10-701: Introduction to Machine Learning Lecture 4 – Linear Regression

Henry Chai & Zack Lipton

9/11/23

Front Matter

- Announcements:
  - HW1 released 9/6, due 9/20 at 11:59 PM
- Recommended Readings:
  - Bishop, Section 3.2
  - Murphy, Sections 7.1-7.3

Recall: Regression

- Learning to diagnose heart disease

  as a **(supervised)** **<u>regression</u> task**

features / targets / data points

x_1 Family History	x_2 Resting Blood Pressure	x_3 Cholesterol	y Heart Disease?
Yes	Low	Normal	\$0
No	Medium	Normal	\$20
No	Low	Abnormal	\$30
Yes	Medium	Normal	\$100
Yes	High	Abnormal	\$5000

Decision Tree Regression

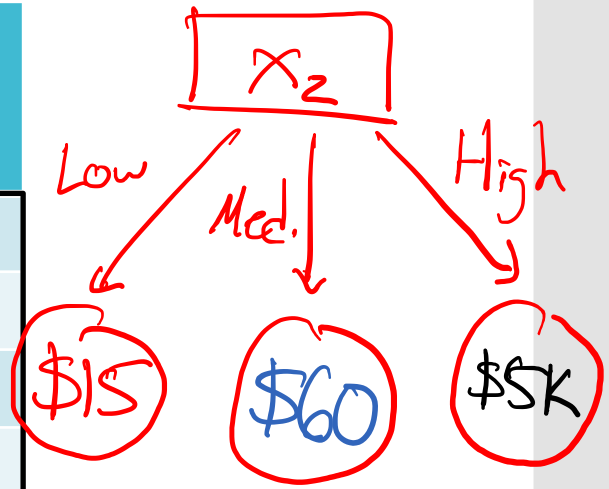
- Learning to diagnose heart disease

  as a **(supervised)** **<u>regression</u> task**

features | targets

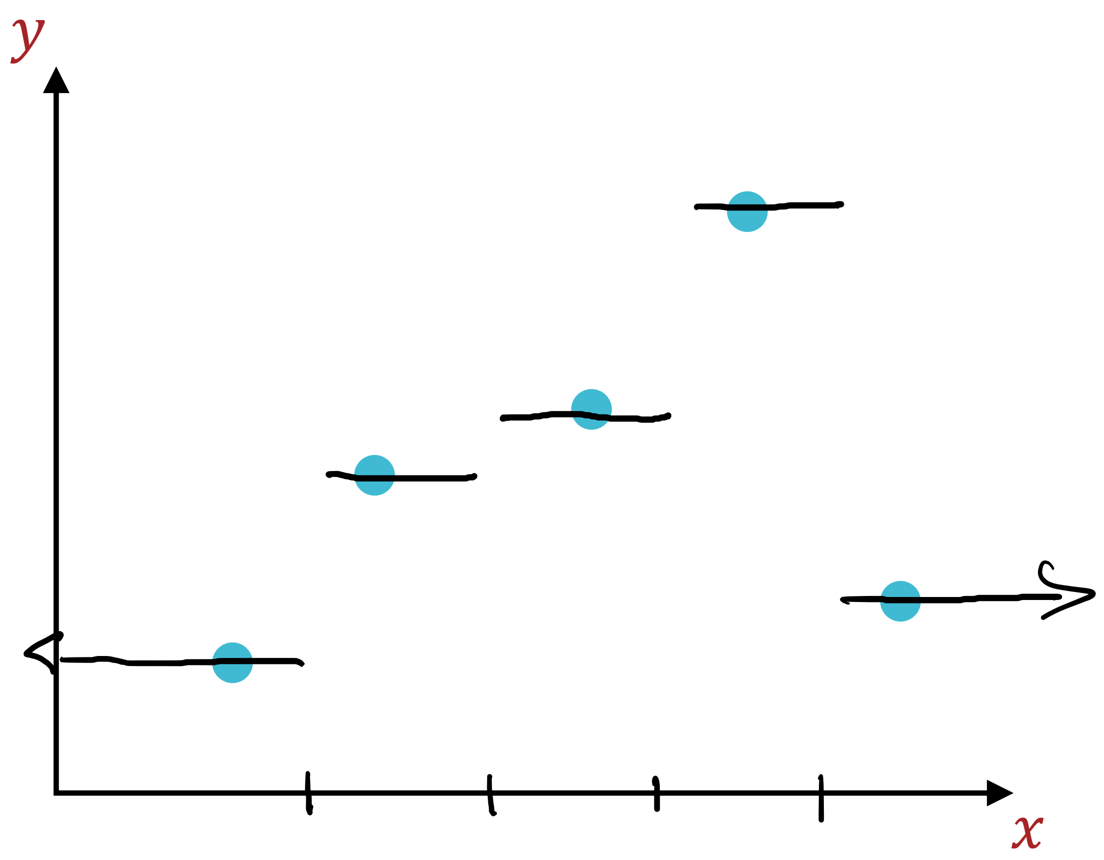
data points

x_1 Family History	x_2 Resting Blood Pressure	x_3 Cholesterol	y Heart Disease?
Yes	Low	Normal	\$0
No	Medium	Normal	\$20
No	Low	Abnormal	\$30
Yes	Medium	Normal	\$100
Yes	High	Abnormal	\$5000



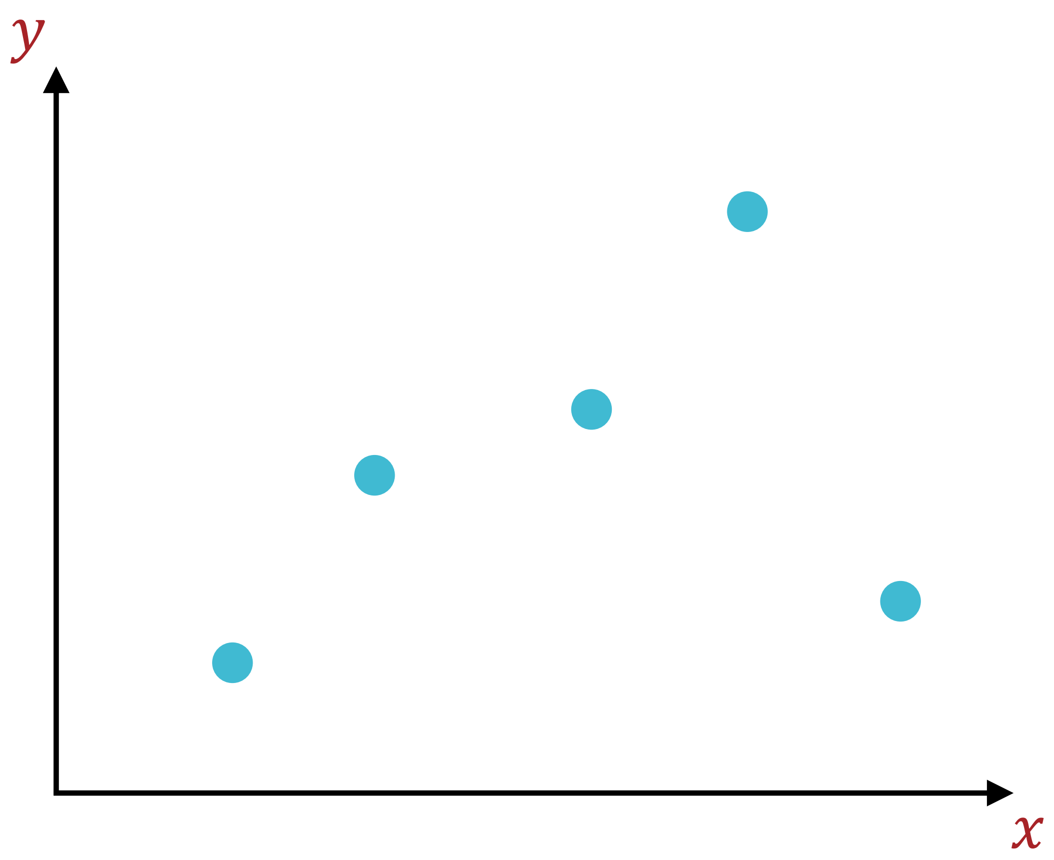
1-NN Regression

- Suppose we have real-valued targets $y \in \mathbb{R}$ and one-dimensional inputs $x \in \mathbb{R}$



2-NN Regression?

- Suppose we have real-valued targets $y \in \mathbb{R}$ and one-dimensional inputs $x \in \mathbb{R}$



Linear Regression

- Suppose we have real-valued targets $y \in \mathbb{R}$ and D-dimensional inputs $\boldsymbol{x} = [\quad x_1, \ldots, x_D]^T \in \mathbb{R}^D$
- **Assume**

$$y = \boldsymbol{w}^T \boldsymbol{x} + w_0$$

Linear Regression

- Suppose we have real-valued targets $y \in \mathbb{R}$ and D-dimensional inputs $\boldsymbol{x} = [1, x_1, \dots, x_D]^T \in \mathbb{R}^{D+1}$
- **Assume**

$w = [w_0, w_1, \dots, w_D]^T$

$$y = \boldsymbol{w}^T \boldsymbol{x}$$

Linear Regression

- Suppose we have real-valued targets $y \in \mathbb{R}$ and D-dimensional inputs $\boldsymbol{x} = [1, x_1, \ldots, x_D]^T \in \mathbb{R}^{D+1}$
- **Assume**

$$y = \boldsymbol{w}^T \boldsymbol{x}$$

- Notation: given training data $\mathcal{D} = \{(\boldsymbol{x}^{(n)}, y^{(n)})\}_{n=1}^{N}$
   - $$X = \begin{bmatrix} 1 & \boldsymbol{x}^{(1)^T} \\ 1 & \boldsymbol{x}^{(2)^T} \\ \vdots & \vdots \\ 1 & \boldsymbol{x}^{(N)^T} \end{bmatrix} = \begin{bmatrix} 1 & x_1^{(1)} & \cdots & x_D^{(1)} \\ 1 & x_1^{(2)} & \cdots & x_D^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(N)} & \cdots & x_D^{(N)} \end{bmatrix} \in \mathbb{R}^{N \times D+1}$$
   is the *design matrix*
   - $\boldsymbol{y} = \left[y^{(1)}, \ldots, y^{(N)}\right]^T \in \mathbb{R}^N$ is the *target vector*

General Recipe for Machine Learning

1. Define a model and model parameters

2. Write down an objective function

3. Optimize the objective w.r.t. the model parameters

Recipe for Linear Regression

1. Define a model and model parameters

   Assume $y = w^T x$

   → parameters $w = [w_0, w_1, \ldots, w_D]$

2. Write down an objective function

   Minimize squared loss

   $$\ell_D(w) = \frac{1}{N}\sum_{n=1}^{N}\left(y^{(n)} - w^T x^{(n)}\right)^2$$

3. Optimize the objective w.r.t. the model parameters

   1. Solve in closed-form: take derivates, set them equal to 0 and solve!

Minimizing the Squared Error

$$\ell_D(w) = \frac{1}{N}\sum_{n=1}^{N}\left(w^T x^{(n)} - y^{(n)}\right)^2 = \frac{1}{N}\sum_{n=1}^{N}\left(x^{(n)T} w - y^{(n)}\right)^2$$

$$Xw - y \in \mathbb{R}^N$$

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

$$= \frac{1}{N}\left(w^T X^T X w - 2w^T X^T y + y^T y\right)$$

gradient

$$\nabla_w \ell_D(w) = \frac{1}{N}\left(2X^T X w - 2X^T y\right)$$

$$\Rightarrow \frac{1}{N}\left(2X^T X \hat{w} - 2X^T y\right) = 0$$

$$\Rightarrow X^T X \hat{w} - X^T y = 0 \Rightarrow X^T X \hat{w} = X^T y$$

$$\Rightarrow \hat{w} = (X^T X)^{-1} X^T y$$

$$H_w \ell_D(w) = \frac{1}{N}\left(2X^T X\right)$$

$$\hat{\boldsymbol{w}} = (X^T X)^{-1} X^T \boldsymbol{y}$$

Closed Form Solution

1. Is $X^T X$ invertible?

2. If so, how computationally expensive is inverting $X^T X$?

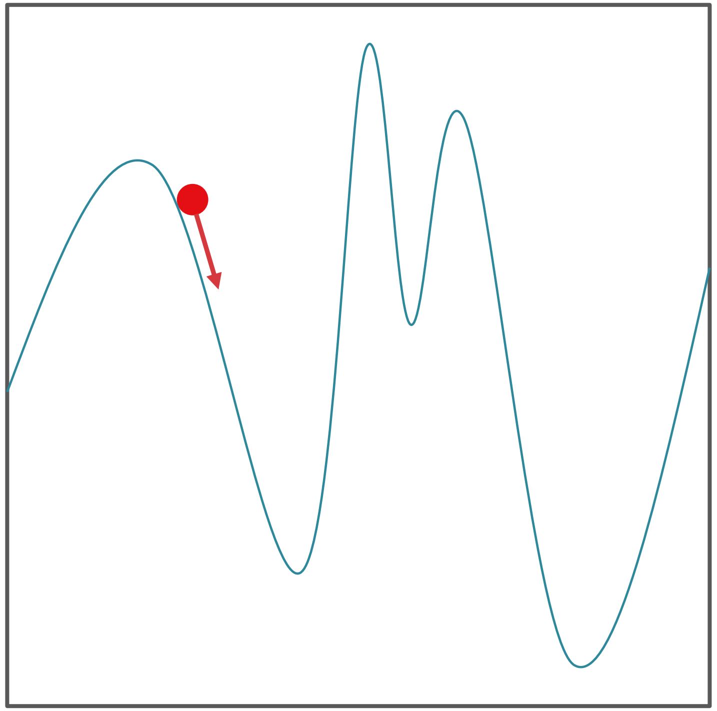
$X \in \mathbb{R}^{N \times (D+1)} \Rightarrow X^T X \in \mathbb{R}^{(D+1) \times (D+1)}$

classically inverting is $O(D^3)$ (but we can get $O(D^{2.373})$)

