Ridge Regression: 
Regulating overfitting when using many features

Training, true, & test error vs. model complexity

Overfitting if:
You learn a model w/ param w, but there exists a model w/ param w' such that
1) true error(w') < true error(w)
2) train error(w) < train error(w')
Bias-variance tradeoff

\[ \text{error} = \text{bias}^2 + \text{variance} + \text{noise} \]

**Simple Models**
- **Bias**: High
- **Variance**: Low

**Complex Models**
- **Bias**: Low
- **Variance**: High

Error vs. amount of data

For fixed model complexity:
- Too little data points, error (bias) is high.
- Too many data points, error (noise) is high.
- In the limit, true error = train error.
- In the limit, error will flatten out to how well model can fit future data.

Error vs. amount of data in training set
Summary of assessing performance

What you can do now...

- Describe what a loss function is and give examples
- Contrast training and test error
- Compute training and test error given a loss function
- Discuss issue of assessing performance on training set
- Describe tradeoffs in forming training/test splits
- List and interpret the 3 sources of avg. prediction error
  - Irreducible error, bias, and variance
Overfitting of polynomial regression

Flexibility of high-order polynomials

\[ y_i = w_0 + w_1 x_i + w_2 x_i^2 + \ldots + w_p x_i^p + \epsilon_i \]
Symptom of overfitting

Often, overfitting associated with very large estimated parameters \( \hat{w} \)

Overfitting of linear regression models more generically
Overfitting with many features

Not unique to polynomial regression, but also if lots of inputs (d large)

Or, generically, lots of features (D large)

\[ y_i = \sum_{j=0}^{D} w_j h_j(x_i) + \varepsilon_i \]

- Square feet
- # bathrooms
- # bedrooms
- Lot size
- Year built
- ...

How does # of observations influence overfitting?

Few observations (N small) → rapidly overfit as model complexity increases

Many observations (N very large) → harder to overfit

\[ f_{\hat{w}} \]
How does # of inputs influence overfitting?

1 input (e.g., sq.ft.):
Data must include representative examples of all possible (sq.ft., $) pairs to avoid overfitting

How does # of inputs influence overfitting?

d inputs (e.g., sq.ft., #bath, #bed, lot size, year,...):
Data must include examples of all possible (sq.ft., #bath, #bed, lot size, year,...., $) combos to avoid overfitting
Adding term to cost-of-fit to prefer small coefficients
Desired total cost format

Want to balance:
1. How well function fits data
2. Magnitude of coefficients

Total cost = measure of fit + measure of magnitude of coefficients

small # = good fit to training data
small # = not overfit

Measure of fit to training data

\[
\text{RSS}(w) = \sum_{i=1}^{N} (y_i - h(x_i)w^T)^2 = \sum_{i=1}^{N} (y_i - \hat{y}_i(w))^2
\]
Measure of magnitude of regression coefficient

What summary # is indicative of size of regression coefficients?

- Sum?
- Sum of absolute value?
- Sum of squares ($L_2$ norm)

Consider specific total cost

Total cost = measure of fit + measure of magnitude of coefficients
Consider specific total cost

Total cost = measure of fit + measure of magnitude of coefficients

\[ \text{RSS}(\mathbf{w}) + \|\mathbf{w}\|^2 \]

Consider resulting objective

What if \( \hat{\mathbf{w}} \) selected to minimize

\[ \text{RSS}(\mathbf{w}) + \lambda \|\mathbf{w}\|^2 \]

If \( \lambda = 0 \):

If \( \lambda = \infty \):

If \( \lambda \) in between:

\[ \text{tuning parameter} = \text{balance of fit and magnitude} \]
Consider resulting objective

What if $\hat{w}$ selected to minimize

$$\text{RSS}(w) + \lambda \|w\|^2_2$$

(tuning parameter = balance of fit and magnitude)

Ridge regression
(a.k.a $L_2$ regularization)

---

Bias-variance tradeoff

Large $\lambda$:
- high bias, low variance
- (e.g., $\hat{w} = 0$ for $\lambda = \infty$)

Small $\lambda$:
- low bias, high variance
- (e.g., standard least squares (RSS) fit of high-order polynomial for $\lambda = 0$)

In essence, $\lambda$ controls model complexity
Revisit polynomial fit demo

What happens if we refit our high-order polynomial, but now using ridge regression?

