CSE 446/546
Lec 6: LASSO

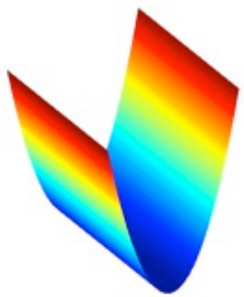
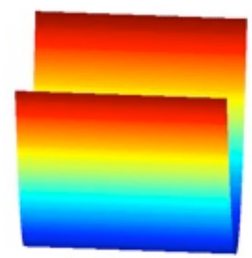
Matt Golub
Hunter Schafer

UNIVERSITY of WASHINGTON

Ridge Regression

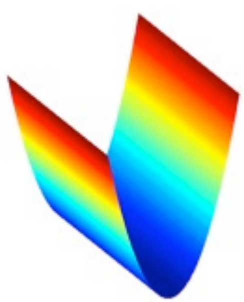
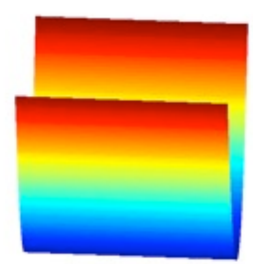
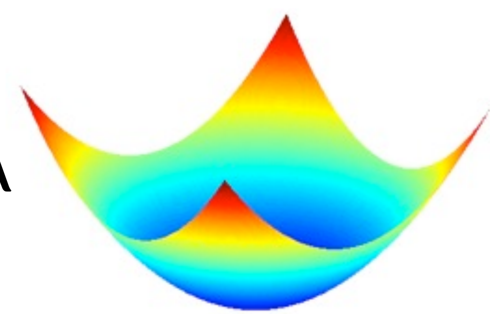
- Old Least squares objective:

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

 +  + ... + 

- Ridge Regression objective:

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

 +  + ... +  + λ 

Shrinkage Properties

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

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

Bias-Variance Properties

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

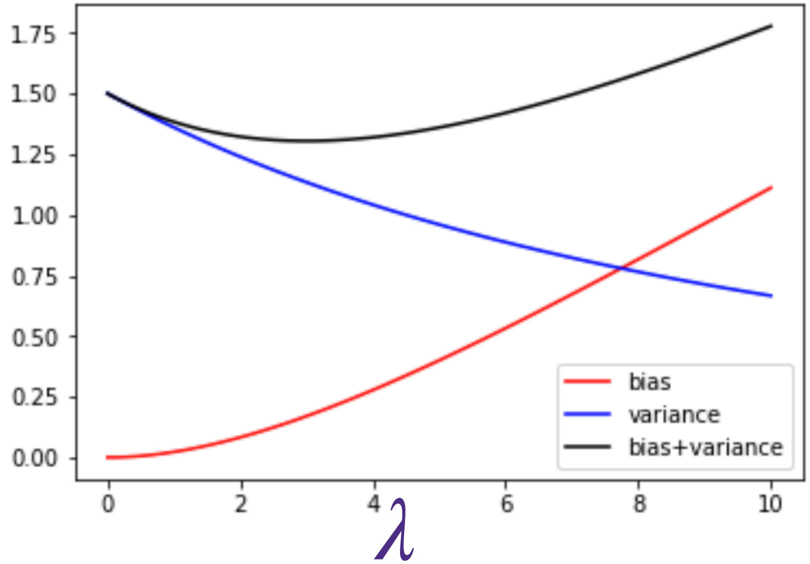
- ***Assume:*** $\mathbf{X}^T\mathbf{X} = nI$ ***and*** $\mathbf{y} = \mathbf{X}w + \boldsymbol{\epsilon}$ $\quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma^2 I)$

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

$$\mathbb{E}_{Y|X,\mathcal{D}}[(Y - x^T\widehat{w}_{ridge})^2 | X = x]$$

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

(verify at home)



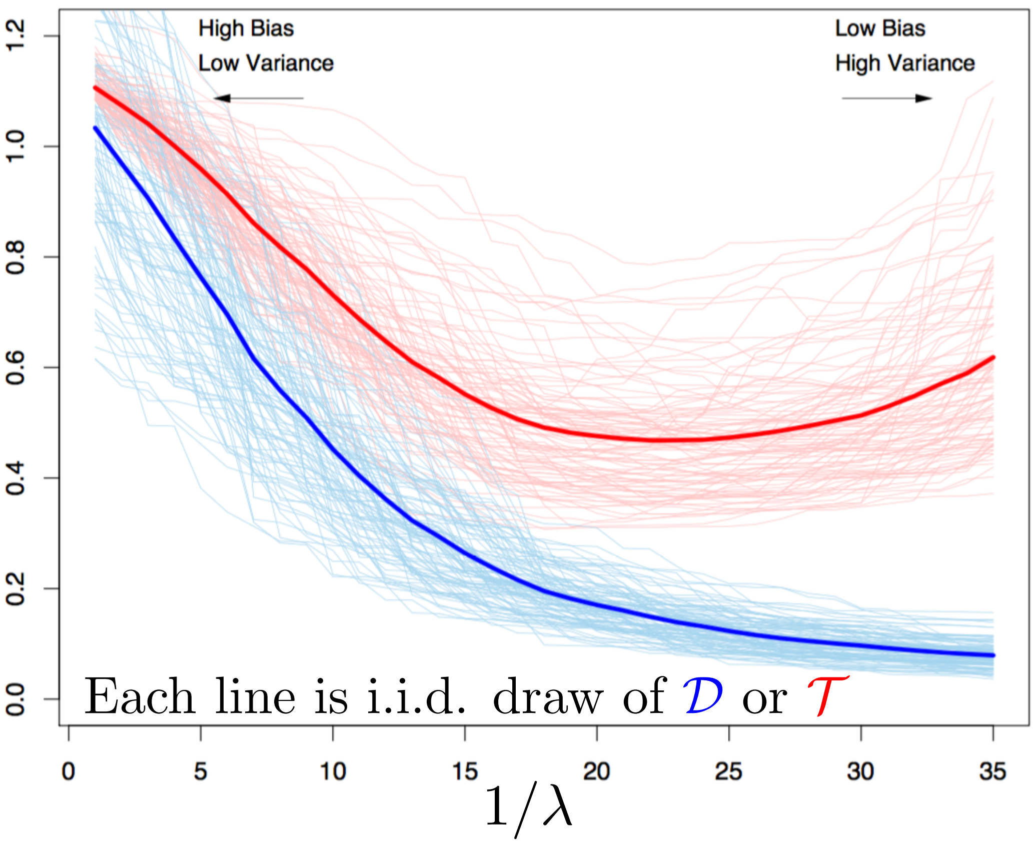
d=10, n=20,

$\sigma^2 = 3.0, \|w\|_2^2 = 10$

Ridge Regression: Effect of Regularization

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

$$\widehat{w}^{(\lambda)}_{\mathcal{D},ridge} = \arg\min_w \frac{1}{|\mathcal{D}|} \sum_{(x_i,y_i)\in\mathcal{D}} (y_i - x_i^T w)^2 + \lambda ||w||_2^2$$



TRAIN error:

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

TRUE error:

$$\mathbb{E}[(Y - X^T \widehat{w}^{(\lambda)}_{\mathcal{D},ridge})^2]$$

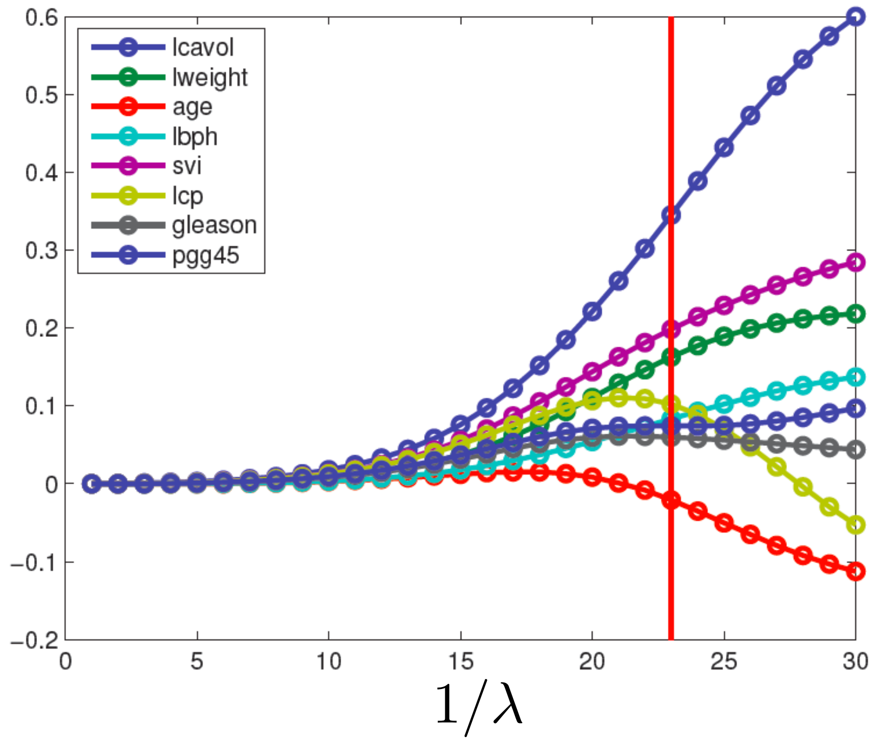
TEST error:

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

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

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

Ridge regression: minimize $\sum_{i=1}^{n}(w^T x_i + b - y_i)^2 + \lambda \|w\|_2^2$



From
Kevin Murphy
textbook

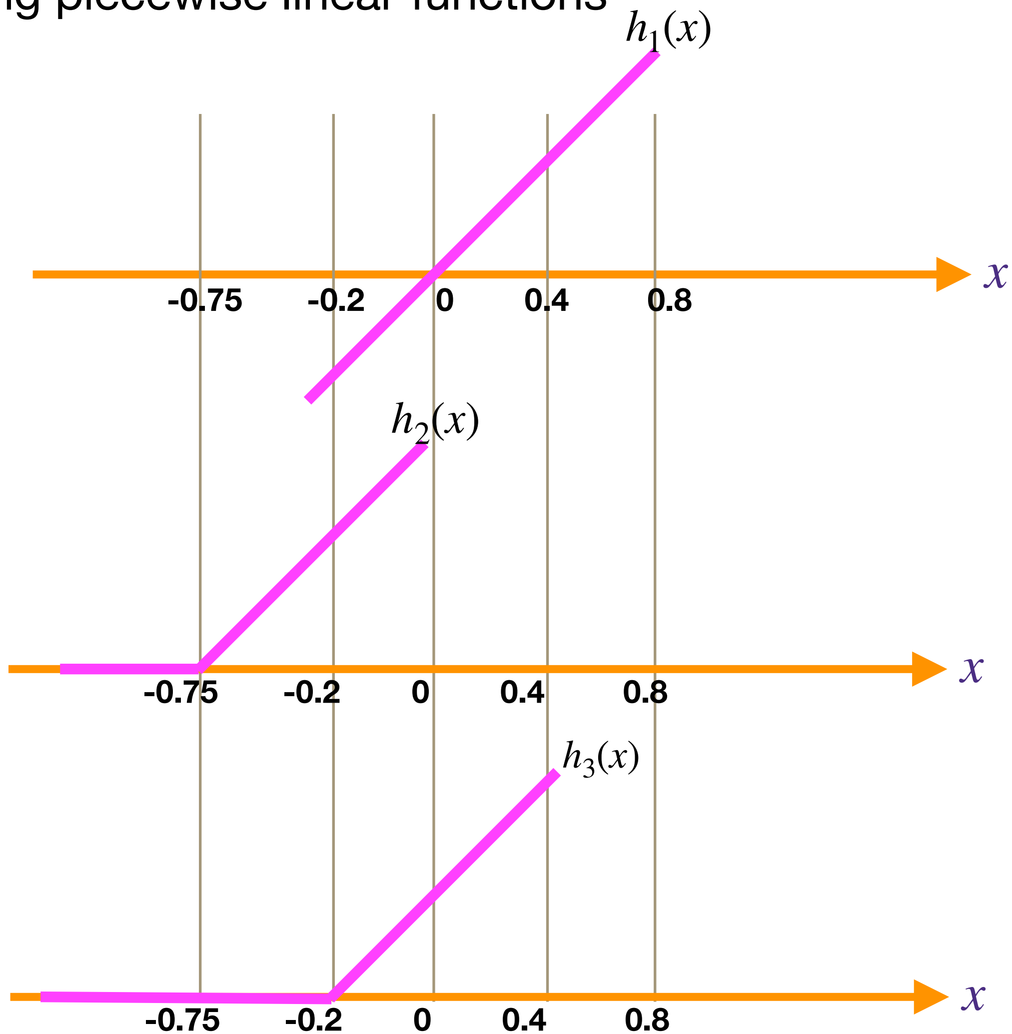
> Typical approach: select λ using cross validation, up next

Example: piecewise linear fit

- we fit a linear model for $x \in [-1,1]$:
  $f(x) = b + w_1 h_1(x) + w_2 h_2(x) + w_3 h_3(x) + w_4 h_4(x) + w_5 h_5(x)$
- with a specific choice of features using piecewise linear functions

$$h(x) = \begin{bmatrix} h_1(x) \\ h_2(x) \\ h_3(x) \\ h_4(x) \\ h_5(x) \end{bmatrix} = \begin{bmatrix} x \\ [x+0.75]^+ \\ [x+0.2]^+ \\ [x-0.4]^+ \\ [x-0.8]^+ \end{bmatrix}$$

$$[a]^+ \triangleq \max\{a, 0\}$$

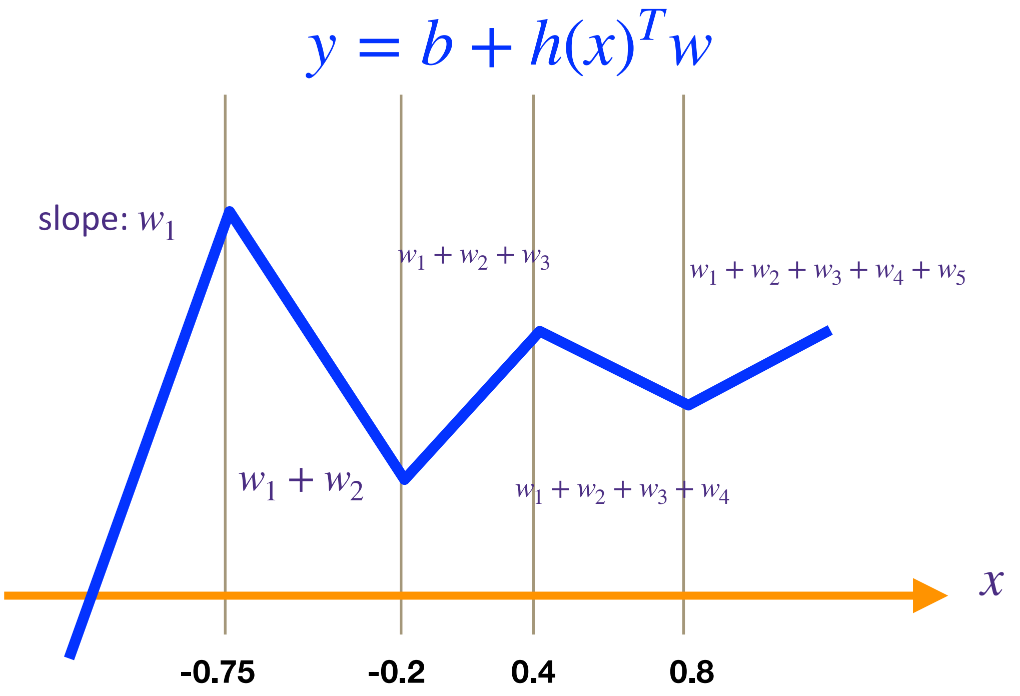


Example: piecewise linear fit

- we fit a linear model:
  $f(x) = b + w_1 h_1(x) + w_2 h_2(x) + w_3 h_3(x) + w_4 h_4(x) + w_5 h_5(x)$
- with a specific choice of features using piecewise linear functions

$$h(x) = \begin{bmatrix} h_1(x) \\ h_2(x) \\ h_3(x) \\ h_4(x) \\ h_5(x) \end{bmatrix} = \begin{bmatrix} x \\ [x+0.75]^+ \\ [x+0.2]^+ \\ [x-0.4]^+ \\ [x-0.8]^+ \end{bmatrix}$$

