Sparsity and the Lasso

CSE 446/546

Sewoong Oh & Pang Wei Koh

The story so far

- Regularization
  - Penalizes complex models towards simpler models
- One type of regularization: Ridge regression
  - Penalty on the L_2-norm of the model parameters
  - Shrinks model parameters w towards 0

- Today: A different form of regularization

Sparsity

- Definition: w is k-sparse if it has at most k non-zero entries
- Why sparsity?
  - Efficiency: If w is k-sparse, prediction depends only on k multiplies

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

Sparsity

- Definition: w is k-sparse if it has at most k non-zero entries
- Why sparsity?
  - Efficiency: If w is k-sparse, prediction depends only on k multiplies
  - Interpretability: What are the relevant dimensions to make a prediction?

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	

Sparsity

- Definition: w is k-sparse if it has at most k non-zero entries
- Why sparsity?
  - Efficiency: If w is k-sparse, prediction depends only on k multiplies
  - Interpretability: What are the relevant dimensions to make a prediction?
  - Correctness: If you believe that the “true model” is sparse

Finding the best subset: Exhaustive search

$x \in \mathbb{R}^d$

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

X train $\Rightarrow$ size d

X test $\Rightarrow$ cheating

slow 2^d

Finding the best subset: Greedy search

Forward stepwise: start from simple model, iteratively add features

$T \leftarrow \emptyset$

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

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

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

Backward stepwise: start with full model, iteratively remove features

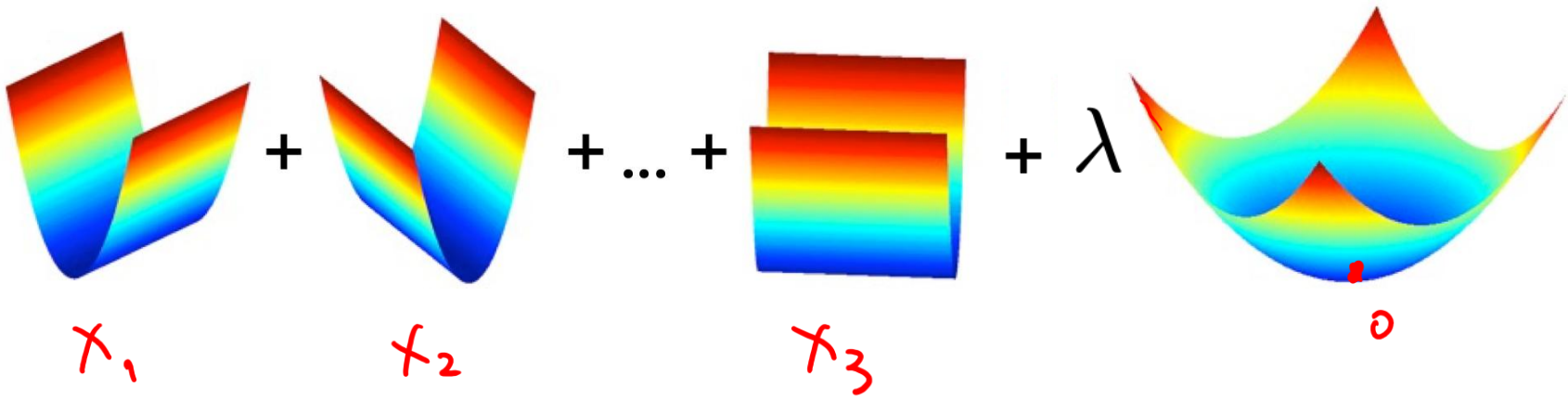
Forward + backward: do forward but remove features no longer important

(and many other variants)

Finding the best subset: Ridge regression

$$\hat{w}_{\text{ridge}} = \underset{w}{\text{argmin}} \sum_i \left(y_i - x_i^\top w\right)^2 + \lambda \|w\|_2^2$$

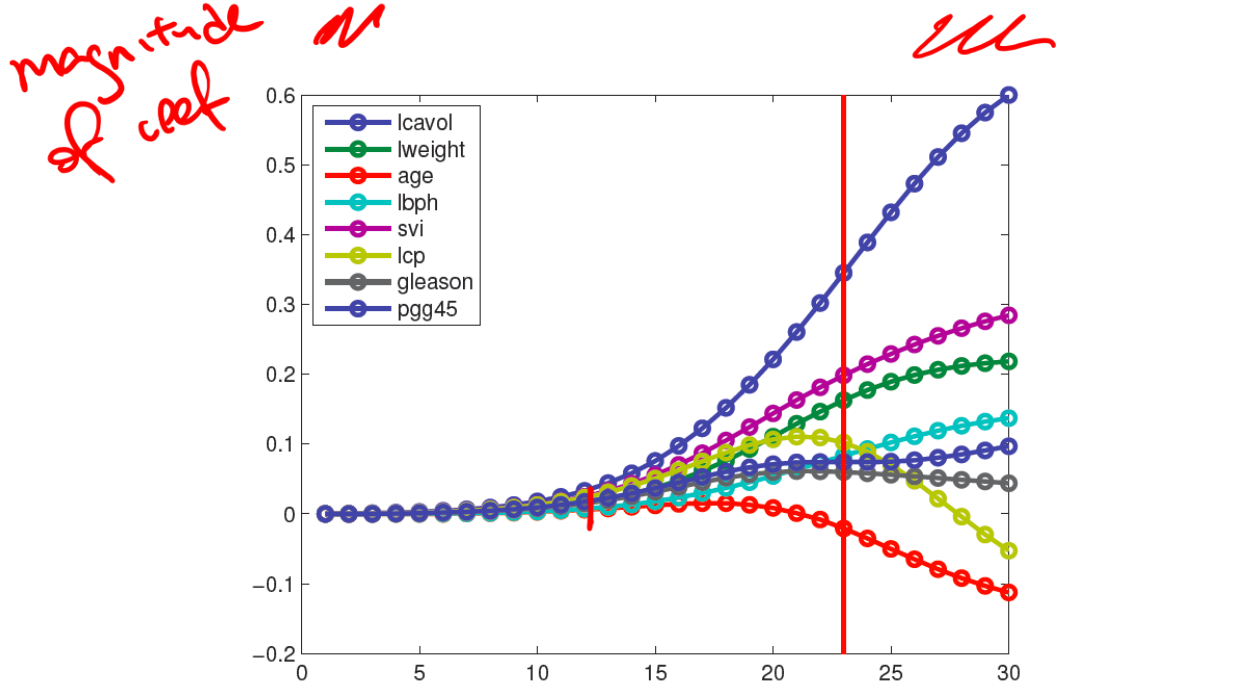
- Ridge regression makes coefficients small



Finding the best subset: Ridge regression

$$\hat{w}_{\text{ridge}} = \underset{w}{\operatorname{argmin}} \sum_i \left(y_i - x_i^\top w\right)^2 + \lambda \|w\|_2^2$$

- Ridge regression makes coefficients small

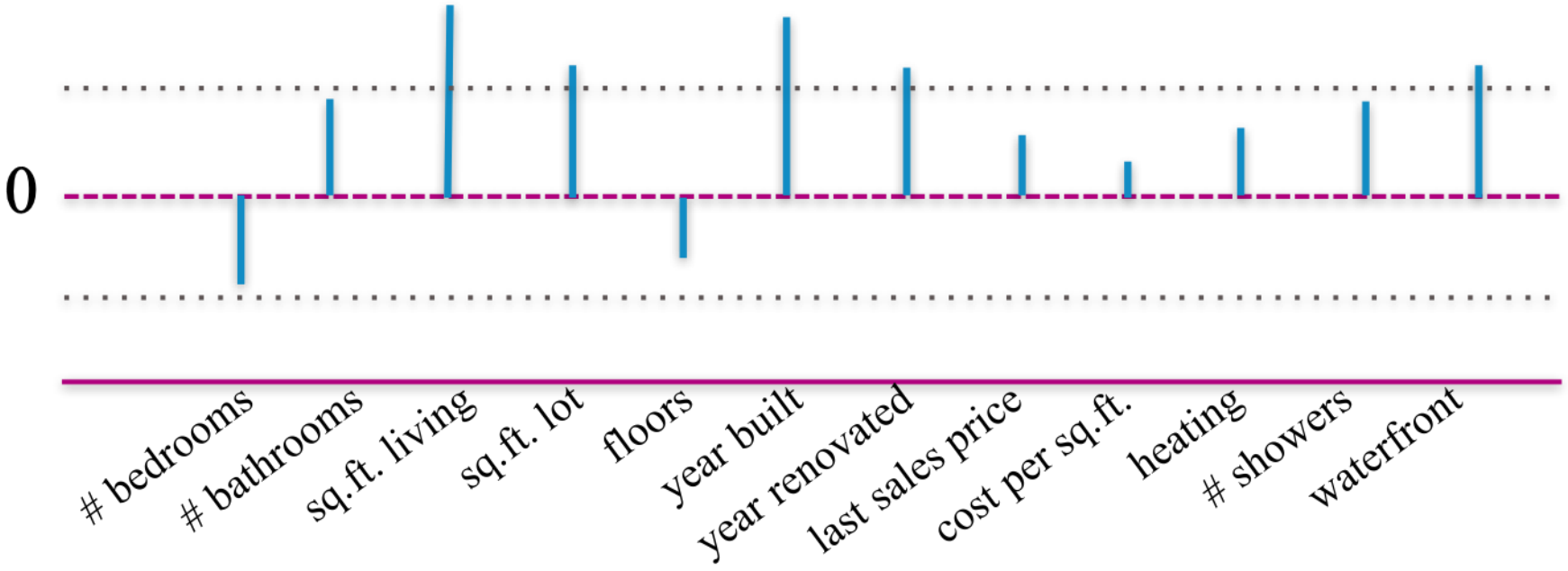


From Kevin Murphy's textbook

Finding the best subset: Ridge regression + threshold

$$\hat{w}_{\text{ridge}} = \underset{w}{\text{argmin}} \sum_i \left(y_i - x_i^\top w\right)^2 + \lambda \|w\|_2^2$$

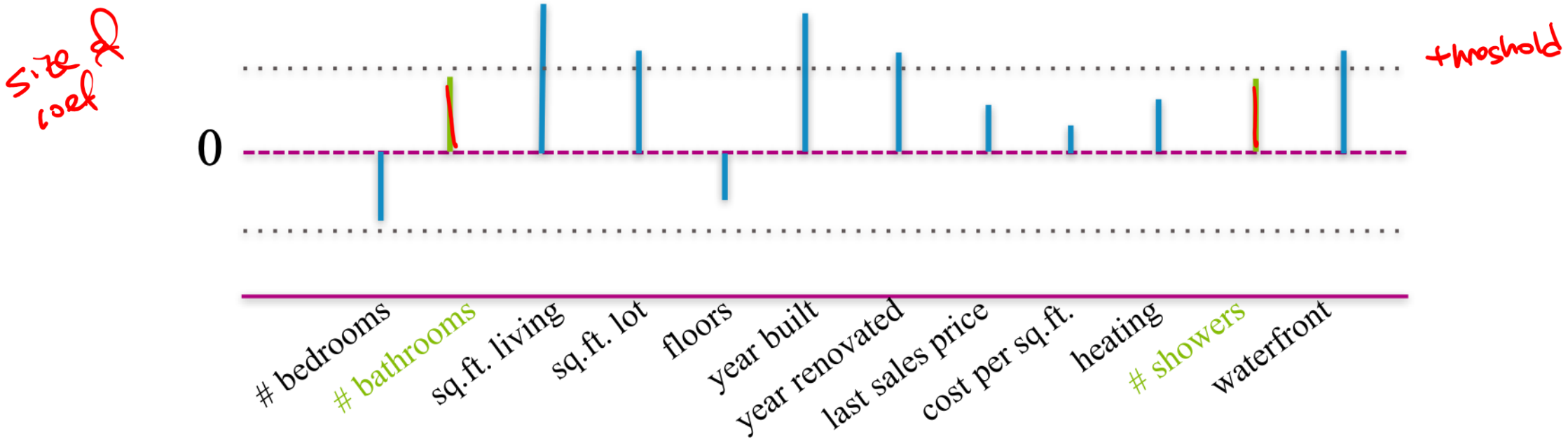
- Why don't we just set small ridge coefficients to 0?



Finding the best subset: Ridge regression + threshold

$$\hat{w}_{\text{ridge}} = \underset{w}{\text{argmin}} \sum_i \left(y_i - x_i^\top w\right)^2 + \lambda \|w\|_2^2$$

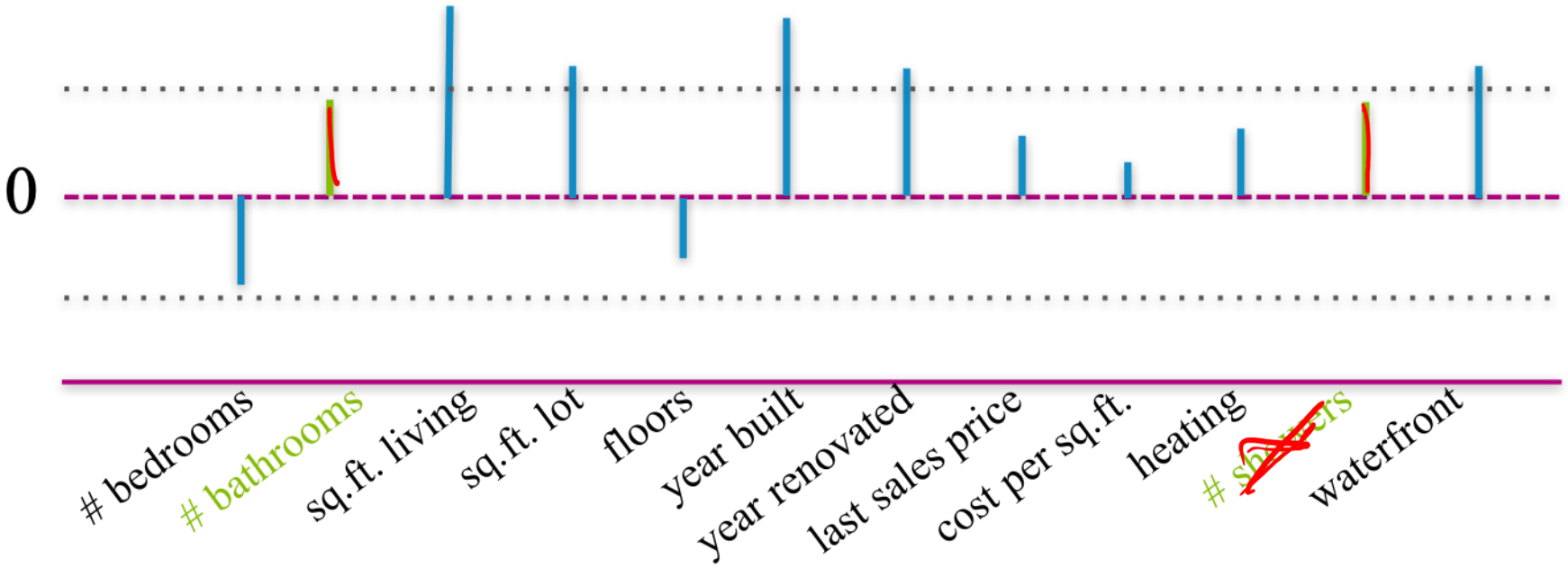
- Consider 2 related features (bathrooms, showers)



Finding the best subset: Ridge regression + threshold

$$\hat{w}_{\text{ridge}} = \underset{w}{\text{argmin}} \sum_i \left(y_i - x_i^\top w\right)^2 + \lambda \|w\|_2^2$$

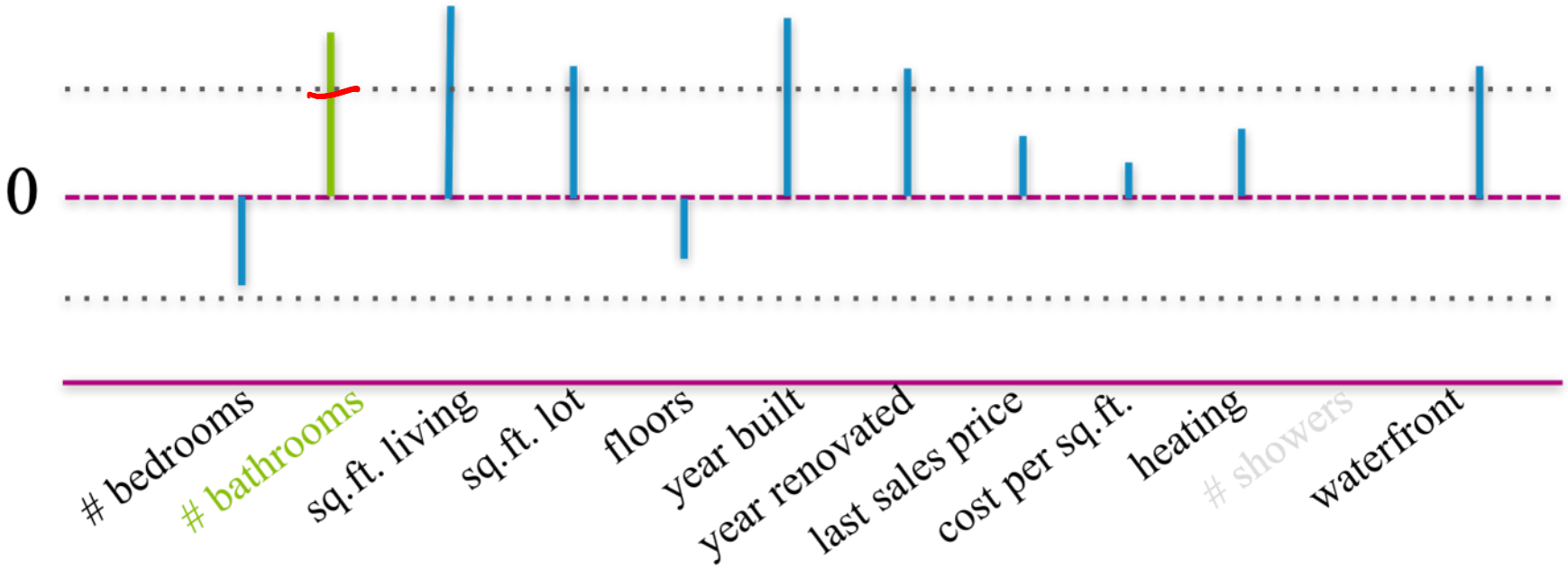
- What if we didn't include showers?



