Regularized Linear Regression

Ryan Henning

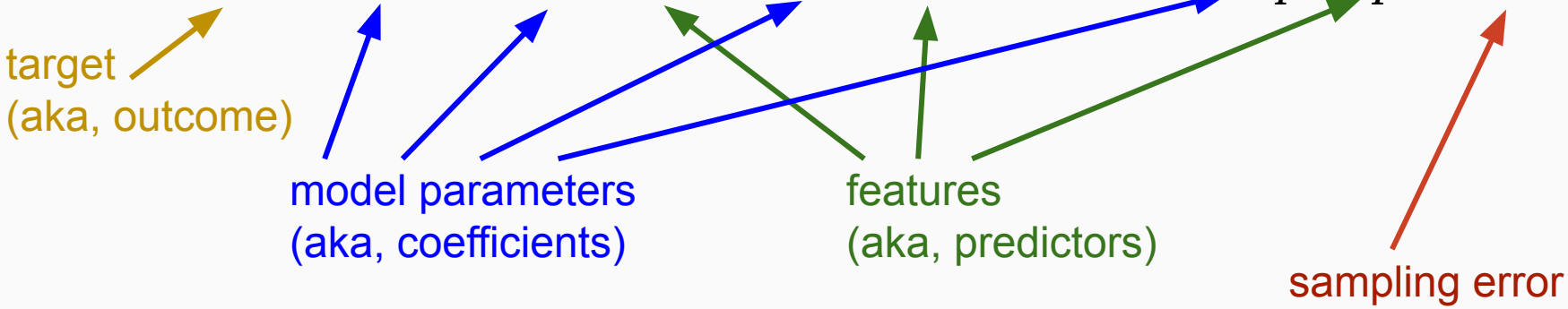
galvanize

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

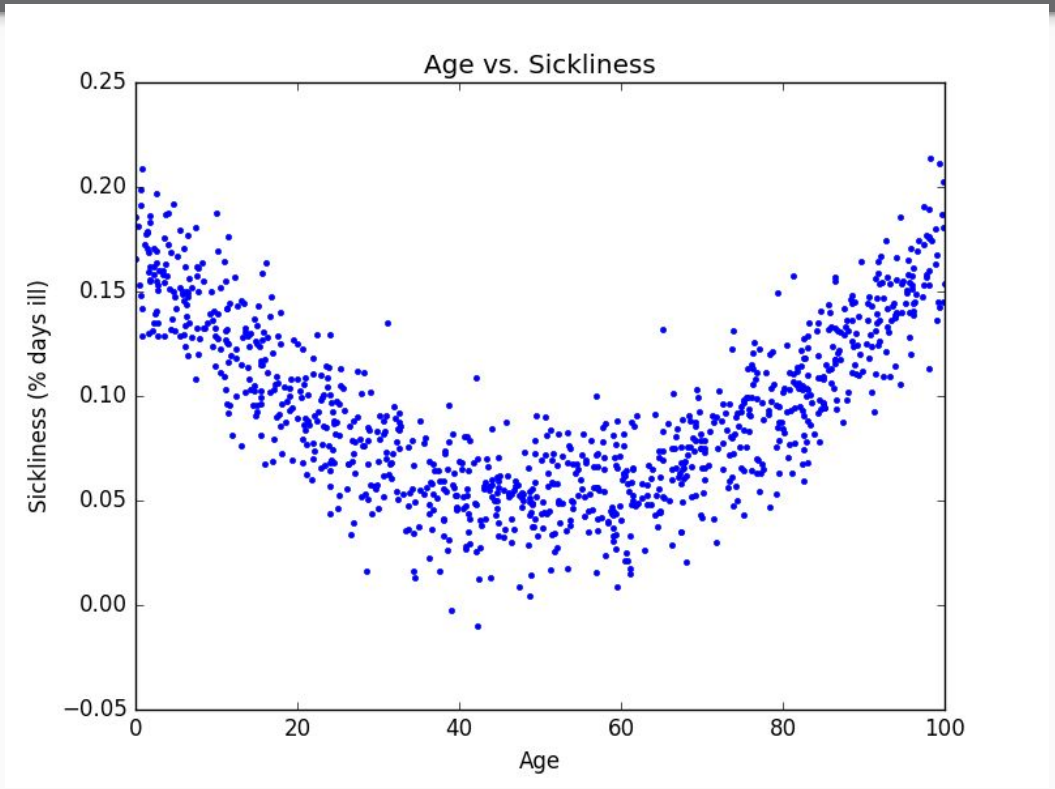
Review: Linear Regression

We assume the world is built on linear relationships. Under that assumption, we can model the relationship between *features* and a *target* like this:

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



We can make linear regression non-linear by inserting extra "interaction" features or higher-order features.

