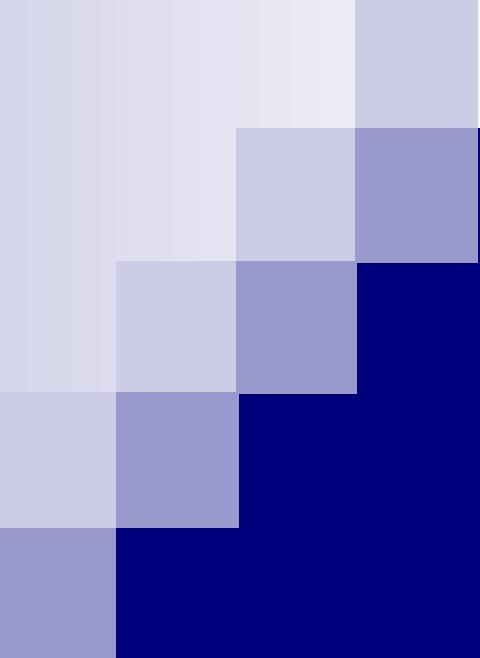


Warm up

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 $\max\{-ax + b, cx\}$



Simple Variable Selection LASSO: Sparse Regression

Machine Learning – CSE546
Kevin Jamieson
University of Washington

April 22, 2019

Sparsity

Many such w vectors will typically work

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

- Vector w is sparse, if many entries are zero

Sparsity

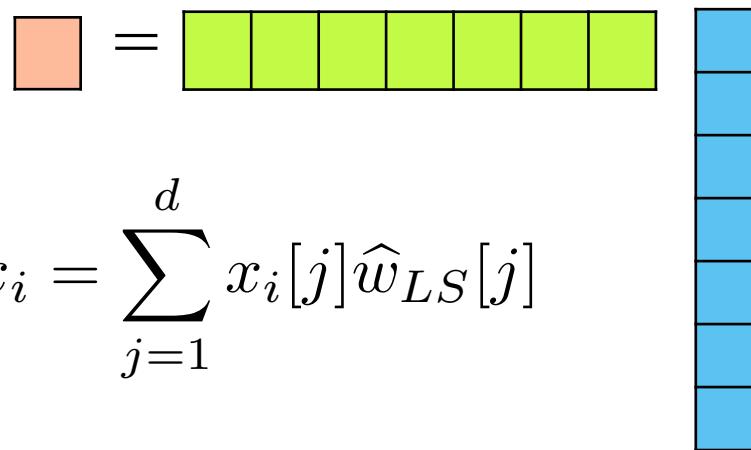
$$\hat{w}_{LS} = \arg \min_w \sum_{i=1}^n (y_i - x_i^T w)^2$$

- Vector w is sparse, if many entries are zero

- Efficiency: If $\text{size}(w) = 100$ Billion, each prediction is expensive:

- If w is sparse, prediction computation only depends on number of non-zeros
store w as pairs mapping index to value rather than an array with mostly zeros:

index 1 = 5
index 7 = 2
index 100 = -1



Sparsity

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

- 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?

These features are representative of the variation in the output. Explain almost all of the variation on their own.
Interpretation: these factors are the ones most important for determining price of the house.

Lot size	Dishwasher
Single Family	Garbage disposal
Year built	Microwave
Last sold price	Range / Oven
Last sale price/sqft	Refrigerator
Finished sqft	Washer
Unfinished sqft	Dryer
Finished basement sqft	Laundry location
# floors	Heating type
Flooring types	Jetted Tub
Parking type	Deck
Parking amount	Fenced Yard
Cooling	Lawn
Heating	Garden
Exterior materials	Sprinkler System
Roof type	
Structure style	

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

Add one variable to system at a time.

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

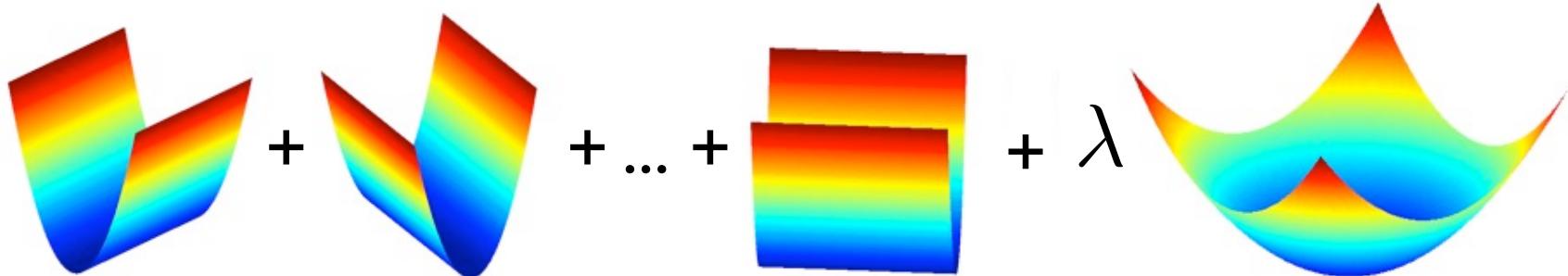
Lots of other variants, too.

NOT consistent across cross-validation, making it difficult to determine which variables are truly most important.

Finding best subset: **Regularize**

Ridge regression makes coefficients small

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

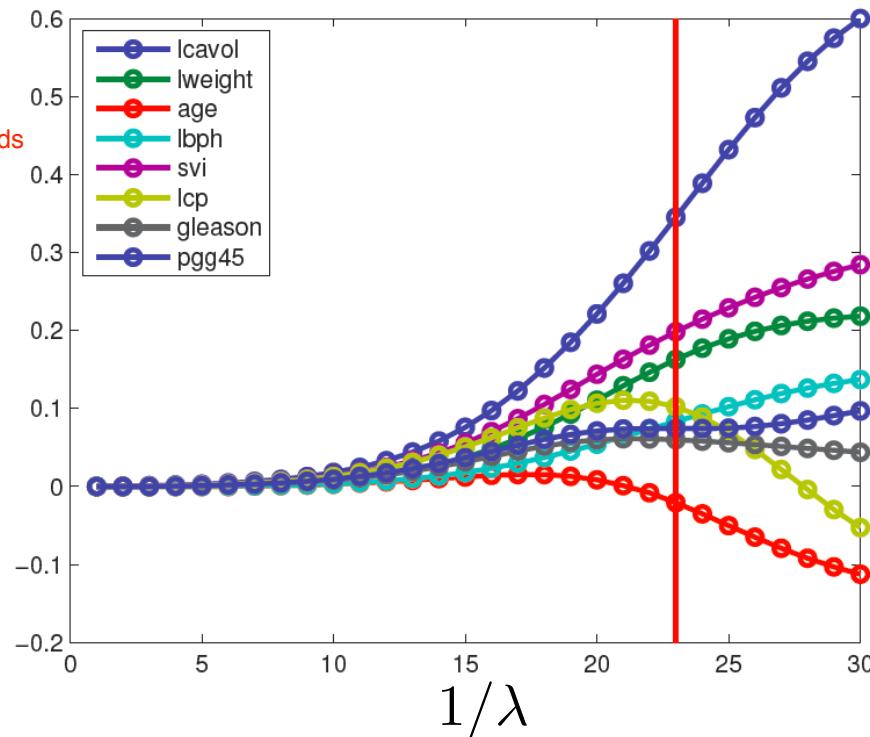


Finding best subset: Regularize

Ridge regression makes coefficients small

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

shrink coefficients towards zero!