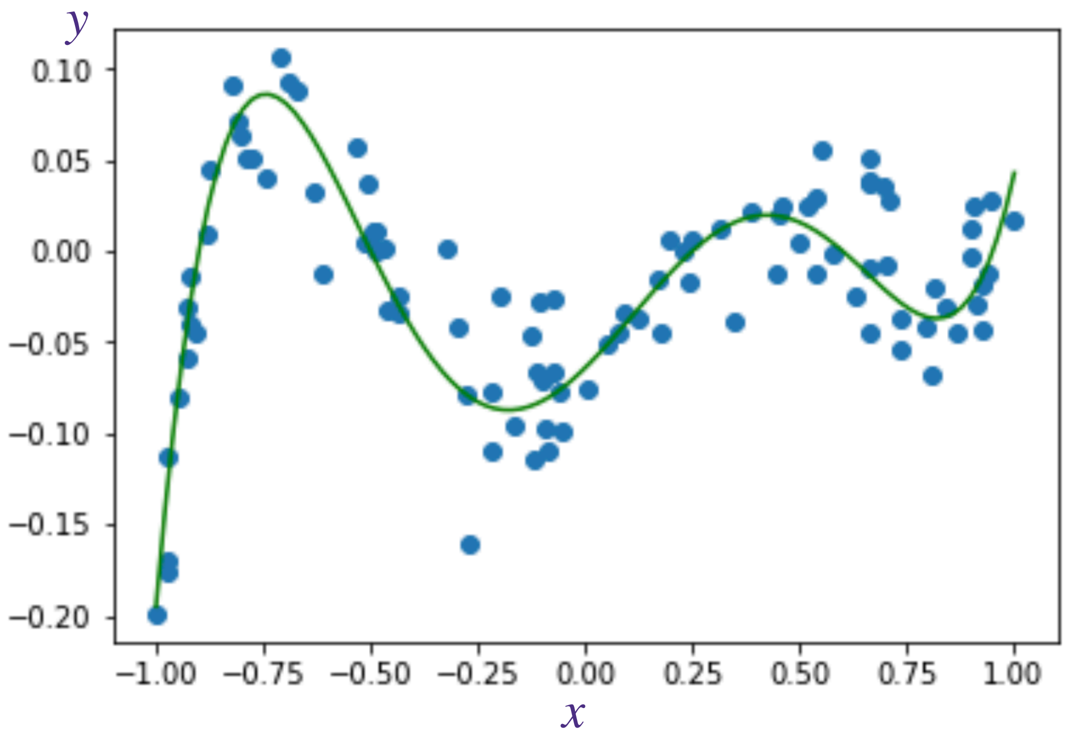
$$[a]^+ \triangleq \max\{a, 0\}$$



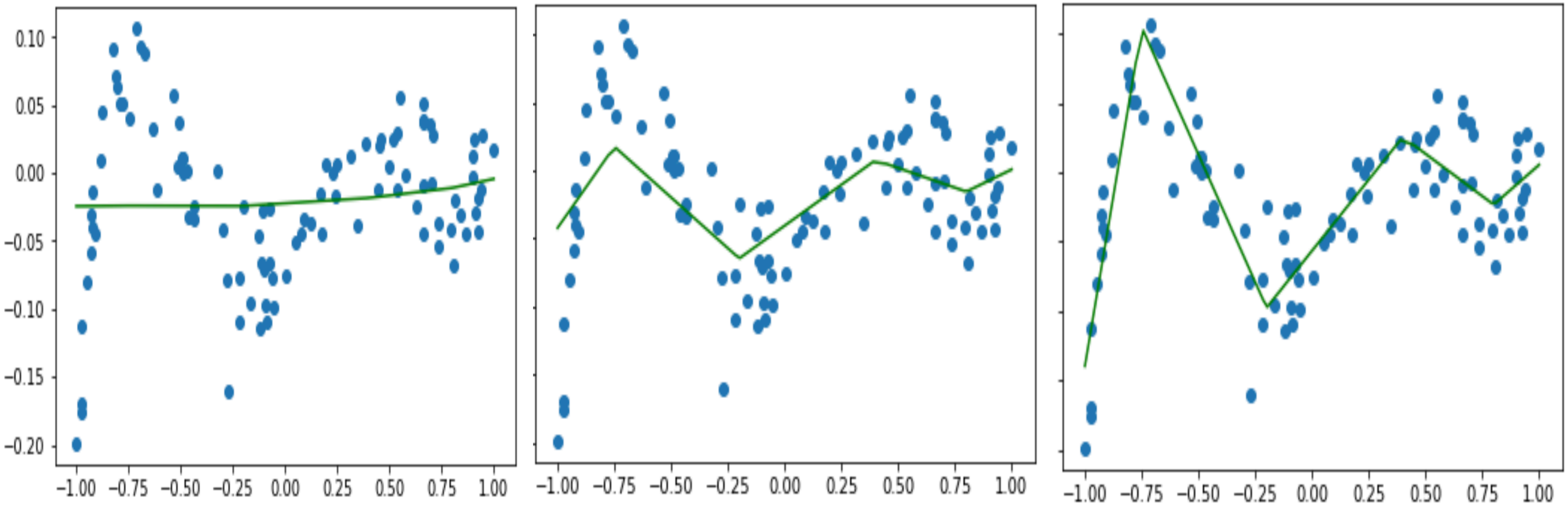
the weights capture the change in the slopes

Example: piecewise linear fit

- we fit a linear model:
  $f(x) = b + w_1 h_1(x) + w_2 h_2(x) + w_3 h_3(x) + w_4 h_4(x) + w_5 h_5(x)$
- with a specific choice of features using piecewise linear functions



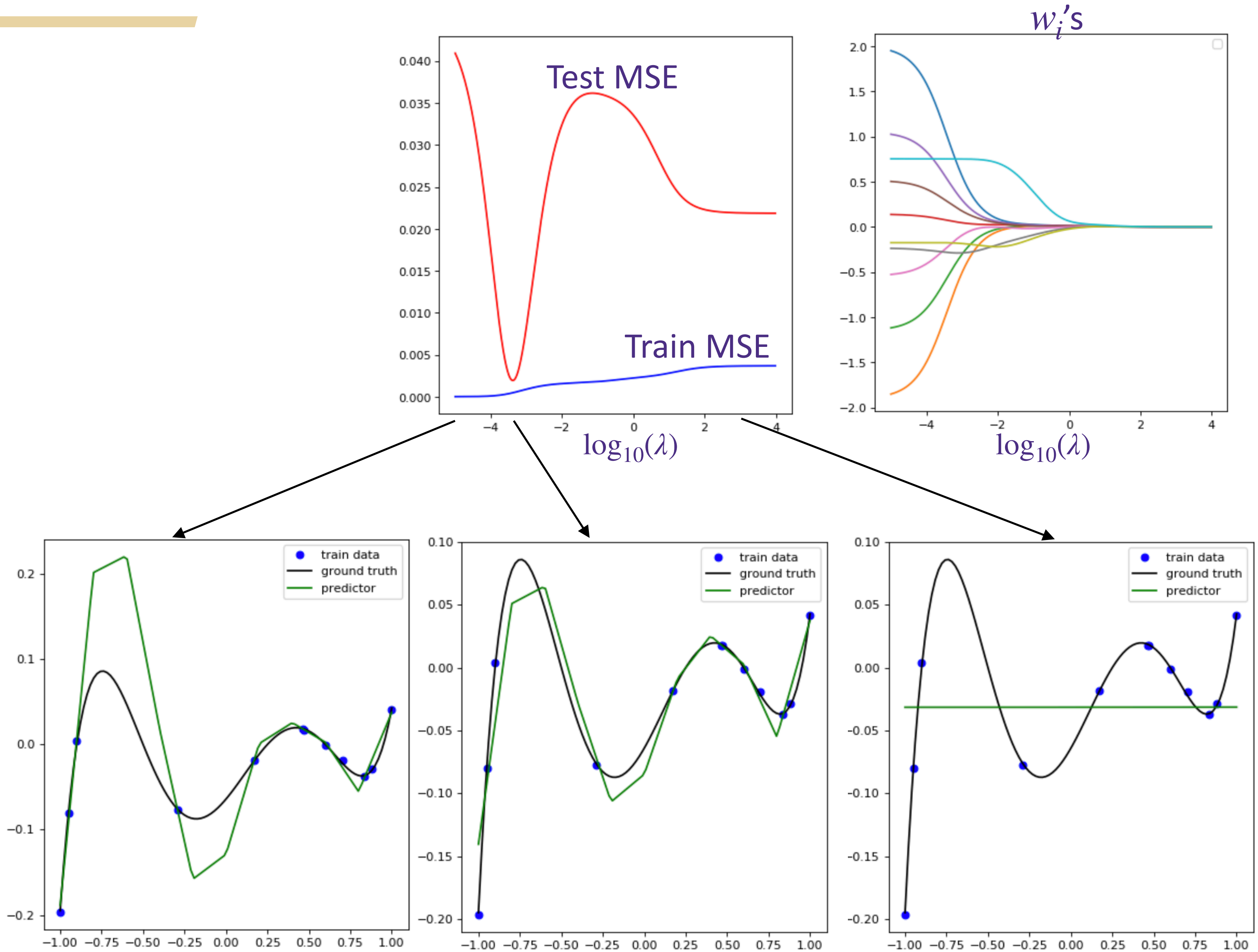
Example: piecewise linear fit (ridge regression)



$\lambda = 1$ $\lambda = 0.005$ $\lambda = 0.000001$

We do not observe overfitting, as d=5 and n=100

Can avoid overfitting even $w \in \mathbb{R}^{10}$ and n=11 samples



What you need to know…

- Regularization
  - Penalizes complex models towards preferred, simpler models
- Ridge regression
  - L_2 penalized least-squares regression
  - Regularization parameter trades off model complexity with training error
  - Practical Notes (link)
    - Never regularize the offset!
    - Generally need to standardize data first to keep inputs on same scale

Simple Variable Selection
LASSO: Sparse Regression

Sparsity

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

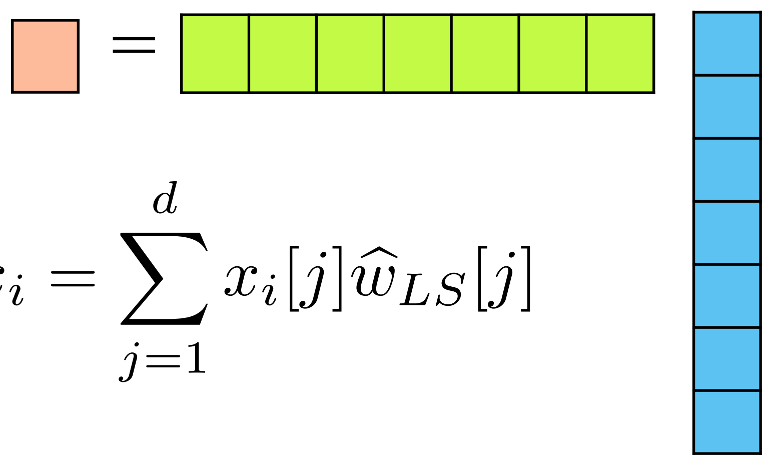
- **Vector w is sparse, if many entries are zero**

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?

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

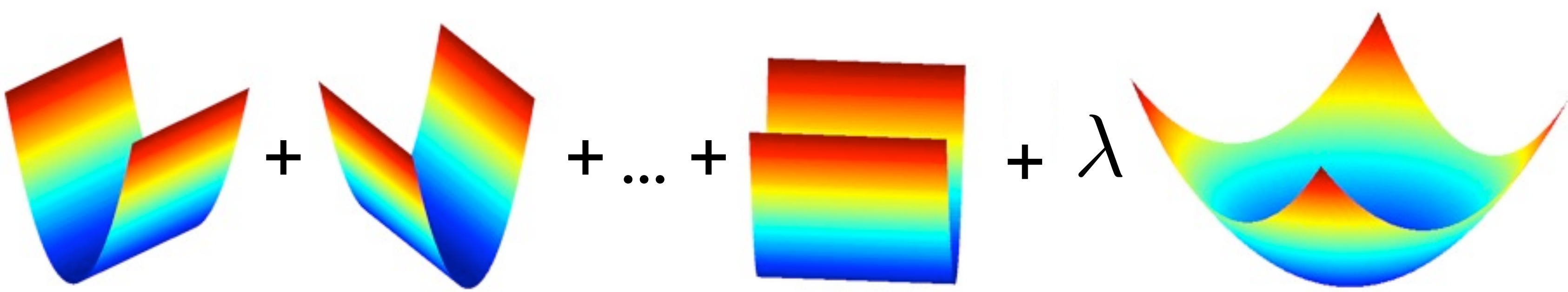
Dishwasher
Garbage disposal
Microwave
Range / Oven
Refrigerator
Washer
Dryer
Laundry location
Heating type
Jetted Tub
Deck
Fenced Yard
Lawn
Garden
Sprinkler System

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

Finding best subset: Regularize

Ridge regression makes coefficients small

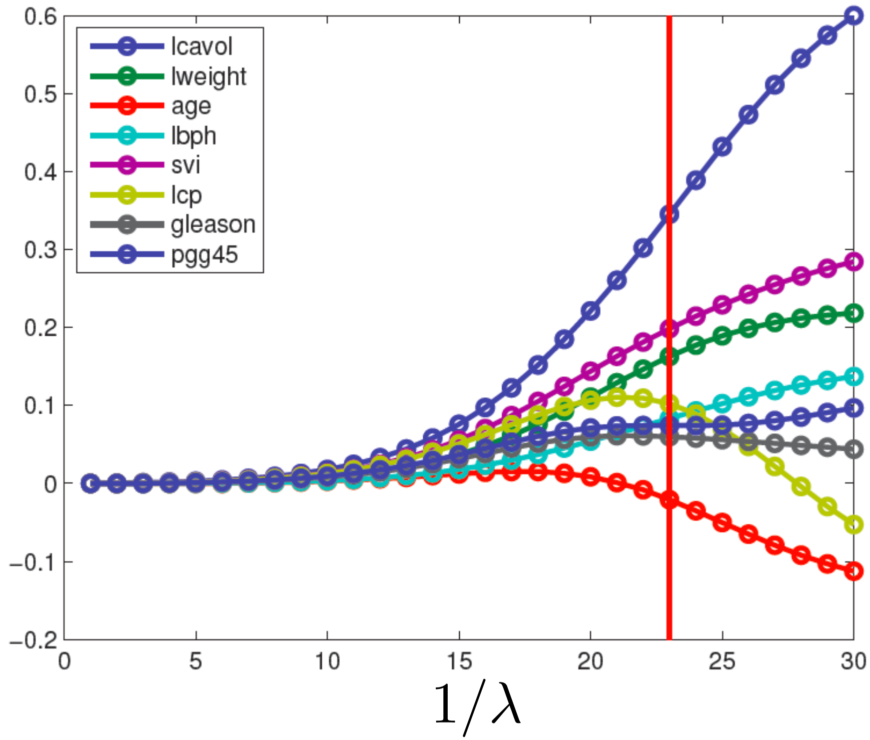
$$\widehat{w}_{ridge} = \arg\min_{w} \sum_{i=1}^{n} \left(y_i - x_i^T w\right)^2 + \lambda ||w||_2^2$$