We need to store X, $O(ND)$

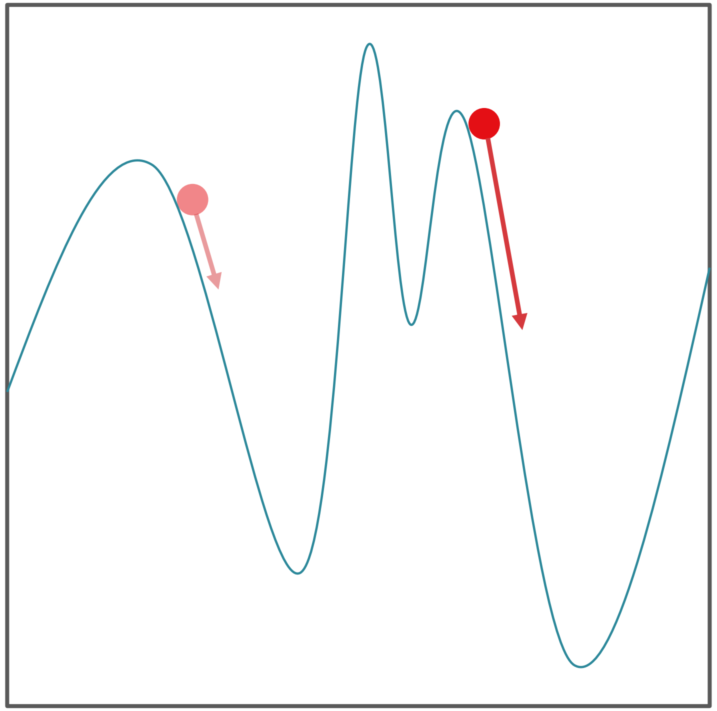
Gradient Descent: Intuition

- An iterative method for minimizing functions
- Requires the gradient to exist everywhere



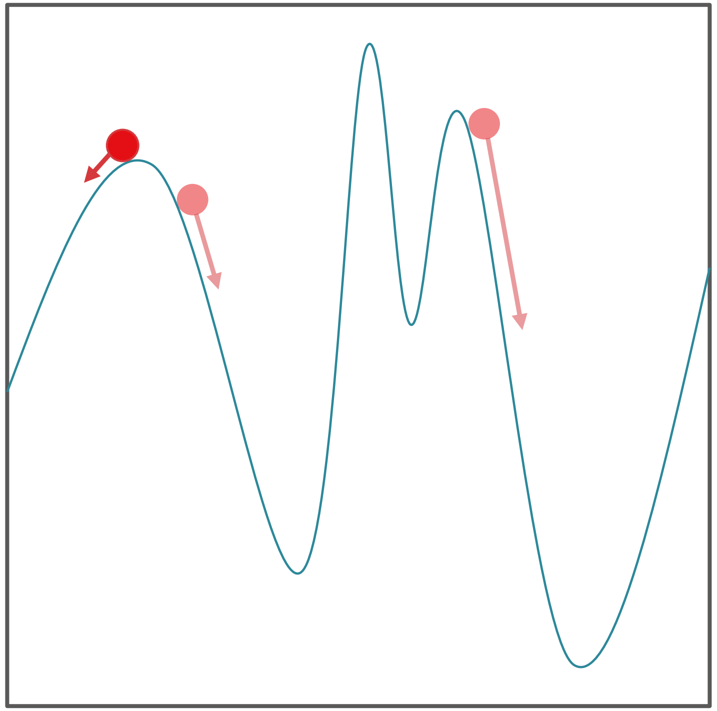
Gradient Descent: Intuition

- An iterative method for minimizing functions
- Requires the gradient to exist everywhere



Gradient Descent: Intuition

- An iterative method for minimizing functions
- Requires the gradient to exist everywhere



Gradient Descent

- Suppose the current weight vector is $\boldsymbol{w}^{(t)}$
- Move some distance, η, in the "most downhill" direction, $\widehat{\boldsymbol{v}}$:

$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} + \eta\widehat{\boldsymbol{v}}$$

Gradient Descent: Step Direction

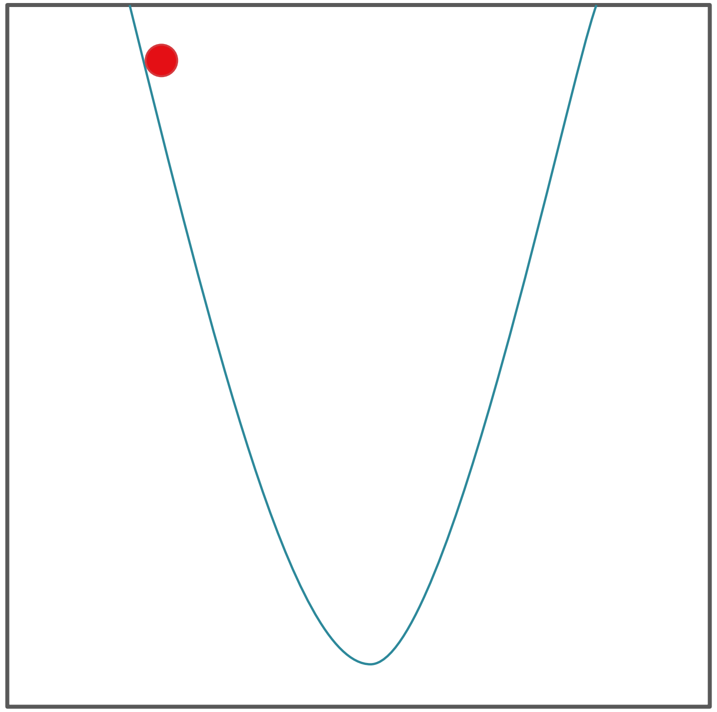
- Suppose the current weight vector is $\boldsymbol{w}^{(t)}$
- Move some distance, η, in the "most downhill" direction, $\hat{\boldsymbol{v}}$:

$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} + \eta\hat{\boldsymbol{v}}$$

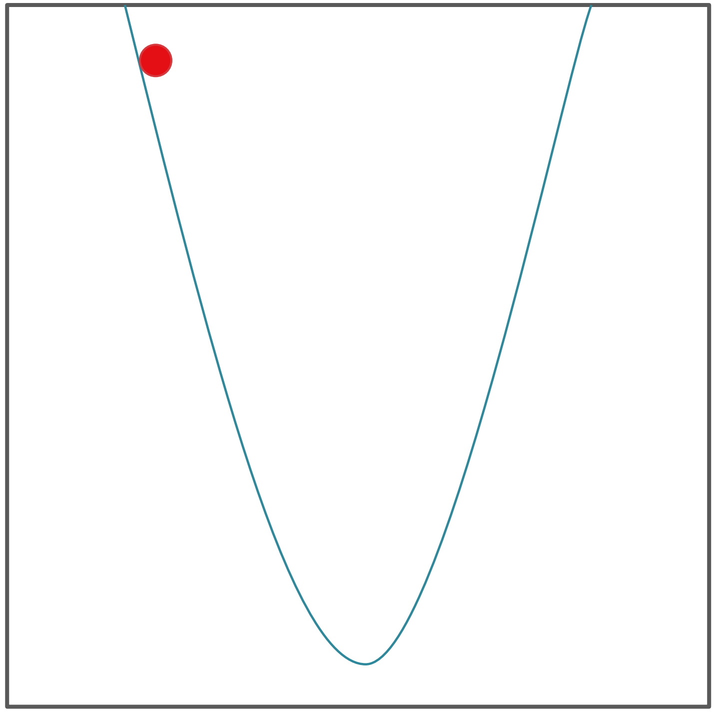
- The gradient points in the direction of steepest *increase* …
- … so $\hat{\boldsymbol{v}}$ should point in the opposite direction:

$$\hat{\boldsymbol{v}}^{(t)} = -\frac{\nabla_{\boldsymbol{w}}\ell_{\mathcal{D}}\left(\boldsymbol{w}^{(t)}\right)}{\left\|\nabla_{\boldsymbol{w}}\ell_{\mathcal{D}}\left(\boldsymbol{w}^{(t)}\right)\right\|}$$