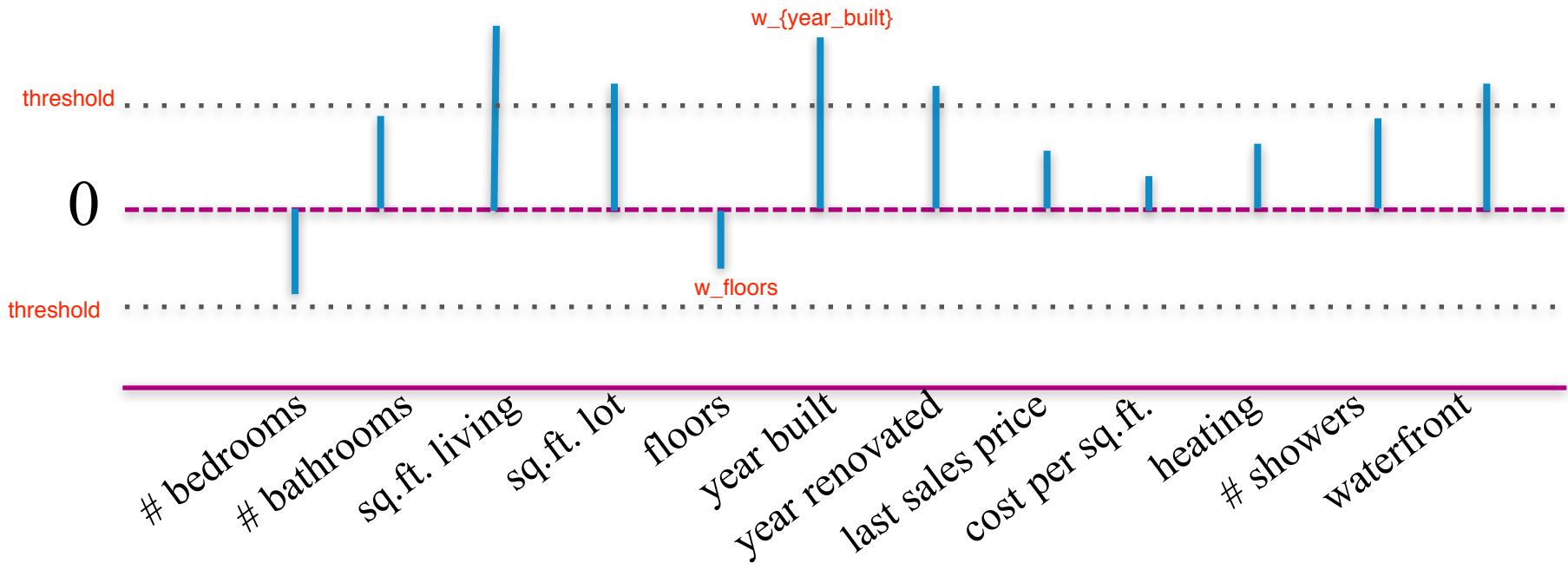
From
Kevin Murphy
textbook

Thresholded Ridge Regression

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

This would certainly help us get the sparsity we are after

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

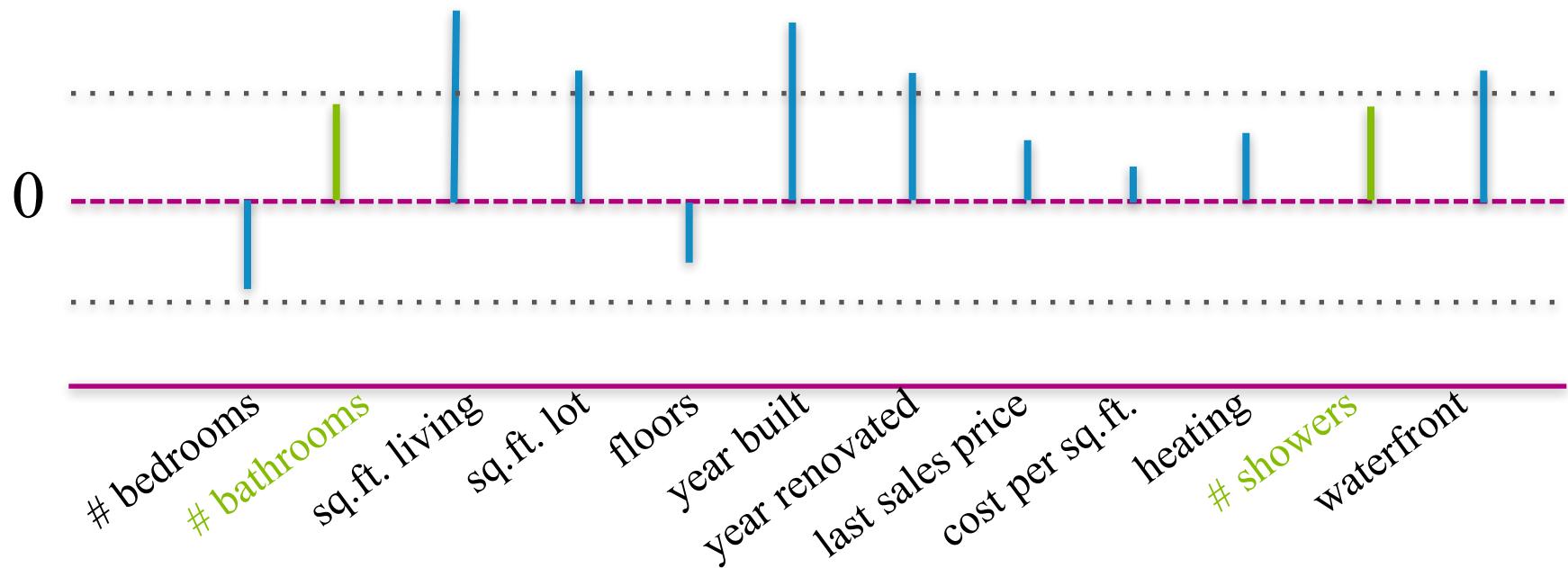


Thresholded Ridge Regression

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

Consider two **related** features (bathrooms, showers)

Since both features are below the threshold, we would throw out both of them and deem they were unimportant.

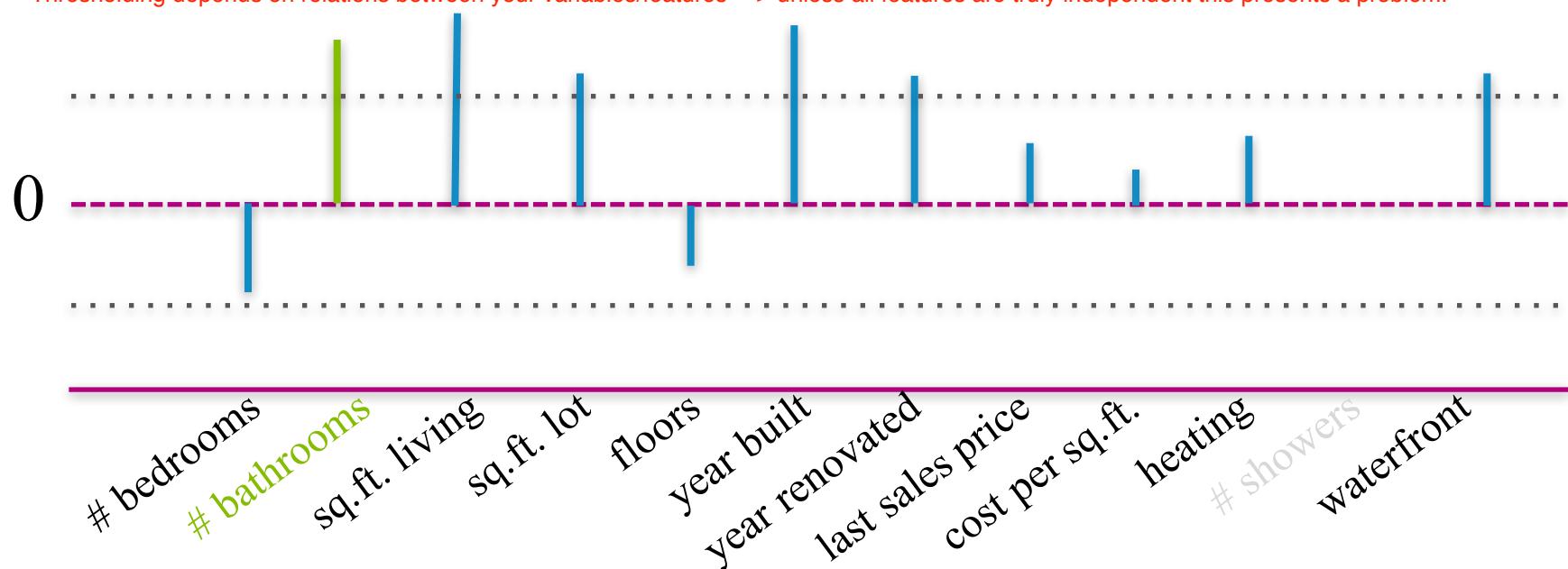


Thresholded Ridge Regression

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

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

Delete the showers variable; now it turns out # bathrooms is an important predictor! Contrary to what we saw on the last slide.
Thresholding depends on relations between your variables/features —> unless all features are truly independent this presents a problem.

