Warm up

```python
# generate some nonsense data for an example
X = np.random.randn(n,d)
y = np.random.randn(n)
```

1 float in NumPy $= 8$ bytes
$10^6 \approx 2^{20}$ bytes $= 1$ MB
$10^9 \approx 2^{30}$ bytes $= 1$ GB

```python
# generate the random features
G = np.random.randn(p, d)*np.sqrt(.1)
b = np.random.rand(p)*2*np.pi
```

```python
H = np.dot(X, G.T) + b.T
HTH = np.dot(H.T, H)
HTy = np.dot(H.T, y)
```

```python
# construct HTH
HTH = np.zeros((p,p))
HTy = np.zeros(p)
for i in range(n):
    hi = np.dot(X[i,:], G.T)+b
    HTH += np.outer(hi, hi)
    HTy += y[i]*hi
    if i % 1000==0: print(i)
```

```python
# construct HTH
HTH = np.zeros((p,p))
HTy = np.zeros(p)
block = p
for i in range(int(np.ceil(n/block))+1):
    Hi = np.dot(X[i*block:min(n,(i+1)*block),:], G.T)+b
    HTH += np.dot(Hi.T, Hi)
    HTy += np.dot(Hi.T, y[i*block:min(n,(i+1)*block)])
```

```python
w = np.linalg.solve(HTH + lam*np.eye(p), HTy)
```

For each block compute the memory required in terms of n, p, d.

If d << p << n, what is the most memory efficient program (blue, green, red)?

If you have unlimited memory, what do you think is the fastest program?

Regularization

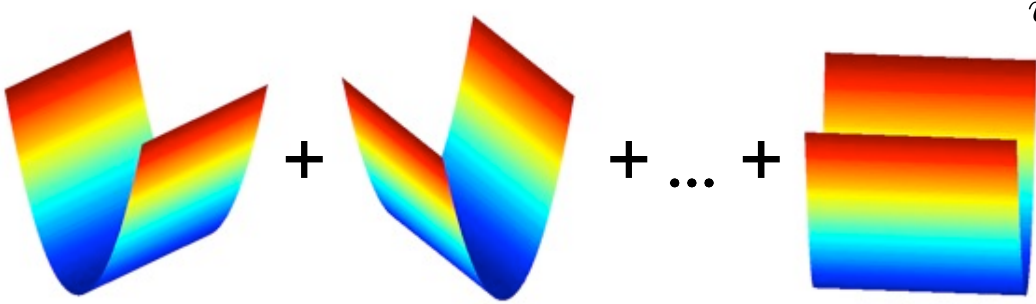
Machine Learning – CSE546
Kevin Jamieson
University of Washington

April 15, 2019

Ridge Regression

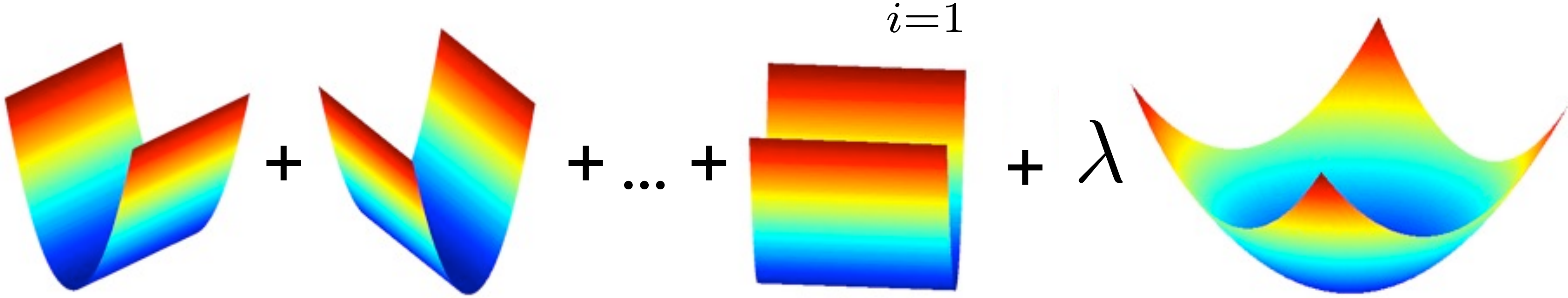
- Old Least squares objective:

$$\widehat{w}_{LS} = \arg\min_w \sum_{i=1}^{n} \left(y_i - x_i^T w\right)^2$$



- Ridge Regression objective:

$$\widehat{w}_{ridge} = \arg\min_w \sum_{i=1}^{n} \left(y_i - x_i^T w\right)^2 + \lambda ||w||_2^2$$



Minimizing the Ridge Regression Objective

$$\widehat{w}_{ridge} = \arg\min_w \sum_{i=1}^n \left(y_i - x_i^T w\right)^2 + \lambda ||w||_2^2$$

$$0 = \nabla_w \left(\sum_{i=1}^n \left(y_i - x_i^T w\right)^2 + \lambda ||w||_2^2 \right)$$

