Lecture 6:
High Dimensional Regression

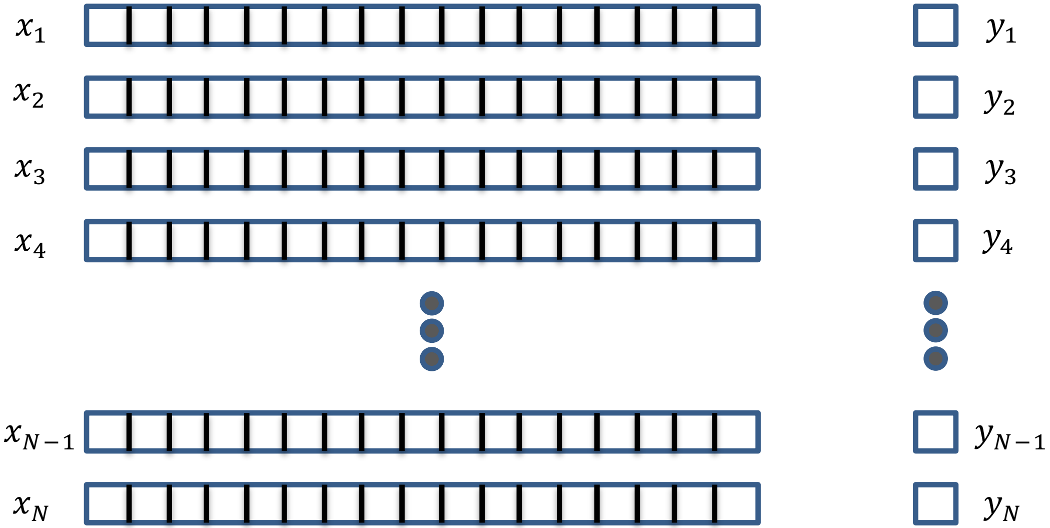
David Carlson

Overview

- What happens happens when we have lots of lots of features, e.g.
  - mRNA expression
  - microbiome
  - Pretty much any -omics
  - Lots of sensing problems (hyperspectral imaging, etc.)

FLASHBACK TO LINEAR REGRESSION

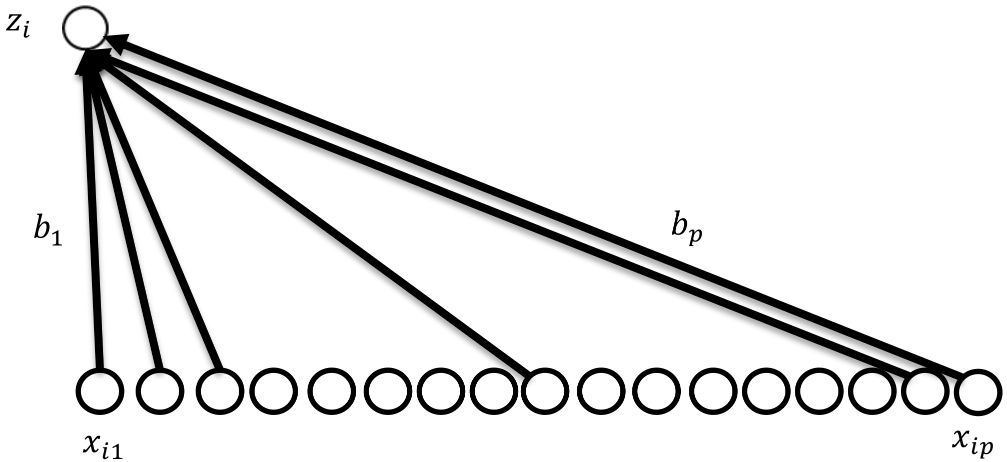
Training Set (Historical Data)



Start by limiting y to a binary outcome:

- False/True
- 0/1

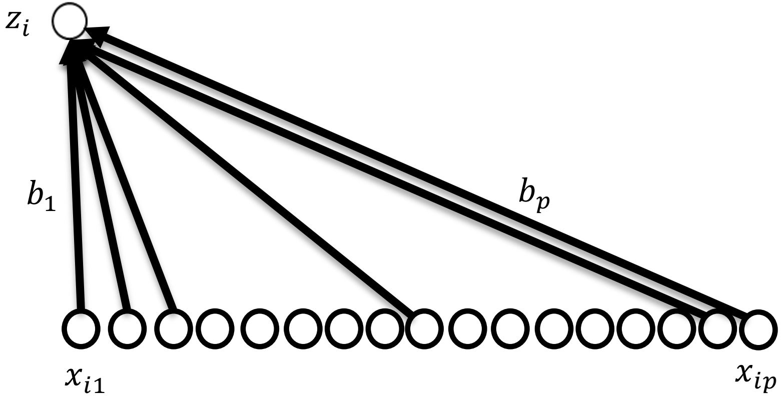
Linear Predictive Model

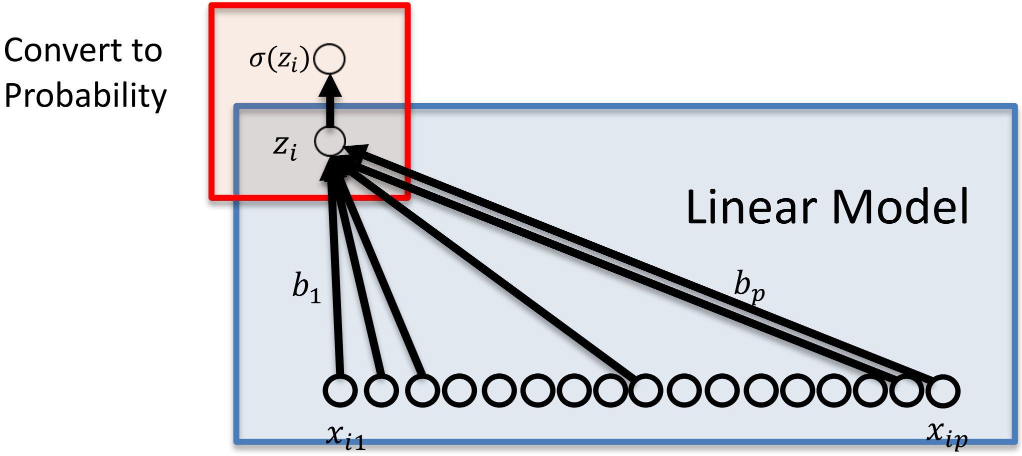


$$z_i = b_0 + (b_1 \times x_{i1}) + (b_2 \times x_{i2}) + \cdots + (b_p \times x_{ip})$$

Regression

Classification



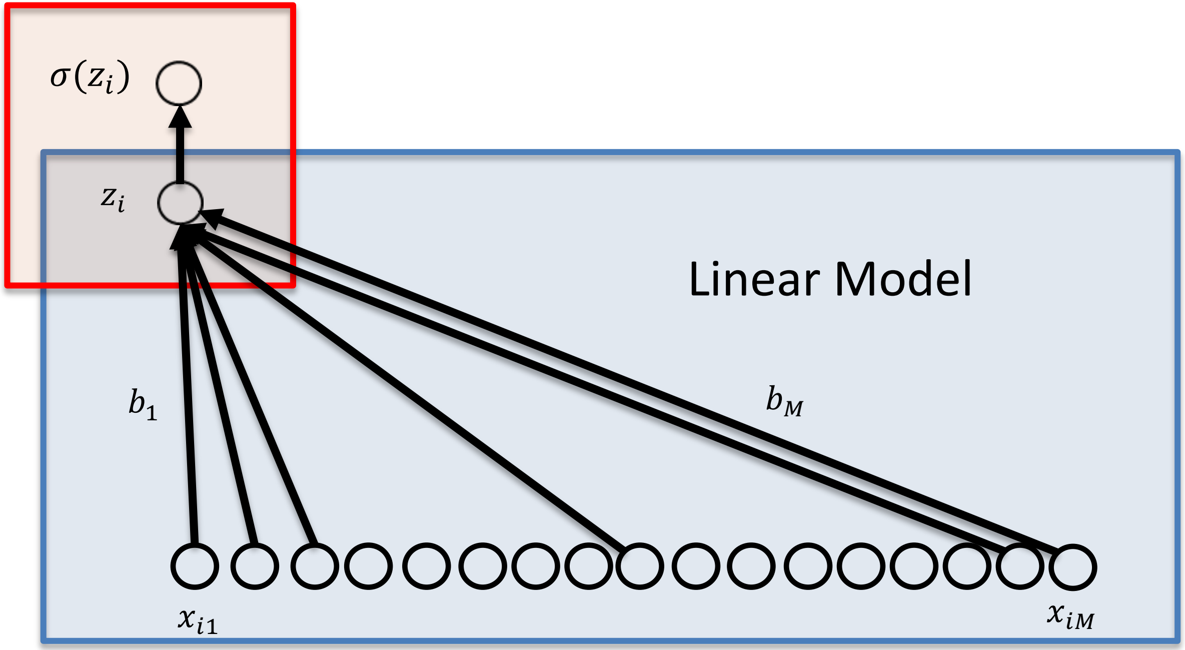


$$z_i = b_0 + (b_1 \times x_{i1}) + (b_2 \times x_{i2}) + \cdots + \left(b_p \times x_{ip}\right)$$

If preforming classification: add sigmoid

Logistic Regression

Convert to
Probability



Flashback to Linear Regression

- In linear regression, we tried to minimize the mean squared error (MSE) or the residual sum of squares (RSS), with the following formulation:

$$\boldsymbol{\beta}^* = \arg\min_{\boldsymbol{\beta}\in\mathbb{R}^p} \sum_{i=1}^{n} (y_i - \sum_{j=1}^{p} x_{ip}\beta_p - \beta_0)^2$$

- As p goes big we're going to have issues.