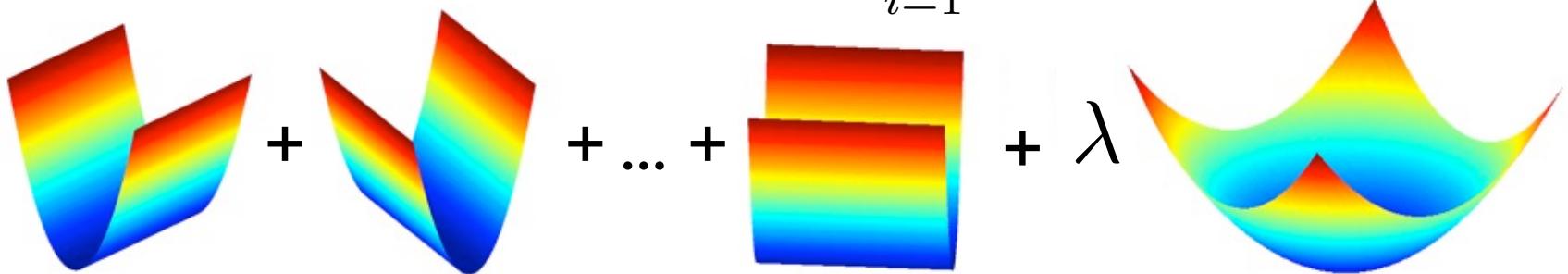


Can another regularizer perform selection automatically?

Recall Ridge Regression

- Ridge Regression objective:

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

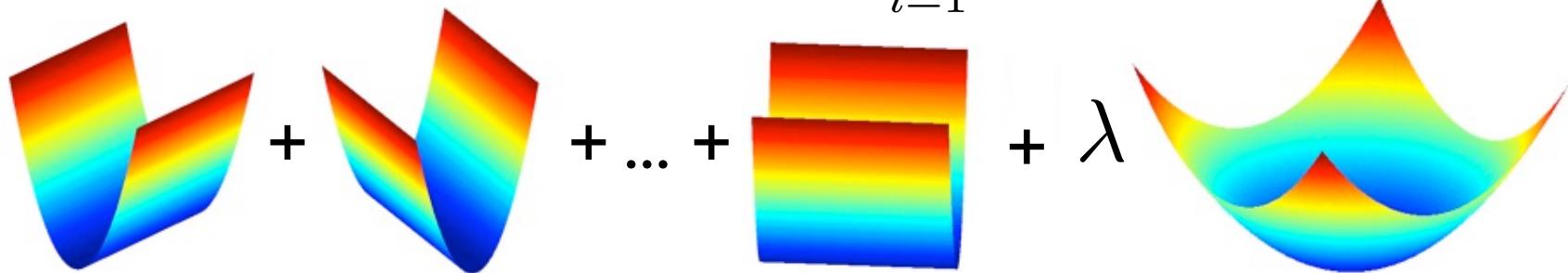


$$\|w\|_p = \left(\sum_{i=1}^d |w|^p \right)^{1/p}$$

Ridge vs. Lasso Regression

- Ridge Regression objective:

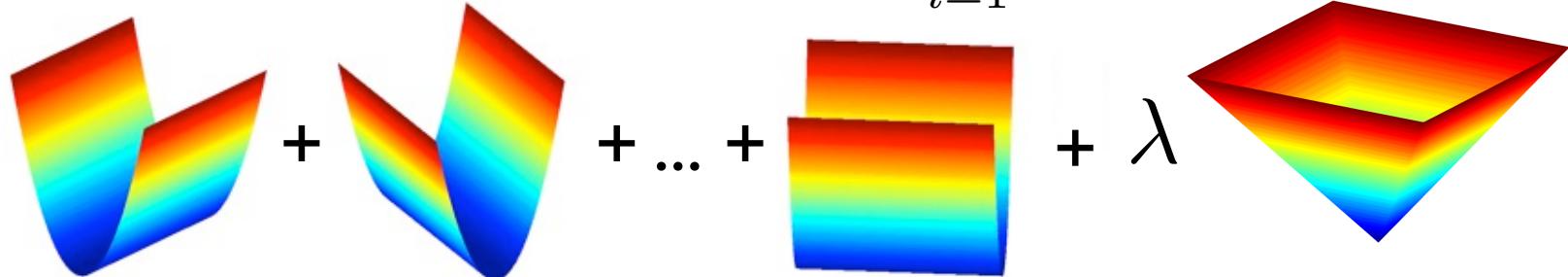
$$\hat{w}_{ridge} = \arg \min_w \sum_{i=1}^n (y_i - x_i^T w)^2 + \lambda ||w||_2^2$$



- Lasso objective:

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

uses different norm to regularize



Penalized Least Squares

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

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

Penalized Least Squares

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

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

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

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

for any convex $r(w)$

only look at w such that $r(w) < \nu$

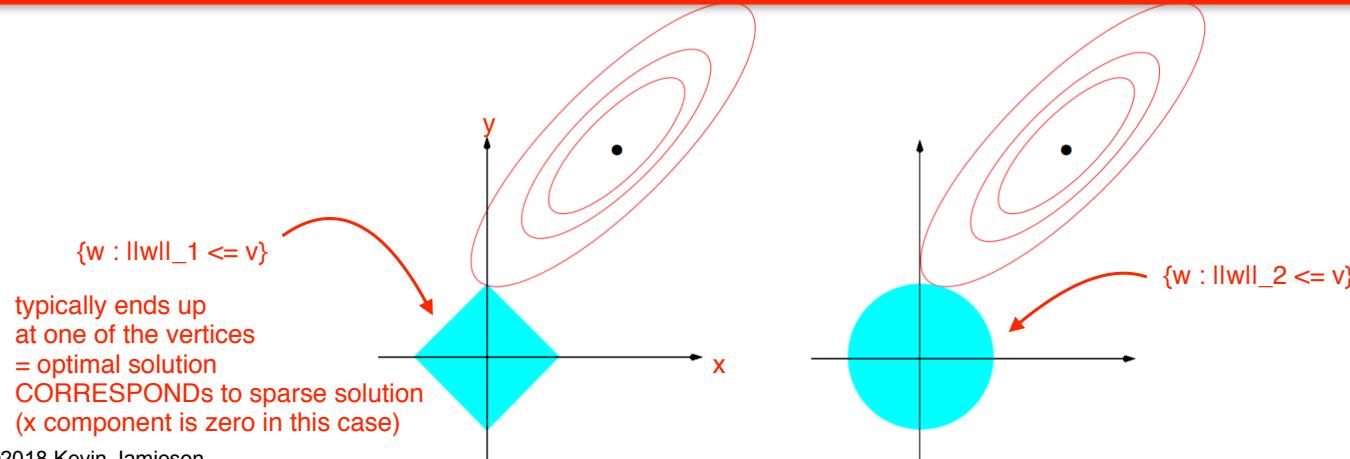
Penalized Least Squares

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

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

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

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



Optimizing the LASSO Objective

- LASSO solution:

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

→ $\hat{b}_{lasso} = \frac{1}{n} \sum_{i=1}^n (y_i - x_i^T \hat{w}_{lasso})$

bias is always zero by demeaning our data, both inputs and outputs

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.

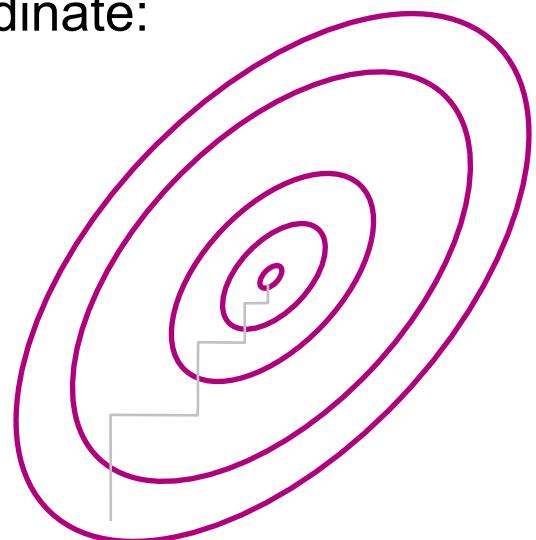
→ $\hat{w}_{lasso} = \arg \min_w \sum_{i=1}^n (y_i - x_i^T w)^2 + \lambda ||w||_1$

NOT differentiable

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:
 - Find minimum one coordinate at a time (assuming our function is convex)
 - Hard to choose step size (eta) in gradient descent as the gradient of L1 norm is constant. Hard to tell when you are reaching near the minimum. How to tell when you stop?
- 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 $j \in \{1, \dots, d\}$

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

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

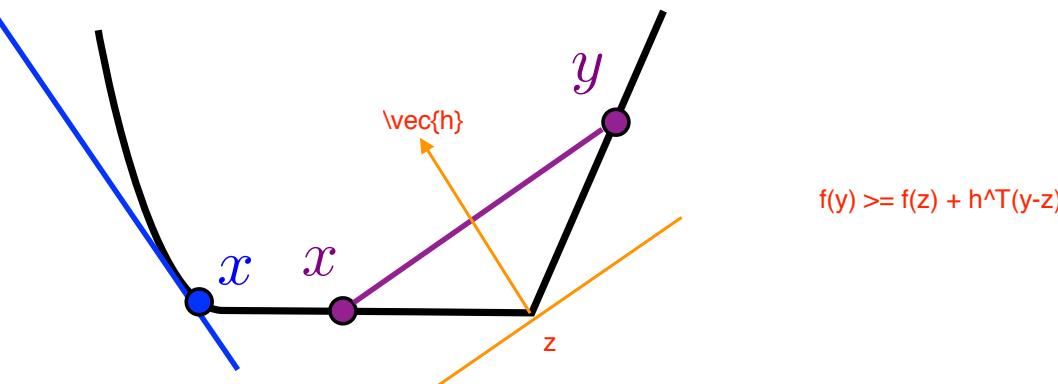
Loop over $j \in \{1, \dots, n\}$: minimize in j th direction.