Finding the best subset: Ridge regression + threshold

$$\hat{w}_{\text{ridge}} = \underset{w}{\text{argmin}} \sum_i \left(y_i - x_i^\top w\right)^2 + \lambda \|w\|_2^2$$

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



Back to the drawing board

- We want to find parameters w that are sparse
- Ridge regression fails on correlated features
- Exhaustive search is too expensive
- Greedy search doesn't typically work well

Goal: Find a regularizer that gives us sparse solutions

$$\hat{w} = \underset{w}{\operatorname{argmin}} \sum_i \left(y_i - x_i^\top w\right)^2 + \lambda r(w)$$

$\|w\|_2^2$

ℓ_p norms

$$\hat{w}_{\text{ridge}} = \underset{w}{\text{argmin}} \sum_i (y_i - x_i^\top w)^2 + \lambda \|w\|_2^2$$

What if we used other norms? $\|w\|_p = \left(\sum_{k=1}^d |w_k|^p\right)^{1/p}$ for $0 < p < \infty$

$p=2$

$\|w\|_2 = \left(\sum_{k=1}^d |w_k|^2\right)^{1/2}$

$\|w\|_2^2 = \sum_{k=1}^d w_k^2$

$p=1$

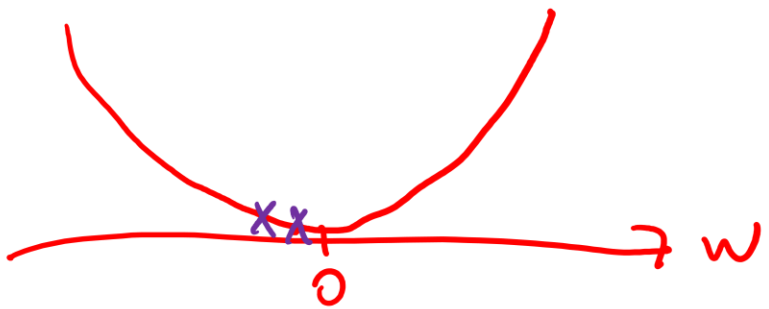
$\|w\|_1 = \sum_{k=1}^d |w_k|$

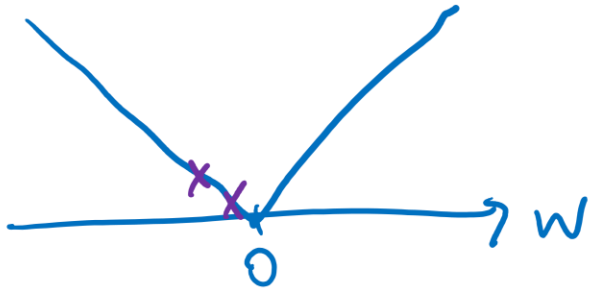
$p=\infty$

$\|w\|_\infty = \max_k (w_k)$

$p=0$

$\|w\|_0 = \text{card}(w)$





Lasso vs. ridge regression

$$\hat{w}_{\text{ridge}} = \underset{w}{\text{argmin}} \sum_i \left(y_i - x_i^\top w\right)^2 + \lambda \|w\|_2^2$$

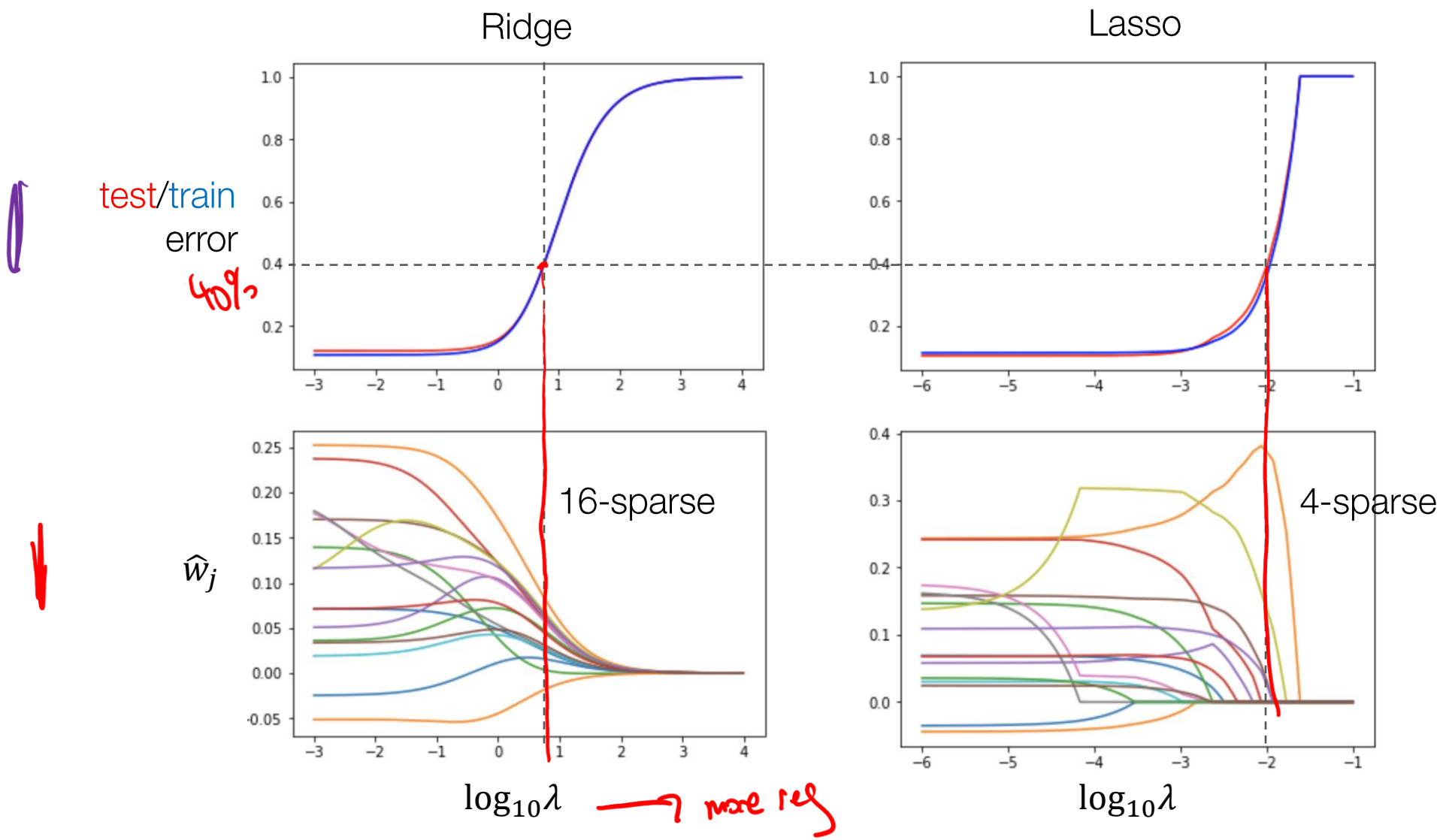
2

$$\hat{w}_{\text{lasso}} = \underset{w}{\text{argmin}} \sum_i \left(y_i - x_i^\top w\right)^2 + \lambda \|w\|_1$$

1

That's it! Somewhat magically, this generates sparse solutions.

Example: Housing price with 16 features



Algorithm: Lasso regression

1. **Model selection:** Choose λ based on cross-validation error, desired sparsity
2. **Feature selection:** Keep only features with non-zero (or not-too-small) weights in w at optimal λ
3. **(Optionally) Retrain** with the sparse model and $\lambda = 0$

Why is retraining helpful?

Example: Piecewise-linear fit

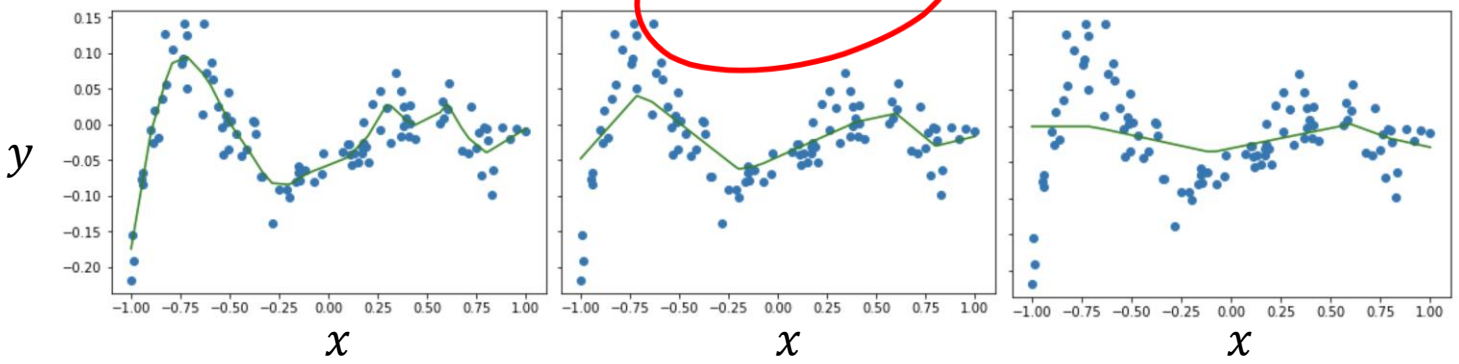
piecewise features

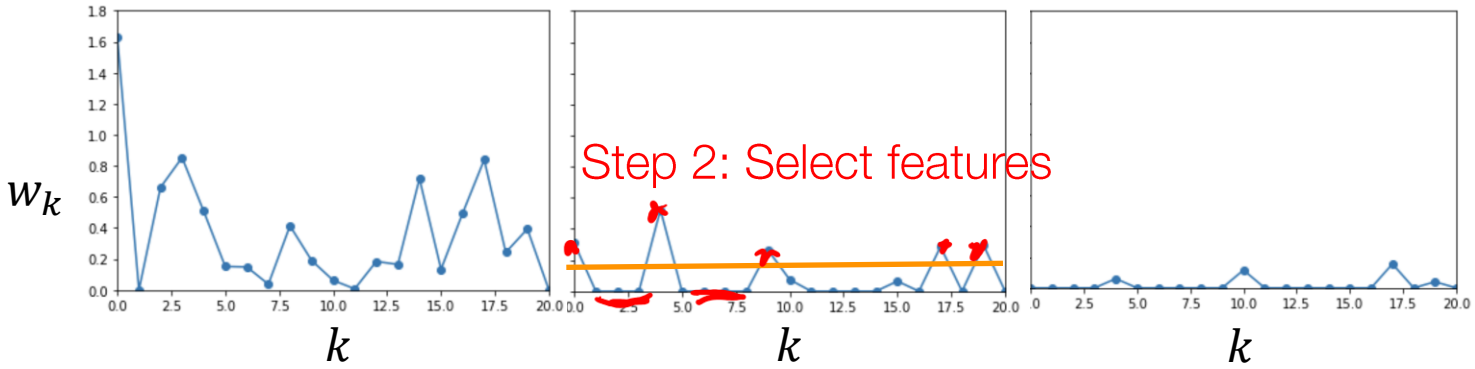
$$h_0(x) = 1$$
$$h_i(x) = [x + 1.1 - 0.1i]^+$$

$$\text{minimize}_w \sum_i \left(y_i - h(x_i)^T w\right)^2 + \lambda \|w\|_1$$

Step 1: Find optimal λ

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

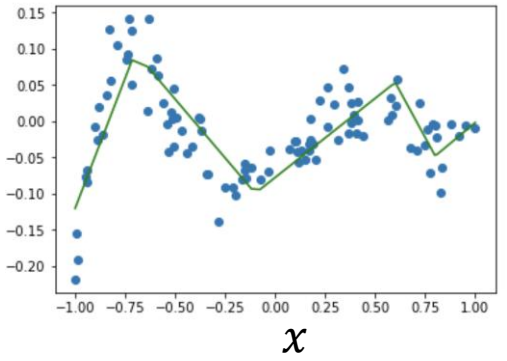


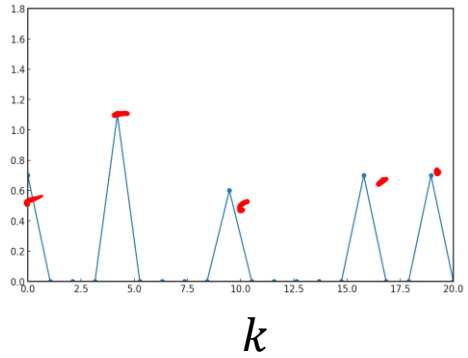


Step 2: Select features

Step 3: Retrain with selected features

$\lambda = 0$





Penalties vs. constraints

$$\hat{w}_{\mathrm{p}} = \underset{w}{\operatorname{argmin}} \sum_i \left(y_i - x_i^{\mathsf{T}} w\right)^2 + \lambda r(w)$$

$$\hat{w}_{\mathrm{c}} = \underset{w}{\operatorname{argmin}} \sum_i \left(y_i - x_i^{\mathsf{T}} w\right)^2 \quad \text{subject to} \quad r(w) \le \mu$$

These problems are equivalent in the following (informal) sense:

For any $\lambda \ge 0$, there exists μ such that $\hat{w}_{\mathrm{p}} = \hat{w}_{\mathrm{c}}$, and vice versa.

$$\hat{w}_{\mathbf{p}} = \underset{w}{\operatorname{argmin}} \sum_i \left(y_i - x_i^\top w\right)^2 + \lambda r(w)$$

$$\hat{w}_{\mathbf{c}} = \underset{w}{\operatorname{argmin}} \sum_i \left(y_i - x_i^\top w\right)^2 \quad \text{subject to} \quad r(w) \leq \mu$$

Note that $\hat{w}_{\mathbf{p}}$ is a function of λ and $\hat{w}_{\mathbf{c}}$ is a function of μ. We could have written $\hat{w}_p(\lambda)$ and $\hat{w}_{\mathbf{c}}(\mu)$ to make this explicit, but we'll leave it implicit to keep the notation cleaner.
Assume for convenience that the solutions to the above optimization problems are unique.
Then for any $\lambda \geq 0$, there exists μ such that $\hat{w}_{\mathbf{p}} = \hat{w}_{\mathbf{c}}$, and vice versa.

Proof:

Given λ, solve for $\hat{w}_{\mathbf{p}}$, then set $\mu = r\left(\hat{w}_{\mathbf{p}}\right)$. Denote $f(w) = \sum_i \left(y_i - x_i^\top w\right)^2$.
Assume for contradiction that $\hat{w}_{\mathbf{p}} \neq \hat{w}_{\mathbf{c}}$.
We know that $\hat{w}_{\mathbf{p}}$ is feasible (i.e., $r\left(\hat{w}_{\mathbf{p}}\right) \leq \mu$) because μ was set to exactly make $r\left(\hat{w}_p\right) = \mu$.
Since $\hat{w}_{\mathbf{c}}$ is the solution of the second optimization problem, we also know that:
(1) $r(\hat{w}_{\mathbf{c}}) \leq \mu = r\left(\hat{w}_{\mathbf{p}}\right)$ (because $\hat{w}_{\mathbf{c}}$ is feasible by definition)
(2) $\mathrm{f}(\hat{w}_{\mathbf{c}}) < \mathrm{f}\left(\hat{w}_{\mathbf{p}}\right)$ (because $\hat{w}_{\mathbf{c}}$ minimizes $f(w)$ out of all feasible points, and $\hat{w}_{\mathbf{p}}$ is feasible).
Putting these together, we get that $\mathrm{f}(\hat{w}_{\mathbf{c}}) + \lambda r(\hat{w}_{\mathbf{c}}) < \mathrm{f}\left(\hat{w}_{\mathbf{p}}\right) + \lambda r\left(\hat{w}_{\mathbf{p}}\right)$.
But this is a contradiction because by definition, $\hat{w}_{\mathbf{p}}$ is supposed to minimize $\mathrm{f}(w) + \lambda r(w)$.
Thus, $\hat{w}_{\mathbf{p}} = \hat{w}_{\mathbf{c}}$.

Feasible sets

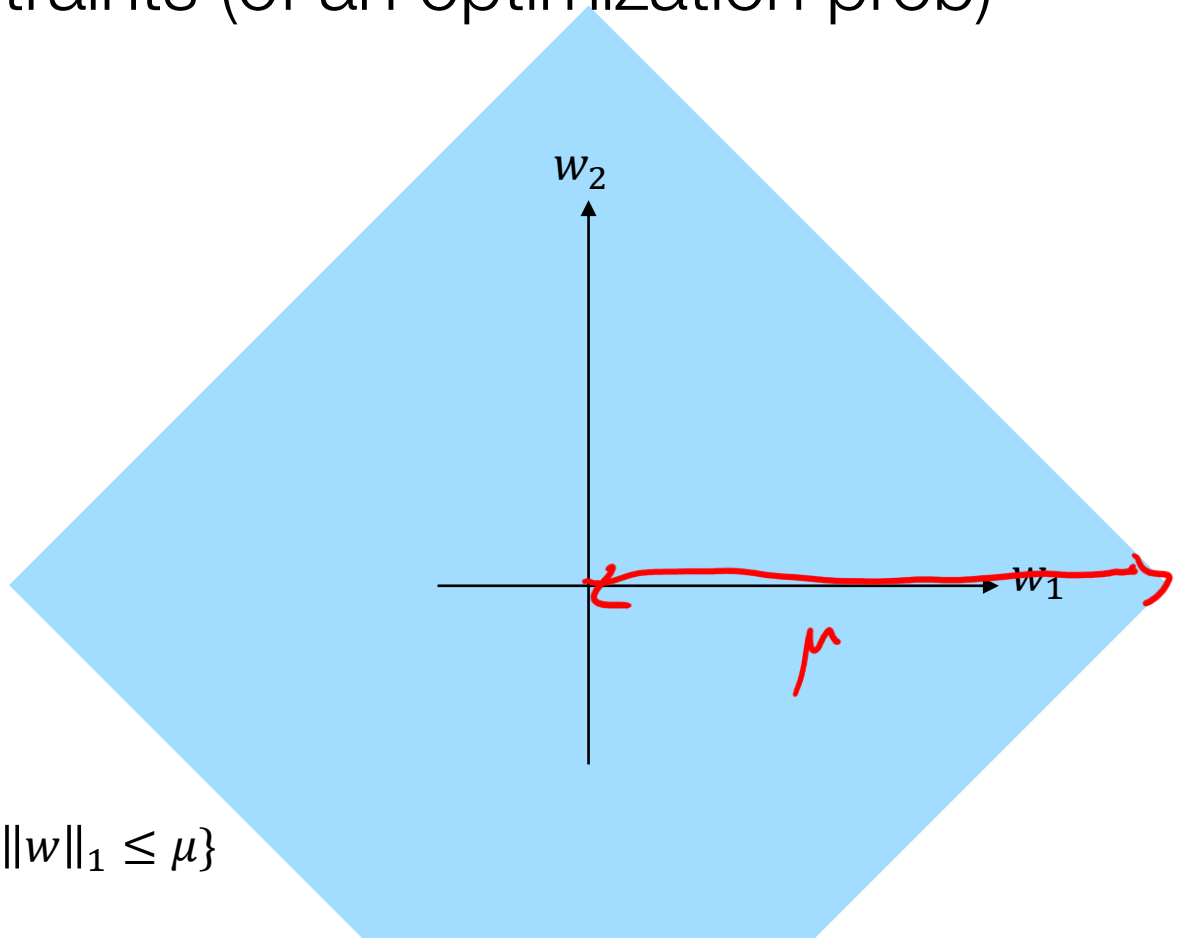
$$\hat{w}_{\text{lasso}} = \underset{w}{\text{argmin}} \sum_i \left(y_i - x_i^\top w\right)^2 \quad \text{subject to} \quad \|w\|_1 \leq \mu$$

- Set of points that satisfy the constraints (of an optimization prob)

Feasible sets

$$\hat{w}_{\text{lasso}} = \underset{w}{\operatorname{argmin}} \sum_i \left(y_i - x_i^\top w\right)^2 \quad \text{subject to} \quad \|w\|_1 \le \mu$$

- Set of points that satisfy the constraints (of an optimization prob)

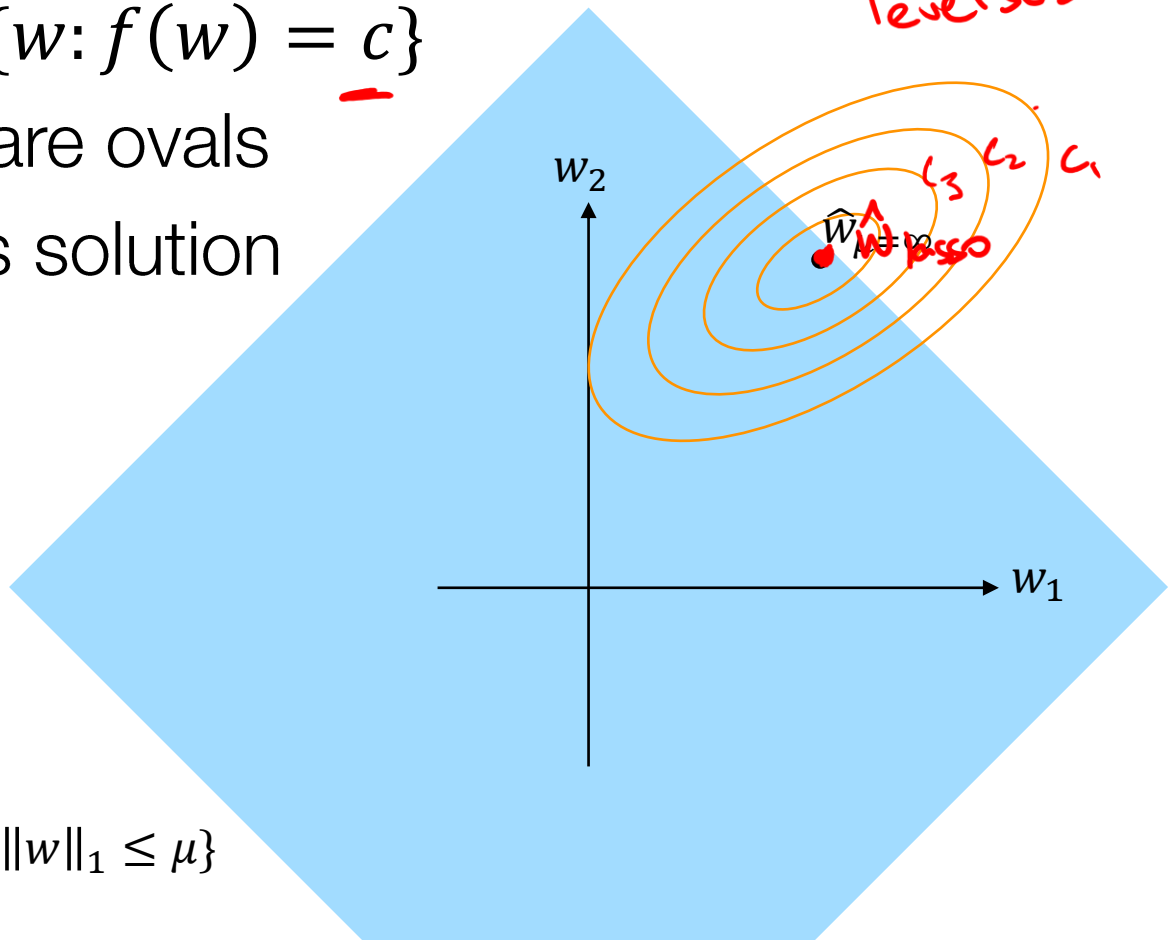


Feasible set $\{w \in \mathbb{R}^2 : \|w\|_1 \le \mu\}$

Level sets

$$\hat{w}_{\text{lasso}} = \underset{w}{\operatorname{argmin}} \sum_i (y_i - x_i^\top w)^2 \quad \text{subject to} \quad \|w\|_1 \leq \mu$$