Gradient Descent: Step Size

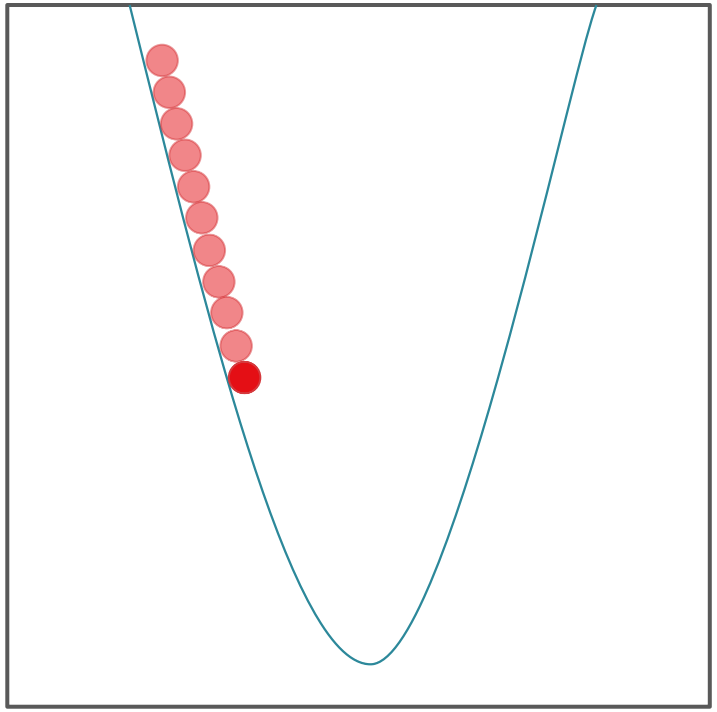


Small η

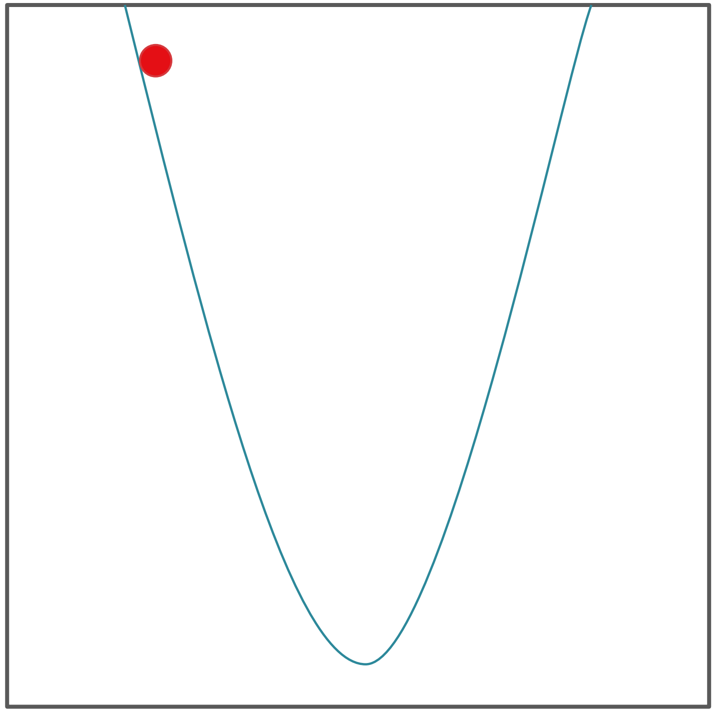


Large η

Gradient Descent: Step Size

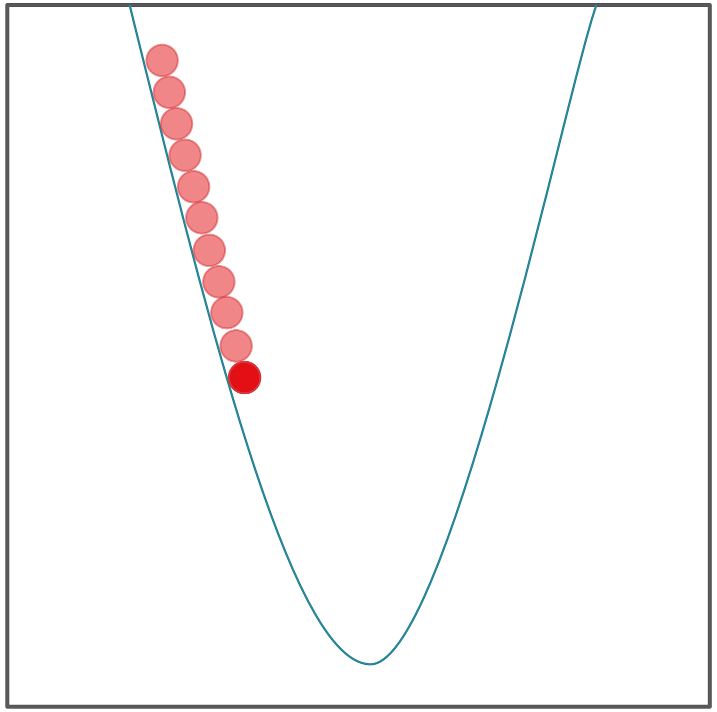


Small η

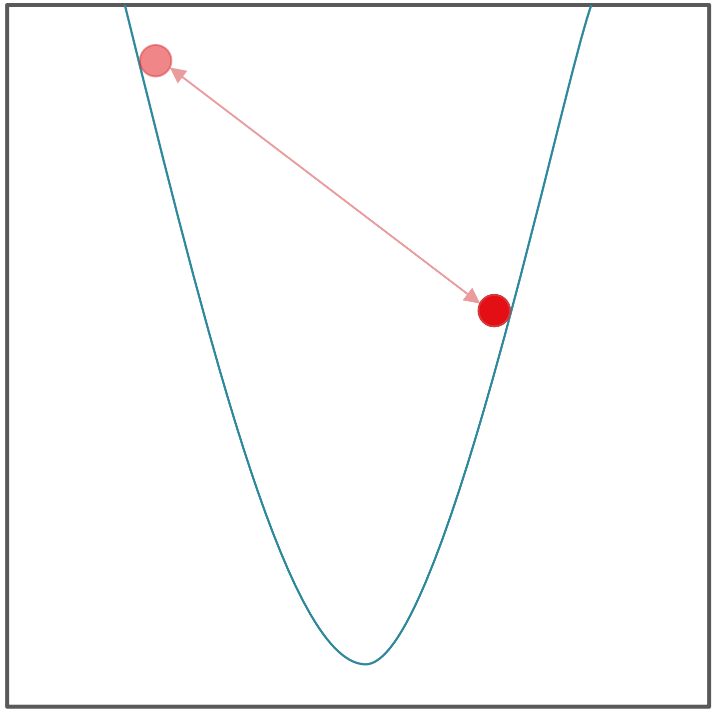


Large η

Gradient Descent: Step Size



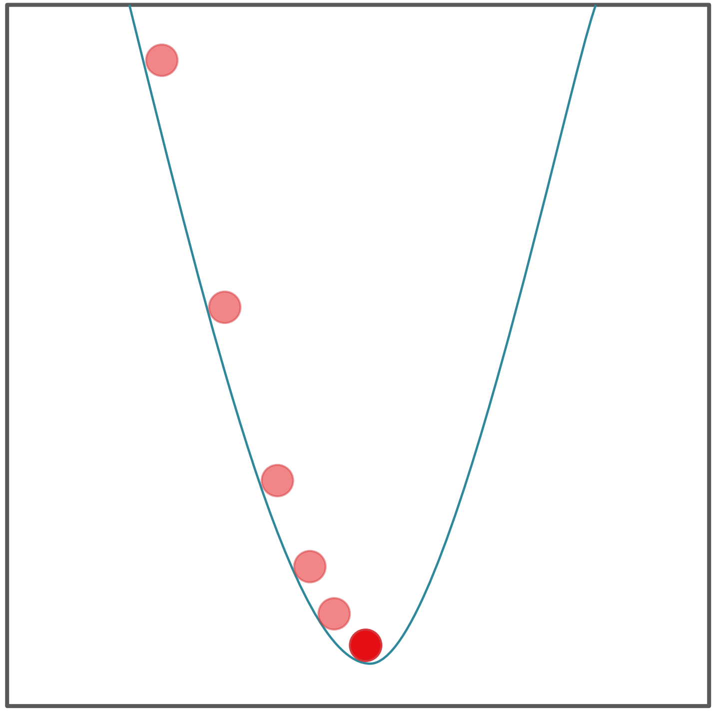
Small η



Large η

Gradient Descent: Step Size

- Use a variable $\eta^{(t)}$ instead of a fixed η!



- Set $\eta^{(t)} = \eta^{(0)} \left\| \nabla_{\boldsymbol{w}} \ell_{\mathcal{D}} \left(\boldsymbol{w}^{(t)} \right) \right\|$
- $\left\| \nabla_{\boldsymbol{w}} \ell_{\mathcal{D}} \left(\boldsymbol{w}^{(t)} \right) \right\|$ decreases as $\ell_{\mathcal{D}}$ approaches its minimum $\rightarrow \eta^{(t)}$ (hopefully) decreases over time

Gradient Descent

- $\hat{\boldsymbol{v}}^{(t)} = -\dfrac{\nabla_{\boldsymbol{w}} \ell_{\mathcal{D}}\left(\boldsymbol{w}^{(t)}\right)}{\left\|\nabla_{\boldsymbol{w}} \ell_{\mathcal{D}}\left(\boldsymbol{w}^{(t)}\right)\right\|}$

- $\eta^{(t)} = \eta^{(0)} \left\|\nabla_{\boldsymbol{w}} \ell_{\mathcal{D}}\left(\boldsymbol{w}^{(t)}\right)\right\|$

- $\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} + \eta^{(t)} \hat{\boldsymbol{v}}^{(t)}$

$$= w^{(t)} + \eta^{(0)} \left\|\nabla_w \ell_D\left(w^{(t)}\right)\right\| \frac{-\nabla_w \ell_D(w)}{\left\|\nabla_w \ell_D\left(w^{(t)}\right)\right\|}$$

$$= w^{(t)} - \eta^{(0)} \nabla_w \ell_D(w)$$

Gradient Descent

- Input: $\mathcal{D} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{N}, \eta^{(0)}$

1. Initialize $\boldsymbol{w}^{(0)}$ to all zeros and set $t = 0$
2. While TERMINATION CRITERION is not satisfied
   a. Compute the gradient:
      $\nabla_{\boldsymbol{w}} \ell_{\mathcal{D}}\left(\boldsymbol{w}^{(t)}\right)$
   b. Update $\boldsymbol{w}$: $\boldsymbol{w}^{(t+1)} \leftarrow \boldsymbol{w}^{(t)} - \eta^{(0)} \nabla_{\boldsymbol{w}} \ell_{\mathcal{D}}\left(\boldsymbol{w}^{(t)}\right)$
   c. Increment t: $t \leftarrow t + 1$

- Output: $\boldsymbol{w}^{(t)}$

Gradient Descent

- Input: $\mathcal{D} = \left\{\left(\boldsymbol{x}^{(i)}, y^{(i)}\right)\right\}_{i=1}^{N}, \eta^{(0)}, \epsilon$

1. Initialize $\boldsymbol{w}^{(0)}$ to all zeros and set $t = 0$
2. While $\left\|\nabla_{\boldsymbol{w}} \ell_{\mathcal{D}}\left(\boldsymbol{w}^{(t)}\right)\right\| > \epsilon$
   a. Compute the gradient: $\nabla_{\boldsymbol{w}} \ell_{\mathcal{D}}\left(\boldsymbol{w}^{(t)}\right)$
   b. Update $\boldsymbol{w}$: $\boldsymbol{w}^{(t+1)} \leftarrow \boldsymbol{w}^{(t)} - \eta^{(0)} \nabla_{\boldsymbol{w}} \ell_{\mathcal{D}}\left(\boldsymbol{w}^{(t)}\right)$
   c. Increment t: $t \leftarrow t + 1$

- Output: $\boldsymbol{w}^{(t)}$

Gradient Descent

- Input: $\mathcal{D} = \left\{\left(\boldsymbol{x}^{(i)}, y^{(i)}\right)\right\}_{i=1}^{N}, \eta^{(0)}, T$

1. Initialize $\boldsymbol{w}^{(0)}$ to all zeros and set $t = 0$
2. While $t < T$
   a. Compute the gradient:
      $\nabla_{\boldsymbol{w}} \ell_{\mathcal{D}}\left(\boldsymbol{w}^{(t)}\right)$
   b. Update $\boldsymbol{w}$: $\boldsymbol{w}^{(t+1)} \leftarrow \boldsymbol{w}^{(t)} - \eta^{(0)} \nabla_{\boldsymbol{w}} \ell_{\mathcal{D}}\left(\boldsymbol{w}^{(t)}\right)$
   c. Increment t: $t \leftarrow t + 1$

- Output: $\boldsymbol{w}^{(t)}$

Why Gradient Descent for linear regression?

- Input: $\mathcal{D} = \left\{\left(\boldsymbol{x}^{(i)}, y^{(i)}\right)\right\}_{i=1}^{N}, \eta^{(0)}, T$

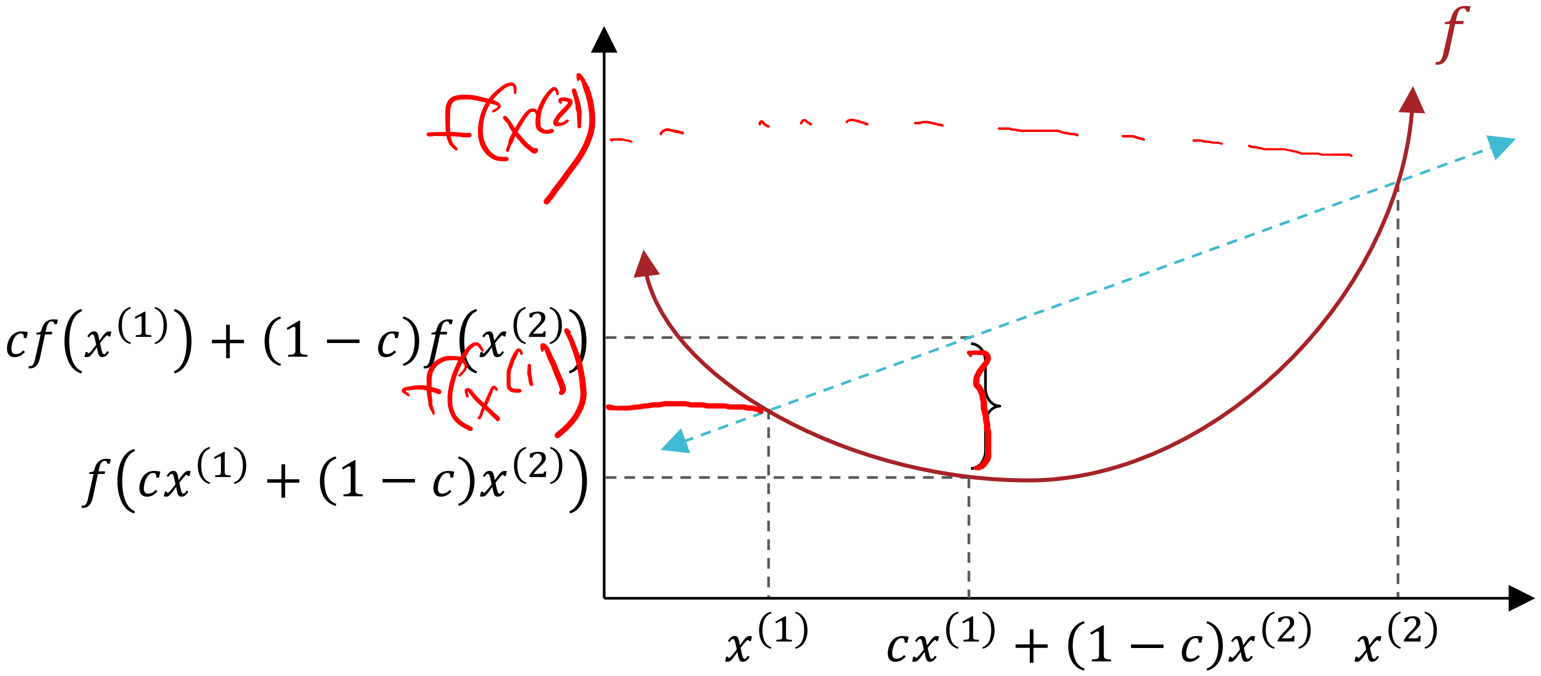
1. Initialize $\boldsymbol{w}^{(0)}$ to all zeros and set $t = 0$
2. While TERMINATION CRITERION is not satisfied
   a. Compute the gradient:
      $\nabla_{\boldsymbol{w}} \ell_{\mathcal{D}}\left(\boldsymbol{w}^{(t)}\right) = \frac{1}{N}\left(2X^T X w - 2X^T y\right)$
   b. Update $\boldsymbol{w}$: $\boldsymbol{w}^{(t+1)} \leftarrow \boldsymbol{w}^{(t)} - \eta^{(0)} \nabla_{\boldsymbol{w}} \ell_{\mathcal{D}}\left(\boldsymbol{w}^{(t)}\right)$
   c. Increment t: $t \leftarrow t + 1$