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

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

Convex Functions

- Equivalent definitions of convexity:



f convex:

line between any two points on the function always overestimates it (lies above function)

$$\text{equivalent definitions for convexity} \quad f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y) \quad \forall x, y, \lambda \in [0, 1]$$

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

gradient always underestimates function requires that the gradient is defined everywhere.

- 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

Now, function does not need to be differentiable anywhere. Can optimize over functions that are differentiable almost everywhere

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

g does not have to be same for all x; but for any x there exists g is NOT necessarily unique.

range of gradients between gradient limit coming from right and left

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

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

$$\partial_{w_j} |w_j| = \begin{cases} + & \text{if } w > 0 \\ \text{anything in } [-1,1] & \text{if } w = 0 \\ - & \text{if } w < 0 \end{cases}$$

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

Setting Subgradient to 0

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

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

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

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

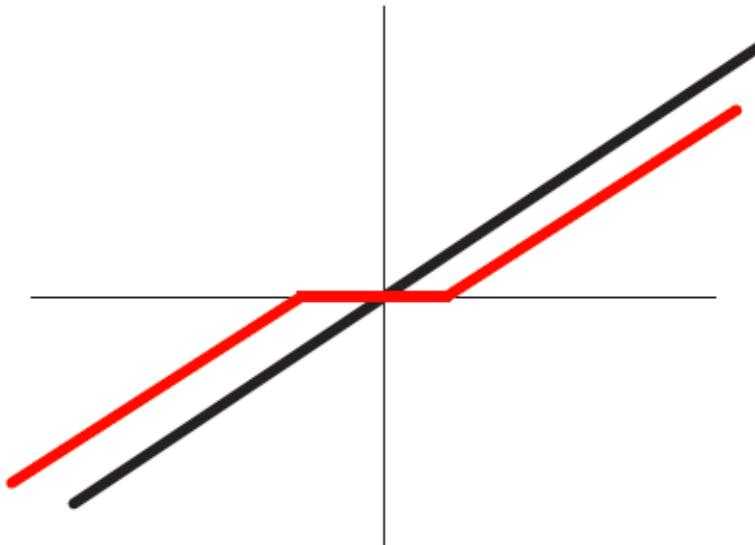
$$\widehat{w}_j = \begin{cases} (c_j + \lambda)/a_j & \text{if } c_j < -\lambda \\ 0 & \text{if } |c_j| \leq \lambda \\ (c_j - \lambda)/a_j & \text{if } c_j > \lambda \end{cases}$$

Soft Thresholding

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

$$a_j = \sum_{i=1}^n x_{i,j}^2$$

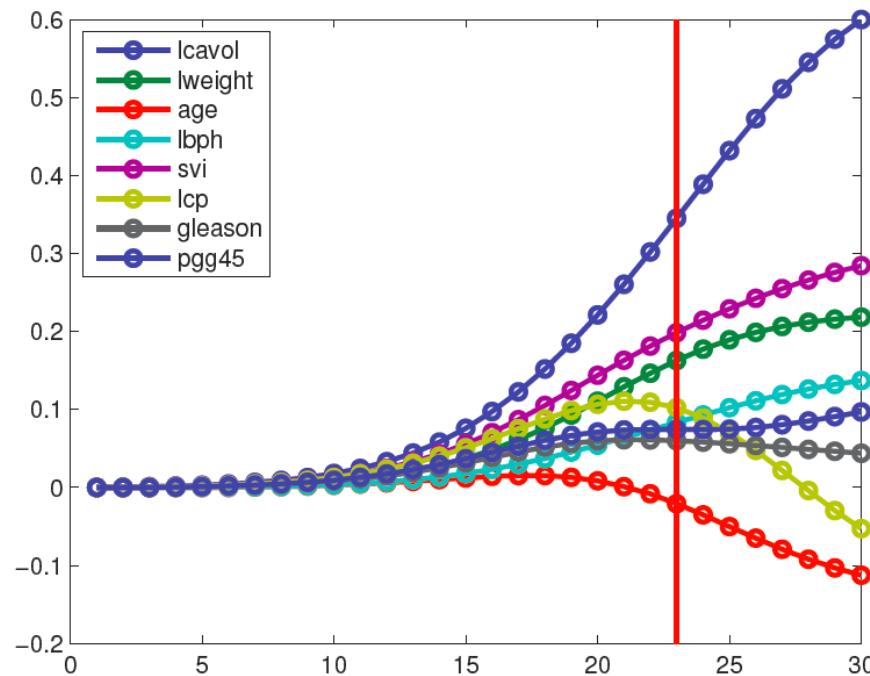
$$c_j = 2 \sum_{i=1}^n \left(y_i - \sum_{k \neq j} x_{i,k} w_k \right) x_{i,j}$$



Coordinate Descent for LASSO (aka Shooting Algorithm)

- Repeat until convergence
 - Pick a coordinate j at (random or sequentially)
 - Set:
$$\hat{w}_j = \begin{cases} (c_j + \lambda)/a_j & \text{if } c_j < -\lambda \\ 0 & \text{if } |c_j| \leq \lambda \\ (c_j - \lambda)/a_j & \text{if } c_j > \lambda \end{cases}$$
 - Where:
$$a_j = \sum_{i=1}^n x_{i,j}^2 \quad c_j = 2 \sum_{i=1}^n \left(y_i - \sum_{k \neq j} x_{i,k} w_k \right) x_{i,j}$$
 - For convergence rates, see Shalev-Shwartz and Tewari 2009
- Other common technique = LARS
 - Least angle regression and shrinkage, Efron et al. 2004

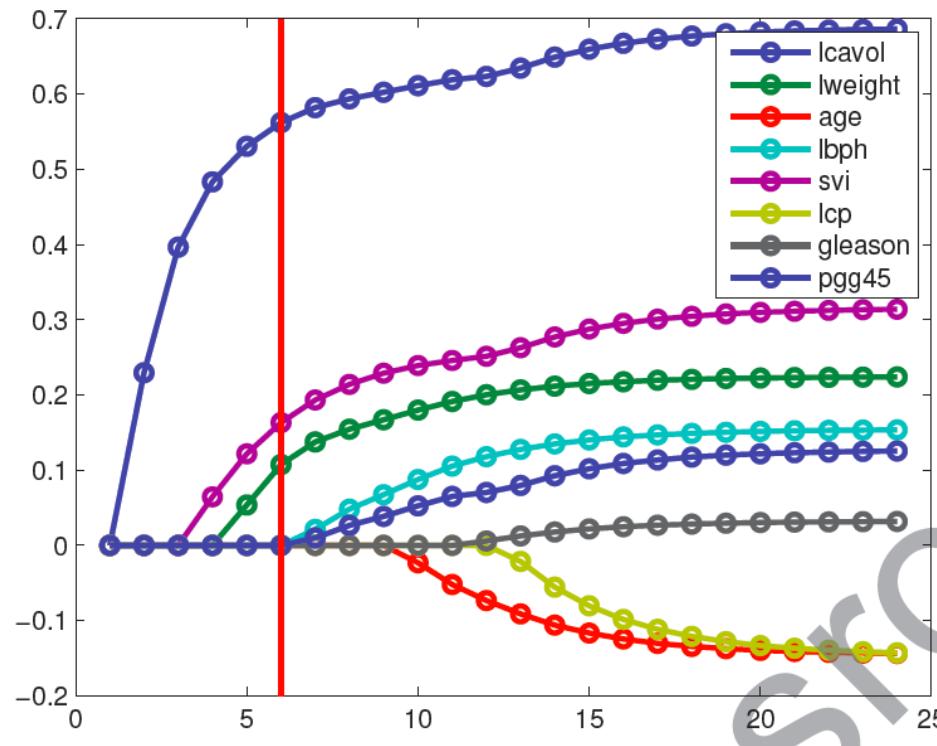
Recall: *Ridge Coefficient Path*



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
- Lasso (or L1 regularization) is just one way to find sparse solutions
- **Be careful about interpreting selected features**
 - sensitive to correlations between features
 - result depends on algorithm used (with same loss function)
 - under certain conditions theoretical guarantees exist for lasso

