Simple Variable Selection
LASSO: Sparse Regression

Sparsity

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

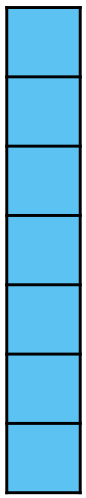
- Vector w is sparse, if many entries are zero

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?

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

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

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

Finding best subset: Exhaustive

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

Finding best subset: Greedy

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

Forward Greedy

1: $T \leftarrow \emptyset$

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

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

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

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

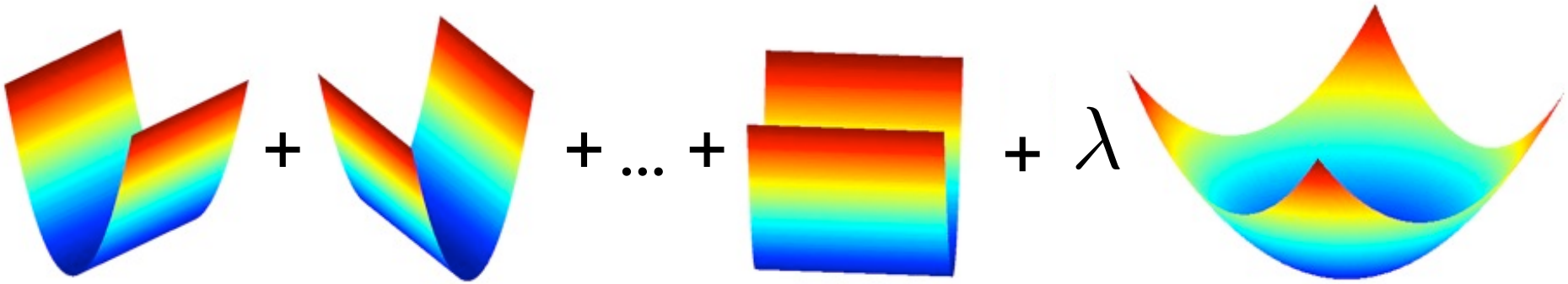
Combining forward and backward steps:
In forward algorithm, insert steps to remove features no longer as important

Lots of other variants, too.

Finding best subset: Regularize

Ridge regression makes coefficients small

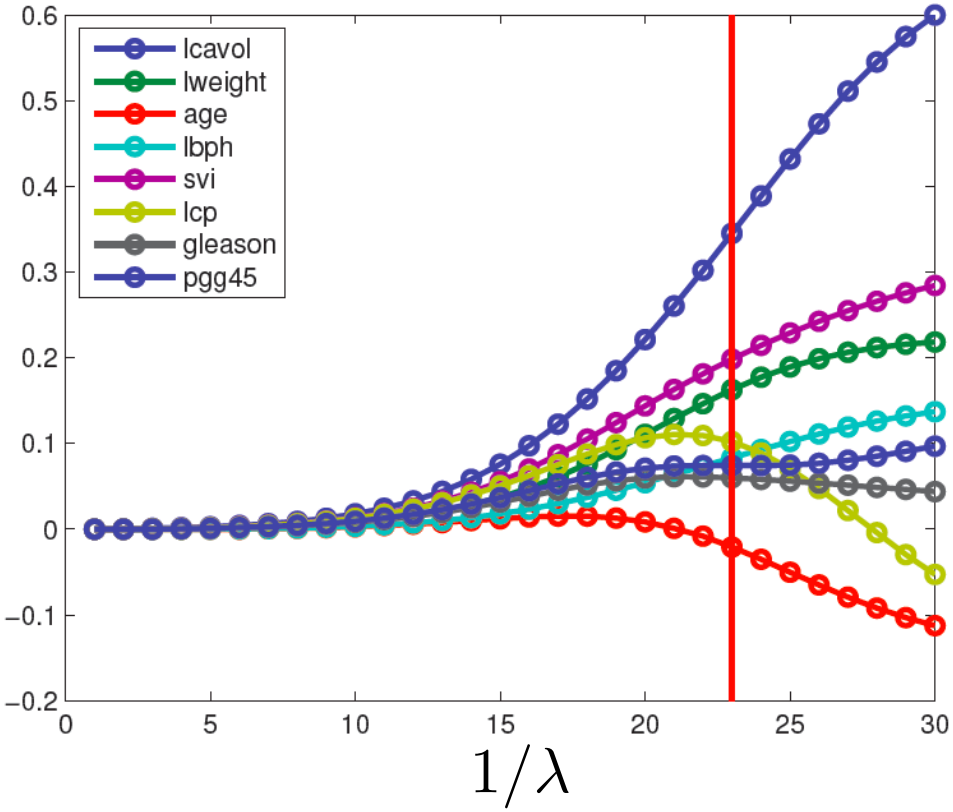
$$\widehat{w}_{ridge} = \arg\min_{w} \sum_{i=1}^{n} \left(y_i - x_i^T w\right)^2 + \lambda ||w||_2^2$$



Finding best subset: Regularize

Ridge regression makes coefficients small

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

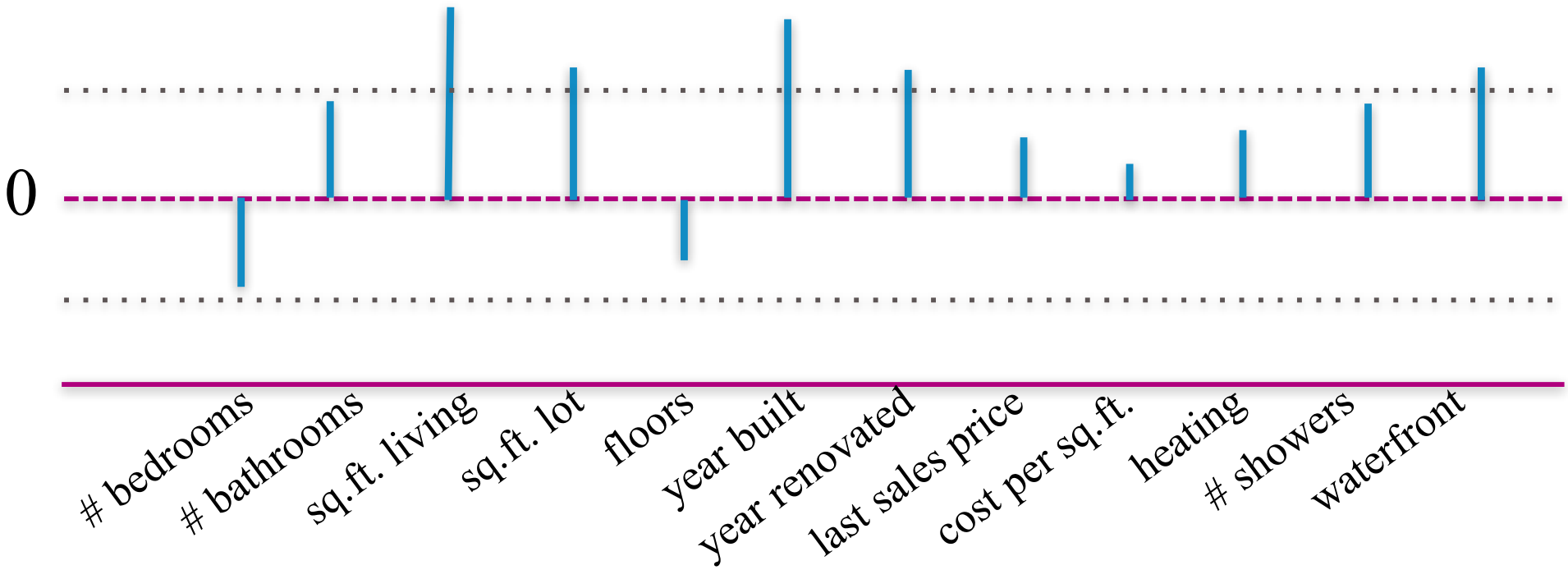


From
Kevin Murphy
textbook

Thresholded Ridge Regression

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

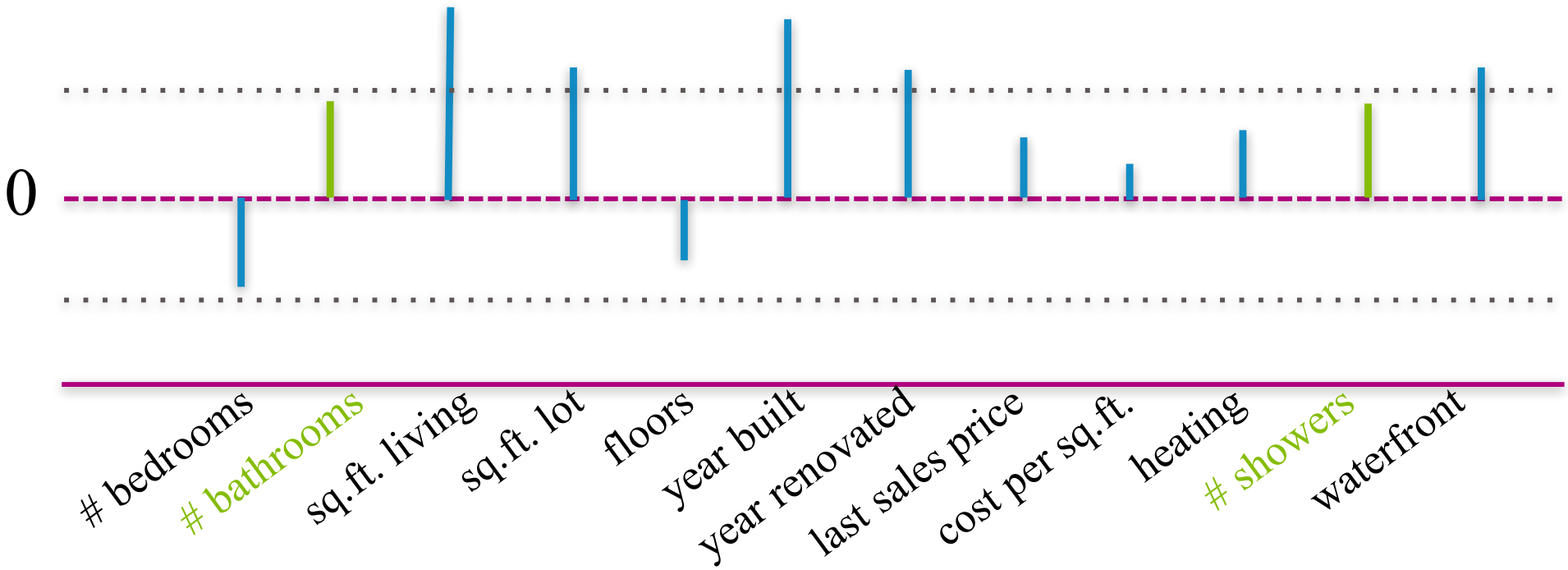
Why don't we just set **small** ridge coefficients to 0?



Thresholded Ridge Regression

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

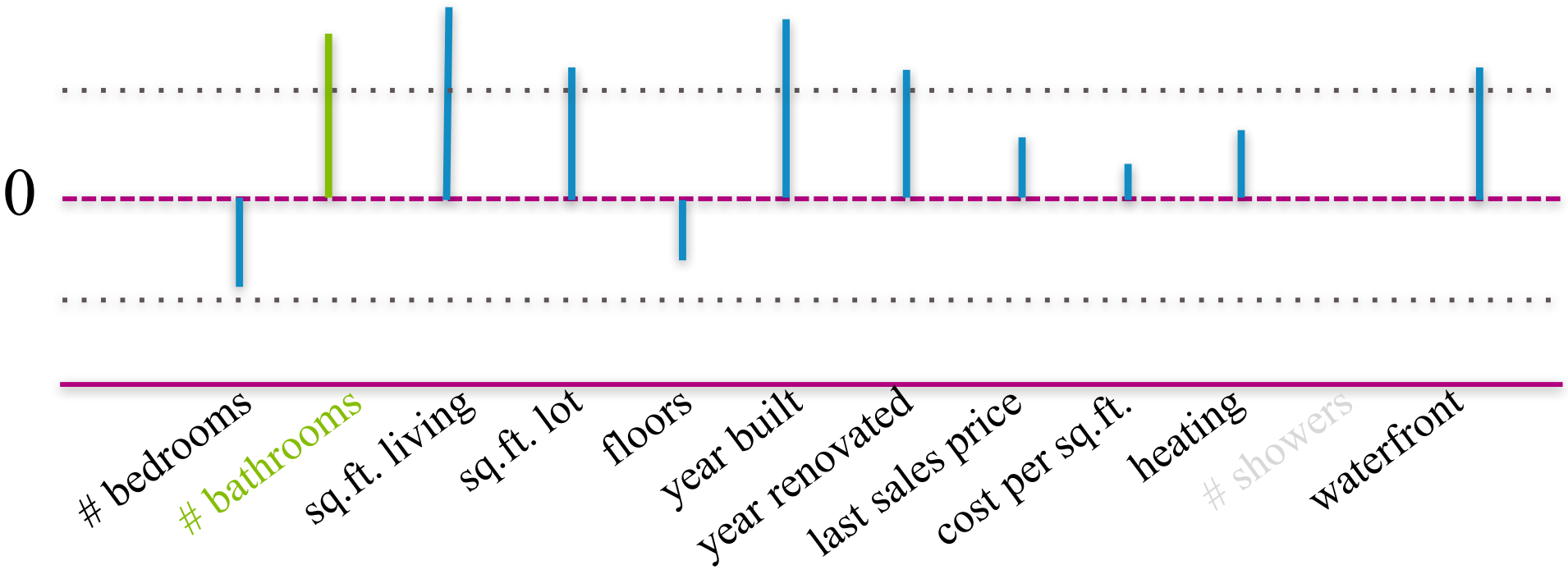
Consider two related features (bathrooms, showers)



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!

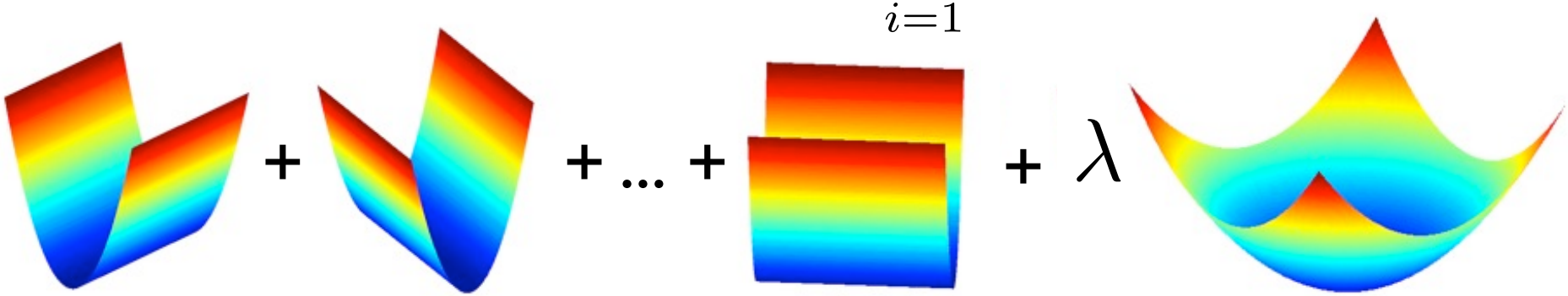


Can another regularizer perform selection automatically?