- Output: $\boldsymbol{w}^{(t)}$

Convexity

- A function $f: \mathbb{R}^D \rightarrow \mathbb{R}$ is convex if $\forall\, \boldsymbol{x}^{(1)} \in \mathbb{R}^D, \boldsymbol{x}^{(2)} \in \mathbb{R}^D$ and $0 \leq c \leq 1$

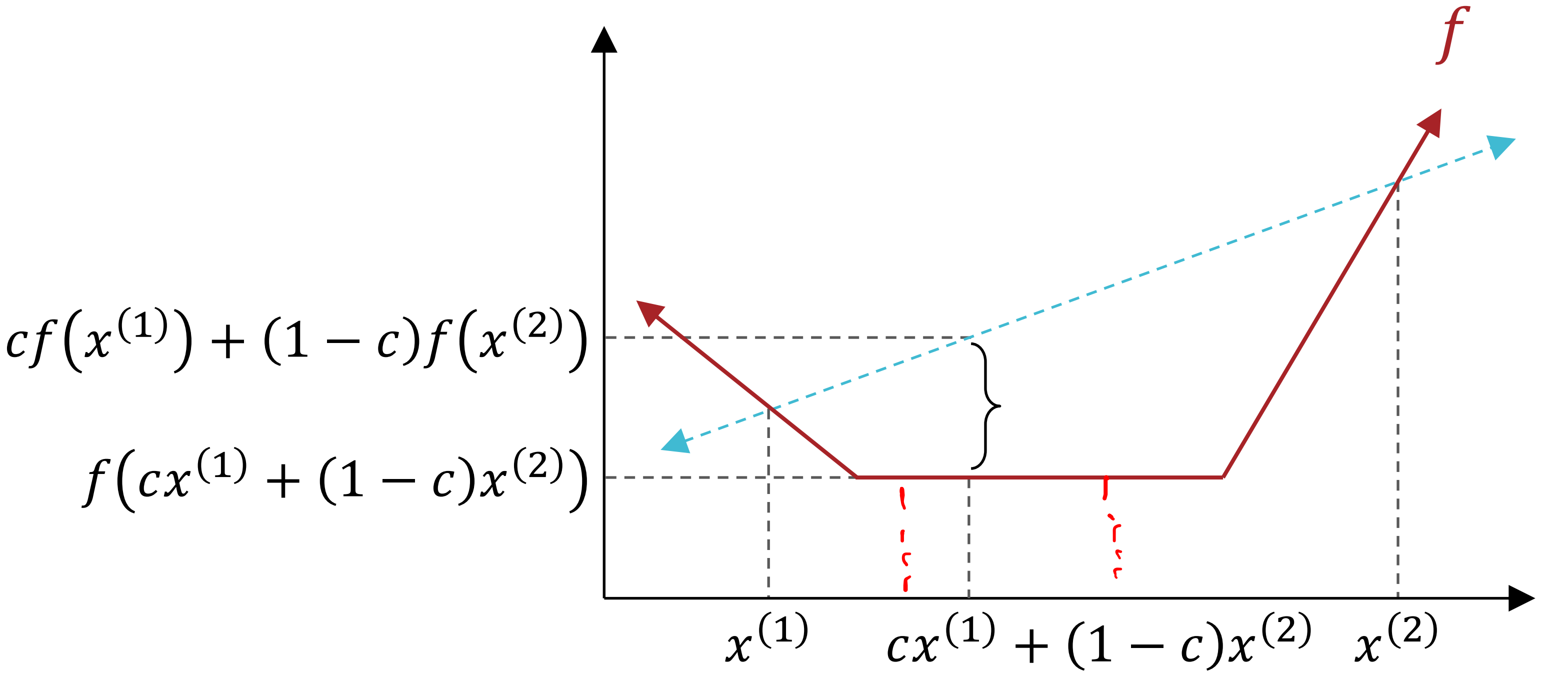
$$f\left(c\boldsymbol{x}^{(1)} + (1-c)\boldsymbol{x}^{(2)}\right) \leq cf\left(\boldsymbol{x}^{(1)}\right) + (1-c)f\left(\boldsymbol{x}^{(2)}\right)$$



Convexity

- A function $f: \mathbb{R}^D \rightarrow \mathbb{R}$ is convex if $\forall\, \boldsymbol{x}^{(1)} \in \mathbb{R}^D, \boldsymbol{x}^{(2)} \in \mathbb{R}^D$ and $0 \leq c \leq 1$

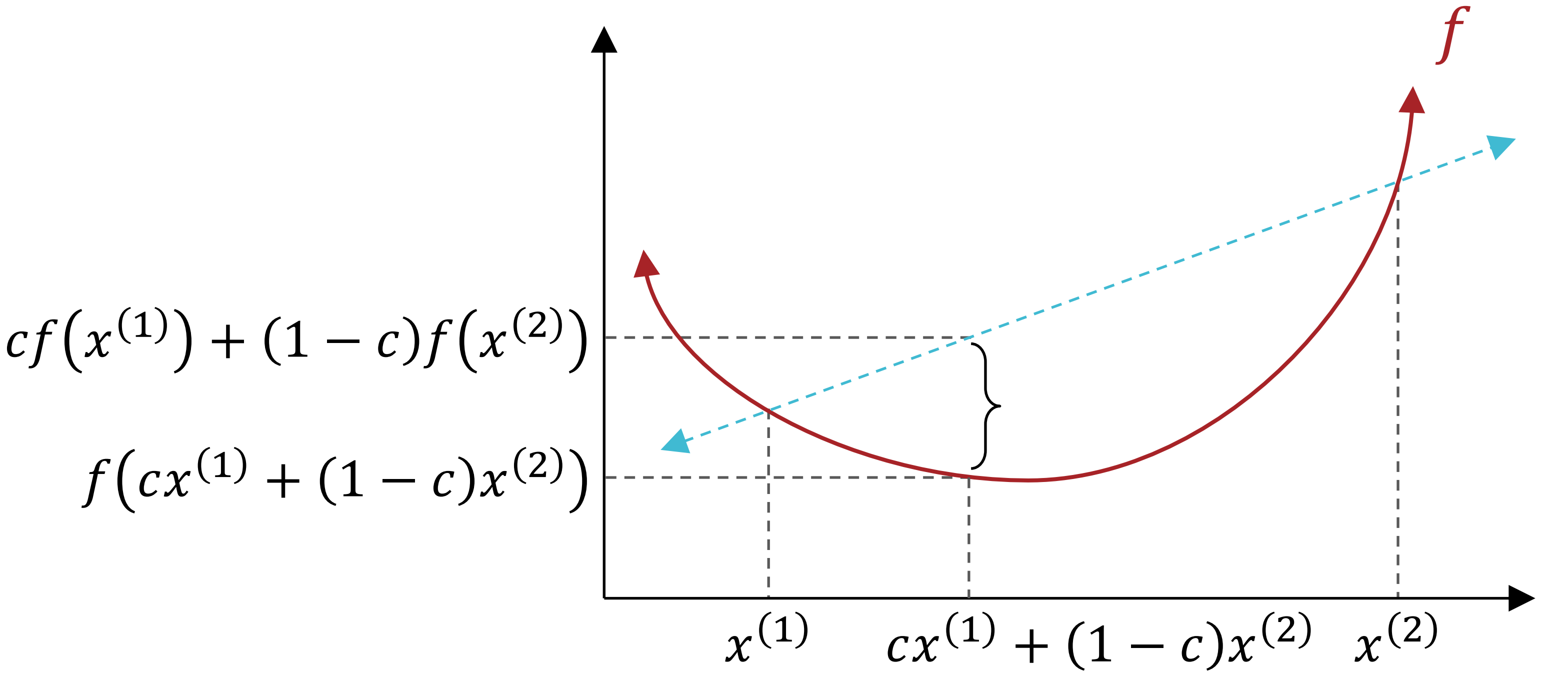
$$f\left(c\boldsymbol{x}^{(1)} + (1-c)\boldsymbol{x}^{(2)}\right) \leq cf\left(\boldsymbol{x}^{(1)}\right) + (1-c)f\left(\boldsymbol{x}^{(2)}\right)$$



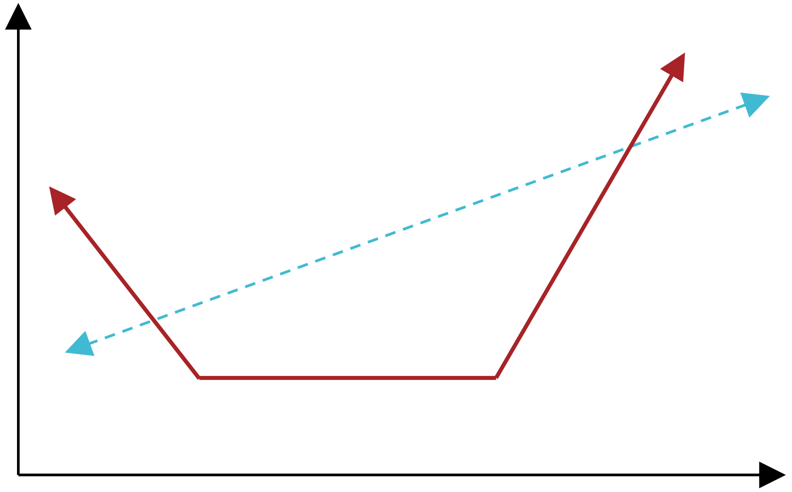
Convexity

- A function $f: \mathbb{R}^D \rightarrow \mathbb{R}$ is *strictly* convex if $\forall\, \boldsymbol{x}^{(1)} \in \mathbb{R}^D, \boldsymbol{x}^{(2)} \in \mathbb{R}^D$ and $0 < c < 1$

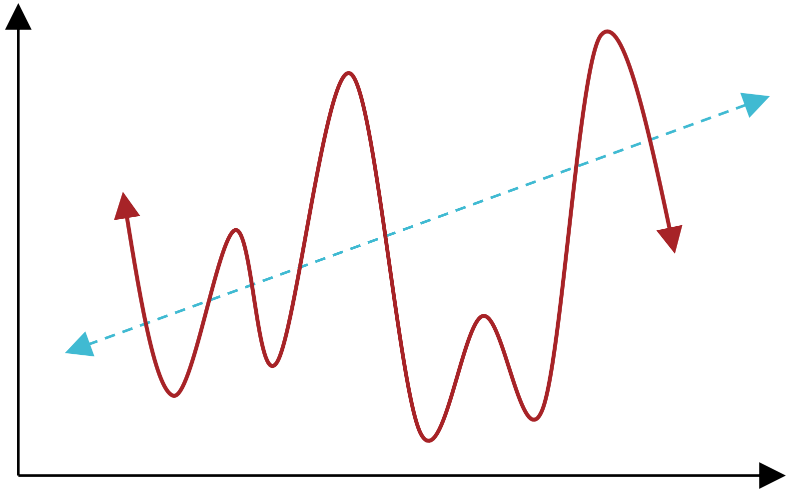
$$f\left(c\boldsymbol{x}^{(1)} + (1-c)\boldsymbol{x}^{(2)}\right) < cf\left(\boldsymbol{x}^{(1)}\right) + (1-c)f\left(\boldsymbol{x}^{(2)}\right)$$