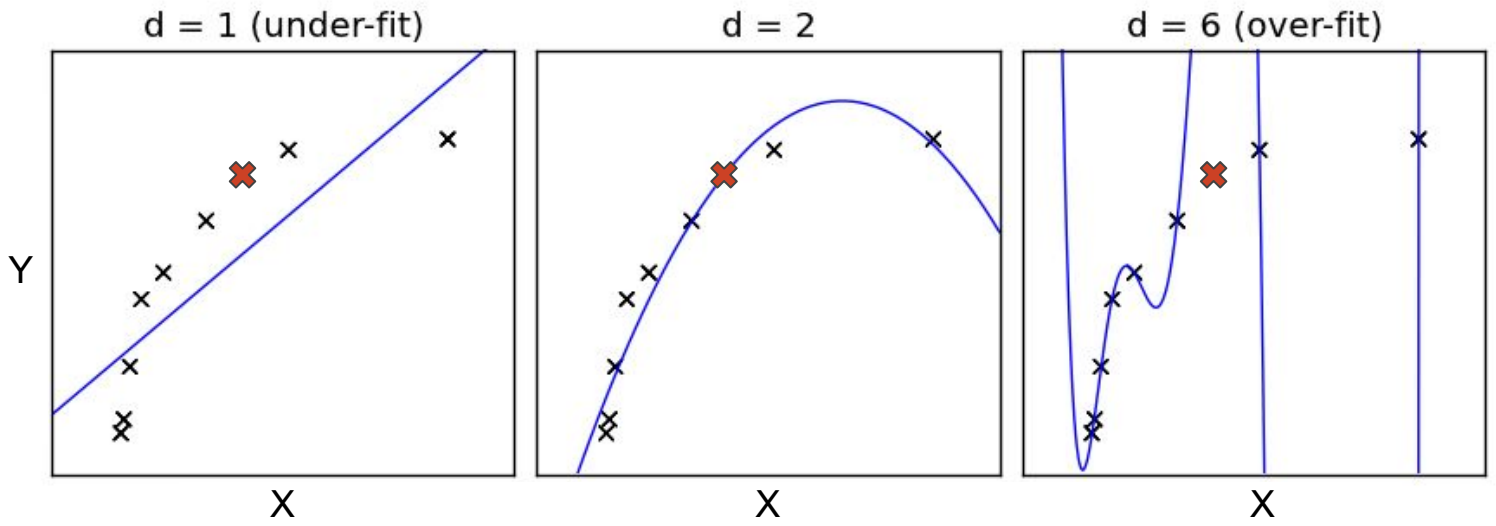


Example:

$$Y = \beta_0 + \beta_1 * \text{age}$$

$$Y = \beta_0 + \beta_1 * \text{age} + \beta_2 * \text{age}^2$$

Oh the woes of overfitting...



What's bad about the first model?

What's bad about the second model?

What's bad about the third model?

Underfitting and Overfitting

Underfitting: The model doesn't fully capture the relationship between predictors and the target. The model has *not* learned the data's signal.

→ What should we do if our model underfits the data? (assume using lin. reg.)

Overfitting: The model has tried to capture the sampling error. The model has learned the data's signal *and* the noise.

→ What should we do if our model overfits the data? (harder… any guesses?)

