Warm up

Fix any $a, b, c > 0$.

1. What is the $x \in \mathbb{R}$ that minimizes $ax^2 + bx + c$

2. What is the $x \in \mathbb{R}$ that minimizes $\max\{-ax + b, cx\}$

Warm up

Solving argmin_w ||Xw - y||^2 + lambda * ||w||^2 = (H^TH + lambda * I)

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

Generate random data

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

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

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

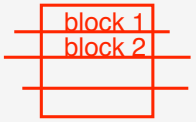
n*p floats
p*p floats
p floats
np + p^2 + p floats total

```
# construct HTH   memory allocation
HTH = np.zeros((p,p))      p^2 floats
HTy = np.zeros(p)          p floats
for i in range(n):
    hi = np.dot(X[i,:], G.T)+b      p
    HTH += np.outer(hi, hi)         p^2
    HTy += y[i]*hi                  p
    if i % 1000==0: print(i)
```

p^2 + 2p floats total
(HTH, HTy, hi)

```
# construct HTH
HTH = np.zeros((p,p))      p^2
HTy = np.zeros(p)          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      p^2
    HTH += np.dot(Hi.T, Hi)
    HTy += np.dot(Hi.T, y[i*block:min(n,(i+1)*block)])
```

divide H into chunks of size block



2p^2 + p total

```
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?

green is the fastest, but has the highest memory cost.
numpy is faster.

Regularization

Machine Learning – CSE546

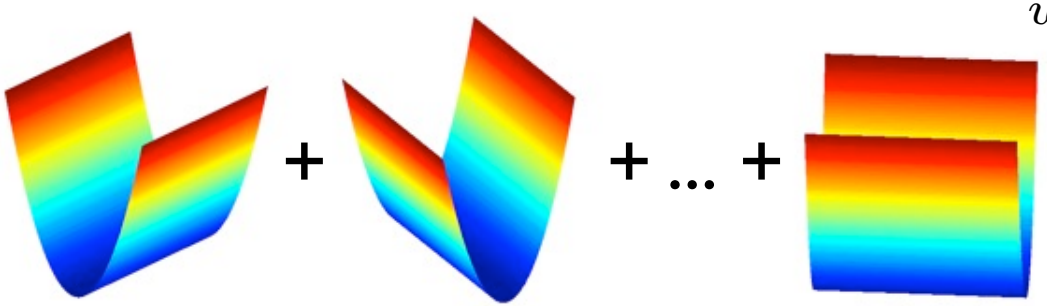
Kevin Jamieson
University of Washington

April 15, 2019

Ridge Regression

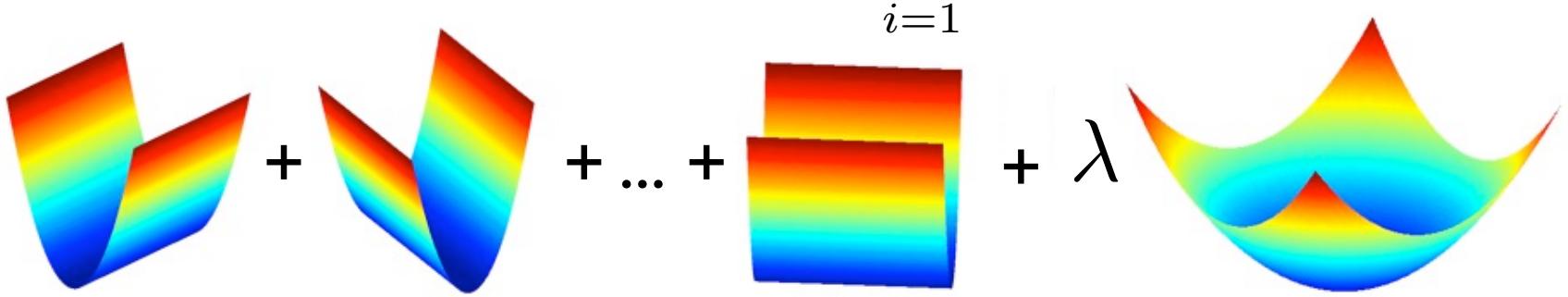
- Old Least squares objective:

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



- 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

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

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

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

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

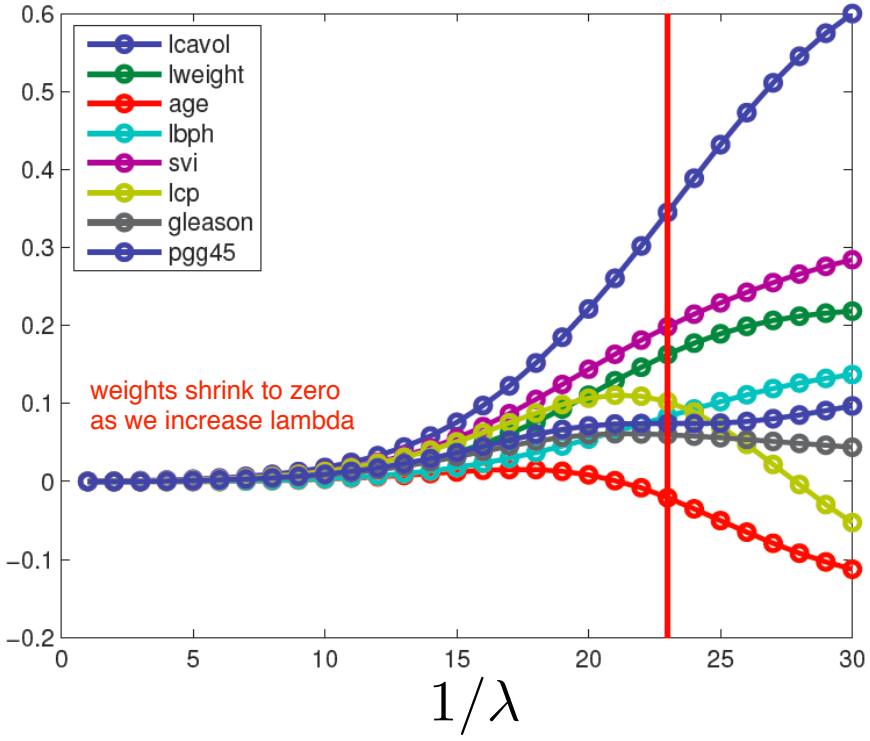
$$\widehat{w}_{ridge} = \left(\sum_{i=1}^{n} x_i x_i^T + \lambda I\right)^{-1} \left(\sum_{i=1}^{n} x_i y_i\right)$$

Unique solution to w_ridge!

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

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

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

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

$$= \sigma^2 + \frac{\lambda^2}{(n+\lambda)^2}(w^T x)^2 + \frac{d\sigma^2 n}{(n+\lambda)^2}\|x\|_2^2$$

(verify at home)

Irreduc. Error — Bias-squared — Variance

tons of data drive bias and variance to zero, but irreducible error remains.

solution shrinks towards zero, but we bias our function towards zero, and thus increase our bias squared.

Ridge Regression: Effect of Regularization

true distribution of
X (observed) and Y (what we want to predict)

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

TRAIN error: no regularizer!

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

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 <= true error

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:

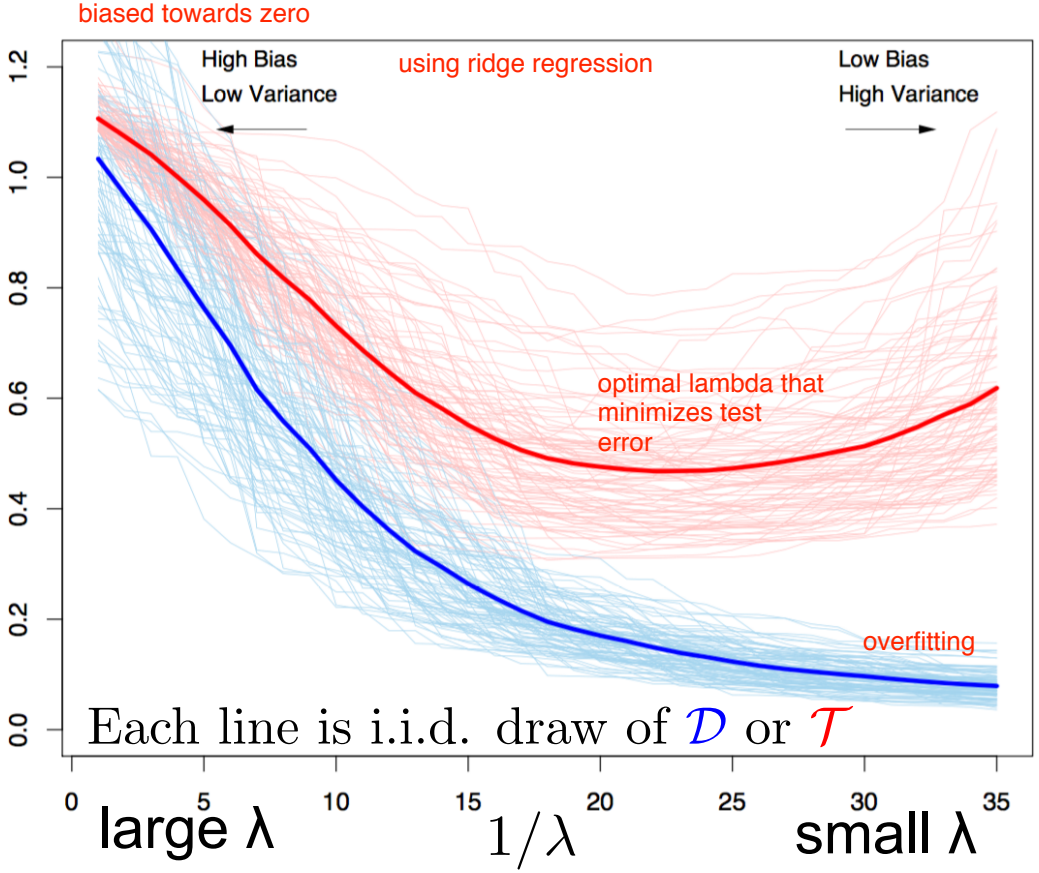
$$\mathcal{T} \overset{i.i.d.}{\sim} P_{XY}$$

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

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:

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

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

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

Cross-Validation

How to choose the proper lambda value

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…

  then it is just training data… Defeats the purpose of the testing set. We want an unbiased view of how our model performs because we can incorporate the test data into the model.