Recall Ridge Regression

- Ridge Regression objective:

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

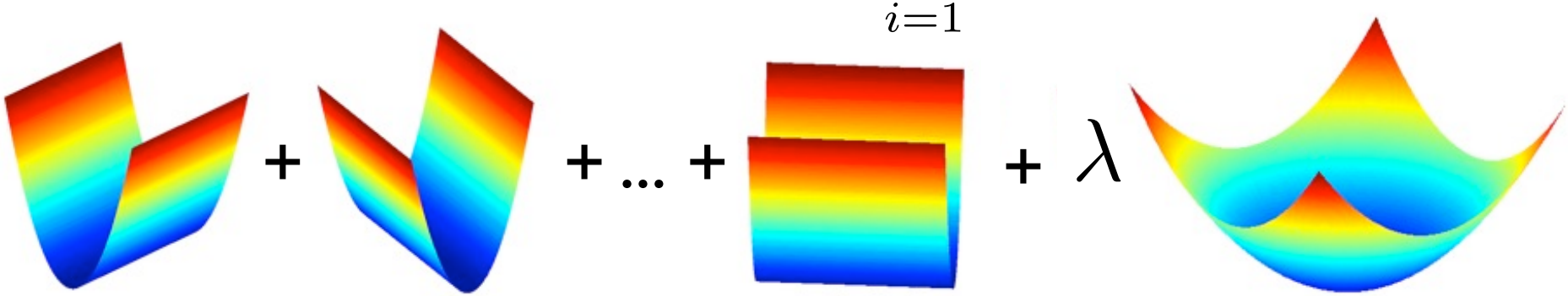


$$\|w\|_p = \left(\sum_{i=1}^{d} |w|^p\right)^{1/p}$$

Ridge vs. Lasso Regression

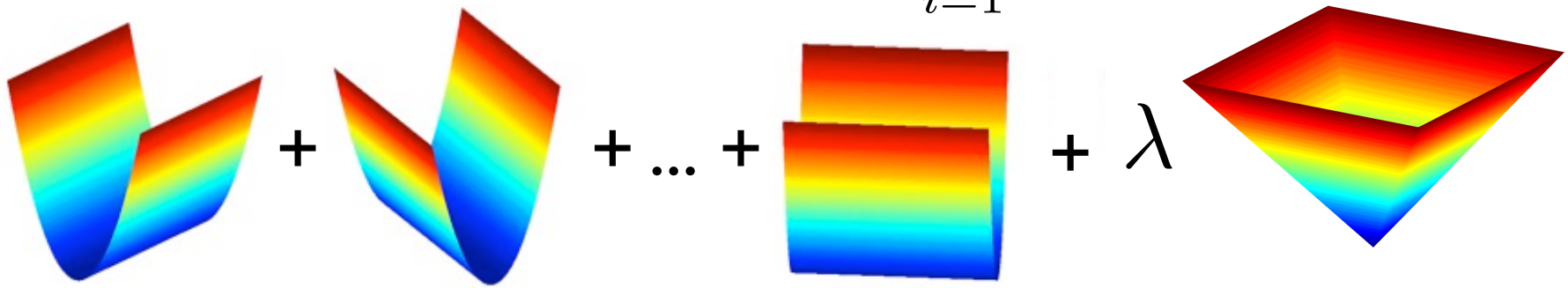
- Ridge Regression objective:

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

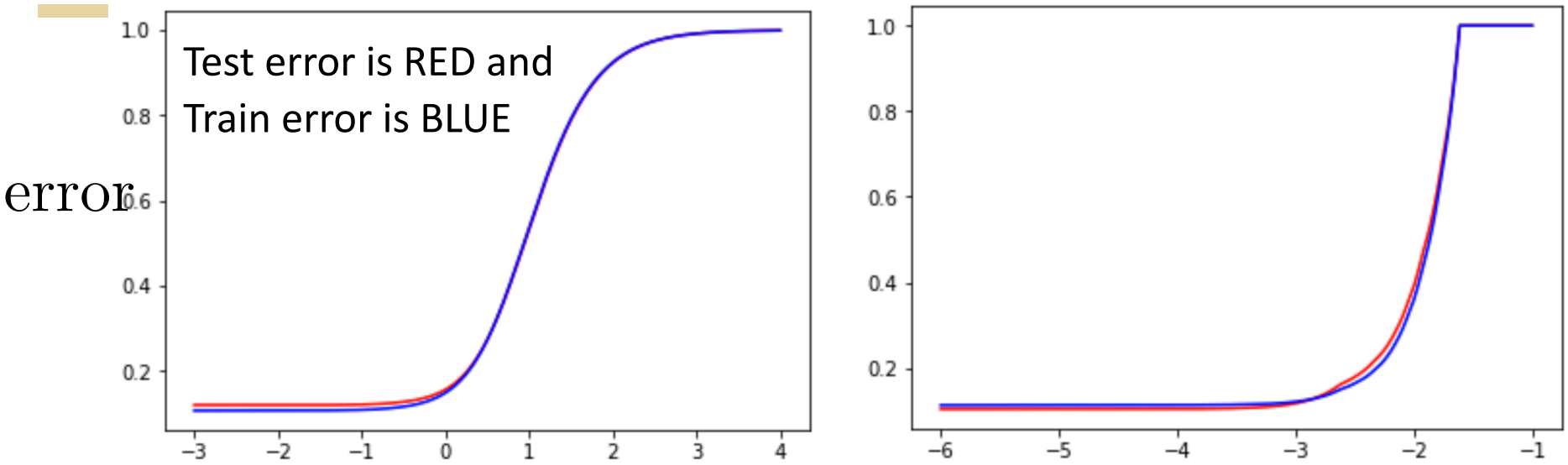


- Lasso objective:

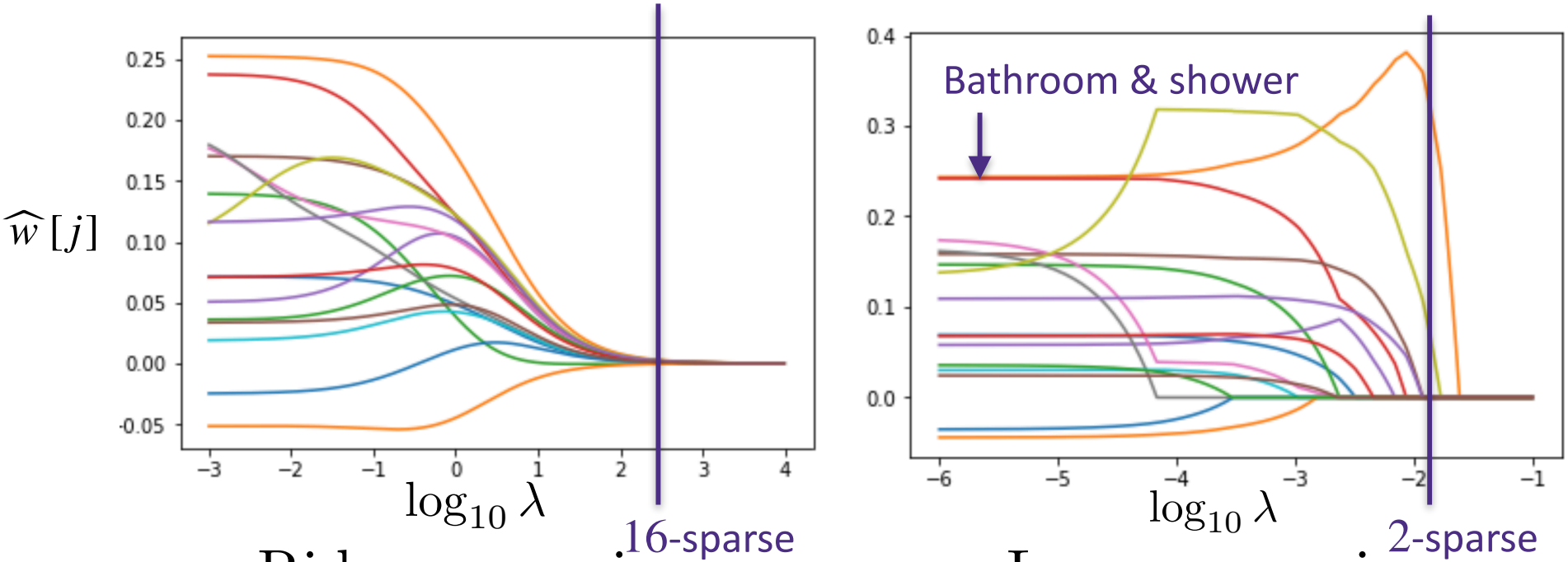
$$\widehat{w}_{lasso} = \arg\min_{w} \sum_{i=1}^{n} \left(y_i - x_i^T w\right)^2 + \lambda ||w||_1$$



Example: house price with 16 features



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



Ridge regression

Lasso regression

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?

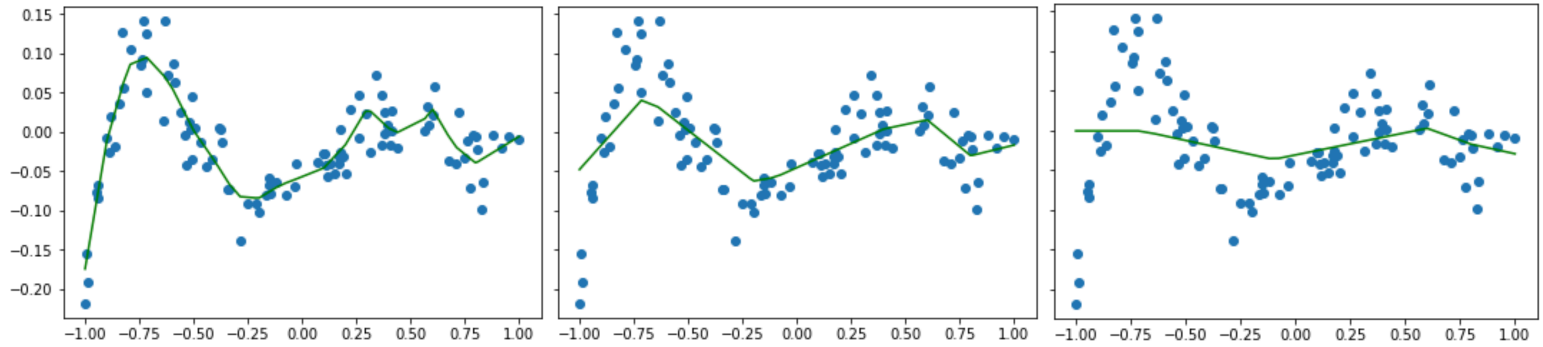
Example: piecewise-linear fit

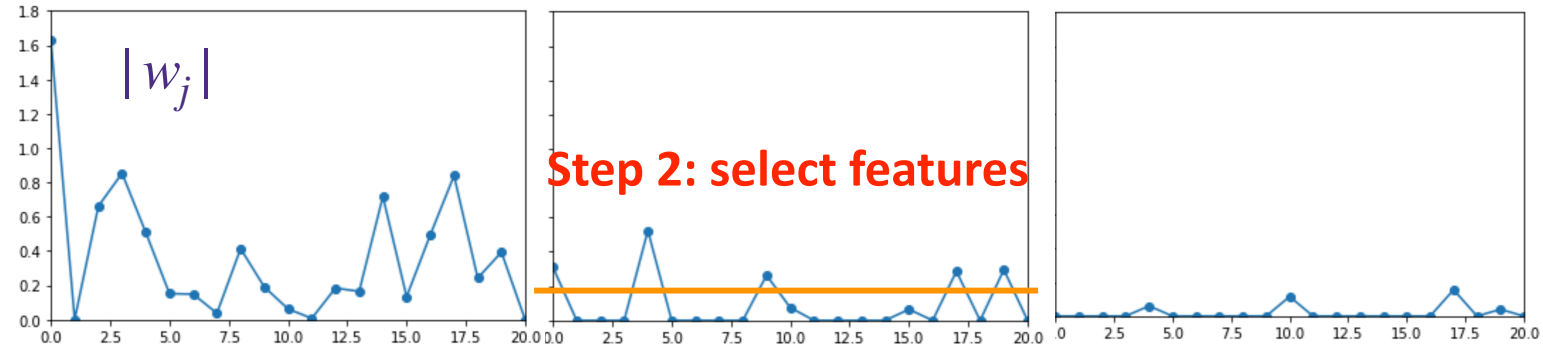
- We use Lasso on the piece-wise linear example

$$h_0(x) = 1$$
$$h_i(x) = [x + 1.1 - 0.1i]^+$$

Step 1: find optimal λ^*

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



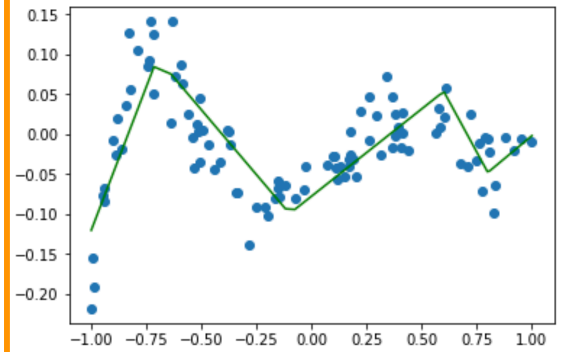


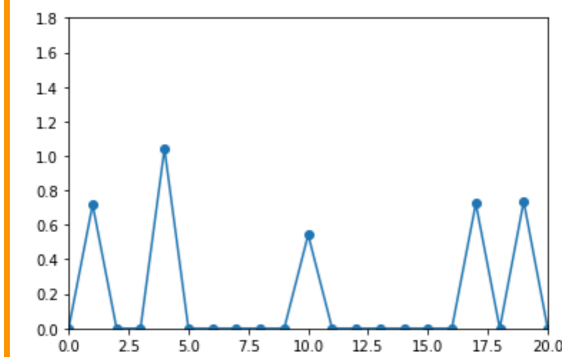
Step 2: select features

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

Step 3: retrain

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





$$\lambda = 0$$

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:

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

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

$$\text{Lasso} : r(w) = ||w||_1$$

- 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

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

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

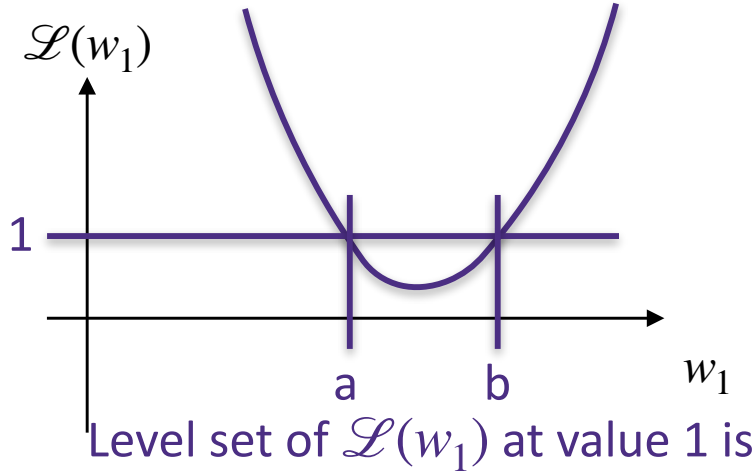
Why does Lasso give sparse solutions?

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

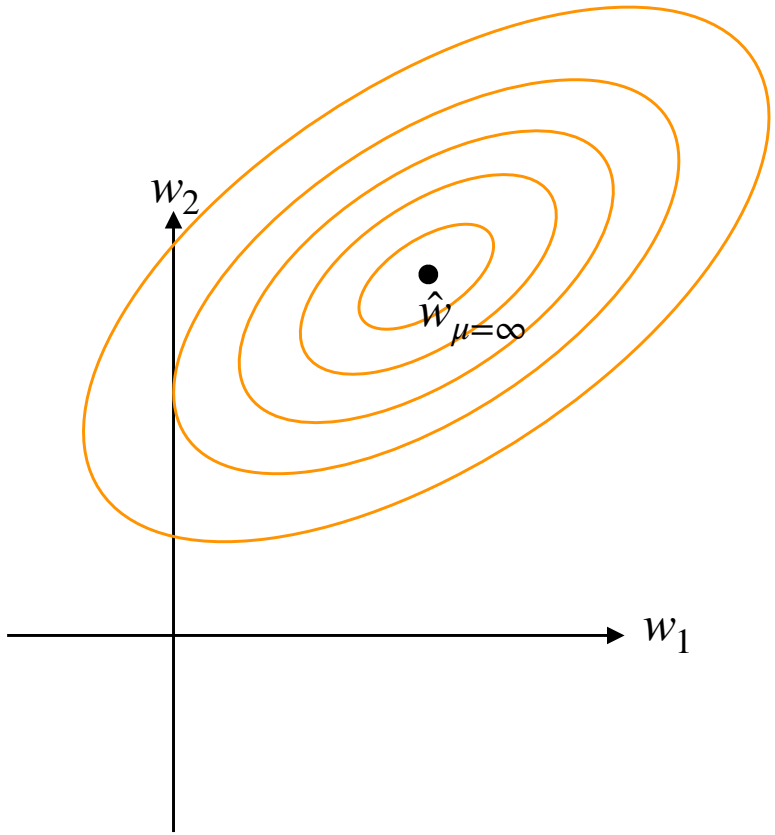
$$\text{subject to} \quad \|w\|_1 \leq \mu$$

- 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?