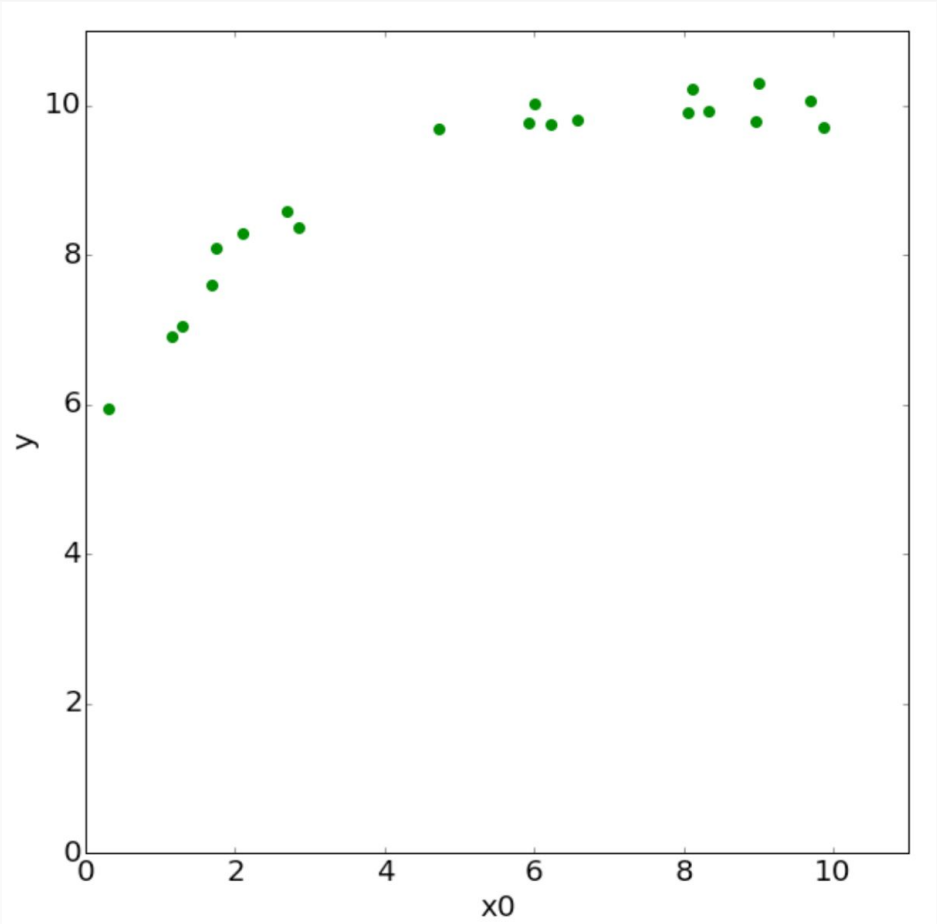
Linear Regression Example

Data: 20 examples x 10 features

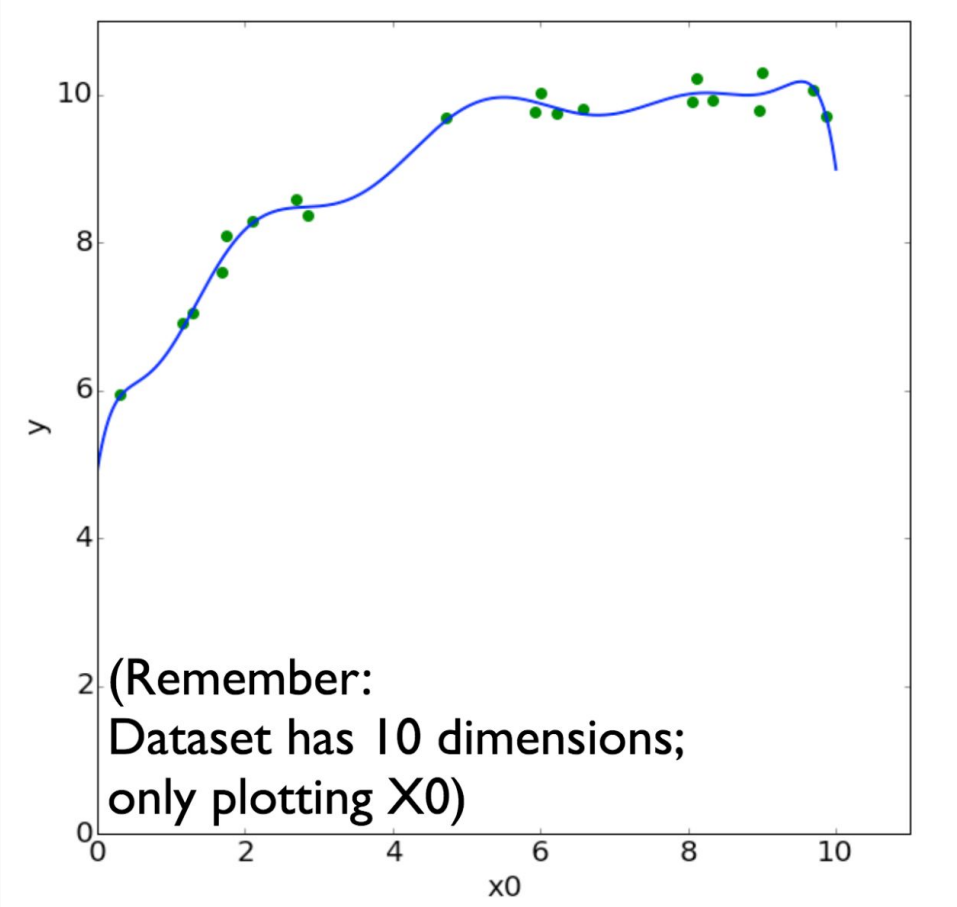
Predict: *y*

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	...
...	...	...	...	...	...

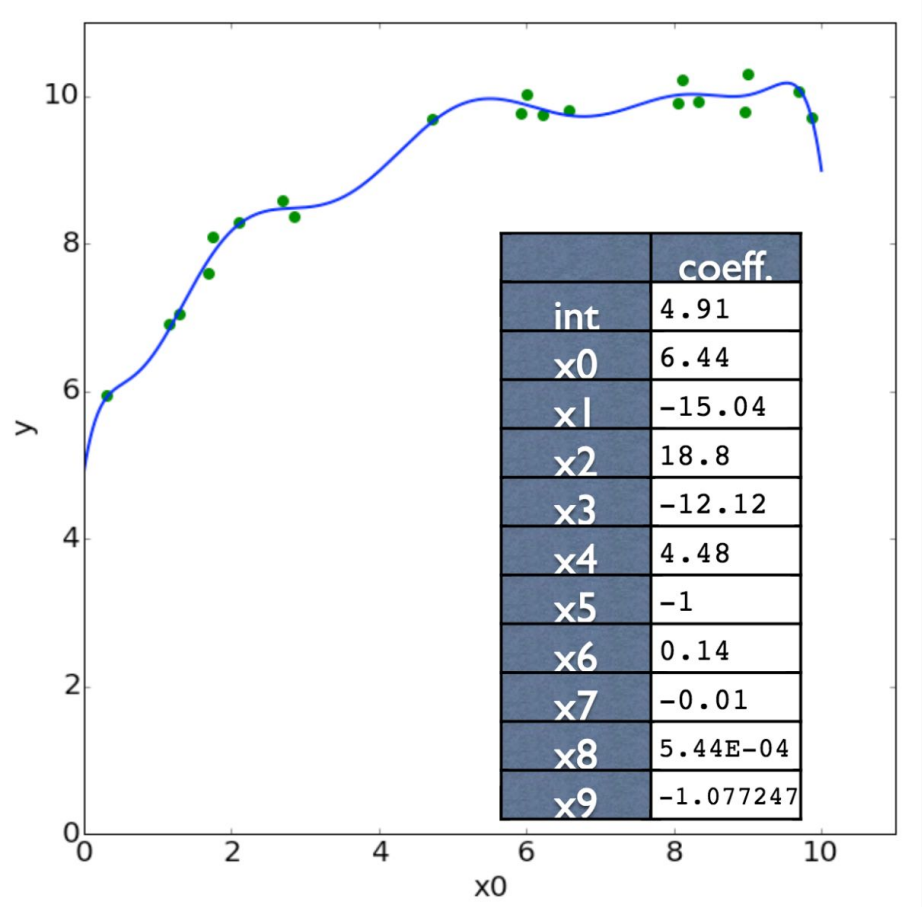
Linear Regression Example (x0 vs y)