Will consider a few settings of $\lambda$ ...
How to choose $\lambda$

The regression/ML workflow

1. Model selection
   Need to choose tuning parameters $\lambda$ controlling model complexity

2. Model assessment
   Having selected a model, assess generalization error
Hypothetical implementation

1. Model selection
   For each considered $\lambda$:
   i. Estimate parameters $\hat{w}_\lambda$ on training data
   ii. Assess performance of $\hat{w}_\lambda$ on test data
   iii. Choose $\lambda^*$ to be $\lambda$ with **lowest test error**

2. Model assessment
   Compute test error of $\hat{w}_{\lambda^*}$ (fitted model for selected $\lambda^*$) to approx. true error

**Issue:** Just like fitting $\hat{w}$ and assessing its performance both on training data
- $\lambda^*$ was selected to minimize test error (i.e., $\lambda^*$ was fit on test data)
- If test data is not representative of the whole world, then $\hat{w}_{\lambda^*}$ will typically perform worse than test error indicates
Practical implementation

Solution: Create two “test” sets!

1. Select $\lambda^*$ such that $\hat{w}_{\lambda^*}$ minimizes error on validation set
2. Approximate true error of $\hat{w}_{\lambda^*}$ using test set
Typical splits

<table>
<thead>
<tr>
<th>Training set</th>
<th>Validation set</th>
<th>Test set</th>
</tr>
</thead>
<tbody>
<tr>
<td>80%</td>
<td>10%</td>
<td>10%</td>
</tr>
<tr>
<td>50%</td>
<td>25%</td>
<td>25%</td>
</tr>
</tbody>
</table>

How to handle the intercept

PRACTICALITIES
Recall multiple regression model

Model:
\[ y_i = w_0 h_0(x_i) + w_1 h_1(x_i) + \cdots + w_D h_D(x_i) + \varepsilon_i \]
\[ = \sum_{j=0}^{D} w_j h_j(x_i) + \varepsilon_i \]

feature 1 = \( h_0(x) \)... often 1 (constant)
feature 2 = \( h_1(x) \)... e.g., \( x[1] \)
feature 3 = \( h_2(x) \)... e.g., \( x[2] \)
...
feature \( D+1 \) = \( h_D(x) \)... e.g., \( x[d] \)

Do we penalize intercept?
Standard ridge regression cost:
\[ \text{RSS}(w) + \lambda \|w\|_2^2 \]
Encourages intercept \( w_0 \) to also be small
Do we want a small intercept?
Conceptually, not indicative of overfitting...
Option 1: Don’t penalize intercept

Modified ridge regression cost:

$$RSS(w_0, w_{\text{rest}}) + \lambda ||w_{\text{rest}}||_2^2$$

Option 2: Center data first

If data are first centered about 0, then favoring small intercept not so worrisome

Step 1: Transform $y$ to have 0 mean
Step 2: Run ridge regression as normal
   (closed-form or gradient algorithms)
Normalizing features

Scale training columns \textbf{(not rows!)} as:

\[
h_j(x_k) = \frac{h_j(x_k)}{\sqrt{\sum_{i=1}^{N} h_j(x_i)^2}}
\]

Apply same training scale factors to test data:

\[
h_j(x_k) = \frac{h_j(x_k)}{\sqrt{\sum_{i=1}^{N} h_j(x_i)^2}}
\]

Normalizer: \[z_j\]

\[
\text{apply to test point}
\]

\[
\text{summing over training points}
\]
Summary for ridge regression

What you can do now...

- Describe what happens to magnitude of estimated coefficients when model is overfit
- Motivate form of ridge regression cost function
- Describe what happens to estimated coefficients of ridge regression as tuning parameter $\lambda$ is varied
- Interpret coefficient path plot
- Use a validation set to select the ridge regression tuning parameter $\lambda$
- Handle intercept and scale of features with care
Lasso Regression:
Regularization for feature selection

Feature selection task
Why might you want to perform feature selection?

Efficiency:
- If \( \text{size}(w) = 100B \), each prediction is expensive
- If \( w \) is sparse, computation only depends on \# of non-zeros

\[
\hat{y}_i = \sum_{\hat{w}_j \neq 0} \hat{w}_j h_j(x_i)
\]

Interpretability:
- Which features are relevant for prediction?