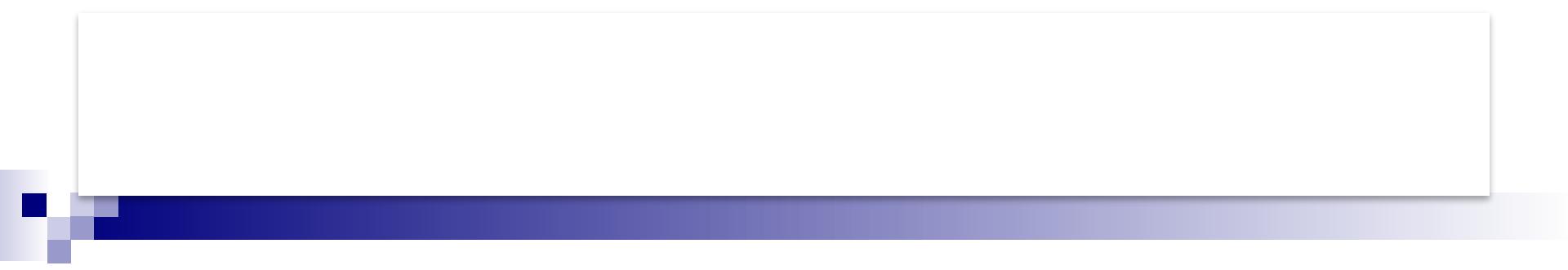
FINISH LECT 7 MON Apr 22



Classification Logistic Regression

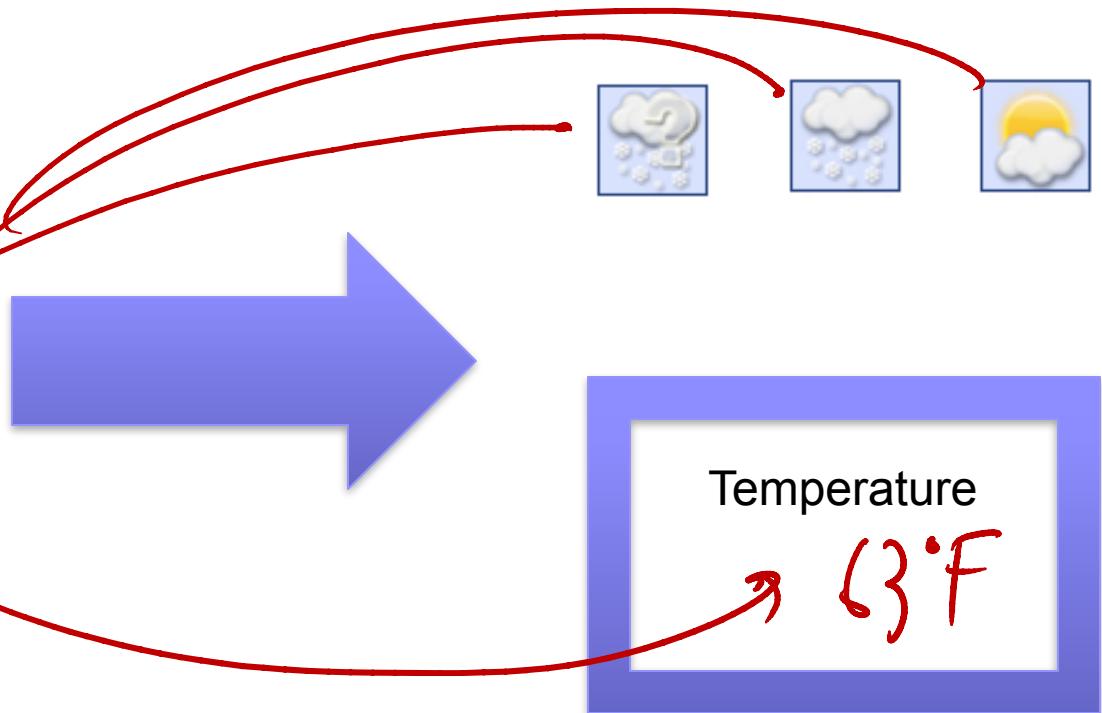
Machine Learning – CSE546
Kevin Jamieson
University of Washington

April 22, 2019



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

Weather prediction revisited

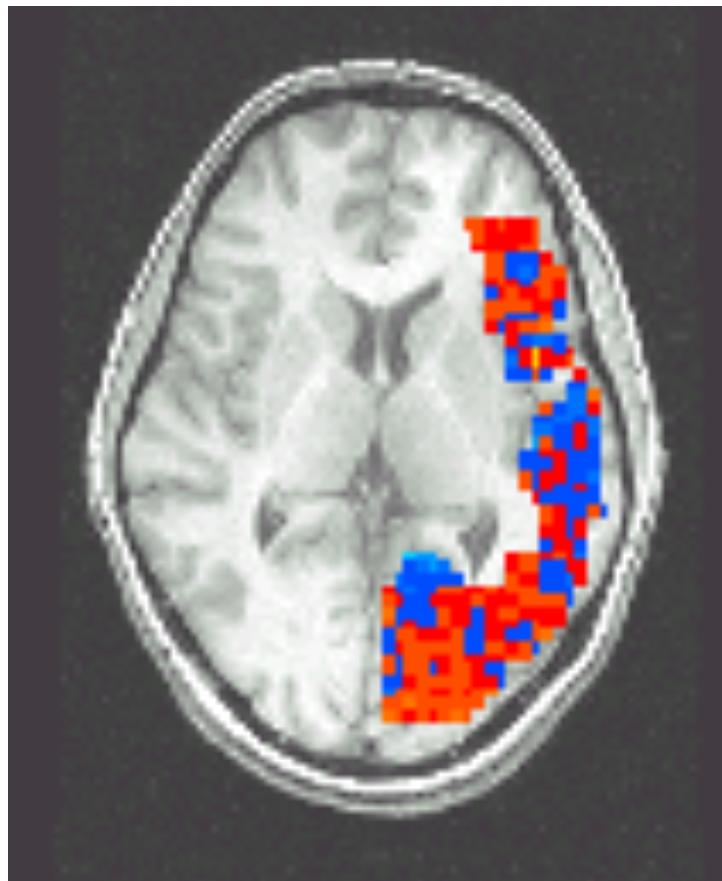


Reading Your Brain, Simple Example

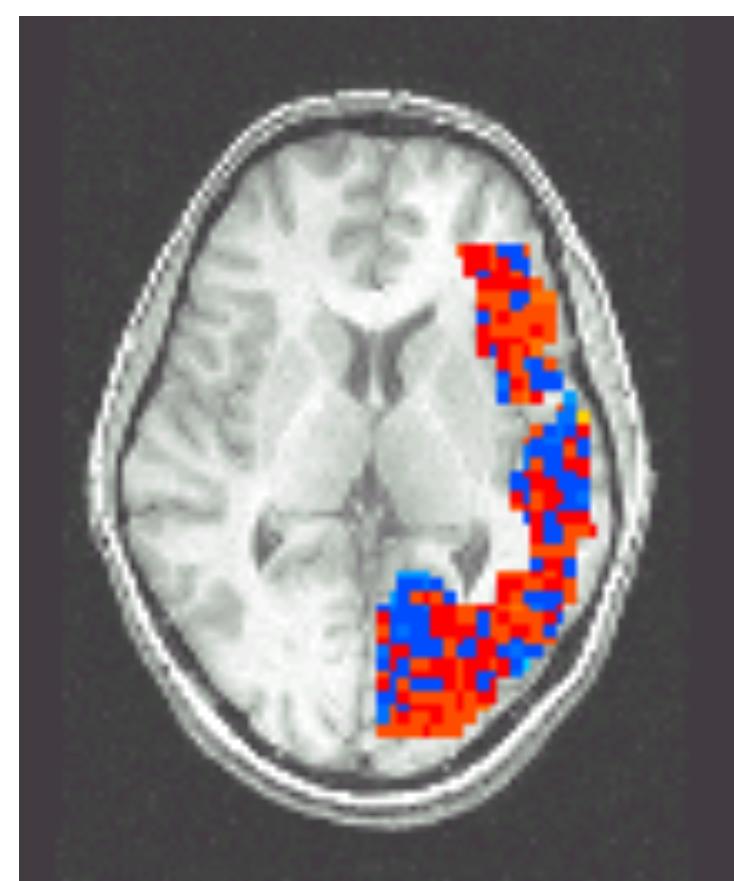
[Mitchell et al.]