Finding best subset: Regularize

Ridge regression makes coefficients small

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

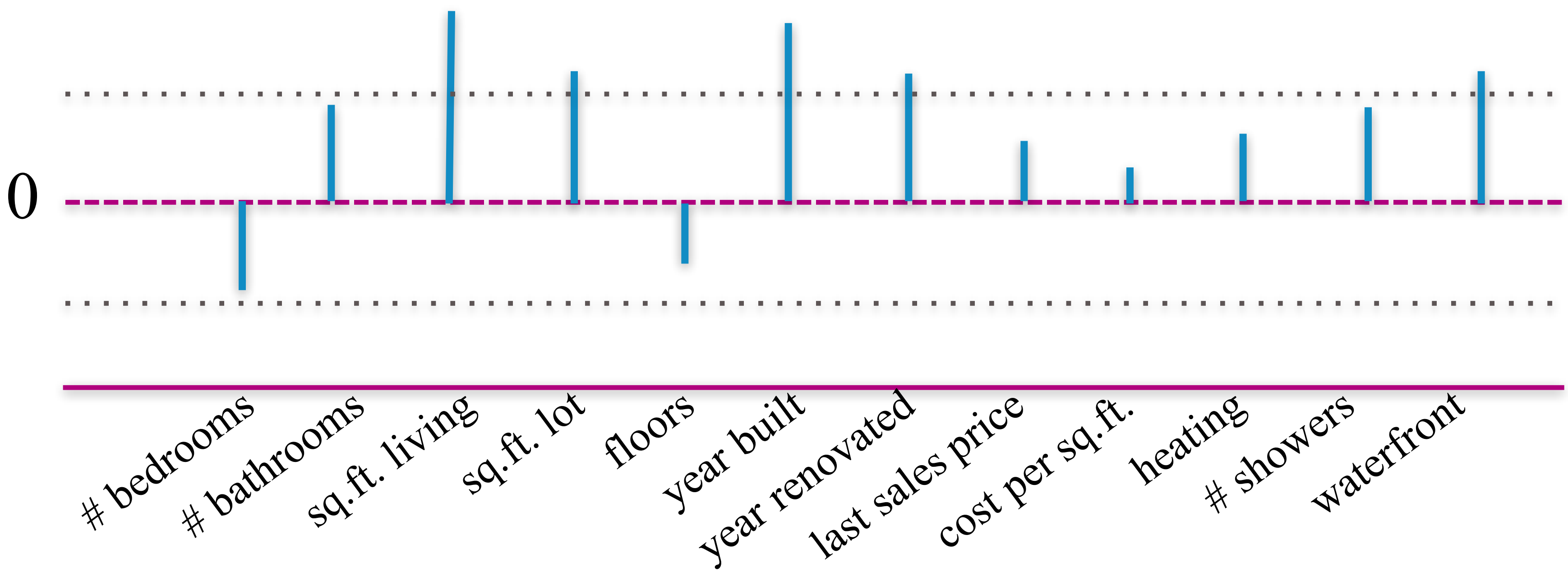


From
Kevin Murphy
textbook

Thresholded Ridge Regression

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

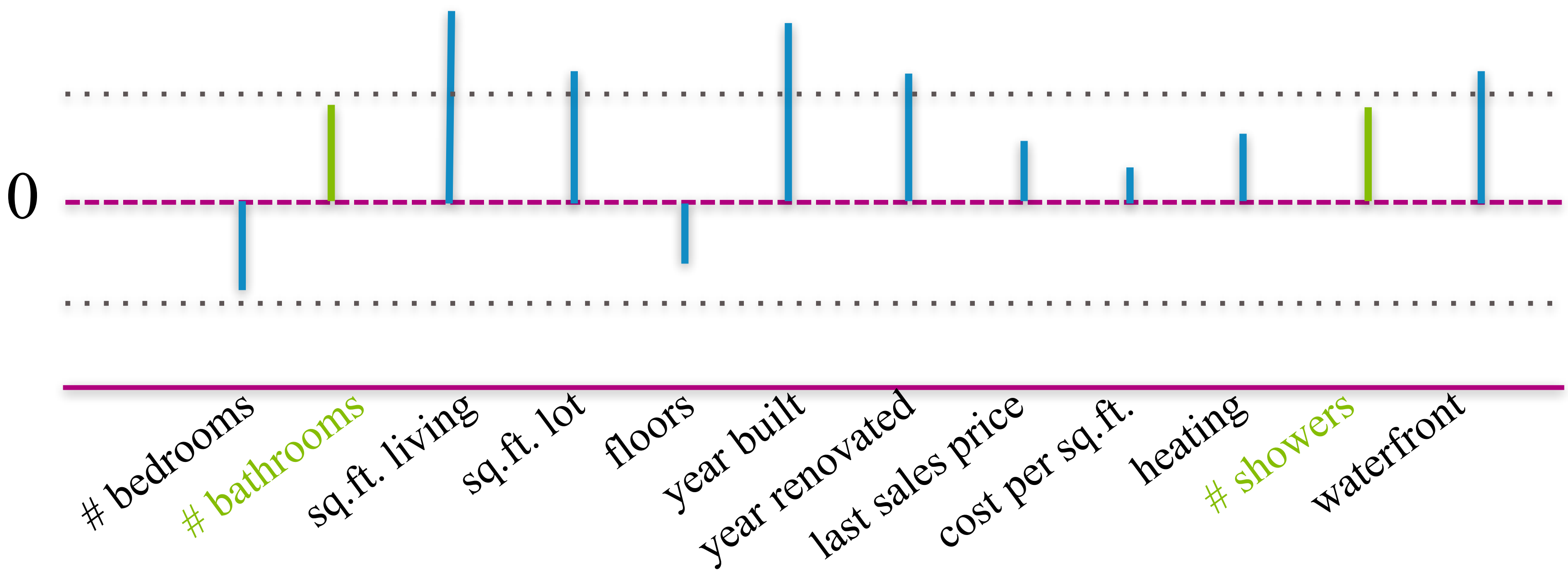
Why don't we just set **small** ridge coefficients to 0?



Thresholded Ridge Regression

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

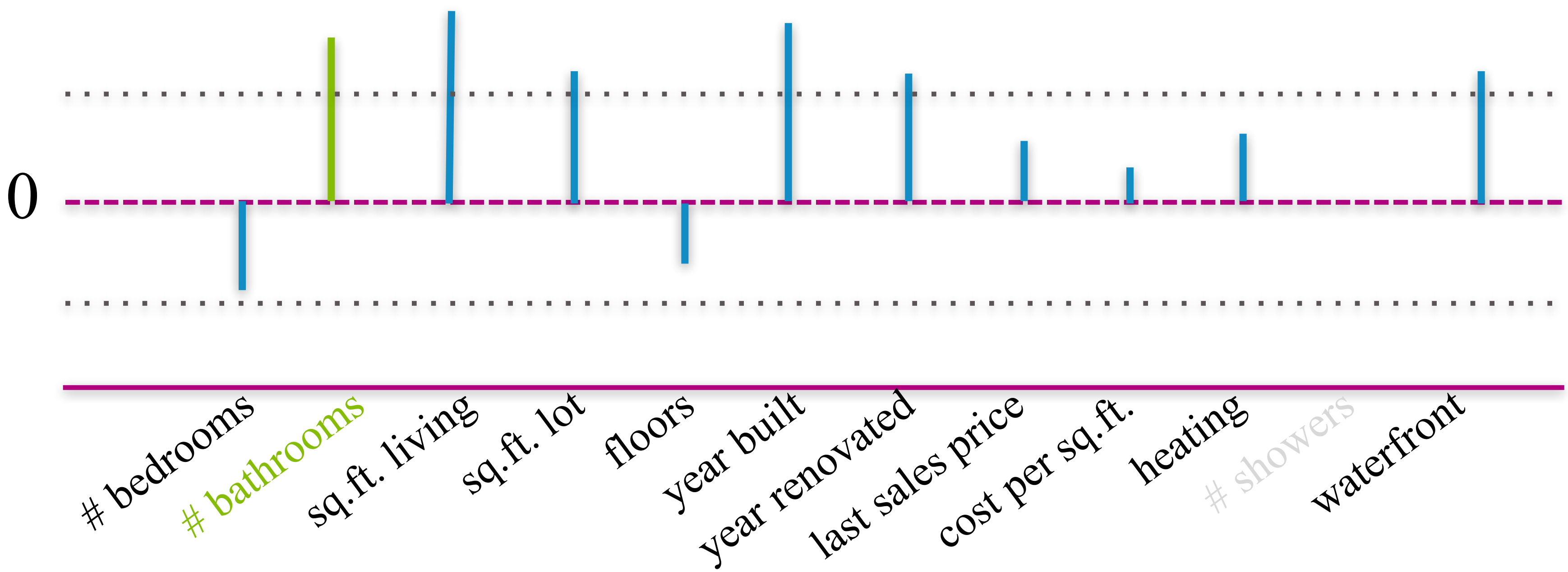
Consider two related features (bathrooms, showers)



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?

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

Forward Greedy

1: $T \leftarrow \varnothing$

2: **For** $j = 1,\ldots,k$ **do**

3: $\quad j^* \leftarrow \arg\min_{\ell} \min_{w} \sum_{i=1}^{n} \left(y_i - \sum_{j \in T \cup \{\ell\}} w[j] \times x_i[j] \right)^2$

4: $\quad T \leftarrow T \cup \{j^*\}$

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

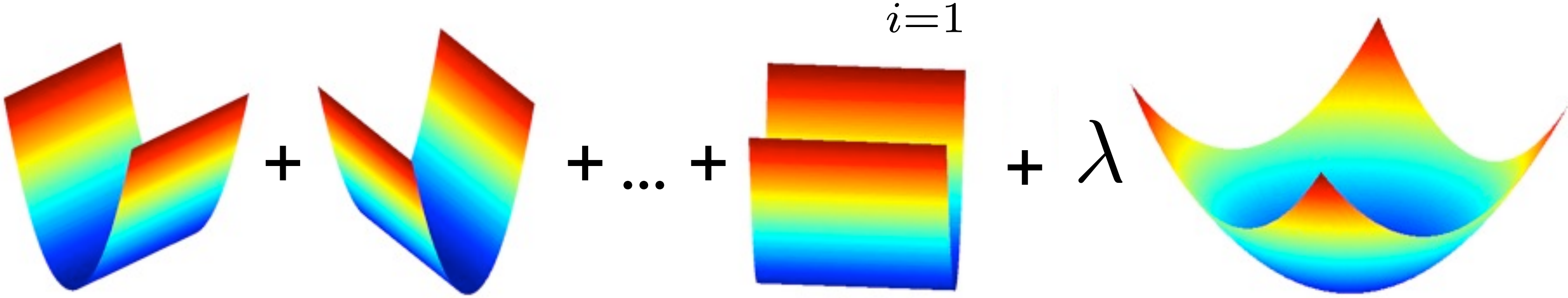
Combining forward and backward steps:
In forward algorithm, insert steps to remove features no longer as important

Lots of other variants, too.

Ridge vs. Lasso Regression

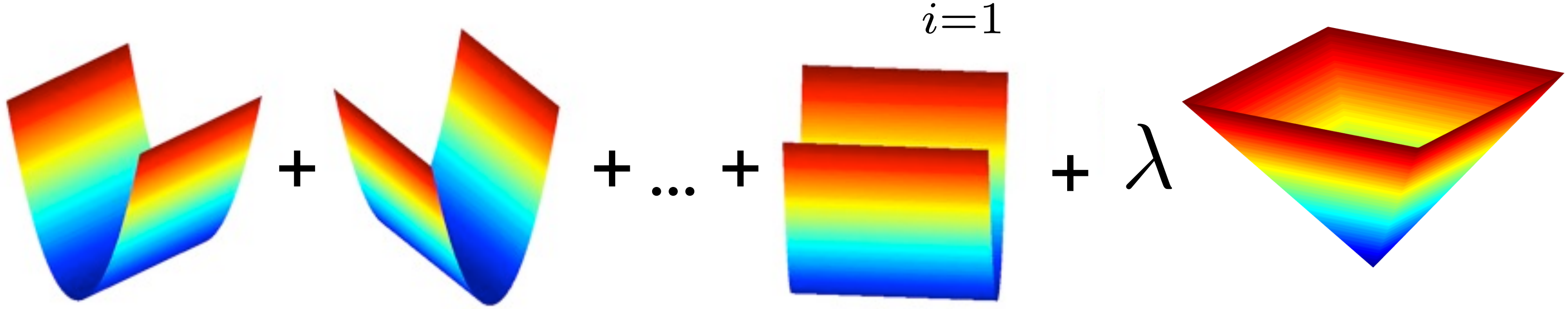
- Ridge Regression objective:

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



- Lasso objective:

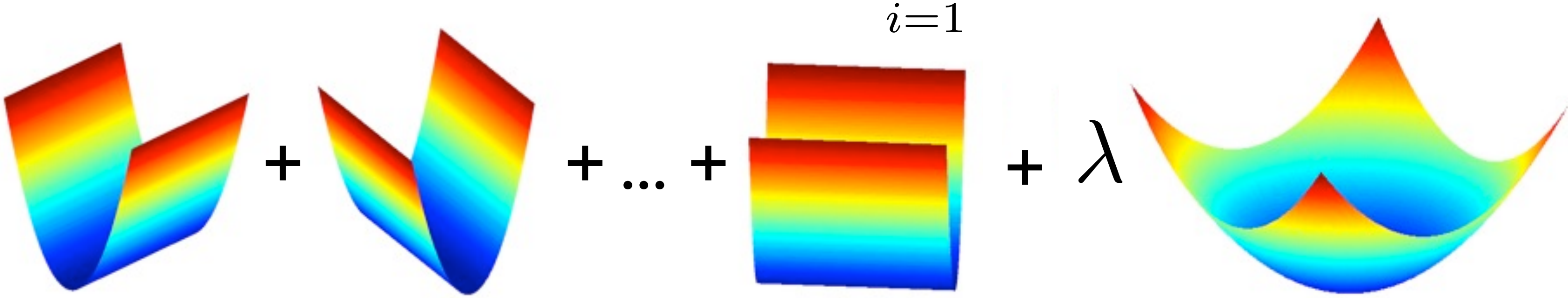
$$\widehat{w}_{lasso} = \arg\min_w \sum_{i=1}^{n} \left(y_i - x_i^T w\right)^2 + \lambda ||w||_1$$



Recall Ridge Regression

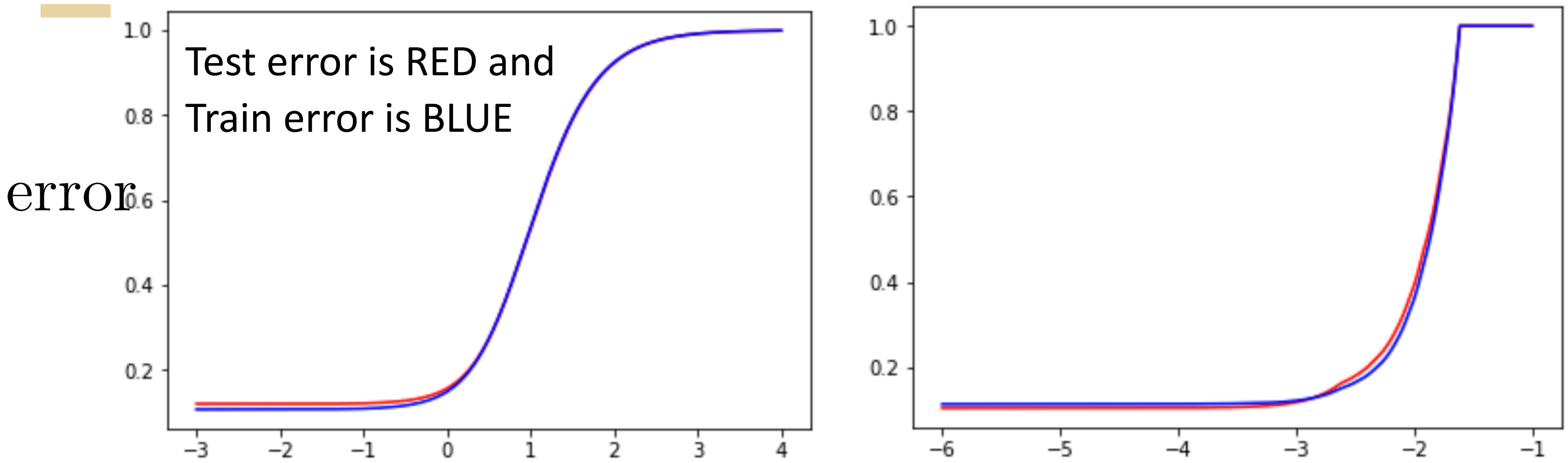
- Ridge Regression objective:

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

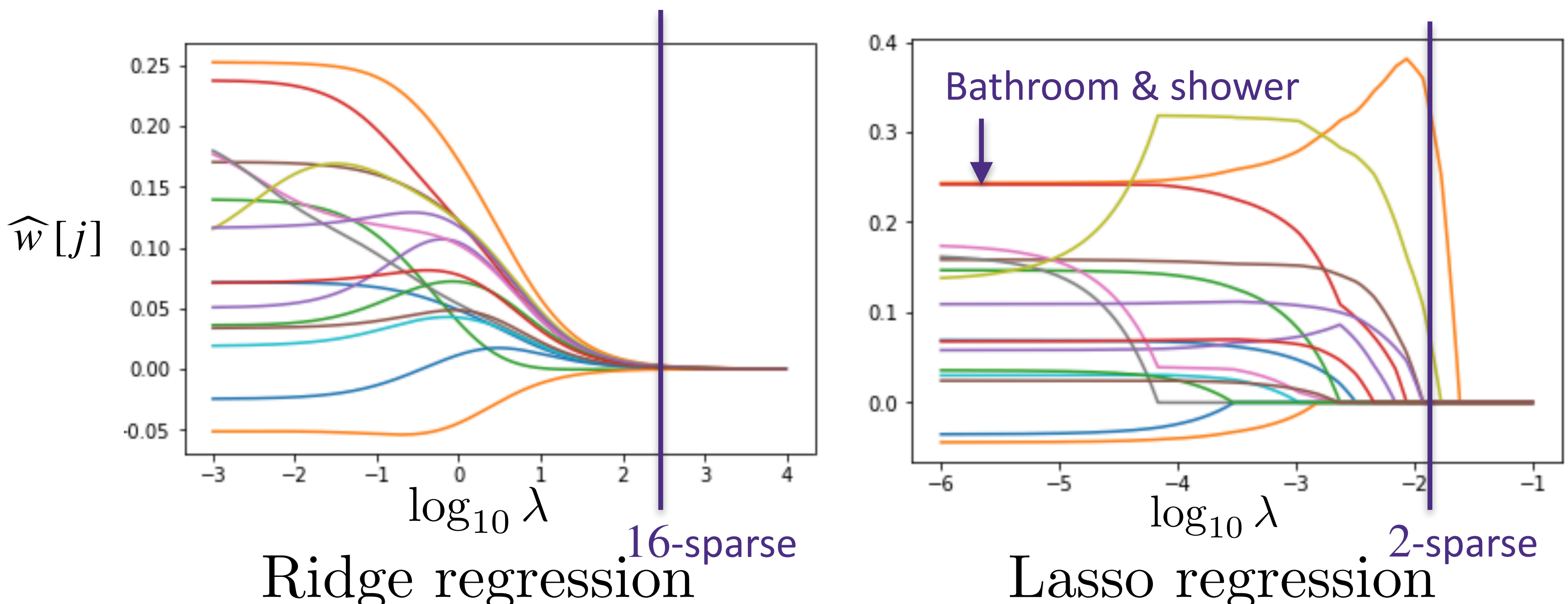


$$\|w\|_p = \left(\sum_{i=1}^{d} |w|^p\right)^{1/p}$$

Example: house price with 16 features



- Regularization path for Lasso shows that weights drop to exactly zero as λ increases



Lasso regression naturally gives sparse features

- **feature selection** with Lasso regression

    1. **Model selection**: choose λ based on cross validation error
    2. **Feature selection**: keep only those features with non-zero (or not-too-small) parameters in w at optimal λ
    3. **retrain** with the sparse model and $\lambda = 0$

    why do we need to retrain?