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

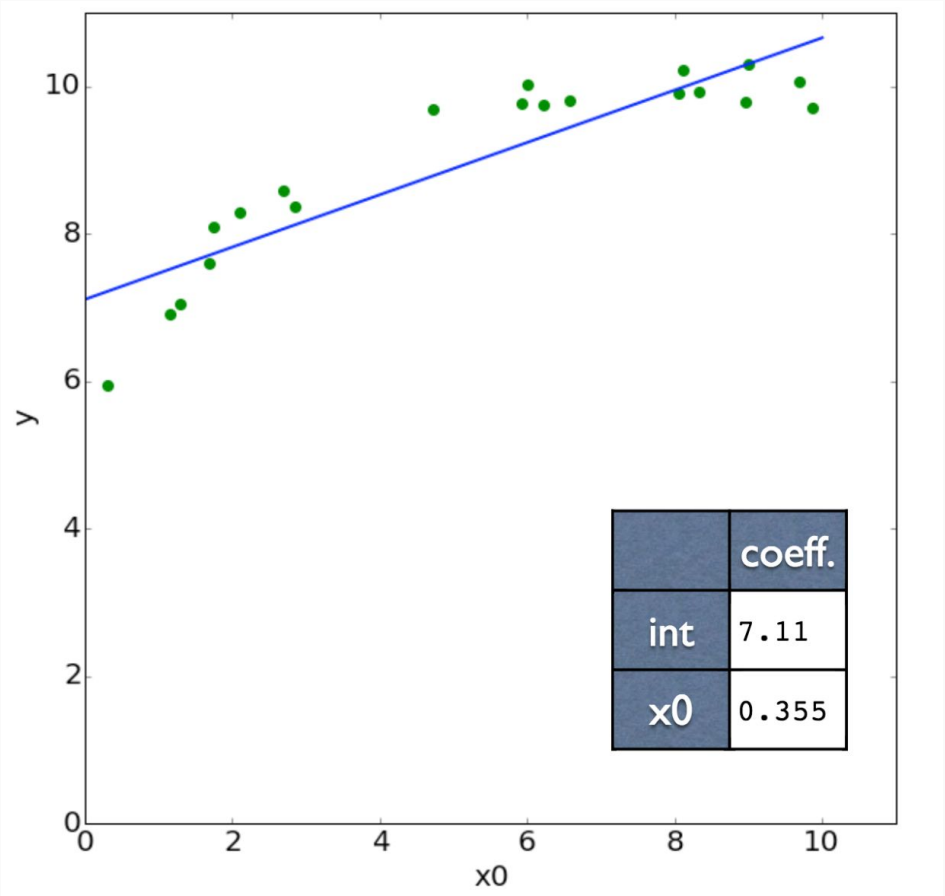


Linear Regression Example (x0 vs y, model over all features)



	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)



In high dimensions, data is (usually) sparse

Again… the **Curse of Dimensionality** bites us.
(we'll talk more about this is a later lecture)

Linear regression can have high variance (i.e. tends to overfit) on high dimensional data…
We'd like to restrict ("normalize", or "regularize") the model so that it has less variance.

Take the 20 example x 10 feature 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)

galvanize

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:

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

galvanize

Ridge Regression
(Linear Regression w/ Tikhonov (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:

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

(the “regularization” parameter)

(new term!)

Ridge Regression

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

What if we set the lambda equal to zero?

What does the new term accomplish?

What happens to a features whose corresponding coefficient value (beta) is zero?

Ridge Regression

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

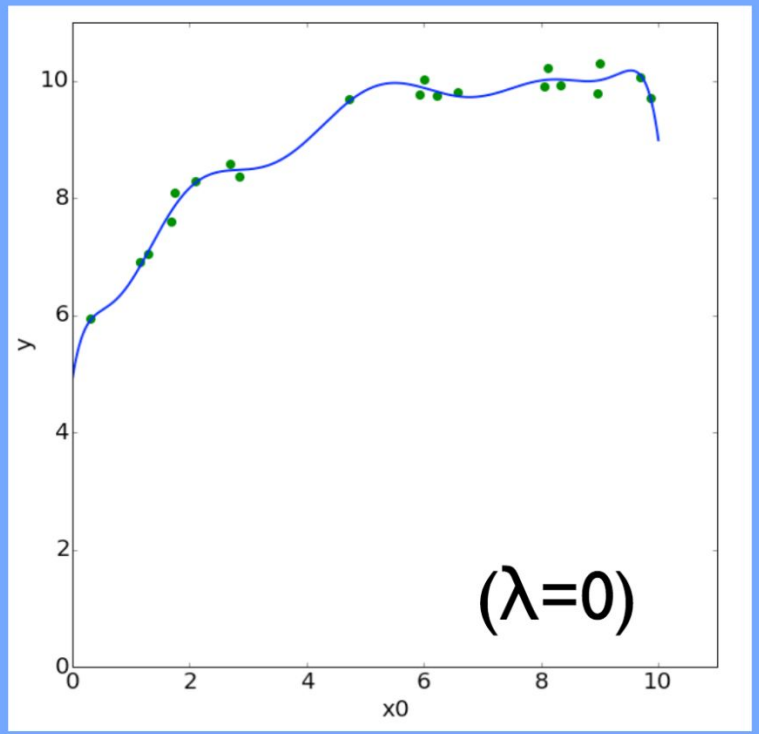
Notice, we do not penalize B_0.