Pairwise classification accuracy: 85%

Person



Animal



Classification

- **Learn:** $f: \mathbf{X} \rightarrow \mathbf{Y}$
 - \mathbf{X} – features
 - \mathbf{Y} – target classes $Y \in \{1, 2, \dots, k\}$
- **Loss function:**
- **Expected loss of f :**
- Suppose you know $P(Y|\mathbf{X})$ exactly, how should you classify?
 - Bayes optimal classifier:

Classification

- **Learn:** $f: \mathbf{X} \rightarrow \mathbf{Y}$
 - \mathbf{X} – features
 - \mathbf{Y} – target classes $Y \in \{1, 2, \dots, k\}$
- **Loss function:** $\ell(f(x), y) = \mathbf{1}\{f(x) \neq y\}$
- **Expected loss of f:**
$$\mathbb{E}_{XY}[\mathbf{1}\{f(X) \neq Y\}] = \mathbb{E}_X[\mathbb{E}_{Y|X}[\mathbf{1}\{f(x) \neq Y\}|X = x]]$$
$$\begin{aligned}\mathbb{E}_{Y|X}[\mathbf{1}\{f(x) \neq Y\}|X = x] &= \sum_i P(Y = i|X = x)\mathbf{1}\{f(x) \neq i\} = \sum_{i \neq f(x)} P(Y = i|X = x) \\ &= 1 - P(Y = f(x)|X = x)\end{aligned}$$
- Suppose you know $P(\mathbf{Y}|\mathbf{X})$ exactly, how should you classify?
 - Bayes optimal classifier:

$$f(x) = \arg \max_y \mathbb{P}(Y = y|X = x)$$

Binary Classification

- **Learn:** $f: \mathbf{X} \rightarrow \mathbf{Y}$
 - \mathbf{X} – features
 - \mathbf{Y} – target classes

$$Y \in \{0, 1\}$$

- **Loss function:** $\ell(f(x), y) = \mathbf{1}\{f(x) \neq y\}$
- **Expected loss of f:**

$$\mathbb{E}_{XY}[\mathbf{1}\{f(X) \neq Y\}] = \mathbb{E}_X[\mathbb{E}_{Y|X}[\mathbf{1}\{f(x) \neq Y\}|X = x]]$$

$$\begin{aligned}\mathbb{E}_{Y|X}[\mathbf{1}\{f(x) \neq Y\}|X = x] &= \sum_i P(Y = i|X = x)\mathbf{1}\{f(x) \neq i\} = \sum_{i \neq f(x)} P(Y = i|X = x) \\ &= 1 - P(Y = f(x)|X = x)\end{aligned}$$

- Suppose you know $P(Y|\mathbf{X})$ exactly, how should you classify?
 - Bayes optimal classifier:

$$f(x) = \arg \max_y \mathbb{P}(Y = y|X = x)$$

Link Functions

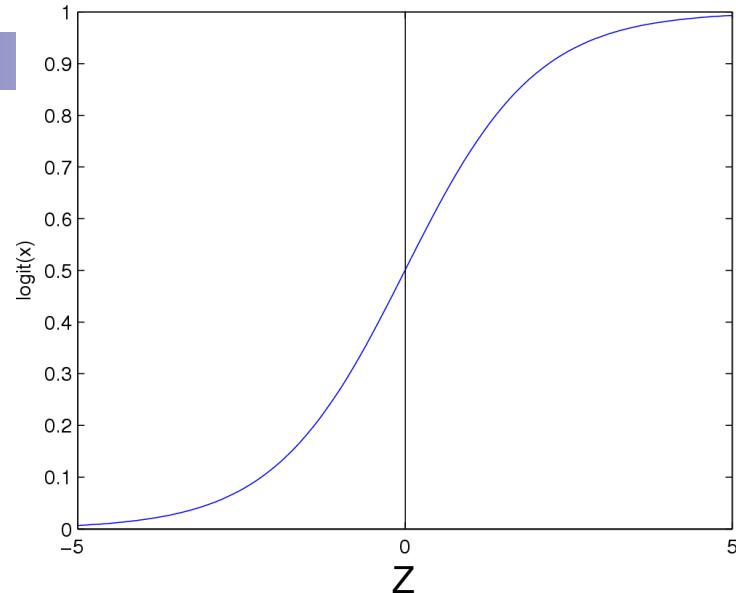
- Estimating $P(Y|X)$: Why not use standard linear regression?
- Combining regression and probability?
 - Need a mapping from real values to $[0,1]$
 - A link function!

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:

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

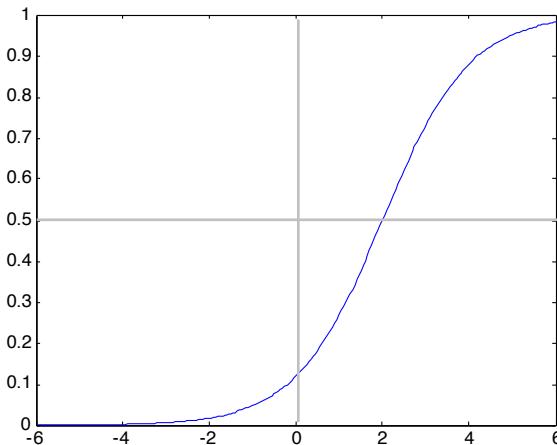


Features can be discrete or continuous!

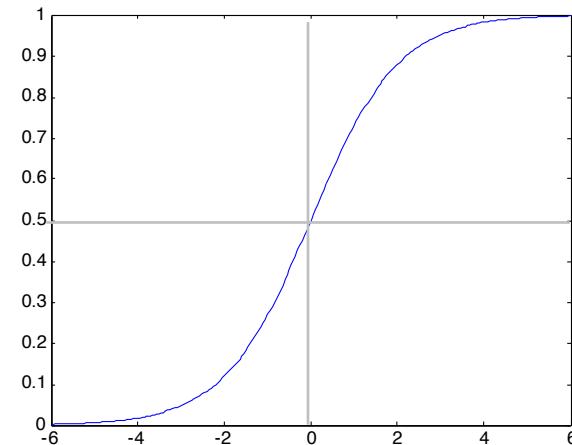
Understanding the sigmoid

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

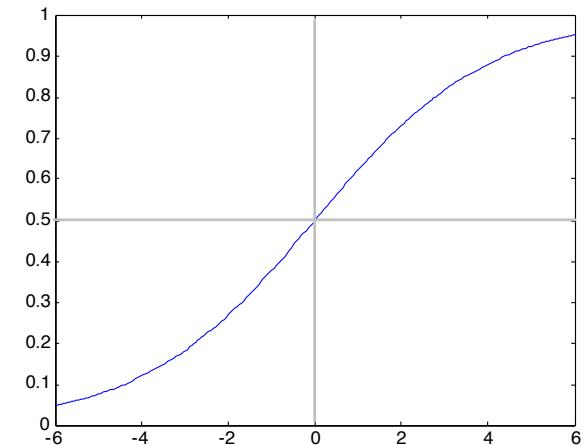
$w_0 = -2, w_1 = -1$



$w_0 = 0, w_1 = -1$



$w_0 = 0, w_1 = -0.5$



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

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

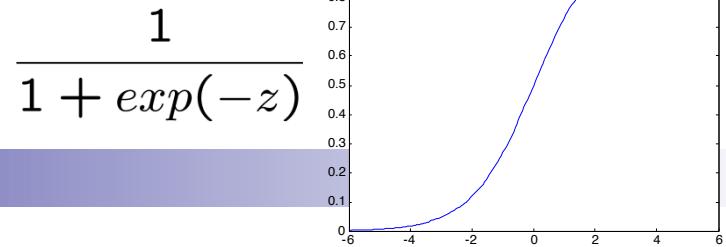
$$\mathbb{P}(Y = 1|w, X) = 1 - \mathbb{P}(Y = 0|w, X) = \frac{\exp(w_0 + \sum_k w_k X_k)}{1 + \exp(w_0 + \sum_k w_k X_k)}$$

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

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

Linear Decision Rule!