Sparsity: Housing application

<table>
<thead>
<tr>
<th>Feature</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lot size</td>
<td>Dishwasher</td>
</tr>
<tr>
<td>Single Family</td>
<td>Garbage disposal</td>
</tr>
<tr>
<td>Year built</td>
<td>Microwave</td>
</tr>
<tr>
<td>Last sold price</td>
<td>Range / Oven</td>
</tr>
<tr>
<td>Last sale price/sqft</td>
<td>Refrigerator</td>
</tr>
<tr>
<td>Finished sqft</td>
<td>Washer</td>
</tr>
<tr>
<td>Unfinished sqft</td>
<td>Dryer</td>
</tr>
<tr>
<td>Finished basement sqft</td>
<td>Laundry location</td>
</tr>
<tr>
<td># floors</td>
<td>Heating type</td>
</tr>
<tr>
<td>Flooring types</td>
<td>Jetted Tub</td>
</tr>
<tr>
<td>Parking type</td>
<td>Deck</td>
</tr>
<tr>
<td>Parking amount</td>
<td>Fenced Yard</td>
</tr>
<tr>
<td>Cooling</td>
<td>Lawn</td>
</tr>
<tr>
<td>Heating</td>
<td>Garden</td>
</tr>
<tr>
<td>Exterior materials</td>
<td>Sprinkler System</td>
</tr>
<tr>
<td>Roof type</td>
<td></td>
</tr>
<tr>
<td>Structure style</td>
<td></td>
</tr>
</tbody>
</table>
Sparsity: Reading your mind

Activity in which brain regions can predict happiness?

Option 1: All subsets
Find best model of size: 0

- # bedrooms
- # bathrooms
- sq.ft. living
- sq.ft. lot
- floors
- year built
- year renovated
- waterfront

# of features

Find best model of size: 1

- # bedrooms
- # bathrooms
- sq.ft. living
- sq.ft. lot
- floors
- year built
- year renovated
- waterfront

# of features
Find best model of size: 1

- # bedrooms
- # bathrooms
- sq.ft. living
- sq.ft. lot
- floors
- year built
- year renovated
- waterfront

# of features

RSS(\(\hat{w}\))

0 1
Find best model of size: 1

- # bedrooms
- # bathrooms
- sq.ft. living
- sq.ft. lot
- floors
- year built
- year renovated
- waterfront
Find best model of size: 1

- # bedrooms
- # bathrooms
- sq.ft. living
- sq.ft. lot
- floors
- year built
- year renovated
- waterfront

RSS(\hat{w})

# of features
Find best model of size: 1

- # bedrooms
- # bathrooms
- sq.ft. living
- sq.ft. lot
- floors
- year built
- year renovated
- waterfront

Find best model of size: 1

- # bedrooms
- # bathrooms
- sq.ft. living
- sq.ft. lot
- floors
- year built
- year renovated
- waterfront
Find best model of size: 2

- # bedrooms
- # bathrooms
- sq.ft. living
- sq.ft. lot
- floors
- year built
- year renovated
- waterfront

Note: Not necessarily nested!

- # bedrooms
- # bathrooms
- sq.ft. living
- sq.ft. lot
- floors
- year built
- year renovated
- waterfront
Note: Not necessarily nested!

Find best model of size: 3
Find best model of size: 4

- # bedrooms
- # bathrooms
- sq.ft. living
- sq.ft. lot
- floors
- year built
- year renovated
- waterfront

Find best model of size: 5

- # bedrooms
- # bathrooms
- sq.ft. living
- sq.ft. lot
- floors
- year built
- year renovated
- waterfront
Find best model of size: 6

Find best model of size: 7
Choosing model complexity?