$$= -\sum_{i=1}^n 2x_i \left(y_i - x_i^T w\right) + 2\lambda w$$

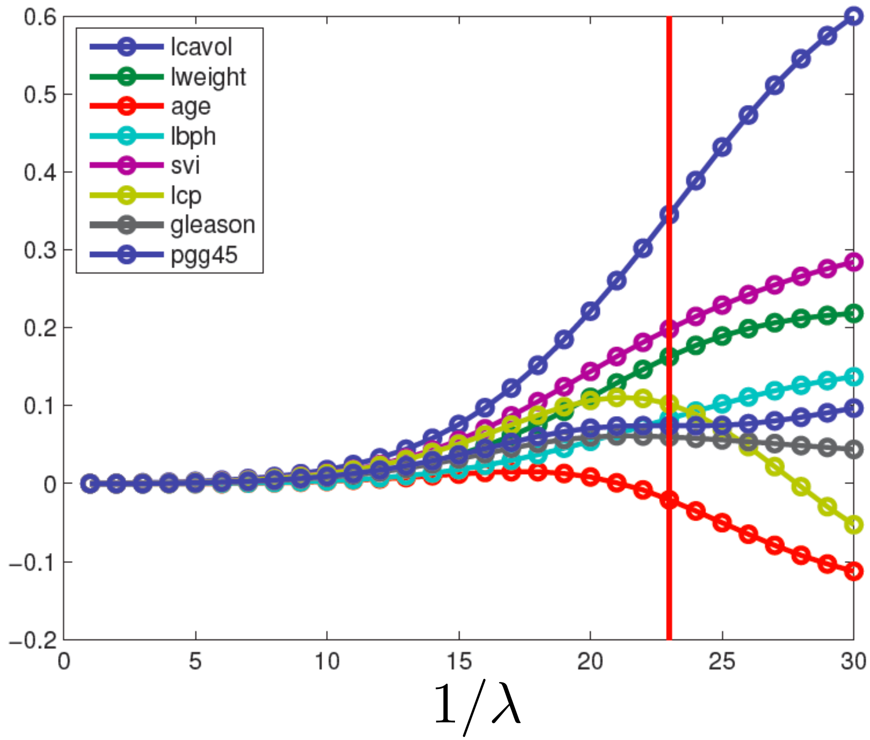
$$= 2\left(\sum_{i=1}^n x_i y_i\right) + 2\left(\sum_{i=1}^n x_i x_i^T + \lambda I\right) w$$

$$\widehat{w}_{ridge} = \left(\sum_{i=1}^n x_i x_i^T + \lambda I\right)^{-1} \left(\sum_{i=1}^n x_i y_i\right)$$

$$= (\mathbf{X}^T\mathbf{X} + \lambda I)^{-1}\mathbf{X}^T\mathbf{y}$$

Ridge Coefficient Path

$\mathbf{X}^T\mathbf{X}$ in general



From
Kevin Murphy
textbook

- Typical approach: select λ using cross validation, up next

Bias-Variance Properties

$$\widehat{w}_{ridge} = (\mathbf{X}^T\mathbf{X} + \lambda I)^{-1}\mathbf{X}^T\mathbf{y}$$

- ***Assume:*** $\mathbf{X}^T\mathbf{X} = nI$ ***and*** $\mathbf{y} = \mathbf{X}w + \boldsymbol{\epsilon}$ $\quad \epsilon \sim \mathcal{N}(0, \sigma^2 I)$

If $x \in \mathbb{R}^d$ and $Y \sim \mathcal{N}(x^T w, \sigma^2)$, what is $\mathbb{E}_{Y|x,\text{train}}[(Y - x^T\widehat{w}_{ridge})^2 | X = x]$?

$$\begin{aligned}
&\mathbb{E}_{Y|X,\mathcal{D}}[(Y - x^T\widehat{w}_{ridge})^2|X = x] \\
&\quad = \mathbb{E}_{Y|X}[(Y - \mathbb{E}_{Y|X}[Y|X=x])^2|X=x] + \mathbb{E}_{\mathcal{D}}[(\mathbb{E}_{Y|X}[Y|X=x] - x^T\widehat{w}_{ridge})^2] \\
&\quad = \mathbb{E}_{Y|X}[(Y - x^T w)^2|X=x] + \mathbb{E}_{\mathcal{D}}[(x^T w - x^T\widehat{w}_{ridge})^2] \\
&\quad = \sigma^2 + (x^T w - \mathbb{E}_{\mathcal{D}}[x^T\widehat{w}_{ridge}])^2 + \mathbb{E}_{\mathcal{D}}[(\mathbb{E}_{\mathcal{D}}[x^T\widehat{w}_{ridge}] - x^T\widehat{w}_{ridge})^2]
\end{aligned}$$

$$= \underbrace{\sigma^2}_{\text{Irreduc. Error}} + \underbrace{\frac{\lambda^2}{(n+\lambda)^2}(w^T x)^2}_{\text{Bias-squared}} + \underbrace{\frac{d\sigma^2 n}{(n+\lambda)^2}\|x\|_2^2}_{\text{Variance}}$$

(verify at home)

Ridge Regression: Effect of Regularization

$$\mathcal{D} \stackrel{i.i.d.}{\sim} P_{XY} \qquad \widehat{w}_{\mathcal{D},ridge}^{(\lambda)} = \arg\min_w \frac{1}{|\mathcal{D}|} \sum_{(x_i,y_i)\in\mathcal{D}} (y_i - x_i^T w)^2 + \lambda ||w||_2^2$$

TRAIN error:

$$\frac{1}{|\mathcal{D}|} \sum_{(x_i,y_i)\in\mathcal{D}} (y_i - x_i^T \widehat{w}_{\mathcal{D},ridge}^{(\lambda)})^2$$

Ridge Regression: Effect of Regularization

$$\mathcal{D} \overset{i.i.d.}{\sim} P_{XY} \qquad \widehat{w}_{\mathcal{D},ridge}^{(\lambda)} = \arg\min_w \frac{1}{|\mathcal{D}|} \sum_{(x_i,y_i)\in\mathcal{D}} (y_i - x_i^T w)^2 + \lambda ||w||_2^2$$

TRAIN error:

$$\frac{1}{|\mathcal{D}|} \sum_{(x_i,y_i)\in\mathcal{D}} (y_i - x_i^T \widehat{w}_{\mathcal{D},ridge}^{(\lambda)})^2$$

TEST error:

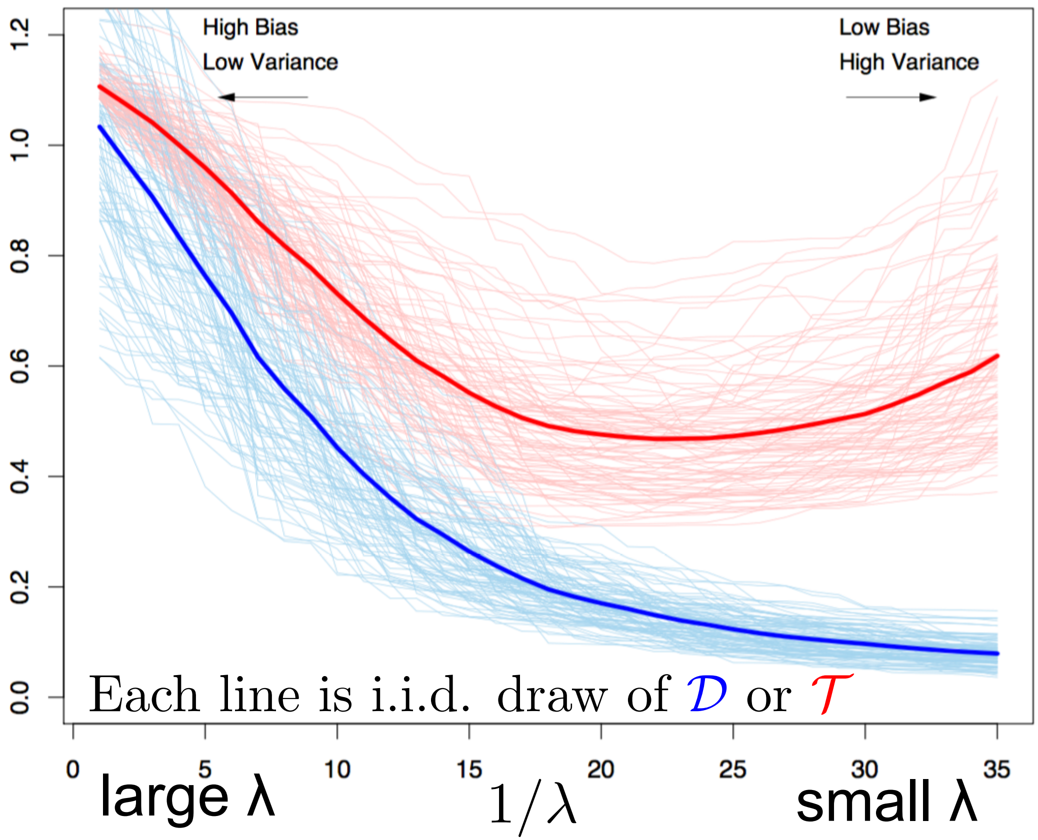
$$\mathcal{T} \overset{i.i.d.}{\sim} P_{XY}$$

$$\frac{1}{|\mathcal{T}|} \sum_{(x_i,y_i)\in\mathcal{D}} (y_i - x_i^T \widehat{w}_{\mathcal{D},ridge}^{(\lambda)})^2$$

$$\boxed{\text{Important: } \mathcal{D} \cap \mathcal{T} = \emptyset}$$

Ridge Regression: Effect of Regularization

$$\mathcal{D} \overset{i.i.d.}{\sim} P_{XY} \qquad \widehat{w}^{(\lambda)}_{\mathcal{D},ridge} = \arg\min_w \frac{1}{|\mathcal{D}|} \sum_{(x_i,y_i)\in\mathcal{D}} (y_i - x_i^T w)^2 + \lambda ||w||_2^2$$



TRAIN error:

$$\frac{1}{|\mathcal{D}|} \sum_{(x_i,y_i)\in\mathcal{D}} (y_i - x_i^T \widehat{w}^{(\lambda)}_{\mathcal{D},ridge})^2$$

TEST error:

$$\mathcal{T} \overset{i.i.d.}{\sim} P_{XY}$$

$$\frac{1}{|\mathcal{T}|} \sum_{(x_i,y_i)\in\mathcal{D}} (y_i - x_i^T \widehat{w}^{(\lambda)}_{\mathcal{D},ridge})^2$$

$$\boxed{\text{Important: } \mathcal{D} \cap \mathcal{T} = \emptyset}$$

Cross-Validation

Machine Learning – CSE546
Kevin Jamieson
University of Washington

October 9, 2016

How… How… How???????

- *How do we pick the regularization constant λ…*
- *How do we pick the number of basis functions…*

- We could use the test data, but…

How… How… How???????

- *How do we pick the regularization constant λ…*
- *How do we pick the number of basis functions…*

- We could use the test data, but…

- Never ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever ever train on the test data

(LOO) Leave-one-out cross validation

- Consider a **validation set with 1 example**:
  - D – training data
  - $D\backslash j$ – training data with j th data point $(\mathbf{x}_j, \mathbf{y}_j)$ moved to validation set
- **Learn classifier $f_{D\backslash j}$ with $D\backslash j$ dataset**
- **Estimate true error** as squared error on predicting $\mathbf{y}_j$:
  - Unbiased estimate of $\text{error}_{true}(f_{D\backslash j})$!

(LOO) Leave-one-out cross validation

- Consider a **validation set with 1 example**:
  - *D* – training data
  - *D*\j – training data with *j* th data point ($\mathbf{x}_j$,$\mathbf{y}_j$) moved to validation set
- **Learn classifier $f_{D\backslash j}$ with *D*\j dataset**
- **Estimate true error** as squared error on predicting $\mathbf{y}_j$:
  - Unbiased estimate of $\text{error}_{\text{true}}(f_{D\backslash j})$!
- **LOO cross validation**: Average over all data points *j*:
  - **For each data point you leave out, learn a new classifier $f_{D\backslash j}$**
  - **Estimate error** as:

$$\text{error}_{LOO} = \frac{1}{n} \sum_{j=1}^{n} (y_j - f_{\mathcal{D}\backslash j}(x_j))^2$$

LOO cross validation is (almost) unbiased estimate of true error of h_D!