Logistic Regression – a Linear classifier



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

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

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:

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

Loss function: Conditional Likelihood

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

$$\begin{aligned}\widehat{w}_{MLE} &= \arg \max_w \prod_{i=1}^n P(y_i | x_i, w) \quad P(Y = y | x, w) = \frac{1}{1 + \exp(-y w^T x)} \\ &= \arg \min_w \sum_{i=1}^n \log(1 + \exp(-y_i x_i^T w))\end{aligned}$$

Loss function: Conditional Likelihood

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

$$\begin{aligned}\widehat{w}_{MLE} &= \arg \max_w \prod_{i=1}^n P(y_i | x_i, w) \quad P(Y = y | x, w) = \frac{1}{1 + \exp(-y w^T x)} \\ &= \arg \min_w \sum_{i=1}^n \log(1 + \exp(-y_i x_i^T w))\end{aligned}$$

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

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

Loss function: Conditional Likelihood

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

$$\begin{aligned}\widehat{w}_{MLE} &= \arg \max_w \prod_{i=1}^n P(y_i | x_i, w) \quad P(Y = y | x, w) = \frac{1}{1 + \exp(-y w^T x)} \\ &= \arg \min_w \sum_{i=1}^n \log(1 + \exp(-y_i x_i^T w)) = J(w)\end{aligned}$$

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

Loss function: Conditional Likelihood

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

$$\begin{aligned}\widehat{w}_{MLE} &= \arg \max_w \prod_{i=1}^n P(y_i|x_i, w) \quad P(Y = y|x, w) = \frac{1}{1 + \exp(-y w^T x)} \\ &= \arg \min_w \sum_{i=1}^n \log(1 + \exp(-y_i x_i^T w)) = J(w)\end{aligned}$$

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

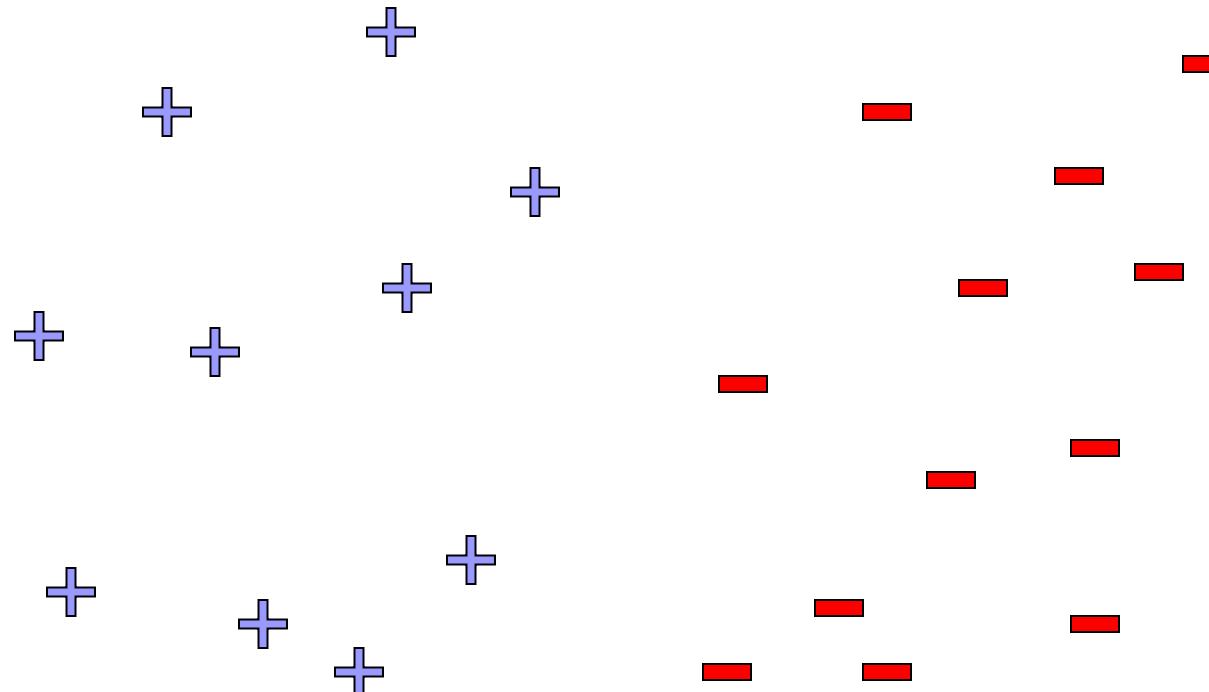
Bad news: no closed-form solution to maximize $J(\mathbf{w})$

Good news: convex functions easy to optimize

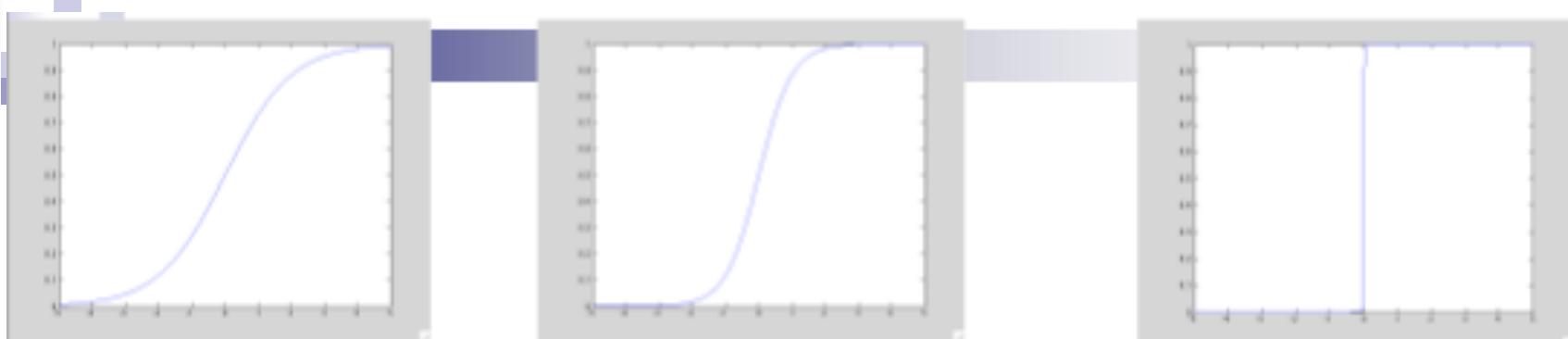
Linear Separability

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

When is this loss small?



Large parameters \rightarrow Overfitting



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

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

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

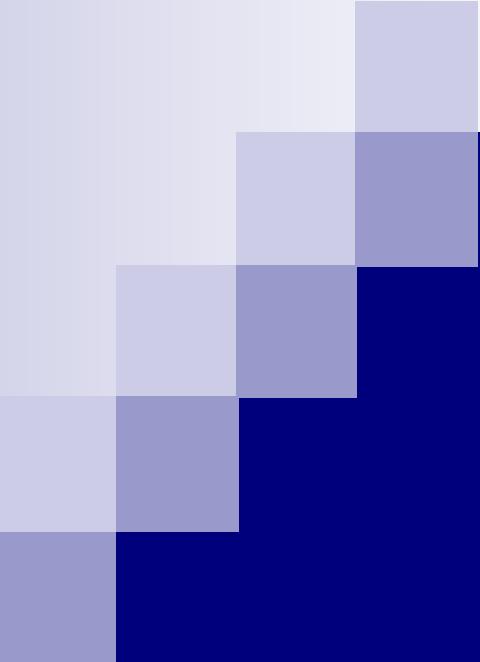
- If data is linearly separable, weights go to infinity
 - In general, leads to overfitting:
- Penalizing high weights can prevent overfitting...

Regularized Conditional Log Likelihood

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

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

Be sure to not regularize the offset b !