Convexity

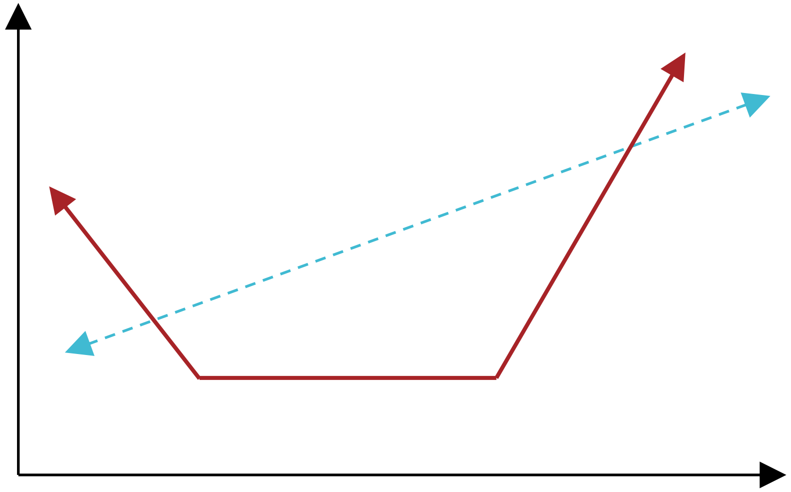


Convex functions



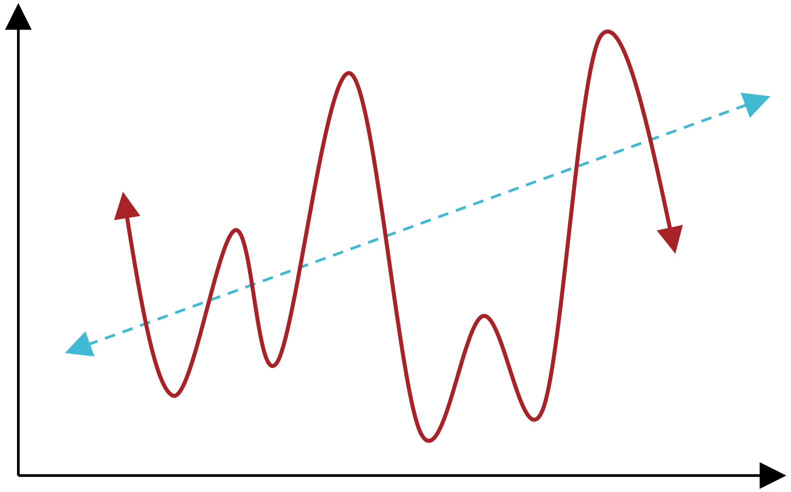
Non-convex functions

Convexity



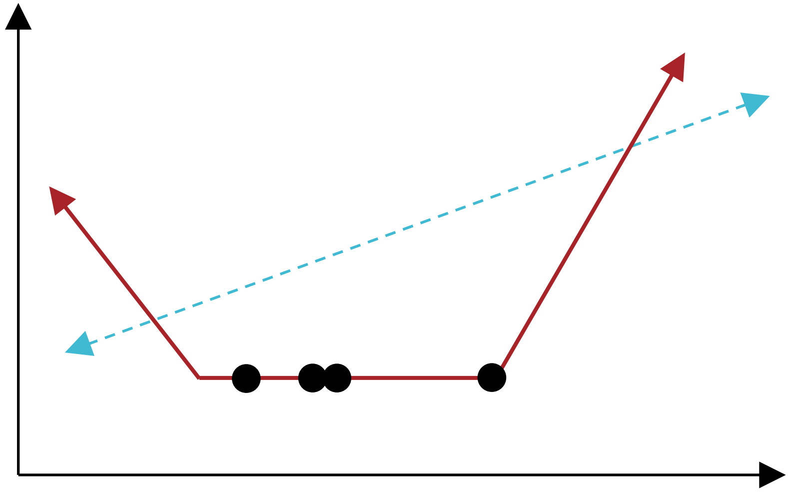
Given a function $f: \mathbb{R}^D \rightarrow \mathbb{R}$

- $\boldsymbol{x}^*$ is a *global* minimum iff $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})\ \forall\ \boldsymbol{x} \in \mathbb{R}^D$



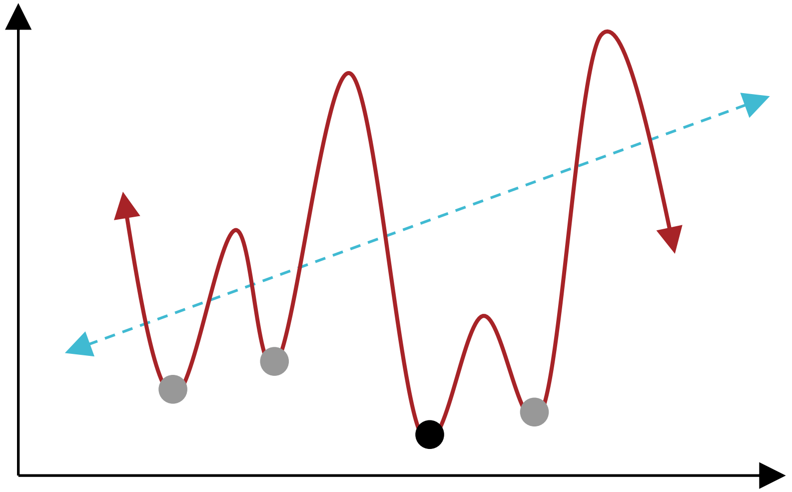
- $\boldsymbol{x}^*$ is a *local* minimum iff $\exists\ \epsilon$ s.t. $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})\ \forall$ $\boldsymbol{x}$ s.t. $\|\boldsymbol{x} - \boldsymbol{x}^*\|_2 < \epsilon$

Convexity



Convex functions:

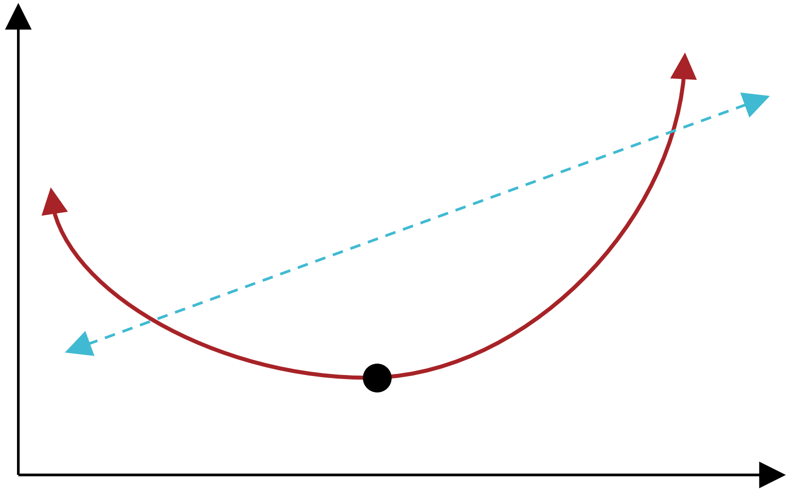
Each local minimum is a global minimum!



Non-convex functions:

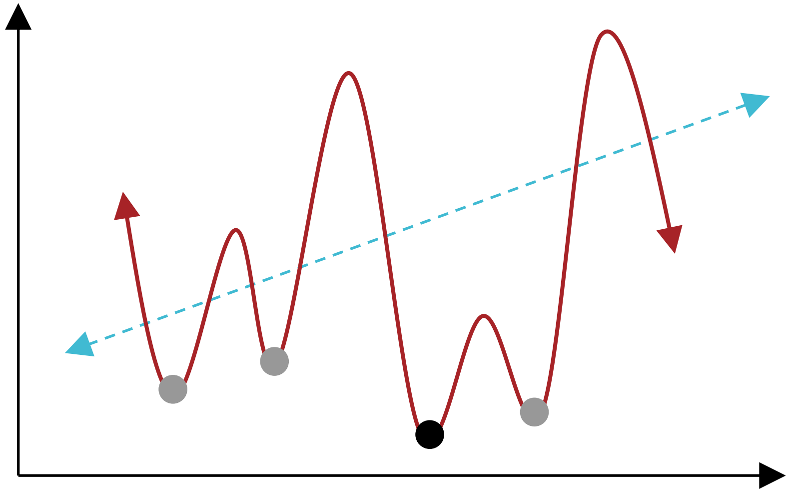
A local minimum may or may not be a global minimum…

Convexity



Strictly convex functions:

There exists a unique global minimum!

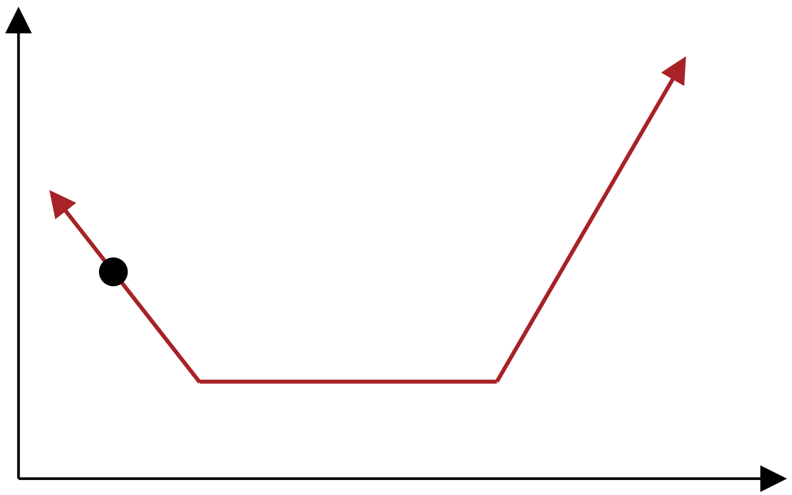


Non-convex functions:

A local minimum may or may not be a global minimum…

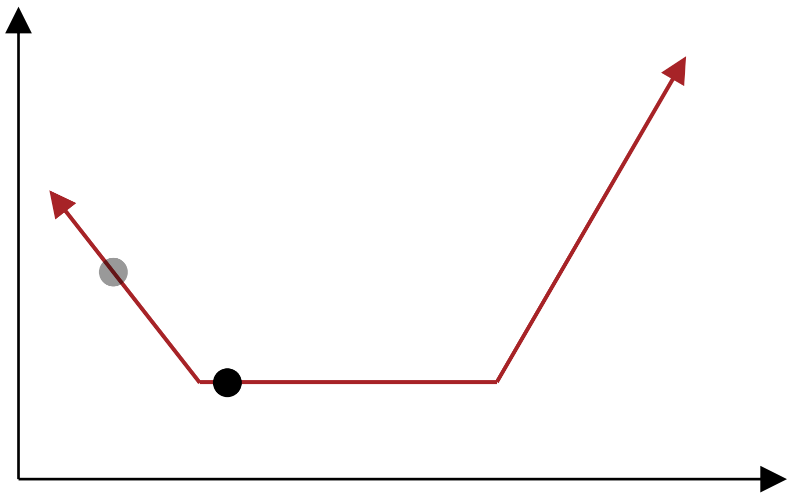
Gradient Descent & Convexity

- Gradient descent is a local optimization algorithm – it will converge to a local minimum (if it converges)
  - Works great if the objective function is convex!