- Level set $L_c(f)$ of a function f is $\{w: f(w) = c\}$
- Level sets of quadratic functions are ovals
- Center of oval is the least squares solution

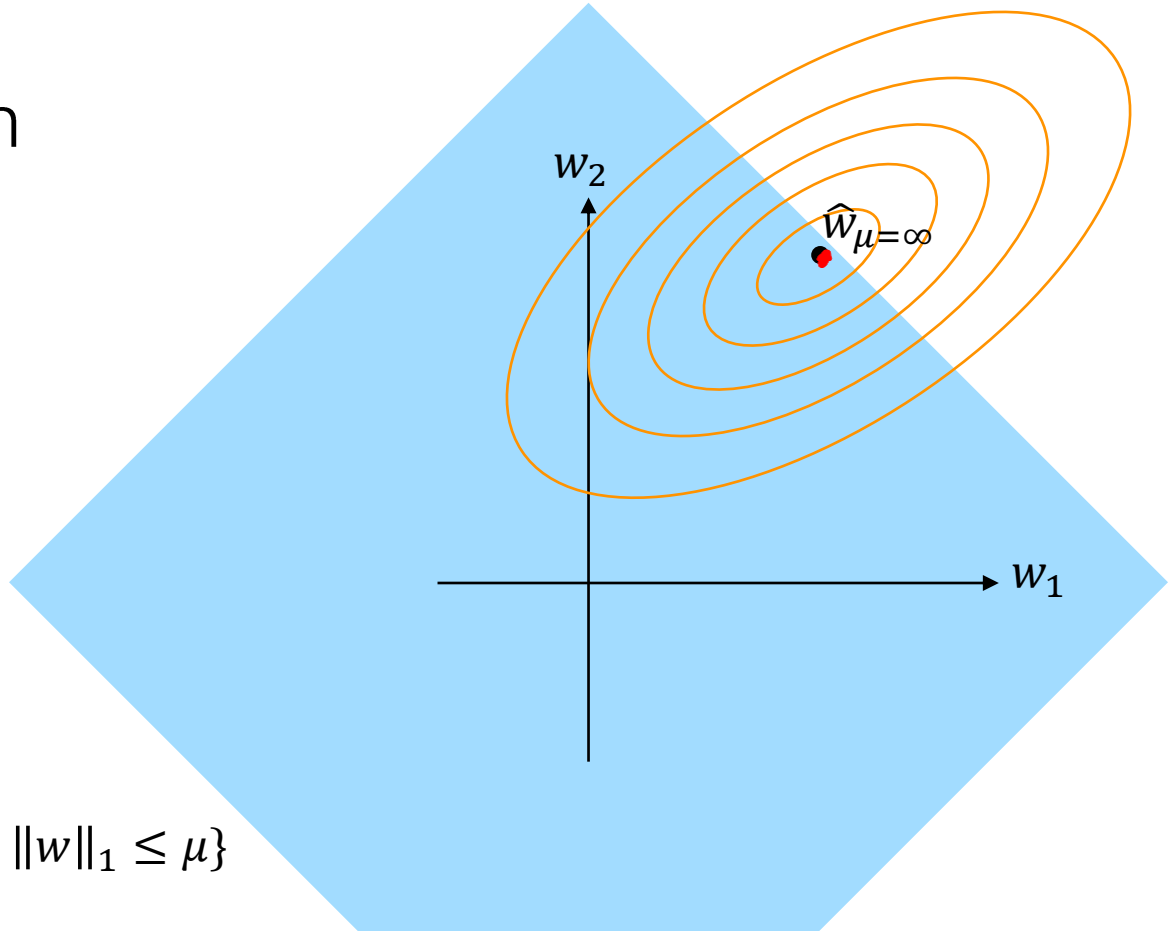


Feasible set $\{w \in \mathbb{R}^2: \|w\|_1 \leq \mu\}$

Why does Lasso give sparse solutions?

$$\hat{w}_{\text{lasso}} = \underset{w}{\operatorname{argmin}} \sum_i \left(y_i - x_i^\top w\right)^2 \quad \text{subject to} \quad \|w\|_1 \leq \mu$$

- For large μ, optimal solution doesn't change

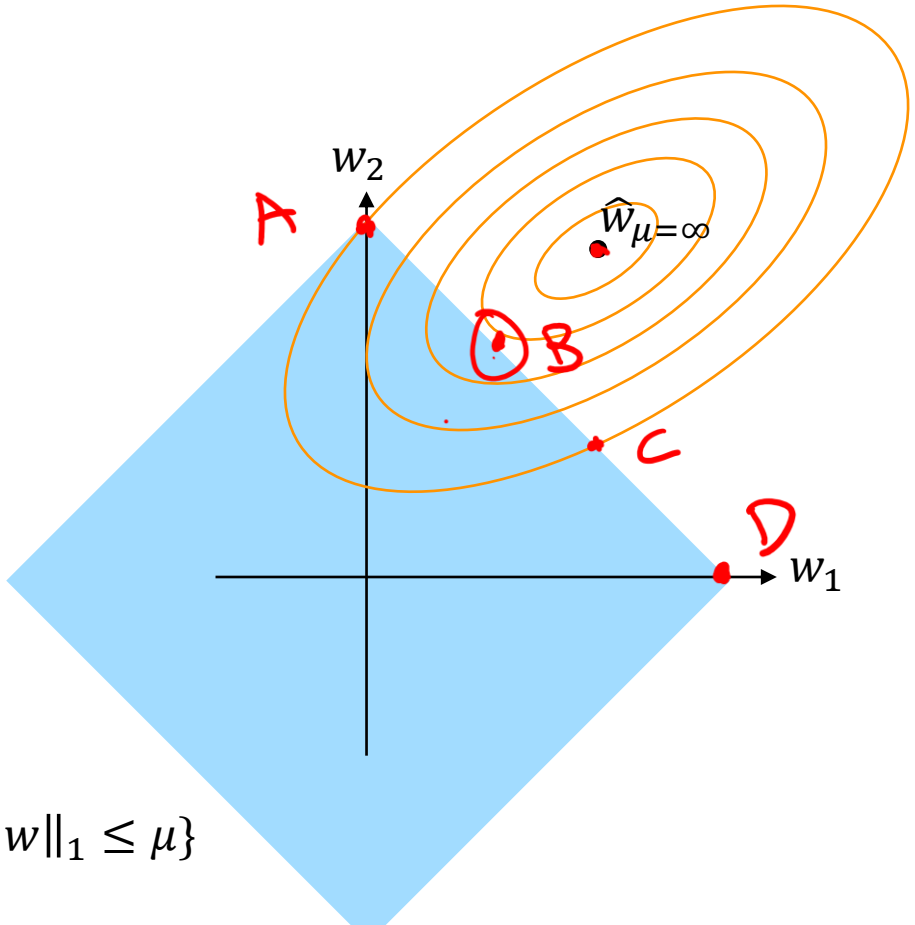


Feasible set $\{w \in \mathbb{R}^2 : \|w\|_1 \leq \mu\}$

Why does Lasso give sparse solutions?

$$\hat{w}_{\text{lasso}} = \underset{w}{\operatorname{argmin}} \sum_i \left(y_i - x_i^\top w\right)^2 \quad \text{subject to} \quad \|w\|_1 \leq \mu$$

- As μ decreases, feasible set shrinks
- Equivalent to increasing regularization strength λ

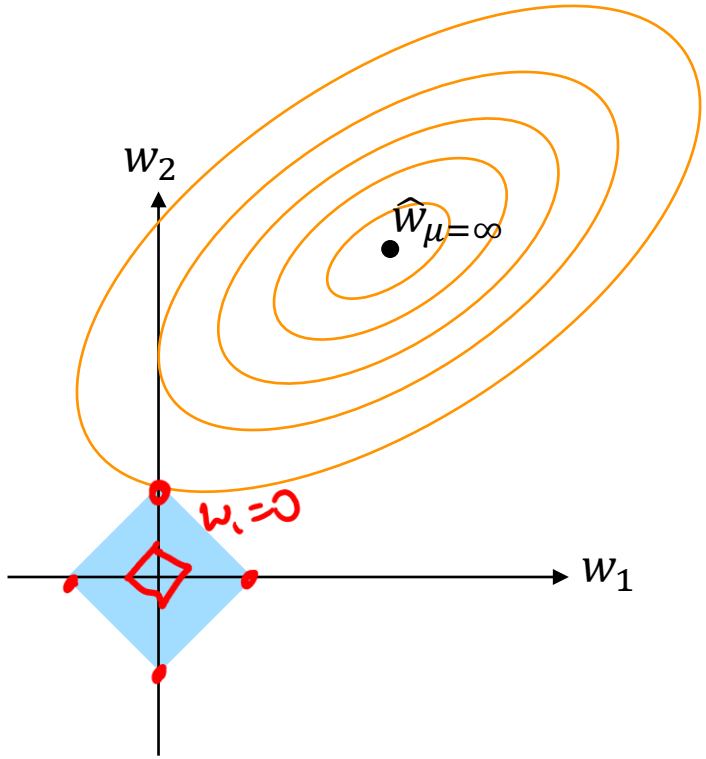


Feasible set $\{w \in \mathbb{R}^2 : \|w\|_1 \leq \mu\}$

Why does Lasso give sparse solutions?

$$\widehat{w}_{\text{lasso}} = \underset{w}{\operatorname{argmin}} \sum_i (y_i - x_i^\top w)^2 \quad \text{subject to} \quad \|w\|_1 \le \mu$$

- For small enough μ, optimal solution becomes sparse!
- This is because ℓ_1 ball is pointy (sharp edges aligned with axes)

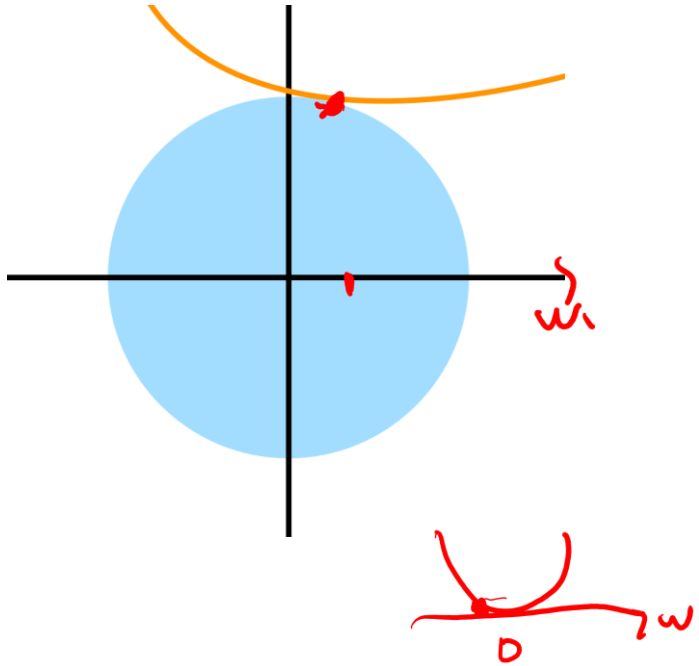


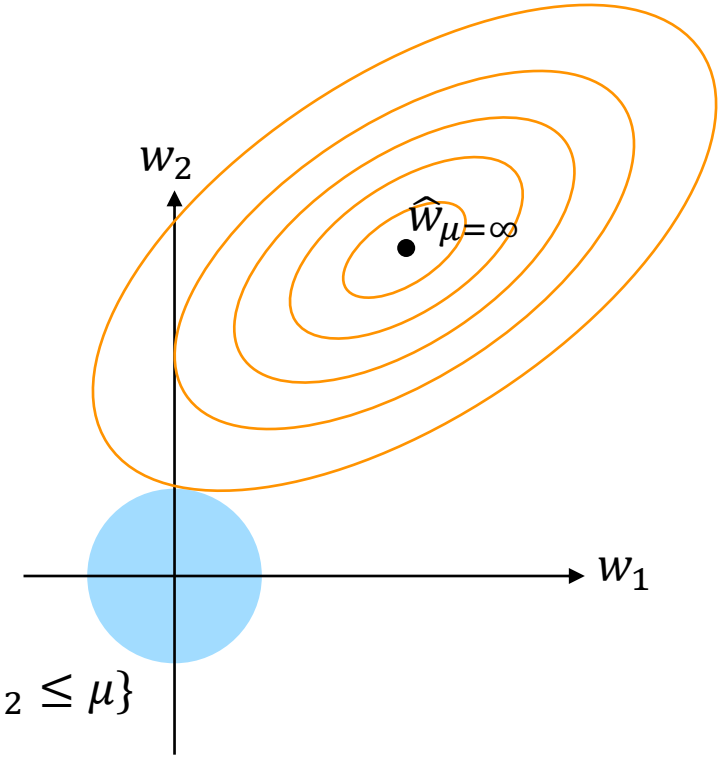
Feasible set $\{w \in \mathbb{R}^2 : \|w\|_1 \le \mu\}$

Why does Lasso give sparse solutions?

$$\widehat{w}_{\text{lasso}} = \underset{w}{\operatorname{argmin}} \sum_i (y_i - x_i^\top w)^2 \quad \text{subject to} \quad \|w\|_1 \le \mu$$

ℓ_2 ball instead finds dense solutions:



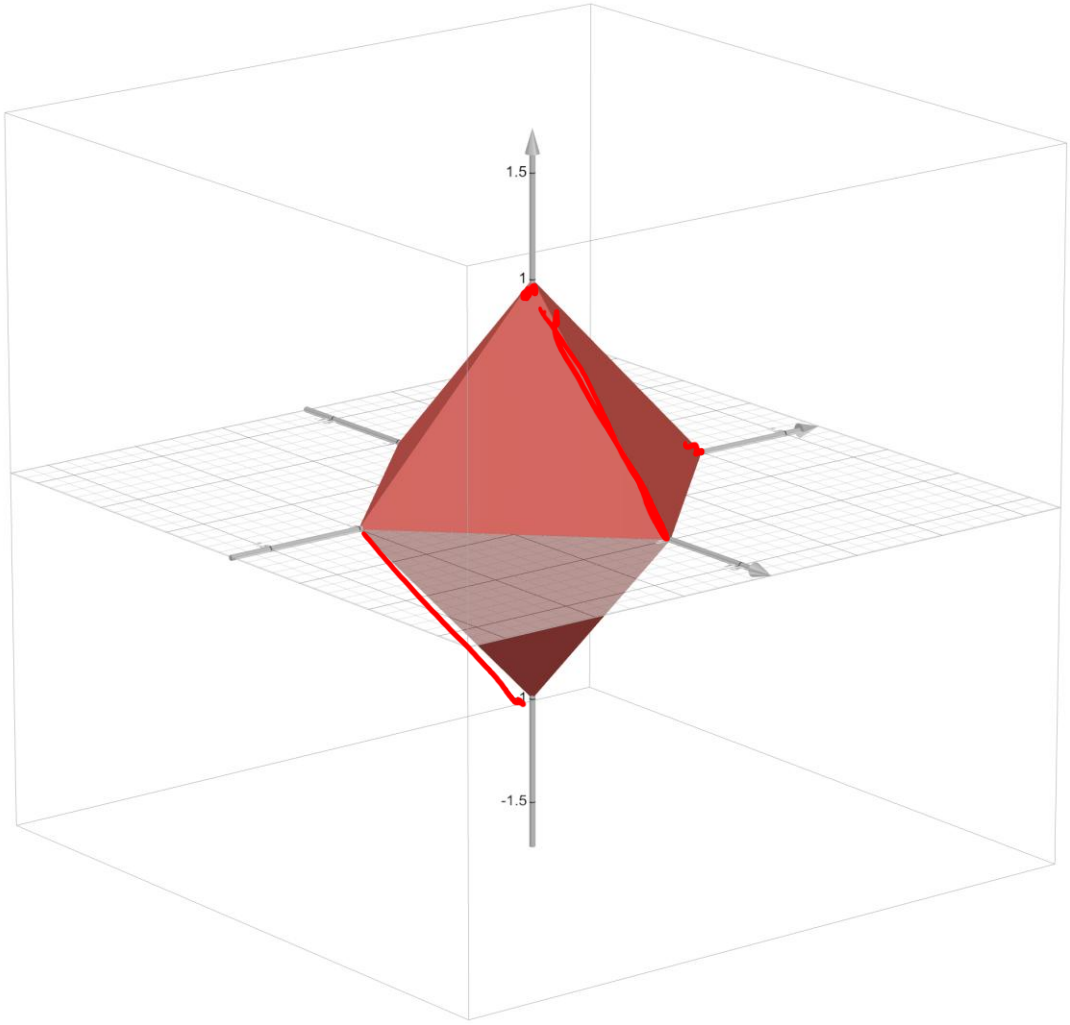


Feasible set $\{w \in \mathbb{R}^2 : \|w\|_2 \le \mu\}$

ℓ_1 ball in 3 dimensions

- Corners: 1-sparse
- Edges: 2-sparse

In higher dimensions, ℓ_1 ball is even “pointier”



Summary: Lasso regression (aka ℓ_1 regularization)

We can efficiently find sparse regression solutions by solving

$$\hat{w}_{\text{lasso}} = \underset{w}{\text{argmin}} \sum_i \left(y_i - x_i^\top w\right)^2 + \lambda \|w\|_1$$

It is equivalent to

$$\hat{w}_{\text{lasso}} = \underset{w}{\text{argmin}} \sum_i \left(y_i - x_i^\top w\right)^2 \quad \text{subject to} \quad \|w\|_1 \leq \mu$$

Next: How do we actually solve it?

Gradient Descent

CSE 446/546

Sewoong Oh & Pang Wei Koh

Why gradient descent?