- When computing **LOOCV error, we only use *N-1* data points**
  - So it's not estimate of true error of learning with *N* data points
  - Usually pessimistic, though – learning with less data typically gives worse answer

- **LOO is almost unbiased! Use LOO error for model selection!!!**
  - **E.g., picking λ**

Computational cost of LOO

- Suppose you have 100,000 data points
- You implemented a great version of your learning algorithm
  - Learns in only 1 second
- Computing LOO will take about 1 day!!!
  -

Use *k*-fold cross validation

- Randomly **divide training data into *k* equal parts**
  - $D_1, \ldots, D_k$
- For each i
  - **Learn classifier $f_{D \backslash Di}$ using data point not in D_i**
  - **Estimate error of $f_{D \backslash Di}$ on validation set D_i:**

1	2	3	4	5
Train	Train	Validation	Train	Train

- $$\text{error}_{\mathcal{D}_i} = \frac{1}{|\mathcal{D}_i|} \sum_{(x_j, y_j) \in \mathcal{D}_i} (y_j - f_{\mathcal{D} \backslash \mathcal{D}_i}(x_j))^2$$

Use *k*-fold cross validation

- Randomly **divide training data into *k* equal parts**
  - $D_1,\ldots,D_k$
- For each *i*
  - **Learn classifier $f_{D \backslash D_i}$ using data point not in D_i**
  - **Estimate error of $f_{D \backslash D_i}$ on validation set D_i:**

1	2	3	4	5
Train	Train	Validation	Train	Train

$$\text{error}_{\mathcal{D}_i} = \frac{1}{|\mathcal{D}_i|} \sum_{(x_j,y_j)\in\mathcal{D}_i} (y_j - f_{\mathcal{D}\backslash\mathcal{D}_i}(x_j))^2$$

- ***k*-fold cross validation error is average** over data splits:

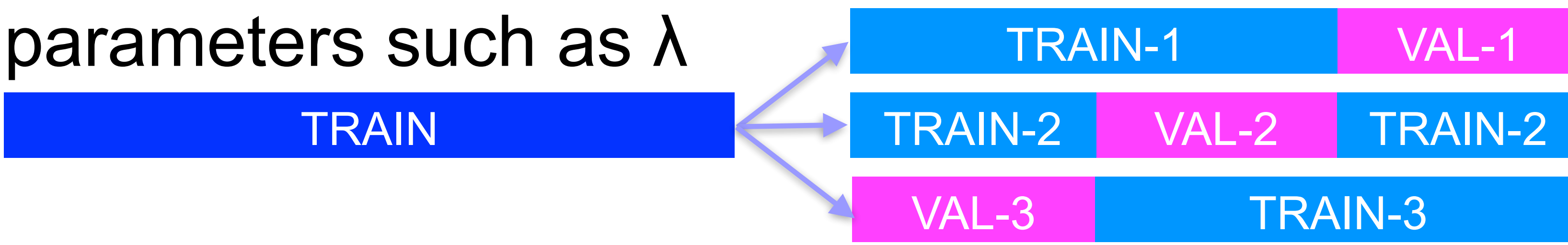
$$error_{k-fold} = \frac{1}{k}\sum_{i=1}^{k} error_{\mathcal{D}_i}$$

- *k*-fold cross validation properties:
  - **Much faster to compute** than LOO
  - **More (pessimistically) biased** – using much less data, only *n(k-1)/k*
  - Usually, **k = 10**

Recap

- Given a dataset, begin by splitting into

- **Model selection**: Use k-fold cross-validation on TRAIN to train predictor and choose magic parameters such as λ



- **Model assessment**: Use TEST to assess the accuracy of the model you output
  - Never ever ever ever ever train or choose parameters based on the test data

Example

- Given 10,000-dimensional data and n examples, we pick a subset of 50 dimensions that have the highest correlation with labels in the training set:

50 indices j that have largest $$\frac{|\sum_{i=1}^n x_{i,j} y_i|}{\sqrt{\sum_{i=1}^n x_{i,j}^2}}$$

- After picking our 50 features, we then use CV to train ridge regression with regularization λ
- What's wrong with this procedure?

Recap

- Learning is…
  - Collect some data
    - E.g., housing info and sale price
  - Randomly split dataset into TRAIN, VAL, and TEST
    - E.g., 80%, 10%, and 10%, respectively
  - Choose a hypothesis class or model
    - E.g., linear with non-linear transformations
  - Choose a loss function
    - E.g., least squares with ridge regression penalty on TRAIN
  - Choose an optimization procedure
    - E.g., set derivative to zero to obtain estimator, cross-validation on VAL to pick num. features and amount of regularization
  - Justifying the accuracy of the estimate
    - E.g., report TEST error

Simple Variable Selection LASSO: Sparse Regression

Machine Learning – CSE546
Kevin Jamieson
University of Washington

October 9, 2016

Sparsity

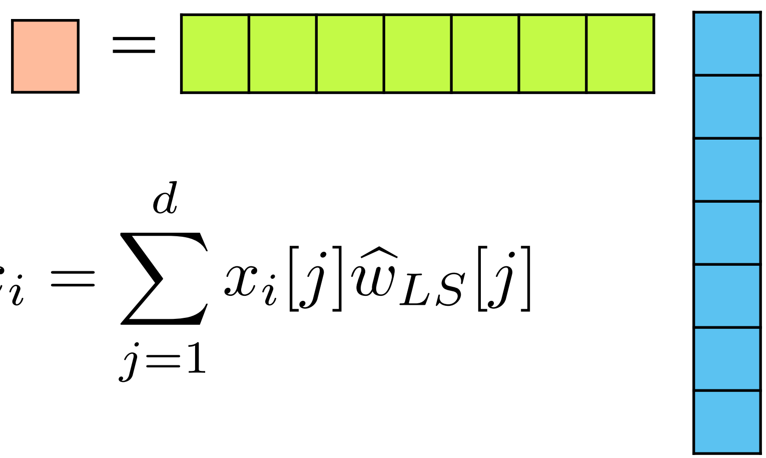
$$\widehat{w}_{LS} = \arg\min_w \sum_{i=1}^n \left(y_i - x_i^T w\right)^2$$