Example: piecewise-linear fit

- We use Lasso on the piece-wise linear example

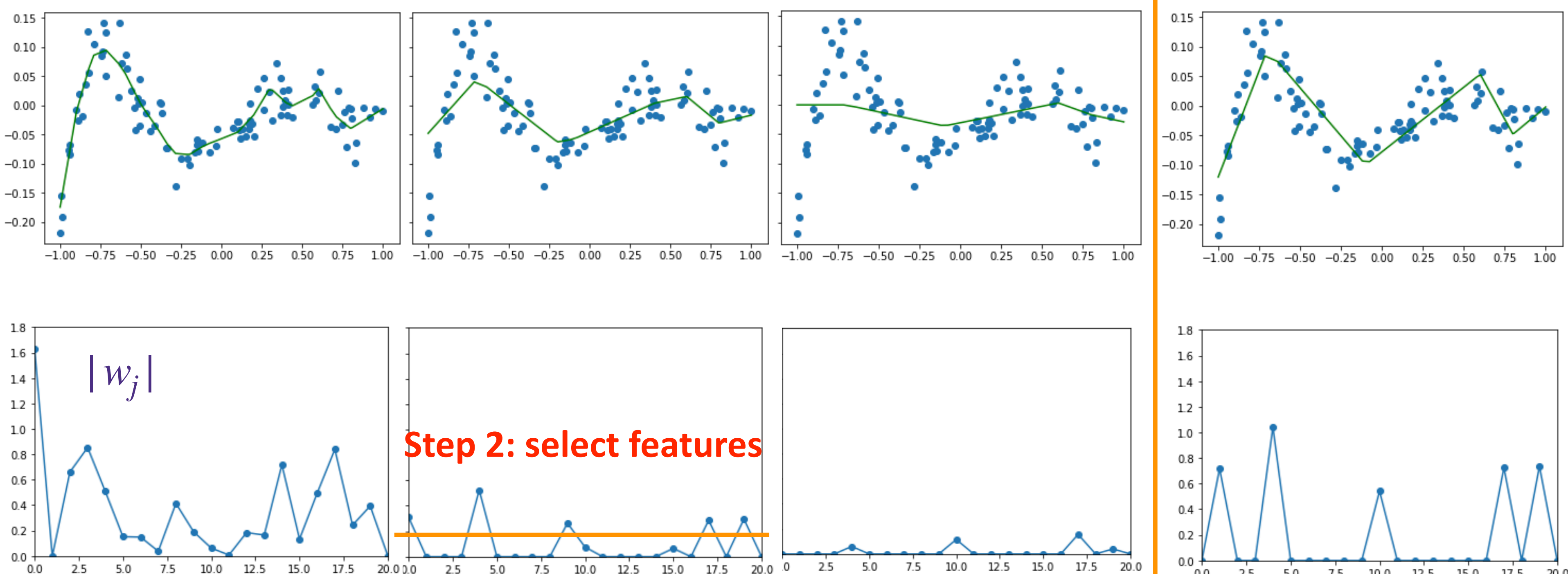
$$\begin{aligned} h_0(x) &= 1 \\ h_i(x) &= [x + 1.1 - 0.1i]^+ \end{aligned}$$

Step 1: find optimal $\lambda*$

$$\text{minimize}_w \ \ \mathscr{L}(w) + \lambda \|w\|_1$$

Step 3: retrain

$$\text{minimize}_w \ \ \mathscr{L}(w)$$



$\lambda = 10^{-8}$ $\lambda = 10^{-4}$ $\lambda = 2 \times 10^{-4}$ $\lambda = 0$

but only use selected features

- de-biasing (via re-training) is critical!

Penalized Least Squares

- Regularized optimization:

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

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

Penalized Least Squares

- Regularized optimization:

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

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

- For any $\lambda^* \geq 0$ for which $\hat{w}_r$ achieves the minimum, there exists a $\mu^* \geq 0$ such that the solution of the constrained optimization, $\widehat{w}_c$, is the same as the solution of the regularized optimization , $\widehat{w}_r$, where

$$\widehat{w}_C = \arg\min_w \sum_{i=1}^{n} (y_i - x_i^T w)^2 \qquad \text{subject to } \; r(w) \leq \mu^*$$

- so there are pairs of (λ, μ) whose optimal solution $\widehat{w}_r$ are the same for the regularizes optimization and constrained optimization

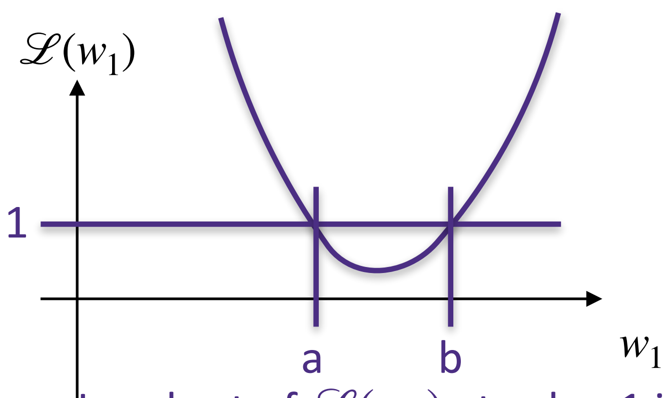
Why does Lasso give sparse solutions?

$$\text{minimize}_w \quad \sum_{i=1}^{n} (w^T x_i - y_i)^2$$

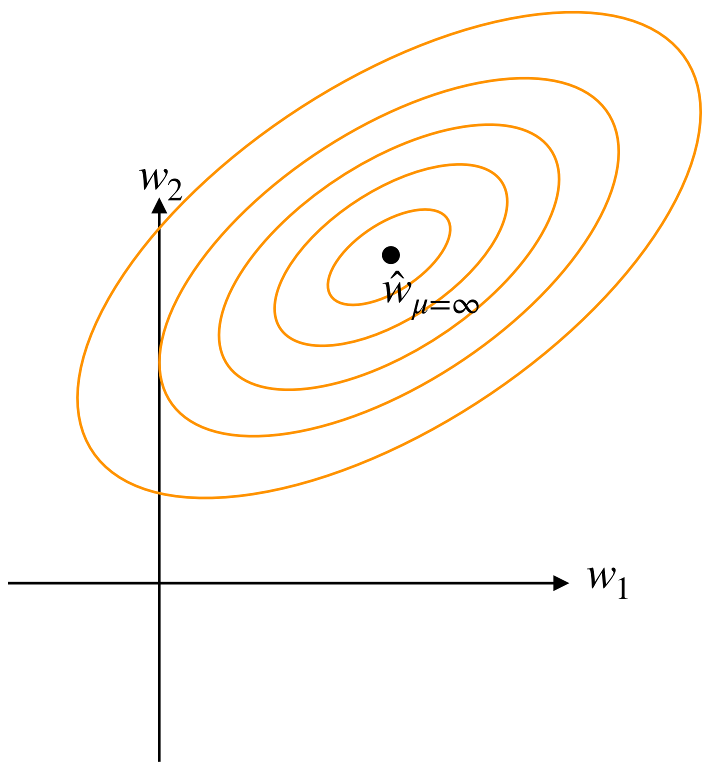
$$\text{subject to} \quad \|w\|_1 \leq \mu$$

- the **level set** of a function $\mathscr{L}(w_1, w_2)$ is defined as the set of points (w_1, w_2) that have the same function value
- the level set of a quadratic function is an oval
- the center of the oval is the least squares solution $\hat{w}_{\mu=\infty} = \hat{w}_{\text{LS}}$

1-D example with quadratic loss



Level set of $\mathscr{L}(w_1)$ at value 1 is [a,b]

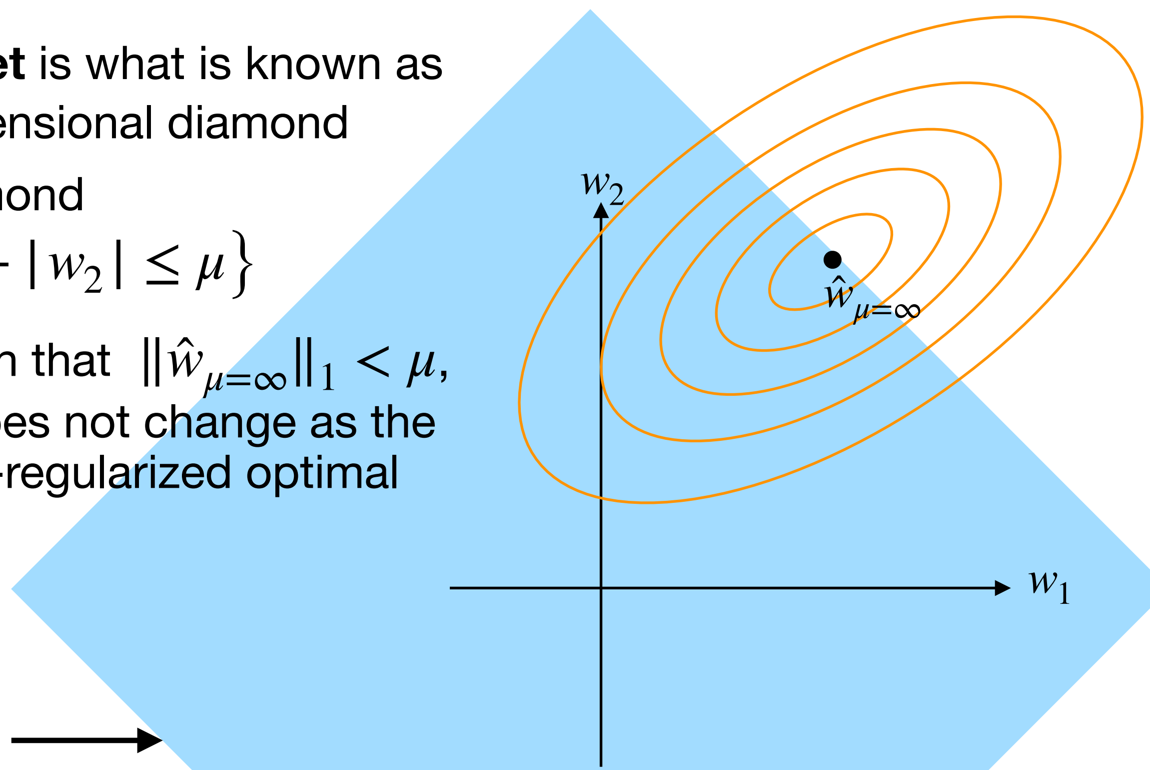


Why does Lasso give sparse solutions?

$$\text{minimize}_w \quad \sum_{i=1}^{n} (w^T x_i - y_i)^2$$

$$\text{subject to} \quad \|w\|_1 \leq \mu$$

- as we decrease μ from infinity, the feasible set becomes smaller
- the shape of the **feasible set** is what is known as L_1 ball, which is a high dimensional diamond
- In 2-dimensions, it is a diamond
$$\left\{(w_1, w_2) \,\middle|\, |w_1| + |w_2| \leq \mu \right\}$$
- when μ is large enough such that $\|\hat{w}_{\mu=\infty}\|_1 < \mu$, then the optimal solution does not change as the feasible set includes the un-regularized optimal solution



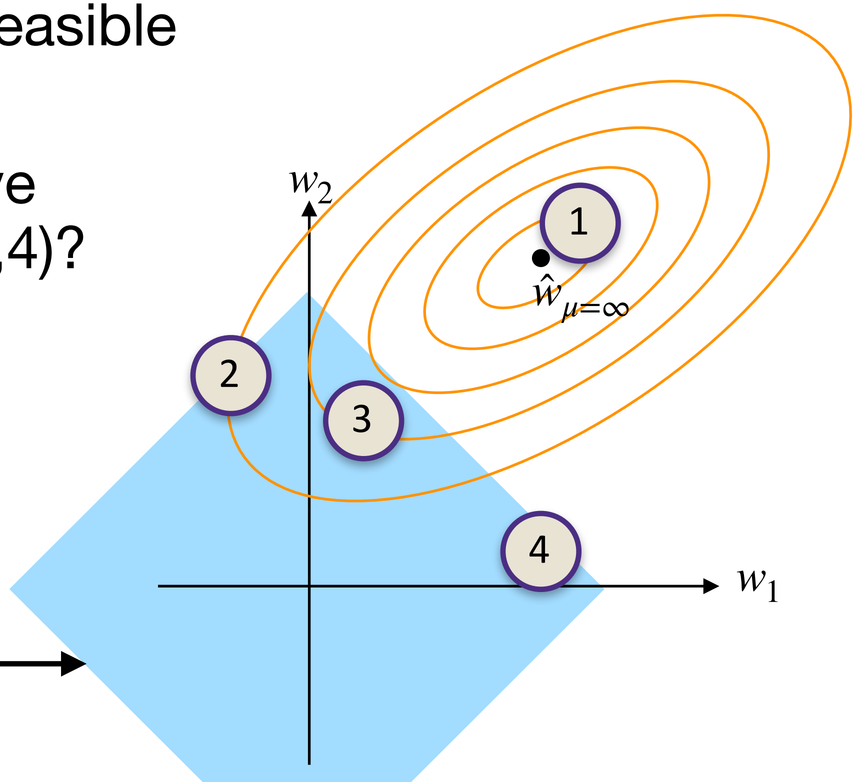
feasible set: $\{w \in \mathbb{R}^2 \mid \|w\|_1 \leq \mu\}$

Why does Lasso give sparse solutions?

$$\text{minimize}_w \quad \sum_{i=1}^{n} (w^T x_i - y_i)^2$$

$$\text{subject to} \quad \|w\|_1 \leq \mu$$

- As μ decreases (which is equivalent to increasing regularization λ) the feasible set (blue diamond) shrinks
- The optimal solution of the above optimization (out of points 1,2,3,4)?



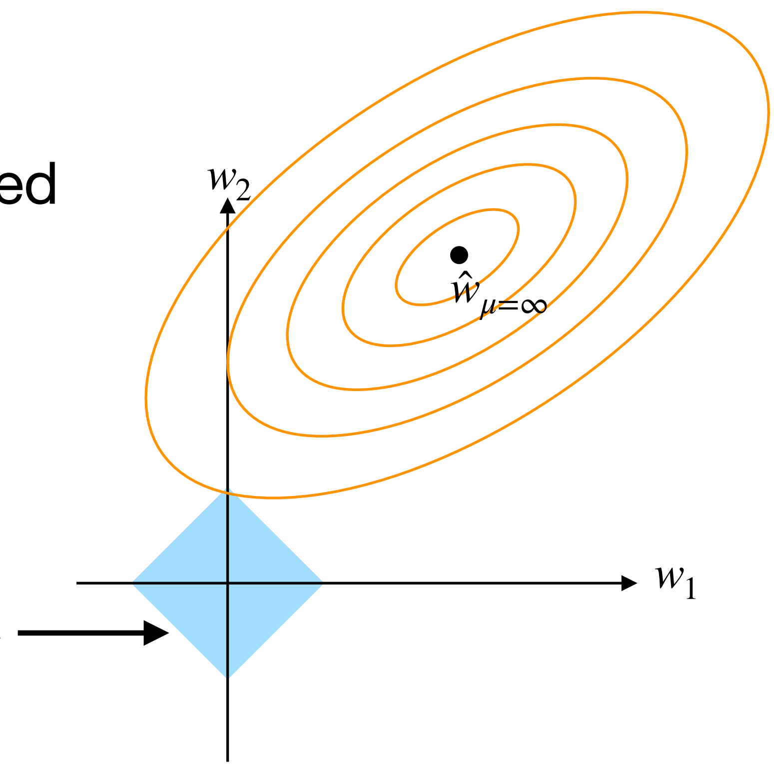
feasible set: $\{w \in \mathbb{R}^2 \mid \|w\|_1 \leq \mu\}$

Why does Lasso give sparse solutions?