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\j – training data with j th data point ($\mathbf{x}_j$,$\mathbf{y}_j$) moved to validation set

take out a single data point!

- **Learn classifier $f_{D\setminus j}$ with D\j dataset**

i.e. use ridge regression to find the optimal solution w given this data.

- **Estimate true error** as squared error on predicting $\mathbf{y}_j$:

the point we removed.

  - Unbiased estimate of $\text{error}_{\text{true}}(f_{D\setminus j})$!

But by only taking out one… we have huge variance in our approximation of the testing error.

(LOO) Leave-one-out cross validation

- Consider a **validation set with 1 example**:
  - D – training data
  - $D\backslash j$ – training data with jth 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 $\text{error}_{\text{true}}(f_{D\backslash j})$!

repeat process for each data point.

- **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:

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

Biased estimator of the true error.

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

Text

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

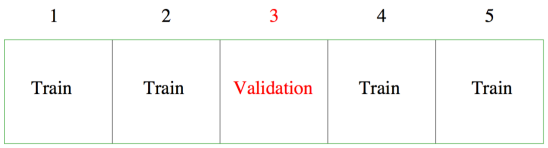
Computational cost of LOO

- 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!!!
  -

Has to retrain the model for each time!

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:**



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

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

validation set helps you choose the hyperparameters of the model (i.e. lambda —> parameters that influence learning)

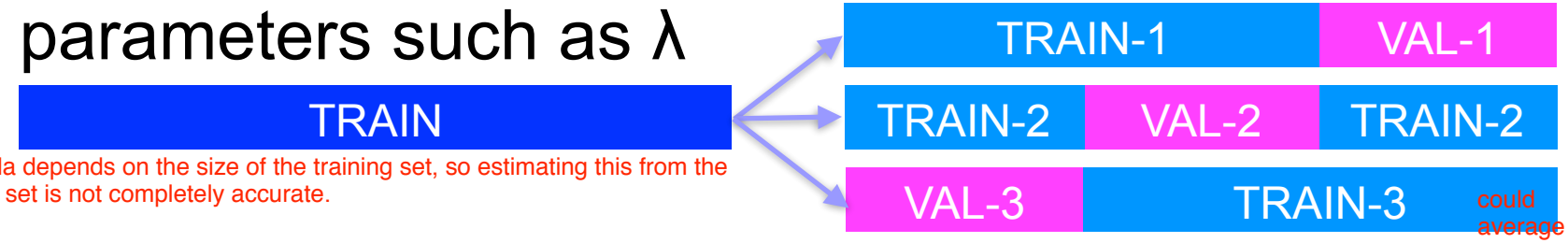
$$error_{k-fold} = \frac{1}{k}\sum_{i=1}^{k} error_{\mathcal{D}_i}$$

- *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



but lambda depends on the size of the training set, so estimating this from the validation set is not completely accurate.

- **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

could average lambda from each one of these? Use calculus to find min lambda?

w_ridge =

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?

Introduced information from the validation set into the training set. Used the validation data to help choose the 50 most correlated dimensions BEFORE separating into training & validation. So validation set wont give a good estimate of the true testing error!

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

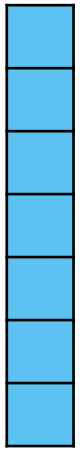
$$\widehat{w}_{LS} = \arg\min_{w} \sum_{i=1}^{n} \left(y_i - x_i^T w\right)^2$$

- 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

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

if many of these are zero, we can ignore all the zero entries!



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?

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

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

Finding best subset: **Exhaustive**

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

lots of subsets…

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

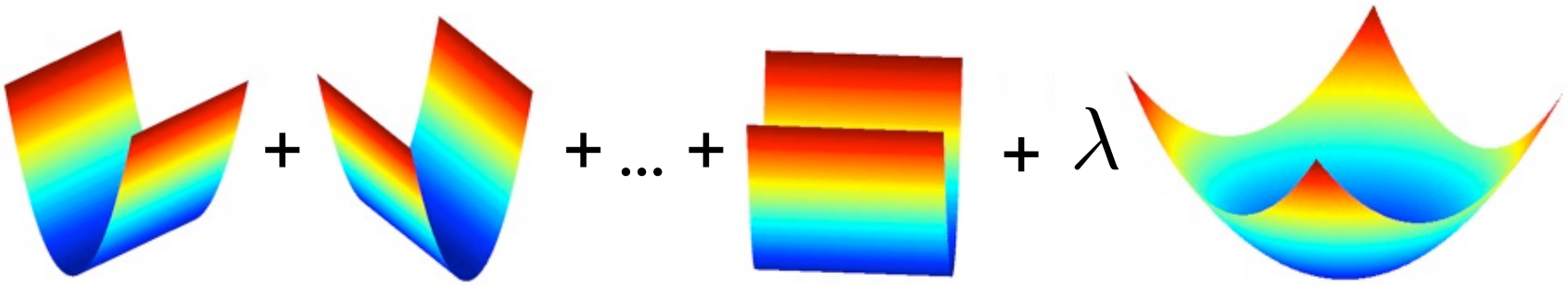
Combining forward and backward steps:
In forward algorithm, insert steps to remove features no longer as important

Lots of other variants, too.