Flashback to Linear Regression

- In linear regression, we minimize

$$\boldsymbol{\beta}^* = \arg\min_{\boldsymbol{\beta}\in\mathbb{R}^p} \sum_{i=1}^{n} (y_i - \sum_{j=1}^{p} x_{ip}\beta_p - \beta_0)^2$$

- This can be written in vector form (ignoring the intercept for simplicity):

$$\boldsymbol{\beta}^* = \arg\min_{\boldsymbol{\beta}\in\mathbb{R}^p} \sum_{i=1}^{n} (y_i - \boldsymbol{\beta} \odot \boldsymbol{x})^2$$

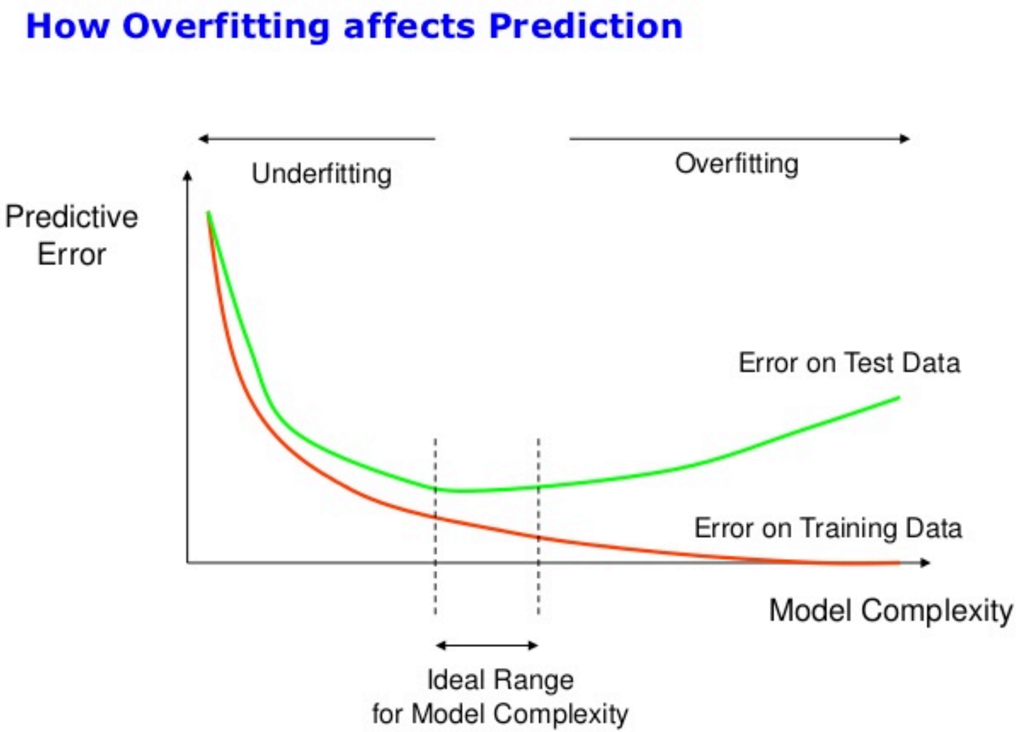
$$= \arg\min_{\boldsymbol{\beta}\in\mathbb{R}^p} \sum_{i=1}^{n} (y_i - \boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{x})^2$$

Model Complexity and Predictive Error

When we perform validation procedures, we're trying to balance "model complexity" with the amount of data we have and the complexity of the problem.

In K-nearest neighbors, using more neighbors is a "less complex" model because it leads to smoother functions. (1-neighbor is really jumpy!)

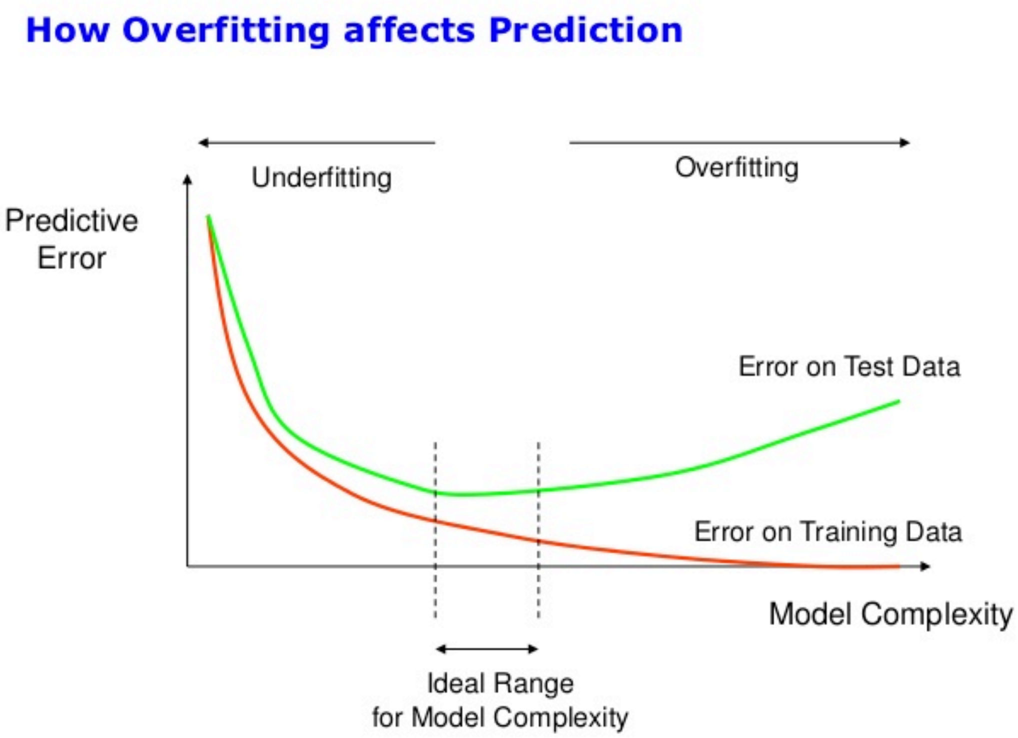
In our examples so far, linear regression was too simple of a model and "underfit" the data.



Model complexity is affected by dimension

More dimensions = more complex

Linear regression will (quite quickly!) start to overfit as the number of dimensions increases…



Increasing dimensionality

High-Dimensional Problems are Ubiquitous

- Lots of areas where the number of features is quite large:
  - Genetics
  - Microbiome
  - Electronic health records
  - Text
  - Images
  - Sensor networks
  - Etc.

Lets consider a large-scale problem

- Let’s consider gene expression from a microarray
  - n=400 (participants in the study)
  - p~=30,000 (genes measures)
- Linear regression is utterly useless here
  - Infinite number of possible parameters that give *perfect* prediction
  - Let’s understand why this happens

Let’s consider an example in 2 dimensions

- Suppose we have 2 features and we’ve observed a single data point
- Want to optimize:

$$\min_{b_1,b_2}(y - b_1x_1 - b_2x_2)^2$$

Let's consider an example in 2 dimensions

- Want to optimize:

$$\min_{b_1,b_2}(y - w_1x_1 - b_2x_2)^2$$

- Let's find derivatives:

$$\text{for } b_1\text{: } (y - b_1x_1 - b_2x_2)x_1$$
$$\text{for } b_2\text{: } (y - b_1x_1 - b_2x_2)x_2$$

- Optimal when both are equal to 0, or

$$(y - b_1x_1 - b_2x_2) = 0$$

$$b_2 = \frac{x_1}{x_2}b_1 - \frac{y}{x_2}$$

- This is a *linear* equation. Any value of b_1 has a value of b_2 that minimizes this function

Best Fit Line

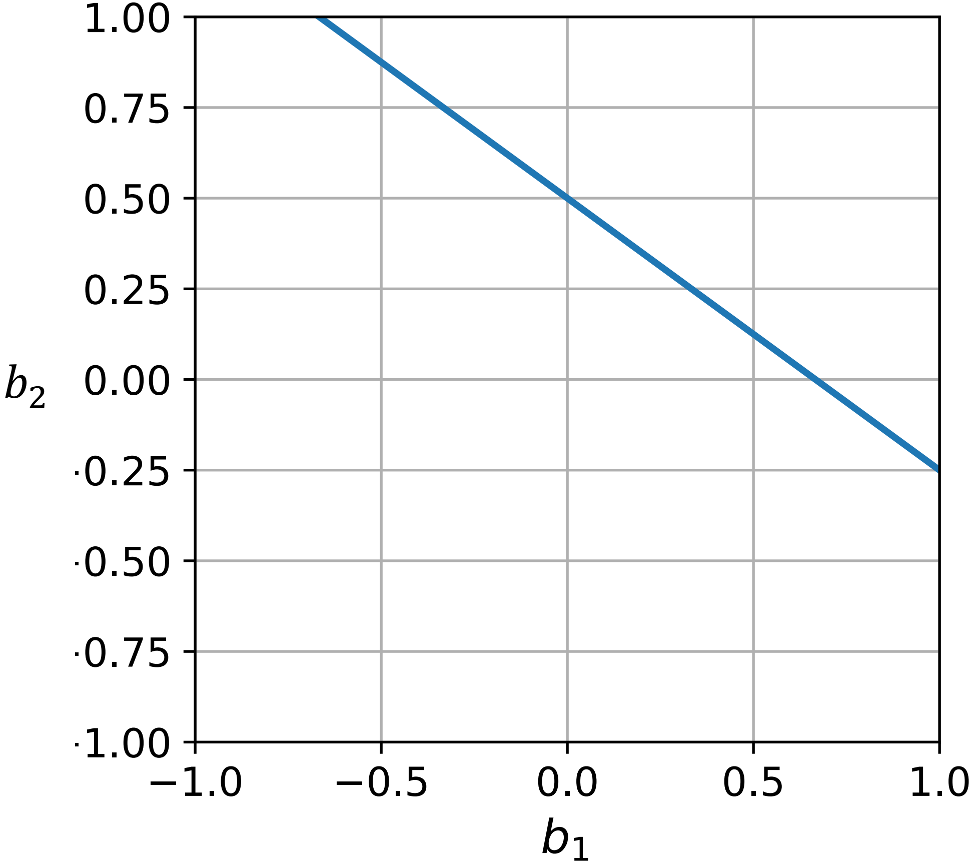
Suppose that we have:

$$x_1 = 3, x_2 = -4, y = 2$$

Then the line of "perfect fits" (i.e. training MSE is 0) is shown on the right.