$f(w) = \|y - Xw\|_2^2$

$\nabla f(w) = 2X^T(y - Xw) = 0$

$\Rightarrow \hat{w} = (X^TX)^{-1}X^Ty$

- Standard ML paradigm: Define loss, then optimize

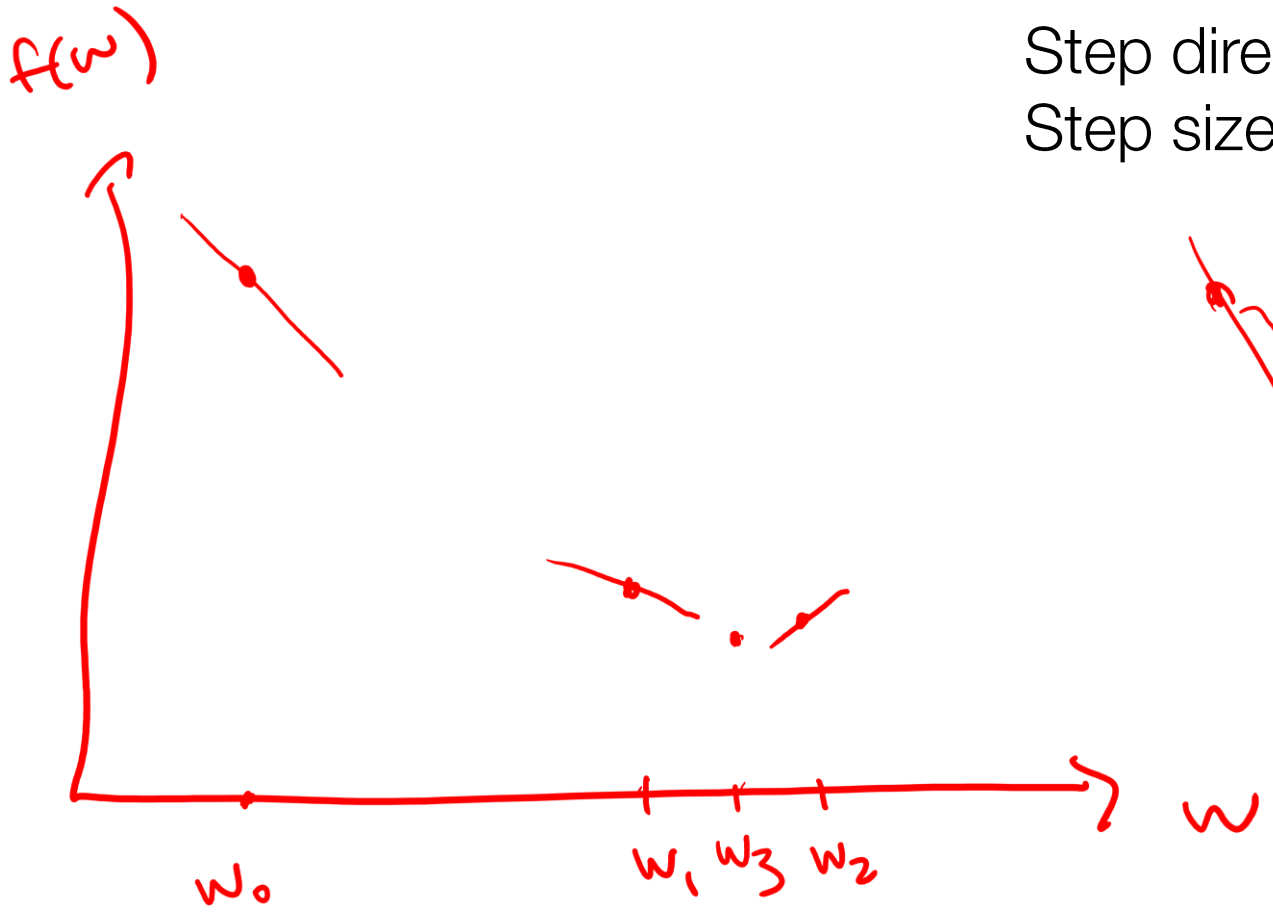
$$\widehat{w}_{LS} = \arg\min_w ||\mathbf{y} - \mathbf{X}w||_2^2 + \lambda\|w\|_1$$

$L(w)$

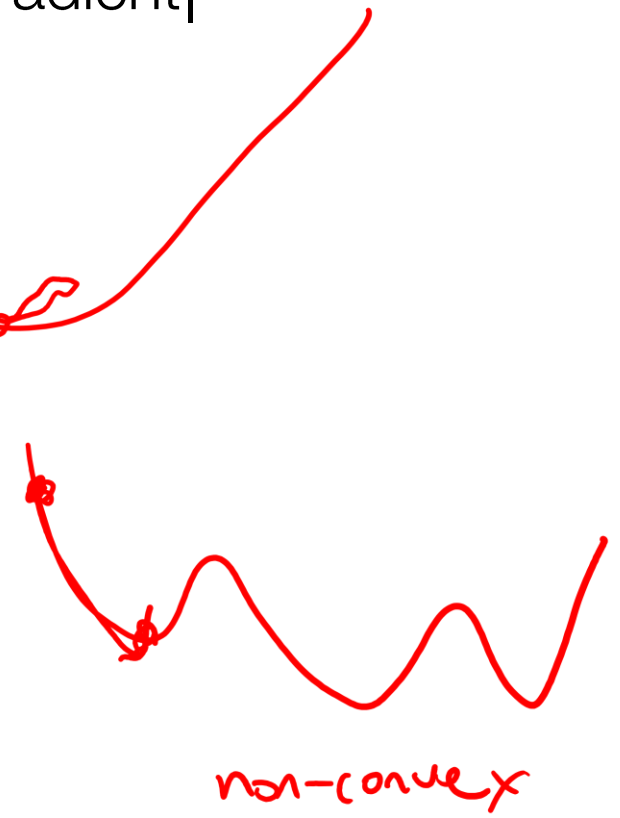
$= (X^TX)^{-1}X^Ty$

- But, no closed-form solutions for most losses we use in practice.
- Key idea: Iterative methods
- Used everywhere!

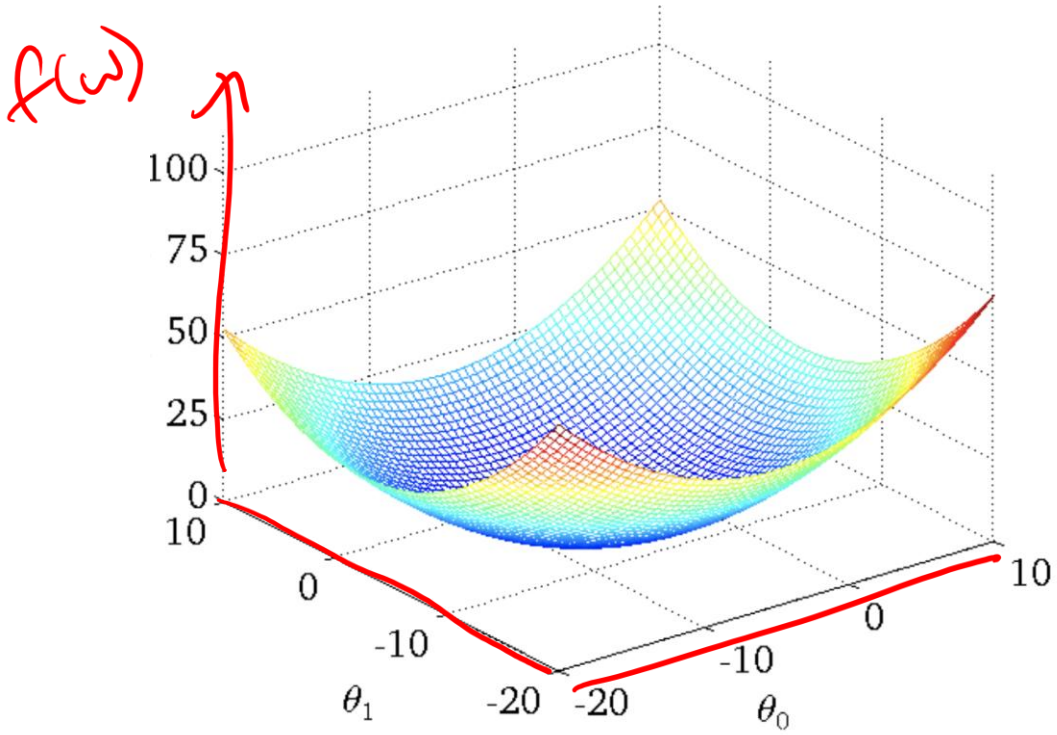
Gradient descent in one dimension

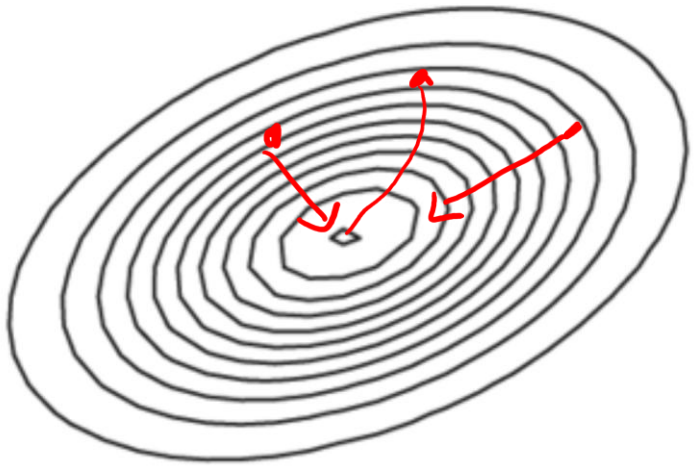


Step direction: −gradient
Step size: ∝ |gradient|



Gradient descent in multiple dimensions





Lecture plan

- Gradient descent algorithm + examples

- Theoretical analysis
  - When does it work?
  - How quickly does it converge?
  - How do we choose a step size?
  - Key idea: Convexity

- Not tested on proof details, but concepts are important & practical

Algorithm: Gradient descent

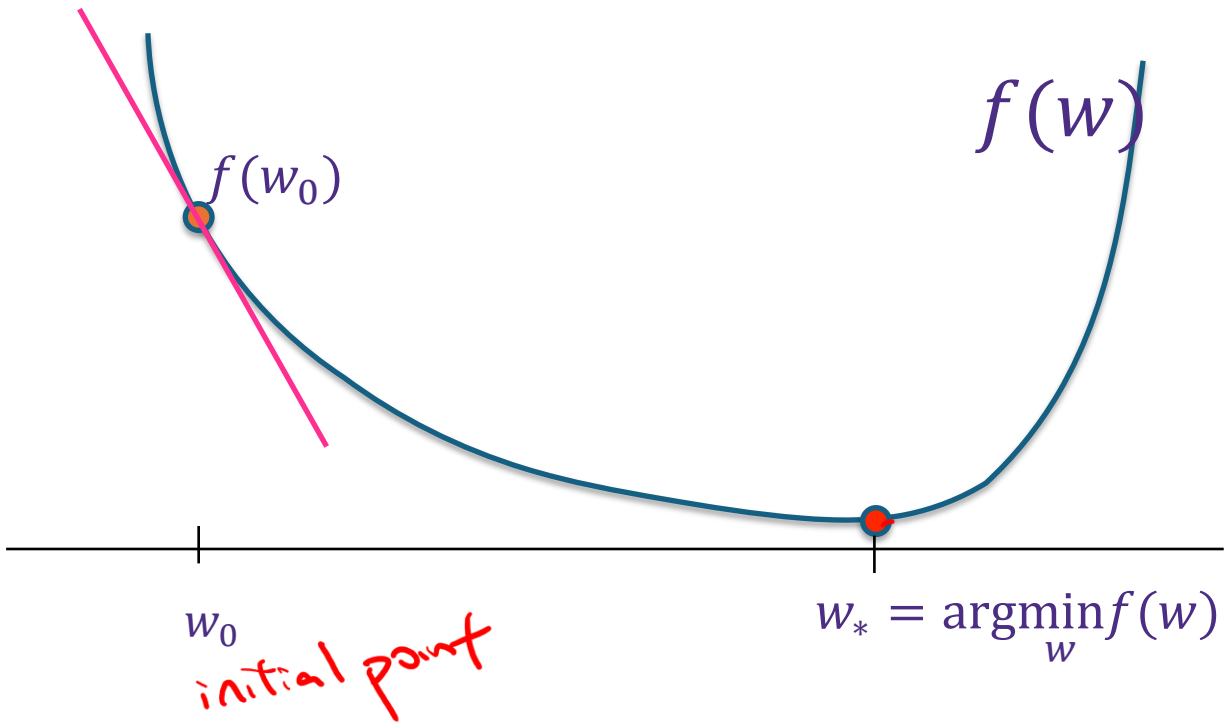
Algorithm

For t=0,1,2,3, …

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

Hyperparameters:

- Initial point w_0
- Step size η

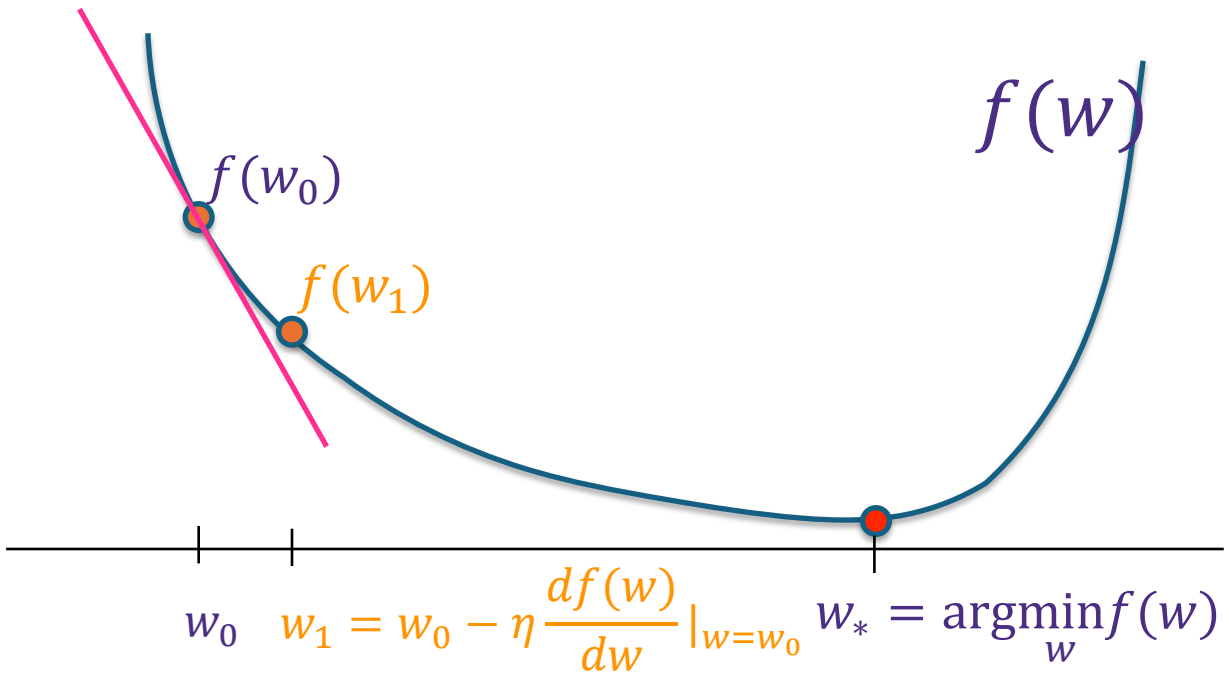


Algorithm: Gradient descent

Algorithm
For t=0,1,2,3, …

$$w_{t+1} = w_t - \eta \frac{df(w)}{dw}|_{w=w_t}$$

Hyperparameters:
- Initial point w_0
- Step size η

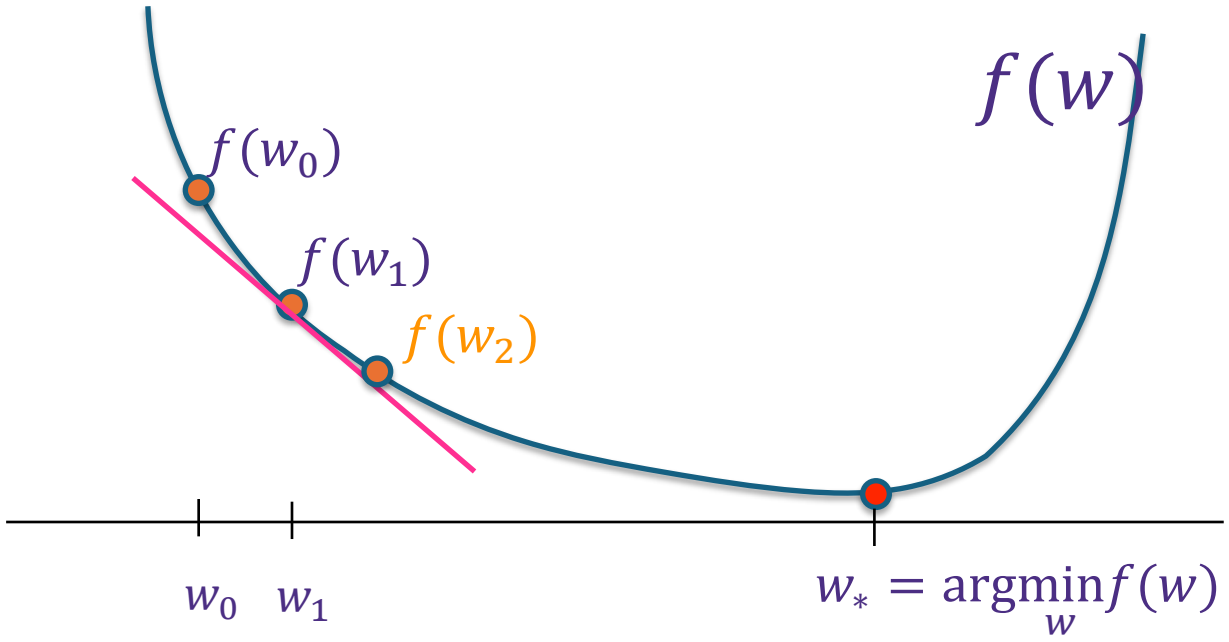


Algorithm: Gradient descent

Algorithm
For t=0,1,2,3, …

$$w_{t+1} = w_t - \eta \frac{df(w)}{dw}|_{w=w_t}$$

Hyperparameters:
- Initial point w_0
- Step size η

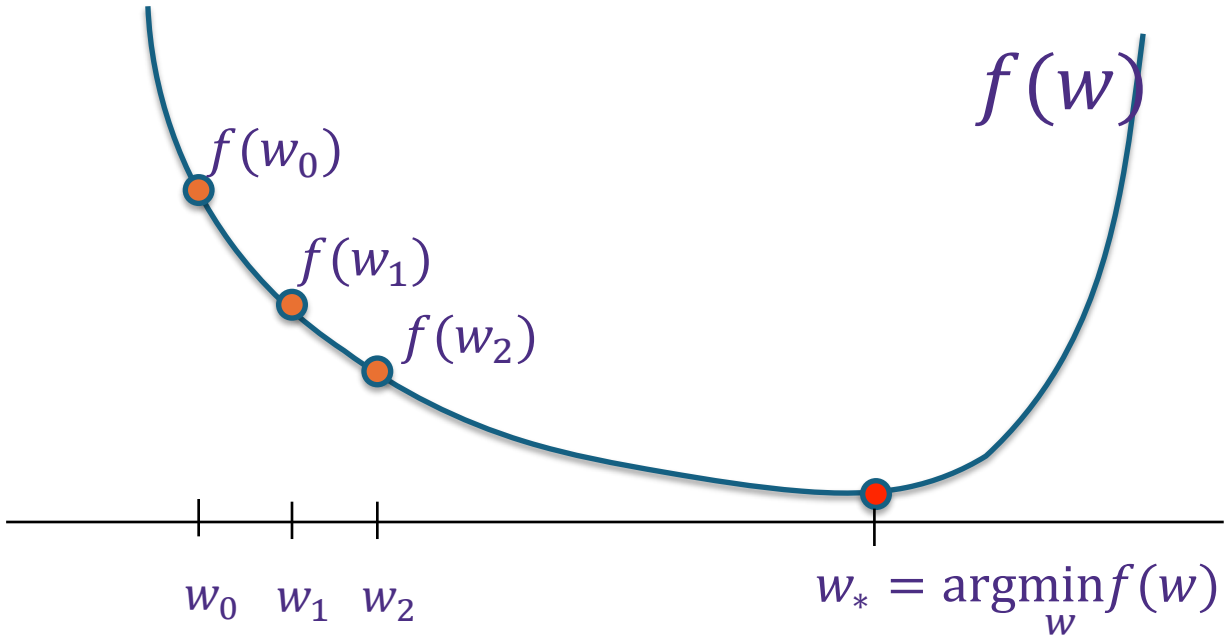


Algorithm: Gradient descent

Algorithm
For t=0,1,2,3, …

$$w_{t+1} = w_t - \eta \frac{df(w)}{dw}|_{w=w_t}$$

Hyperparameters:
- Initial point w_0
- Step size η

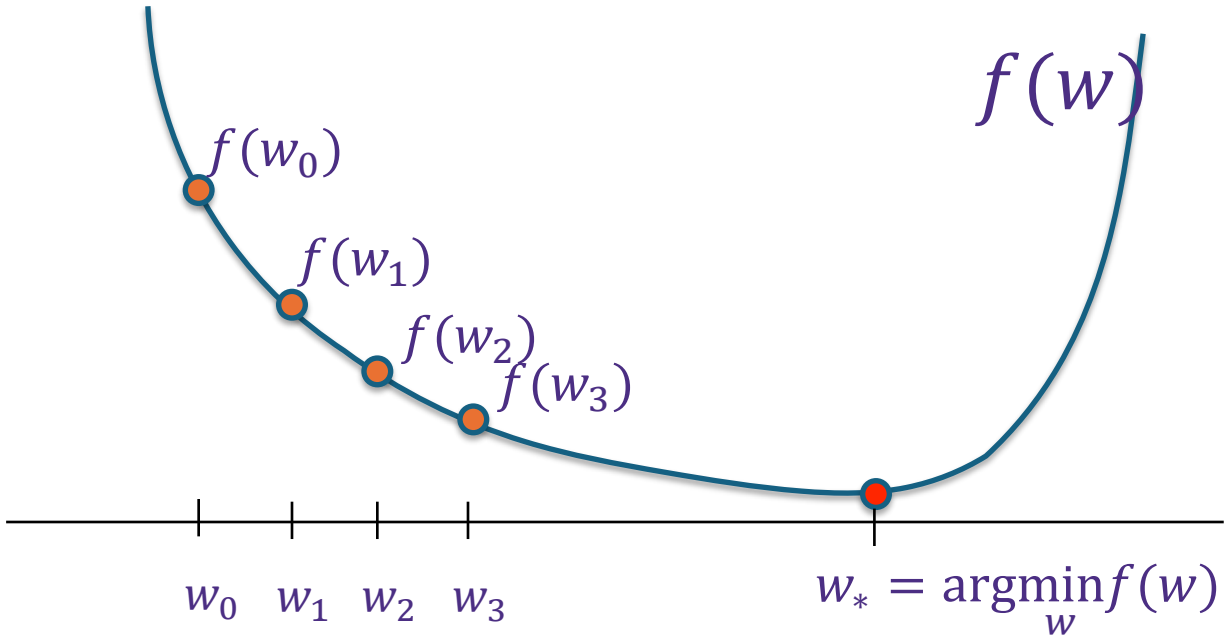


Algorithm: Gradient descent

Algorithm
For t=0,1,2,3, …

$$w_{t+1} = w_t - \eta \frac{df(w)}{dw}|_{w=w_t}$$

Hyperparameters:
- Initial point w_0
- Step size η

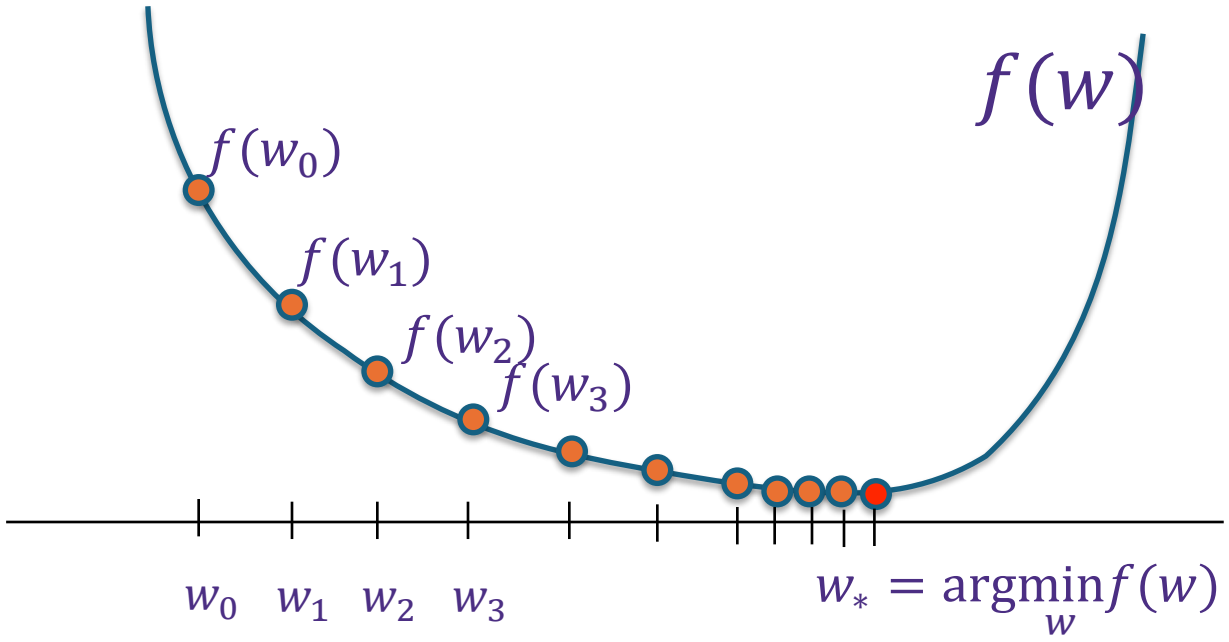


Algorithm: Gradient descent

Algorithm
For t=0,1,2,3, …

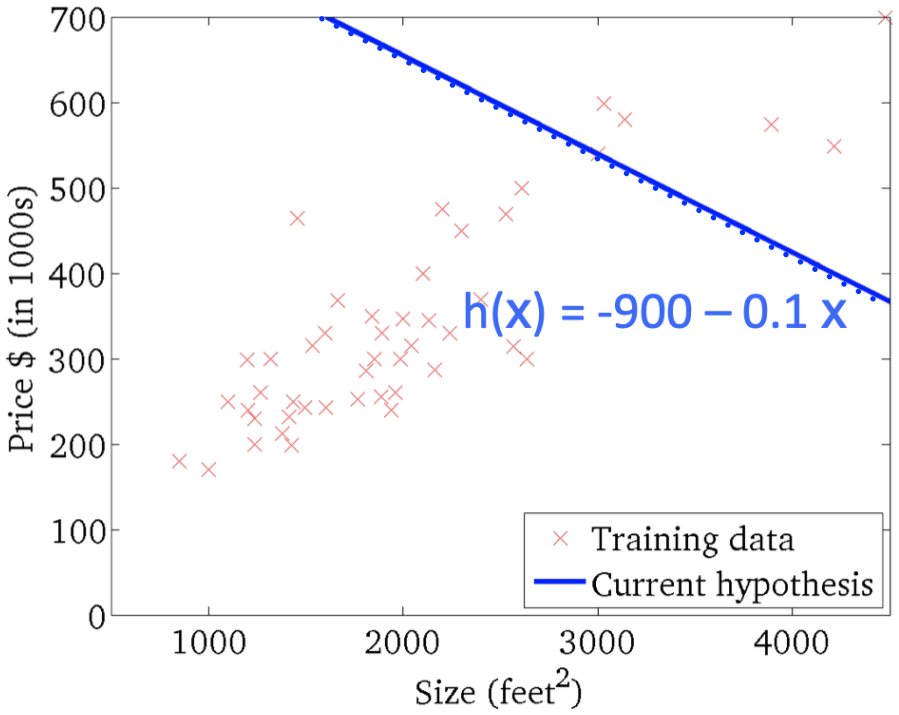
$$w_{t+1} = w_t - \eta \frac{df(w)}{dw}|_{w=w_t}$$

Hyperparameters:
- Initial point w_0
- Step size η



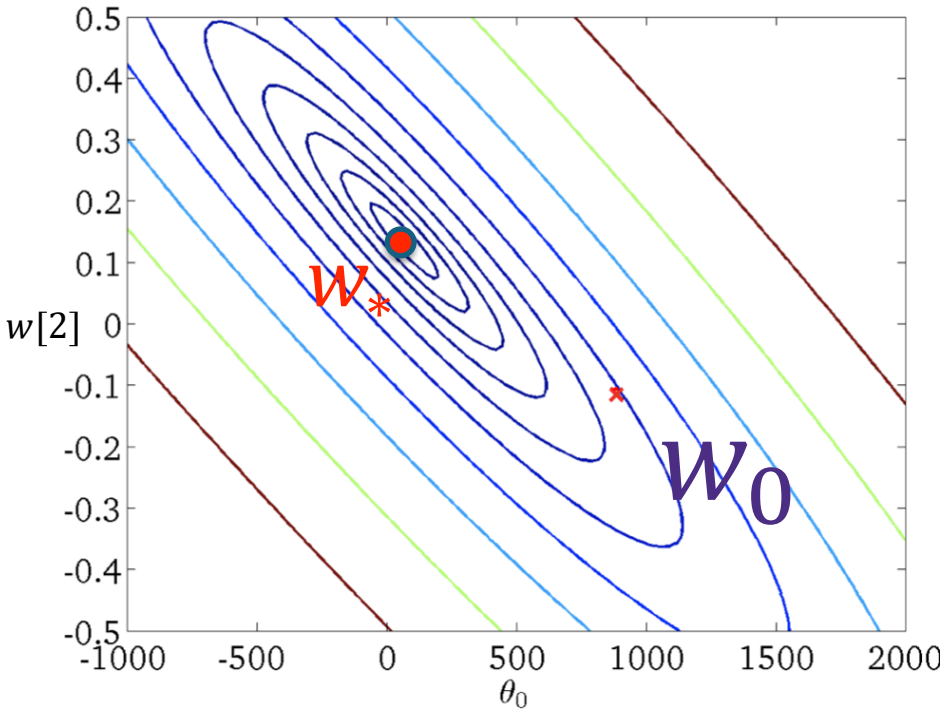
Note that as $t \to \infty$ we have $\frac{df(w)}{dw}|_{w=w_t} \to 0$

$$\{(x_i, y_i)\}_{i=1}^{n}$$

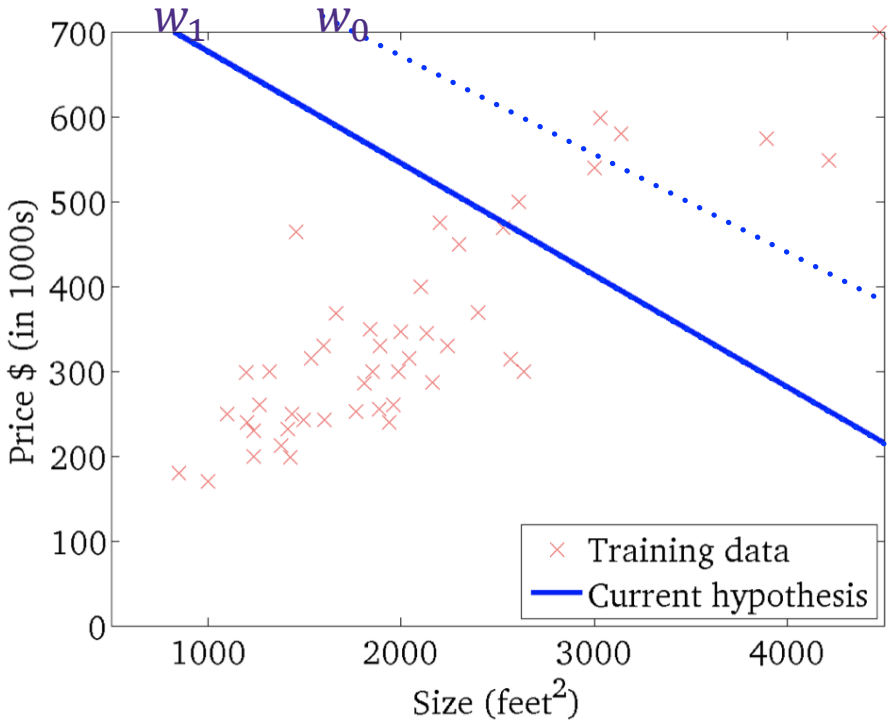


Evolution of the predictor

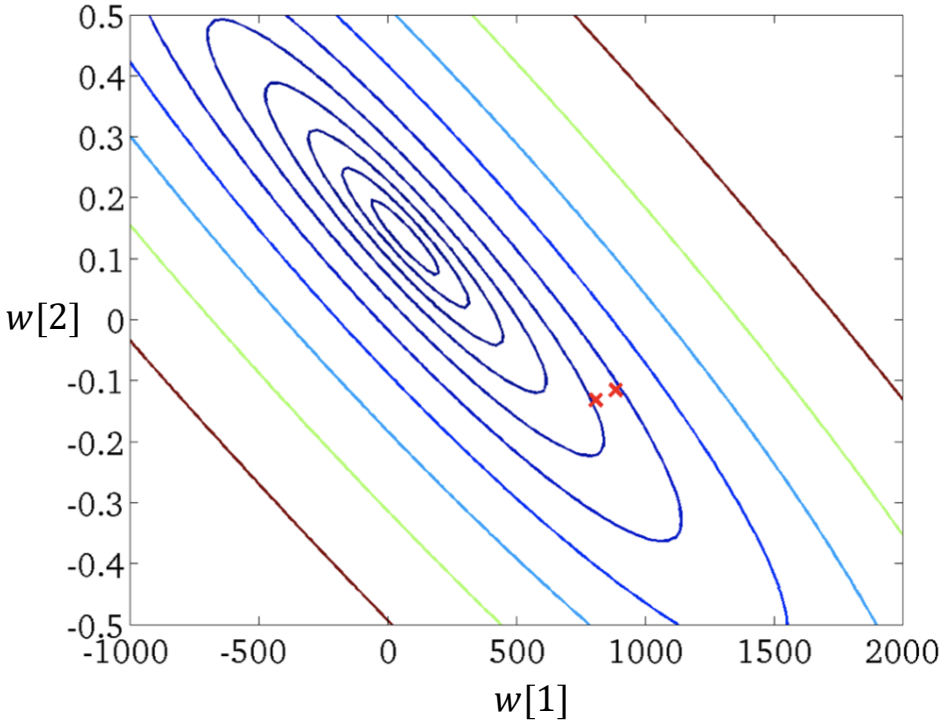
$$w_{t+1} \leftarrow w_t - \eta \cdot \nabla_w f(w_t)$$