$$\text{minimize}_w \quad \sum_{i=1}^{n} (w^T x_i - y_i)^2$$

$$\text{subject to} \quad \|w\|_1 \ \leq \ \mu$$

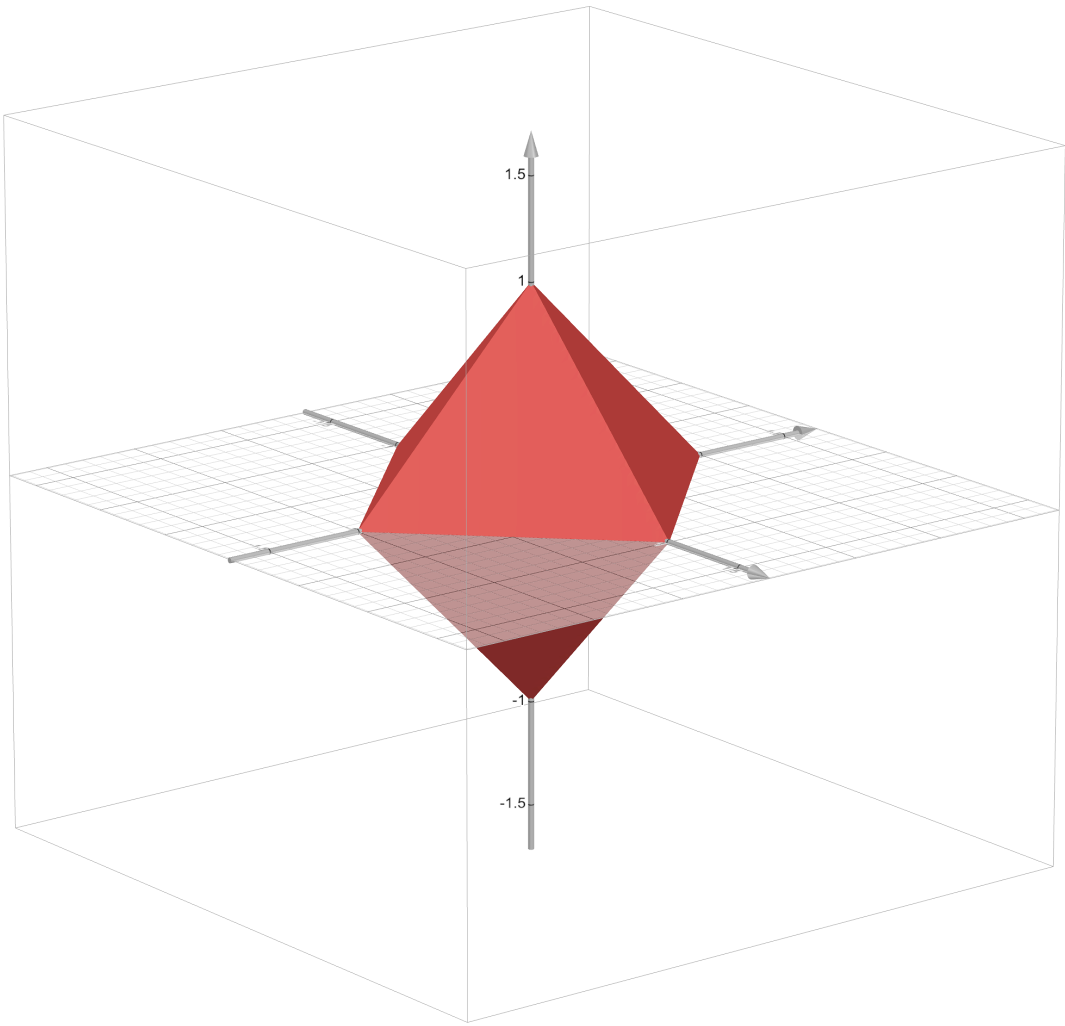
- For small enough μ, the optimal solution becomes **sparse**
- This is because the L_1-ball is "pointy",i.e., has sharp edges aligned with the axes



feasible set: $\{w \in \mathbb{R}^2 \mid \|w\|_1 \leq \mu\}$

L1 Ball in Higher Dimensions

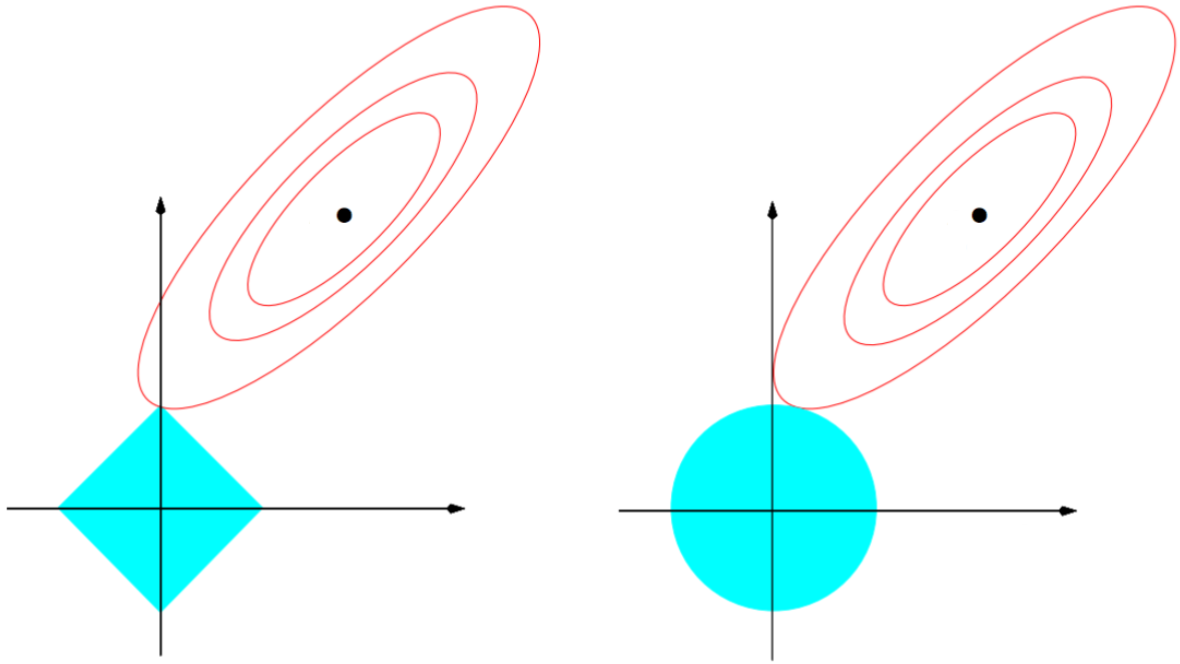
- L1 ball 3 dimensions
  - Corners 2-sparse
  - Edges: 1-sparse



- In higher dimensions, the L1 ball is “even pointier”

Penalized Least Squares

- Lasso regression finds sparse solutions, as L_1-ball is "pointy"
- Ridge regression finds dense solutions, as L_2-ball is "smooth"



$$\text{minimize}_w \quad \sum_{i=1}^{n} (w^T x_i - y_i)^2$$

$$\text{subject to} \quad \|w\|_1 \leq \mu$$

$$\text{minimize}_w \quad \sum_{i=1}^{n} (w^T x_i - y_i)^2$$

$$\text{subject to} \quad \|w\|_2^2 \leq \mu$$

Gradient Descent

- how are we going to find the solution for

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

- e.g., Lasso, Logistic Regression do not have closed form solution for

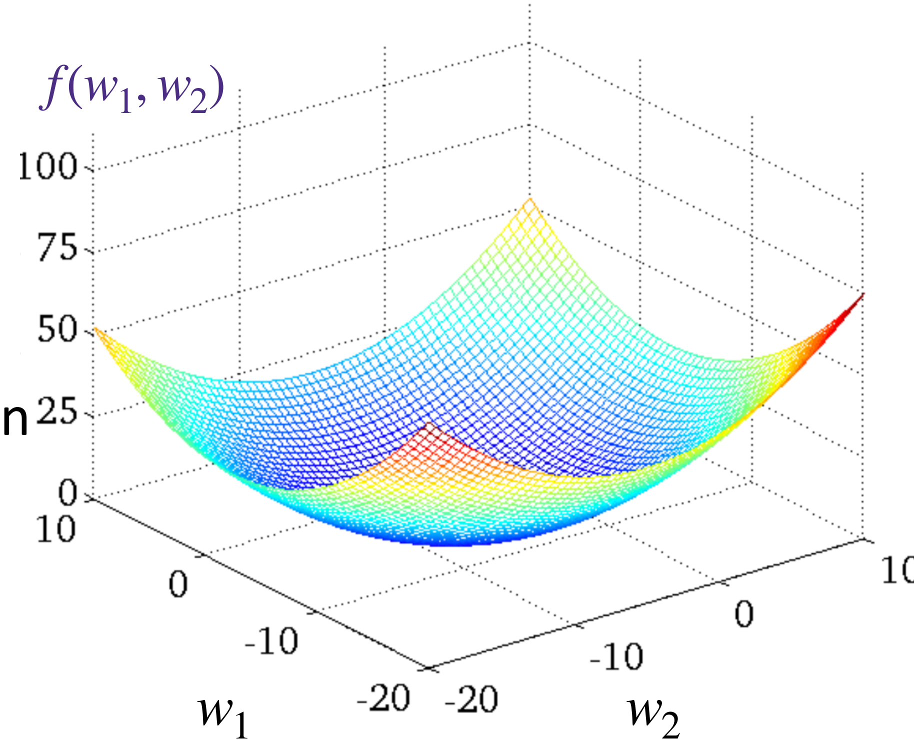
$$\nabla_{b,w} \mathscr{L}(b, w) = 0$$

Running example: linear regression

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

- **Learning model parameters:**

$$\hat{w}_{\text{LS}} = \arg\min_{w \in \mathbb{R}^d} \underbrace{\|\mathbf{y} - \mathbf{X}w\|_2^2}_{f(w)}$$



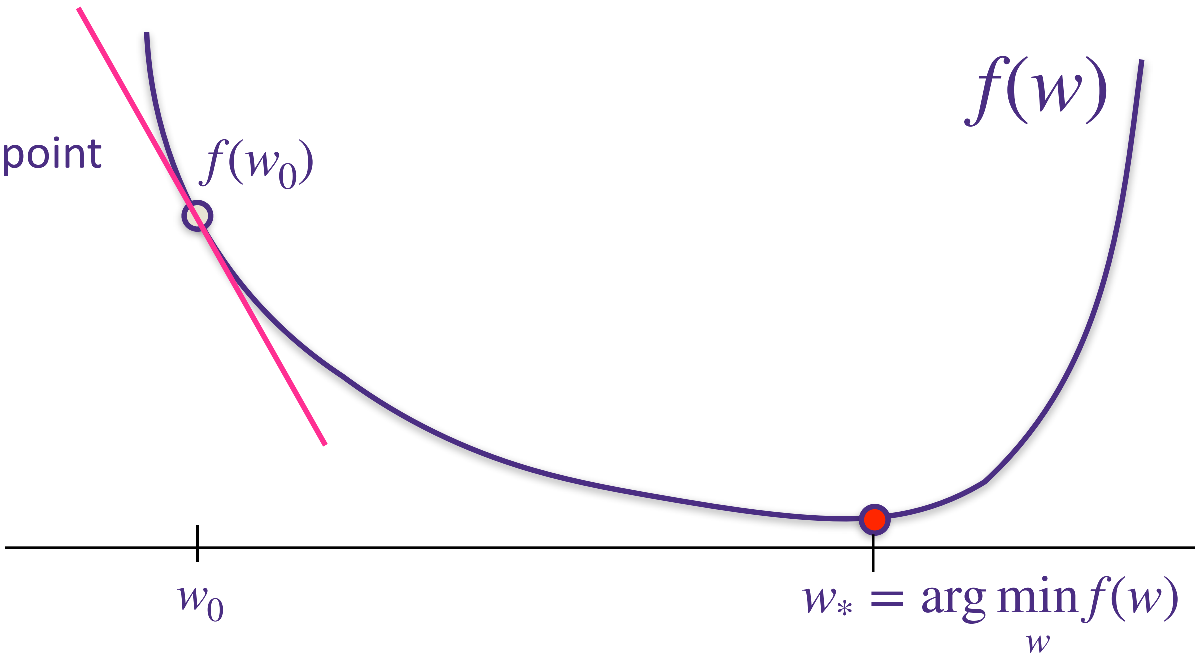
- Although we know the optimal solution in a closed form, we will use this as a running example to understand GD

1-dimensional gradient descent

Let w_0 be an initial guess. How can we improve this solution?

Derivative tells rate of change at a point

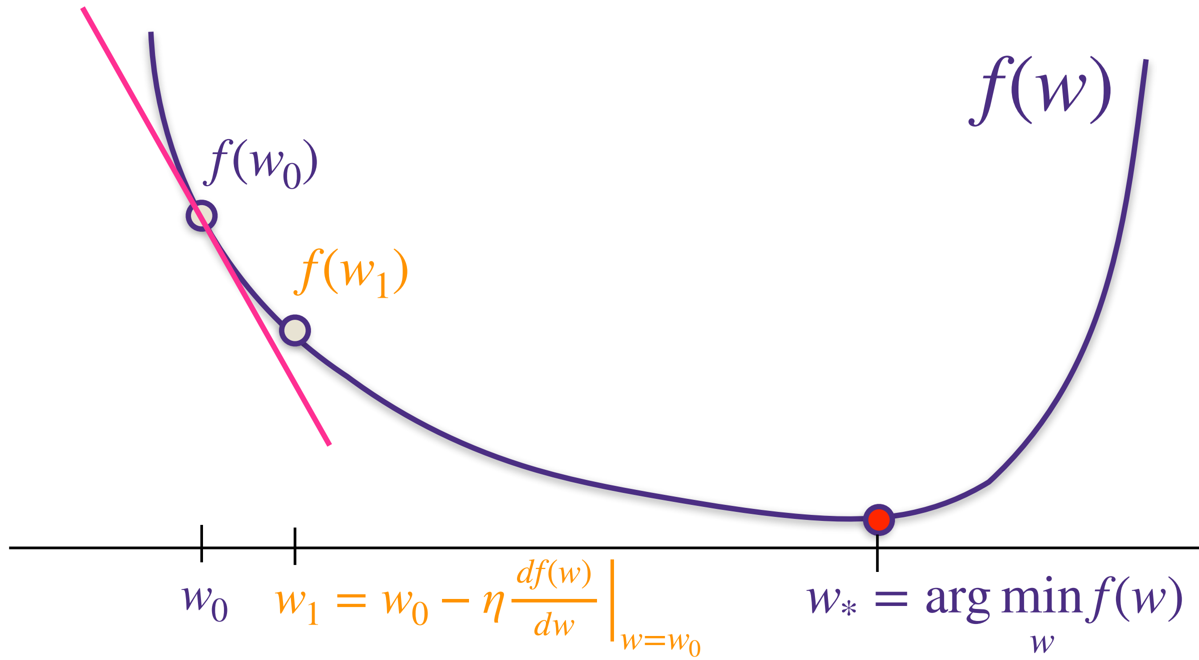
$$\frac{\partial}{\partial w} f(w) \Big|_{w=w_0}$$



Idea: If the function is convex, then stepping the *opposite* direction of the derivative gets us closer to minimizing the function

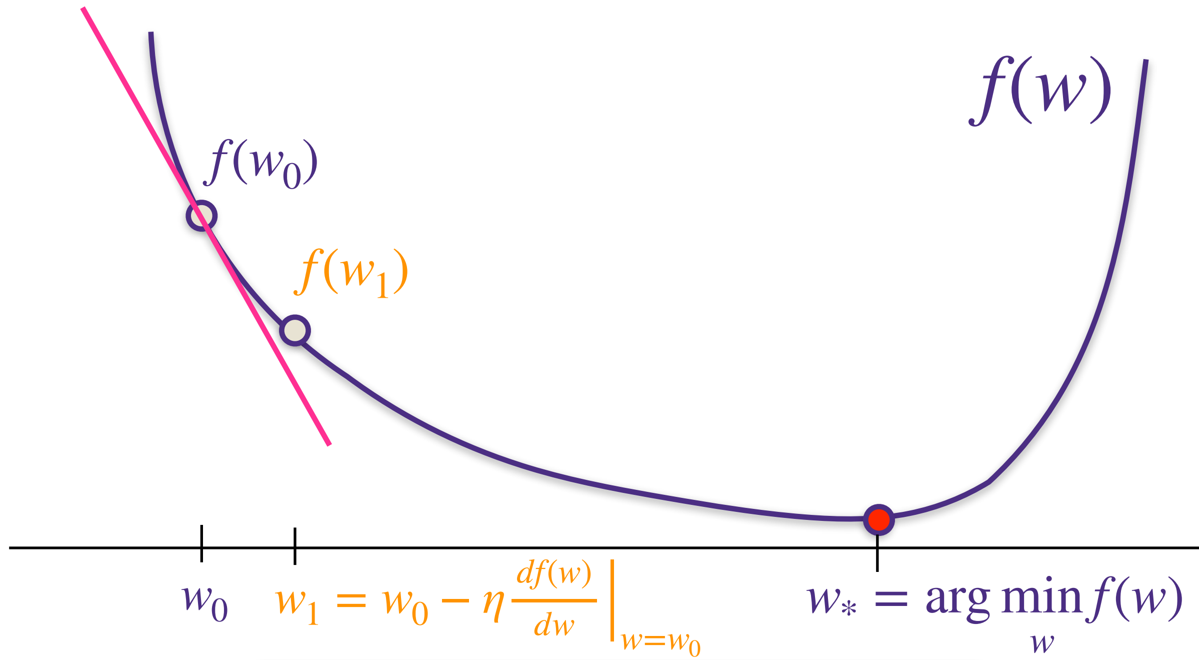
1-dimensional gradient descent

Let w_0 be an initial guess. How can we improve this solution?



1-dimensional gradient descent

Let w_0 be an initial guess. How can we improve this solution?



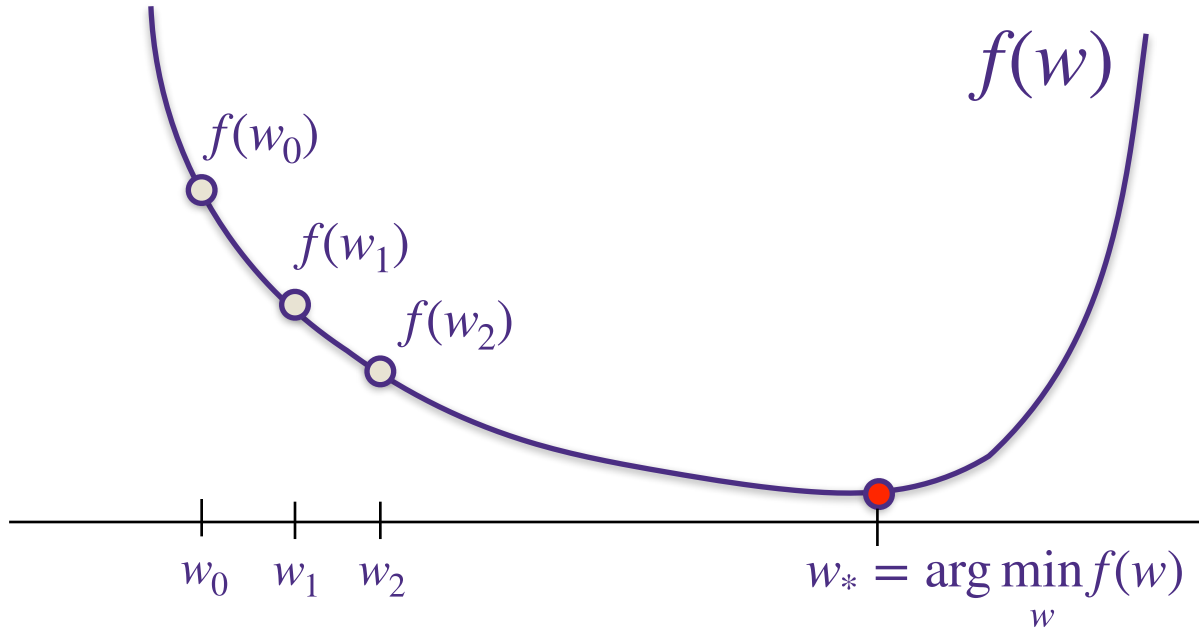
Gradient descent

For k=0,1,2,3,...

$$w_{k+1} = w_k - \eta \left.\frac{df(w)}{dw}\right|_{w=w_k}$$

1-dimensional gradient descent

Let w_0 be an initial guess. How can we improve this solution?