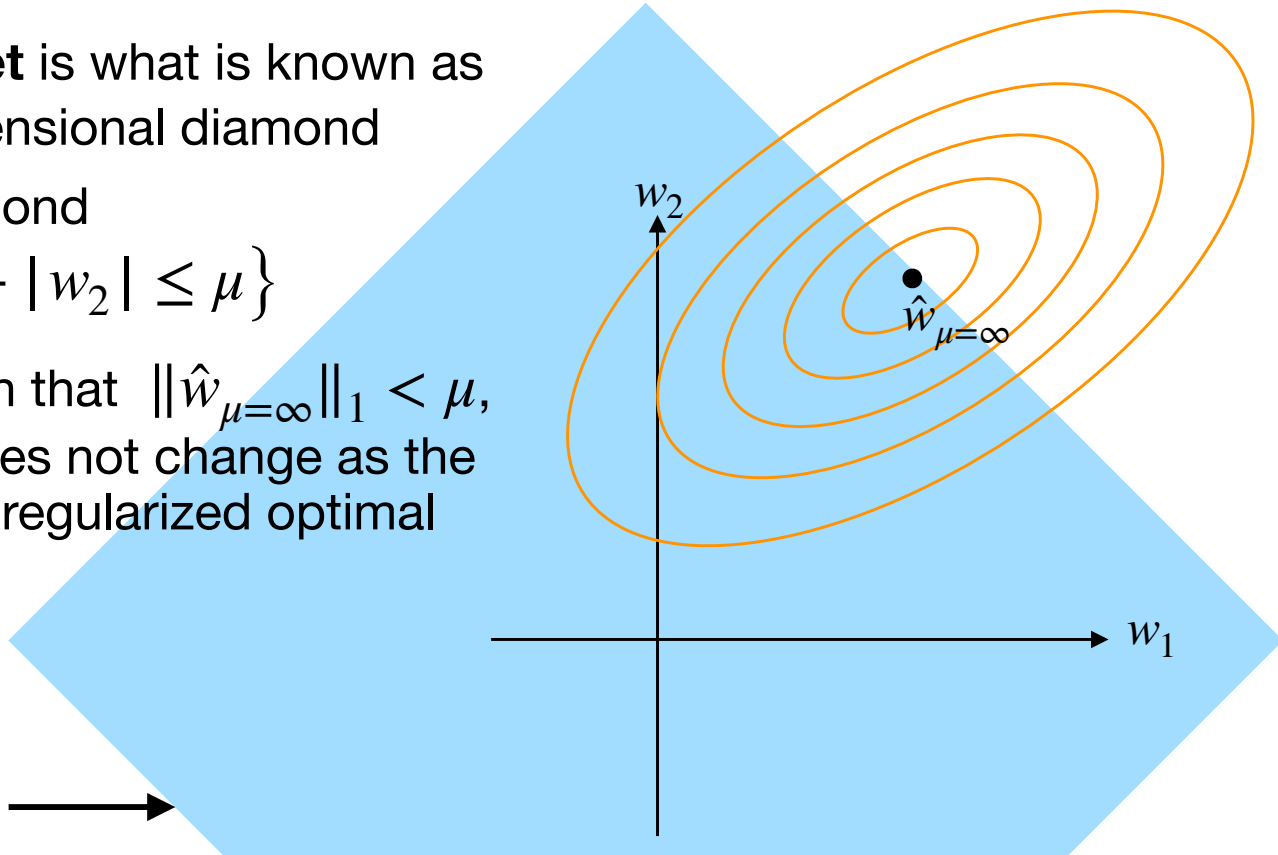
$$\text{minimize}_w \quad \sum_{i=1}^{n} (w^T x_i - y_i)^2$$

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

- 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



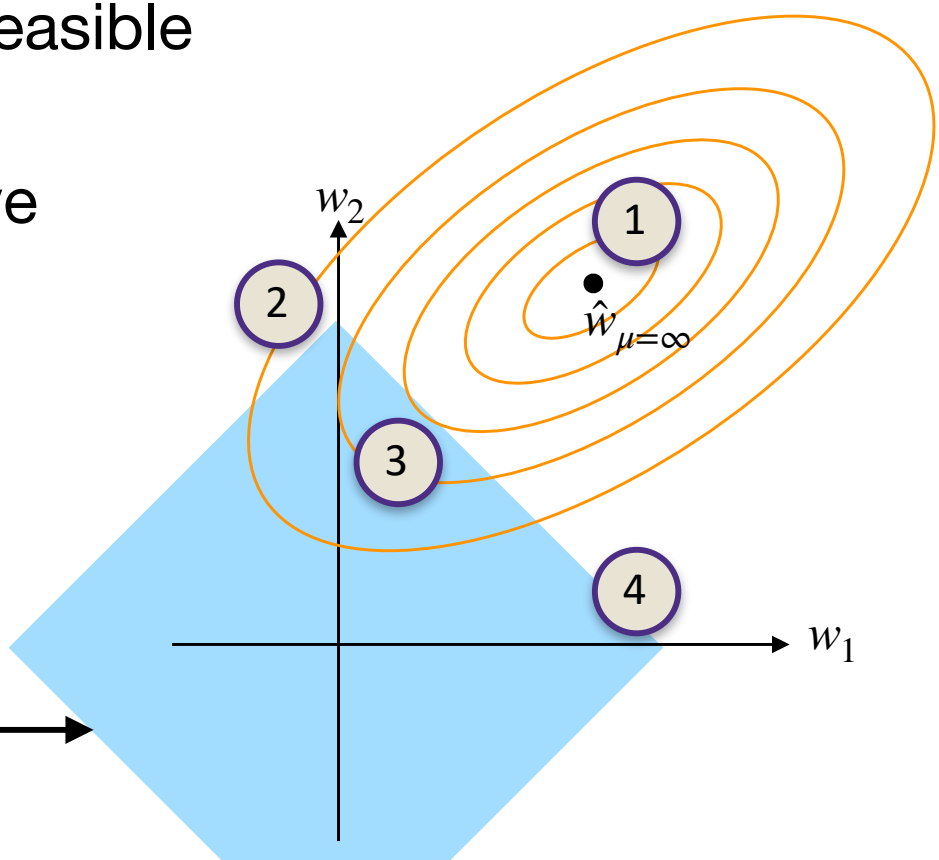
feasible set: $\{w \in \mathbb{R}^2 \mid \|w\|_1 \leq \mu\}$ ⟶

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 is ?



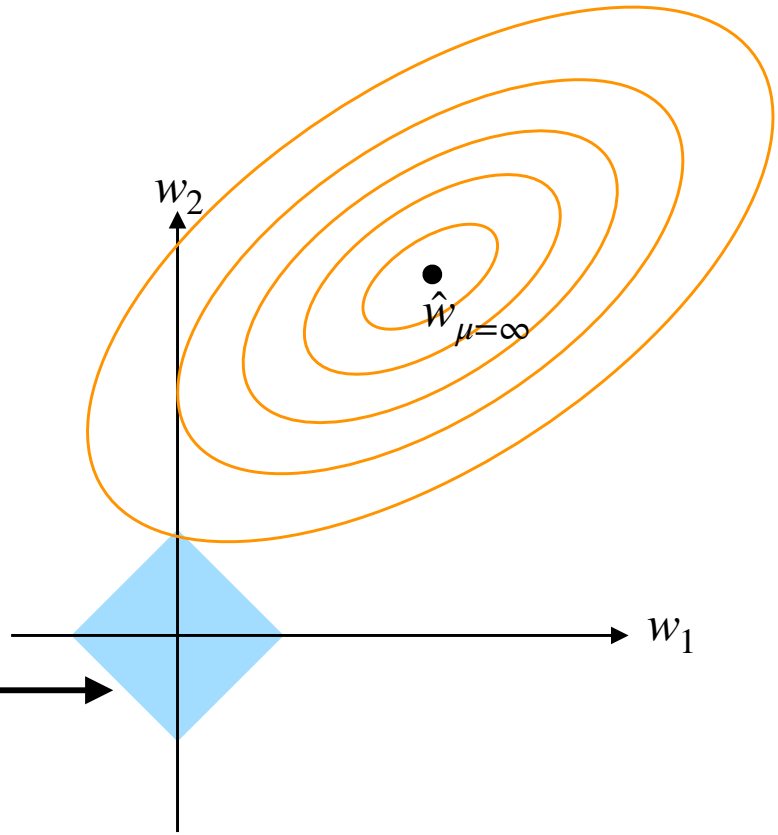
feasible set: $\{w \in \mathbb{R}^2 \mid \|w\|_1 \leq \mu\}$

Why does Lasso give sparse solutions?

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

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

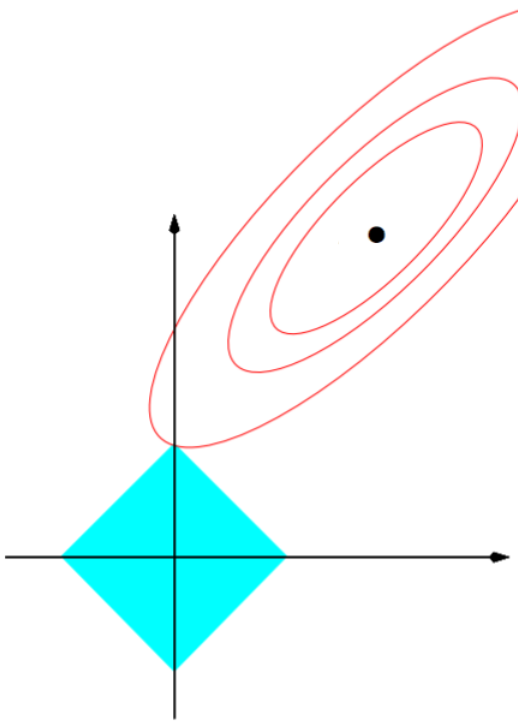
- For small enough μ, the optimal solution becomes **sparse**
- This is because the L_1-ball is "pointy",i.e., has sharp edges aligned with the axes

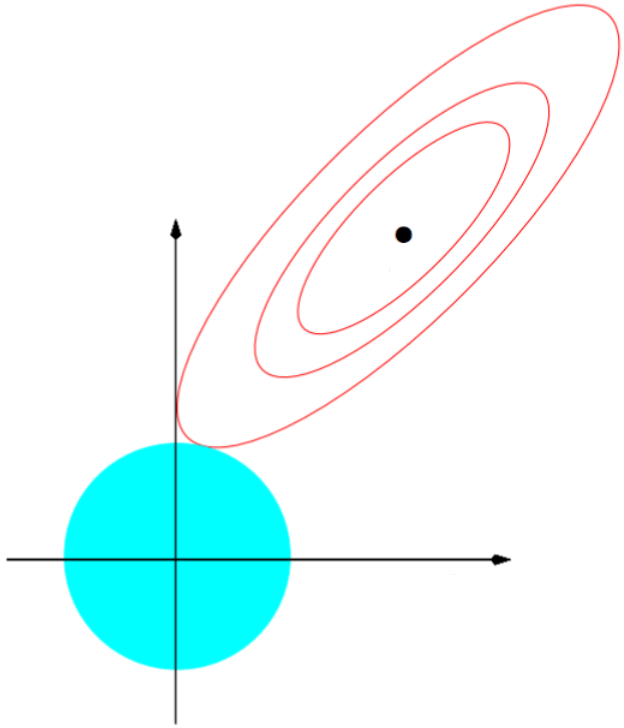


feasible set: $\{w \in \mathbb{R}^2 \mid \|w\|_1 \leq \mu\}$

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”





$$\text{minimize}_w \sum_{i=1}^{n} (w^T x_i - y_i)^2 \quad \text{subject to} \quad \|w\|_1 \leq \mu$$

$$\text{minimize}_w \sum_{i=1}^{n} (w^T x_i - y_i)^2 \quad \text{subject to} \quad \|w\|_2^2 \leq \mu$$

Memory vs compute

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

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

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

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

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

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

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

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?

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

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

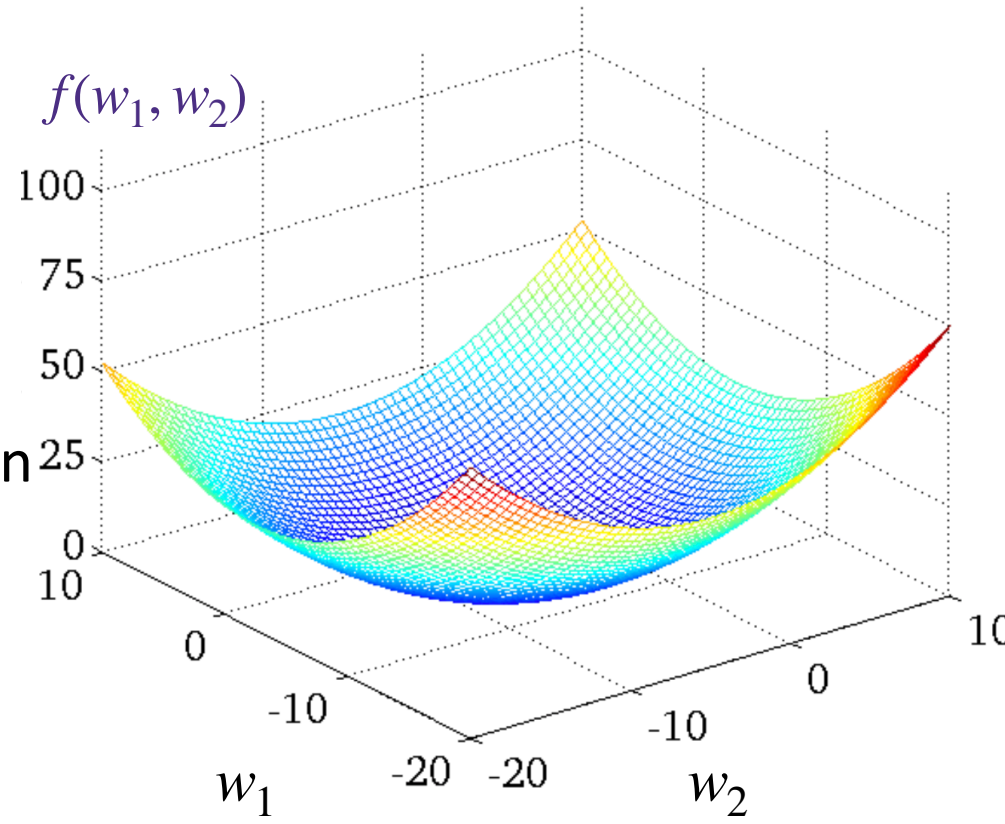
Running example: linear regression

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

- **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



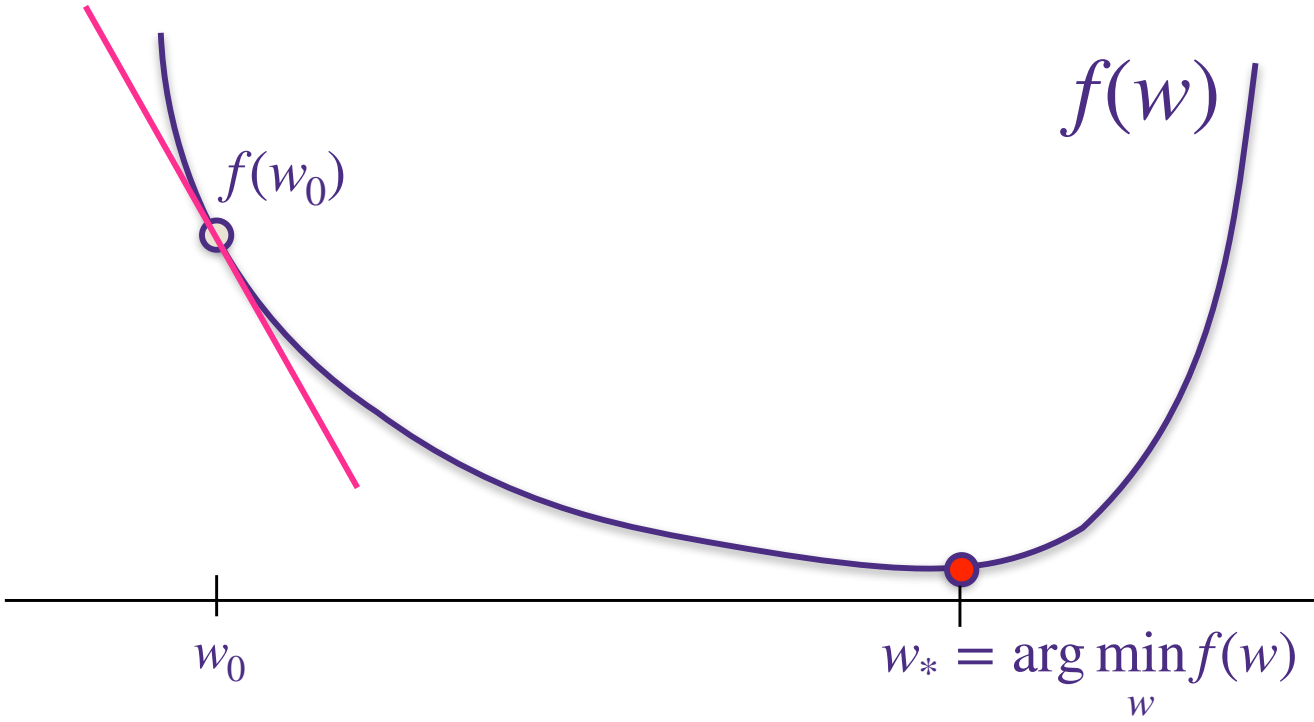
1-dimensional gradient descent

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

Taylor series approximation:
For w very close to w_0 we have

$$f(w_0) + (w - w_0)\frac{df(w)}{dw}\Big|_{w=w_0}$$

is very close to $f(w)$



1-dimensional gradient descent

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

Taylor series approximation:

For w very close to w_0 we have

$$f(w_0) + (w - w_0)\frac{df(w)}{dw}\Big|_{w=w_0}$$

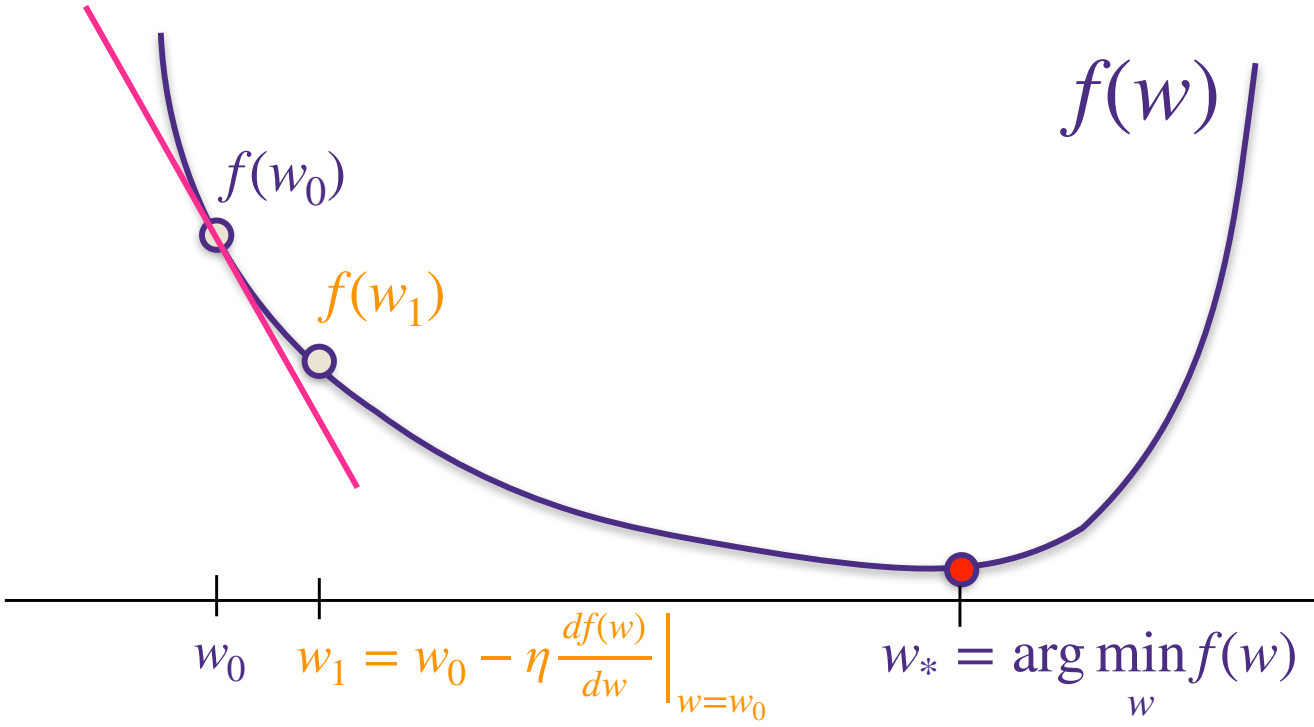
is very close to $f(w)$

Thus, for very small $\eta > 0$,

if $w_1 = w_0 - \eta \frac{df(w)}{dw}\Big|_{w=w_0}$ then

$$f(w_0) - \eta\left(\frac{df(w)}{dw}\Big|_{w=w_0}\right)^2$$

is very close to $f(w_1) < f(w_0)$



1-dimensional gradient descent

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

Taylor series approximation:

For w very close to w_0 we have

$$f(w_0) + (w - w_0)\frac{df(w)}{dw}\Big|_{w=w_0}$$

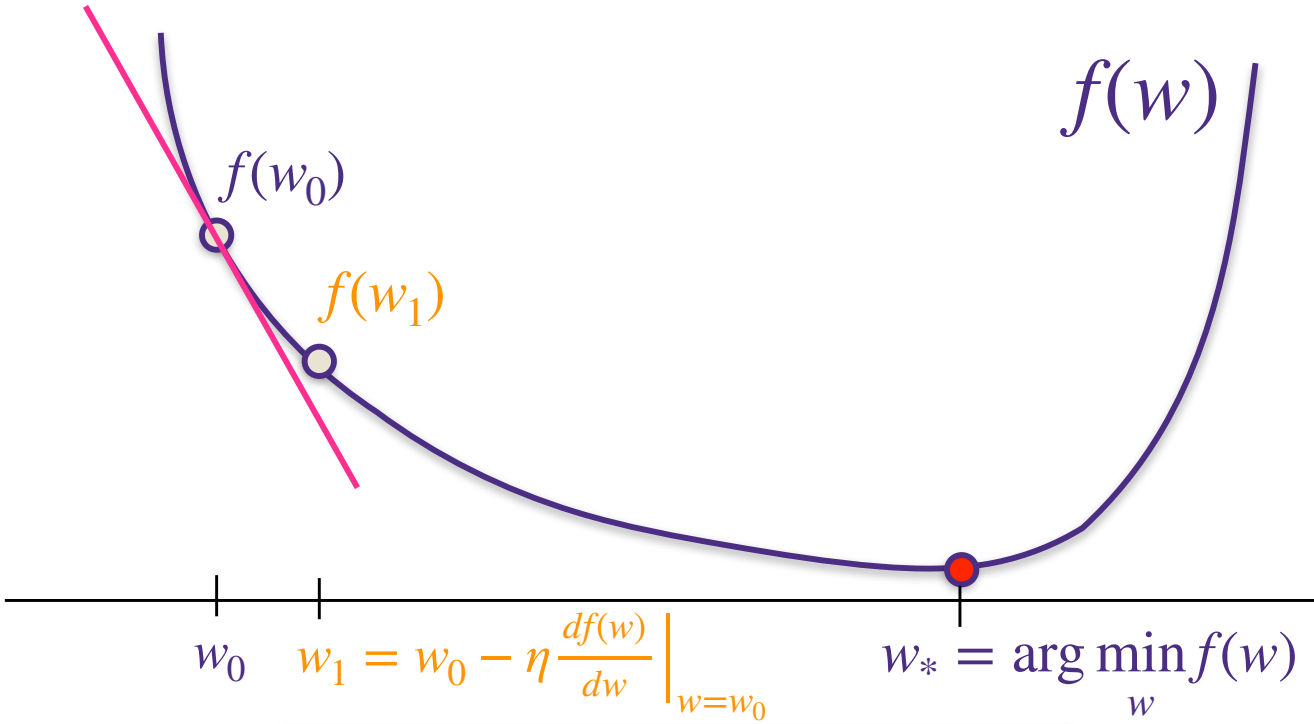
is very close to $f(w)$

Thus, for very small $\eta > 0$,

if $w_1 = w_0 - \eta \frac{df(w)}{dw}\Big|_{w=w_0}$ then

$$f(w_0) - \eta\left(\frac{df(w)}{dw}\Big|_{w=w_0}\right)^2$$

is very close to $f(w_1) < f(w_0)$



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?

Taylor series approximation:
For w very close to w_0 we have

$$f(w_0) + (w - w_0)\frac{df(w)}{dw}\Big|_{w=w_0}$$

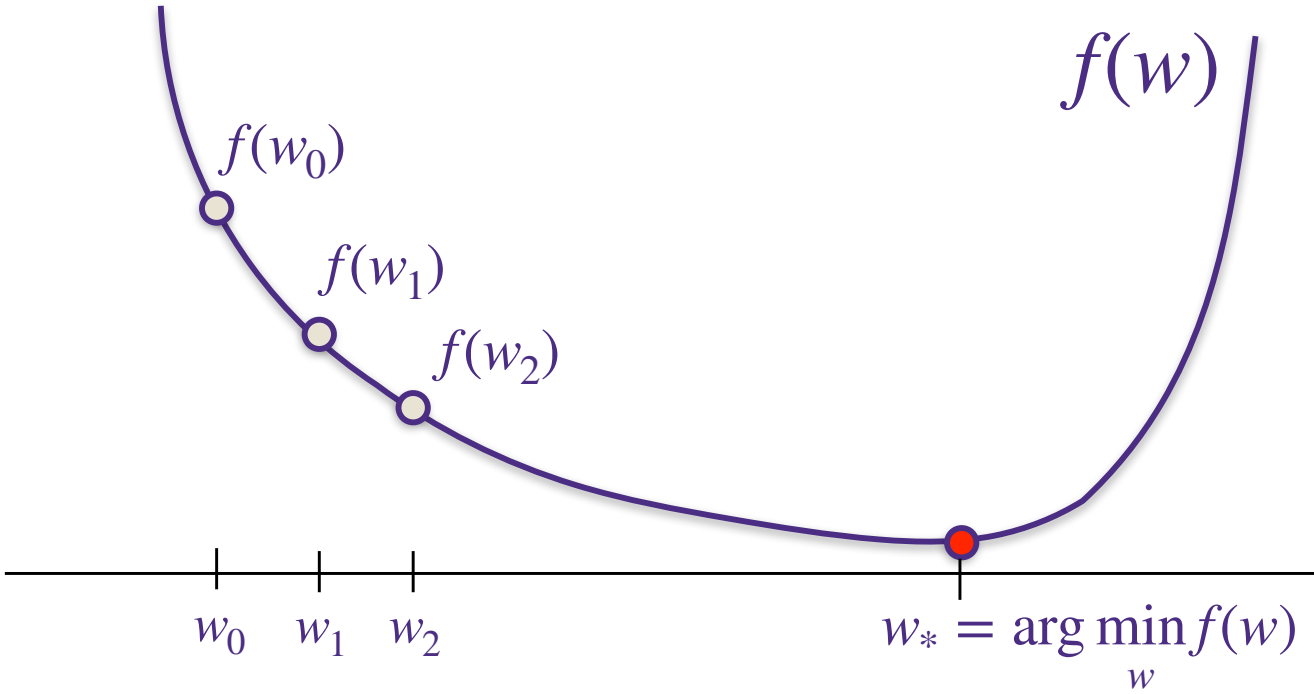
is very close to $f(w)$

Thus, for very small $\eta > 0$,

if $w_1 = w_0 - \eta \frac{df(w)}{dw}\Big|_{w=w_0}$ then

$$f(w_0) - \eta\left(\frac{df(w)}{dw}\Big|_{w=w_0}\right)^2$$

is very close to $f(w_1) < f(w_0)$



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?

Taylor series approximation:
For w very close to w_0 we have

$$f(w_0) + (w - w_0)\frac{df(w)}{dw}\Big|_{w=w_0}$$

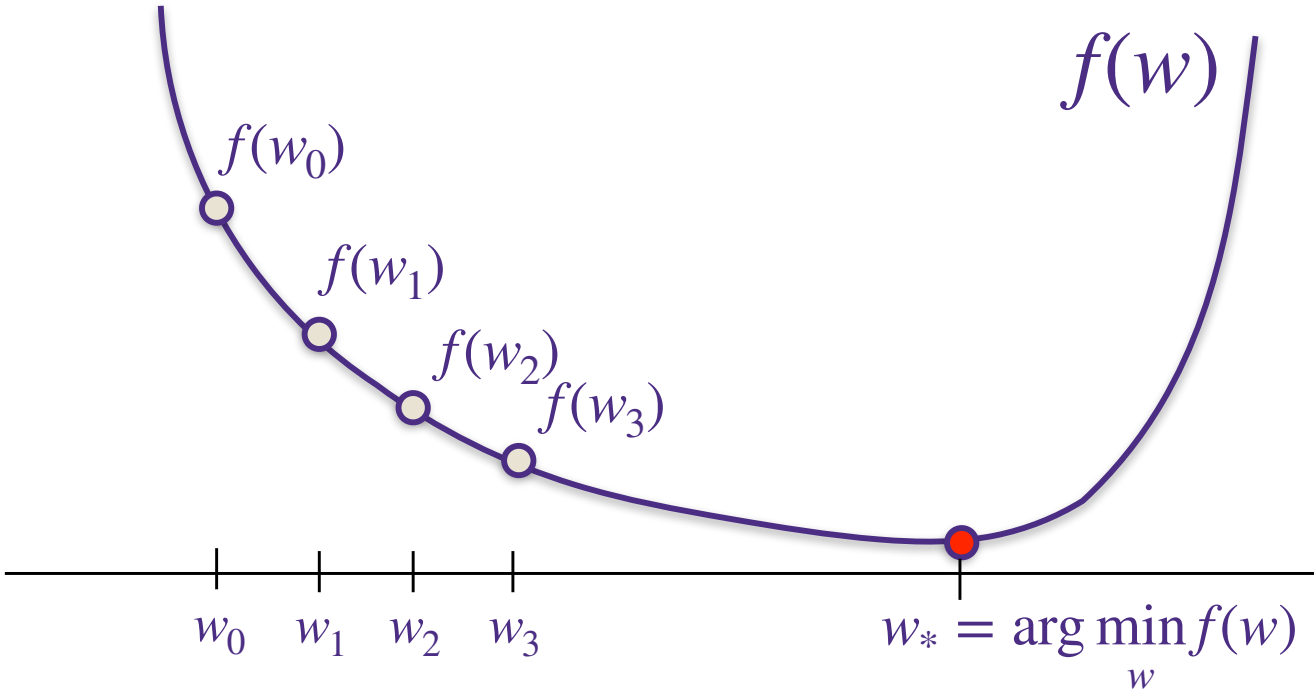
is very close to $f(w)$

Thus, for very small $\eta > 0$,

if $w_1 = w_0 - \eta \frac{df(w)}{dw}\Big|_{w=w_0}$ then

$$f(w_0) - \eta\left(\frac{df(w)}{dw}\Big|_{w=w_0}\right)^2$$

is very close to $f(w_1) < f(w_0)$



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?