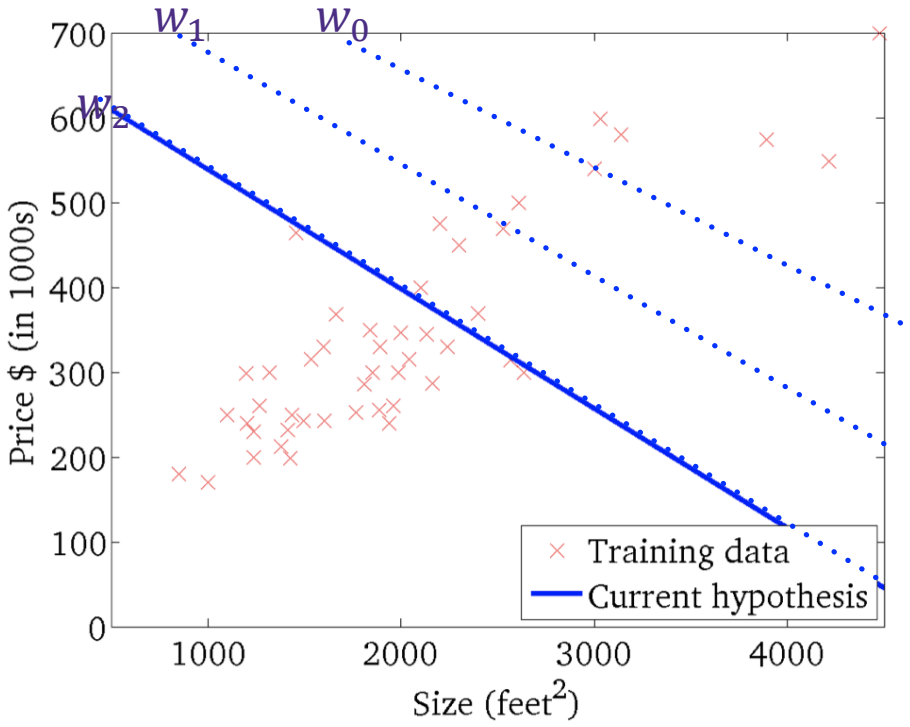
GD dynamics in the parameter space



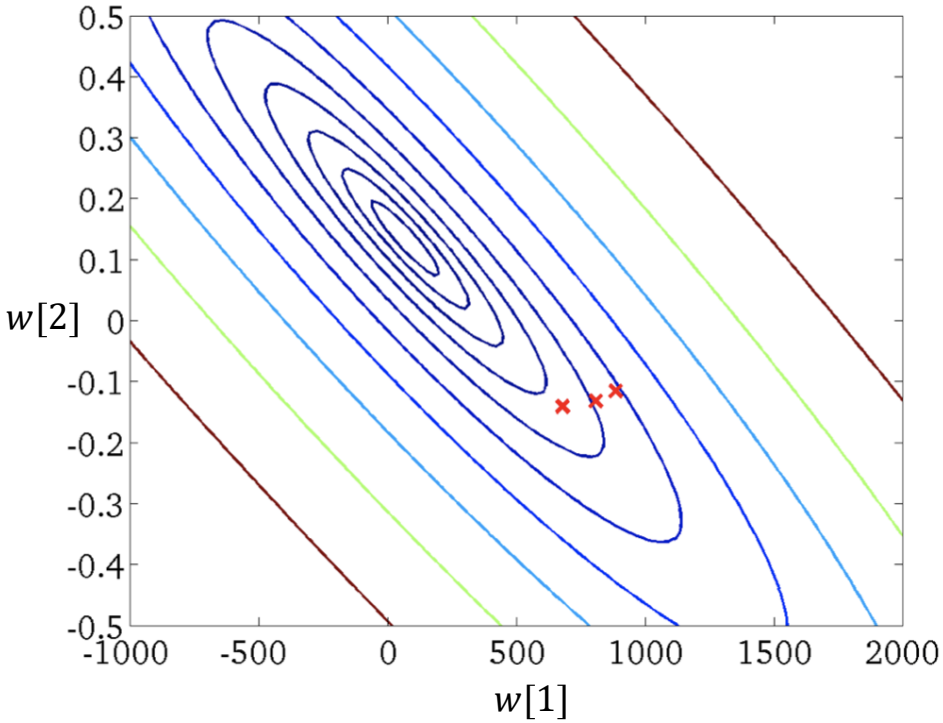
Evolution of the predictor



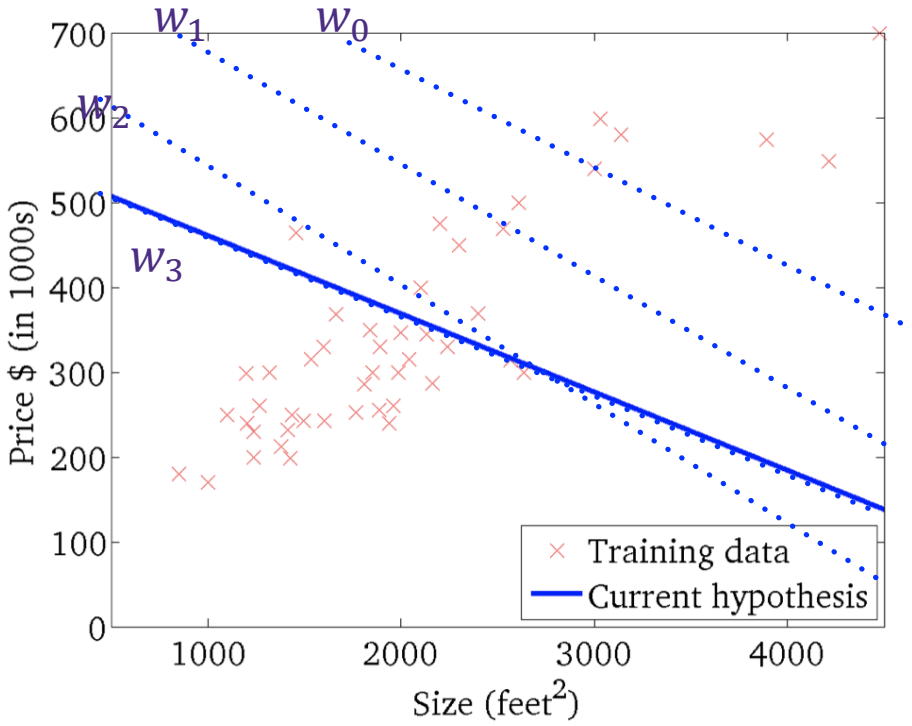
GD dynamics in the parameter space



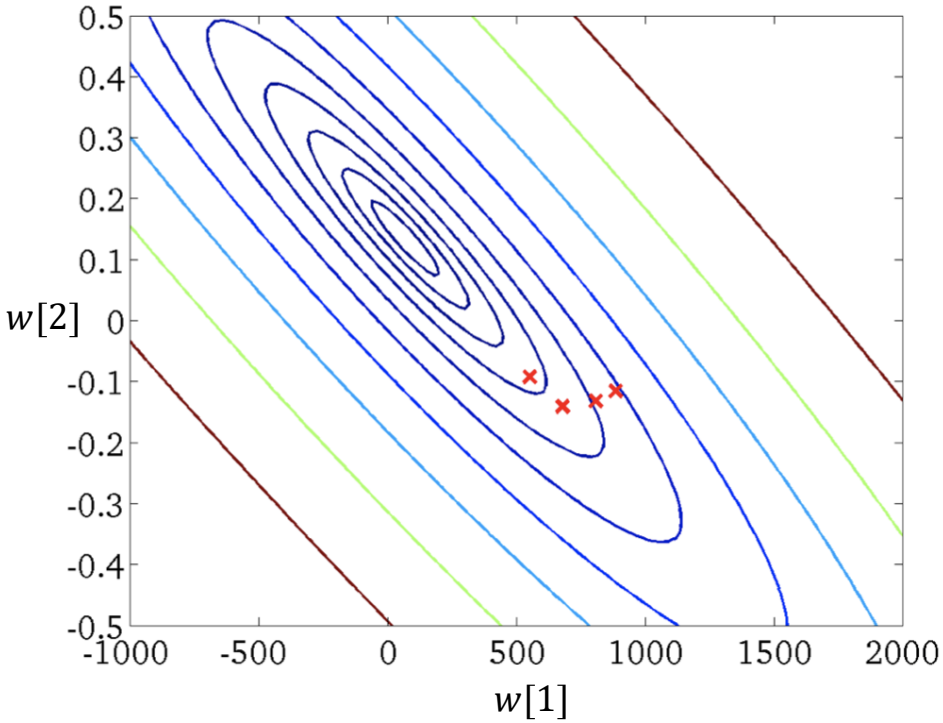
Evolution of the predictor



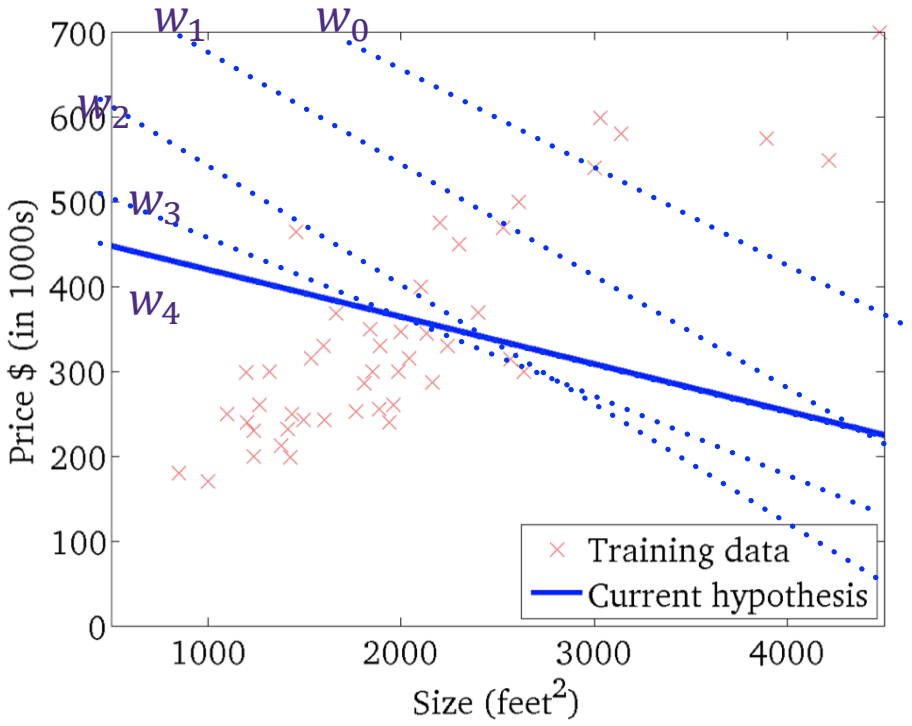
GD dynamics in the parameter space



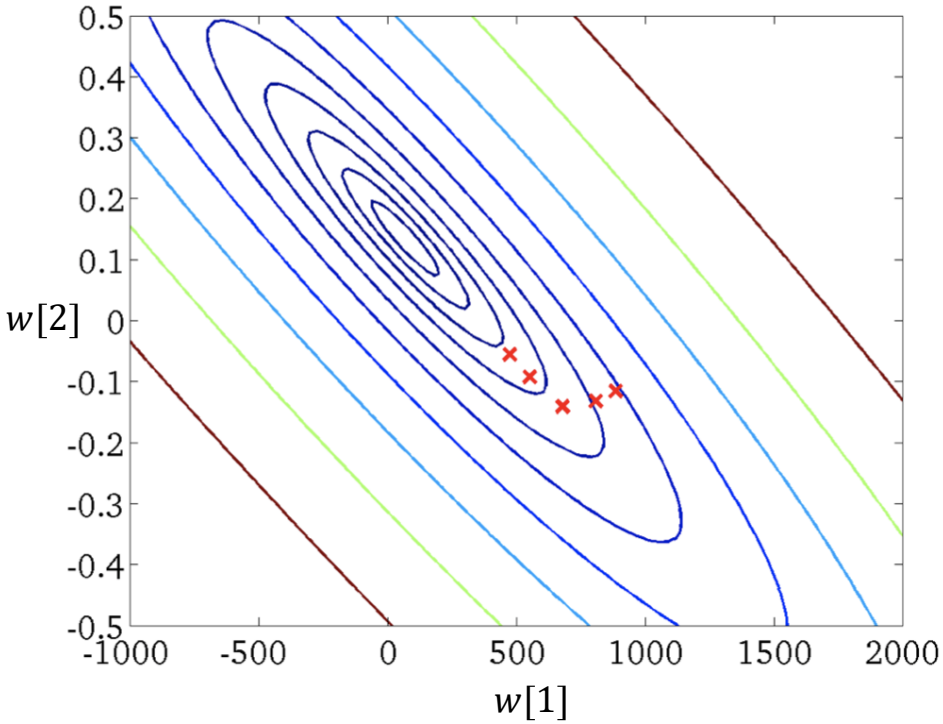
Evolution of the predictor



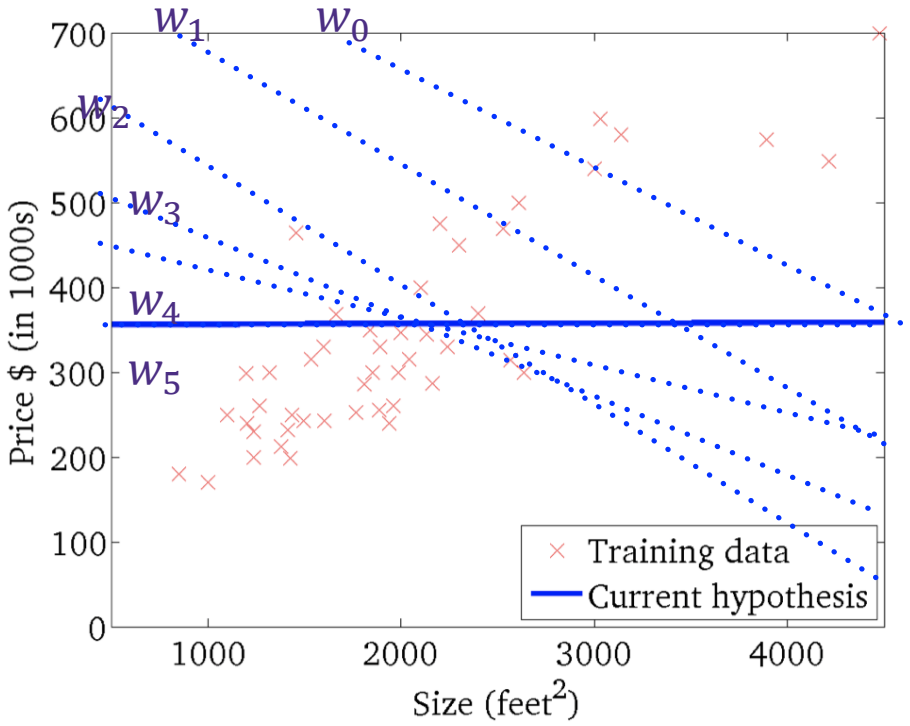
GD dynamics in the parameter space



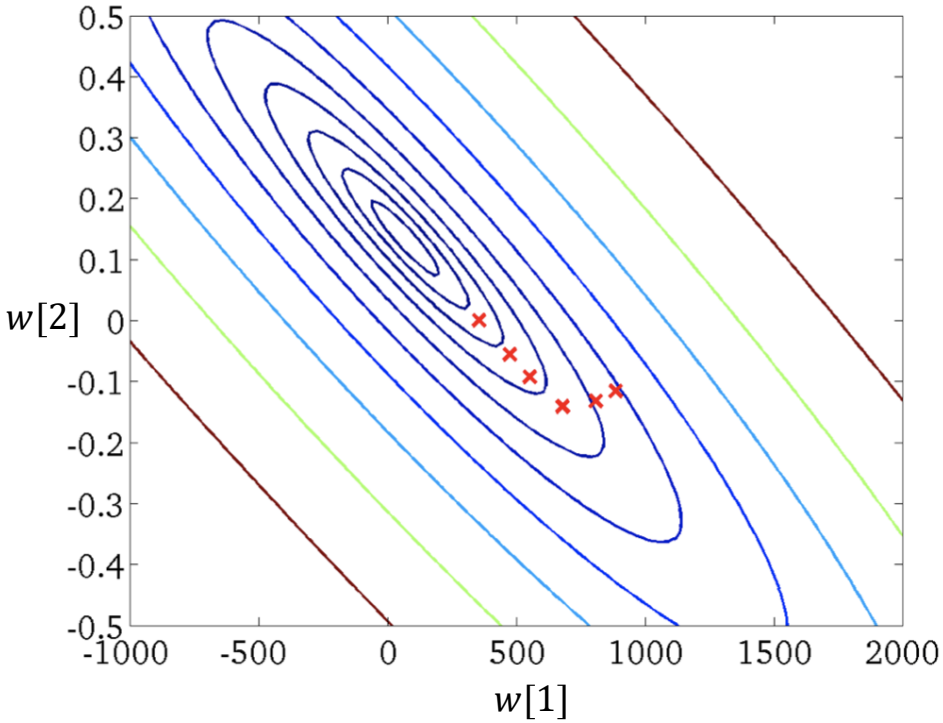
Evolution of the predictor



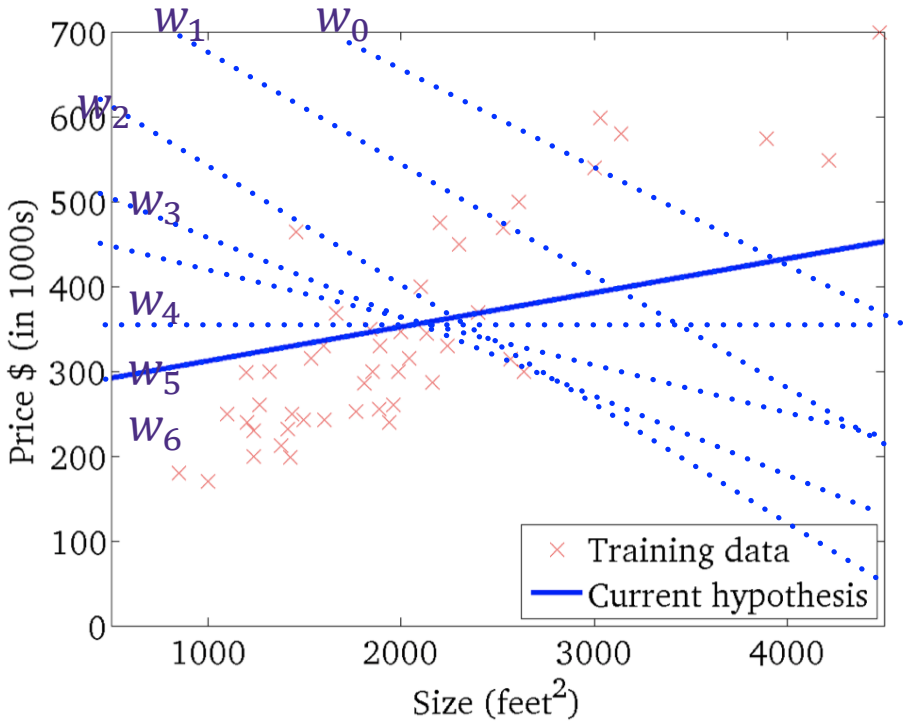
GD dynamics in the parameter space



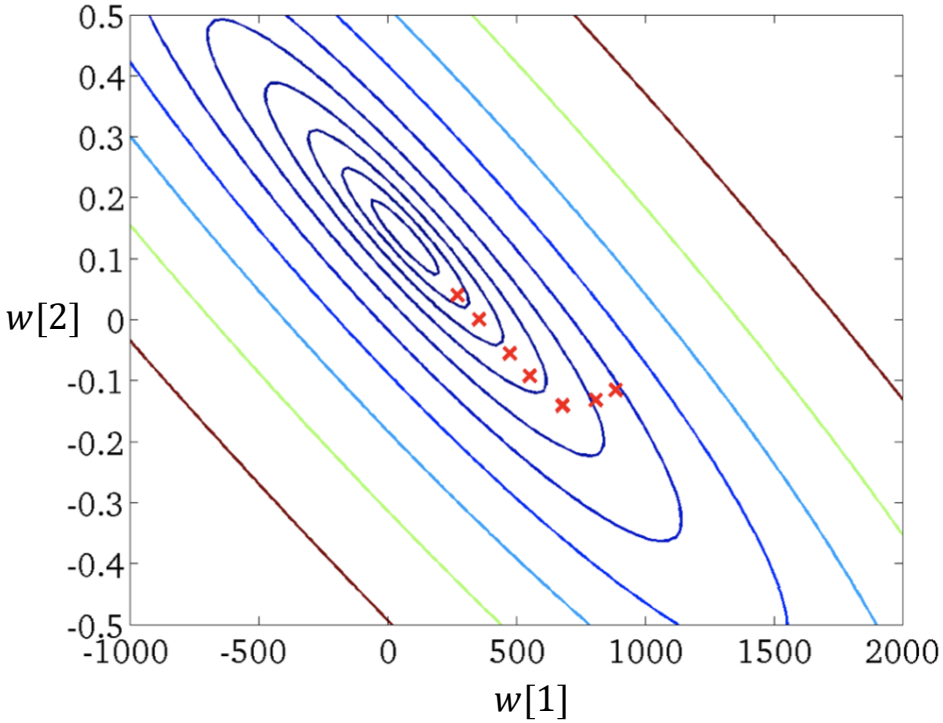
Evolution of the predictor



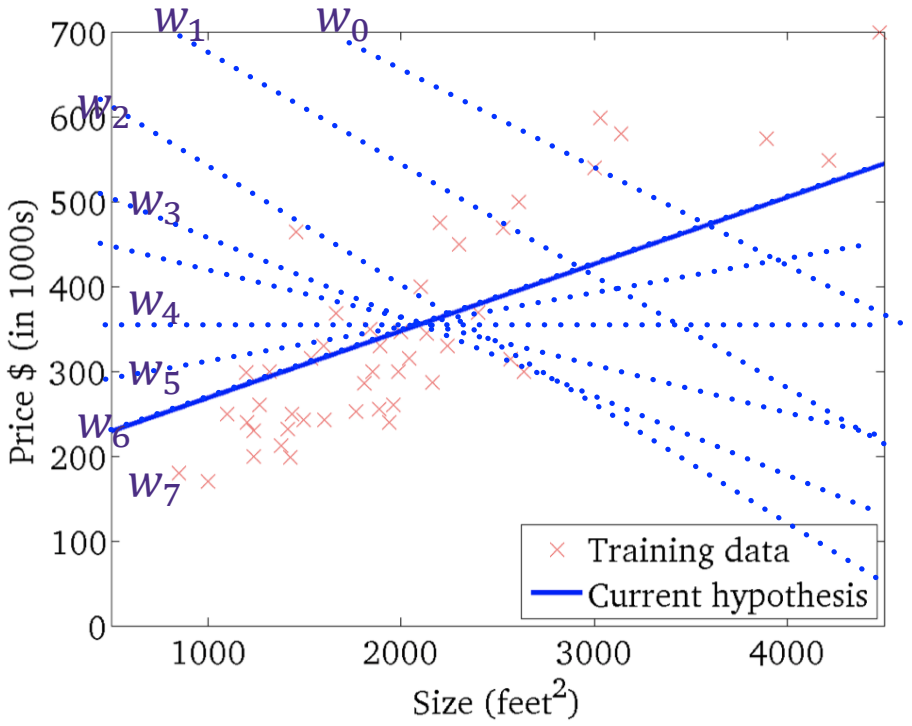
GD dynamics in the parameter space



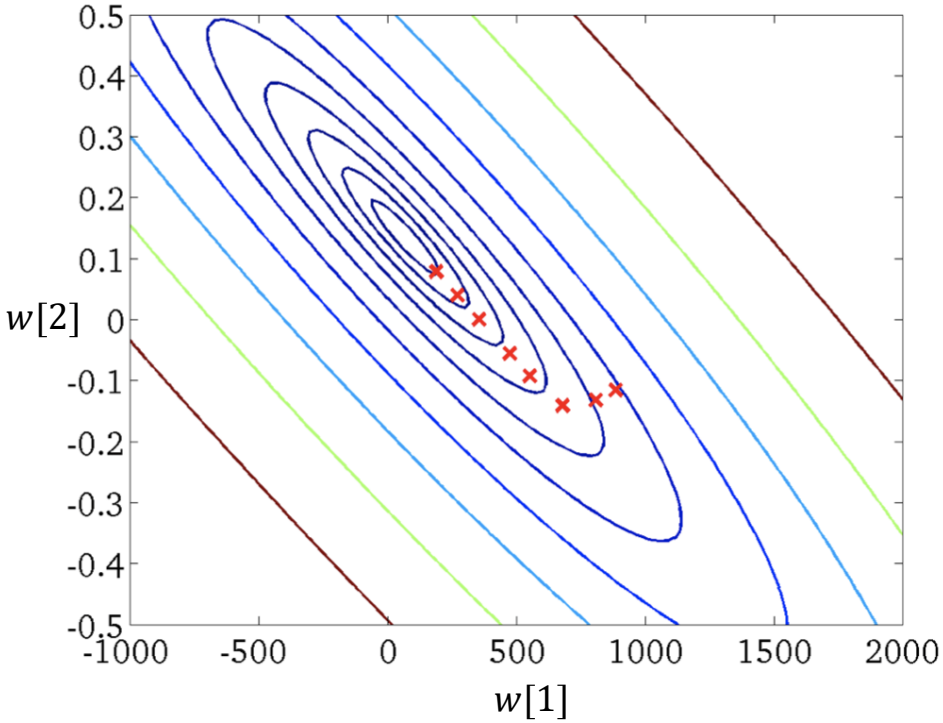
Evolution of the predictor



GD dynamics in the parameter space

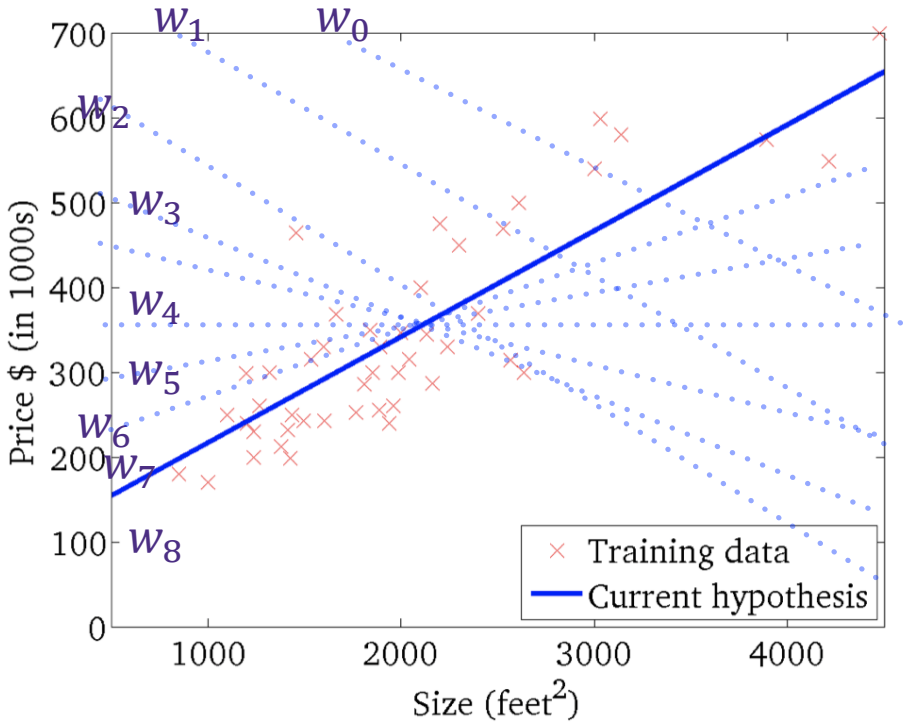


Evolution of the predictor

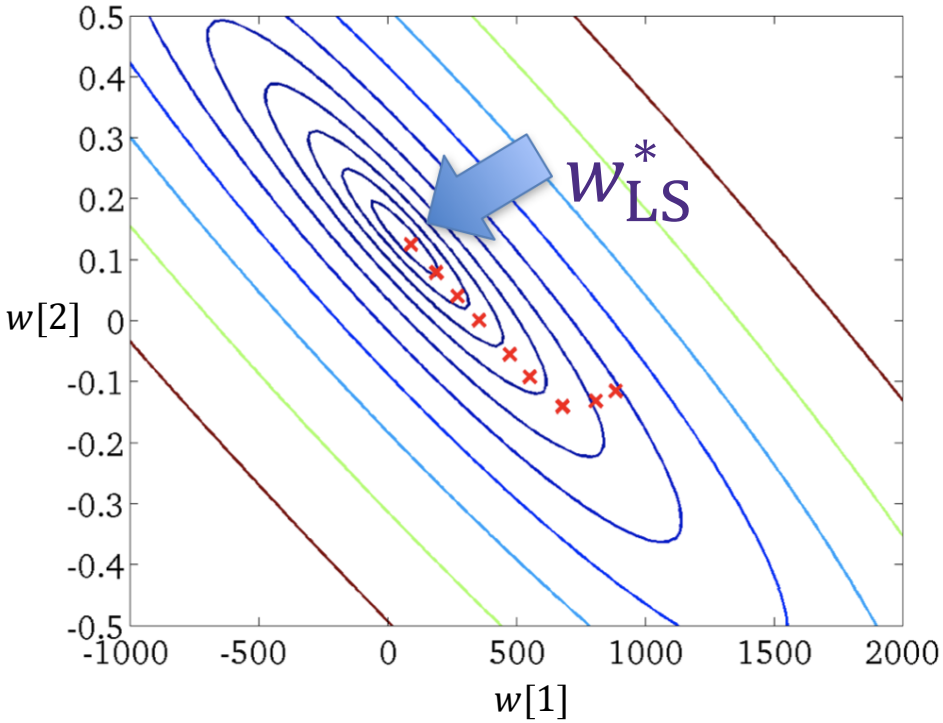


GD dynamics in the parameter space

See website for some gradient descent demo code



Evolution of the predictor



GD dynamics in the parameter space

Warmup: Quadratic functions

$$\hat{w} = \underset{w}{\operatorname{argmin}} \; \underbrace{aw^2 + bw + c}_{f(w)}$$

$$\frac{df}{dw} = 2aw + b$$

$$w_{t+1} := w_t - \eta \frac{df}{dw}\Big|_{w=w_t}$$

$$= w_t - \eta(2aw_t + b)$$

Example: Linear regression

$$\hat{w} = \underset{w}{\operatorname{argmin}} \underbrace{\frac{1}{2}\sum_i \left(y_i - x_i^\top w\right)^2}_{f(w)}$$

$$\nabla f(w) = -\sum_i (y_i - x_i^\top w)\, x_i$$

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

$$= w_t + \eta \sum_i (y_i - x_i^\top w_t)\, x_i$$

Example: Lasso

$$\hat{w} = \underset{w}{\operatorname{argmin}} \frac{1}{2} \|y - Xw\|_2^2 + \lambda \|w\|_1$$

How do you choose a step size?