Gradient Descent

Machine Learning – CSE546
Kevin Jamieson
University of Washington

April 22, 2019

Machine Learning Problems

- Have a bunch of iid data of the form:

$$\{(x_i, y_i)\}_{i=1}^n \quad x_i \in \mathbb{R}^d \quad y_i \in \mathbb{R}$$

- Learning a model's parameters:

Each $\ell_i(w)$ is convex.

$$\sum_{i=1}^n \ell_i(w)$$

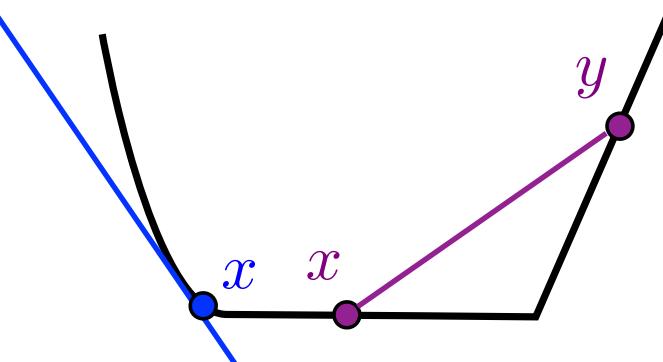
Machine Learning Problems

- Have a bunch of iid data of the form:

$$\{(x_i, y_i)\}_{i=1}^n \quad x_i \in \mathbb{R}^d \quad y_i \in \mathbb{R}$$

- Learning a model's parameters:
Each $\ell_i(w)$ is convex.

$$\sum_{i=1}^n \ell_i(w)$$



f convex:

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

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

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

Machine Learning Problems

- Have a bunch of iid data of the form:

$$\{(x_i, y_i)\}_{i=1}^n \quad x_i \in \mathbb{R}^d \quad y_i \in \mathbb{R}$$

- Learning a model's parameters:

Each $\ell_i(w)$ is convex.

$$\sum_{i=1}^n \ell_i(w)$$

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

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

Least squares

- Have a bunch of iid data of the form:

$$\{(x_i, y_i)\}_{i=1}^n \quad x_i \in \mathbb{R}^d \quad y_i \in \mathbb{R}$$

- Learning a model's parameters:

Each $\ell_i(w)$ is convex.

$$\sum_{i=1}^n \ell_i(w)$$

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

How does software solve: $\frac{1}{2} ||\mathbf{X}w - \mathbf{y}||_2^2$

Least squares

- Have a bunch of iid data of the form:

$$\{(x_i, y_i)\}_{i=1}^n \quad x_i \in \mathbb{R}^d \quad y_i \in \mathbb{R}$$

- Learning a model's parameters:

Each $\ell_i(w)$ is convex.

$$\sum_{i=1}^n \ell_i(w)$$

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

How does software solve:

$$\frac{1}{2} ||\mathbf{X}w - \mathbf{y}||_2^2$$

...its complicated:
(LAPACK, BLAS, MKL...)

Do you need high precision?
Is \mathbf{X} column/row sparse?
Is \hat{w}_{LS} sparse?
Is $\mathbf{X}^T \mathbf{X}$ “well-conditioned”?
Can $\mathbf{X}^T \mathbf{X}$ fit in cache/memory?

Taylor Series Approximation

- Taylor series in one dimension:

$$f(x + \delta) = f(x) + f'(x)\delta + \frac{1}{2}f''(x)\delta^2 + \dots$$

- Gradient descent:

Taylor Series Approximation

- Taylor series in **d** dimensions:

$$f(x + v) = f(x) + \nabla f(x)^T v + \frac{1}{2}v^T \nabla^2 f(x)v + \dots$$

- Gradient descent:

Gradient Descent

$$f(w) = \frac{1}{2} \|Xw - y\|_2^2$$


$$w_{t+1} = w_t - \eta \nabla f(w_t)$$

$$\nabla f(w) =$$

$$w_{t+1} = w_t - \eta \nabla f(w_t) \qquad w_* = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

$$w_{t+1} - w_* =$$

Gradient Descent

$$f(w) = \frac{1}{2} \| \mathbf{X}w - \mathbf{y} \|_2^2$$

$$w_{t+1} = w_t - \eta \nabla f(w_t)$$

$$\nabla f(w) = \mathbf{X}^T(\mathbf{X}w - \mathbf{y}) = \mathbf{X}^T\mathbf{X}w - \mathbf{X}^T\mathbf{y}$$

$$w_{t+1} = w_t - \eta \nabla f(w_t) \qquad \qquad \qquad w_* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

$$= (I - \eta \mathbf{X}^T\mathbf{X})w_t + \eta \mathbf{X}^T\mathbf{y}$$

$$(w_{t+1} - w_*) = (I - \eta \mathbf{X}^T\mathbf{X})(w_t - w_*) - \eta \mathbf{X}^T\mathbf{X}w_* + \eta \mathbf{X}^T\mathbf{y}$$

Gradient Descent

$$f(w) = \frac{1}{2} \|Xw - y\|_2^2$$

$$w_{t+1} = w_t - \eta \nabla f(w_t)$$

$$\begin{aligned}(w_{t+1} - w_*) &= (I - \eta X^T X)(w_t - w_*) \\ &= (I - \eta X^T X)^{t+1}(w_0 - w_*)\end{aligned}$$

Example: $X = \begin{bmatrix} 10^{-3} & 0 \\ 0 & 1 \end{bmatrix}$ $y = \begin{bmatrix} 10^{-3} \\ 1 \end{bmatrix}$ $w_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ $w_* =$

Taylor Series Approximation

- Taylor series in one dimension:

$$f(x + \delta) = f(x) + f'(x)\delta + \frac{1}{2}f''(x)\delta^2 + \dots$$

- Newton's method:

Taylor Series Approximation

- Taylor series in **d** dimensions:

$$f(x + v) = f(x) + \nabla f(x)^T v + \frac{1}{2}v^T \nabla^2 f(x)v + \dots$$

- Newton's method:

Newton's Method

$$f(w) = \frac{1}{2} \|Xw - y\|_2^2$$

$$\nabla f(w) =$$

$$\nabla^2 f(w) =$$

v_t is solution to : $\nabla^2 f(w_t)v_t = -\nabla f(w_t)$

$$w_{t+1} = w_t + \eta v_t$$

Newton's Method

$$f(w) = \frac{1}{2} \|Xw - y\|_2^2$$

$$\nabla f(w) = X^T(Xw - y)$$

$$\nabla^2 f(w) = X^T X$$

v_t is solution to : $\nabla^2 f(w_t)v_t = -\nabla f(w_t)$

$$w_{t+1} = w_t + \eta v_t$$

For quadratics, Newton's method converges in one step! (Not a surprise, why?)

$$w_1 = w_0 - \eta(X^T X)^{-1} X^T (Xw_0 - y) = w_*$$

General case

In general for Newton's method to achieve $f(w_t) - f(w_*) \leq \epsilon$:

So why are ML problems overwhelmingly solved by gradient methods?

Hint: v_t is solution to : $\nabla^2 f(w_t)v_t = -\nabla f(w_t)$

General Convex case

$$f(w_t) - f(w_*) \leq \epsilon$$

Newton's method:

$$t \approx \log(\log(1/\epsilon))$$

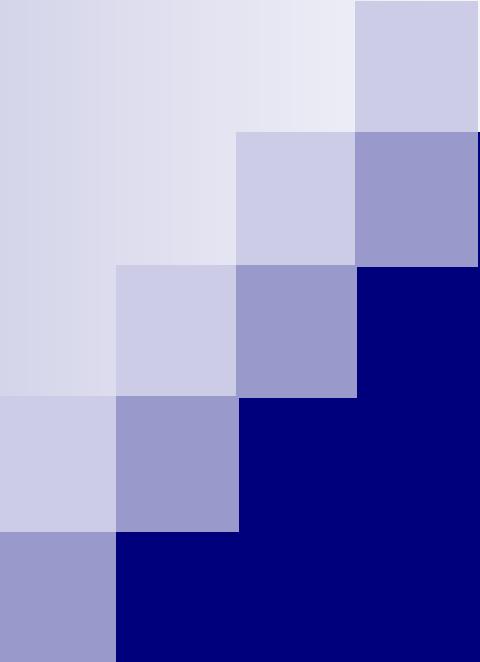
Gradient descent:

- f is *smooth* and *strongly convex*: $aI \preceq \nabla^2 f(w) \preceq bI$
- f is *smooth*: $\nabla^2 f(w) \preceq bI$
- f is potentially non-differentiable: $||\nabla f(w)||_2 \leq c$

Other: BFGS, Heavy-ball, BCD, SVRG, ADAM, Adagrad,...

Clean
converge
nice
proofs:
Bubeck

Nocedal
+Wright,
Bubeck



Revisiting... Logistic Regression

Machine Learning – CSE546
Kevin Jamieson
University of Washington

April 22, 2019

Loss function: Conditional Likelihood

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

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

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

$$\nabla f(w) =$$