Without any other information, all of those lines are equally good at fitting the data.

How do we pick which one?



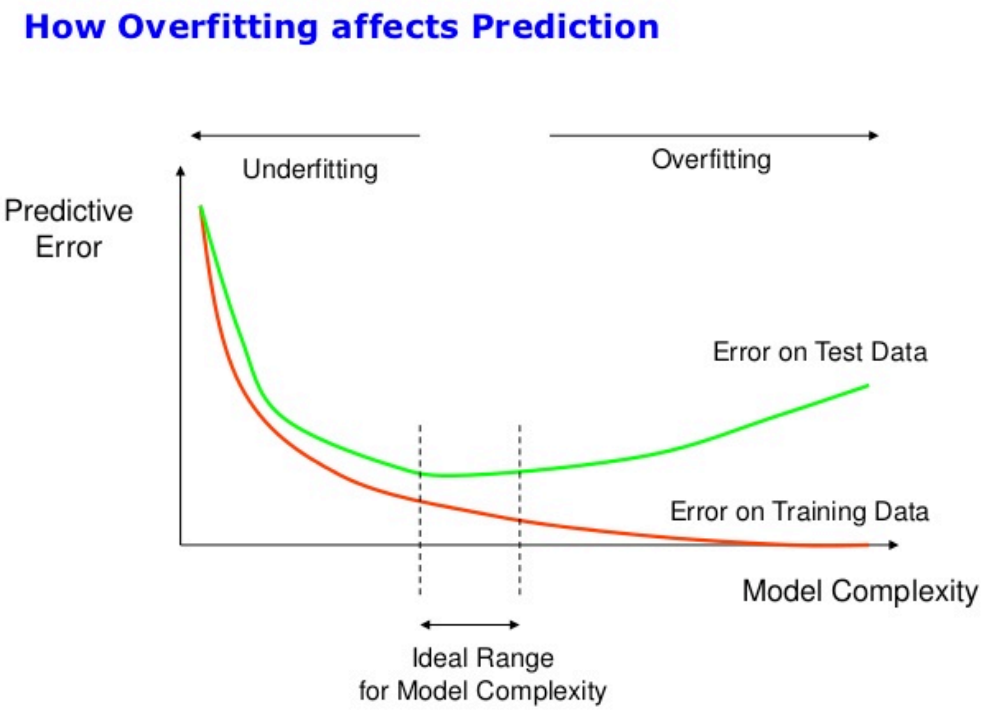
We want as simple a model as possible

We want a simple as model as fits the data “well.”

Some ways we could think about doing this:

- Use as few features to predict as possible (i.e. we have 1000 features, only use a handful)
- Use a ”simpler” set of features, describing by a smaller length

We can do this by adding constraints or penalties.



Two Different Norms

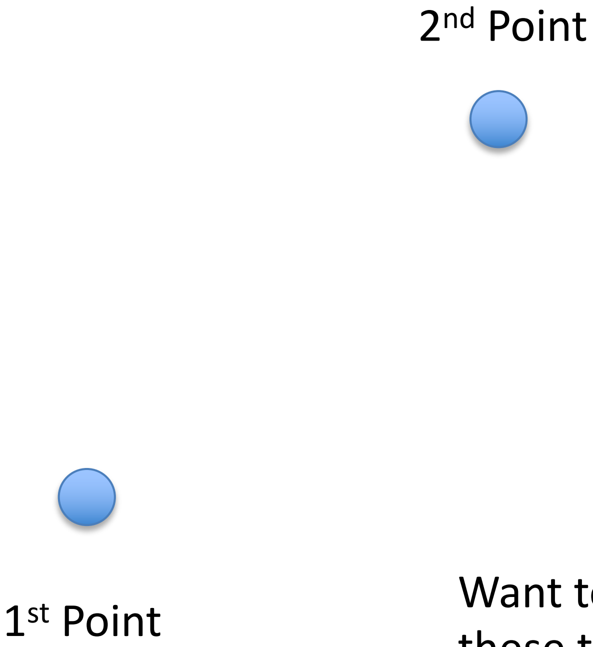
- A norm is a particular way of defining distance
- We have already talked about the l2 norm:

  $$||x||_2 = \sqrt{\sum_i x_i^2}\,,$$

  which is the length of a straight line in a high dimensional space.
- We also will care about the l1 norm or "city-block" distance:

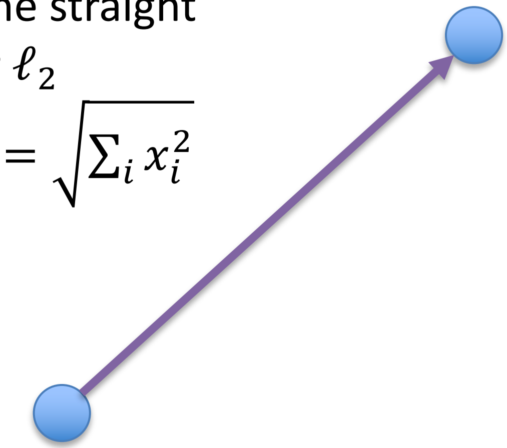
  $$||x||_1 = \sum_i |x_i|$$

Visualization in 2-D



Want to calculate distance between these two points.

Visualization in 2-D



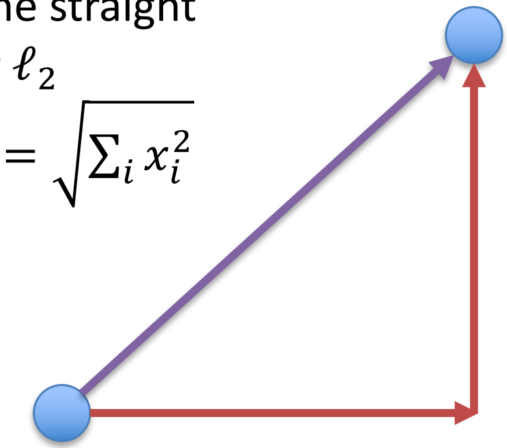
2nd Point

The length of the straight line is the L^2 or ℓ_2 distance, $\|x\|_2 = \sqrt{\sum_i x_i^2}$

1st Point

Want to calculate distance between these two points.

Visualization in 2-D



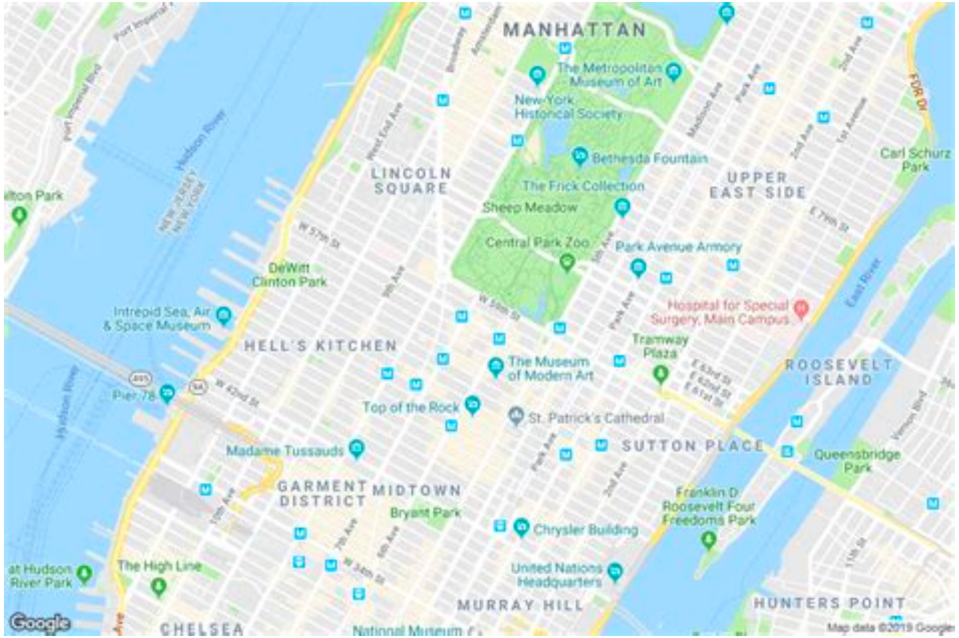
2nd Point

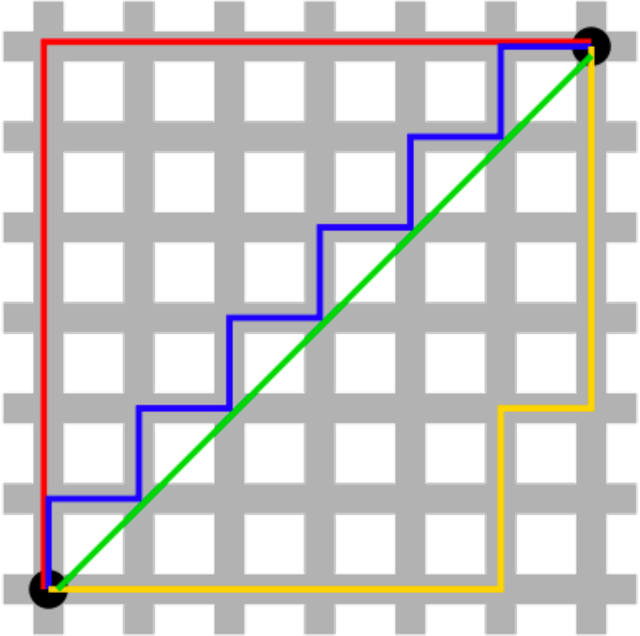
The length of the straight line is the L^2 or ℓ_2 distance, $\|x\|_2 = \sqrt{\sum_i x_i^2}$

