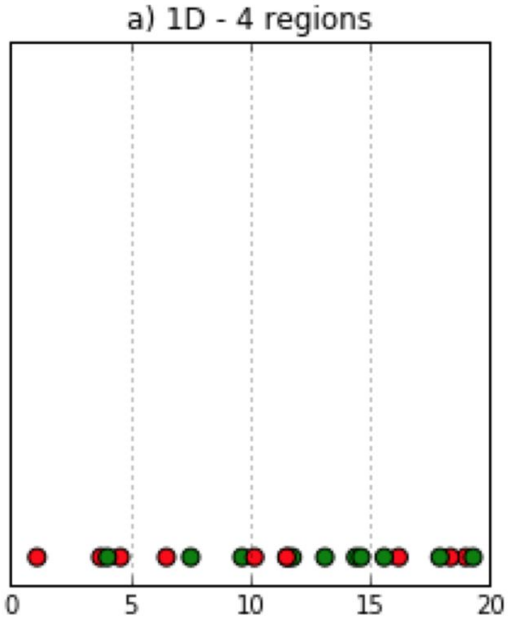
Curse of Dimensionality

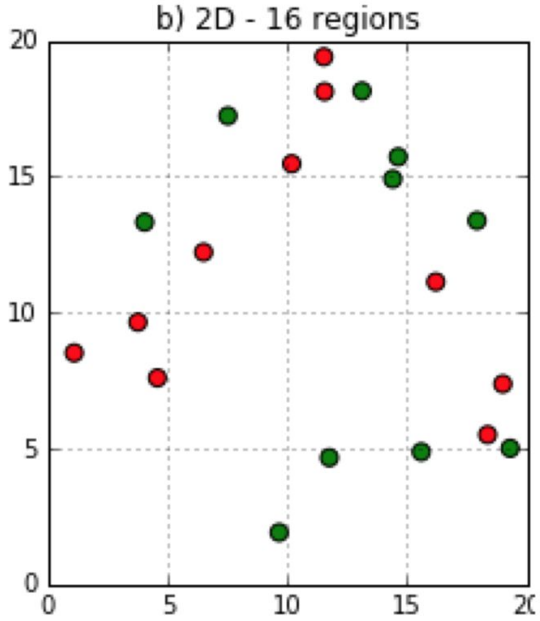
As the number of dimensions increase, the volume that data can occupy grows exponentially.

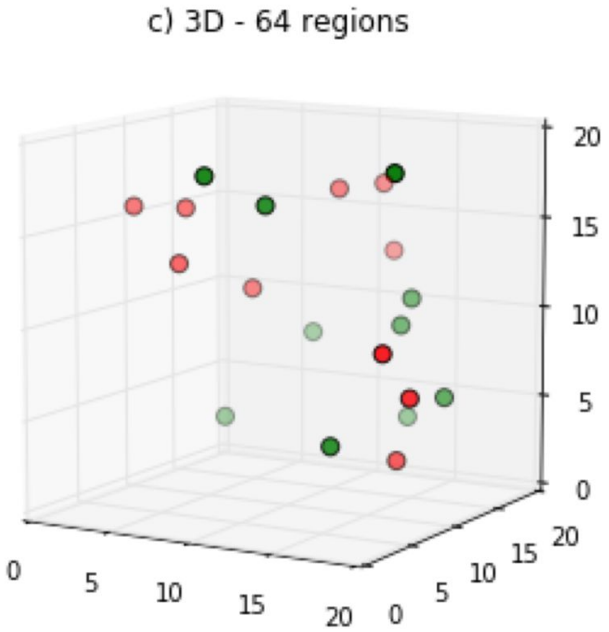
D = number of dimensions
N = number of datapoints

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

Curse of Dimensionality example







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

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

galvanize

In high dimensions, data is (usually) sparse

Again… the **Curse of Dimensionality** bites us.

Linear regression can have high variance (i.e. tends to overfit) on high dimensional (many featured) data…

We'd like to restrict ("normalize", or "regularize") the model so that it has less variance.

Take the 20 example (rows) x 10 feature (columns) dataset as an example… when we fit over all features, the complexity of the model grew dramatically.