- Vector **w** is sparse, if many entries are zero

Sparsity

$$\widehat{w}_{LS} = \arg\min_w \sum_{i=1}^n \left(y_i - x_i^T w\right)^2$$

- Vector **w** is sparse, if many entries are zero
  - **Efficiency**: If size(**w**) = 100 Billion, each prediction is expensive:
    - If **w** is sparse, prediction computation only depends on number of non-zeros



$$\widehat{y}_i = \widehat{w}_{LS}^\top x_i = \sum_{j=1}^d x_i[j]\widehat{w}_{LS}[j]$$

Sparsity

$$\widehat{w}_{LS} = \arg\min_w \sum_{i=1}^{n} \left(y_i - x_i^T w\right)^2$$

- Vector **w** is sparse, if many entries are zero
  - **Interpretability**: What are the relevant dimension to make a prediction?

- How do we find "best" subset among all possible?

Lot size
Single Family
Year built
Last sold price
Last sale price/sqft
Finished sqft
Unfinished sqft
Finished basement sqft
floors
Flooring types
Parking type
Parking amount
Cooling
Heating
Exterior materials
Roof type
Structure style

Dishwasher
Garbage disposal
Microwave
Range / Oven
Refrigerator
Washer
Dryer
Laundry location
Heating type
Jetted Tub
Deck
Fenced Yard
Lawn
Garden
Sprinkler System

Finding best subset: **Exhaustive**

- Try all subsets of size 1, 2, 3, … and one that minimizes validation error
- Problem?

Finding best subset: Greedy

Forward stepwise:
Starting from simple model and iteratively add features most useful to fit

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

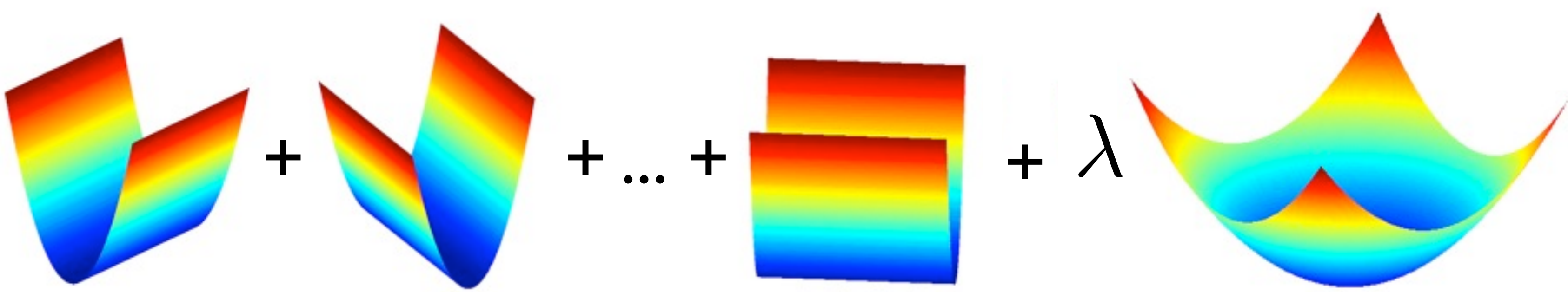
Combining forward and backward steps:
In forward algorithm, insert steps to remove features no longer as important

Lots of other variants, too.

Finding best subset: **Regularize**

Ridge regression makes coefficients small

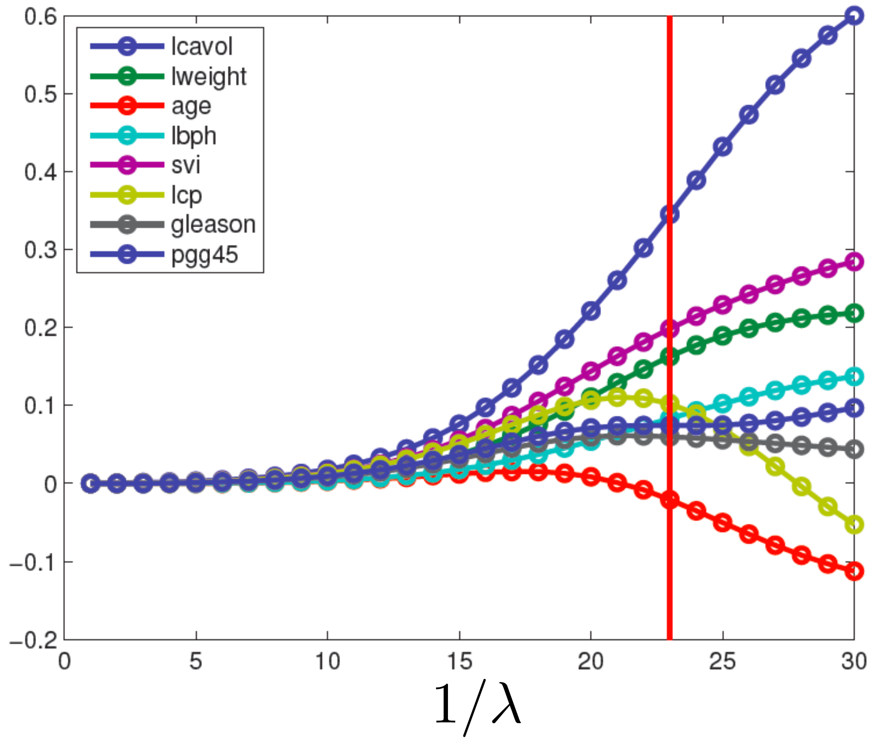
$$\widehat{w}_{ridge} = \arg\min_{w} \sum_{i=1}^{n} \left(y_i - x_i^T w\right)^2 + \lambda ||w||_2^2$$