The length of each dimension added together is the L^1 or ℓ_1 distance, $\|x\|_1 = \sum_i |x_i|$

1st Point

Can view this as a distance for cities (or taxi-cabs)





Taxis can only go on streets…

https://en.wikipedia.org/wiki/Taxicab_geometry

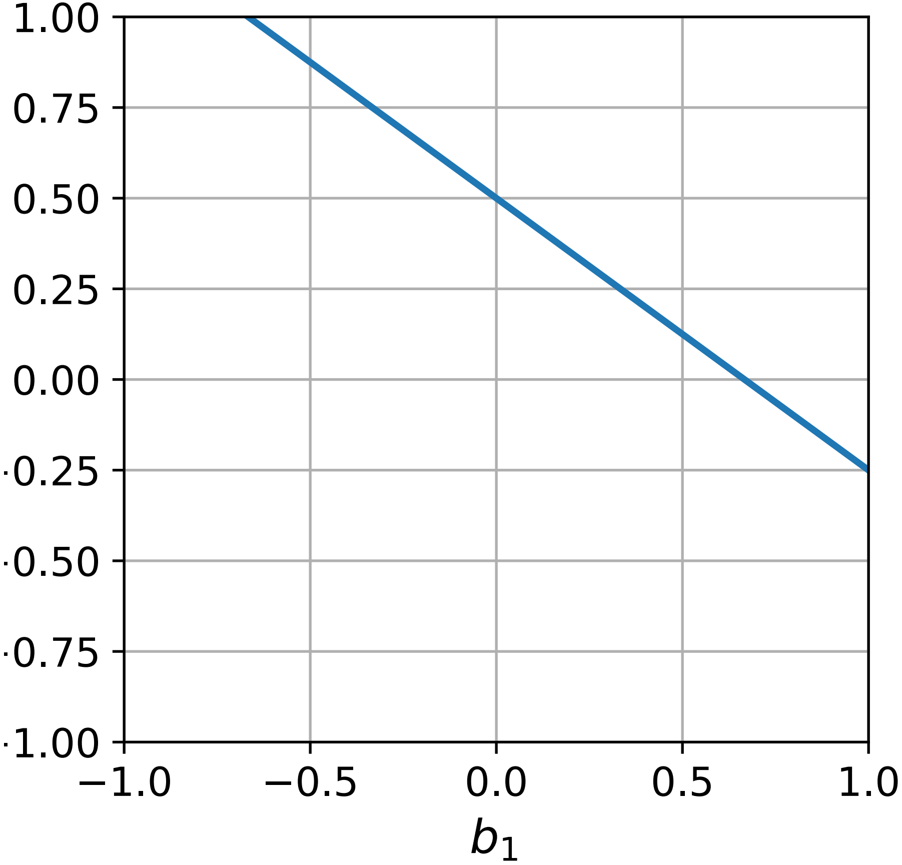
How does this impact our linear regression?

On the right we have our best fit line.

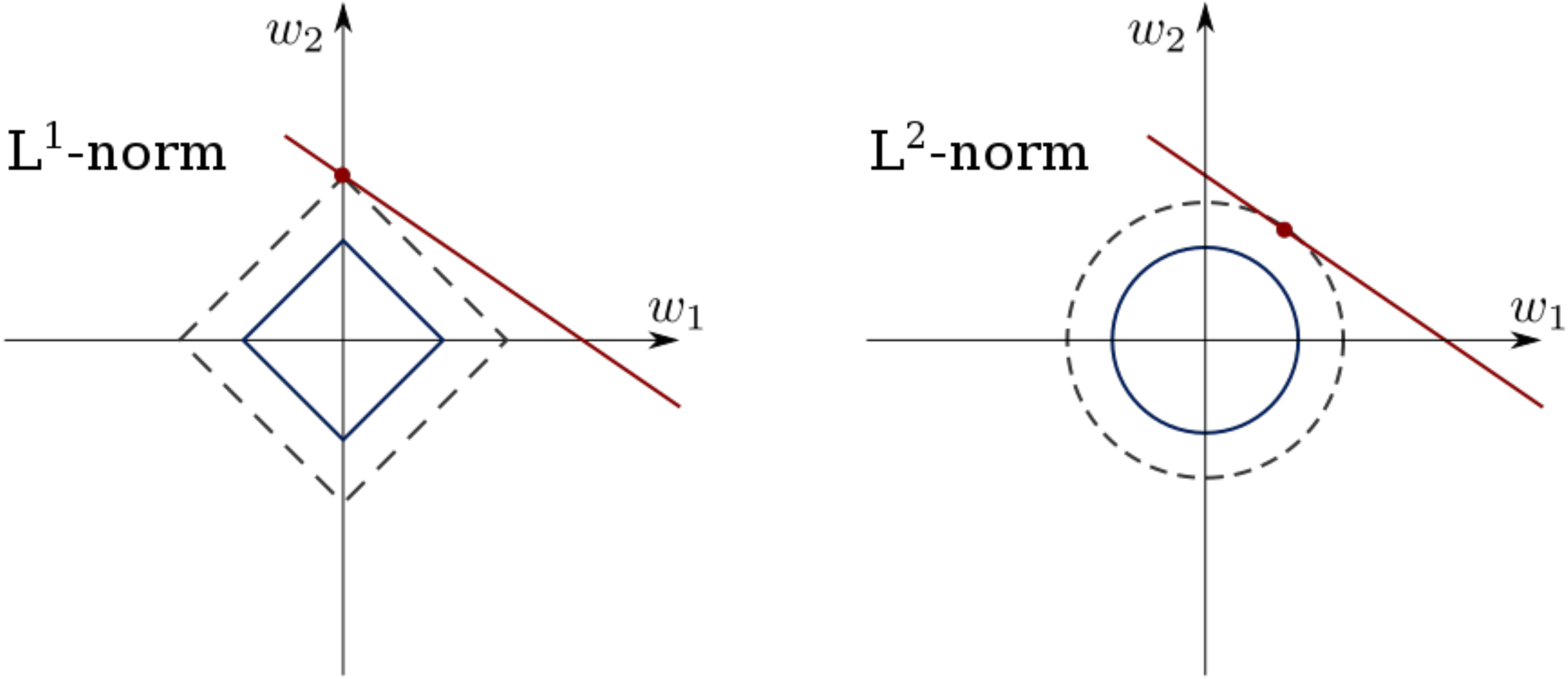
We could think about choosing the perfect fit with the smallest distance to the origin with either of these distances (i.e.):

- $\min \|b\|_2$
- $\min \|b\|_1$

What does this actually do?



Geometric Interpretation



Returning to Higher Dimensions

- We care about our full linear regression problem:

$$\boldsymbol{\beta}^* = \arg\min_{\boldsymbol{\beta}\in\mathbb{R}^p} \sum_{i=1}^{n} (y_i - \boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{x})^2$$

- Can we incorporate these same ideas into this regression more generally?

Ridge Regression

- One common assumption on linear regression is that our vector of parameters should be as close to 0 as possible.
- Using an l2 distance is called “Ridge Regression,” which augments the original problem statement for linear regression.

$$\boldsymbol{\beta}^* = \arg\min_{\boldsymbol{\beta}\in\mathbb{R}^p} \sum_{i=1}^{n} \left(y_i - \boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{x}\right)^{\mathbf{2}} + \lambda \sum_{j=1}^{p} \beta_j^2$$

$$= \arg\min_{\boldsymbol{\beta}\in\mathbb{R}^p} \sum_{i=1}^{n} \left(y_i - \boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{x}\right)^{\mathbf{2}} + \lambda||\boldsymbol{\beta}||_2^2$$

$$= \arg\min_{\boldsymbol{\beta}\in\mathbb{R}^p} ||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}||_2^2 + \lambda||\boldsymbol{\beta}||_2^2 \text{ (matrix shorthand)}$$

- The term $\lambda||\boldsymbol{\beta}||_2^2$ is the ridge regression *penalty.*
  - This is a minimization problem, so we want to minimize the MSE
  - Moving a regression value away from 0 *increases* the penalty
  - There’s a trade-off here: moving away from 0 must decrease the MSE (improve predictive performance) more than the penalty paid
  - λ controls how strong this penalization is; what happens at λ=0?