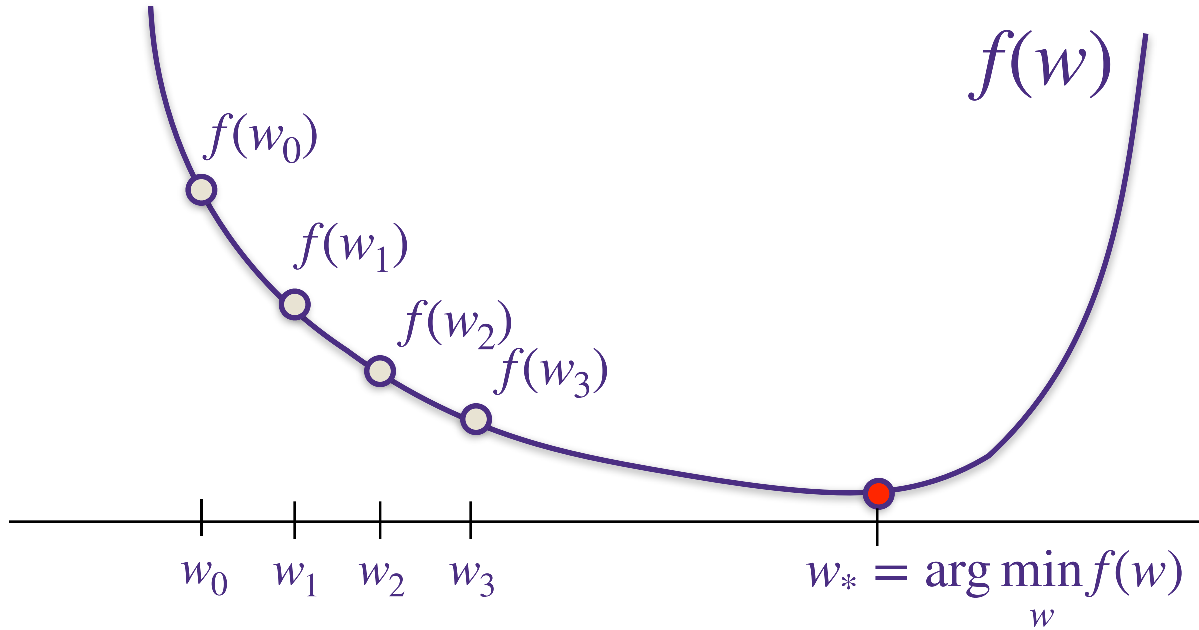
Gradient descent

For k=0,1,2,3,…

$$w_{k+1} = w_k - \eta \frac{df(w)}{dw}\Big|_{w=w_k}$$

1-dimensional gradient descent

Let w_0 be an initial guess. How can we improve this solution?



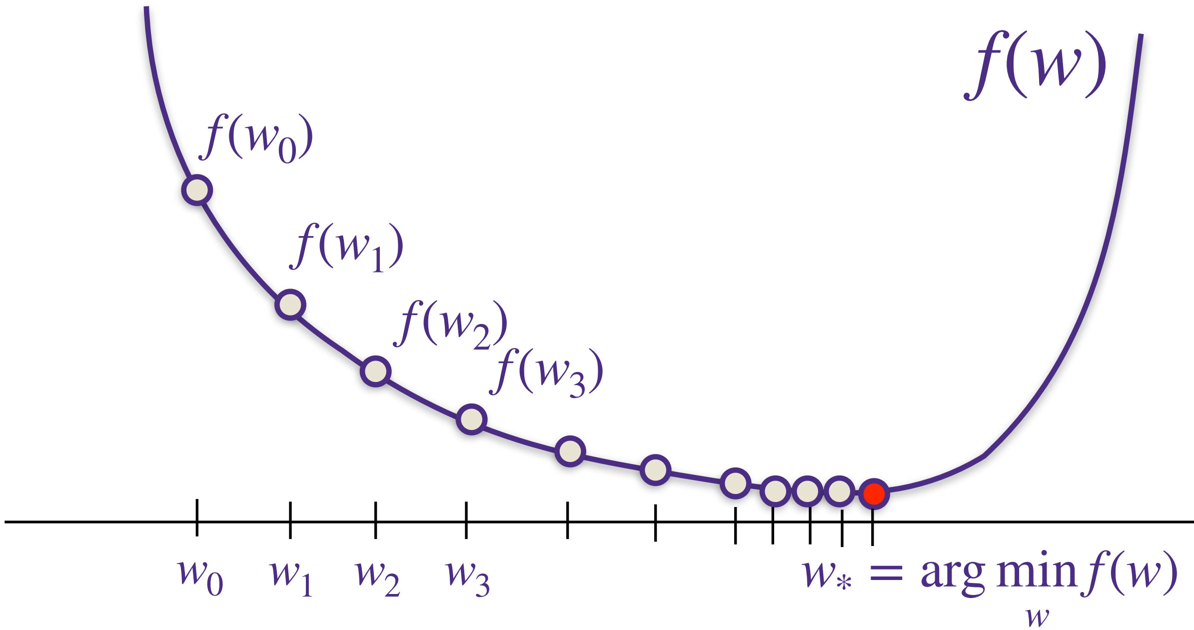
Gradient descent

For k=0,1,2,3,...

$$w_{k+1} = w_k - \eta \frac{df(w)}{dw}\Big|_{w=w_k}$$

1-dimensional gradient descent

Let w_0 be an initial guess. How can we improve this solution?



Gradient descent

For k=0,1,2,3,...

$$w_{k+1} = w_k - \eta \frac{df(w)}{dw}\Big|_{w=w_k}$$

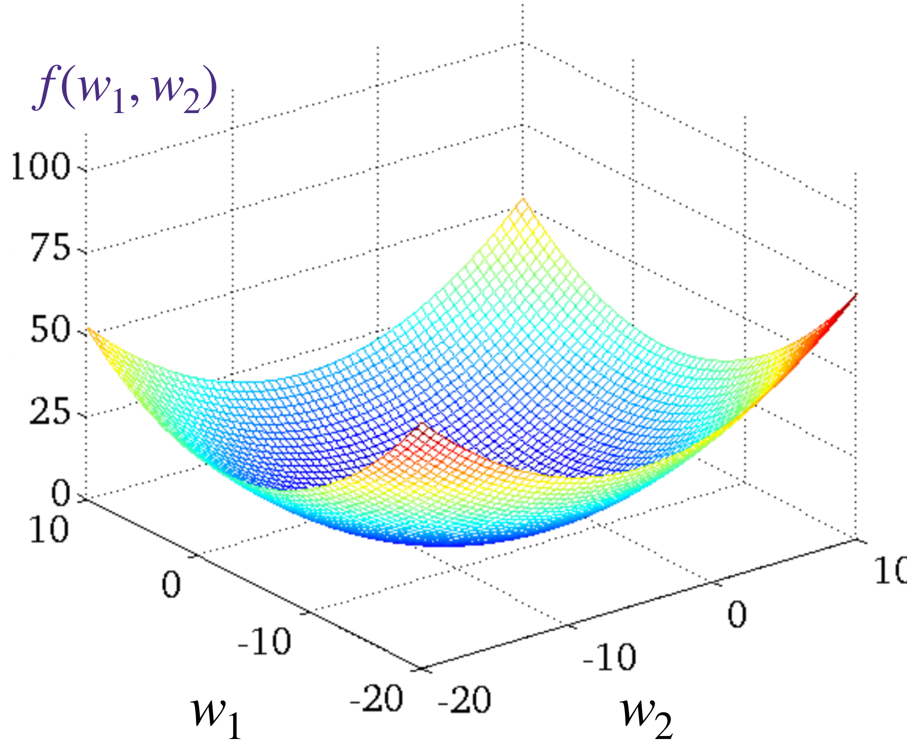
Note that as $k \to \infty$ we have $\frac{df(w)}{dw}\Big|_{w=w_k} \to 0$ (assuming small enough η)

Running example: Linear Regression

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

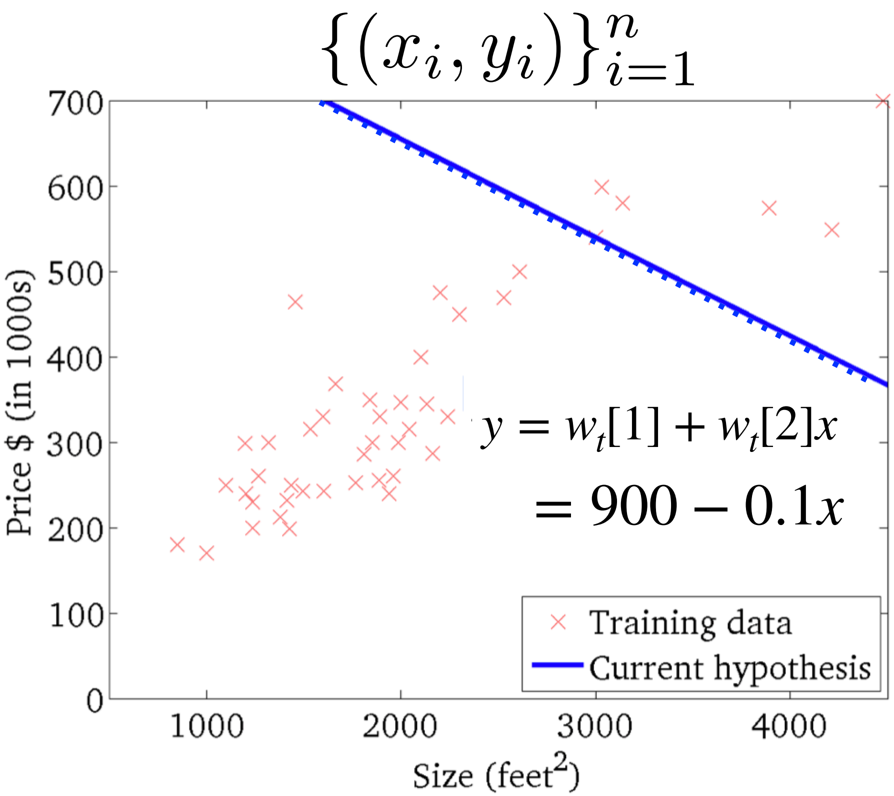
- **Learning model parameters:**

$$\hat{w}_{\text{LS}} = \arg\min_{w \in \mathbb{R}^d} \underbrace{\|\mathbf{y} - \mathbf{X}w\|_2^2}_{f(w)}$$

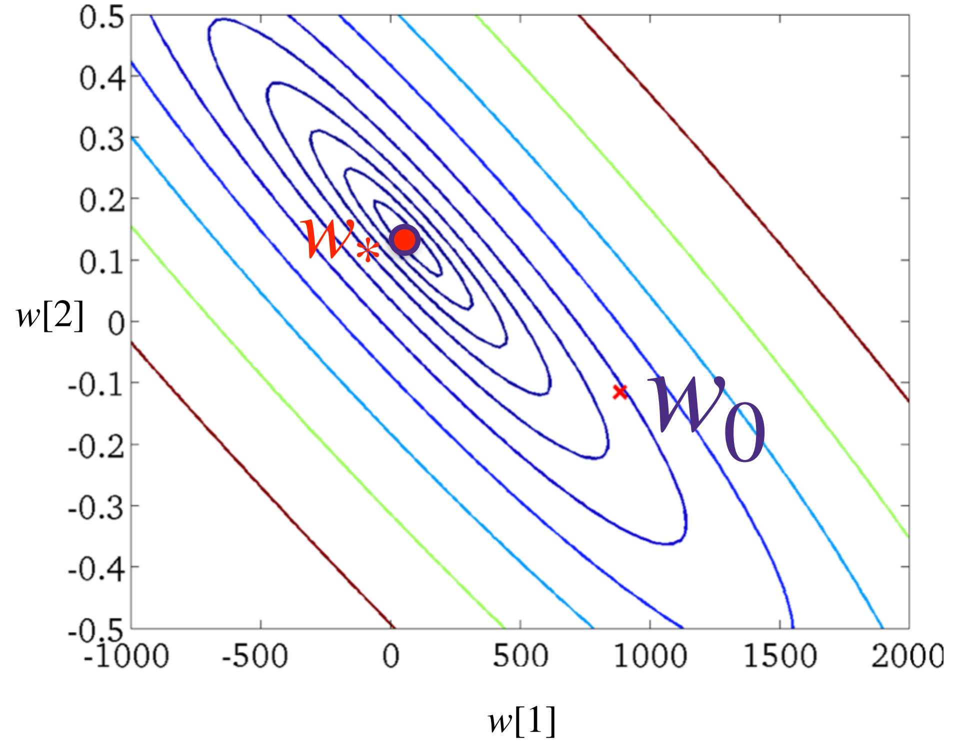


- **Gradient descent:**
  - Initialize: $w_0 = 0$
  - **For** t=0,1,2,…
    - $w_{t+1} \leftarrow w_t - \eta \cdot \nabla_w f(w_t)$

- $w_0 = (900, -0.1)$
- **For** t=0,1,2,...
  - $w_{t+1} \leftarrow w_t - \eta \cdot \nabla_w f(w_t)$



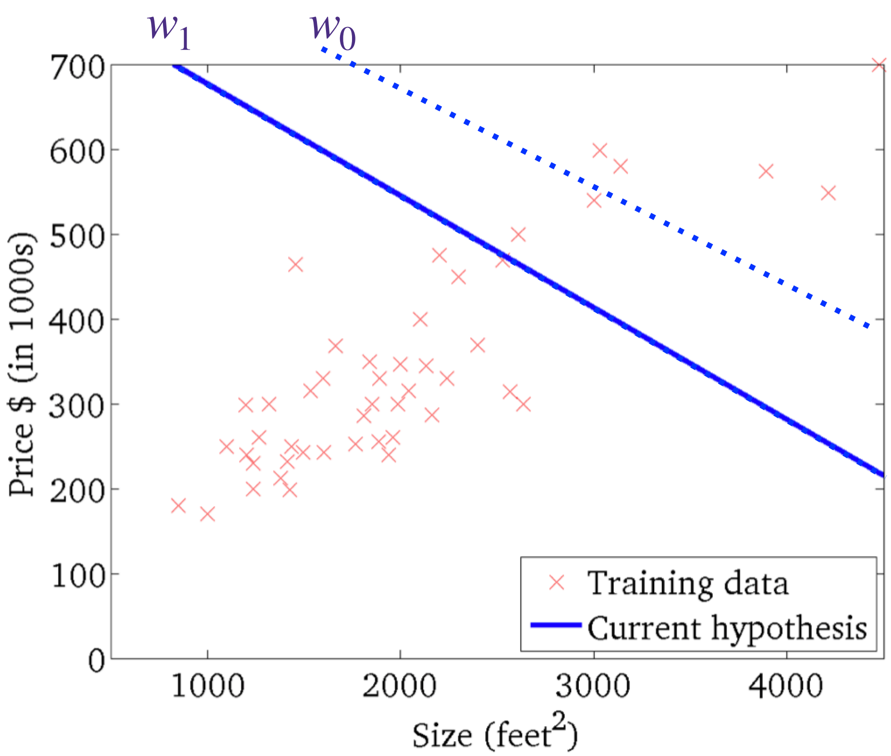
Evolution of the predictor



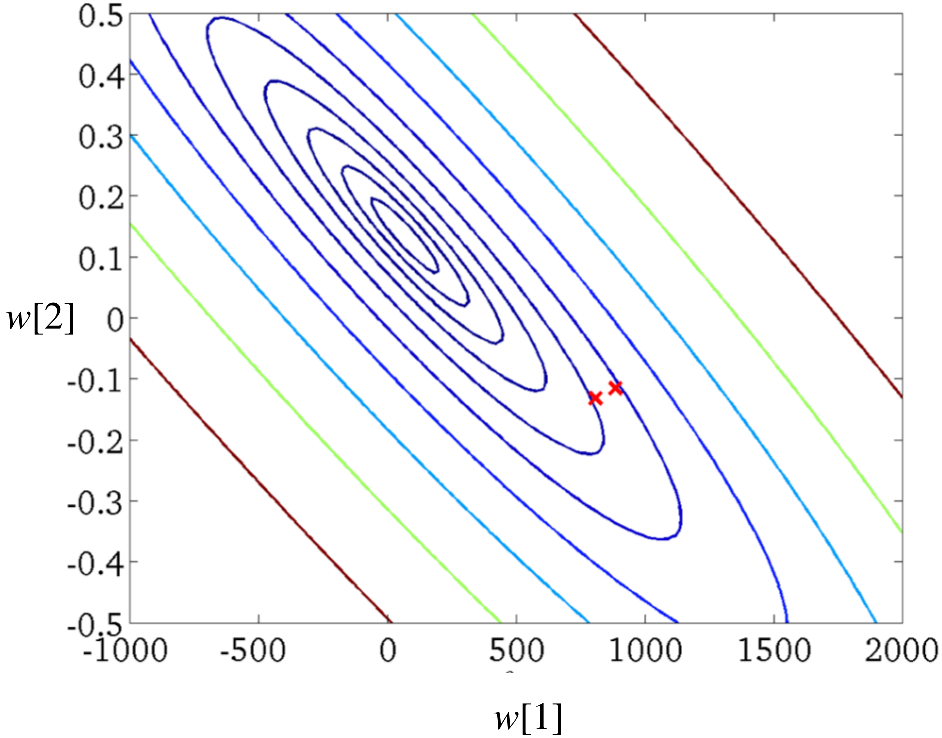
GD dynamics in the Parameter space

- Which direction will the GD move?