Finding best subset: Regularize

Ridge regression makes coefficients small

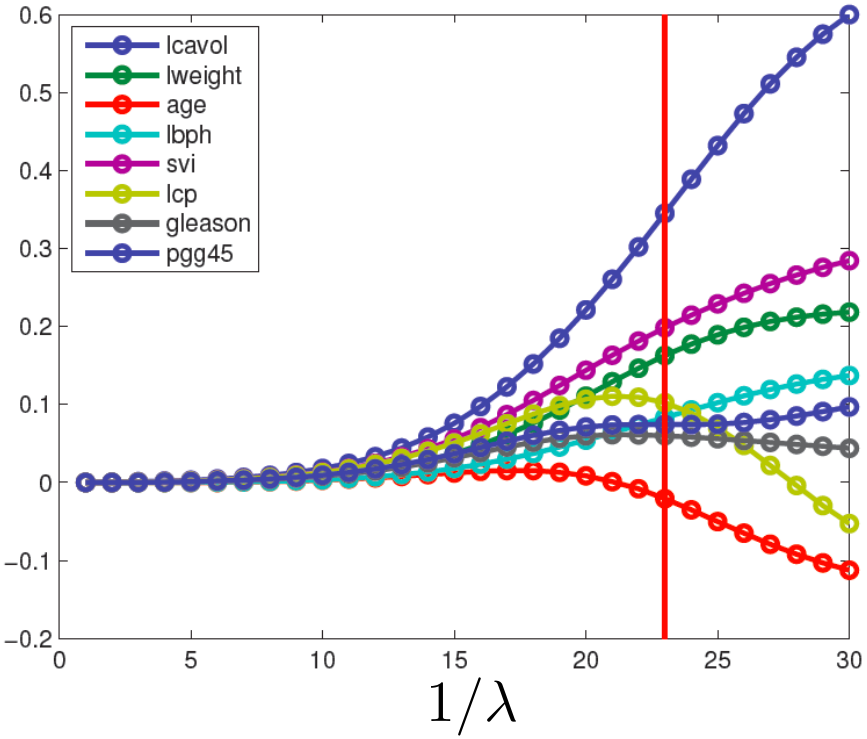
$$\widehat{w}_{ridge} = \arg\min_{w} \sum_{i=1}^{n} \left(y_i - x_i^T w\right)^2 + \lambda ||w||_2^2$$



Finding best subset: Regularize

Ridge regression makes coefficients small

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

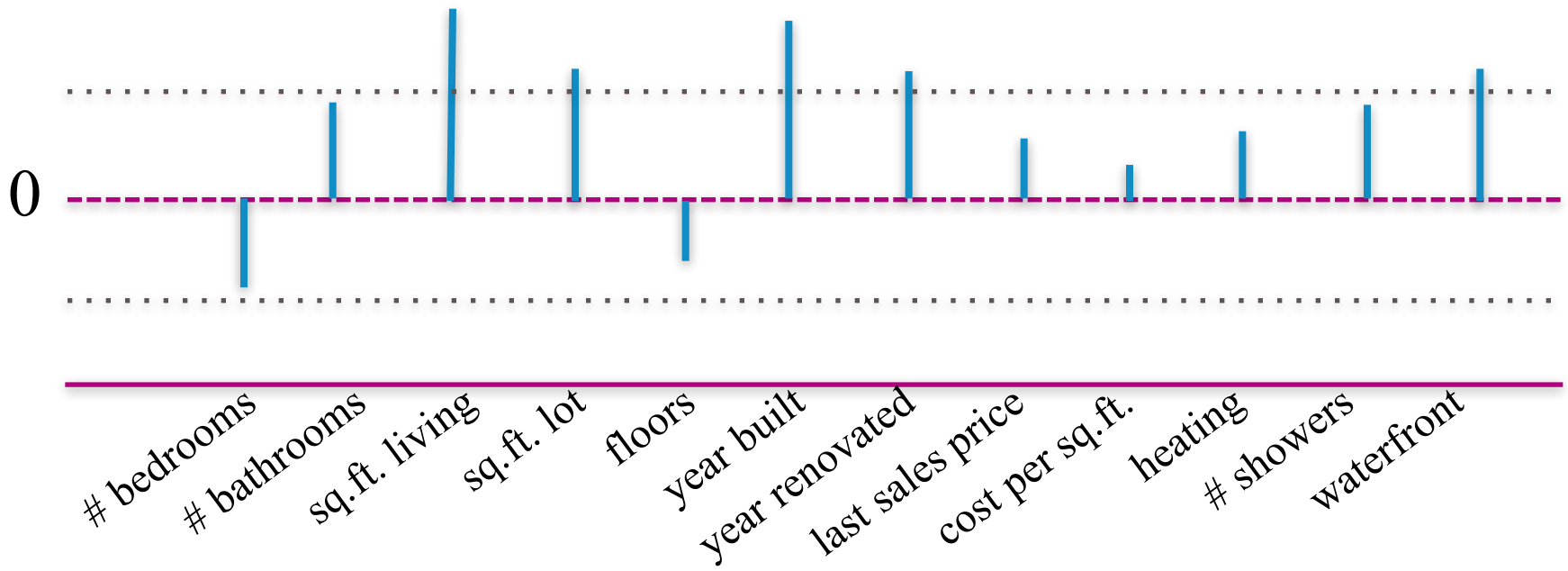


From
Kevin Murphy
textbook

Thresholded Ridge Regression

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

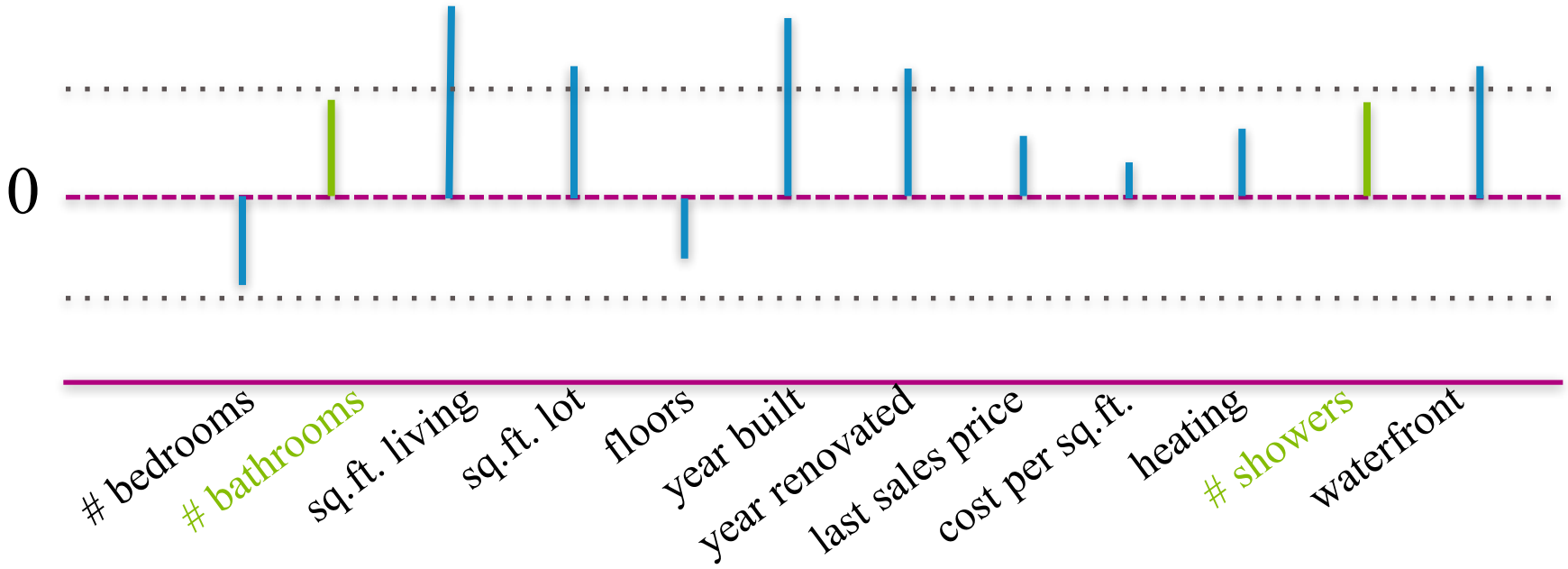
Why don't we just set **small** ridge coefficients to 0?



Thresholded Ridge Regression

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

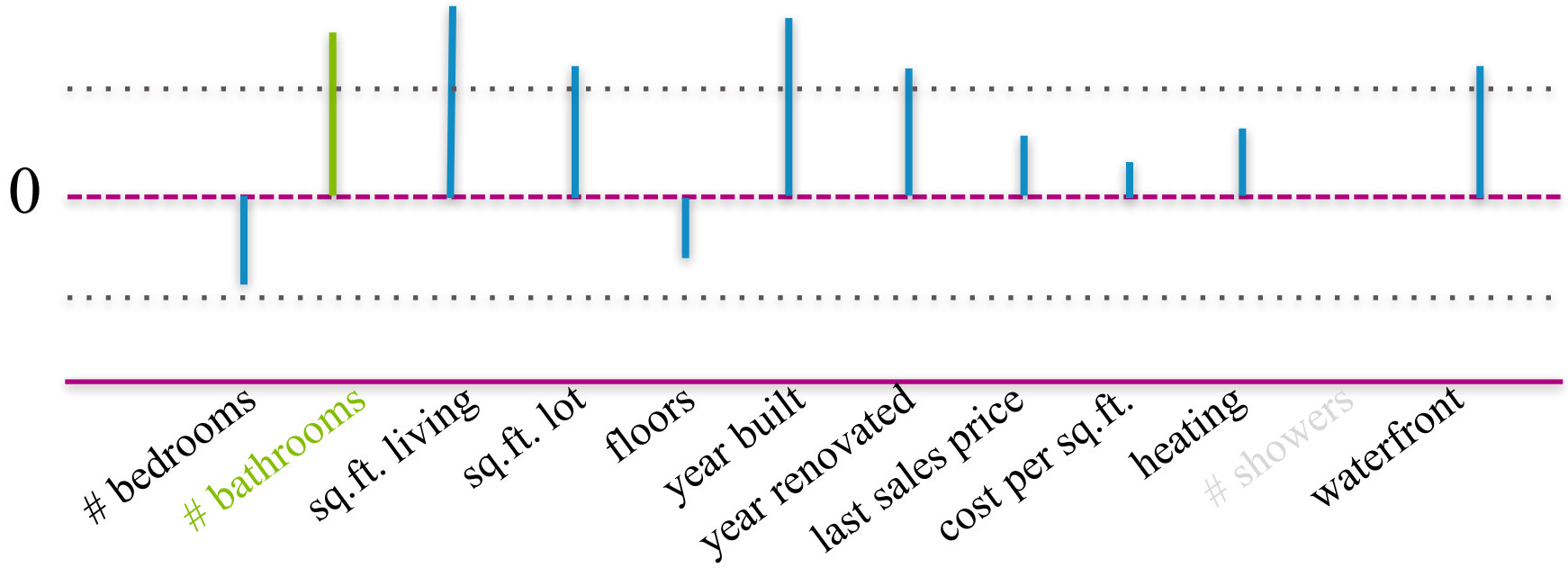
Consider two related features (bathrooms, showers)



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!