(and keep in mind, some datasets have thousands of features)

Linear Regression (another review)

We model the world as:

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

We estimate the model parameters by minimizing:

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

Enter Regularization!
Ridge Regression
(Linear Regression with (L2) Regularization)

galvanize

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)

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

(new term!)

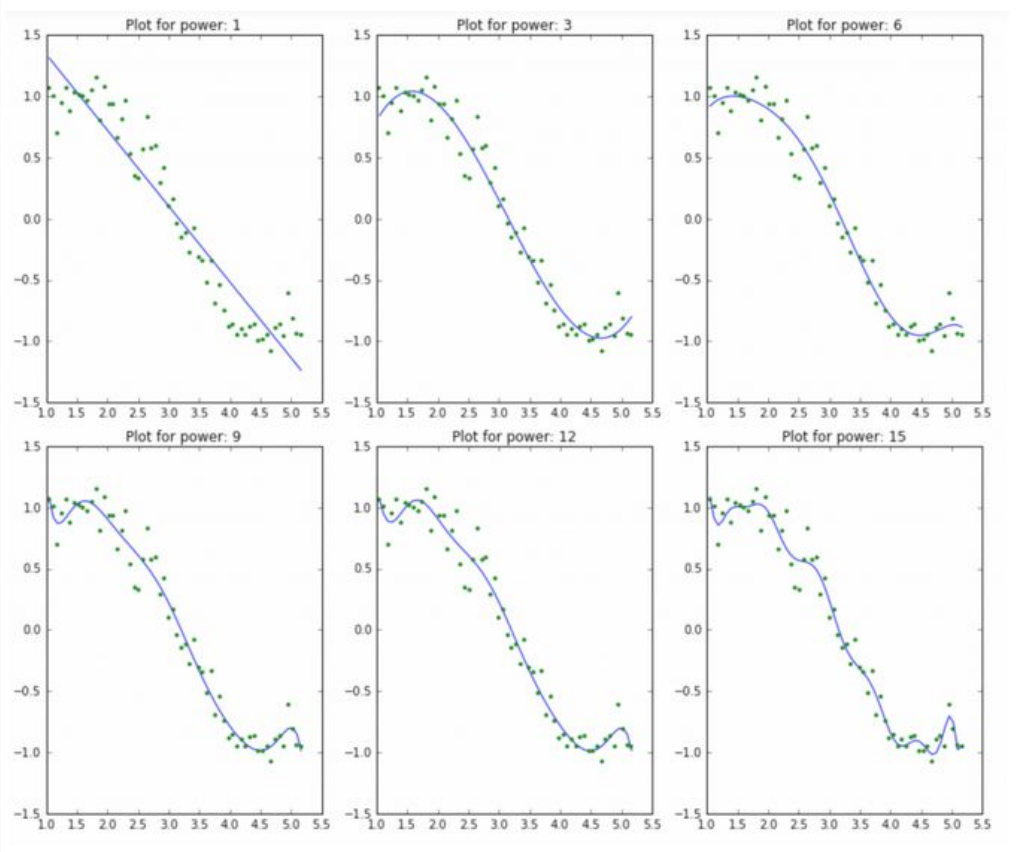
Ridge Regression (L2)

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

What does the new term accomplish?

Why would limiting the magnitude of the coefficients prevent overfitting?

Magnitude of coefficients as model complexity increases
Part 1: the fits (as complexity increases)



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

Magnitude of coefficients as model complexity increases, Part 2: the coefficients

galvanize

	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	-4
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/

Take-away

As model complexity (e.g. order, # features) increases, the magnitude of the coefficients tends to increase AND there is more of a tendency to overfit.