- $w_0 = (900, -0.1)$
- **For** t=0,1,2,…
  - $w_{t+1} \leftarrow w_t - \eta \cdot \nabla_w f(w_t)$

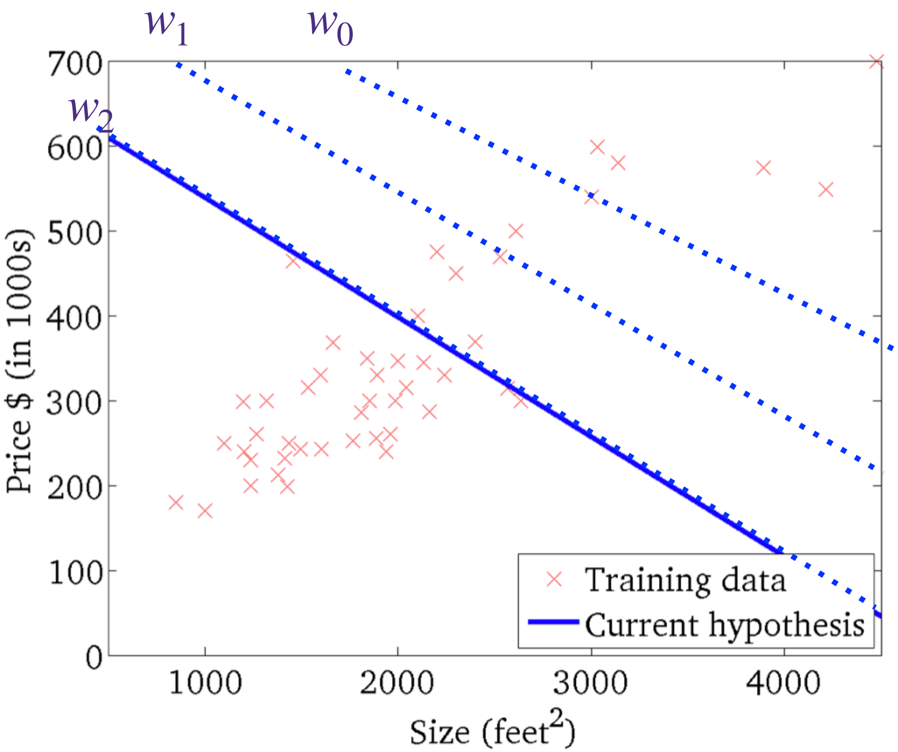


Evolution of the predictor

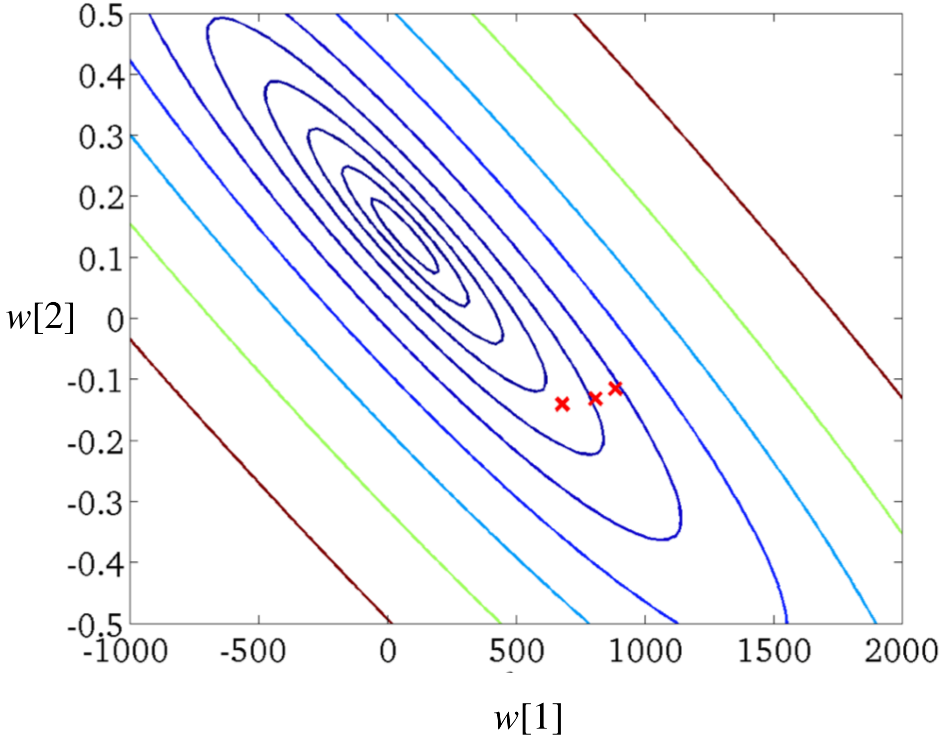


GD dynamics in the Parameter space

- $w_0 = (900, -0.1)$
- **For** t=0,1,2,…
  - $w_{t+1} \leftarrow w_t - \eta \cdot \nabla_w f(w_t)$

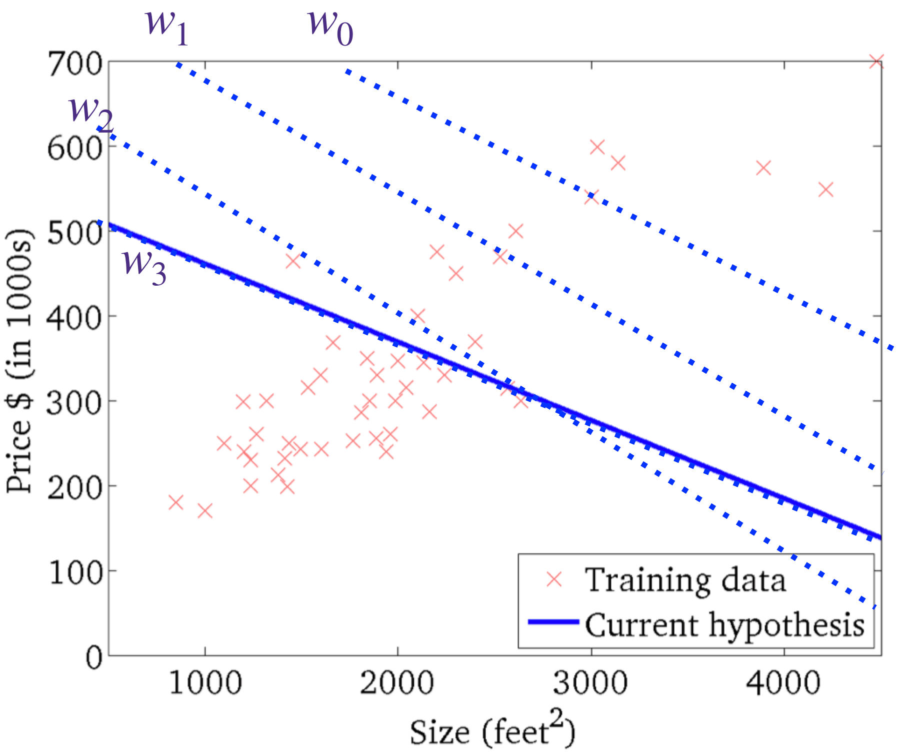


Evolution of the predictor

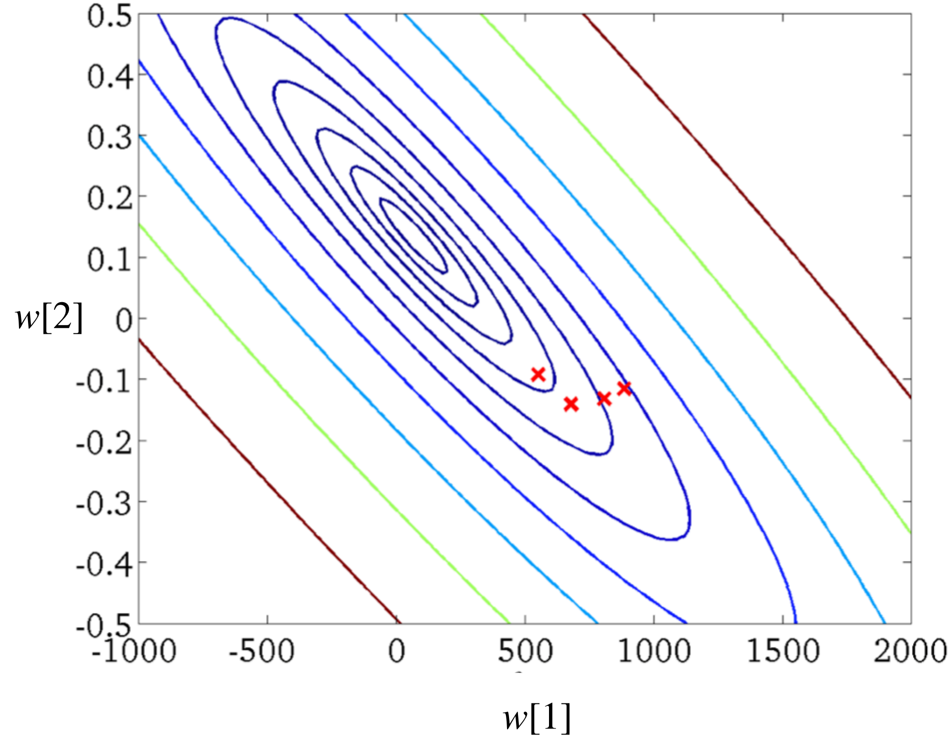


GD dynamics in the Parameter space

- $w_0 = (900, -0.1)$
- **For** t=0,1,2,…
  - $w_{t+1} \leftarrow w_t - \eta \cdot \nabla_w f(w_t)$

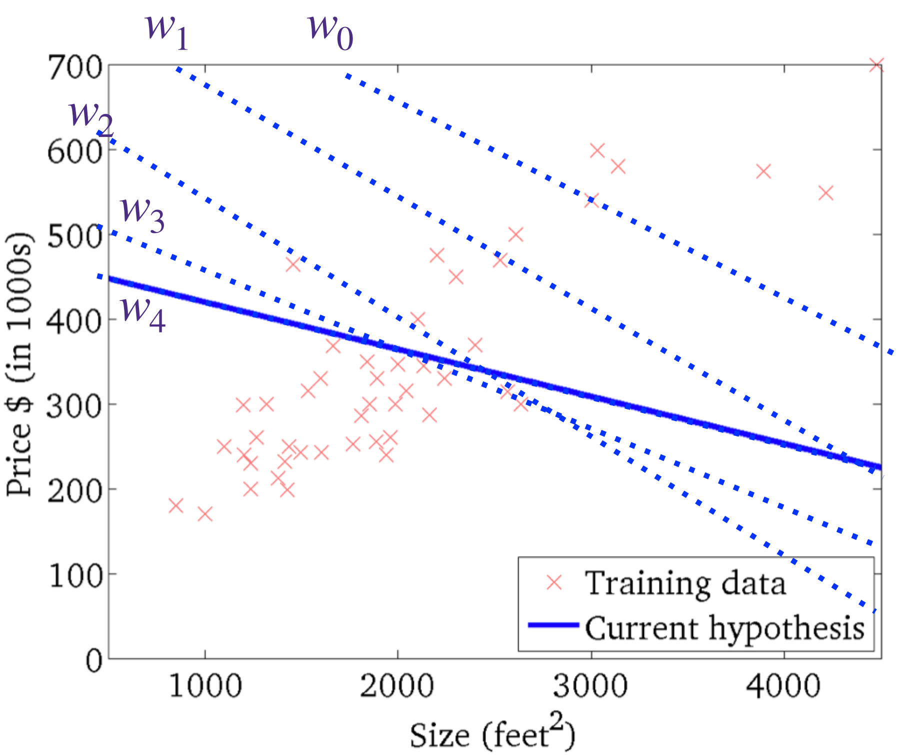


Evolution of the predictor

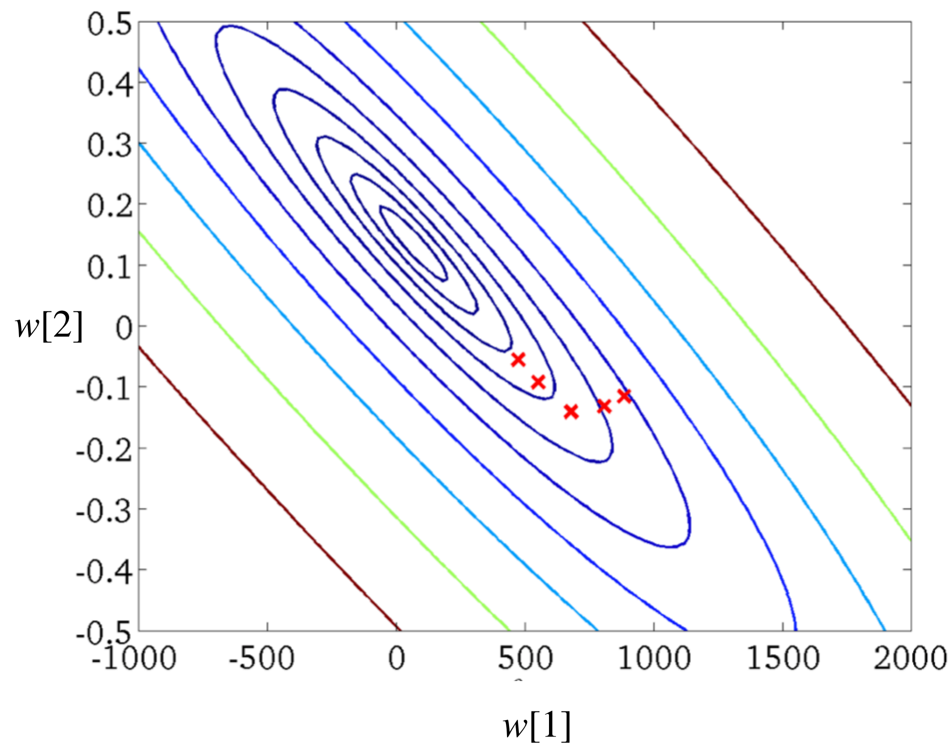


GD dynamics in the Parameter space

- $w_0 = (900, -0.1)$
- **For** t=0,1,2,...
  - $w_{t+1} \leftarrow w_t - \eta \cdot \nabla_w f(w_t)$