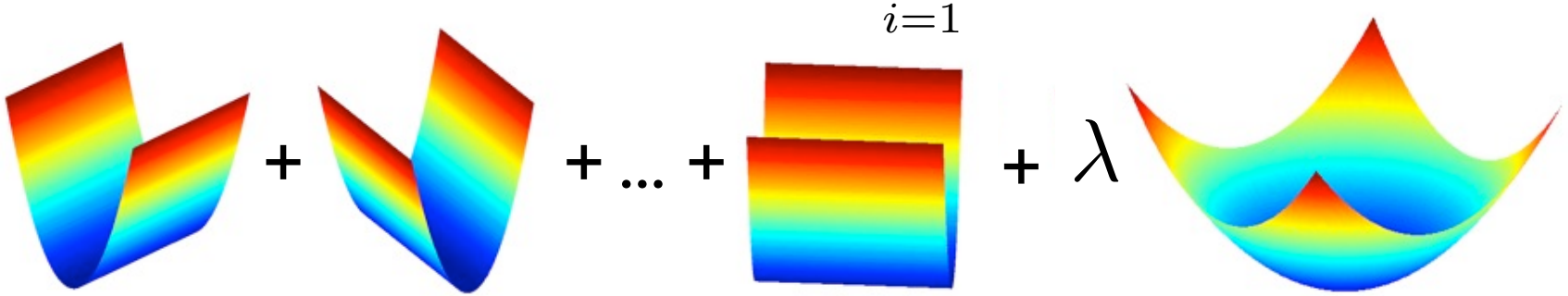


Can another regularizer perform selection automatically?

Recall Ridge Regression

- Ridge Regression objective:

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

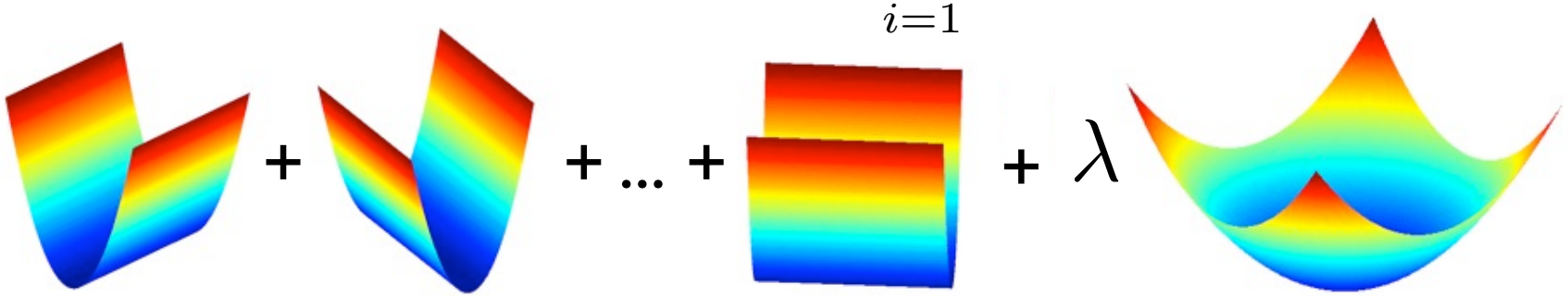


$$\|w\|_p = \left(\sum_{i=1}^{d} |w|^p\right)^{1/p}$$

Ridge vs. Lasso Regression

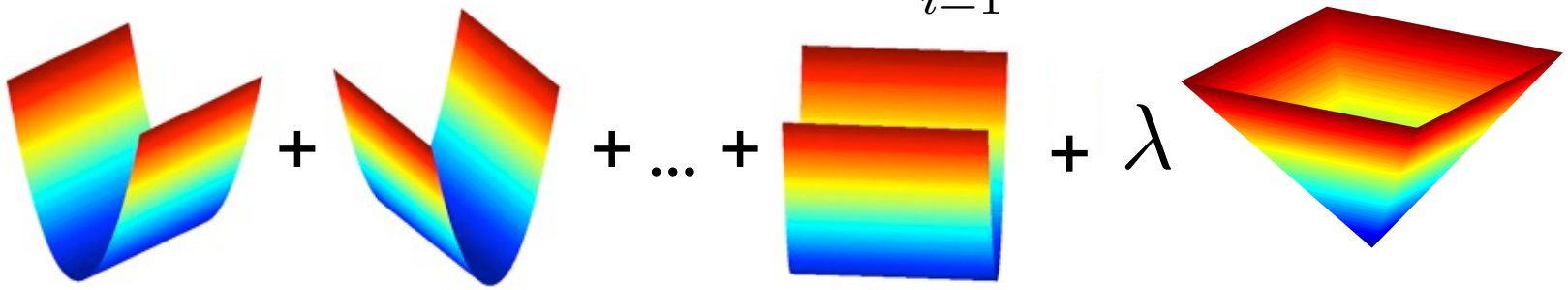
- Ridge Regression objective:

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



- 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

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

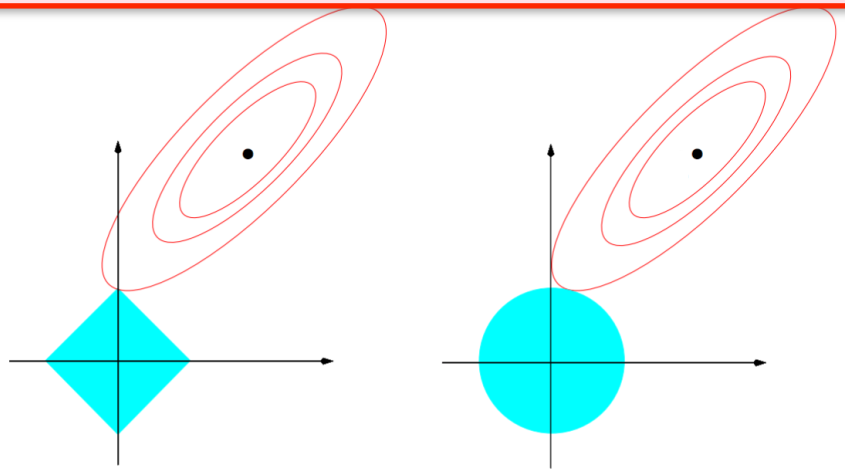
Penalized Least Squares

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

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

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

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



Optimizing the LASSO Objective

- LASSO solution:

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

$$\widehat{b}_{lasso} = \arg\min_{w,b} \frac{1}{n} \sum_{i=1}^{n} \left(y_i - x_i^T \widehat{w}_{lasso}\right))$$

Optimizing the LASSO Objective

- LASSO solution:

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

$$\widehat{b}_{lasso} = \arg\min_{w,b} \frac{1}{n} \sum_{i=1}^{n} \left(y_i - x_i^T \widehat{w}_{lasso}\right)$$

So as usual, preprocess to make sure that $\frac{1}{n}\sum_{i=1}^{n} y_i = 0, \frac{1}{n}\sum_{i=1}^{n} x_i = \mathbf{0}$

so we don't have to worry about an offset.

Optimizing the LASSO Objective

- LASSO solution:

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

$$\widehat{b}_{lasso} = \arg\min_{w,b} \frac{1}{n} \sum_{i=1}^{n} \left(y_i - x_i^T \widehat{w}_{lasso}\right)$$

So as usual, preprocess to make sure that $\frac{1}{n}\sum_{i=1}^{n} y_i = 0, \frac{1}{n}\sum_{i=1}^{n} x_i = \mathbf{0}$

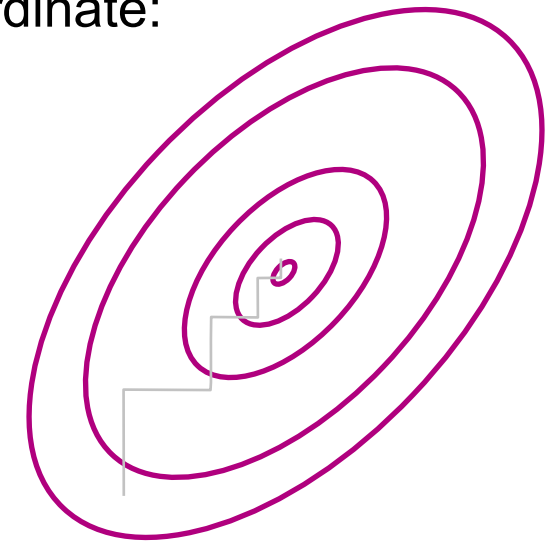
so we don't have to worry about an offset.

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

How do we solve this?

Coordinate Descent

- Given a function, we want to find minimum
- Often, it is easy to find minimum along a single coordinate:



- How do we pick next coordinate?