Lasso

- Instead, we can use the city block distance from before (the L1 norm):

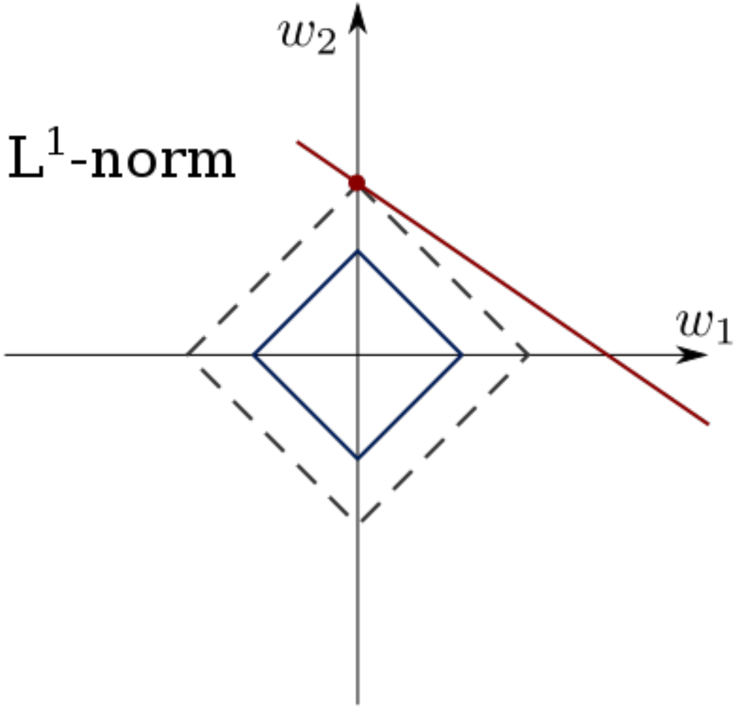
$$\boldsymbol{\beta}^* = \arg\min_{\boldsymbol{\beta}\in\mathbb{R}^p} \sum_{i=1}^{n} \left(y_i - \boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{x}\right)^{\mathbf{2}} + \lambda \sum_{j=1}^{p} |\beta_j|$$

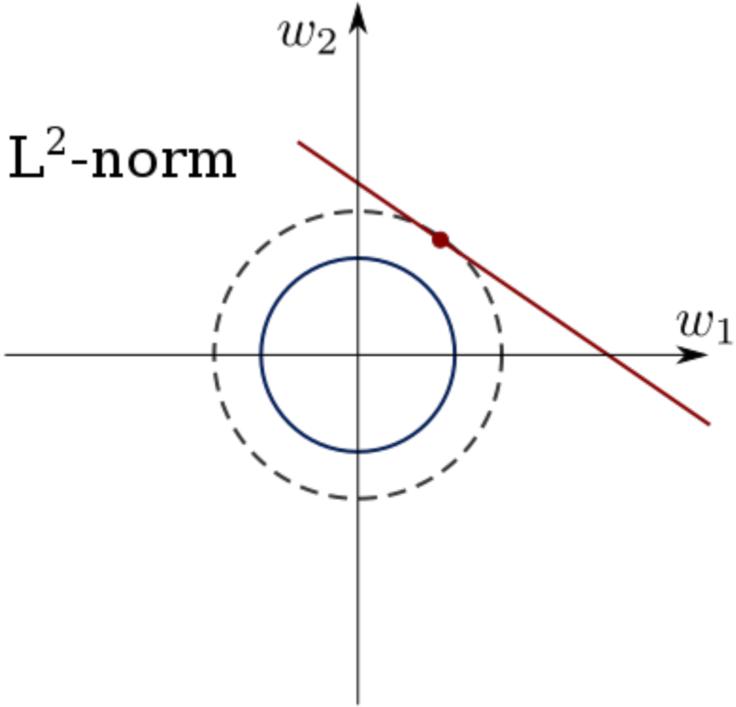
$$= \arg\min_{\boldsymbol{\beta}\in\mathbb{R}^p} \sum_{i=1}^{n} \left(y_i - \boldsymbol{\beta}^{\mathrm{T}}\boldsymbol{x}\right)^{\mathbf{2}} + \lambda ||\boldsymbol{\beta}||_1$$

$$= \arg\min_{\boldsymbol{\beta}\in\mathbb{R}^p} ||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}||_2^2 + \lambda ||\boldsymbol{\beta}||_1 \text{ (matrix shorthand)}$$

- When should we prefer one to the other?

Returning to our Geometric Interpretation

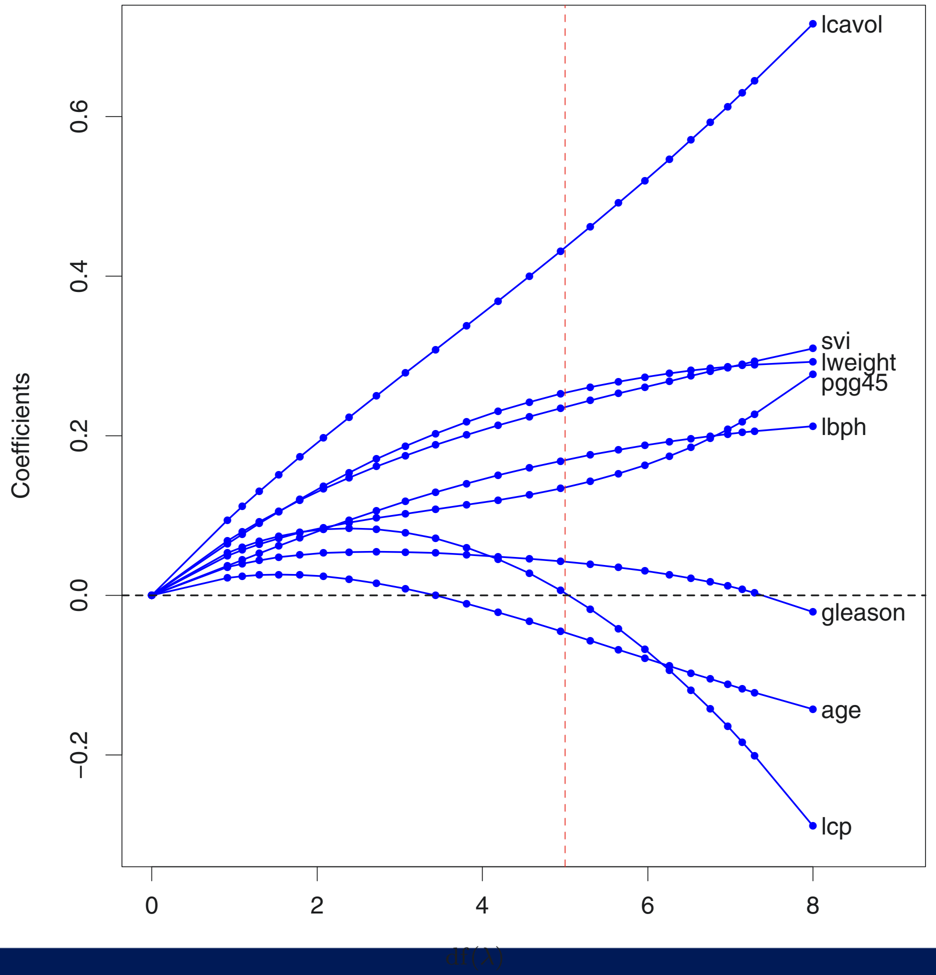




Ridge Regression Visualization

On a small prostate cancer dataset.

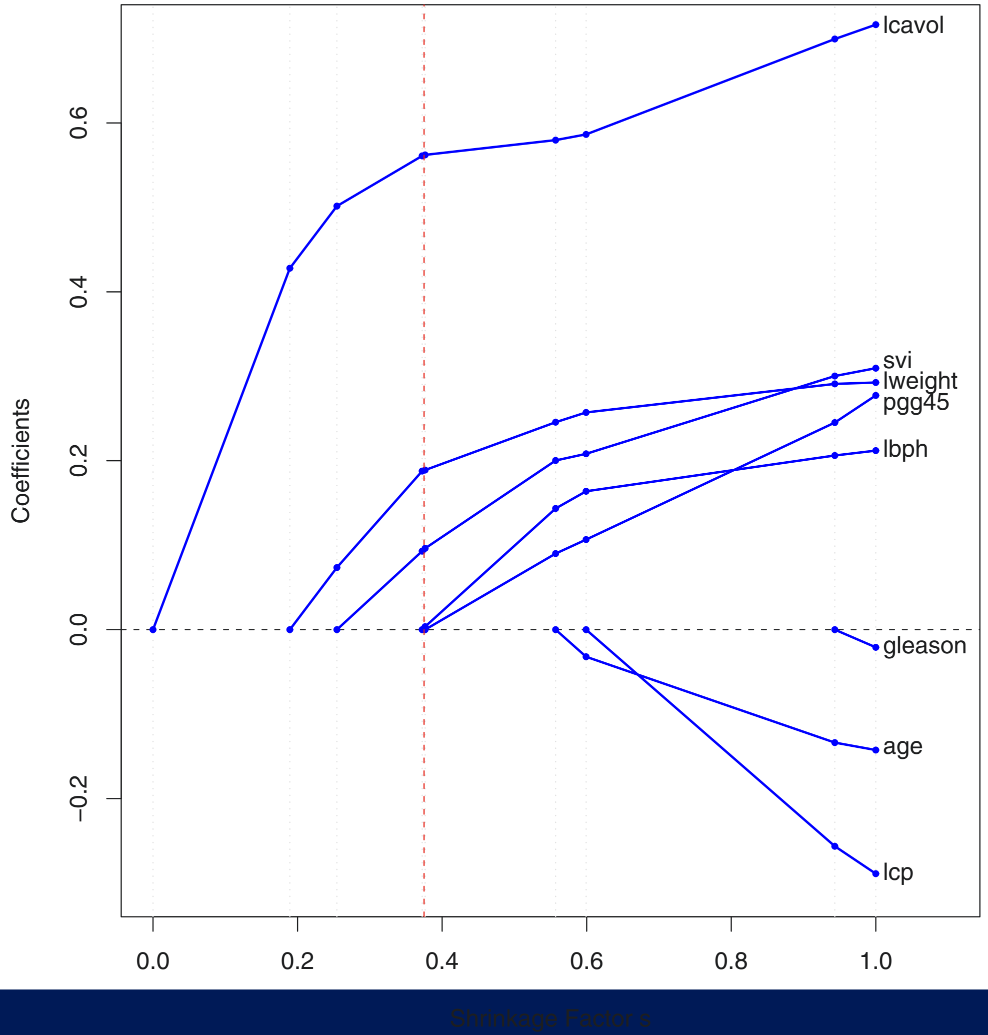
The strength of that penalty controls what the regression does.



Lasso Visualization

On a small prostate cancer dataset.