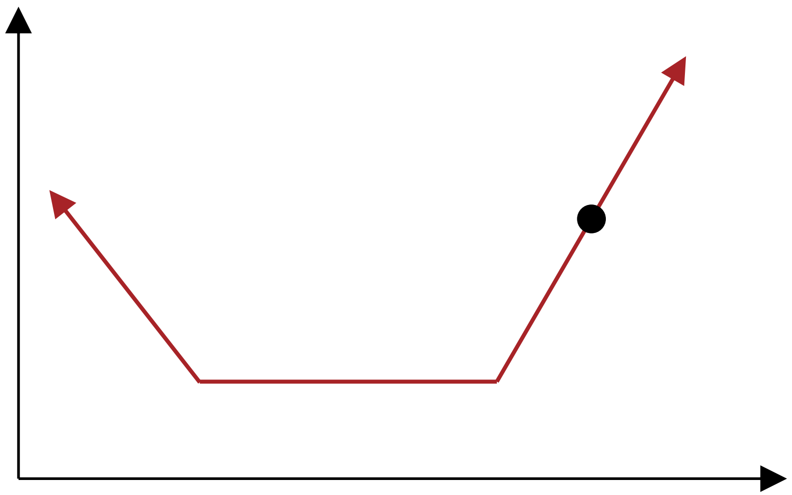
Gradient Descent & Convexity

- Gradient descent is a local optimization algorithm – it will converge to a local minimum (if it converges)
    - Works great if the objective function is convex!



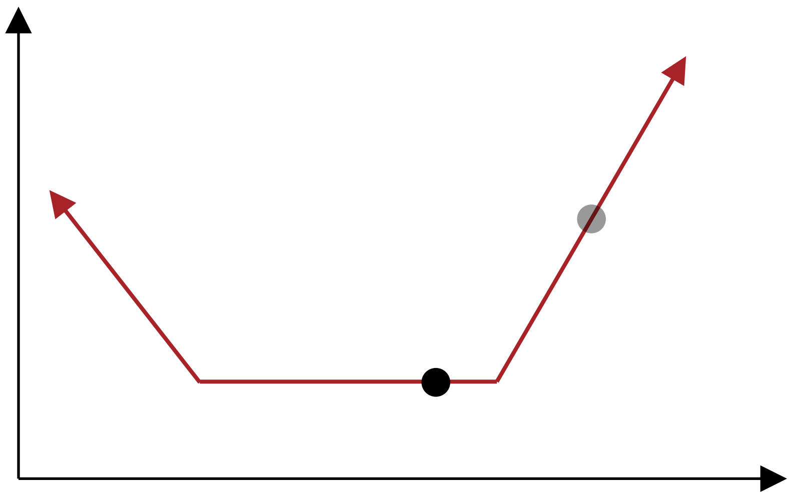
Gradient Descent & Convexity

- Gradient descent is a local optimization algorithm – it will converge to a local minimum (if it converges)
  - Works great if the objective function is convex!



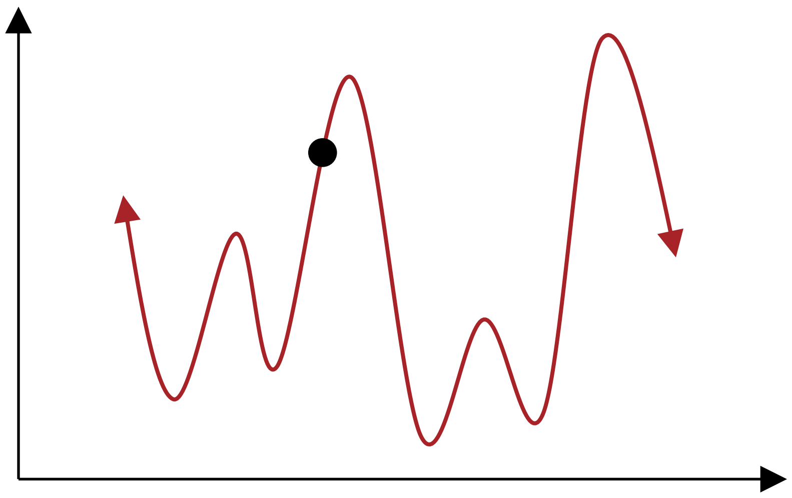
Gradient Descent & Convexity

- Gradient descent is a local optimization algorithm – it will converge to a local minimum (if it converges)
  - Works great if the objective function is convex!



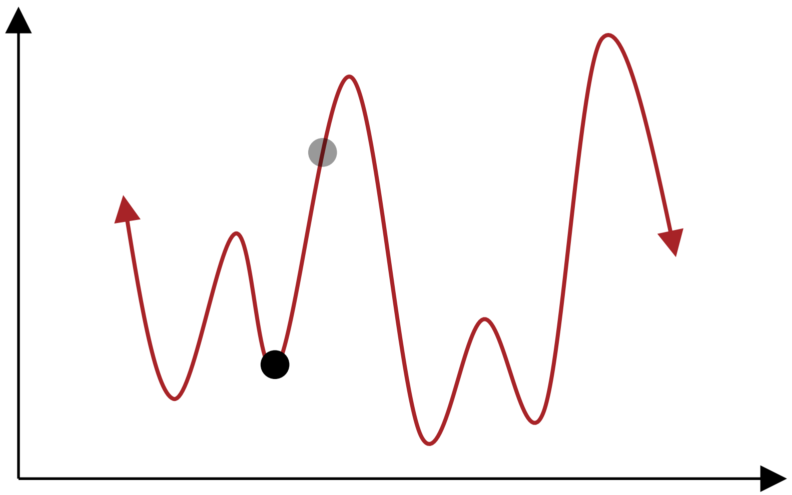
Gradient Descent & Convexity

- Gradient descent is a local optimization algorithm – it will converge to a local minimum (if it converges)
  - Not ideal if the objective function is non-convex…



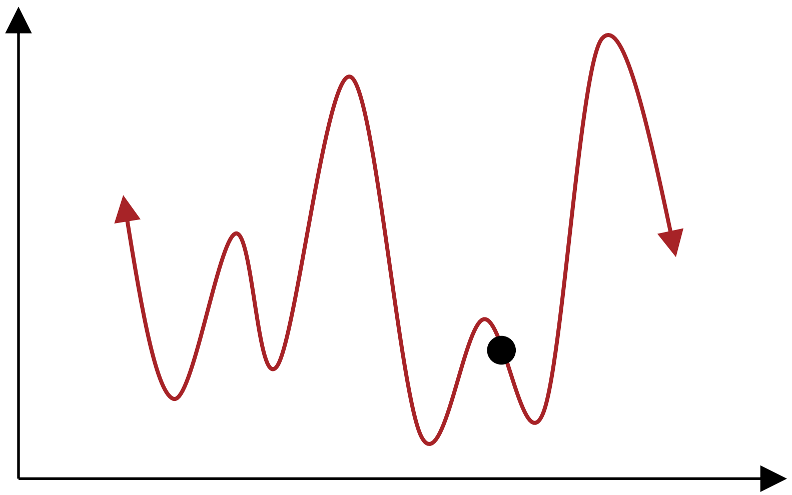
Gradient Descent & Convexity

- Gradient descent is a local optimization algorithm – it will converge to a local minimum (if it converges)
  - Not ideal if the objective function is non-convex…



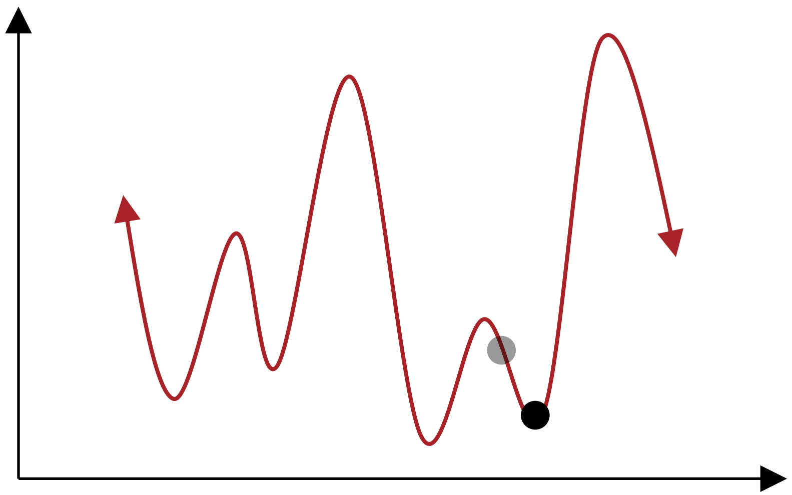
Gradient Descent & Convexity

- Gradient descent is a local optimization algorithm – it will converge to a local minimum (if it converges)
  - Not ideal if the objective function is non-convex…



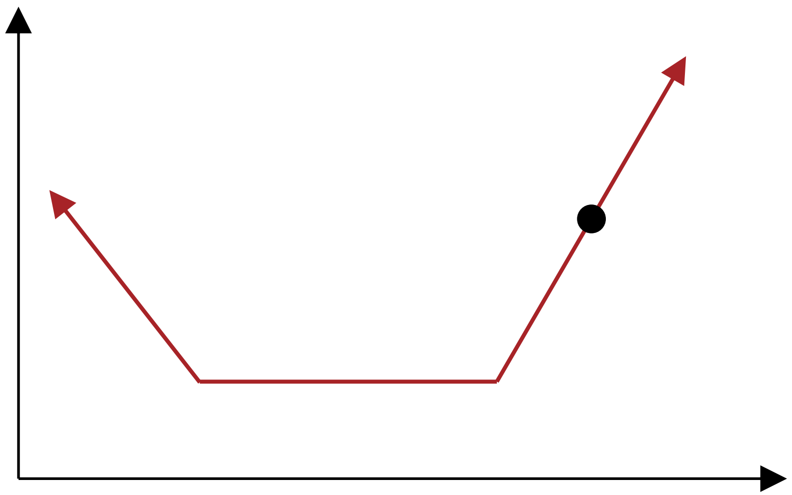
Gradient Descent & Convexity

- Gradient descent is a local optimization algorithm – it will converge to a local minimum (if it converges)
  - Not ideal if the objective function is non-convex…



The squared error for linear regression is convex (but not strictly convex)!

- Gradient descent is a local optimization algorithm – it will converge to a local minimum (if it converges)
  - Works great if the objective function is convex!



$$\nabla_{\boldsymbol{w}} \ell_{\mathcal{D}}(\boldsymbol{w}) = (2X^T X \boldsymbol{w} - 2X^T \boldsymbol{y})$$

$H_{\boldsymbol{w}} \ell_{\mathcal{D}}(\boldsymbol{w}) = 2X^T X$ which is positive *semi*-definite

$$\hat{\boldsymbol{w}} = (X^T X)^{-1} X^T \boldsymbol{y}$$

Closed Form Solution

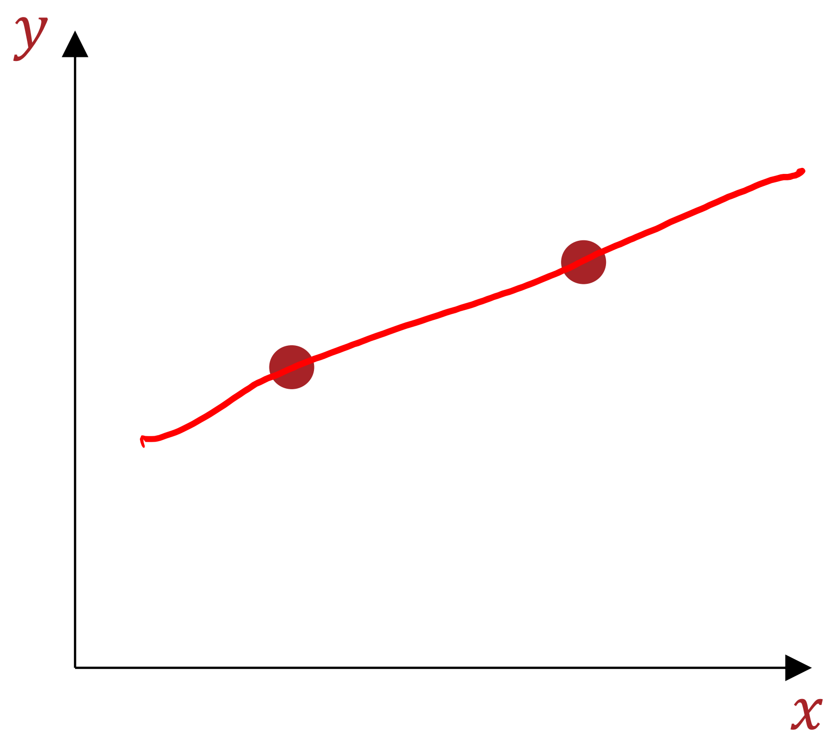
1. Is $X^T X$ invertible?

   Almost always (as long as features are linearly independent)

2. If so, how computationally expensive is inverting $X^T X$?
   - $X^T X \in \mathbb{R}^{D+1 \times D+1}$ so inverting $X^T X$ takes $O(D^3)$ time…
     - Computing $X^T X$ takes $O(ND^2)$ time
   - Can use gradient descent to (potentially) speed things up when N and D are large!