- Super useful approach for *many* problems
  - Converges to optimum in some cases, such as LASSO

Optimizing LASSO Objective One Coordinate at a Time

Fix any $j \in \{1, \ldots, d\}$

$$\sum_{i=1}^{n} \left(y_i - x_i^T w\right)^2 + \lambda ||w||_1 = \sum_{i=1}^{n} \left(y_i - \sum_{k=1}^{d} x_{i,k}\, w_k \right)^2 + \lambda \sum_{k=1}^{d} |w_k|$$

$$= \sum_{i=1}^{n} \left(\left(y_i - \sum_{k \neq j} x_{i,k}\, w_k \right) - x_{i,j}\, w_j \right)^2 + \lambda \sum_{k \neq j} |w_k| + \lambda |w_j|$$

Optimizing LASSO Objective One Coordinate at a Time

Fix any $j \in \{1, \ldots, d\}$

$$\sum_{i=1}^{n} \left(y_i - x_i^T w\right)^2 + \lambda ||w||_1 = \sum_{i=1}^{n} \left(y_i - \sum_{k=1}^{d} x_{i,k}\, w_k\right)^2 + \lambda \sum_{k=1}^{d} |w_k|$$

$$= \sum_{i=1}^{n} \left(\left(y_i - \sum_{k \neq j} x_{i,k}\, w_k\right) - x_{i,j}\, w_j\right)^2 + \lambda \sum_{k \neq j} |w_k| + \lambda |w_j|$$

Initialize $\widehat{w}_k = 0$ for all $k \in \{1, \ldots, d\}$

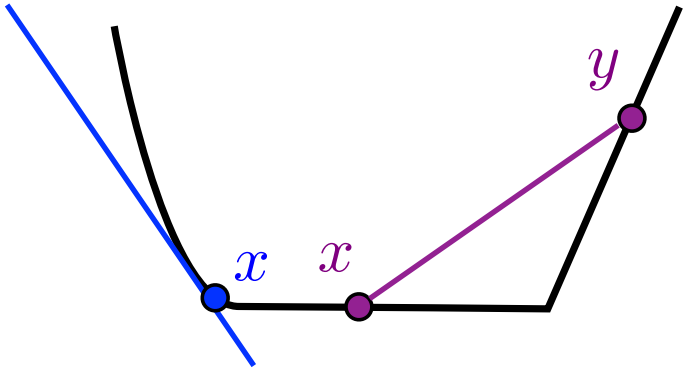
Loop over $j \in \{1, \ldots, n\}$:

$$r_i^{(j)} = \sum_{k \neq j} x_{i,j} \widehat{w}_k$$

$$\widehat{w}_j = \arg\min_{w_j} \sum_{i=1}^{n} \left(r_i^{(j)} - x_{i,j}\, w_j\right)^2 + \lambda |w_j|$$

Convex Functions

- Equivalent definitions of convexity:



f convex:

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y) \qquad \forall x, y, \lambda \in [0,1]$$

$$f(y) \geq f(x) + \nabla f(x)^T (y - x) \qquad \forall x, y$$

- **Gradients** lower bound convex functions and are unique at **x** iff function differentiable at **x**
- **Subgradients** generalize gradients to non-differentiable points:
  - Any supporting hyperplane at **x** that lower bounds entire function

$$g \text{ is a subgradient at } x \text{ if } f(y) \geq f(x) + g^T (y - x)$$

Taking the Subgradient

$$\widehat{w}_j = \arg\min_{w_j} \sum_{i=1}^{n} \left(r_i^{(j)} - x_{i,j}\, w_j \right)^2 + \lambda |w_j|$$

$$g \text{ is a subgradient at } x \text{ if } f(y) \geq f(x) + g^T (y - x)$$

- Convex function is minimized at w if 0 is a sub-gradient at w.

$$\partial_{w_j} |w_j| =$$

$$\partial_{w_j} \sum_{i=1}^{n} \left(r_i^{(j)} - x_{i,j}\, w_j \right)^2 =$$

Setting Subgradient to 0

$$\partial_{w_j}\left(\sum_{i=1}^{n}\left(r_i^{(j)} - x_{i,j}\, w_j\right)^2 + \lambda|w_j|\right) = \begin{cases} a_j w_j - c_j - \lambda & \text{if } w_j < 0 \\ [-c_j - \lambda, -c_j + \lambda] & \text{if } w_j = 0 \\ a_j w_j - c_j + \lambda & \text{if } w_j > 0 \end{cases}$$

$$a_j = (\sum_{i=1}^{n} x_{i,j}^2) \qquad c_j = 2(\sum_{i=1}^{n} r_i^{(j)} x_{i,j})$$

Setting Subgradient to 0