So, to limit overfitting limit the magnitude of the coefficients. That's the idea behind regularization (and why it works).

galvanize

Ridge Regression

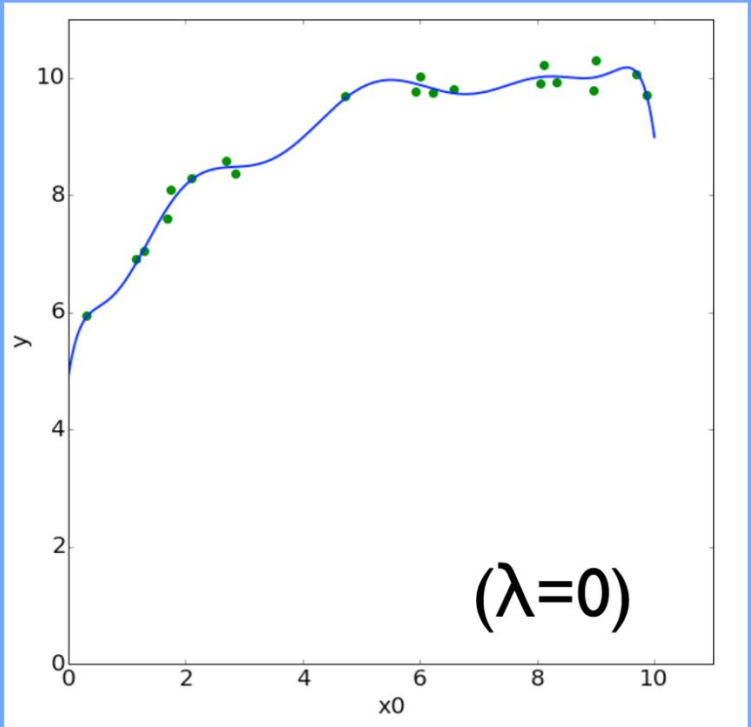
$$\sum_{i=1}^{N}(y_i - \hat{\beta}_0 - \sum_{j=1}^{p} x_{ij}\hat{\beta}_j)^2 + \lambda \sum_{i=1}^{p} \hat{\beta}_i^2$$

Notice, we do not penalize B_0.

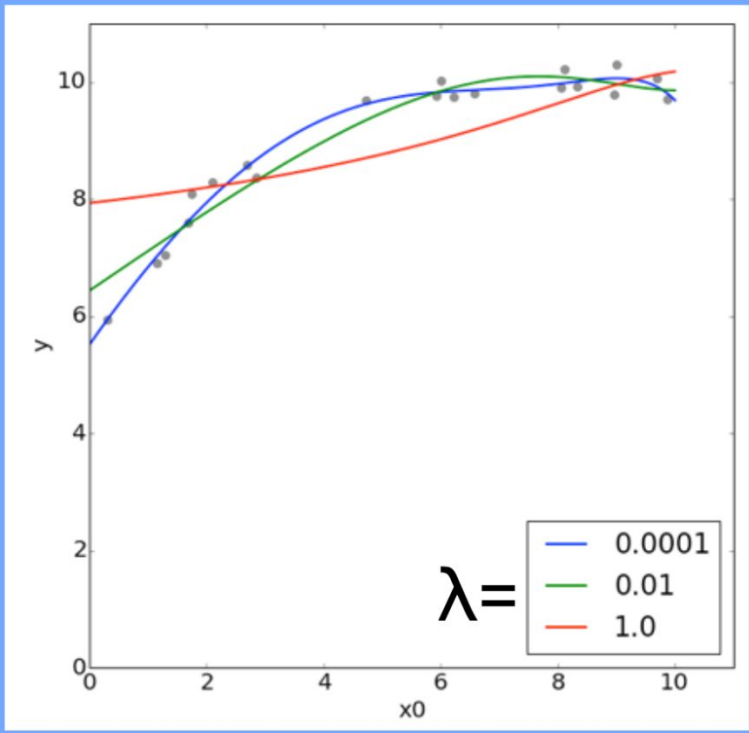
Changing lambda changes the amount that large coefficients are penalized.