Taylor series approximation:
For w very close to w_0 we have

$$f(w_0) + (w - w_0)\frac{df(w)}{dw}\Big|_{w=w_0}$$

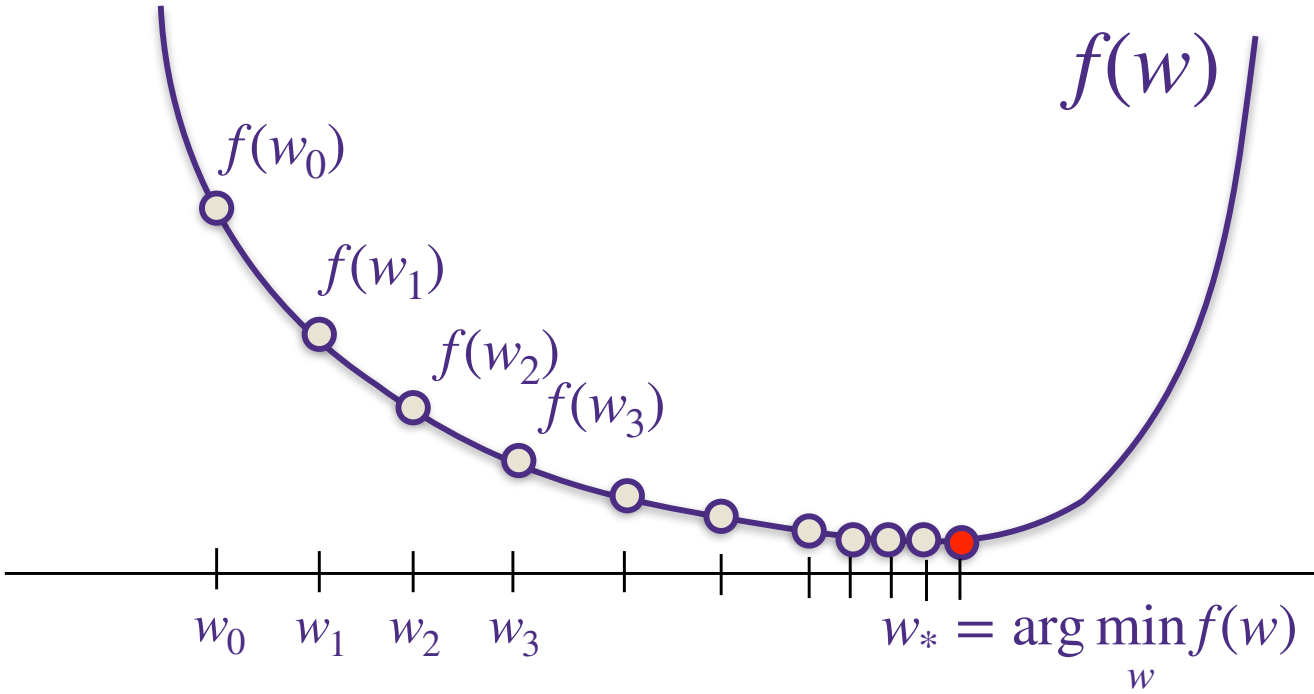
is very close to $f(w)$

Thus, for very small $\eta > 0$,

if $w_1 = w_0 - \eta \frac{df(w)}{dw}\Big|_{w=w_0}$ then

$$f(w_0) - \eta\left(\frac{df(w)}{dw}\Big|_{w=w_0}\right)^2$$

is very close to $f(w_1) < f(w_0)$



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

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

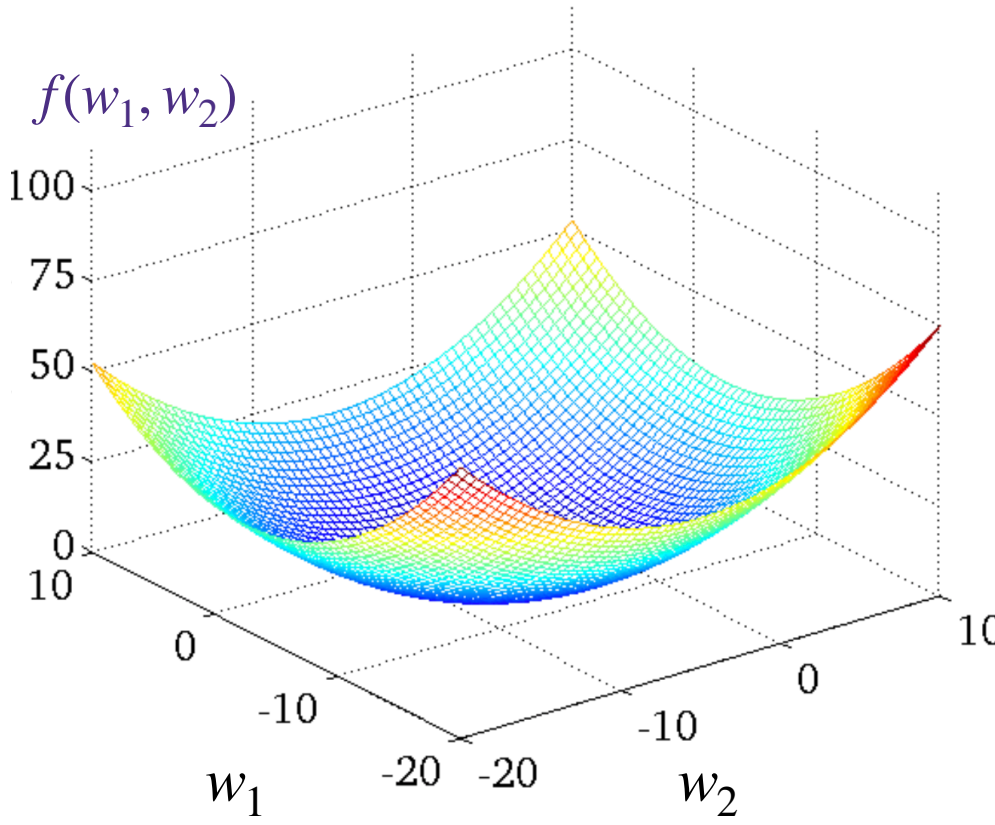
Note that as $k \to \infty$ we have $\frac{df(w)}{dw}\Big|_{w=w_k} \to 0$

Running example: linear regression

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

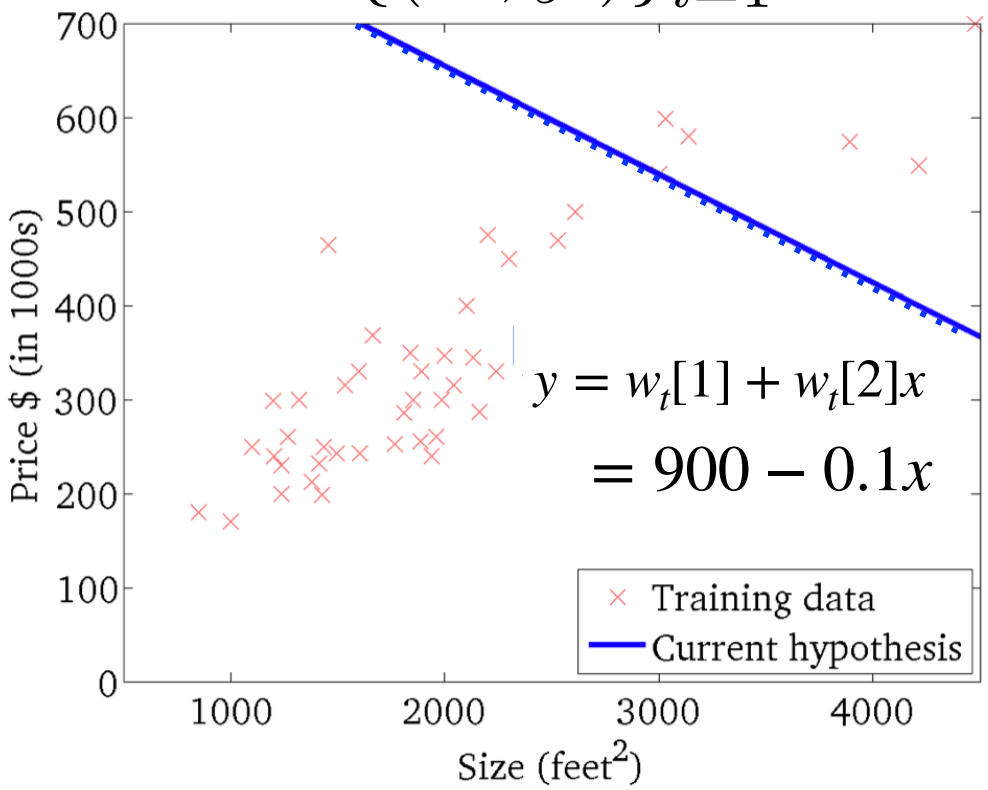
- **Learning model parameters:**

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

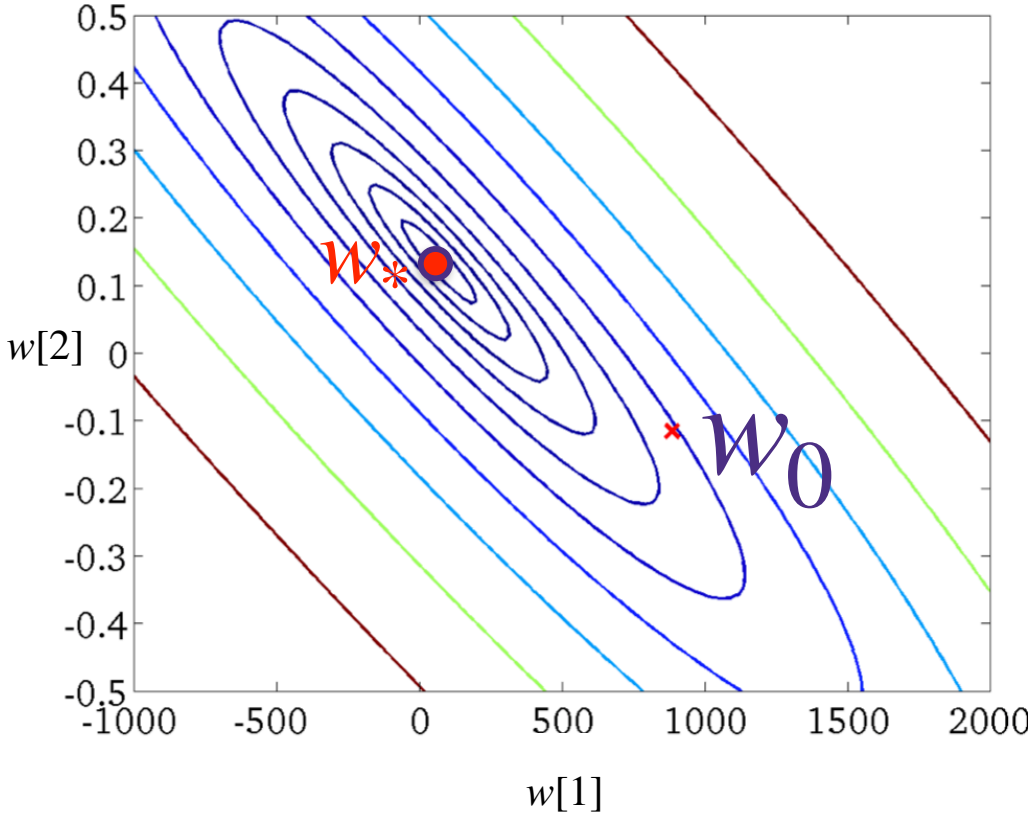


- **Gradient descent:**
  - Initialize: $w_0 = 0$
  - **For** t=0,1,2,…
    - $w_{t+1} \leftarrow w_t - \eta \cdot \nabla_w f(w_t)$

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



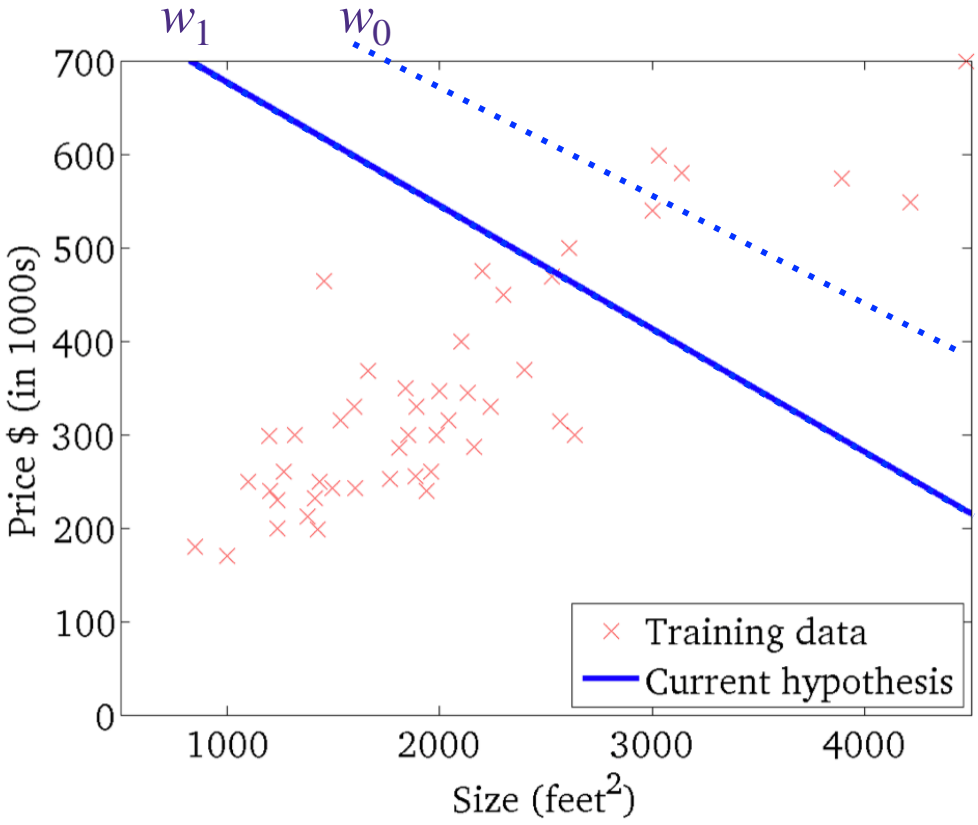
Evolution of the predictor



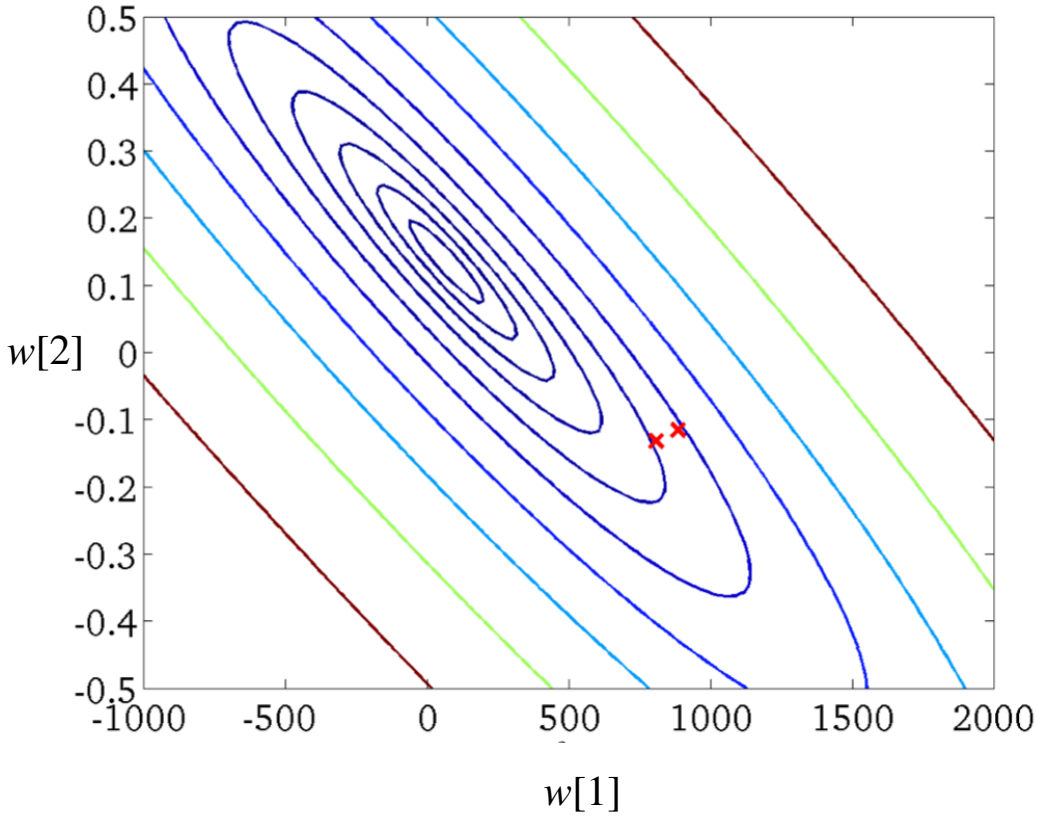
GD dynamics in the Parameter space

- Which direction will the GD move?