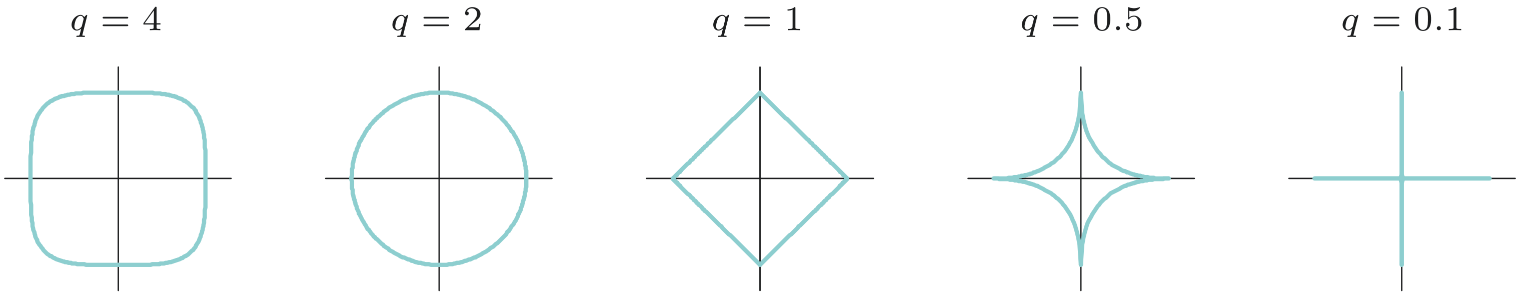
The strength of that penalty controls what the regression does.



Which regularization to choose?

- Properties of Ridge Regression:
  - Minimizes impact of mostly irrelevant features
  - Uses all variables to some extent
- Properties of Lasso:
  - Many parameters will *exactly* be zero; implicitly performs feature selection.
  - Hard zeros can be overly restrictive
  - Sparse feature set leads to easier interpretations
  - Rarely includes correlated features (i.e. if feature 1 & 2 are nearly collinear, typically will only include one of them; this is both good and bad)
  - If there are *n* data points, can only include *n* features at most (a bigger issue when *p>>n)*
- Are other distances also good to use?

Visualization of Other Norms



$q = 4$ $q = 2$ $q = 1$ $q = 0.5$ $q = 0.1$

What about L$_0$?

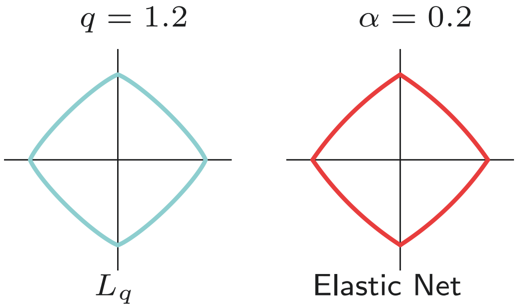
- Other, we want to pick a small subset of features to use, which we can write as:

$$\boldsymbol{\beta}^* = \arg\min_{\boldsymbol{\beta}\in\mathbb{R}^p} ||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}||_2^2 + \lambda \sum_{j=1}^{p} 1_{(\beta_j \neq 0)}$$

- The penalty term is often written as $\lambda||\boldsymbol{\beta}||_0$, which is the L0 pseudo-norm
  - It is not a norm! Doesn't satisfy all the mathematical properties…
- Want to use as few features as possible. Doesn't matter how big or small they are.
- This is conceptually and statistically nice—unfortunately, turns into an extremely difficult math problem to solve

Elastic Net

- Well, if the Lasso has some benefits and the Elastic net has some benefits, why don't we just combine them?
- This strategy is called the elastic net: $\boldsymbol{\beta}^* = \arg\min_{\boldsymbol{\beta}\in\mathbb{R}^p} ||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}||_2^2 + \lambda((1-\alpha)||\boldsymbol{\beta}||_1 + \alpha||\boldsymbol{\beta}||_2^2)$
- A big downside is that we now have 2 parameters to tune through cross-validation rather than a single tuning parameter.



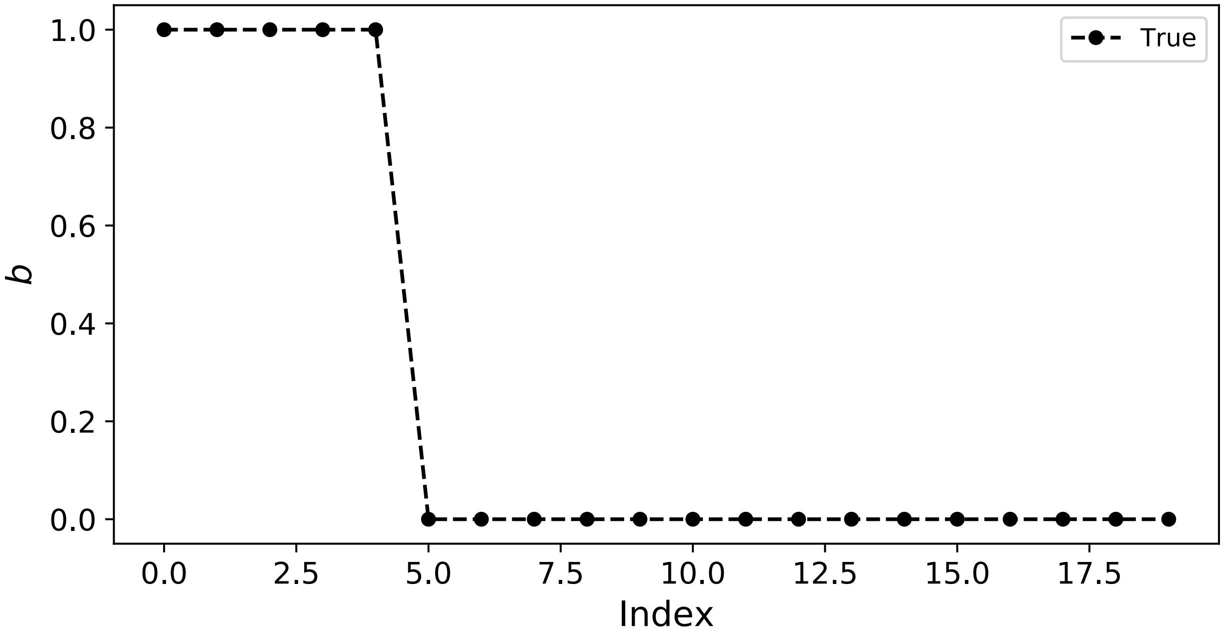
What about high *q*?

- Why do we stop a *q=2*? What magical about the range from 1 to 2?
- Higher values of *q* actually **encourage** model complexity!
  - When q is large, we pay a premium for using any one feature too much, and are *encouraged to use more* features.
- Generally, no benefits from a higher *q*. Like everything, there are special cases…

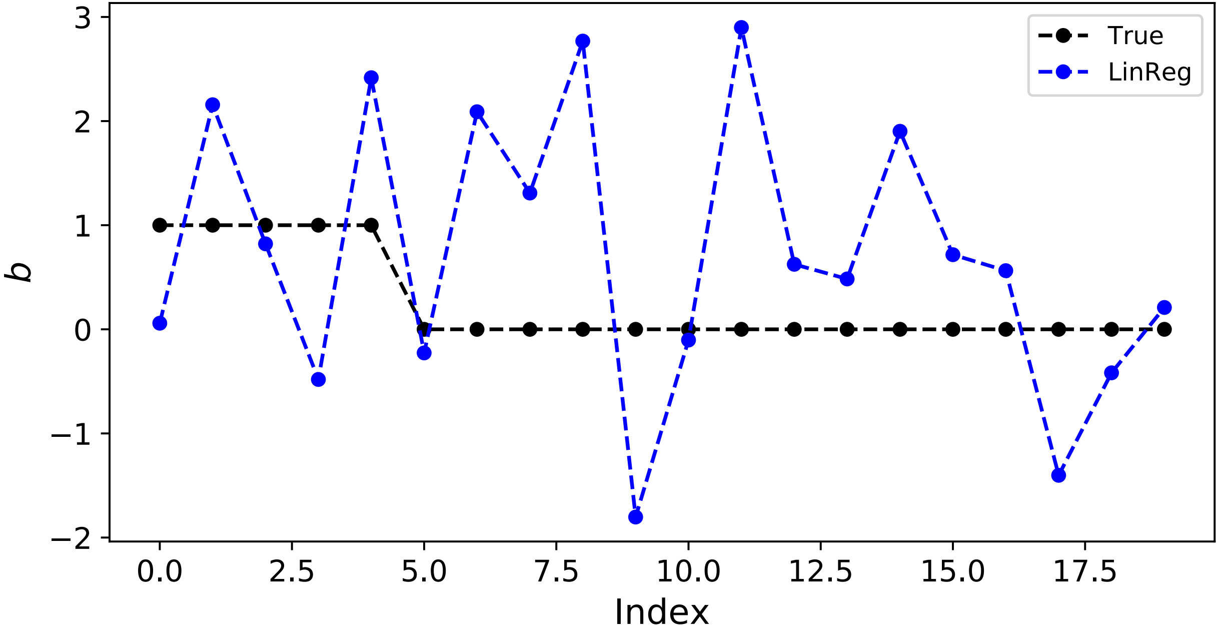
MODEL SELECTION IN REGRESSION

What does this look like in practice?