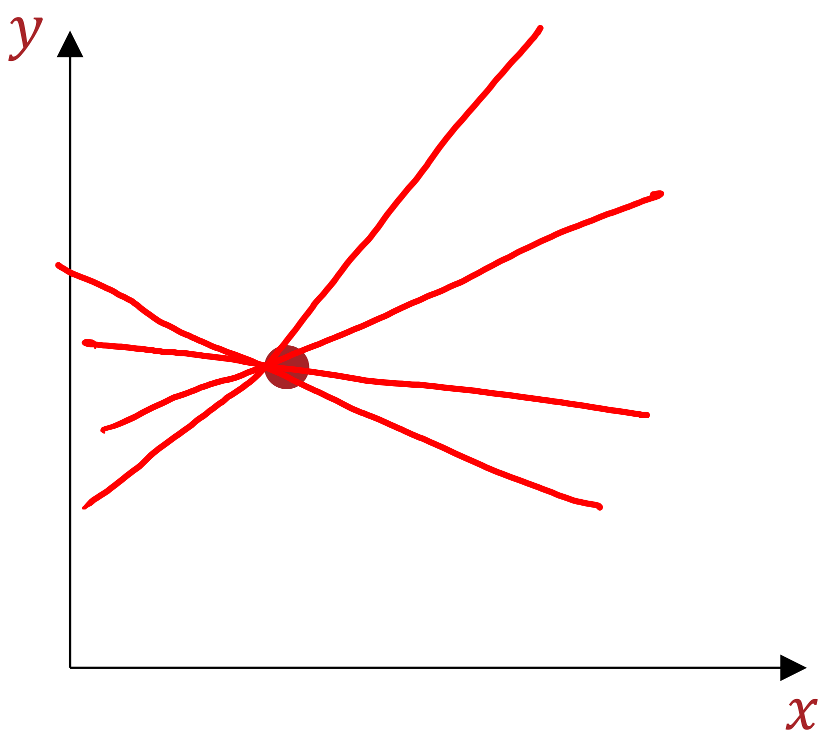
Linear Regression: Uniqueness

- Consider a 1D linear regression model trained to minimize the mean squared error: how many optimal solutions (i.e., sets of weights $\boldsymbol{w}$) are there for the given dataset?



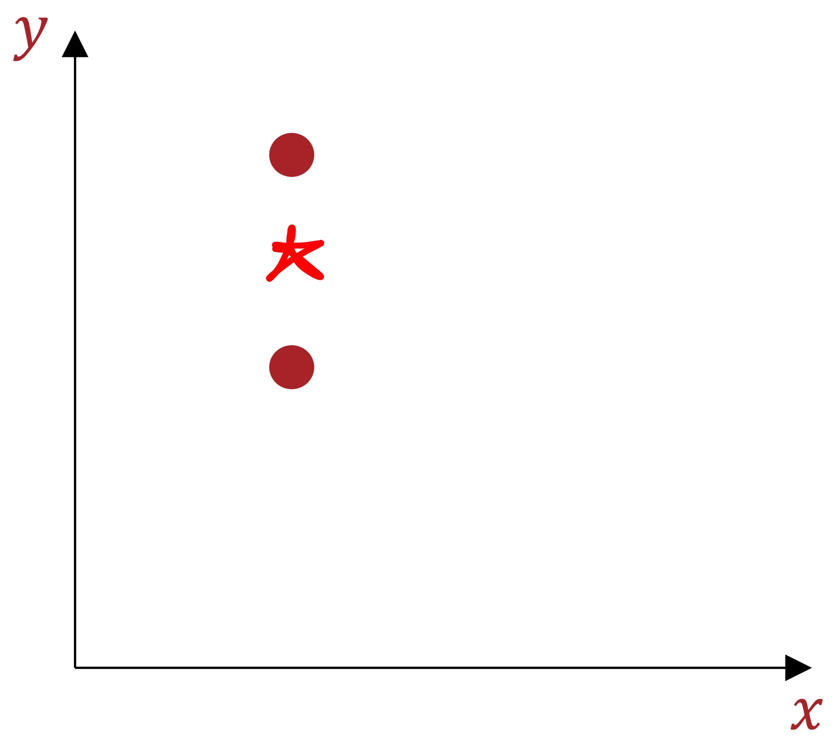
Linear Regression: Uniqueness

- Consider a 1D linear regression model trained to minimize the mean squared error: how many optimal solutions (i.e., sets of weights $\boldsymbol{w}$) are there for the given dataset?



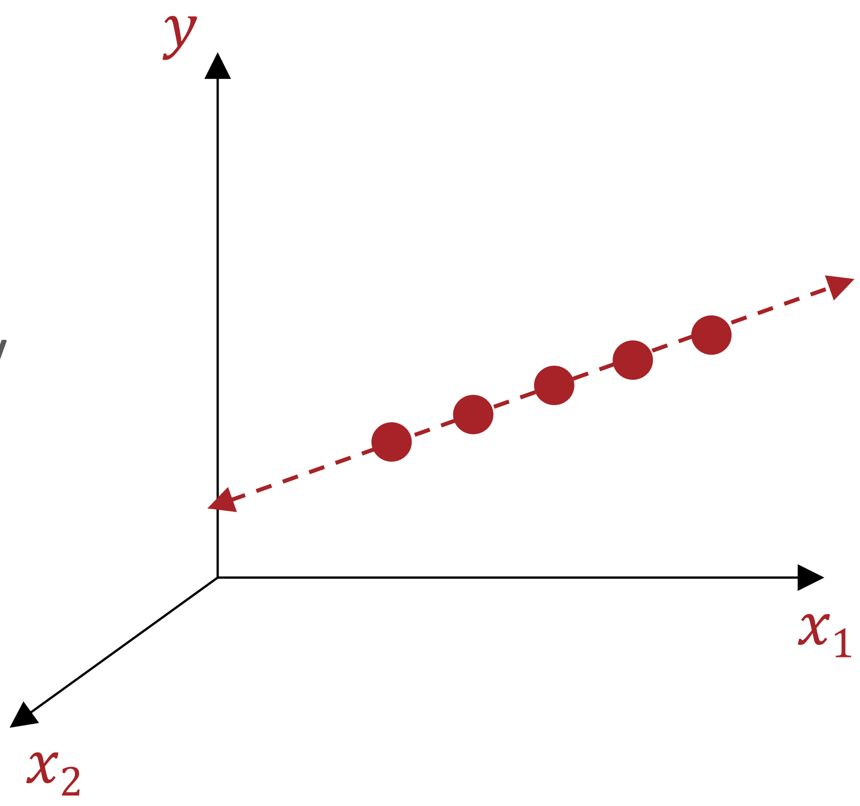
Linear Regression: Uniqueness

- Consider a 1D linear regression model trained to minimize the mean squared error: how many optimal solutions (i.e., sets of weights $\boldsymbol{w}$) are there for the given dataset?



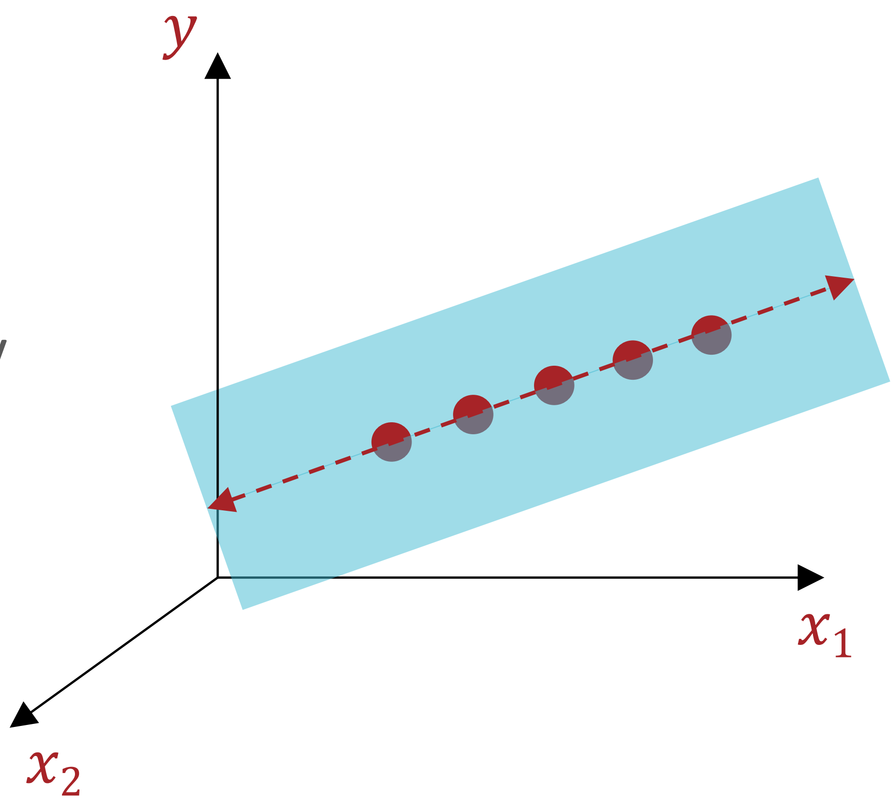
Linear Regression: Uniqueness

- Consider a 2D linear regression model trained to minimize the mean squared error: how many optimal solutions (i.e., sets of parameters θ) are there for the given dataset?



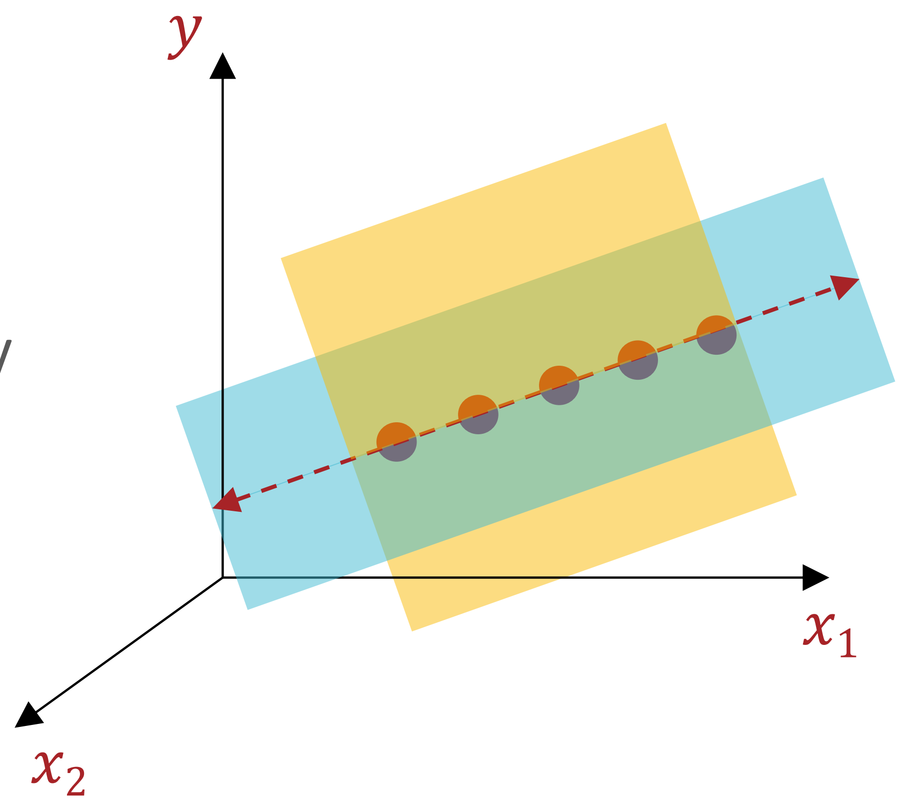
Linear Regression: Uniqueness

- Consider a 2D linear regression model trained to minimize the mean squared error: how many optimal solutions (i.e., sets of weights $\boldsymbol{w}$) are there for the given dataset?



Linear Regression: Uniqueness

- Consider a 2D linear regression model trained to minimize the mean squared error: how many optimal solutions (i.e., sets of weights $\boldsymbol{w}$) are there for the given dataset?