Option 1: Assess on validation set

Option 2: Cross validation

Option 3+: Other metrics for penalizing model complexity like BIC...
Complexity of “all subsets”

How many models were evaluated?
- each indexed by features included

\[ y_i = \varepsilon_i \]
\[ y_i = w_0 h_0(x) + \varepsilon_i \]
\[ y_i = w_1 h_1(x) + \varepsilon_i \]
\[ \vdots \]
\[ y_i = w_0 h_0(x) + w_1 h_1(x) + \varepsilon_i \]
\[ \vdots \]
\[ y_i = w_0 h_0(x) + w_1 h_1(x) + \cdots + w_D h_D(x) + \varepsilon_i \]

Typically, computationally infeasible

\[ 2^8 = 256 \]
\[ 2^{10} = 1,073,741,824 \]
\[ 2^{1000} = 1.071509 \times 10^{301} \]
\[ 2^{1000} = \text{HUGE!!!!!!} \]

Option 2: Greedy algorithms
Forward stepwise algorithm

1. Pick a dictionary of features \{h_0(x),\ldots,h_D(x)\} 
   - e.g., polynomials for linear regression
2. Greedy heuristic:
   i. Start with empty set of features \(F_0 = \emptyset\) 
      (or simple set, like just \(h_0(x) = 1 \rightarrow y_i = w_0 + \epsilon_i\))
   ii. Fit model using current feature set \(F_t\) to get \(\hat{w}^{(t)}\)
   iii. Select next best feature \(h_j(x)\)
      - e.g., \(h_j(x)\) resulting in lowest training error when learning with \(F_t + \{h_j(x)\}\)
   iv. Set \(F_{t+1} \leftarrow F_t + \{h_j(x)\}\)
   v. Recurse
Visualizing greedy procedure

- # bedrooms
- # bathrooms
- sq.ft. living
- sq.ft. lot
- floors
- year built
- year renovated
- waterfront
Visualizing greedy procedure

RSS($\hat{w}$) vs. # of features

- # bedrooms
- # bathrooms
- sq.ft. living
- sq.ft. lot
- floors
- year built
- year renovated
- waterfront
Visualizing greedy procedure

RSS($\mathbf{w}$) vs. # of features

- # bedrooms
- # bathrooms
- sq.ft. living
- sq.ft. lot
- floors
- year built
- year renovated
- waterfront

error never increases
+ solutions eventually meet
When do we stop?

When *training error* is low enough?  
No!

When *test error* is low enough?  
No!

*Use validation set or cross validation!*  

Complexity of forward stepwise

How many models were evaluated?
- 1<sup>st</sup> step, *D* models
- 2<sup>nd</sup> step, *D-1* models (add 1 feature out of *D-1* possible)
- 3<sup>rd</sup> step, *D-2* models (add 1 feature out of *D-2* possible)
- ...

How many steps?  
- Depends
- At most *D* steps (to full model)

\[ O(D^2) \ll 2^D \]  
for large *D*
Other greedy algorithms

Instead of starting from simple model and always growing...

**Backward stepwise:**
Start with full model and iteratively remove features least useful to fit

**Combining forward and backward steps:**
In forward algorithm, insert steps to remove features no longer as important

*Lots of other variants, too.*

---

**Option 3: Regularize**
Ridge regression: $L_2$ regularized regression

Total cost = measure of fit + $\lambda$ measure of magnitude of coefficients

$\text{RSS}(w) = \|w\|_2^2$ if $w_0^2 + \ldots + w_D^2$

Encourages small weights but not exactly 0

Coefficient path – ridge

![Coefficient path graph]

- bedrooms
- bathrooms
- sqft living
- sqft lot
- floors
- yr built
- yr renovation
- waterfront
Using regularization for feature selection

Instead of searching over a discrete set of solutions, can we use regularization?

- Start with full model (all possible features)
- “Shrink” some coefficients exactly to 0
  - i.e., knock out certain features
- Non-zero coefficients indicate “selected” features

Thresholding ridge coefficients?

Why don’t we just set small ridge coefficients to 0?
Thresholding ridge coefficients?

Selected features for a given threshold value

Let’s look at two related features...

Nothing measuring bathrooms was included!
Thresholding ridge coefficients?

If only one of the features had been included...

Would have included bathrooms in selected model

Can regularization lead directly to sparsity?
Try this cost instead of ridge...

Total cost = \[ \text{measure of fit} + \lambda \text{ measure of magnitude of coefficients} \]

\[ \text{RSS}(w) + \lambda ||w||_1 = |w_0| + \ldots + |w_D| \]

Leads to \textit{sparse} solutions!