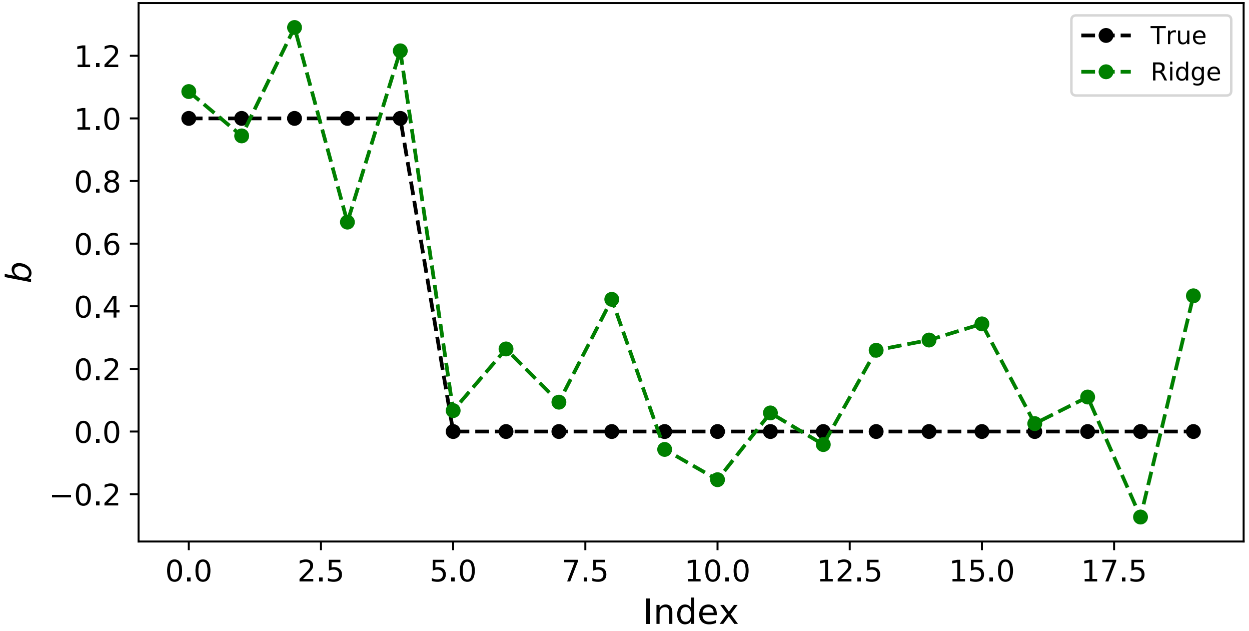
Generate synthetic data:
n=21
p=20



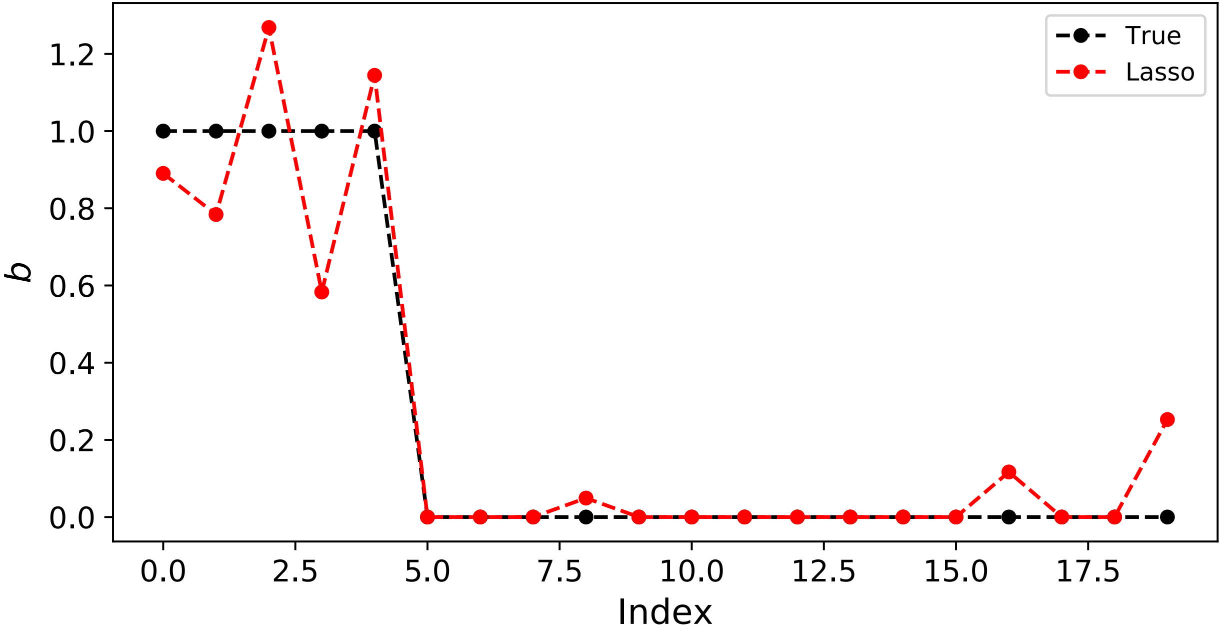
Linear Regression



Ridge Regression



Lasso Regression

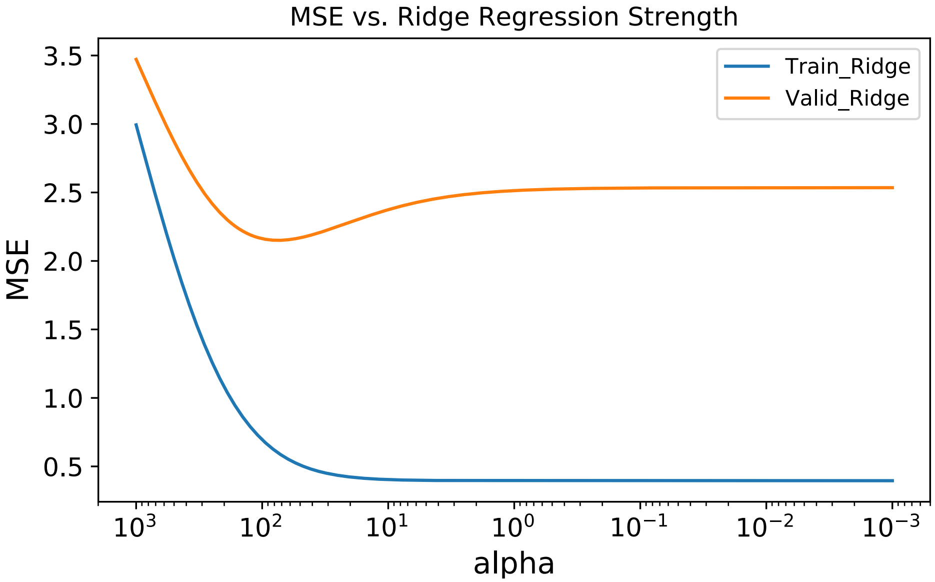


Model selection is quite similar as in kNN

We get to choose what strength of penalty we use, which requires setting a single tuning parameter.

We do the same exact thing that we did before, where we:

- Split data
- Try different settings for the tuning parameters
- Choose the best model



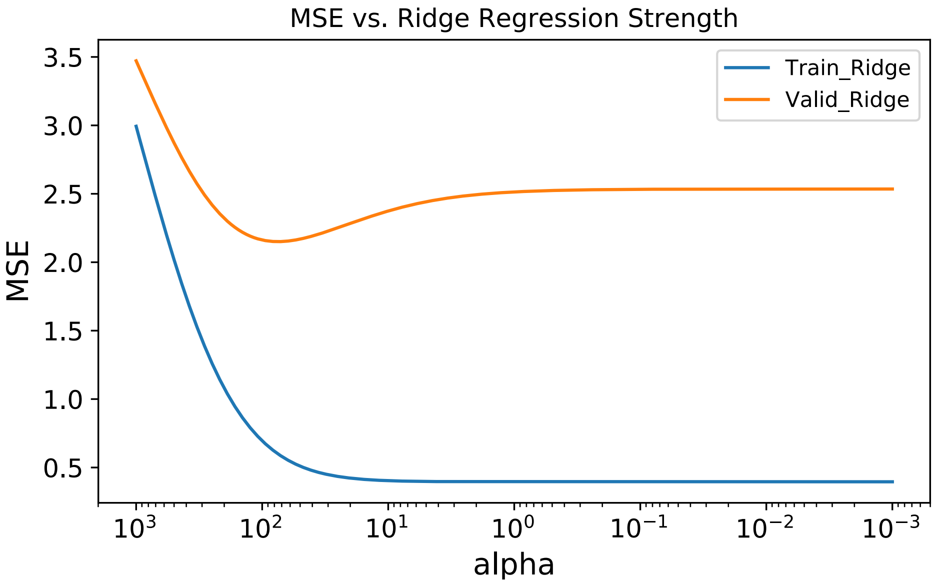
Model selection is quite similar to in kNN

We get to choose what strength of penalty we use, which requires setting a single tuning parameter.

We do the same exact thing that we did before, where we:

- Split data
- Try different settings for the tuning parameters
- Choose the best model

On the right is with ridge regression



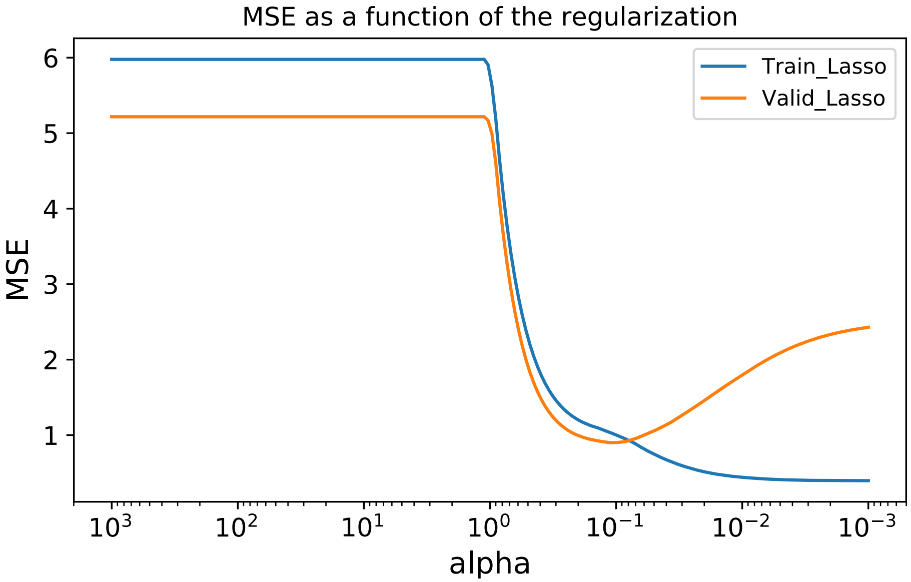
Model selection is quite similar to in kNN

We get to choose what strength of penalty we use, which requires setting a single tuning parameter.