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

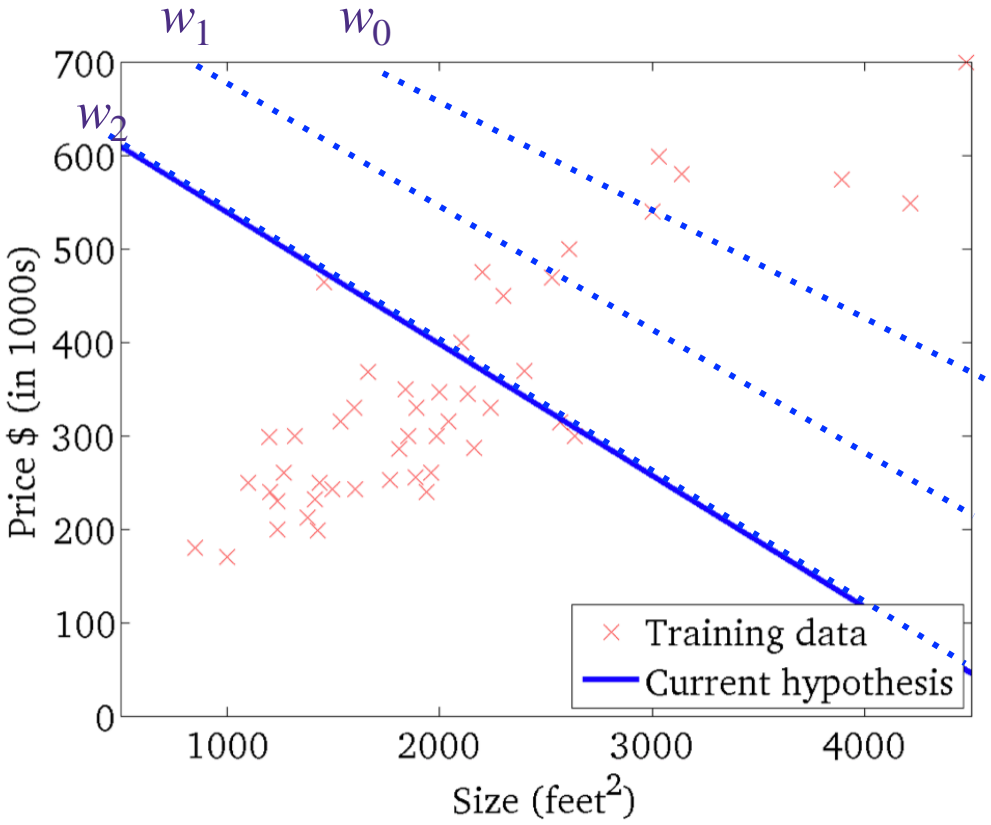


Evolution of the predictor

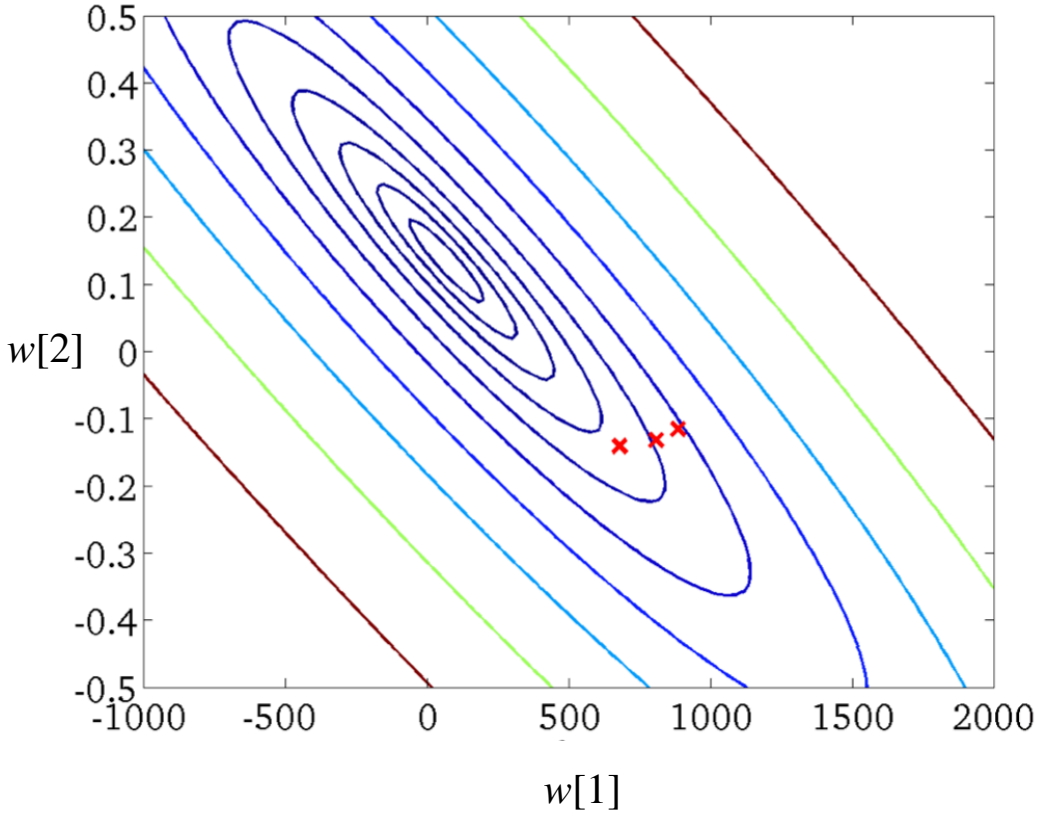


GD dynamics in the Parameter space

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

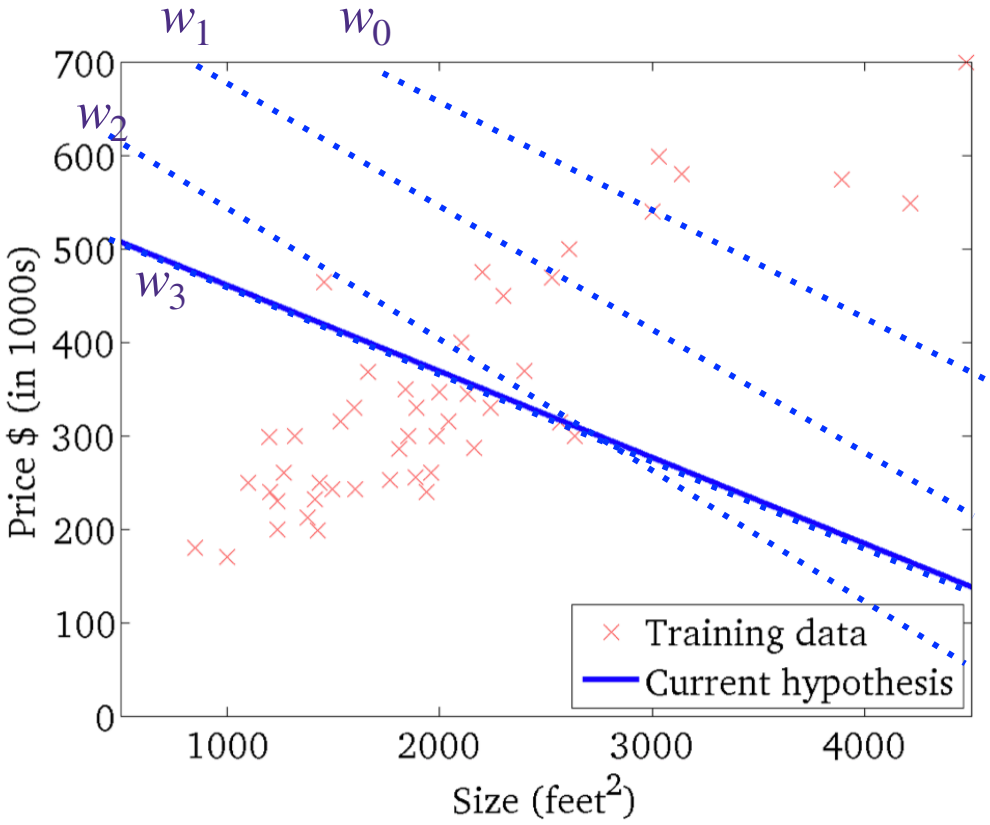


Evolution of the predictor

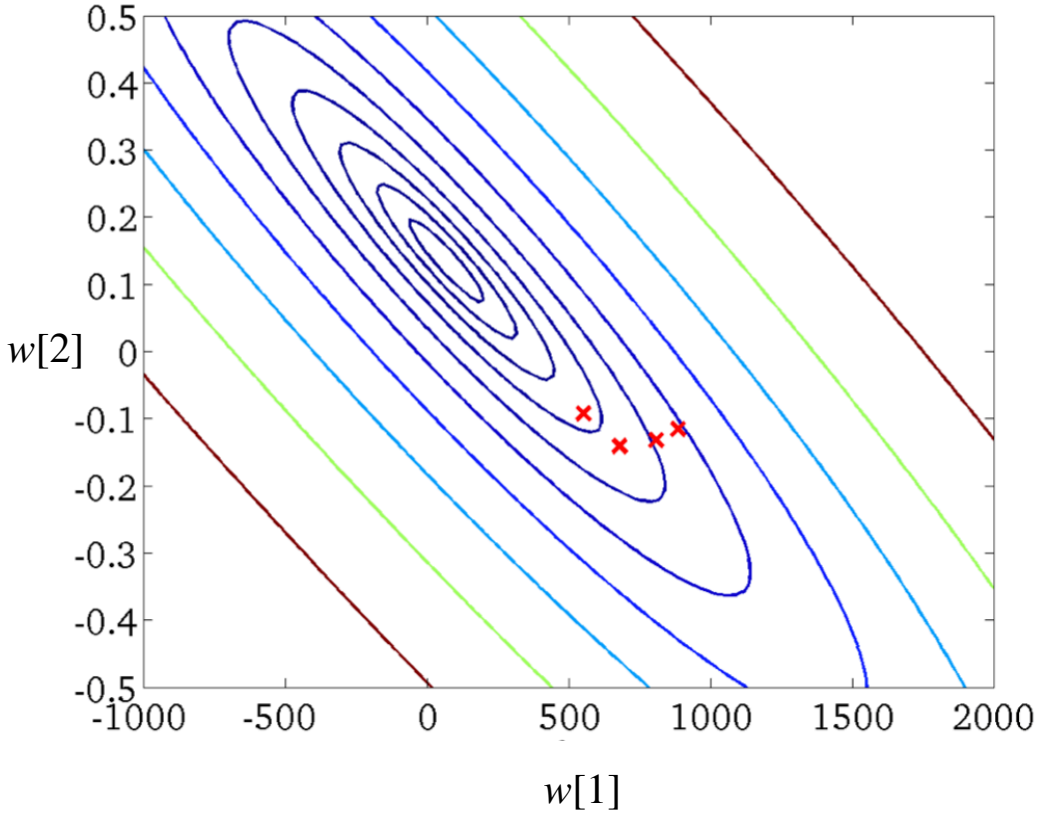


GD dynamics in the Parameter space

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

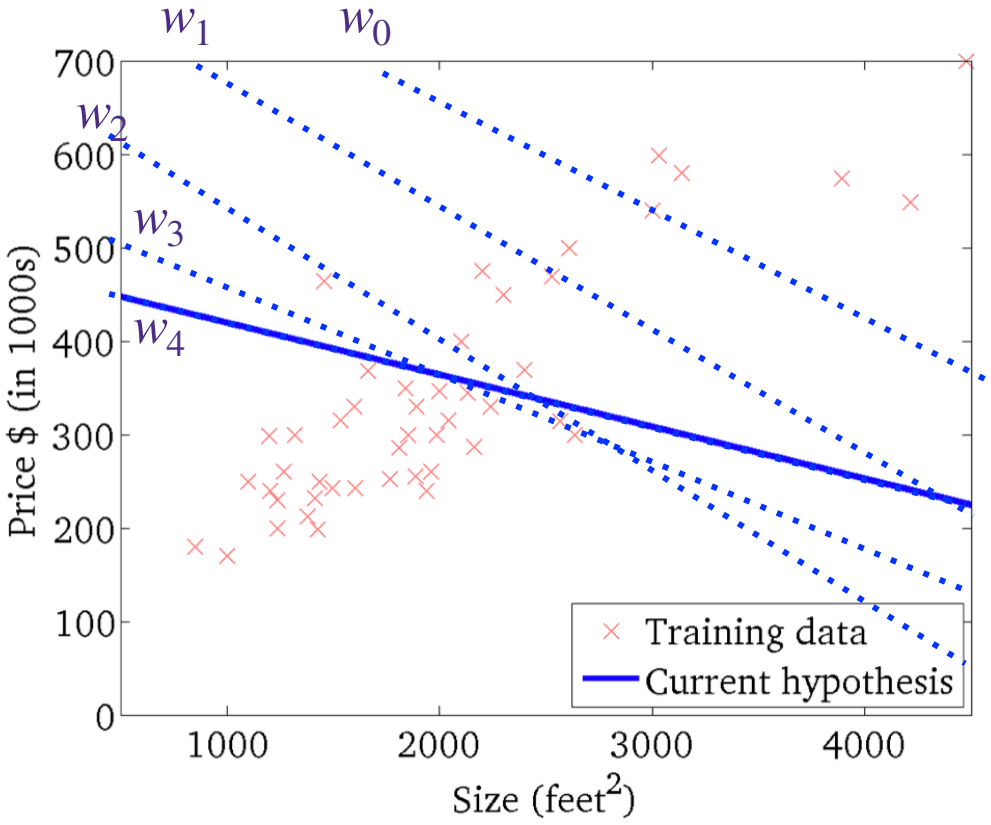


Evolution of the predictor

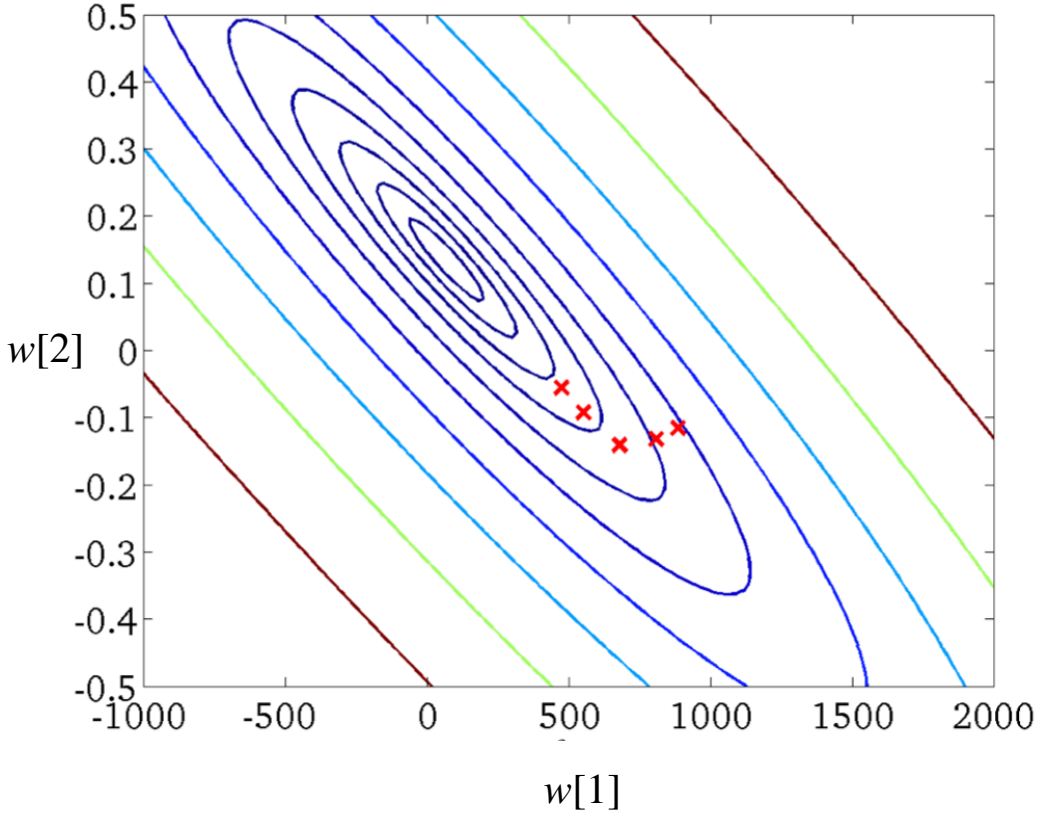


GD dynamics in the Parameter space

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

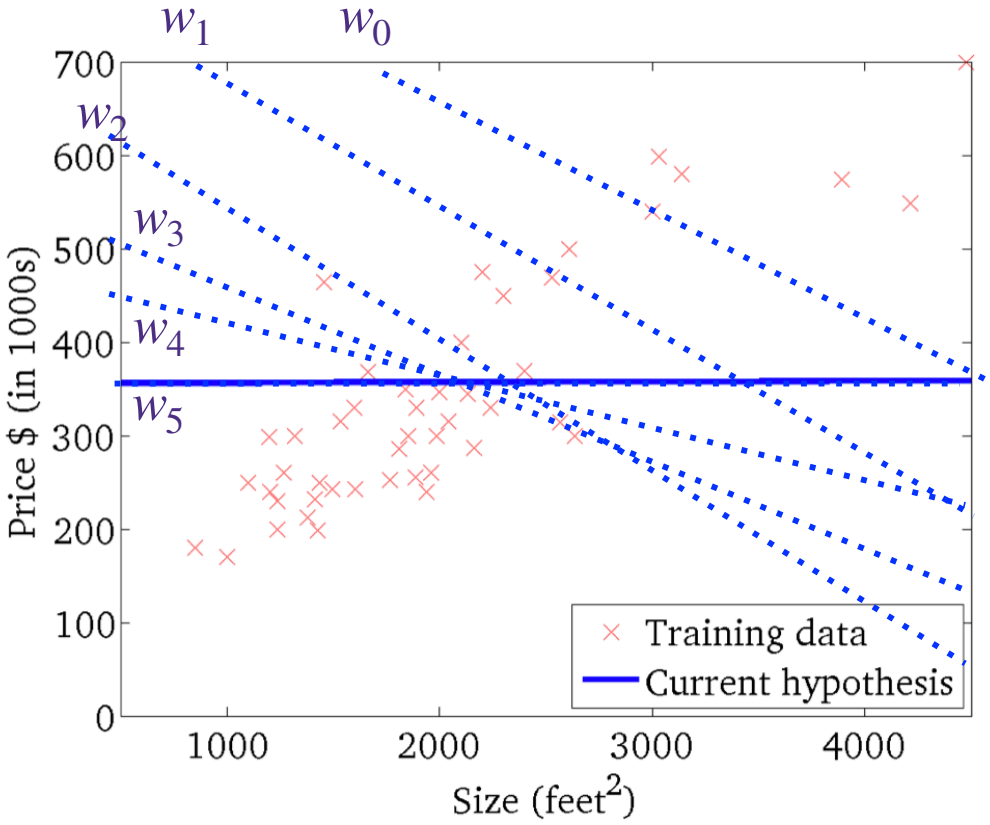


Evolution of the predictor

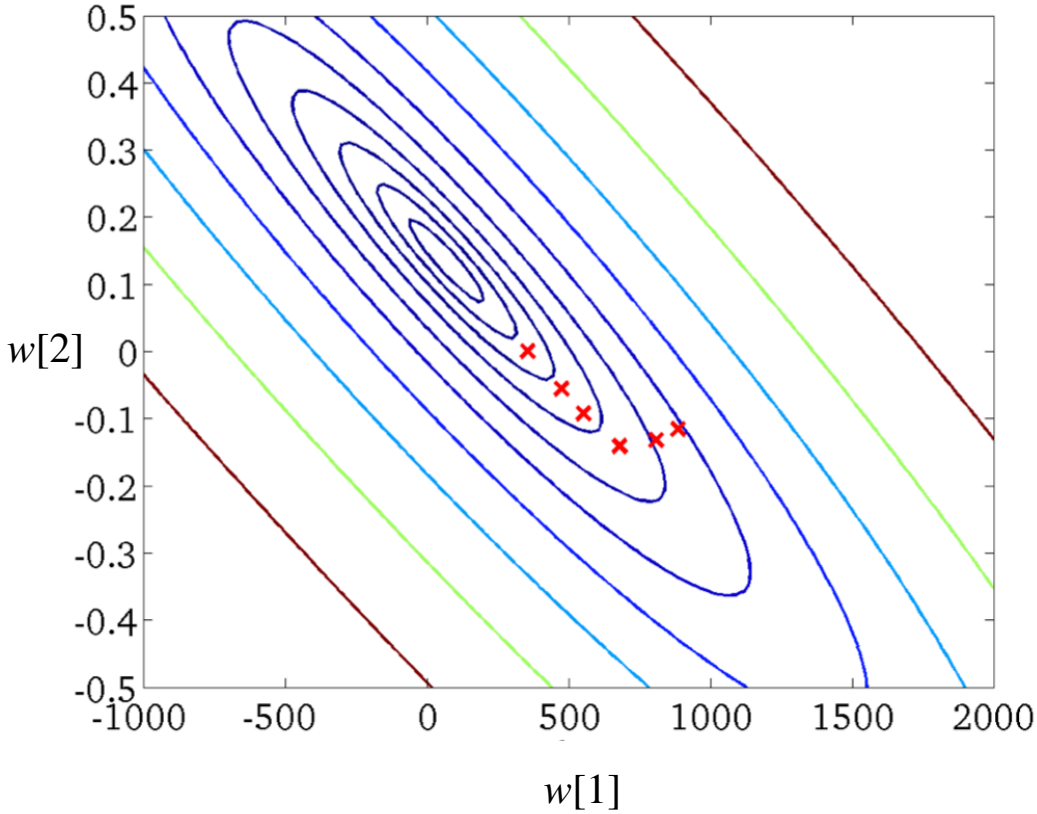


GD dynamics in the Parameter space

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

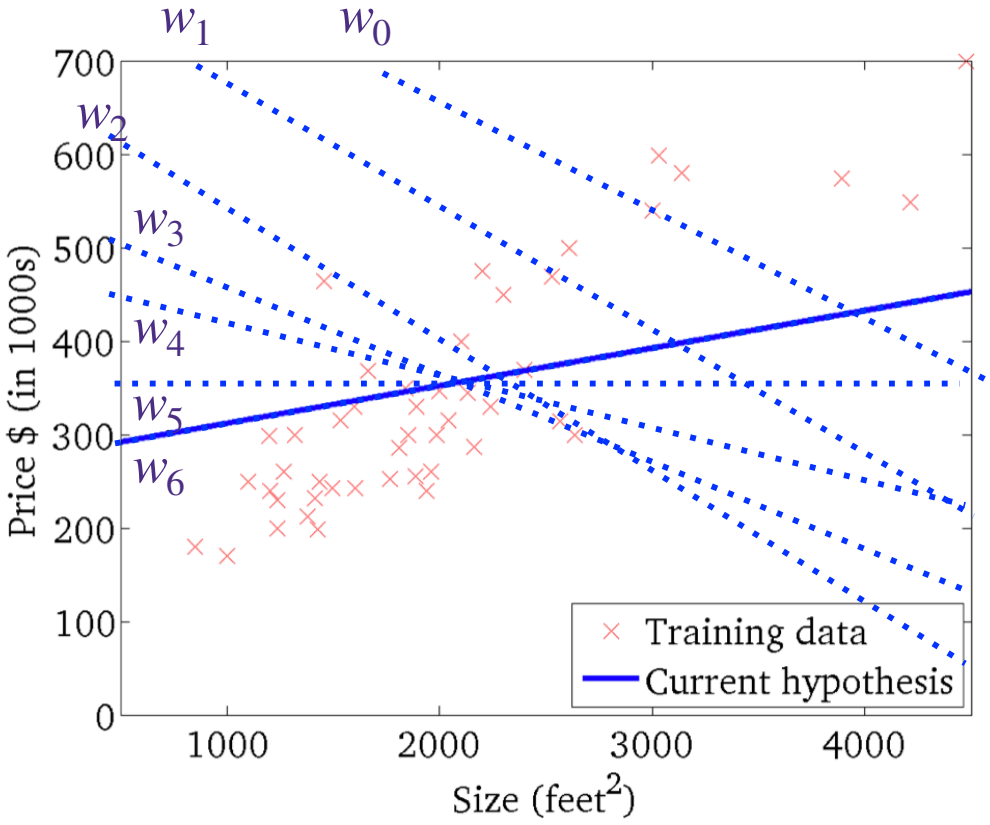


Evolution of the predictor

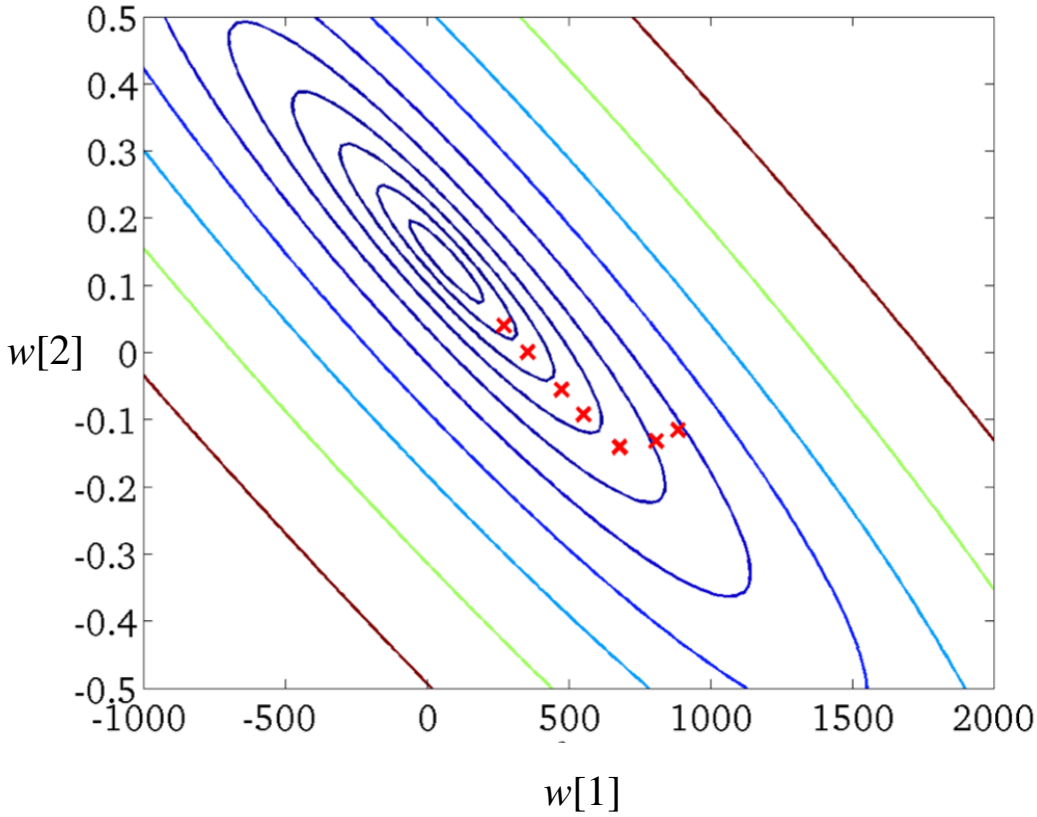


GD dynamics in the Parameter space

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

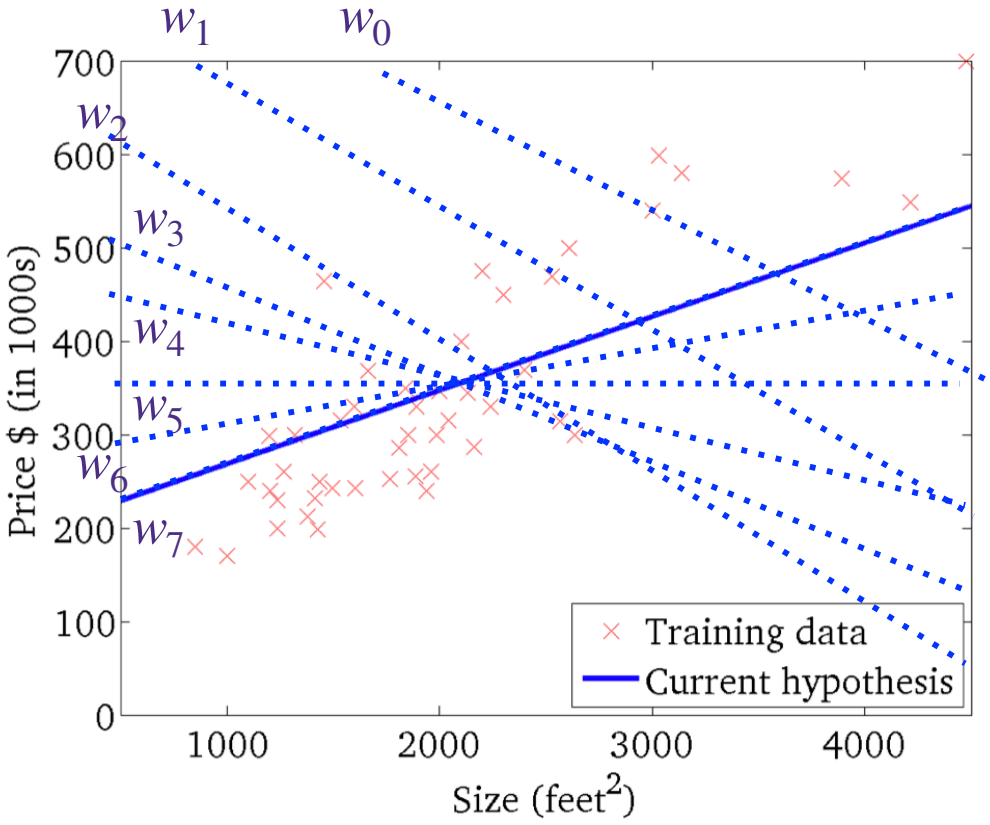


Evolution of the predictor

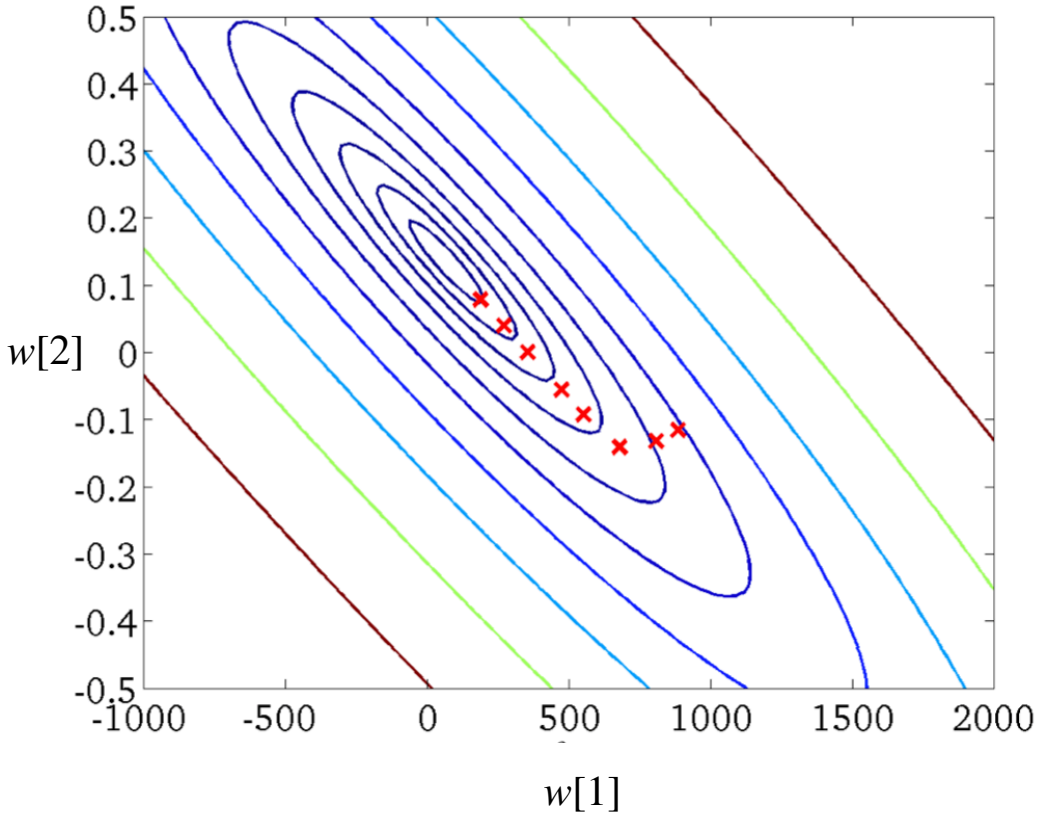


GD dynamics in the Parameter space

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

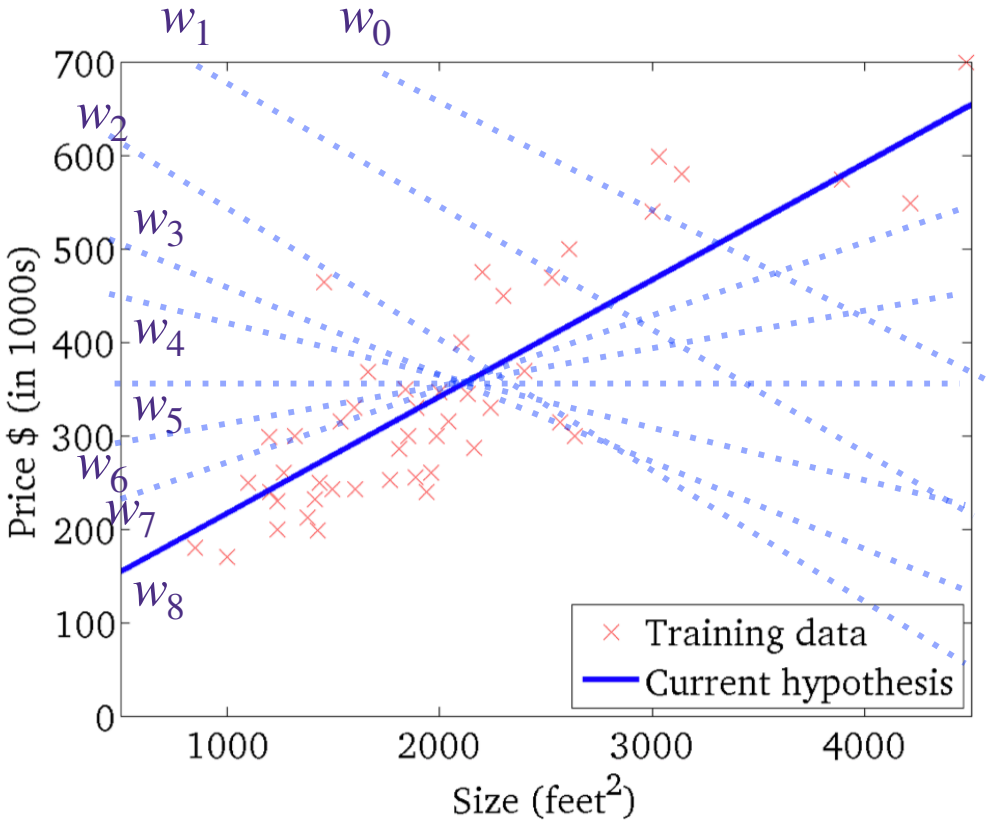


Evolution of the predictor

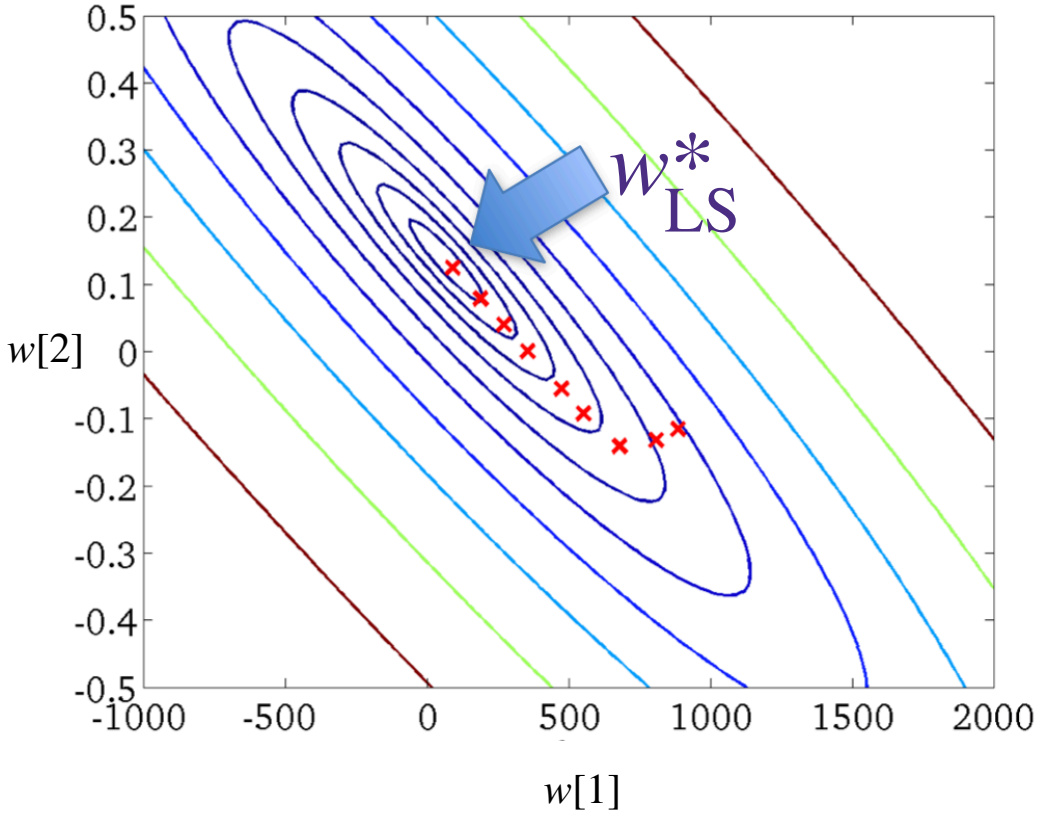


GD dynamics in the Parameter space

- $w_0 = (900, -0.1)$
- **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 for linear regression

- In this example of linear regression, we can derive exactly the gradient descent trajectory
- Initialize: $w_0 = 0$
- **For** t=0,1,2,…
  - $w_{t+1} \leftarrow w_t - \eta \cdot \nabla_w f(w_t)$

$$\nabla f(w_t) = -2\mathbf{X}^T(\mathbf{y} - \mathbf{X}w_t)$$

For linear regression, we have

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

Gradient descent for linear regression

- In this example of linear regression, we can derive exactly the gradient descent trajectory