Finding best subset: Regularize

Ridge regression makes coefficients small

$$\widehat{w}_{ridge} = \arg\min_w \sum_{i=1}^n \left(y_i - x_i^T w\right)^2 + \lambda ||w||_2^2$$

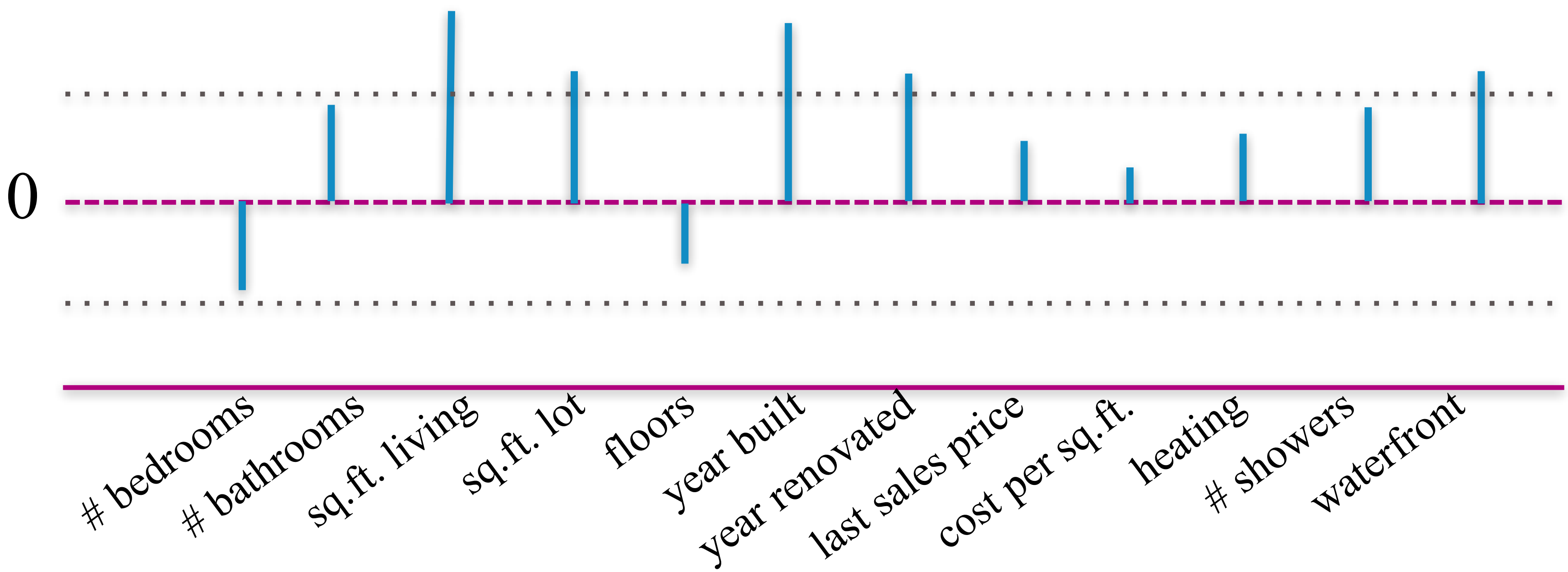


From
Kevin Murphy
textbook

Thresholded Ridge Regression

$$\widehat{w}_{ridge} = \arg\min_{w} \sum_{i=1}^{n} \left(y_i - x_i^T w\right)^2 + \lambda ||w||_2^2$$

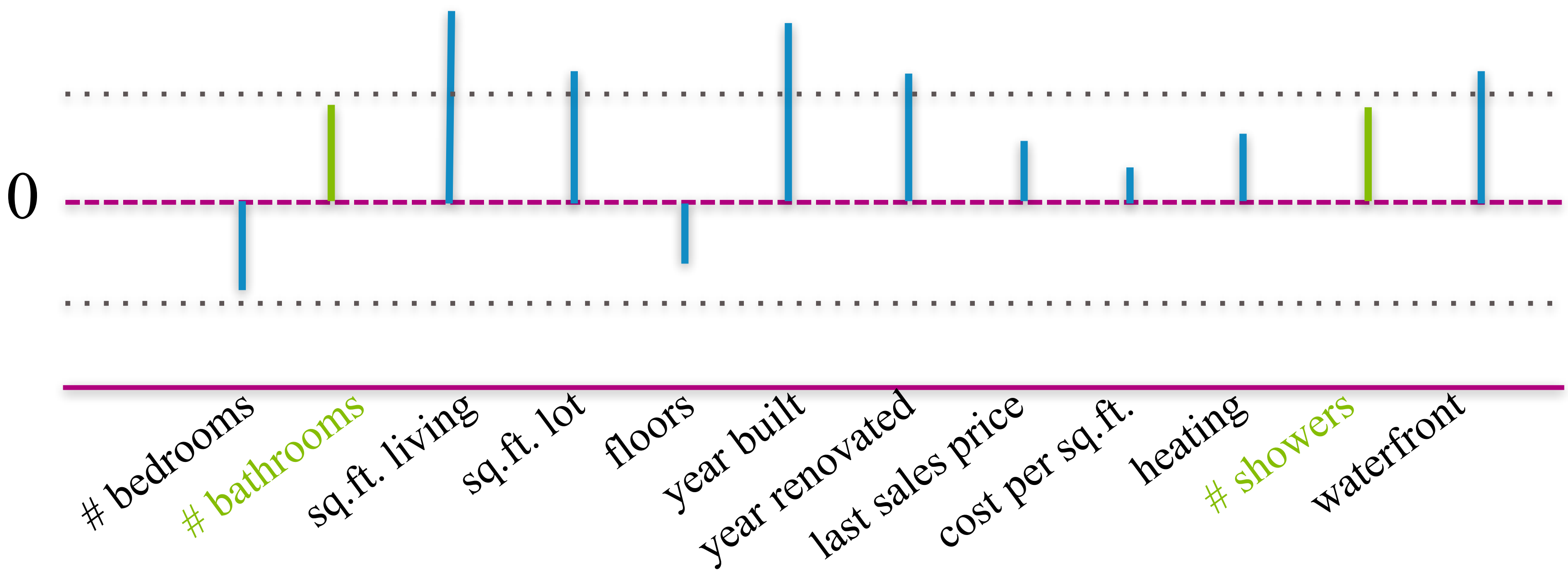
Why don't we just set **small** ridge coefficients to 0?