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

Linear Regression

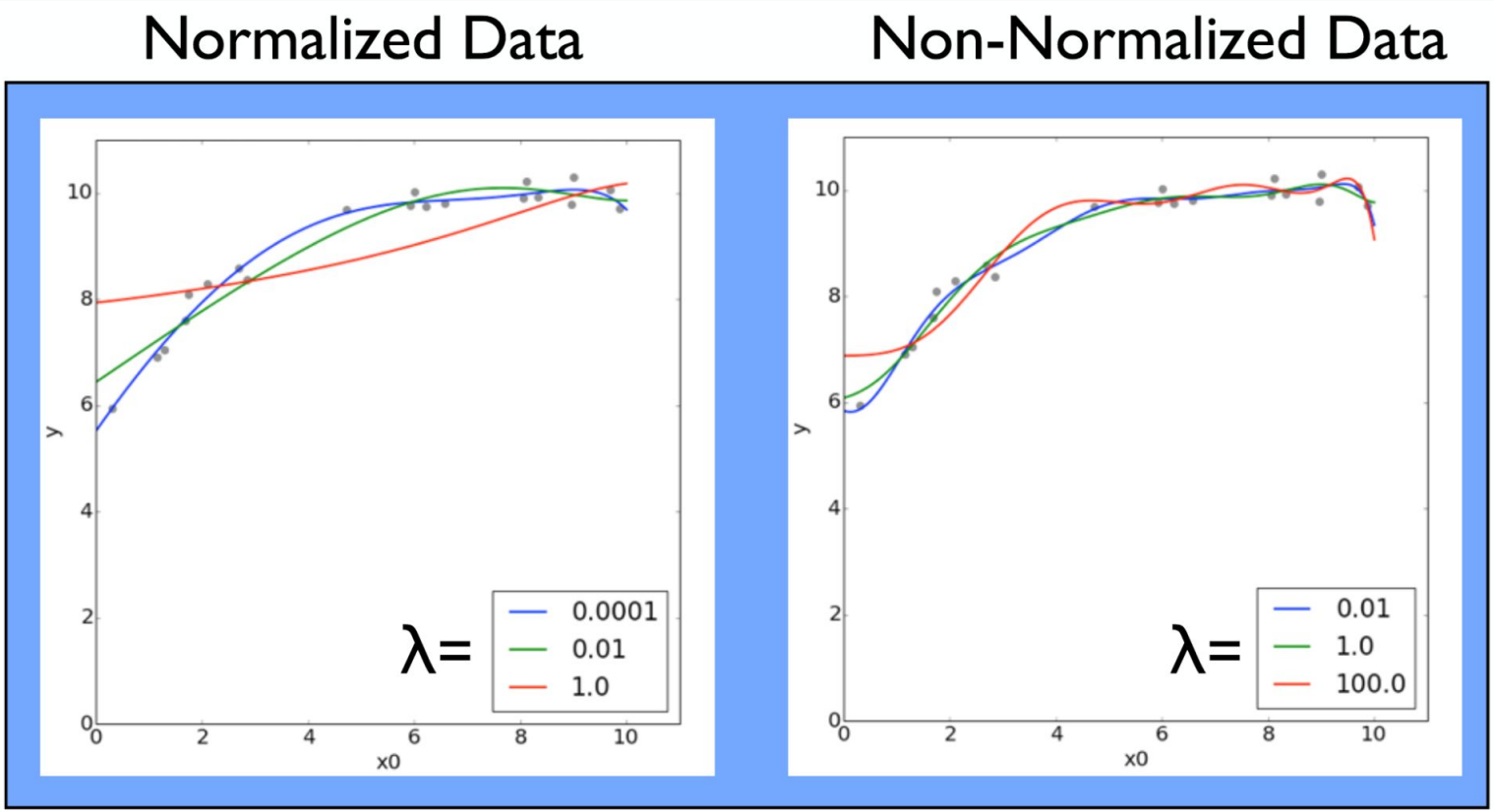


Ridge Regression



galvanize

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



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

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

We model the world as:

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

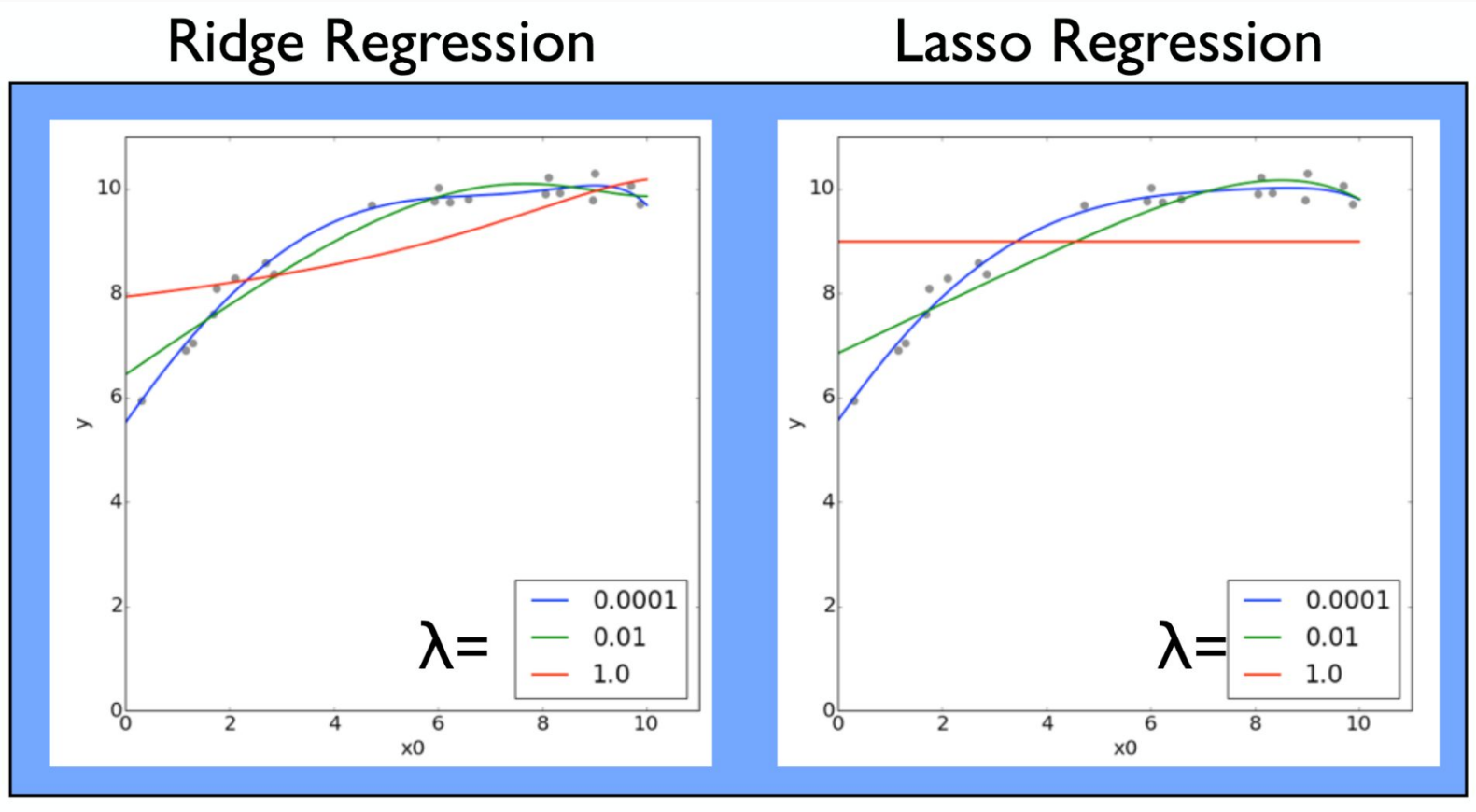
(same as before)

We estimate the model parameters to minimizing:

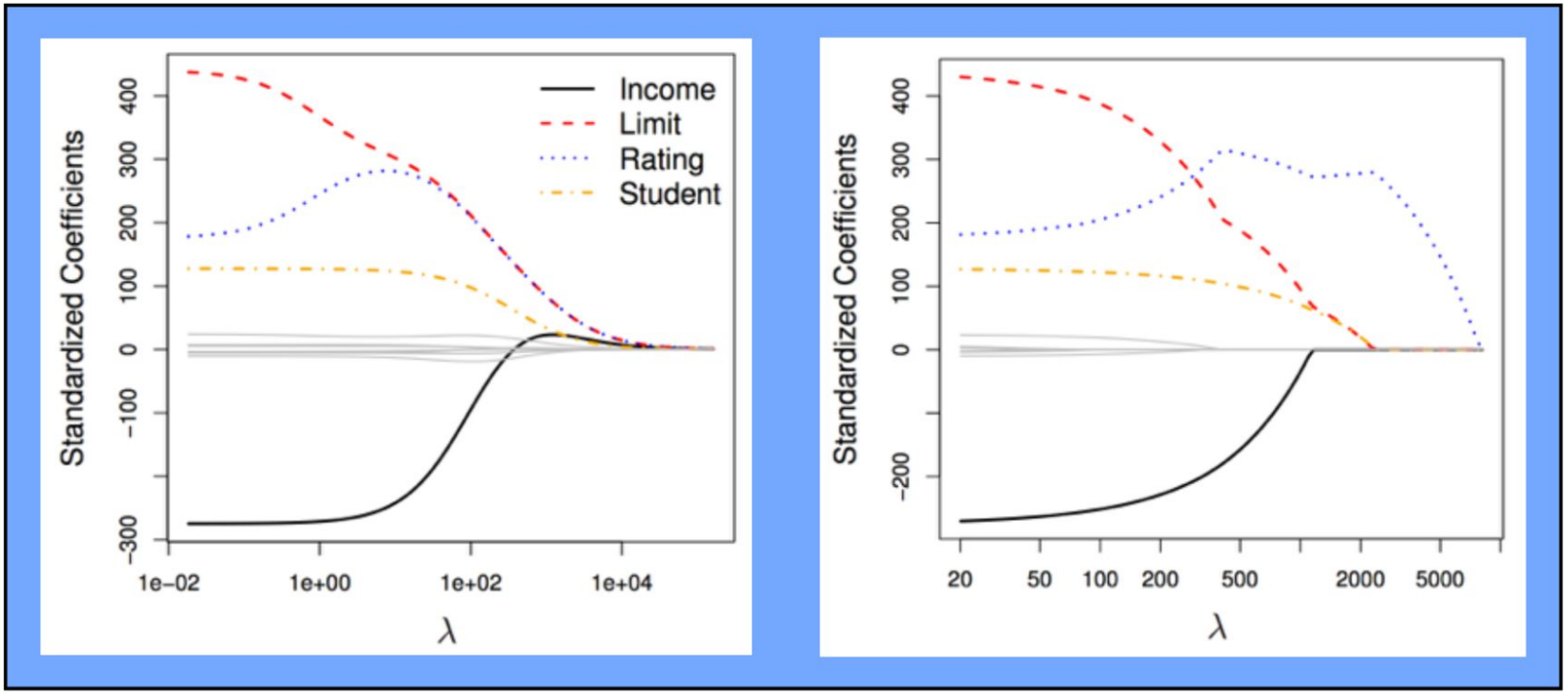
$$\sum_{i=1}^{N}(y_i - \hat{\beta}_0 - \sum_{j=1}^{p} x_{ij}\hat{\beta}_j)^2 + \lambda \sum_{i=1}^{p} |\hat{\beta}_i|$$

(the “regularization” parameter)

(absolute value instead of squared)