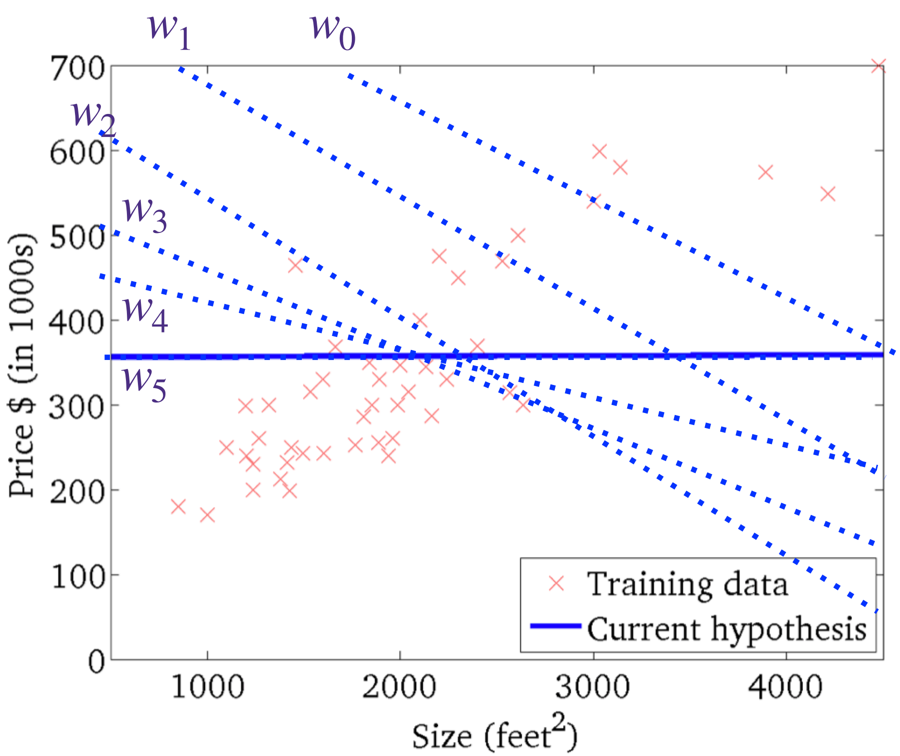


Evolution of the predictor

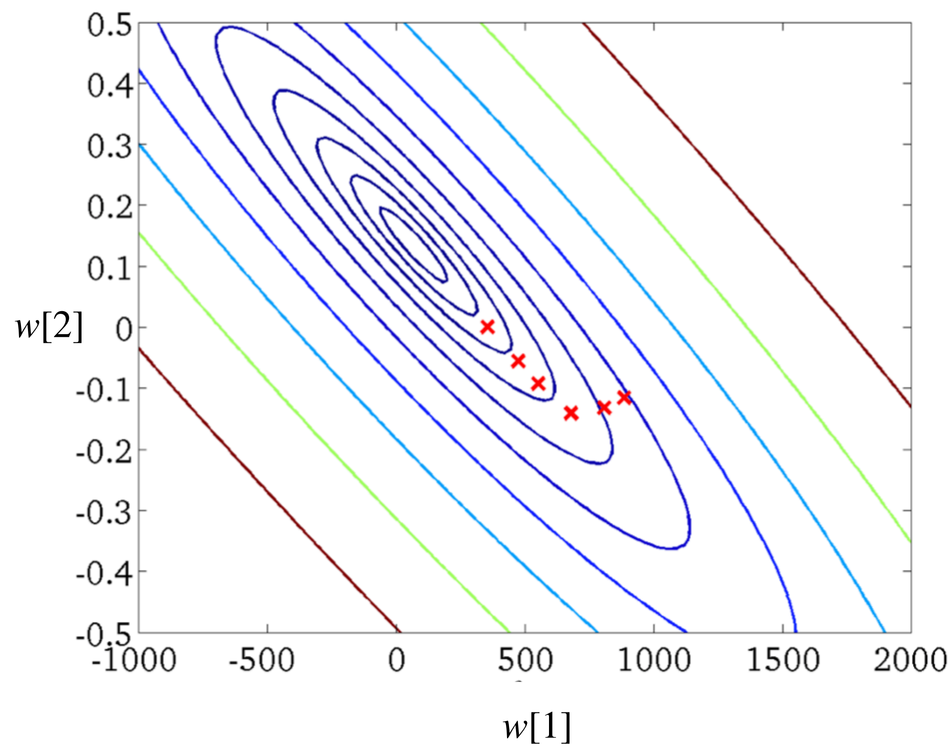


GD dynamics in the Parameter space

- $w_0 = (900, -0.1)$
- **For** t=0,1,2,…
  - $w_{t+1} \leftarrow w_t - \eta \cdot \nabla_w f(w_t)$

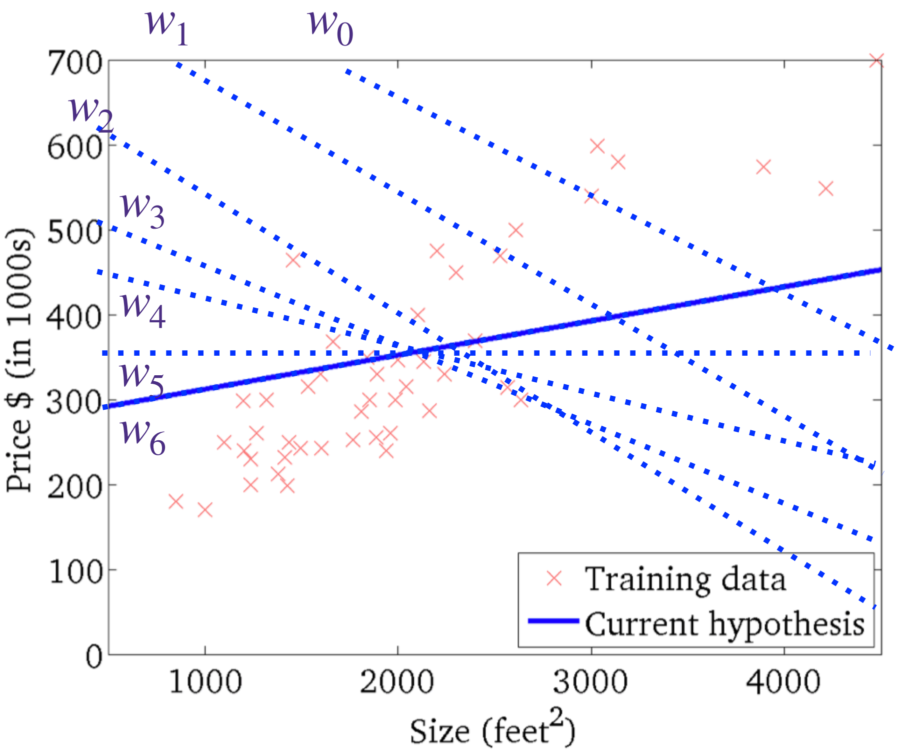


Evolution of the predictor

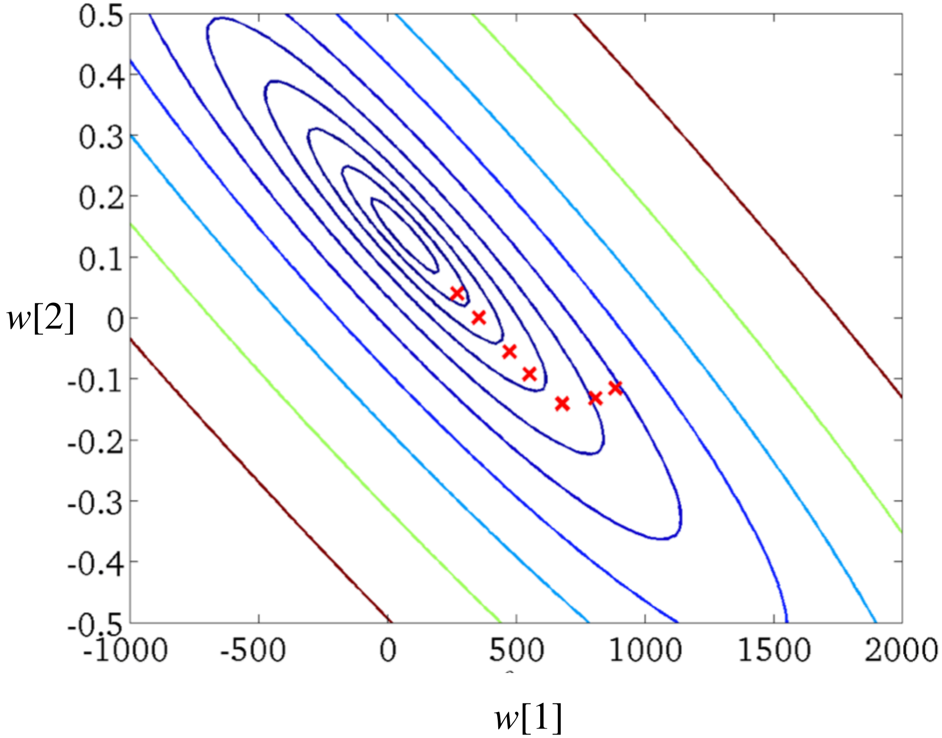


GD dynamics in the Parameter space

- $w_0 = (900, -0.1)$
- **For** t=0,1,2,…
  - $w_{t+1} \leftarrow w_t - \eta \cdot \nabla_w f(w_t)$

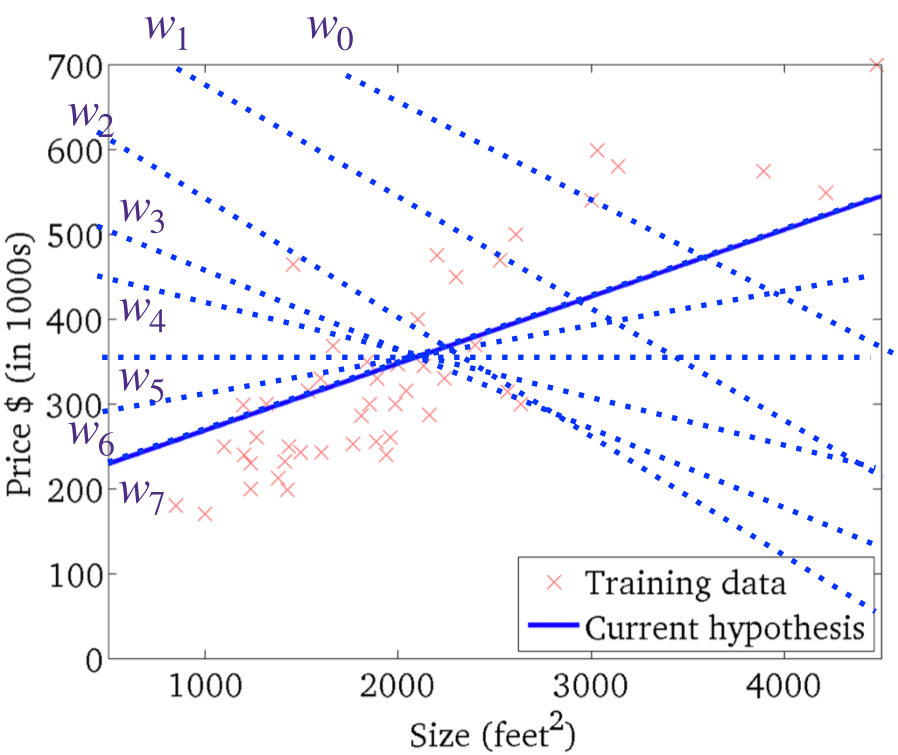


Evolution of the predictor

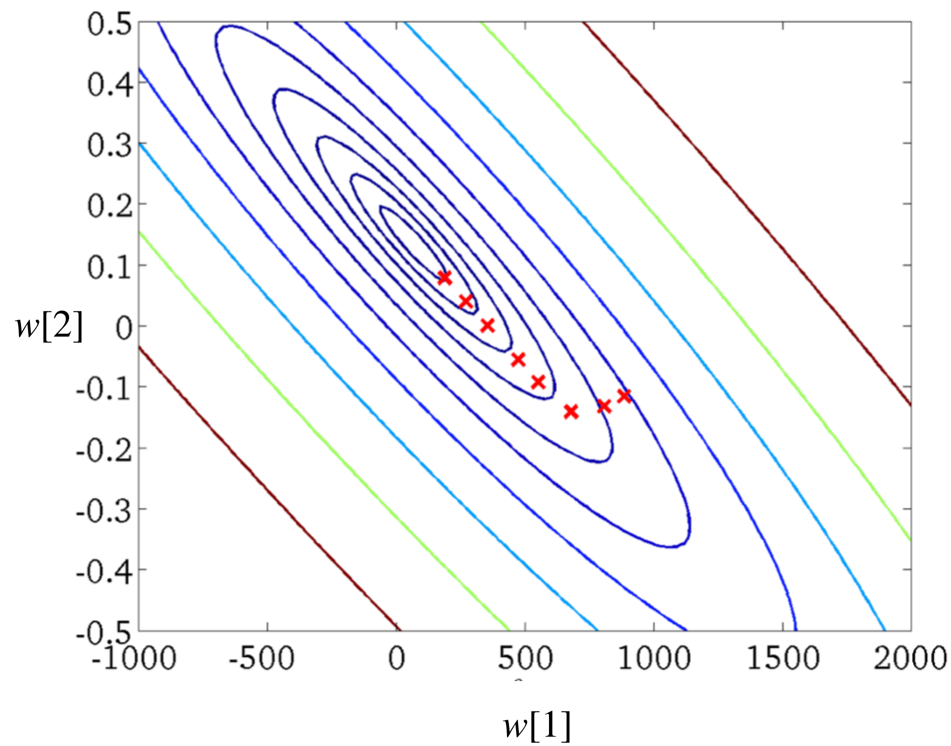


GD dynamics in the Parameter space

- $w_0 = (900, -0.1)$
- **For** t=0,1,2,...
  - $w_{t+1} \leftarrow w_t - \eta \cdot \nabla_w f(w_t)$

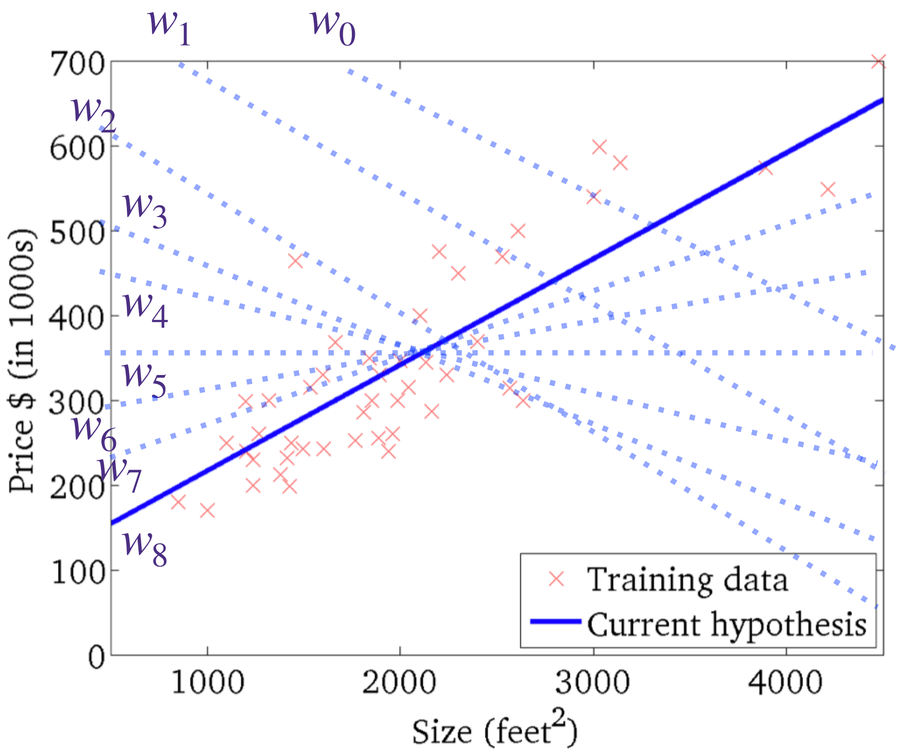


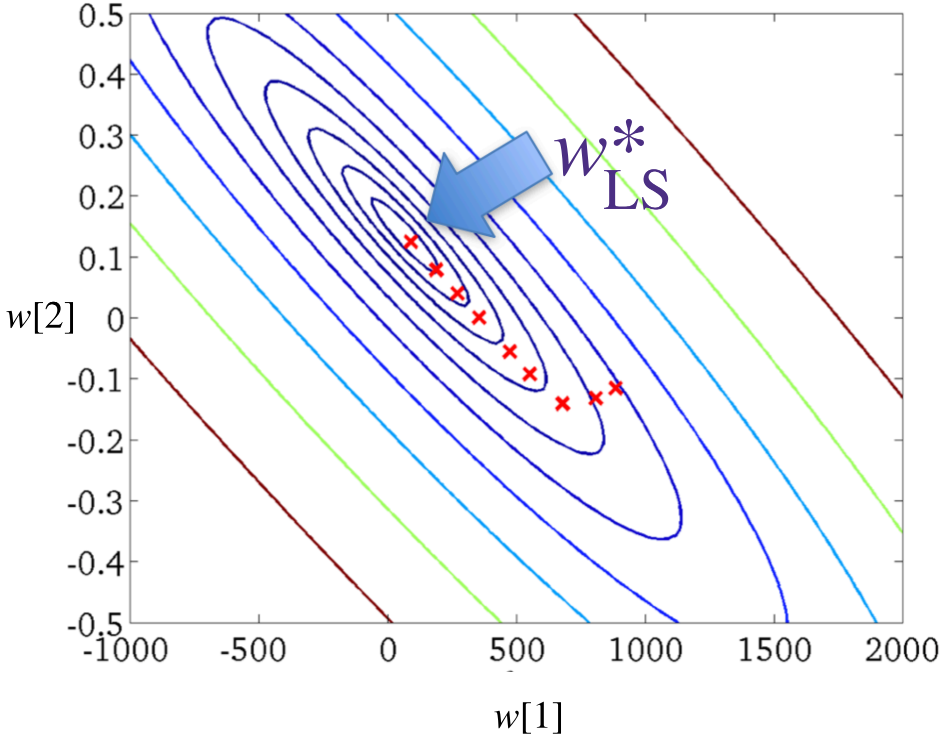
Evolution of the predictor



GD dynamics in the Parameter space

- $w_0 = (900, -0.1)$
- **For** t=0,1,2,…
  - $w_{t+1} \leftarrow w_t - \eta \cdot \nabla_w f(w_t)$





Evolution of the predictor

GD dynamics in the Parameter space

Gradient Descent Practicalities

Practicalities

- How to initialize w_0?
  - Usually pick something at random
  - or if you have a good guess start there
- How to choose η?
  - Step size matters!
    - What happens if it is too small?
    - What happens if it is too large?
  - How to choose?
    - Special case: Solve for optimal
    - General case: Hyperparameter tuning (another one???)
- When to stop?
  - Stop when convergence is reached
  - Or stop after some fixed number of iterations (also hyperparameter 😔)

Gradient Descent Algorithm

- Initialize: w_0
- **For** t=0,1,2,...
  - $w_{t+1} \leftarrow w_t - \eta \cdot \nabla_w f(w_t)$