Thresholded Ridge Regression

$$\widehat{w}_{ridge} = \arg\min_w \sum_{i=1}^{n} \left(y_i - x_i^T w\right)^2 + \lambda ||w||_2^2$$

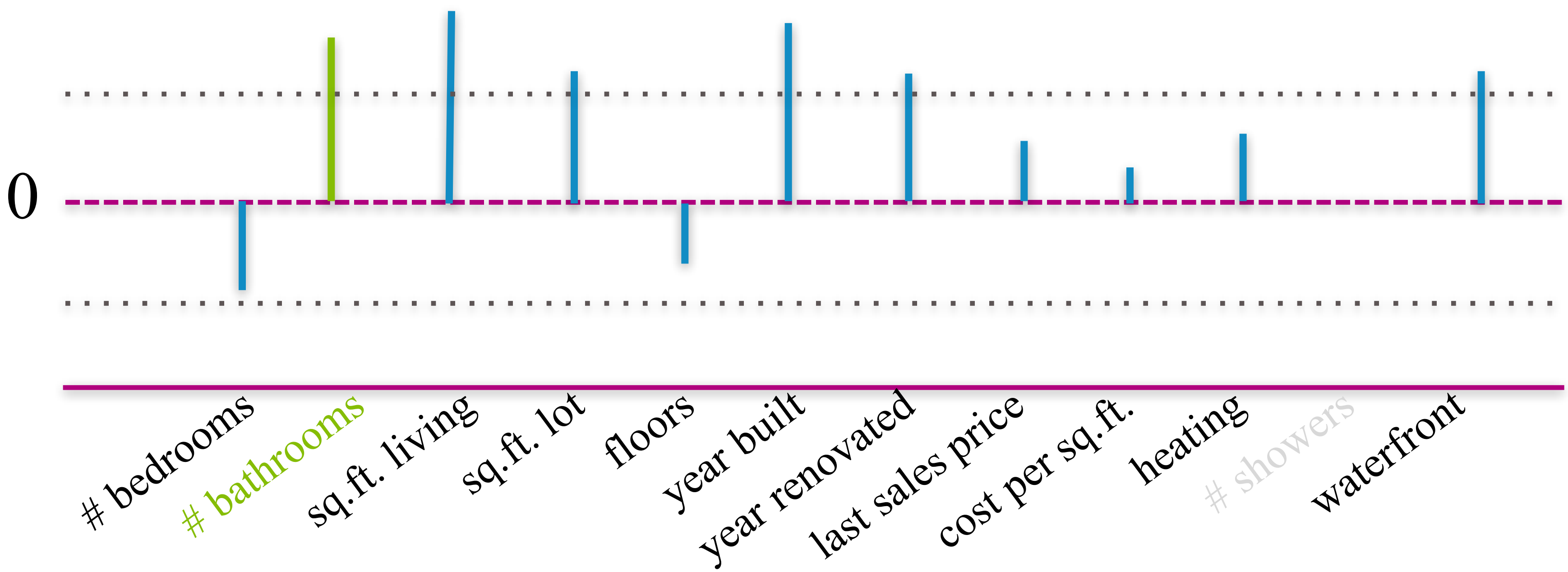
Consider two related features (bathrooms, showers)



Thresholded Ridge Regression

$$\widehat{w}_{ridge} = \arg\min_w \sum_{i=1}^n \left(y_i - x_i^T w\right)^2 + \lambda ||w||_2^2$$

What if we **didn't** include showers? Weight on bathrooms increases!

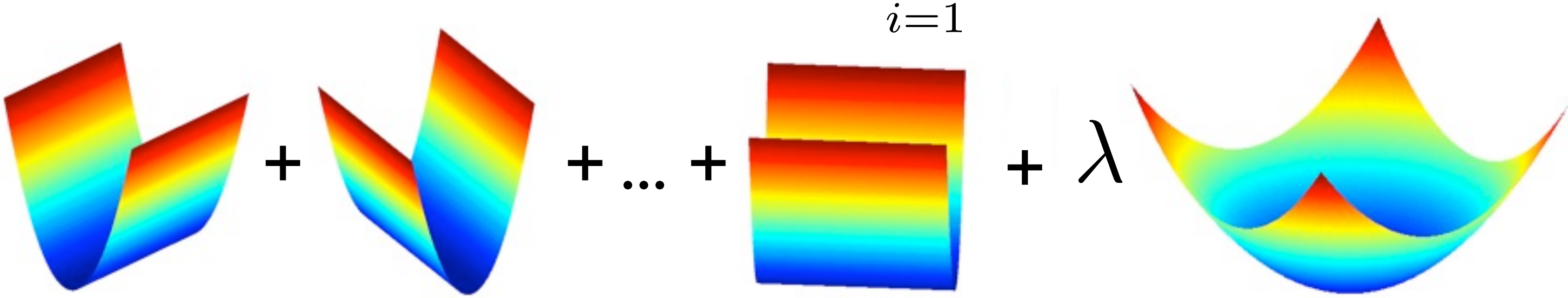


Can another regularizer perform selection automatically?