$$\partial_{w_j}\left(\sum_{i=1}^{n}\left(r_i^{(j)}-x_{i,j}\,w_j\right)^2+\lambda|w_j|\right)=\begin{cases}a_jw_j-c_j-\lambda & \text{if } w_j<0\\ [-c_j-\lambda,-c_j+\lambda] & \text{if } w_j=0\\ a_jw_j-c_j+\lambda & \text{if } w_j>0\end{cases}$$

$$a_j=(\sum_{i=1}^{n}x_{i,j}^2)\qquad c_j=2(\sum_{i=1}^{n}r_i^{(j)}x_{i,j})$$

$$\widehat{w}_j=\arg\min_{w_j}\sum_{i=1}^{n}\left(r_i^{(j)}-x_{i,j}\,w_j\right)^2+\lambda|w_j|$$

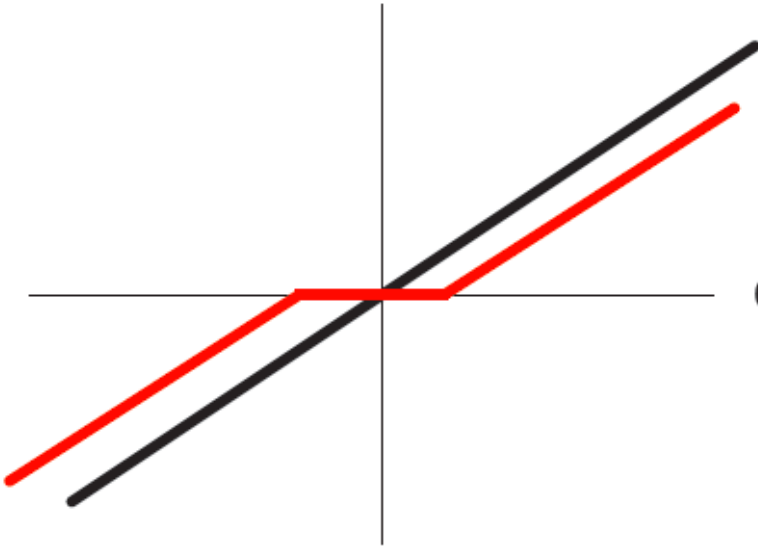
w is a minimum if
0 is a sub-gradient at w

$$\widehat{w}_j=\begin{cases}(c_j+\lambda)/a_j & \text{if } c_j<-\lambda\\ 0 & \text{if } |c_j|\le\lambda\\ (c_j-\lambda)/a_j & \text{if } c_j>\lambda\end{cases}$$

Soft Thresholding

$$\widehat{w}_j = \begin{cases} (c_j + \lambda)/a_j & \text{if } c_j < -\lambda \\ 0 & \text{if } |c_j| \leq \lambda \\ (c_j - \lambda)/a_j & \text{if } c_j > \lambda \end{cases}$$

$$a_j = \sum_{i=1}^{n} x_{i,j}^2 \qquad c_j = 2\sum_{i=1}^{n} \left(y_i - \sum_{k \neq j} x_{i,k}\, w_k \right) x_{i,j}$$



Coordinate Descent for LASSO (aka Shooting Algorithm)

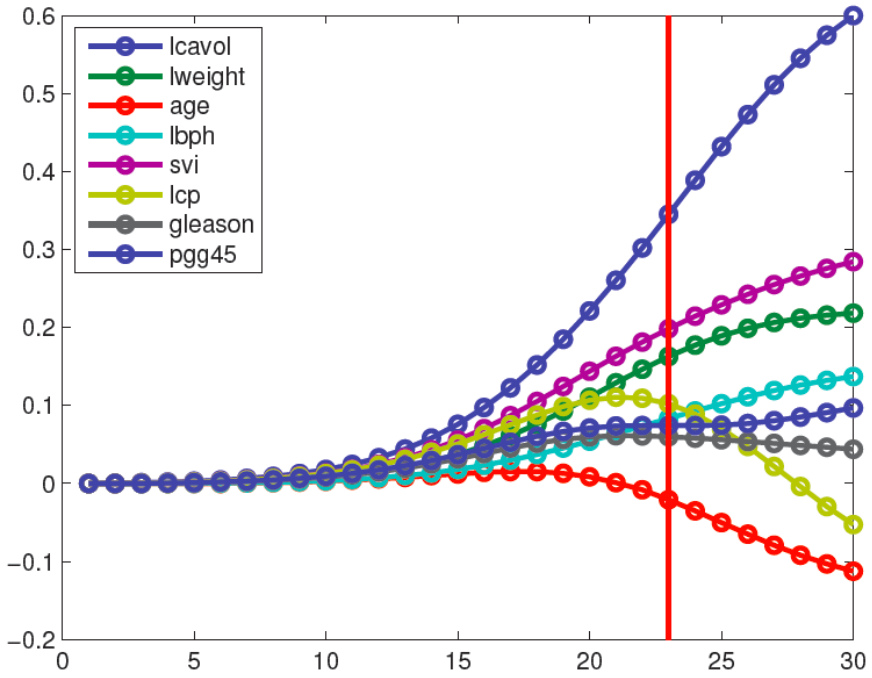
- Repeat until convergence
  - Pick a coordinate *l* at (random or sequentially)
    - Set:
    - Where:

$$\widehat{w}_j = \begin{cases} (c_j + \lambda)/a_j & \text{if } c_j < -\lambda \\ 0 & \text{if } |c_j| \leq \lambda \\ (c_j - \lambda)/a_j & \text{if } c_j > \lambda \end{cases}$$

$$a_j = \sum_{i=1}^{n} x_{i,j}^2 \qquad c_j = 2 \sum_{i=1}^{n} \left(y_i - \sum_{k \neq j} x_{i,k}\, w_k \right) x_{i,j}$$

  - For convergence rates, see Shalev-Shwartz and Tewari 2009
- Other common technique = LARS
  - Least angle regression and shrinkage, Efron et al. 2004

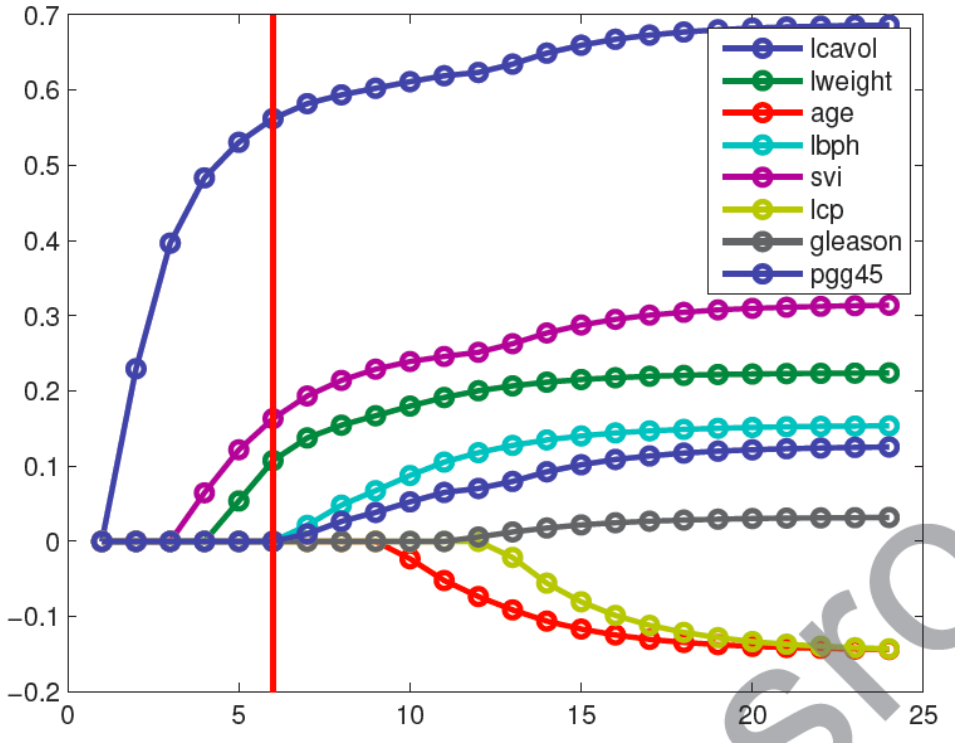
Recall: *Ridge* Coefficient Path



From
Kevin Murphy
textbook

- Typical approach: select λ using cross validation

Now: ***LASSO*** *Coefficient Path*



From
Kevin Murphy
textbook

What you need to know

- Variable Selection: find a sparse solution to learning problem
- L_1 regularization is one way to do variable selection
  - Applies beyond regression
  - Hundreds of other approaches out there
- LASSO objective non-differentiable, **but convex** → Use subgradient
- No closed-form solution for minimization → Use coordinate descent
- Shooting algorithm is simple approach for solving LASSO

Classification Logistic Regression

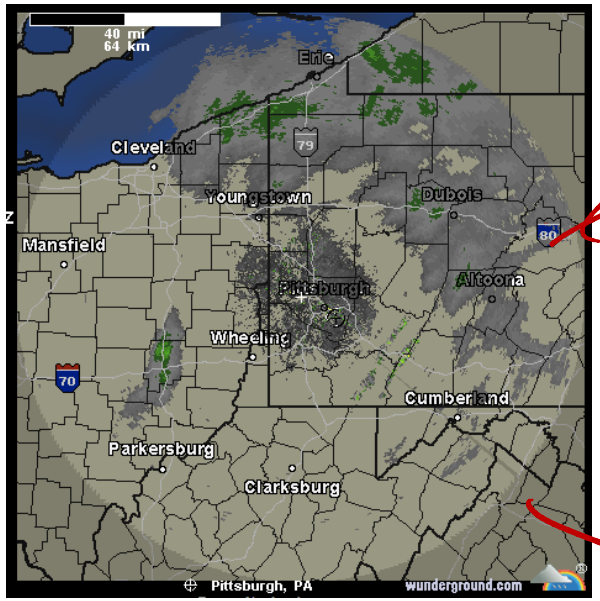
Machine Learning – CSE546

Kevin Jamieson
University of Washington

October 9, 2016

THUS FAR, REGRESSION: PREDICT A CONTINUOUS VALUE GIVEN SOME INPUTS

Weather prediction revisted



Temperature

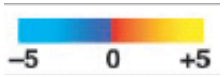
63°F

Reading Your Brain, Simple Example

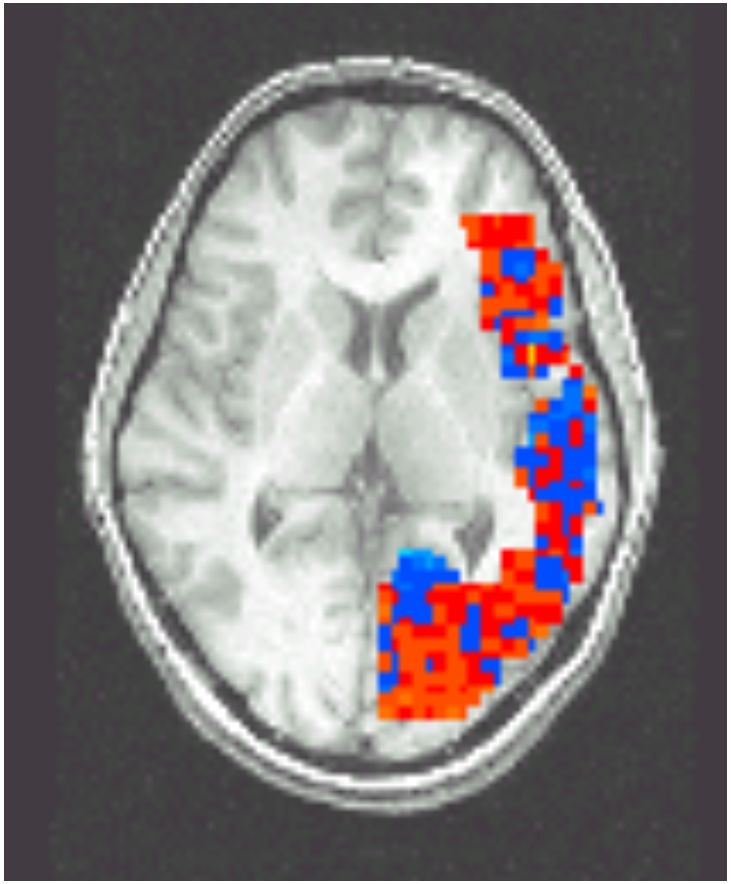
[Mitchell et al.]