\textbf{Lasso regression} (a.k.a. \textit{L}_1 \textit{ regularized regression})

\textbf{Lasso regression: } \textit{L}_1 \textit{ regularized regression}

Just like ridge regression, solution is governed by a continuous parameter \( \lambda \)

\[ \text{RSS}(w) + \lambda ||w||_1 \]

tuning parameter = balance of fit and \textit{sparsity}

If \( \lambda = 0 \): 

If \( \lambda = \infty \): 

If \( \lambda \) in between:
Coefficient path – ridge

Coefficient path – lasso
Revisit polynomial fit demo

What happens if we refit our high-order polynomial, but now using lasso regression?

Will consider a few settings of $\lambda$ ...

How to choose $\lambda$:
Cross validation
If sufficient amount of data...

- **Training set**
- **Validation set**
- **Test set**

- Fit $\hat{w}_\lambda$
- Test performance of $\hat{w}_\lambda$ to select $\lambda^*$
- Assess generalization error of $\hat{w}_{\lambda^*}$

Start with smallish dataset

- All data
Still form test set and hold out

Rest of data

Test set

How do we use the other data?

Rest of data

use for both training and validation, but not so naively
Recall naïve approach

Is validation set enough to compare performance of $\hat{w}_\lambda$ across $\lambda$ values?

No

Choosing the validation set

Didn’t have to use the last data points tabulated to form validation set

Can use any data subset
Choosing the validation set

Which subset should I use?

ALL!

average performance over all choices

K-fold cross validation

Rest of data

Preprocessing: Randomly assign data to K groups

(use same split of data for all other steps)
K-fold cross validation

For k=1,...,K
1. Estimate $\hat{w}_\lambda^{(k)}$ on the training blocks
2. Compute error on validation block: $\text{error}_k(\lambda)$
For $k=1,\ldots,K$

1. Estimate $\hat{w}_\lambda^{(k)}$ on the training blocks
2. Compute error on validation block: $\text{error}_k(\lambda)$
K-fold cross validation

For \( k=1,\ldots,K \)
1. Estimate \( \hat{w}_\lambda^{(k)} \) on the training blocks
2. Compute error on validation block: \( \text{error}_k(\lambda) \)

Compute average error: \( \text{CV}(\lambda) = \frac{1}{K} \sum_{k=1}^{K} \text{error}_k(\lambda) \)

Repeat procedure for each choice of \( \lambda \)
Choose \( \lambda^* \) to minimize \( \text{CV}(\lambda) \)
What value of $K$?

Formally, the best approximation occurs for validation sets of size 1 ($K=N$)

Computationally intensive
- requires computing $N$ fits of model per $\lambda$

Typically, $K=5$ or 10

Choosing $\lambda$ via cross validation for lasso

Cross validation is choosing the $\lambda$ that provides best predictive accuracy

Tends to favor less sparse solutions, and thus smaller $\lambda$, than optimal choice for feature selection

c.f., "Machine Learning: A Probabilistic Perspective", Murphy, 2012 for further discussion
Debiasing lasso

Lasso shrinks coefficients relative to LS solution → more bias, less variance

Can reduce bias as follows:
1. Run lasso to select features
2. Run least squares regression with only selected features

“Relevant” features no longer shrunk relative to LS fit of same reduced model
Issues with standard lasso objective

1. With group of highly correlated features, lasso tends to select amongst them arbitrarily
   - Often prefer to select all together

2. Often, empirically ridge has better predictive performance than lasso, but lasso leads to sparser solution

Elastic net aims to address these issues
   - hybrid between lasso and ridge regression
   - uses $L_1$ and $L_2$ penalties

See Zou & Hastie ‘05 for further discussion

Summary for feature selection and lasso regression
Impact of feature selection and lasso

Lasso has changed machine learning, statistics, & electrical engineering

But, for feature selection in general, be careful about interpreting selected features
- selection only considers features included
- sensitive to correlations between features
- result depends on algorithm used
- there are theoretical guarantees for lasso under certain conditions

What you can do now...

- Describe "all subsets" and greedy variants for feature selection
- Analyze computational costs of these algorithms
- Formulate lasso objective
- Describe what happens to estimated lasso coefficients as tuning parameter $\lambda$ is varied
- Interpret lasso coefficient path plot
- Contrast ridge and lasso regression
- Estimate lasso regression parameters using an iterative coordinate descent algorithm
- Implement K-fold cross validation to select lasso tuning parameter $\lambda$