Recall Ridge Regression

- Ridge Regression objective:

$$\widehat{w}_{ridge} = \arg\min_{w} \sum_{i=1}^{n} \left(y_i - x_i^T w\right)^2 + \lambda ||w||_2^2$$

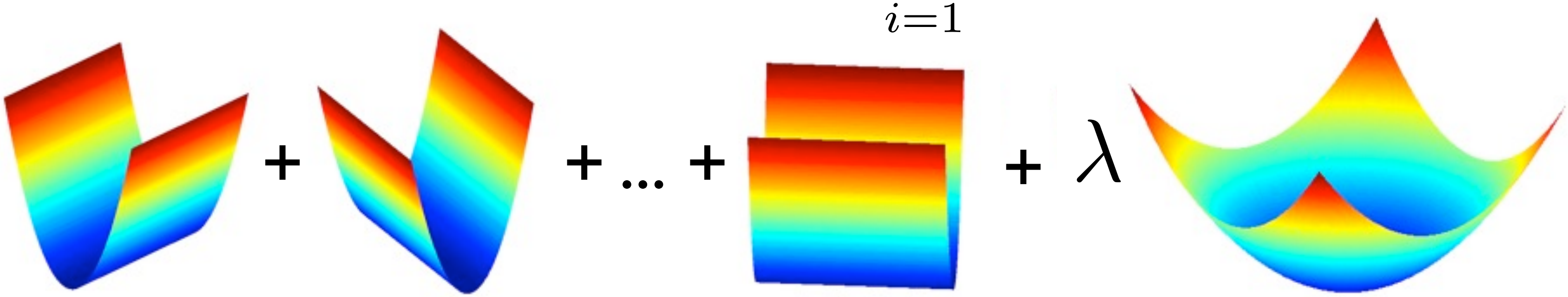


$$\|w\|_p = \left(\sum_{i=1}^{d} |w|^p \right)^{1/p}$$

Ridge vs. Lasso Regression

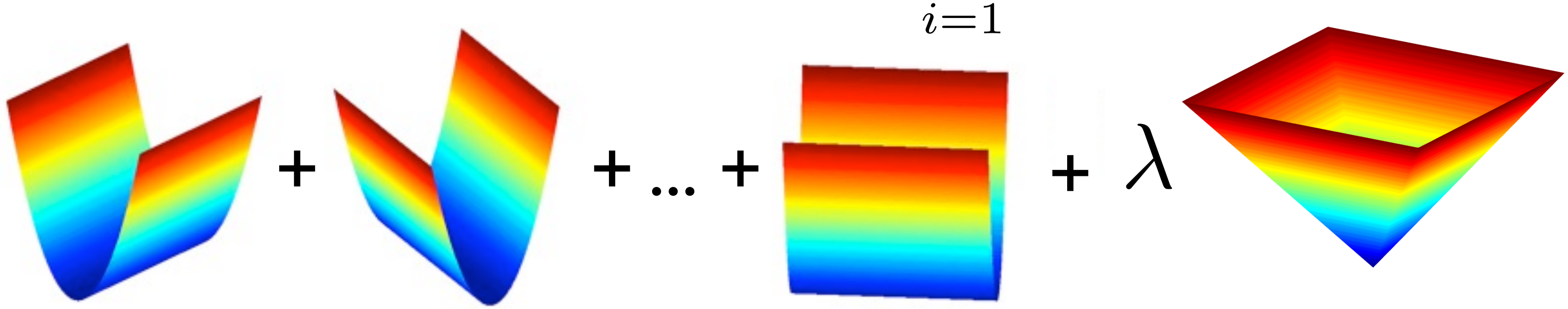
- Ridge Regression objective:

$$\widehat{w}_{ridge} = \arg\min_{w} \sum_{i=1}^{n} \left(y_i - x_i^T w\right)^2 + \lambda ||w||_2^2$$



- Lasso objective:

$$\widehat{w}_{lasso} = \arg\min_{w} \sum_{i=1}^{n} \left(y_i - x_i^T w\right)^2 + \lambda ||w||_1$$



Penalized Least Squares

$$\text{Ridge} : r(w) = ||w||_2^2 \qquad \text{Lasso} : r(w) = ||w||_1$$

$$\widehat{w}_r = \arg\min_w \sum_{i=1}^n \left(y_i - x_i^T w\right)^2 + \lambda r(w)$$

Penalized Least Squares

$$\text{Ridge} : r(w) = ||w||_2^2 \qquad \text{Lasso} : r(w) = ||w||_1$$

$$\widehat{w}_r = \arg\min_w \sum_{i=1}^n \left(y_i - x_i^T w\right)^2 + \lambda r(w)$$

For any $\lambda \geq 0$ for which $\widehat{w}_r$ achieves the minimum, there exists a $\nu \geq 0$ such that

$$\widehat{w}_r = \arg\min_w \sum_{i=1}^n \left(y_i - x_i^T w\right)^2 \qquad \text{subject to } r(w) \leq \nu$$

Penalized Least Squares

$$\text{Ridge} : r(w) = ||w||_2^2 \qquad \text{Lasso} : r(w) = ||w||_1$$

$$\widehat{w}_r = \arg\min_w \sum_{i=1}^n \left(y_i - x_i^T w\right)^2 + \lambda r(w)$$

For any $\lambda \geq 0$ for which $\widehat{w}_r$ achieves the minimum, there exists a $\nu \geq 0$ such that

$$\widehat{w}_r = \arg\min_w \sum_{i=1}^n \left(y_i - x_i^T w\right)^2 \qquad \text{subject to } r(w) \leq \nu$$