Changing lambda changes the amount that large coefficients are penalized.

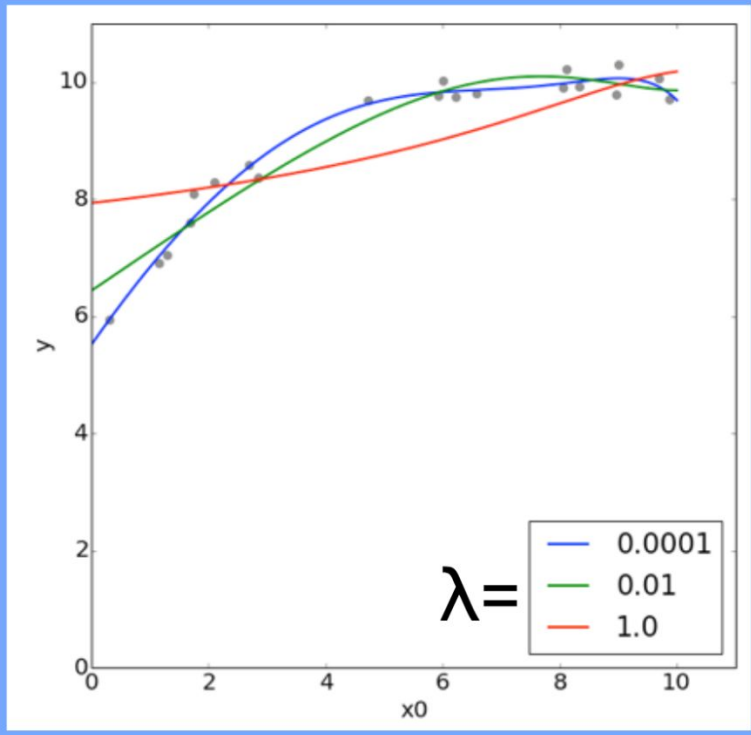
Increasing lambda increases the model's bias and decreases its variance. ← this is cool!

galvanize

Linear Regression

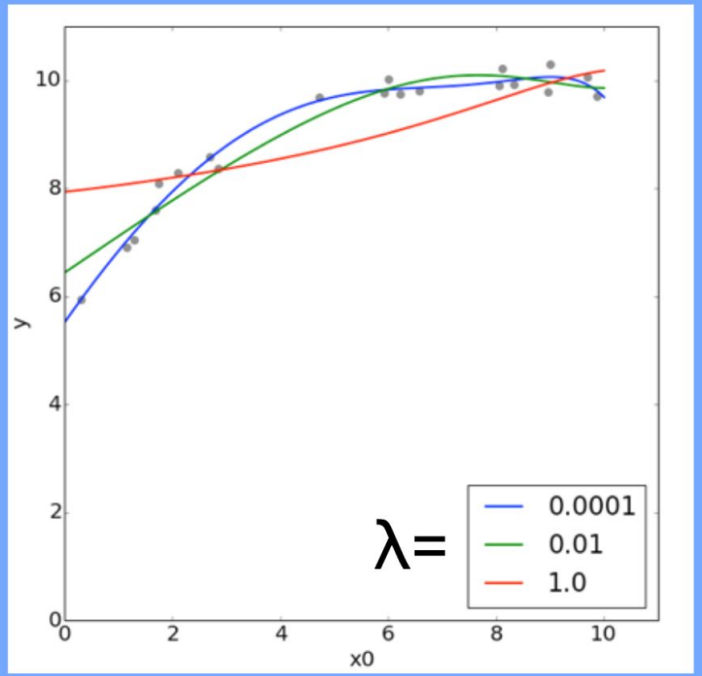


Ridge Regression

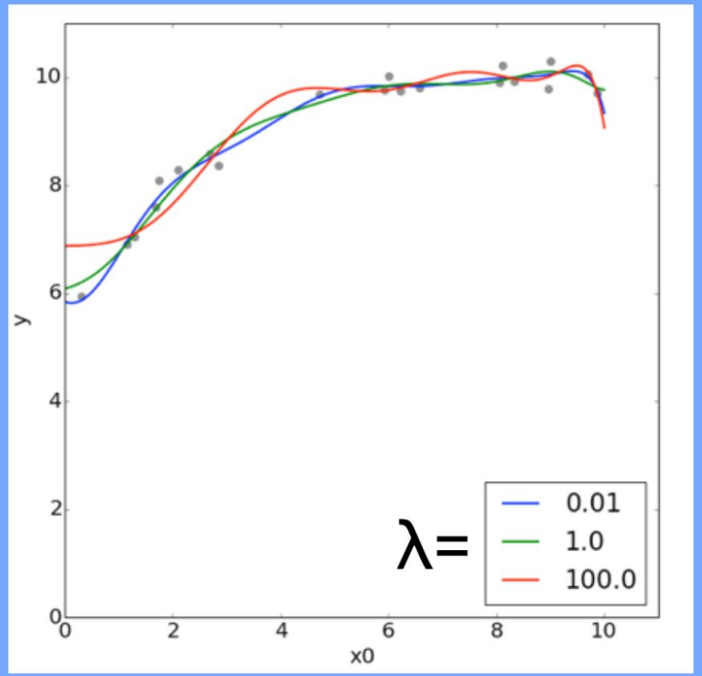


galvanize

Normalized Data



Non-Normalized Data



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

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

We model the world as:

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

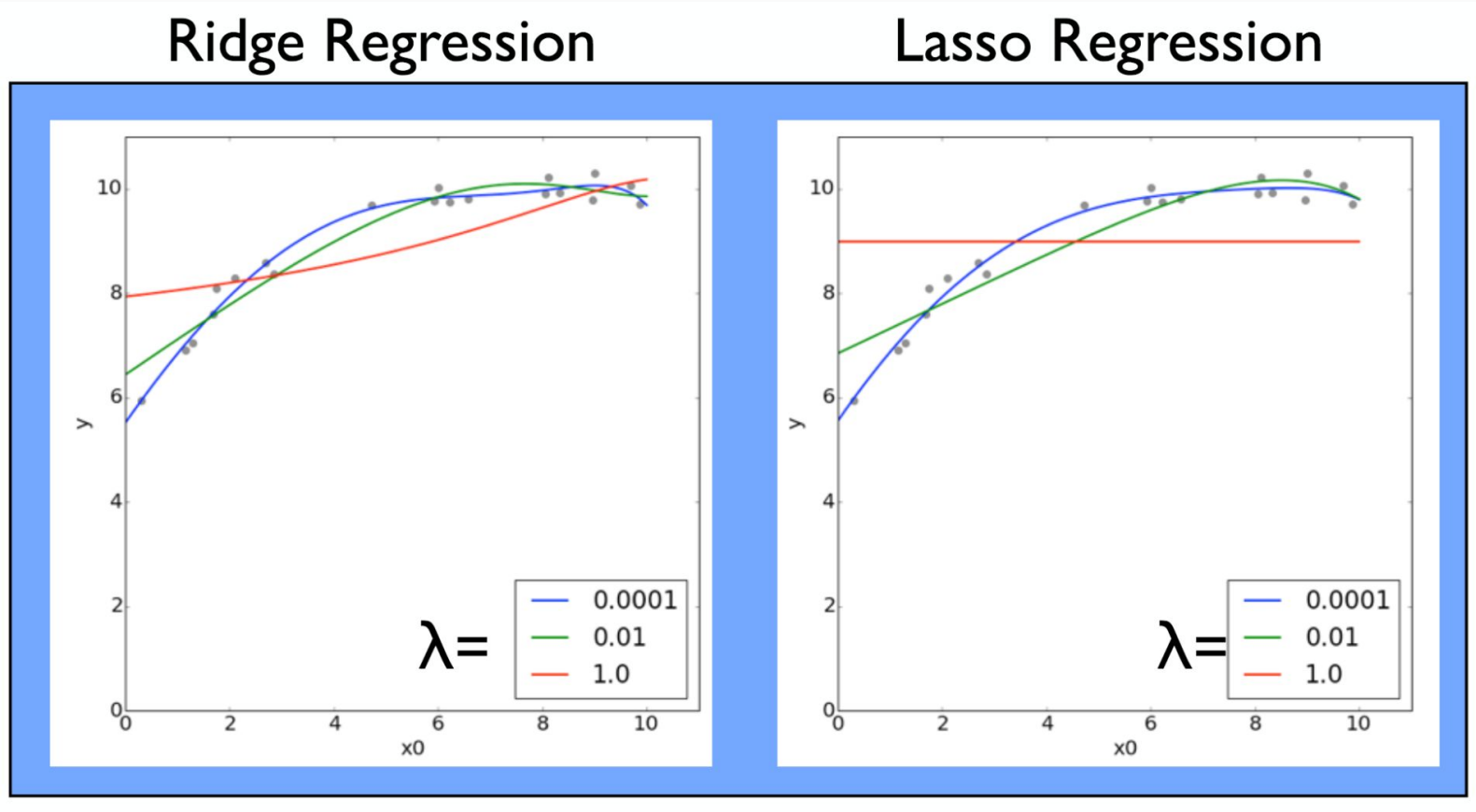
(same as before)

We estimate the model parameters to minimizing:

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

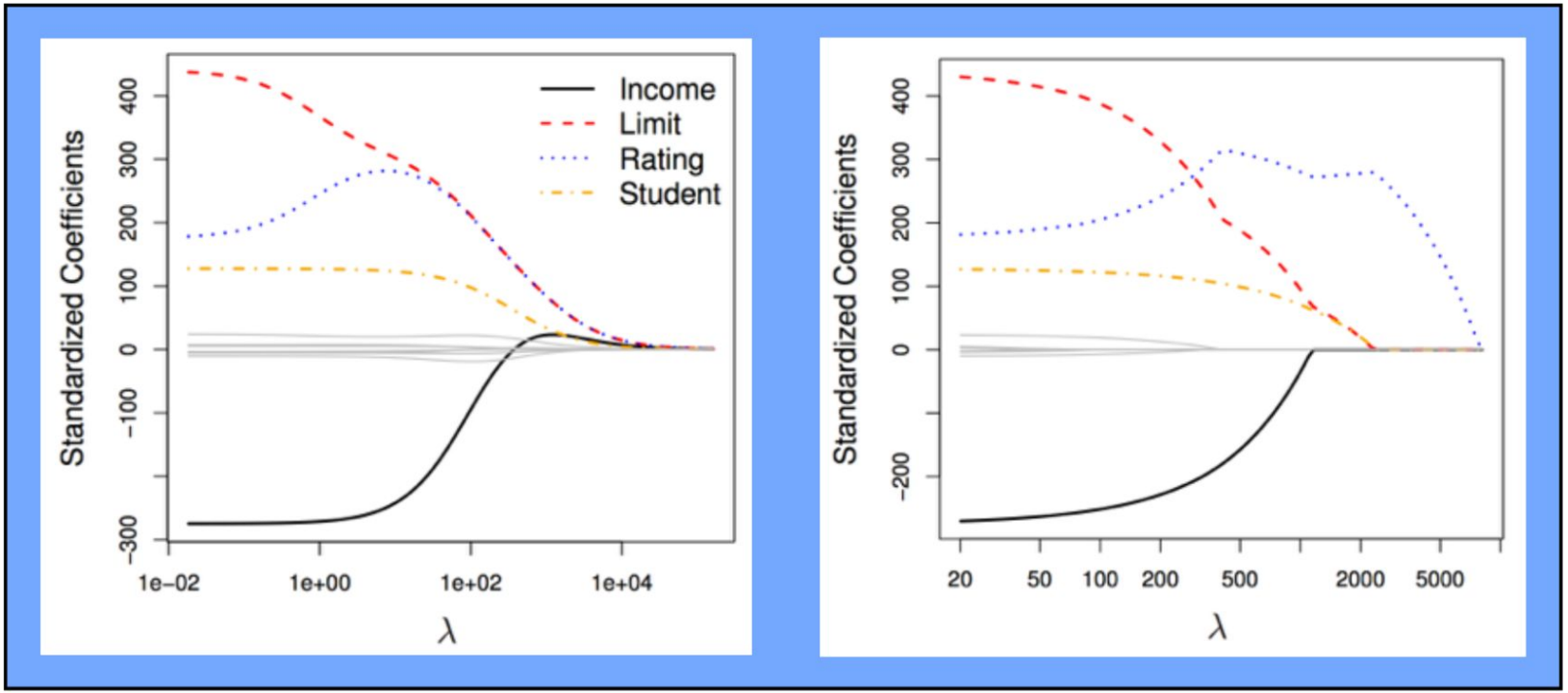
(the “regularization” parameter)

(absolute value instead of squared)