- Initialize: $w_0 = 0$
- **For** t=0,1,2,…
  - $w_{t+1} \leftarrow w_t - \eta \cdot \nabla_w f(w_t)$

For linear regression, we have

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

$$\nabla f(w_t) = -2\mathbf{X}^T(\mathbf{y} - \mathbf{X}w_t)$$

$$w_{t+1} = w_t + \eta 2\mathbf{X}^T(\mathbf{y} - \mathbf{X}w_t) = (\mathbf{I} - 2\eta\mathbf{X}^T\mathbf{X})w_t + 2\eta\mathbf{X}^T\mathbf{y}$$

Gradient descent for linear regression

- In this example of linear regression, we can derive exactly the gradient descent trajectory
- Initialize: $w_0 = 0$
- **For** t=0,1,2,…
  - $w_{t+1} \leftarrow w_t - \eta \cdot \nabla_w f(w_t)$

For linear regression, we have

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

$$\nabla f(w_t) = -2\mathbf{X}^T(\mathbf{y} - \mathbf{X}w_t)$$

$$w_{t+1} = w_t + \eta 2\mathbf{X}^T(\mathbf{y} - \mathbf{X}w_t) = (\mathbf{I} - 2\eta\mathbf{X}^T\mathbf{X})w_t + 2\eta\mathbf{X}^T\mathbf{y}$$

Let the least-squares solution be $w^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$

$$\begin{aligned} w_{t+1} - w^* &= (\mathbf{I} - 2\eta\mathbf{X}^T\mathbf{X})w_t + 2\eta\mathbf{X}^T\mathbf{y} - w^* \\ &= (\mathbf{I} - 2\eta\mathbf{X}^T\mathbf{X})(w_t - w^*) + 2\eta\mathbf{X}^T\mathbf{y} - 2\eta\mathbf{X}^T\mathbf{X}w^* \\ &= (\mathbf{I} - 2\eta\mathbf{X}^T\mathbf{X})(w_t - w^*) \end{aligned}$$

Gradient descent for linear regression

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

For linear regression, we have

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

$$\nabla f(w_t) = -2\mathbf{X}^T(\mathbf{y} - \mathbf{X}w_t)$$

$$w_{t+1} = w_t + \eta 2\mathbf{X}^T(\mathbf{y} - \mathbf{X}w_t) = (\mathbf{I} - 2\eta\mathbf{X}^T\mathbf{X})w_t + 2\eta\mathbf{X}^T\mathbf{y}$$

Let the least-squares solution be $w^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$

$$\begin{aligned} w_{t+1} - w^* &= (\mathbf{I} - 2\eta\mathbf{X}^T\mathbf{X})w_t + 2\eta\mathbf{X}^T\mathbf{y} - w^* \\ &= (\mathbf{I} - 2\eta\mathbf{X}^T\mathbf{X})(w_t - w^*) + 2\eta\mathbf{X}^T\mathbf{y} - 2\eta\mathbf{X}^T\mathbf{X}w^* \\ &= (\mathbf{I} - 2\eta\mathbf{X}^T\mathbf{X})(w_t - w^*) \end{aligned}$$