- If η too small, converges very, very slowly.
- If η too big, does not converge!
- In practice: guess and check

Lecture plan

- Gradient descent algorithm + examples

- Theoretical analysis ← we are here
  - When does it work?
  - How quickly does it converge?
  - How do we choose a step size?
  - Key idea: Convexity

- Not tested on proof details, but concepts are important & practical

Convexity

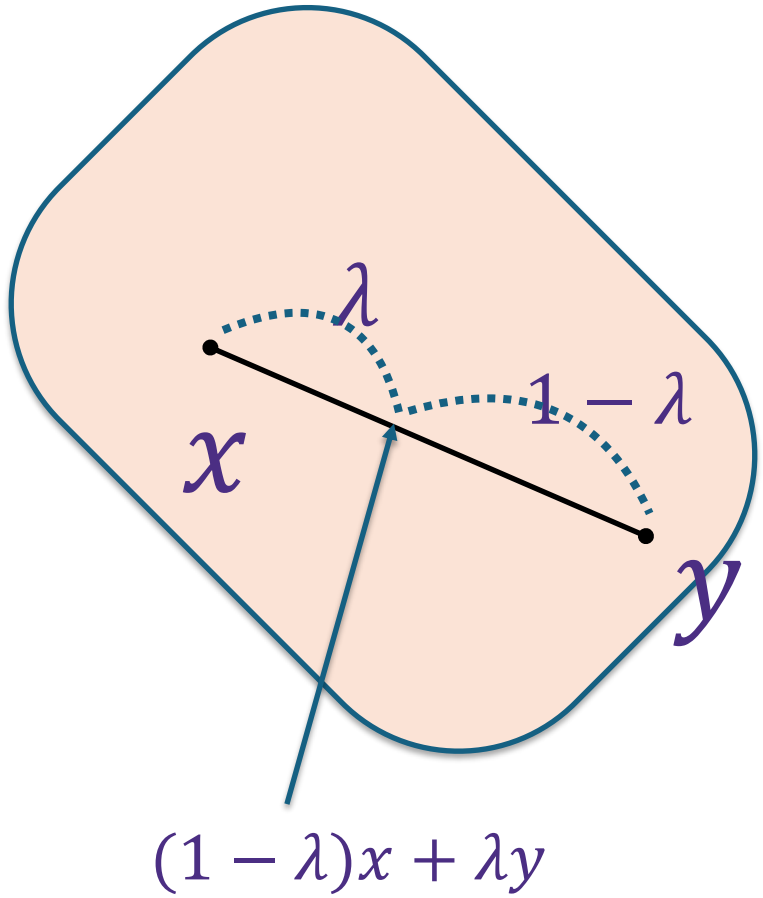
- Optimization problems are hard to solve in general
- The exception: convex optimization
  - Objective is a convex function
  - Constraints are convex sets

- Special class of problems that can be solved efficiently
- Surprisingly common in practice

What is a convex set?

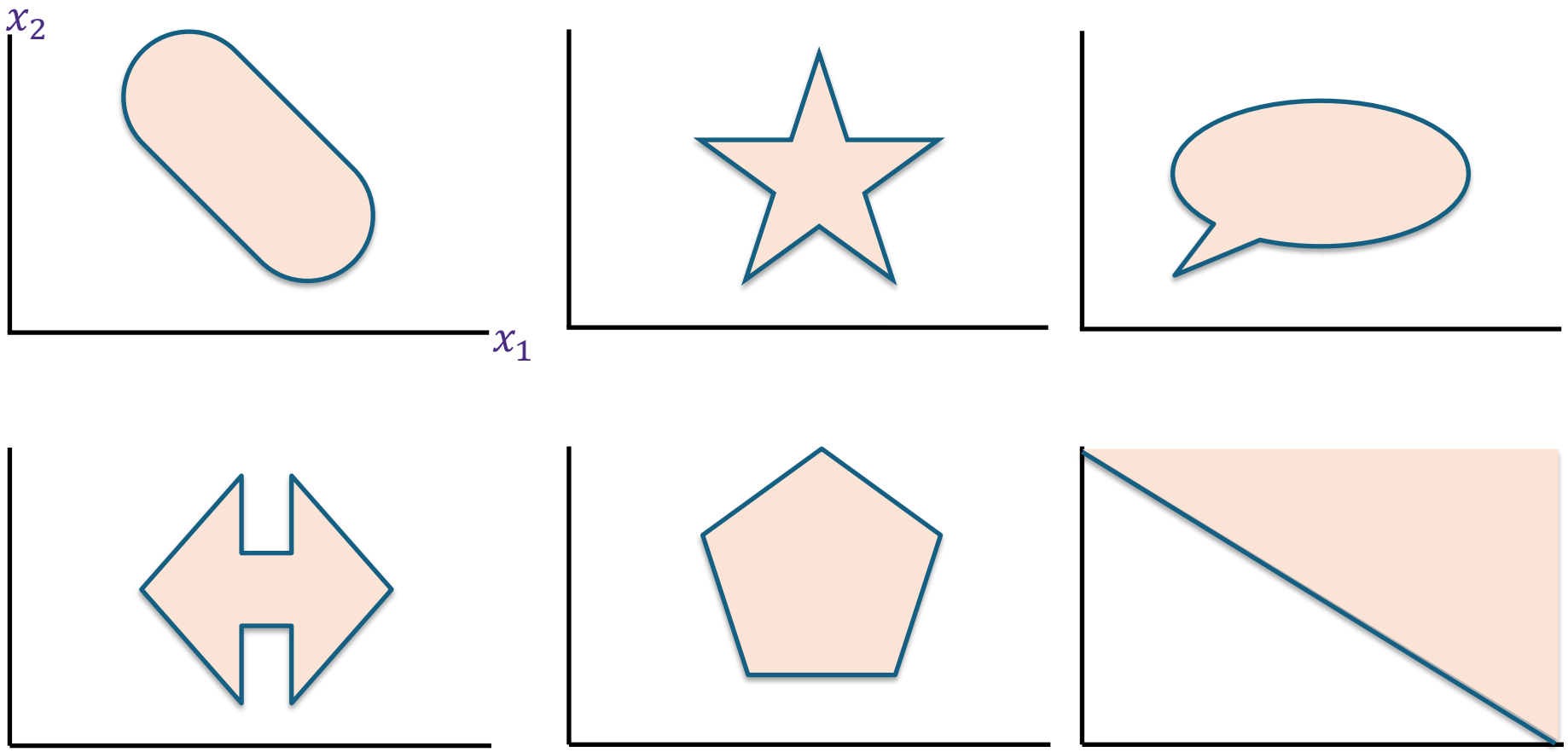
What is a convex set?

A set $K \subset \mathbb{R}^d$ is convex if $(1-\lambda)x + \lambda y \in K$ for all $x, y \in K$ and $\lambda \in [0,1]$



What is a convex set?

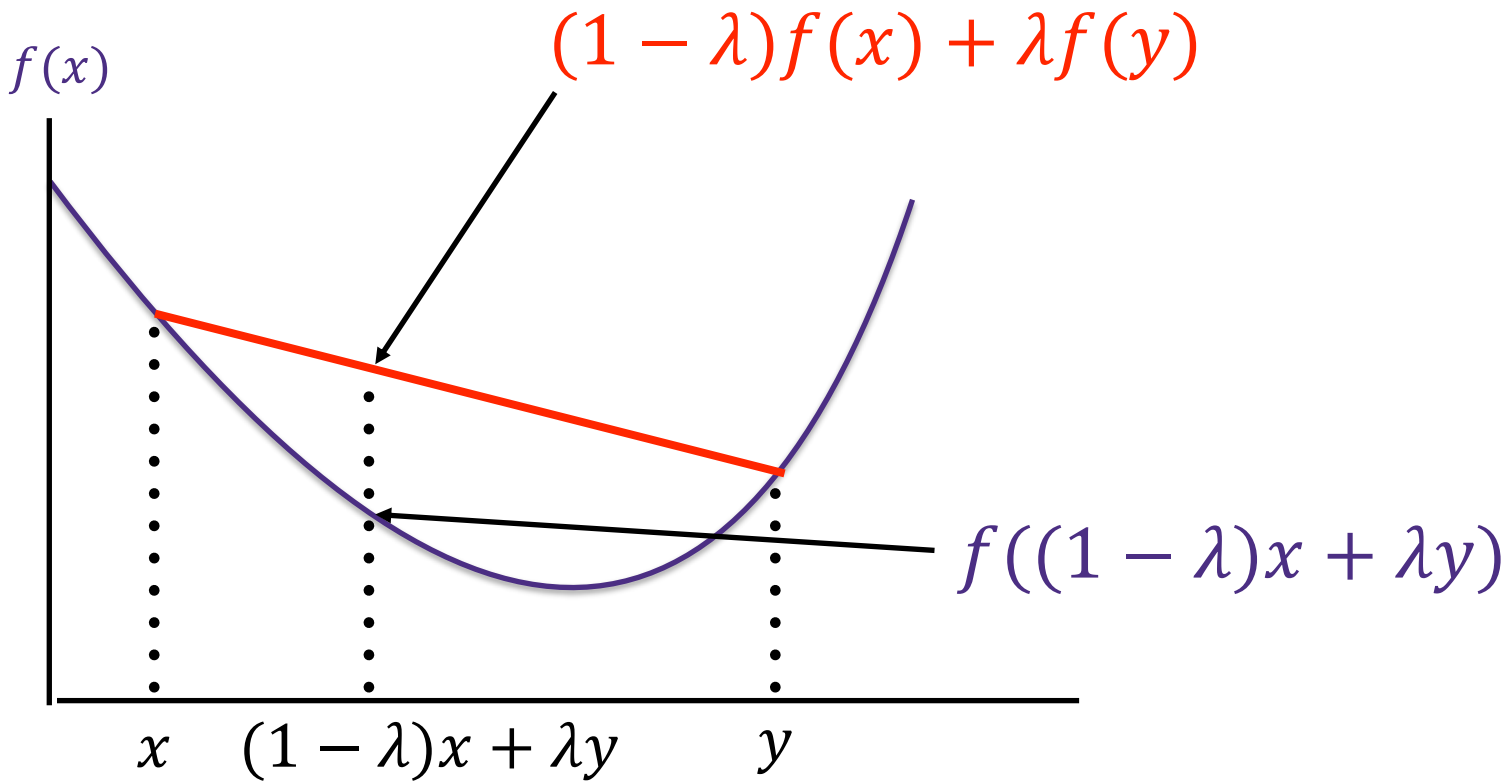
A set $K \subset \mathbb{R}^d$ is convex if $(1-\lambda)x + \lambda y \in K$ for all $x, y \in K$ and $\lambda \in [0,1]$



What is a convex function?

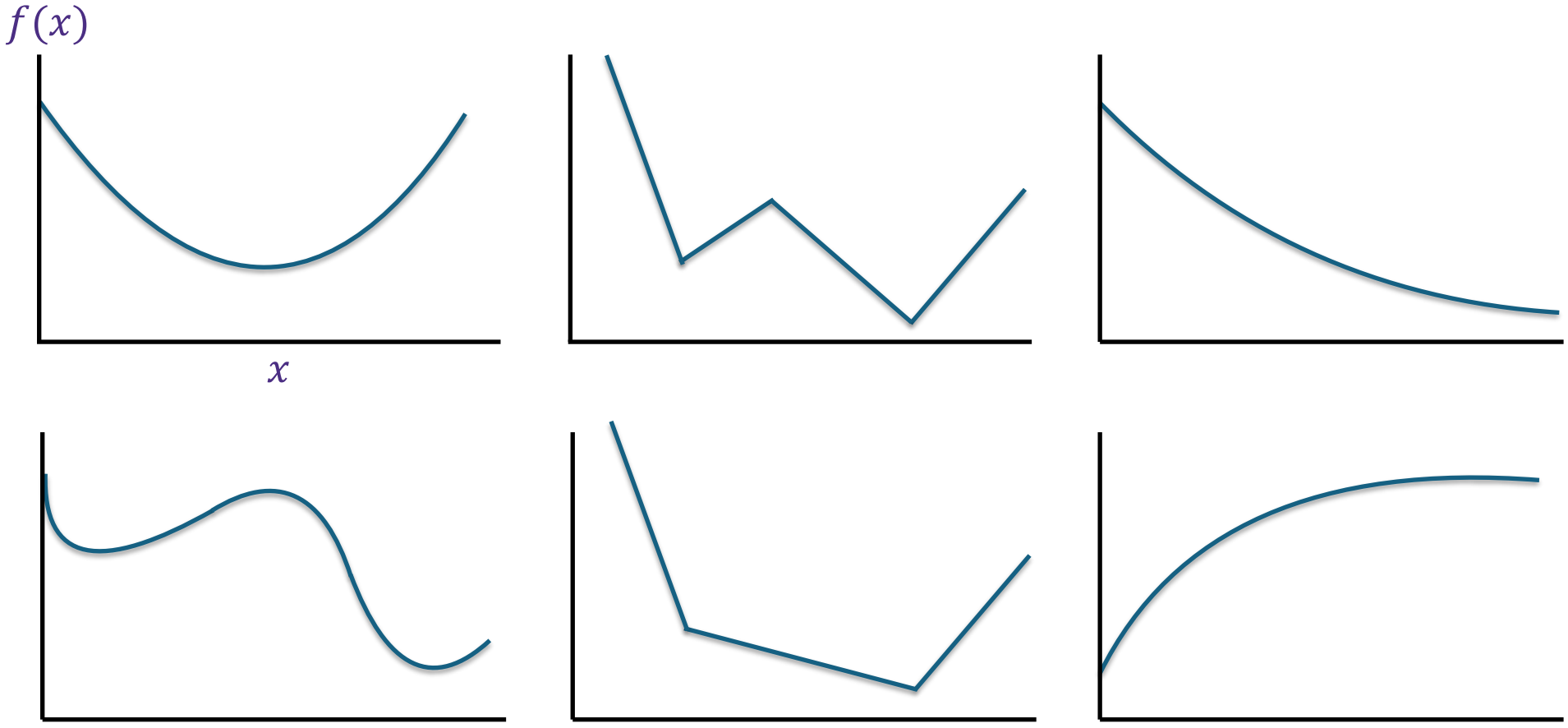
What is a convex function?

A function $f : \mathbb{R}^d \to \mathbb{R}$ is convex if $f((1-\lambda)x + \lambda y) \leq (1-\lambda)f(x) + \lambda f(y)$ for all $x, y \in \mathbb{R}^d$ and $\lambda \in [0, 1]$



What is a convex function?

A function $f : \mathbb{R}^d \to \mathbb{R}$ is convex if $f((1-\lambda)x + \lambda y) \leq (1-\lambda)f(x) + \lambda f(y)$ for all $x, y \in \mathbb{R}^d$ and $\lambda \in [0, 1]$



Convex functions and convex sets

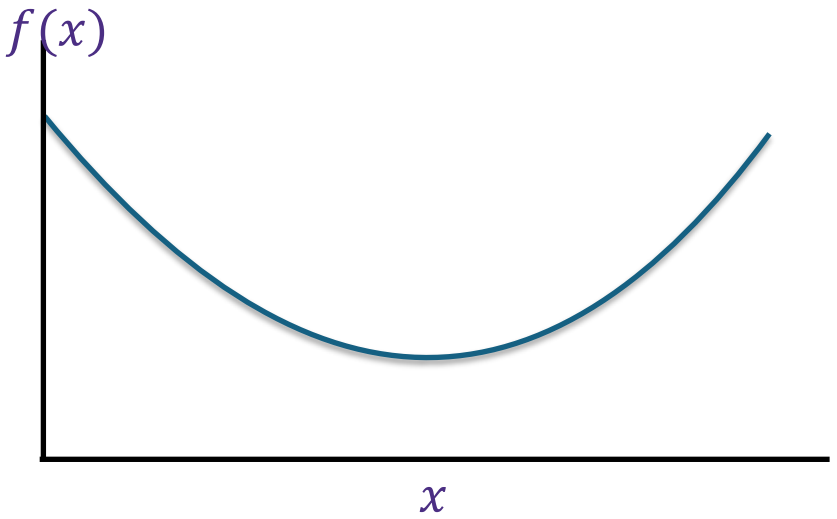
Convex functions and convex sets

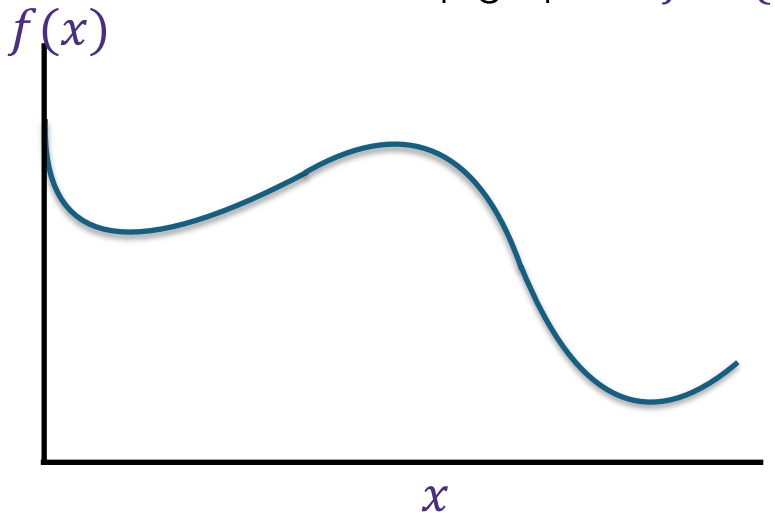
A set $K \subset \mathbb{R}^d$ is convex if $(1-\lambda)x + \lambda y \in K$ for all $x, y \in K$ and $\lambda \in [0,1]$

A function $f : \mathbb{R}^d \to \mathbb{R}$ is convex if $f((1-\lambda)x + \lambda y) \leq (1-\lambda)f(x) + \lambda f(y)$ for all $x, y \in \mathbb{R}^d$ and $\lambda \in [0,1]$

A function $f : \mathbb{R}^d \to \mathbb{R}$ is convex if the set $\{(x,t) \in \mathbb{R}^{d+1} : f(x) \leq t\}$ is convex