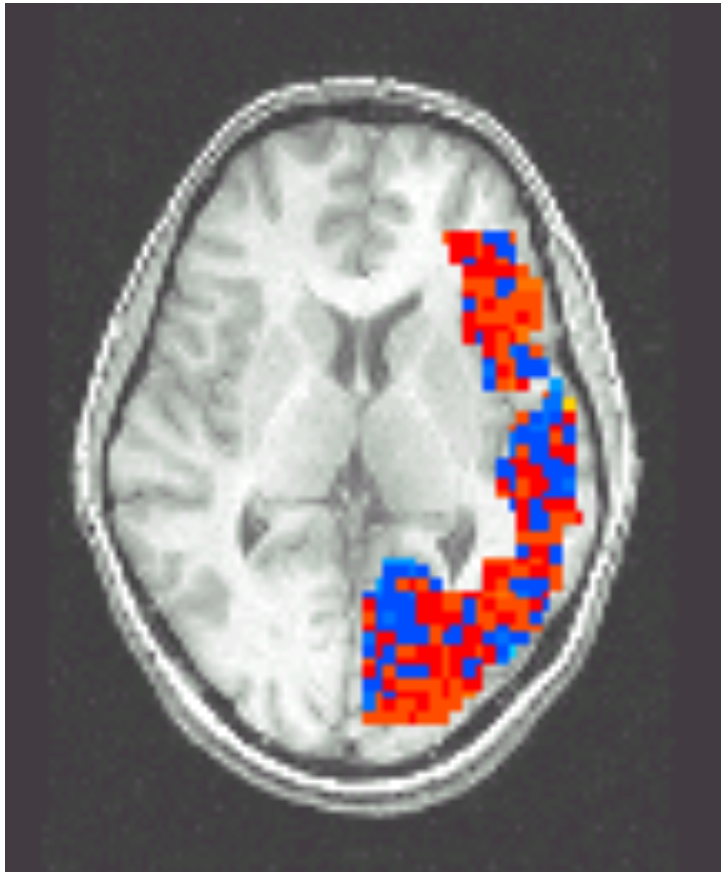
Pairwise classification accuracy: 85%

Person



Animal





Binary Classification

- **Learn**: f:**X** —>Y
  - **X** – features
  - Y – target classes
    $Y \in \{0, 1\}$
- **Loss function:**
- **Expected loss of f:**
- Suppose you know P(Y|**X**) exactly, how should you classify?
  - Bayes optimal classifier:

Link Functions

- Estimating P(Y|**X**): Why not use standard linear regression?

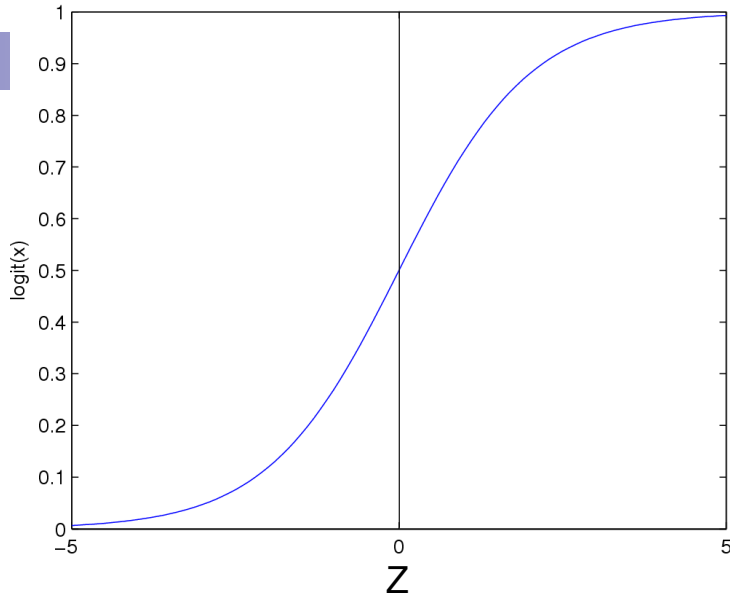
- Combining regression and probability?
  - Need a mapping from real values to [0,1]
  - A link function!

Logistic Regression

Logistic function (or Sigmoid): $\frac{1}{1+exp(-z)}$

- Learn P(Y|**X**) directly
  - Assume a particular functional form for link function
  - Sigmoid applied to a linear function of the input features:

$$P(Y=0|X,W)=\frac{1}{1+exp(w_0+\sum_i w_i X_i)}$$

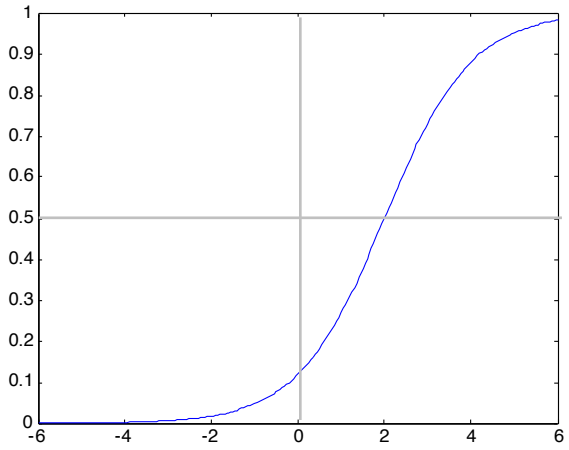


Features can be discrete or continuous!

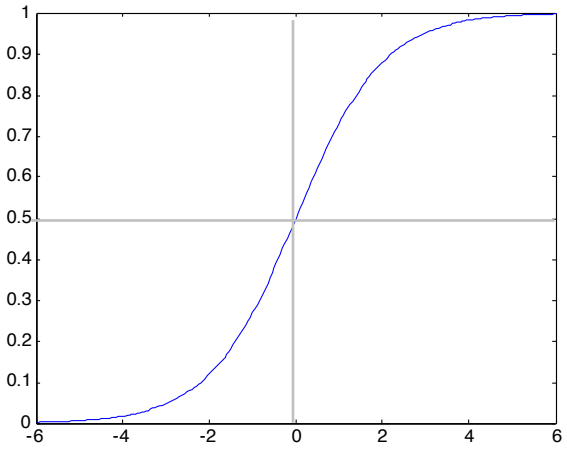
Understanding the sigmoid

$$g(w_0 + \sum_i w_i x_i) = \frac{1}{1 + e^{w_0 + \sum_i w_i x_i}}$$

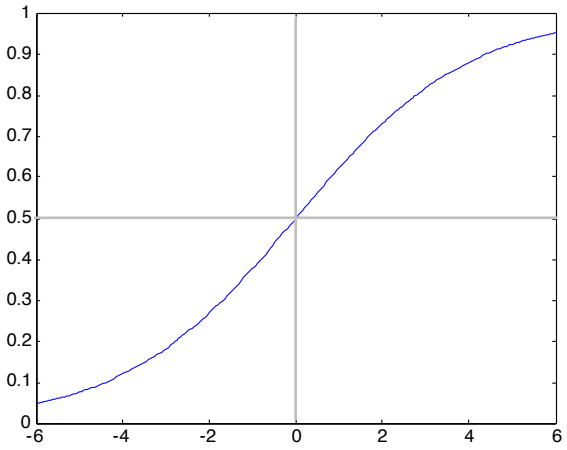
w_0=-2, w_1=-1



w_0=0, w_1=-1



w_0=0, w_1=-0.5



Sigmoid for binary classes

$$\mathbb{P}(Y=0|w,X) = \frac{1}{1+\exp(w_0+\sum_k w_k X_k)}$$

$$\mathbb{P}(Y=1|w,X) = 1-\mathbb{P}(Y=0|w,X) = \frac{\exp(w_0+\sum_k w_k X_k)}{1+\exp(w_0+\sum_k w_k X_k)}$$

$$\frac{\mathbb{P}(Y=1|w,X)}{\mathbb{P}(Y=0|w,X)} =$$

Sigmoid for binary classes

$$\mathbb{P}(Y=0|w,X) = \frac{1}{1+\exp(w_0+\sum_k w_k X_k)}$$

$$\mathbb{P}(Y=1|w,X) = 1-\mathbb{P}(Y=0|w,X) = \frac{\exp(w_0+\sum_k w_k X_k)}{1+\exp(w_0+\sum_k w_k X_k)}$$

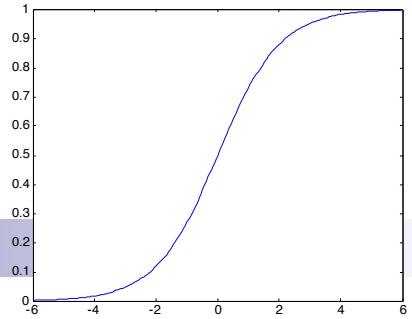
$$\frac{\mathbb{P}(Y=1|w,X)}{\mathbb{P}(Y=0|w,X)} = \exp(w_0+\sum_k w_k X_k)$$

$$\log \frac{\mathbb{P}(Y=1|w,X)}{\mathbb{P}(Y=0|w,X)} = w_0+\sum_k w_k X_k$$

Linear Decision Rule!