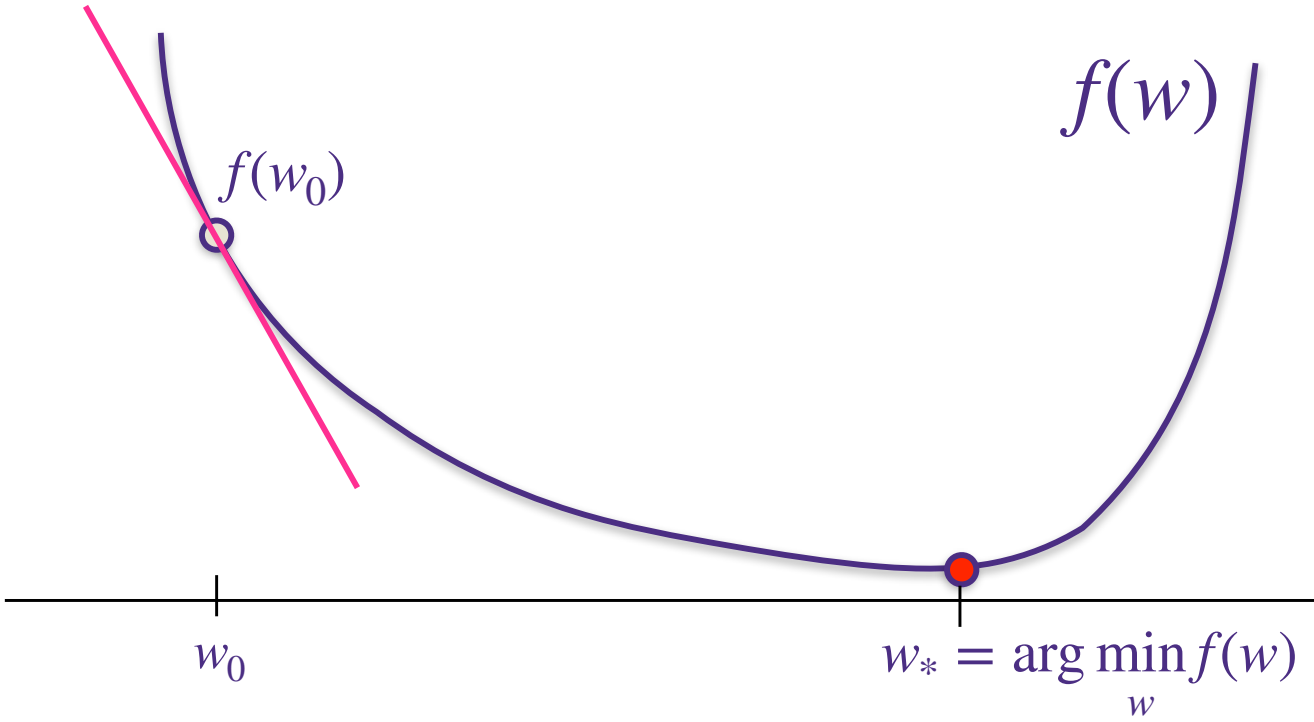
How do you choose step size?

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

Taylor series approximation:
For w very close to w_0 we have

$$f(w_0) + (w - w_0)\frac{df(w)}{dw}\Big|_{w=w_0}$$

is very close to $f(w)$



If η too big, does not converge!

If η too small, converges very, very slowly.

In practice: choose the largest value of η that converges (guess and check)

Gradient descent for Ridge regression

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

For Ridge we have

$$\hat{w}_{\text{Ridge}} = \arg\min_{w \in \mathbb{R}^d} \underbrace{\frac{1}{2}\|\mathbf{y} - \mathbf{X}w\|_2^2 + \frac{\lambda}{2}\|w\|_2^2}_{f(w)}$$

$$\nabla f(w_t) =$$

$$w_{t+1} =$$

Gradient descent for Ridge regression

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

For Ridge we have

$$\hat{w}_{\text{Ridge}} = \arg\min_{w \in \mathbb{R}^d} \underbrace{\frac{1}{2}\|\mathbf{y} - \mathbf{X}w\|_2^2 + \frac{\lambda}{2}\|w\|_2^2}_{f(w)}$$

$$\nabla f(w_t) = -\mathbf{X}^T(\mathbf{y} - \mathbf{X}w_t) + \lambda w_t$$

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

Gradient descent for Lasso regression

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

For Lasso we have

$$\hat{w}_{\text{Lasso}} = \arg\min_{w \in \mathbb{R}^d} \underbrace{\frac{1}{2}\|\mathbf{y} - \mathbf{X}w\|_2^2 + \lambda\|w\|_1}_{f(w)}$$

$$\nabla f(w_t) =$$

$$w_{t+1} =$$

Gradient descent for Lasso regression

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

For Lasso we have

$$\hat{w}_{\text{Lasso}} = \arg\min_{w \in \mathbb{R}^d} \underbrace{\frac{1}{2}\|\mathbf{y} - \mathbf{X}w\|_2^2 + \lambda\|w\|_1}_{f(w)}$$

$$\nabla f(w_t) = -\mathbf{X}^T(\mathbf{y} - \mathbf{X}w_t) + \lambda \text{sign}(w_t)$$

$$w_{t+1} = w_t + \eta\mathbf{X}^T(\mathbf{y} - \mathbf{X}w_t) - \lambda \text{sign}(w_t)$$

Gradient Descent Demo