Graph of f is $\{(x,t): f(x) = t\}$
Epigraph of f is $\{(x,t): f(x) \leq t\}$



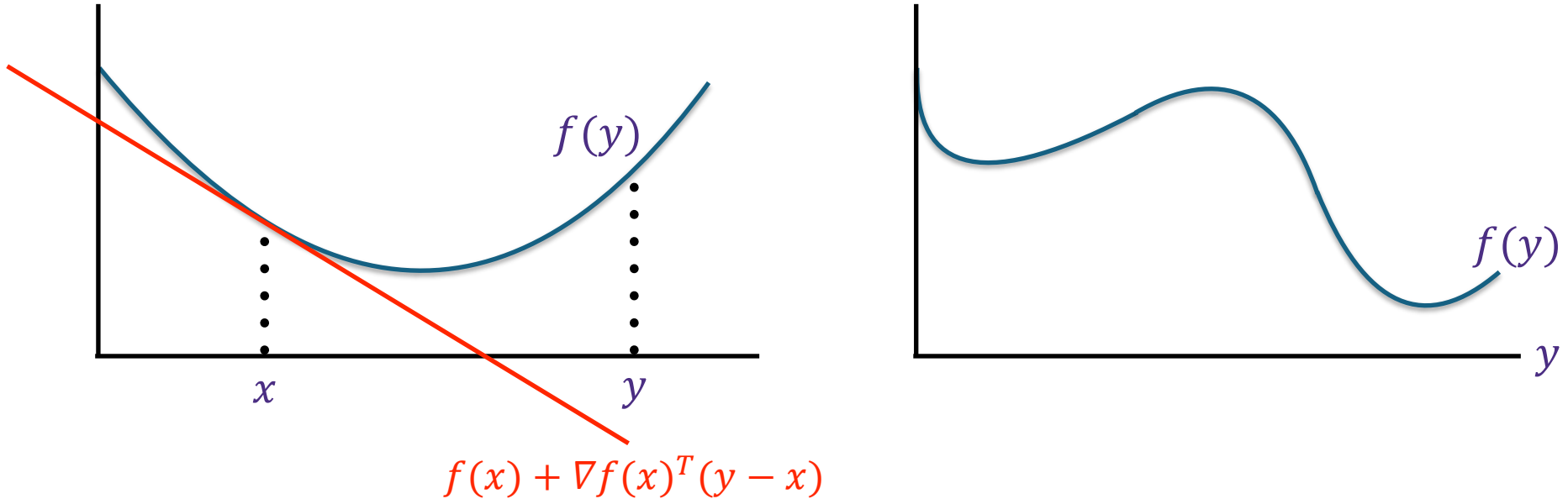


Convexity of differentiable functions

Convexity of differentiable functions

A function $f : \mathbb{R}^d \to \mathbb{R}$ is convex if the set $\{(x, t) \in \mathbb{R}^{d+1} : f(x) \leq t\}$ is convex

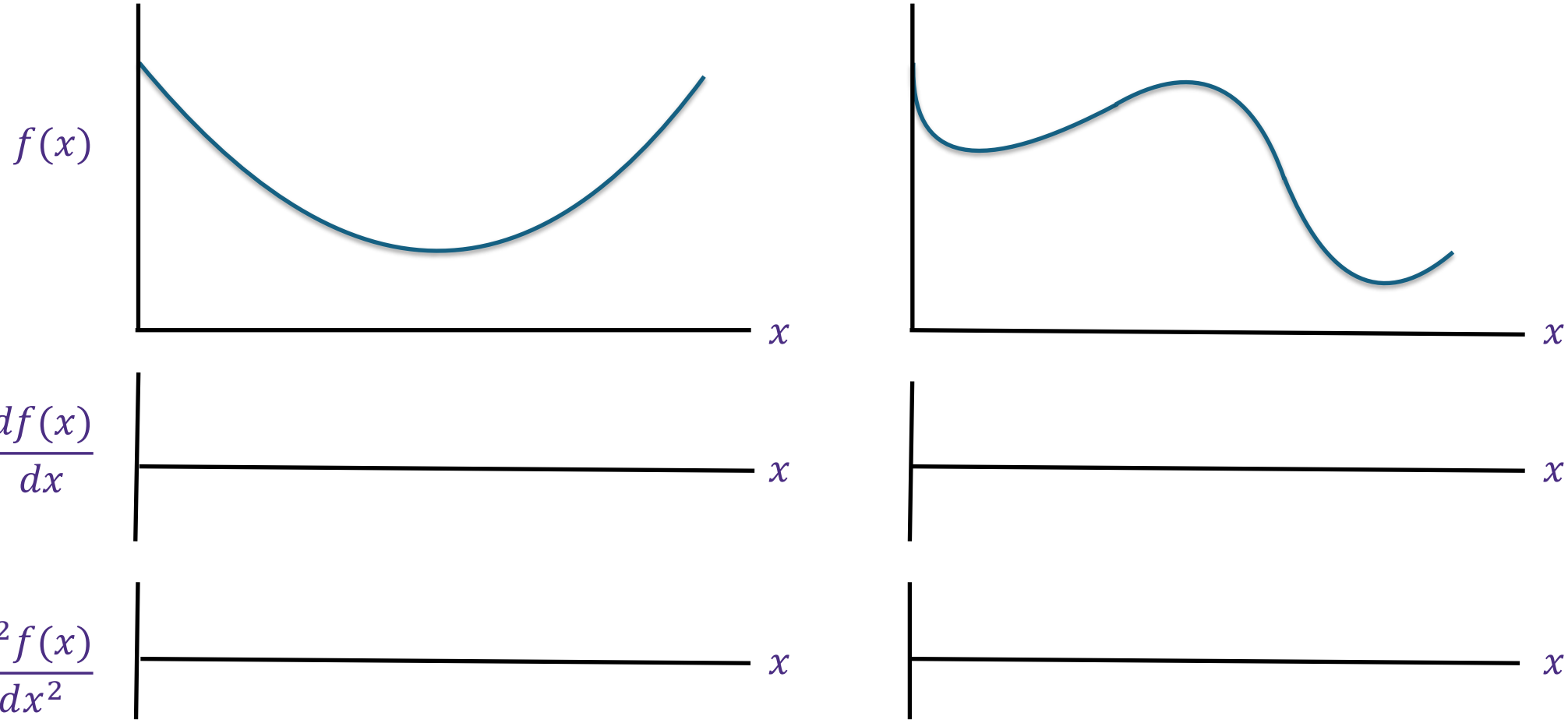
A function $f : \mathbb{R}^d \to \mathbb{R}$ that is differentiable everywhere is convex if $f(y) \geq f(x) + \nabla f(x)^\top (y - x)$ for all $x, y \in dom(f)$



Convexity of twice-differentiable functions

Convexity of twice-differentiable functions

A function $f : \mathbb{R}^d \to \mathbb{R}$ that is twice-differentiable everywhere is convex if $\nabla^2 f(x) \succeq 0$ for all $x \in dom(f)$



Convexity of twice-differentiable functions

A function $f : \mathbb{R}^d \to \mathbb{R}$ that is twice-differentiable everywhere is convex if $\nabla^2 f(x) \succeq 0$ for all $x \in dom(f)$

Recap: Definitions of convexity

A set $K \subset \mathbb{R}^d$ is convex if $(1-\lambda)x + \lambda y \in K$ for all $x, y \in K$ and $\lambda \in [0,1]$

A function $f : \mathbb{R}^d \to \mathbb{R}$ is convex if $f((1-\lambda)x + \lambda y) \leq (1-\lambda)f(x) + \lambda f(y)$ for all $x, y \in \mathbb{R}^d$ and $\lambda \in [0,1]$

A function $f : \mathbb{R}^d \to \mathbb{R}$ is convex if the set $\{(x,t) \in \mathbb{R}^{d+1} : f(x) \leq t\}$ is convex

A function $f : \mathbb{R}^d \to \mathbb{R}$ that is differentiable everywhere is convex if $f(y) \geq f(x) + \nabla f(x)^\top (y - x)$ for all $x, y \in dom(f)$

A function $f : \mathbb{R}^d \to \mathbb{R}$ that is twice-differentiable everywhere is convex if $\nabla^2 f(x) \succeq 0$ for all $x \in dom(f)$

Example: Ridge regression

$$\underset{w}{\operatorname{argmin}} \|y - Xw\|_2^2 + \lambda \|w\|_2^2$$

$$\underset{\|w\|_2^2 \leq \rho}{\operatorname{argmin}} \|y - Xw\|_2^2$$

Example: Lasso

$$\underset{w}{\operatorname{argmin}} \|y - Xw\|_2^2 + \lambda \|w\|_1$$

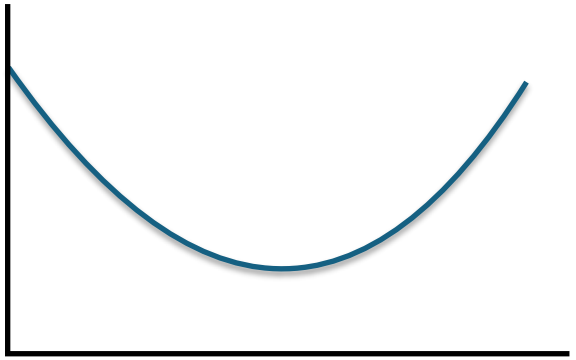
Why not directly solve $\underset{w}{\operatorname{argmin}} \|y - Xw\|_2^2 + \lambda \operatorname{card}(w)$?

Can interpret lasso as convex relaxation of cardinality objective

Convexity and gradient descent

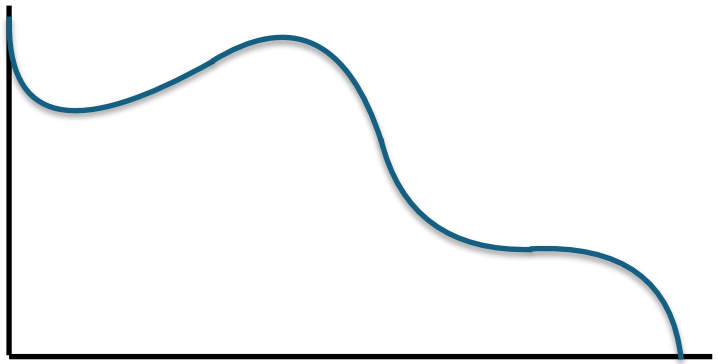
- All local minima are global minima

Convex function



Stationary points with $\nabla f(x) = 0$
are global minima

Non-convex function



Stationary points with $\nabla f(x) = 0$
could be a local minima,
a local maxima, or a saddle point

- Won't get stuck navigating the parameter constraints

Convexity and gradient descent

- Convexity => all local minima are global minima
- All local minima are global minima => ?

Convexity and gradient descent

- You can always run gradient descent whether $f(w)$ is convex or not!
- But if $f(w)$ is convex, we have guarantees on converging to the global minimum
- Linear regression, ridge regression, Lasso → all convex!

Lecture plan

- Gradient descent algorithm + examples

- Theoretical analysis
  - When does it work?
  - How quickly does it converge? ← we are here
  - How do we choose a step size?
  - Key idea: Convexity

- Not tested on proof details, but concepts are important & practical

Convergence analysis steps

1. Study single iteration: $f(w_{t+1})$ vs. $f(w_t)$
2. Piece iterations together to study how they converge

Single-iteration progress bound

Don't need convexity (yet).

Assume f is C^2, and gradient of f is Lipschitz continuous.

There exists L such that:

For all u, v, $\quad \|\nabla f(u) - \nabla f(v)\| \leq L\|u - v\|$.

For all w, $\quad \nabla^2 f(w) \preccurlyeq LI$.

For any u, v, $\quad v^T \nabla^2 f(u) v \leq L\|v\|^2$.

Single-iteration progress bound

For any $u, v,$ $\quad v^T \nabla^2 f(u) v \leq L\|v\|^2$.

Take Taylor expansion:

Single-iteration progress bound

$$w_{t+1} = w_t - \frac{1}{\mathrm{L}} \nabla f(w_t)$$

$$f(w_{t+1}) \leq f(w_t) + \nabla f(w_t)^{\top}(w_{t+1} - w_t) + \frac{L}{2}\|w_{t+1} - w_t\|_2^2$$

Single-iteration progress bound

When $\eta = \frac{1}{L}$,

$$f(w_{t+1}) \leq f(w_t) - \frac{1}{2L} \|\nabla f(w_t)\|_2^2$$

Same argument shows any $\eta < \frac{2}{L}$ will decrease f.

Convergence analysis steps

1. Study single iteration: $f(w_{t+1})$ vs. $f(w_t)$
2. Piece iterations together to study how they converge

Convergence rate of gradient descent

$$\|\nabla f(w_t)\| \to 0$$

$$f(w_t) - f(w^*) \to 0$$

Convergence rate of gradient descent

For some ϵ, how many iterations before $\|\nabla f(w_t)\|^2 \leq \epsilon$?

Assumptions:

- Gradient is Lipschitz continuous (as before)
- Step size is small enough (assume $1/L$)
- f is bounded below by $f(w^*)$

Proof sketch:

- Each iteration decreases f by at least $\frac{1}{2L}\|\nabla f(w_t)\|^2$
- Can't decrease below $f(w^*)$
- So $\|\nabla f(w_t)\|^2$ must be decaying fast enough

Convergence rate of gradient descent

$$f(w_{t+1}) \leq f(w_t) - \frac{1}{2L} \|\nabla f(w_t)\|_2^2$$

Convergence rate of gradient descent

$$T \geq \frac{2L\big(f(w_0) - f(w^*)\big)}{\epsilon}$$

Gradient descent requires

$T = O(1/\epsilon)$ iterations

to achieve $\|\nabla f(w_t)\|^2 \leq \epsilon$

Convergence rate of gradient descent

$$\|\nabla f(w_t)\| \leq \epsilon$$

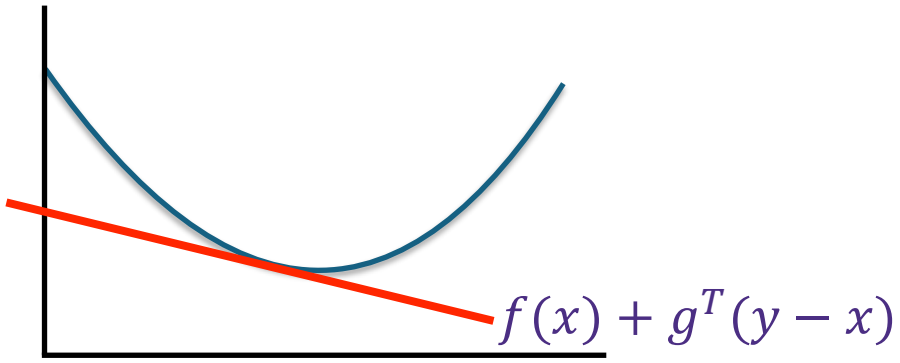
$$f(w_t) - f(w^*) \leq \epsilon$$

Lasso revisited

Subgradients

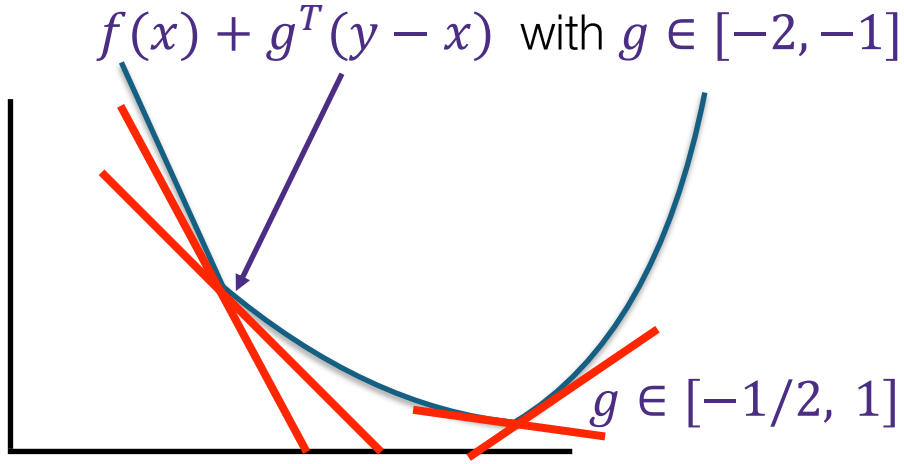
A vector $g \in \mathbb{R}^d$ is a **subgradient** at x if it satisfies
$f(y) \geq f(x) + g^T(y - x)$ for all $y \in \mathbb{R}^d$

Smooth convex function



Gradient is unique sub-gradient
Minimum at points where gradient is 0

Non-smooth convex function

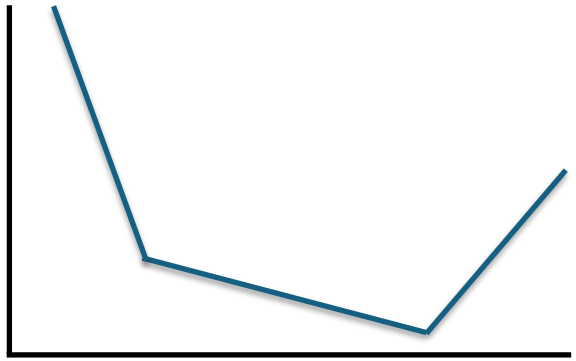


Minimum achieved at points where
subgradient set includes 0 vector

Subgradient descent for non-smooth functions

For each t,

Find any subgradient g_t, then set $w_{t+1} \leftarrow w_t - \eta_t g_t$



Works on non-smooth convex functions
Slower compared to smooth convex functions
Gradients don't get smaller near global minima

- Instead of last iterate w_t, keep track of best one
- Step size needs to decrease with t

Stochastic gradient descent (SGD)

$$\widehat{w} = \operatorname*{argmin}_{w} \frac{1}{n} \sum_{i=1}^{n} \ell_i(w)$$

Gradient descent: $w_{t+1} = w_t - \eta \nabla_w \left(\frac{1}{n} \sum_{i=1}^{n} \ell_i(w) \right) |_{w=w_t}$

Stochastic gradient descent: $w_{t+1} = w_t - \eta \nabla_w \ell_{I_t}(w) |_{w=w_t}$

I_t drawn uniformly at random from $\{1, \ldots, n\}$

n times faster per iteration!

And can even be better minimizer.

Minibatch stochastic gradient descent

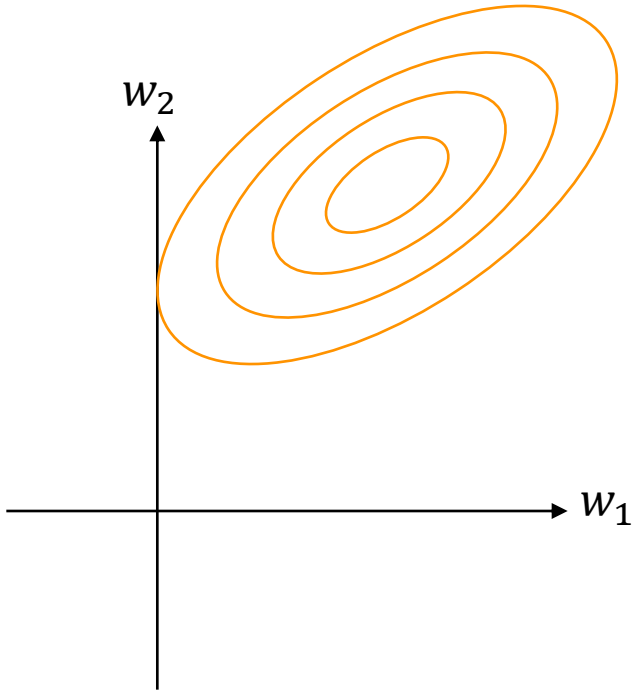
- Instead of one iterate, average B stochastic gradients together
- Advantages:
  - Smaller variance (by $1/B$)
  - Parallelization: Each gradient in the minibatch can be computed in parallel

- This is very widely used!

Summary

- Closed form -> iterative methods
- (Minibatch stochastic) gradient descent as a general-purpose optimizer
- Key theoretical tool: Convexity
- Many many variants. Highly active research area!
  - Schedulers
  - Adaptive step sizes
  - Momentum
  - Higher-order methods
  - Non-convex analysis
  - …

Bonus: Coordinate descent



Coordinate descent for lasso

$$\hat{w} = \underset{w}{\operatorname{argmin}} \ \frac{1}{2}\sum_{i=1}^{n}\left(y_i - x_i^\top w\right)^2 + \lambda\|w\|_1$$

$$\frac{d}{dw_k} f(w)$$
$$= \sum_{i=1}^{n}\left(x_i^\top w - y_i\right)x_{ik} + \lambda \operatorname{sign}(w_k)$$
$$= \sum_{i=1}^{n}\left(\sum_{j\neq k} x_{ij} w_j + x_{ik} w_k - y_i\right)x_{ik} + \lambda \operatorname{sign}(w_k)$$
$$= \sum_{i=1}^{n}\left(\sum_{j\neq k} x_{ij} w_j - y_i\right)x_{ik} + w_k \sum_{i=1}^{n} x_{ik} + \lambda \operatorname{sign}(w_k) \ni 0$$
...

Further reading

- Example gradient descent code on class website
- Boyd and Vandenberghe, Convex Optimization https://stanford.edu/~boyd/cvxbook/
- Mark Schmidt's CPSC 540 notes: https://www.cs.ubc.ca/~schmidtm/Courses/540-W18/L4.pdf
- 3Blue1Brown