Logistic Regression – a Linear classifier

$$\frac{1}{1+exp(-z)}$$



$$g(w_0 + \sum_i w_i x_i) = \frac{1}{1 + e^{w_0 + \sum_i w_i x_i}}$$

$$\ln \frac{P(Y=0|X)}{P(Y=1|X)} = w_0 + \sum_i w_i X_i$$

Loss function: Conditional Likelihood

- Have a bunch of iid data of the form: $\{(x_i, y_i)\}_{i=1}^n \quad x_i \in \mathbb{R}^d, \quad y_i \in \{-1, 1\}$

$$P(Y = -1|x, w) = \frac{1}{1 + \exp(w^T x)}$$

$$P(Y = 1|x, w) = \frac{\exp(w^T x)}{1 + \exp(w^T x)}$$

- This is equivalent to:

$$P(Y = y|x, w) = \frac{1}{1 + \exp(-y\, w^T x)}$$

- So we can compute the maximum likelihood estimator:

$$\widehat{w}_{MLE} = \arg\max_w \prod_{i=1}^n P(y_i|x_i, w)$$

Loss function: Conditional Likelihood

- Have a bunch of iid data of the form: $\{(x_i, y_i)\}_{i=1}^{n} \quad x_i \in \mathbb{R}^d, \quad y_i \in \{-1, 1\}$

$$\widehat{w}_{MLE} = \arg\max_{w} \prod_{i=1}^{n} P(y_i | x_i, w) \qquad P(Y = y | x, w) = \frac{1}{1 + \exp(-y\, w^T x)}$$

Loss function: Conditional Likelihood

- Have a bunch of iid data of the form: $\{(x_i, y_i)\}_{i=1}^n \quad x_i \in \mathbb{R}^d, \quad y_i \in \{-1, 1\}$

$$\widehat{w}_{MLE} = \arg\max_w \prod_{i=1}^n P(y_i|x_i, w) \qquad P(Y = y|x, w) = \frac{1}{1 + \exp(-y\, w^T x)}$$

$$= \arg\min_w \sum_{i=1}^n \log(1 + \exp(-y_i\, x_i^T w))$$

Loss function: Conditional Likelihood

- Have a bunch of iid data of the form: $\{(x_i, y_i)\}_{i=1}^n \quad x_i \in \mathbb{R}^d, \quad y_i \in \{-1, 1\}$

$$\widehat{w}_{MLE} = \arg\max_w \prod_{i=1}^n P(y_i|x_i, w) \qquad P(Y = y|x, w) = \frac{1}{1 + \exp(-y\, w^T x)}$$

$$= \arg\min_w \sum_{i=1}^n \log(1 + \exp(-y_i\, x_i^T w))$$

Logistic Loss: $\ell_i(w) = \log(1 + \exp(-y_i\, x_i^T w))$

Squared error Loss: $\ell_i(w) = (y_i - x_i^T w)^2$ (MLE for Gaussian noise)

Loss function: Conditional Likelihood

- Have a bunch of iid data of the form: $\{(x_i, y_i)\}_{i=1}^n \quad x_i \in \mathbb{R}^d, \quad y_i \in \{-1, 1\}$

$$\widehat{w}_{MLE} = \arg\max_w \prod_{i=1}^n P(y_i|x_i, w) \qquad P(Y = y|x, w) = \frac{1}{1 + \exp(-y\, w^T x)}$$

$$= \arg\min_w \sum_{i=1}^n \log(1 + \exp(-y_i\, x_i^T w)) = J(w)$$

What does $J(w)$ look like? Is it convex?

Loss function: Conditional Likelihood

- Have a bunch of iid data of the form: $\{(x_i, y_i)\}_{i=1}^{n} \quad x_i \in \mathbb{R}^d, \quad y_i \in \{-1, 1\}$

$$\widehat{w}_{MLE} = \arg\max_w \prod_{i=1}^{n} P(y_i | x_i, w) \qquad P(Y = y | x, w) = \frac{1}{1 + \exp(-y\, w^T x)}$$

$$= \arg\min_w \sum_{i=1}^{n} \log(1 + \exp(-y_i\, x_i^T w)) = J(w)$$

Good news: $J(\mathbf{w})$ is convex function of $\mathbf{w}$, no local optima problems

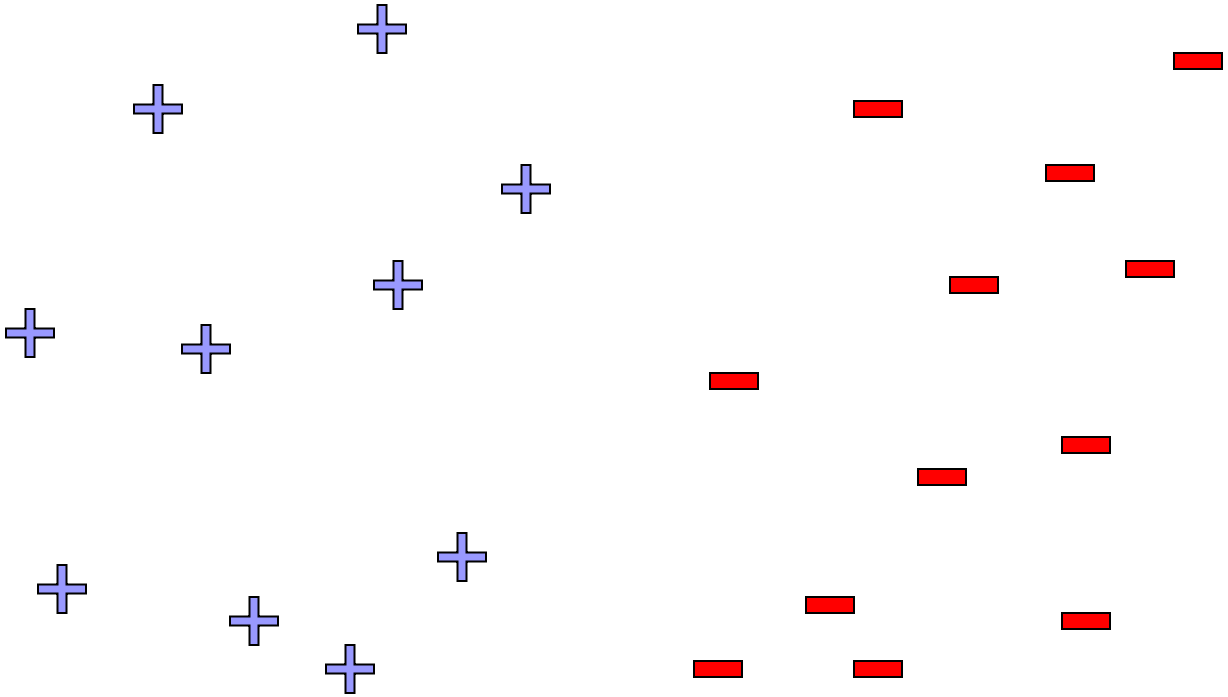
Bad news: no closed-form solution to maximize $J(\mathbf{w})$

Good news: convex functions easy to optimize

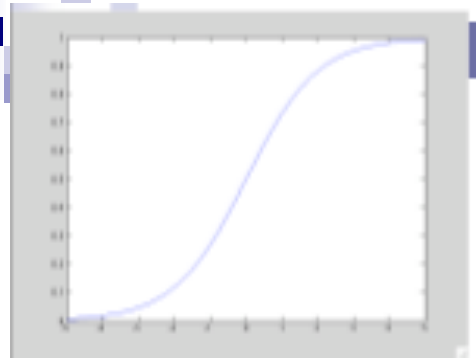
Linear Separability

$$\arg\min_{w} \sum_{i=1}^{n} \log(1 + \exp(-y_i \, x_i^T w))$$

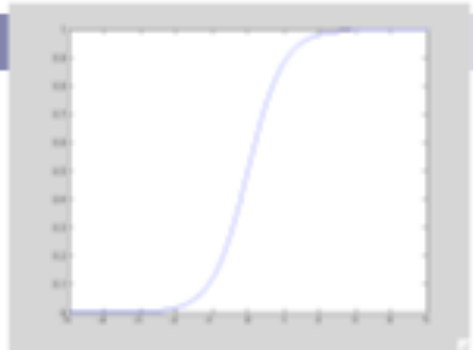
When is this loss small?



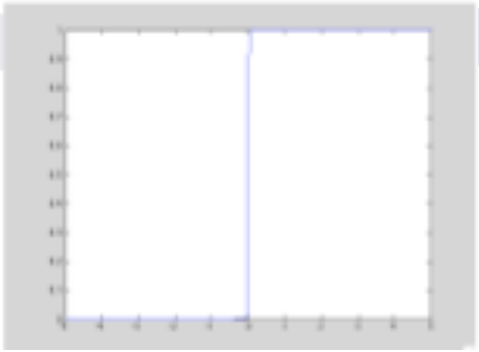
Large parameters → Overfitting



$$\frac{1}{1+e^{-x}}$$



$$\frac{1}{1+e^{-2x}}$$



$$\frac{1}{1+e^{-100x}}$$

- If data is linearly separable, weights go to infinity
  - In general, leads to overfitting:
- Penalizing high weights can prevent overfitting…

Regularized Conditional Log Likelihood

- Add regularization penalty, e.g., L_2:

$$\arg\min_{w} \sum_{i=1}^{n} \log(1 + \exp(-y_i \, x_i^T w)) + \lambda ||w||_2^2$$