Ridge Regression
Lasso Regression
y
x0
λ=
0.0001
0.01
1.0

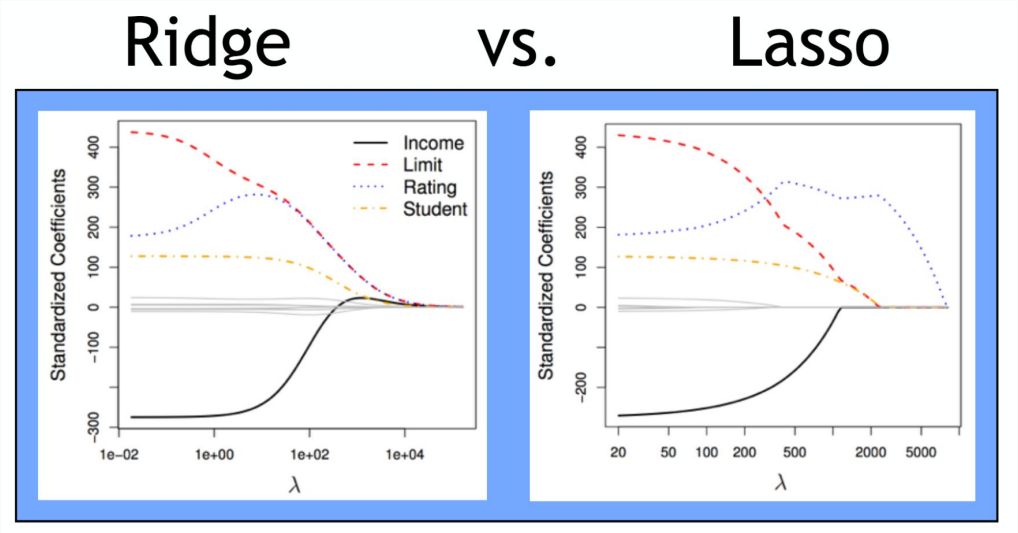
Ridge vs. Lasso



Ridge vs LASSO

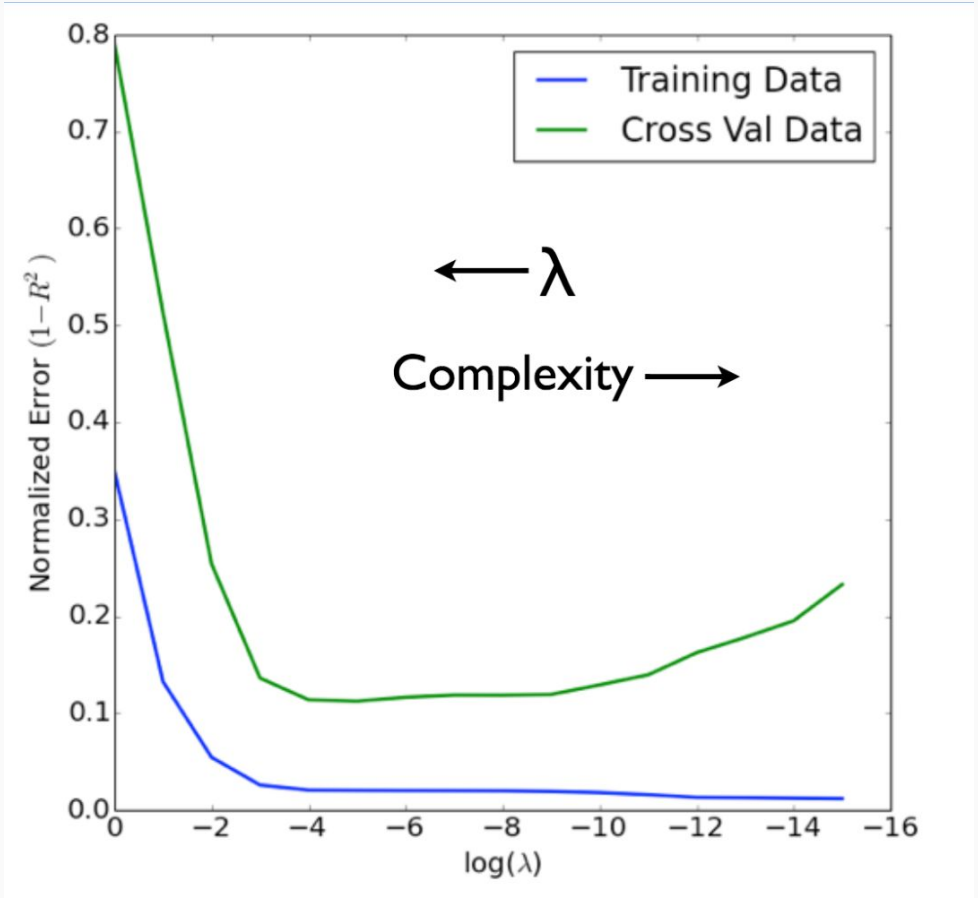
- Ridge forces parameters to be small + Ridge is computationally easier because it is differentiable
- Lasso tends to set coefficients exactly equal to zero
  - This is useful as a sort-of “automatic feature selection” mechanism,
  - leads to “sparse” models, and
  - serves a similar purpose to stepwise 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:

galvanize

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)