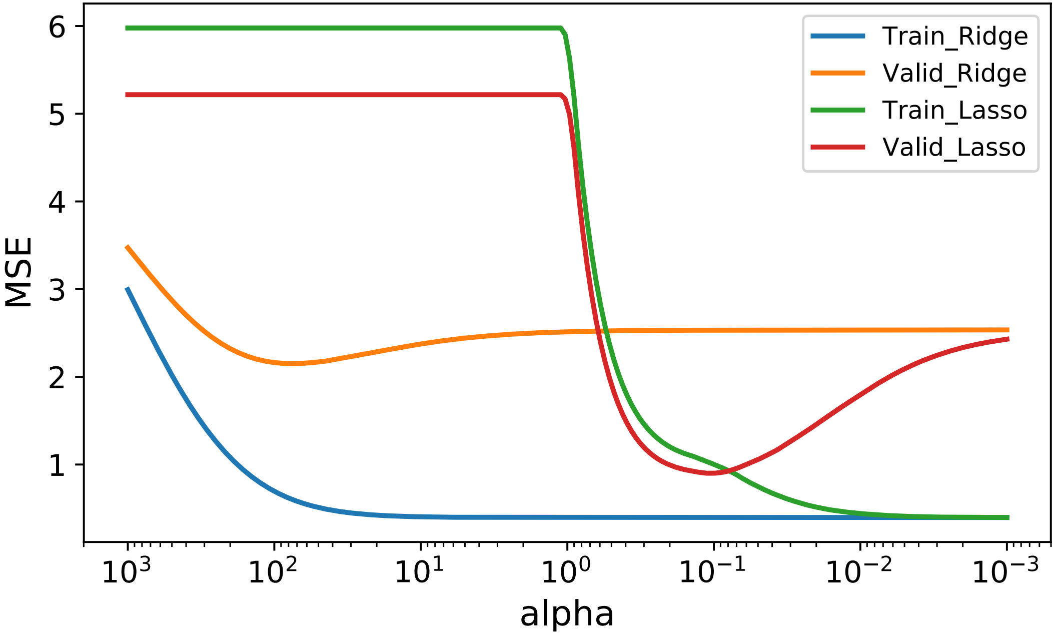
We do the same exact thing that we did before, where we:

- Split data
- Try different settings for the tuning parameters
- Choose the best model

On the right is lasso regression



Penalization is different for both models



Conclusions up to now

- Penalized regression methods are extremely common in practice
  - Some of the most widely used techniques; fundamental to many analysis techniques
- The idea that lasso can be used to pick *relevant* features can also be extended to work with other classifiers, including our k-Nearest Neighbors

Penalized Logistic Regression

- These penalization methods aren’t limited to linear regression, can easily be included in any generalized linear model, such as our previously discussed Logistic Regression.

- We can do this by combining the logistic loss function with the ridge regression penalty:
- $\boldsymbol{\beta}^* = \arg\min\left[\sum_{i=1}^{n} y \log \sigma(\boldsymbol{x}_i^T \boldsymbol{\beta}) \; + (1-y)\log(1-\sigma\left(\boldsymbol{x}_i^T \boldsymbol{\beta}\right)) \;\right] + \lambda ||\boldsymbol{\beta}||_2^2$
- We can alternatively use the Lasso or Elastic Net penalizations.

- Same intuition applies! Nonlinearities complicate the analytic form of the math.

BUILDING A DEEPER MATHEMATICAL UNDERSTANDING

Mathematical Analysis

- Here we want to understand why different penalties give such different properties
- Previously talked about the *geometry*, now we will go into the math a little bit
- Will only go through this superficially, recommend that you read on your own

Lasso Regression

- What about the Lasso penalty?
- Consider the simplest case first (note, not necessary simple!):
  - $f(\beta) = \frac{1}{2}(y-\beta)^2 + \lambda|\beta|$
- Where is this minimized?

- We can’t just take the derivative…
- Solution is given by a ”soft-thresholding,” where values near zero are set to 0.

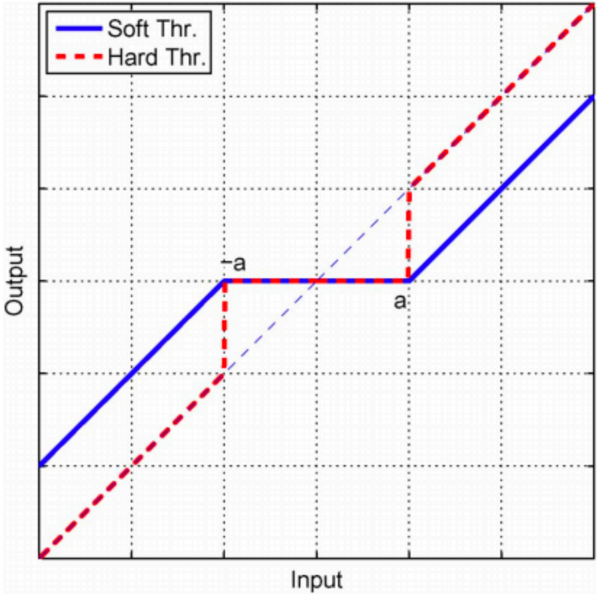


Fig. 1. ***The soft and hard shrinkage curves.***

Soft vs. Hard Thresholding

- $f(\beta) = \frac{1}{2}(y - \beta)^2 + \lambda|\beta|$
- Mathematically, soft thresholding is given by:
- $g(\beta) = \begin{cases} y - \lambda, & y > \lambda \\ 0, & -\lambda \leq y \leq \lambda \\ y + \lambda, & y > -\lambda \end{cases}$
- To understand what's happening in the soft vs hard thresholding, we are visualizing it here.
- $g(\beta) = \arg\min_{\beta} f(\beta)$

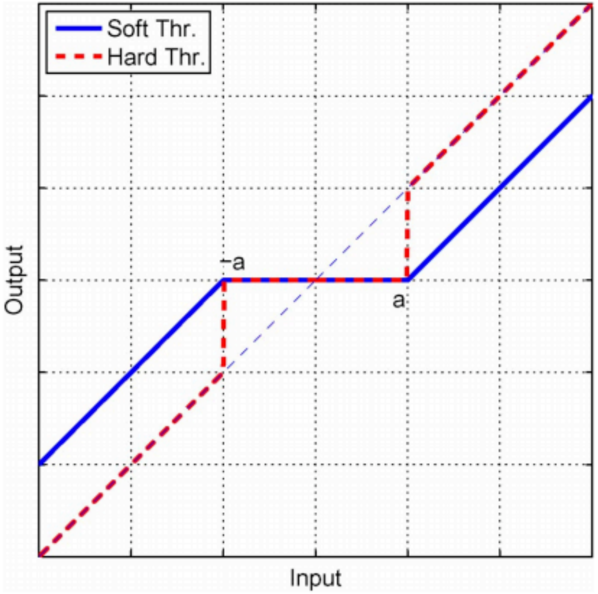


Fig. 1. ***The soft and hard shrinkage curves.***

With uncorrelated features:

- Let features be “uncorrelated” such that $\boldsymbol{X}^T\boldsymbol{X} = \boldsymbol{I}$, and then let $\hat{\beta}_j$ be the estimate of the *j*th parameter from ordinary least squares (i.e. linear regression).
- Under this formulation, previous formulations are *analytic* in how they change the estimate of the coefficients.

Estimator	Formula
Best subset (size M)	$\hat{\beta}_j \cdot I(\lvert\hat{\beta}_j\rvert \geq \lvert\hat{\beta}_{(M)}\rvert)$
Ridge	$\hat{\beta}_j/(1+\lambda)$
Lasso	$\text{sign}(\hat{\beta}_j)(\lvert\hat{\beta}_j\rvert - \lambda)_+$

- How do we come to these forms?

What is a “Langrangian Dual?”

- We won’t be using the Langrangian Dual to derive things in this class, but it is often thrown around in the literature, so I wanted to introduce it.
- Essentially, if we have a constrained optimization problem, such as
- $\boldsymbol{\beta}^* = \arg\min_{\boldsymbol{\beta}\in\mathbb{R}^p} ||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}||_2^2$ subject to $||\boldsymbol{\beta}||_2^2 < \tau$,
- it will yield the same solution for $\boldsymbol{\beta}^*$ as
- $\boldsymbol{\beta}^* = \arg\min_{\boldsymbol{\beta}\in\mathbb{R}^p} ||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}||_2^2 + \lambda||\boldsymbol{\beta}||_2^2$,
- for some pair of values τ and λ.
- I.E. as λ increases the τ that gives the equivalent answer decreases.

Why does regularization “restrict” complexity?

- Let’s consider the ridge regression penalty.

- When we say that we add a penalization term $\lambda||\beta||_2^2$, we equivalently have that the penalized likelihood solution has:
  - $||\boldsymbol{\beta}||_2^2 \leq \tau$.
- Ergo, as the penalization increases, we are limiting ourselves to a smaller set.