Ridge Regression
Lasso Regression
y
x0
λ=
0.0001
0.01
1.0

galvanize

Ridge vs. Lasso

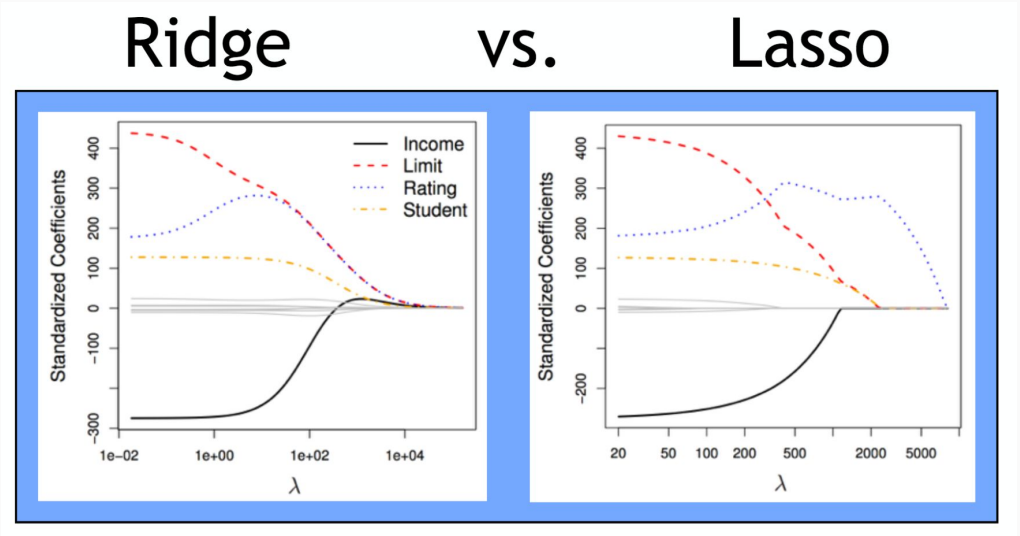


Ridge vs LASSO

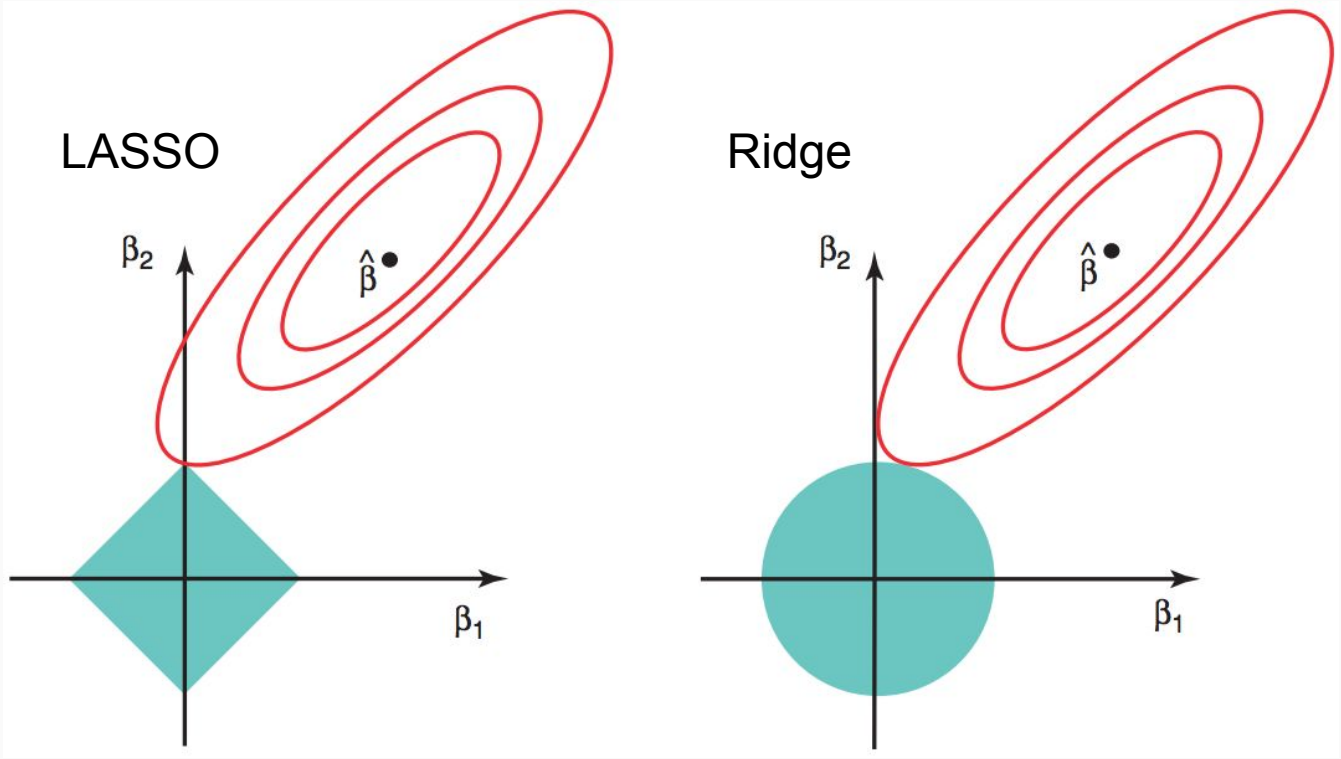
- Ridge forces parameters to be small + Ridge is computationally easier because it is differentiable
- Lasso tends to set coefficients exactly equal to zero
  - This is useful as a sort-of “automatic feature selection” mechanism,
  - leads to “sparse” models, and
  - serves a similar purpose to stepwise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

Which is better depends on your dataset!

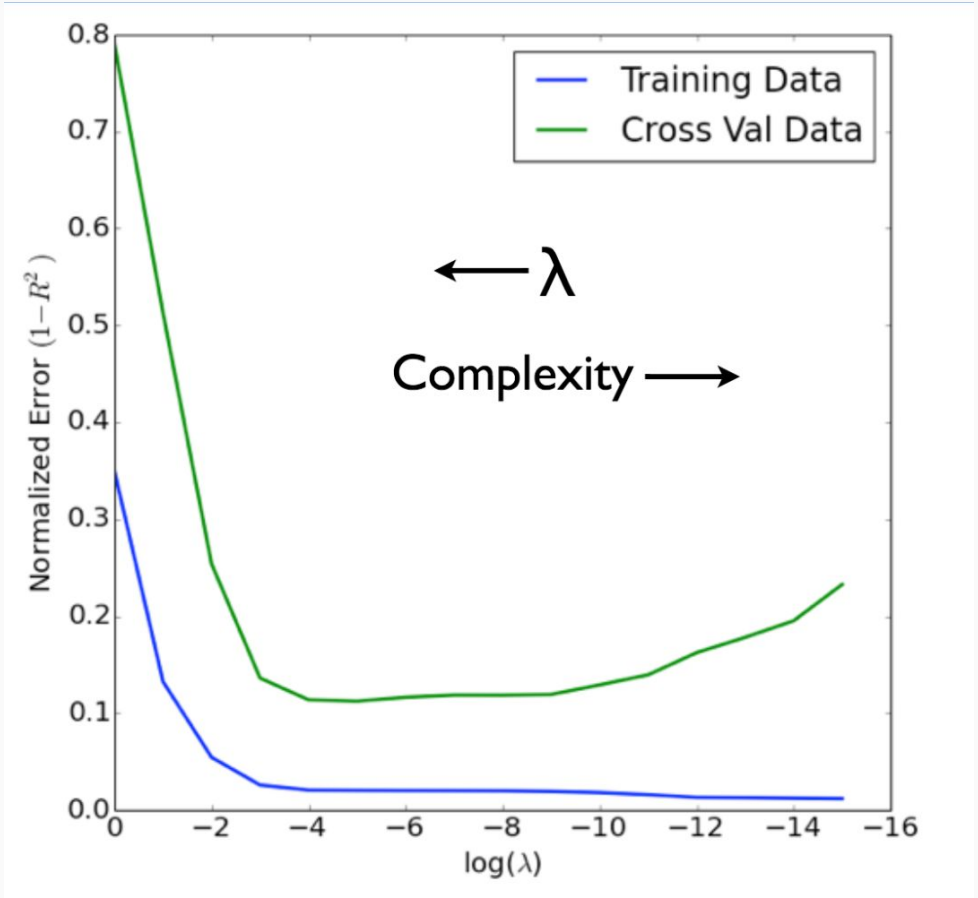
True sparse models will benefit from lasso; true dense models will benefit from ridge.



Ridge vs LASSO



Chose lambda via Cross-Validation



galvanize

scikit-learn

Classes:

- sklearn.linear_model.**LinearRegression**(...)
- sklearn.linear_model.**Ridge**(alpha=my_alpha, ...)
- sklearn.linear_model.**Lasso**(alpha=my_alpha, ...)

All have these methods:

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

galvanize

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)