Key Takeaways

- Closed form solution for linear regression
  - Setting the gradient equal to 0 and solving for critical points
  - Potential issues: invertibility and computational costs
- Gradient descent
  - Effect of step size
  - Termination criteria
- Convexity vs. non-convexity
  - Strong vs. weak convexity
  - Implications for local, global and unique optima

Bias-Variance Tradeoff

- Suppose you have a regression task and your goal is to minimize the *true* squared error:

$$err(h) = \mathbb{E}_{\boldsymbol{x} \sim \mathcal{P}}\left[\left(h(\boldsymbol{x}) - f(\boldsymbol{x})\right)^2\right]$$

where f is the target function and

$\mathcal{P}$ is some distribution of interest over all possible inputs

- Let $h_{\mathcal{D}}$ be the hypothesis returned when the input training dataset is $\mathcal{D}$
- Assume each data point in $\mathcal{D}$ is drawn independently from $\mathcal{P}$

Bias-Variance Tradeoff

- $err(h_{\mathcal{D}}) = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}}\left[\left(h_{\mathcal{D}}(\boldsymbol{x}) - f(\boldsymbol{x})\right)^2\right]$

- $\mathbb{E}_{\mathcal{D}}[err(h_{\mathcal{D}})]$

$$= E_D\left[E_{x\sim p}\left[(h_D(x) - f(x))^2\right]\right]$$

$$= E_{x\sim p}\left[E_D\left[(h_D(x) - f(x))^2\right]\right]$$

$$= E_{x\sim p}\left[E_D\left[h_D(x)^2 - 2h_D(x)f(x) + f(x)^2\right]\right]$$

$$= E_{x\sim p}\left[E_D[h_D(x)^2] - 2E_D[h_D(x)]f(x) + f(x)^2\right]$$

$$\bar{h}(x) = E_D[h_D(x)] \approx \frac{1}{C}\sum_{i=1}^{C} h_{D_i}(x)$$

Bias-Variance Tradeoff

- $\mathbb{E}_{\mathcal{D}}[err(h_{\mathcal{D}})]$

$$= E_{x\sim p}\left[E_D[h_D(x)^2] - 2\bar{h}(x)f(x) + f(x)^2\right]$$

$$= E_{x\sim p}\left[E_D[h_D(x)^2] - \bar{h}(x)^2 + \underbrace{\bar{h}(x)^2 - 2\bar{h}(x)f(x) + f(x)^2}\right]$$

$$= E_{x\sim p}\left[\underbrace{E_D[h_D(x)^2 - \bar{h}(x)^2]}_{\text{variance of } h_D(x)} + \underbrace{(\bar{h}(x) - f(x))^2}_{(\text{bias of } \bar{h}(x))^2}\right]$$

Bias-Variance Tradeoff

How variable is $h_{\mathcal{D}}$?

$$\mathbb{E}_{\mathcal{D}}[err(h_{\mathcal{D}})] = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}}\left[\mathbb{E}_{\mathcal{D}}\left[h_{\mathcal{D}}(\boldsymbol{x})^2 - \bar{h}(\boldsymbol{x})^2\right] + \left(\bar{h}(\boldsymbol{x}) - f(\boldsymbol{x})\right)^2\right]$$

How well, on average, does $h_{\mathcal{D}}$ approximate f?

Bias-Variance Tradeoff

How well could $h_{\mathcal{D}}$ approximate anything?

$$\mathbb{E}_{\mathcal{D}}[err(h_{\mathcal{D}})] = \mathbb{E}_{\boldsymbol{x} \sim \mathcal{P}}\left[\mathbb{E}_{\mathcal{D}}\left[h_{\mathcal{D}}(\boldsymbol{x})^2 - \bar{h}(\boldsymbol{x})^2\right] + \left(\bar{h}(\boldsymbol{x}) - f(\boldsymbol{x})\right)^2\right]$$

How well, on average,
does $h_{\mathcal{D}}$ approximate f?

Bias-Variance Tradeoff

How well could $h_{\mathcal{D}}$ approximate random noise?

$$\mathbb{E}_{\mathcal{D}}[err(h_{\mathcal{D}})] = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}}\left[\mathbb{E}_{\mathcal{D}}\left[h_{\mathcal{D}}(\boldsymbol{x})^2 - \bar{h}(\boldsymbol{x})^2\right] + \left(\bar{h}(\boldsymbol{x}) - f(\boldsymbol{x})\right)^2\right]$$

How well, on average, does $h_{\mathcal{D}}$ approximate f?

Bias-Variance Tradeoff

Increases as the model becomes more complex

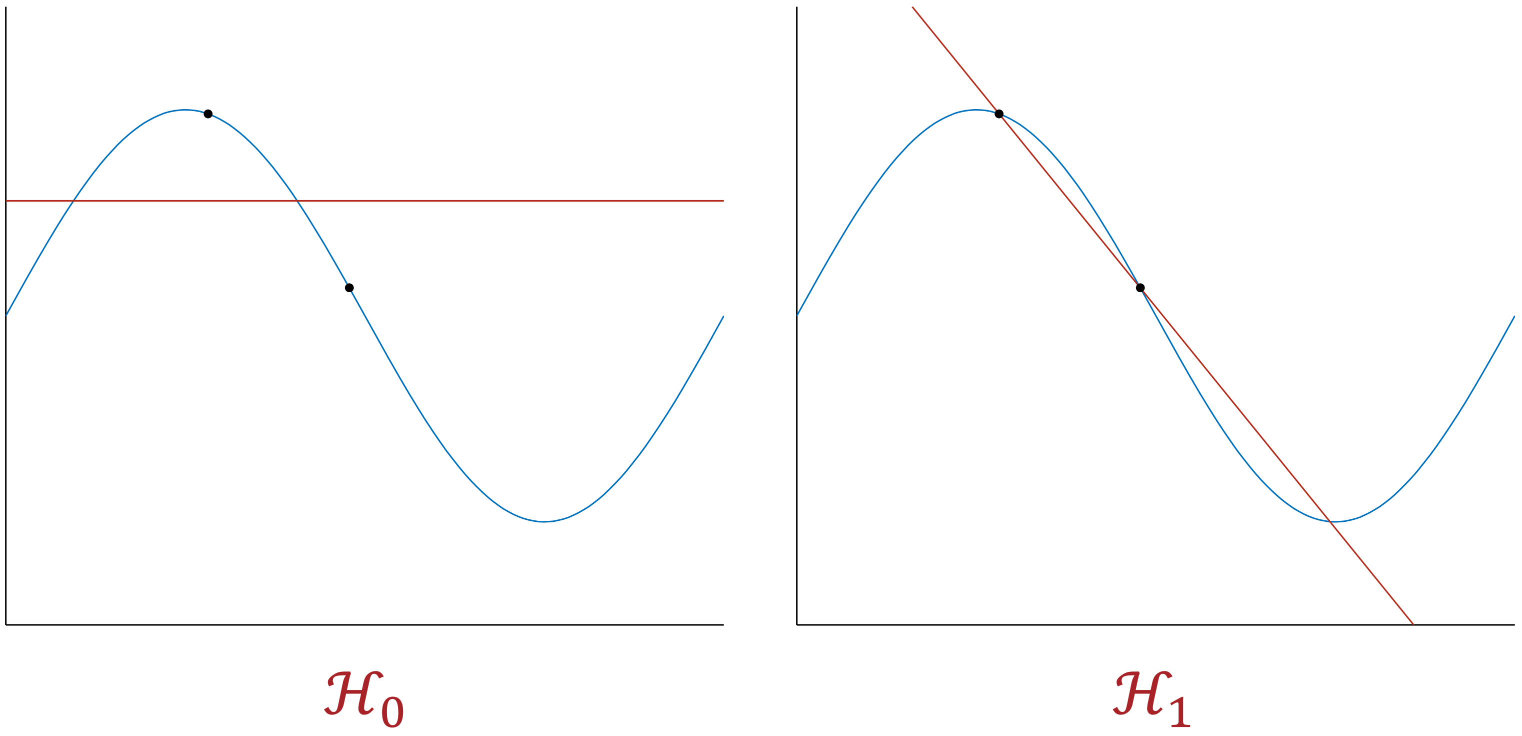
$$\mathbb{E}_{\mathcal{D}}[err(h_{\mathcal{D}})] = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}}\left[\mathbb{E}_{\mathcal{D}}[h_{\mathcal{D}}(\boldsymbol{x})^2 - \bar{h}(\boldsymbol{x})^2] + \left(\bar{h}(\boldsymbol{x}) - f(\boldsymbol{x})\right)^2\right]$$

Decreases as the model becomes more complex

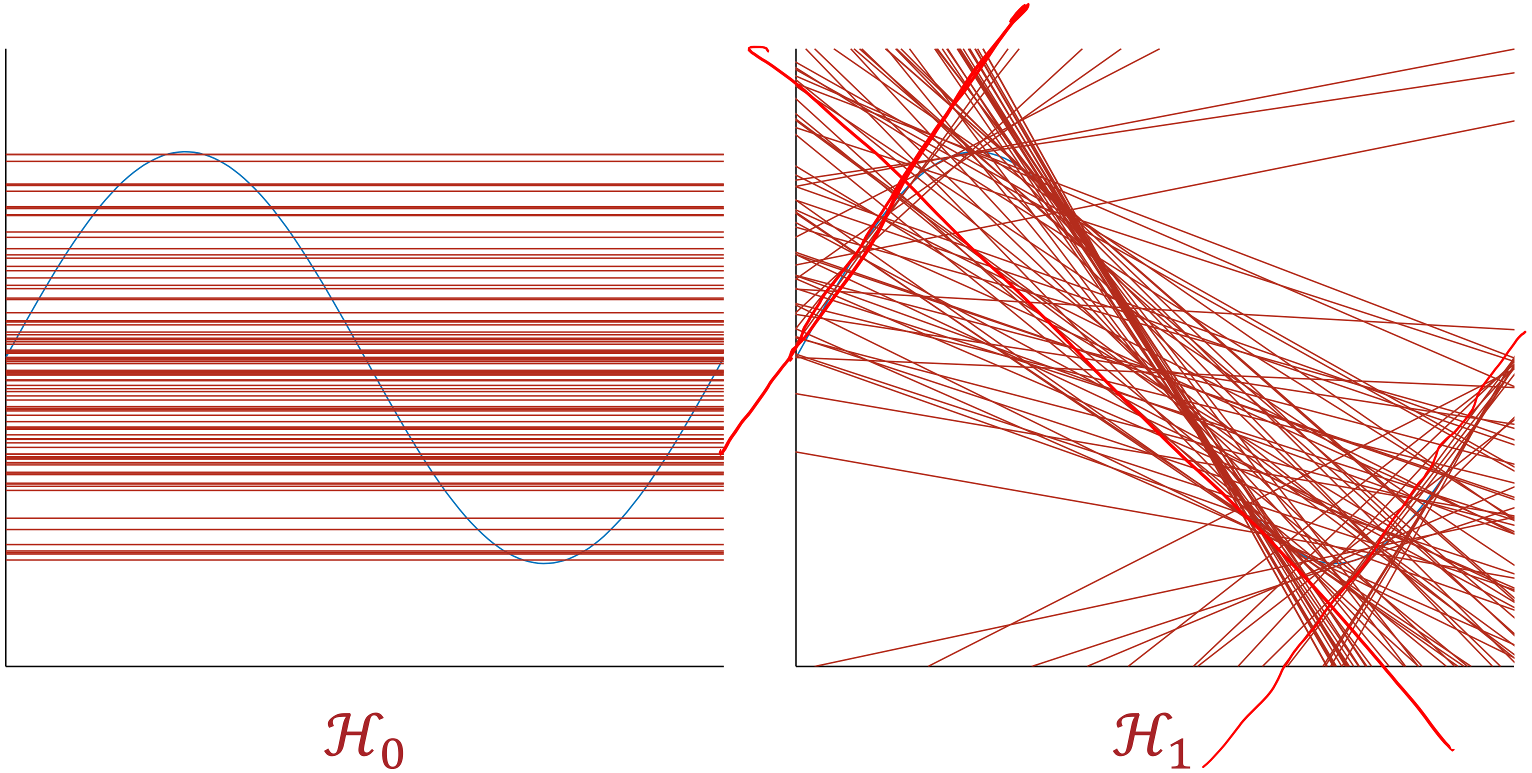
Bias-Variance Tradeoff (Example)

- $\mathcal{X} = \mathbb{R}$ and $\mathcal{P} = \text{Uniform}(0, 2\pi)$
- $f(x) = \sin(x)$
- $N = 2 \rightarrow \mathcal{D} = \{(x_1, \sin(x_1)), (x_2, \sin(x_2))\}$
- Consider two models:
  - The “constant” model - $\mathcal{H}_0 = \{h : h(x) = b\}$
  - Linear regression - $\mathcal{H}_1 = \{h : h(x) = ax + b\}$

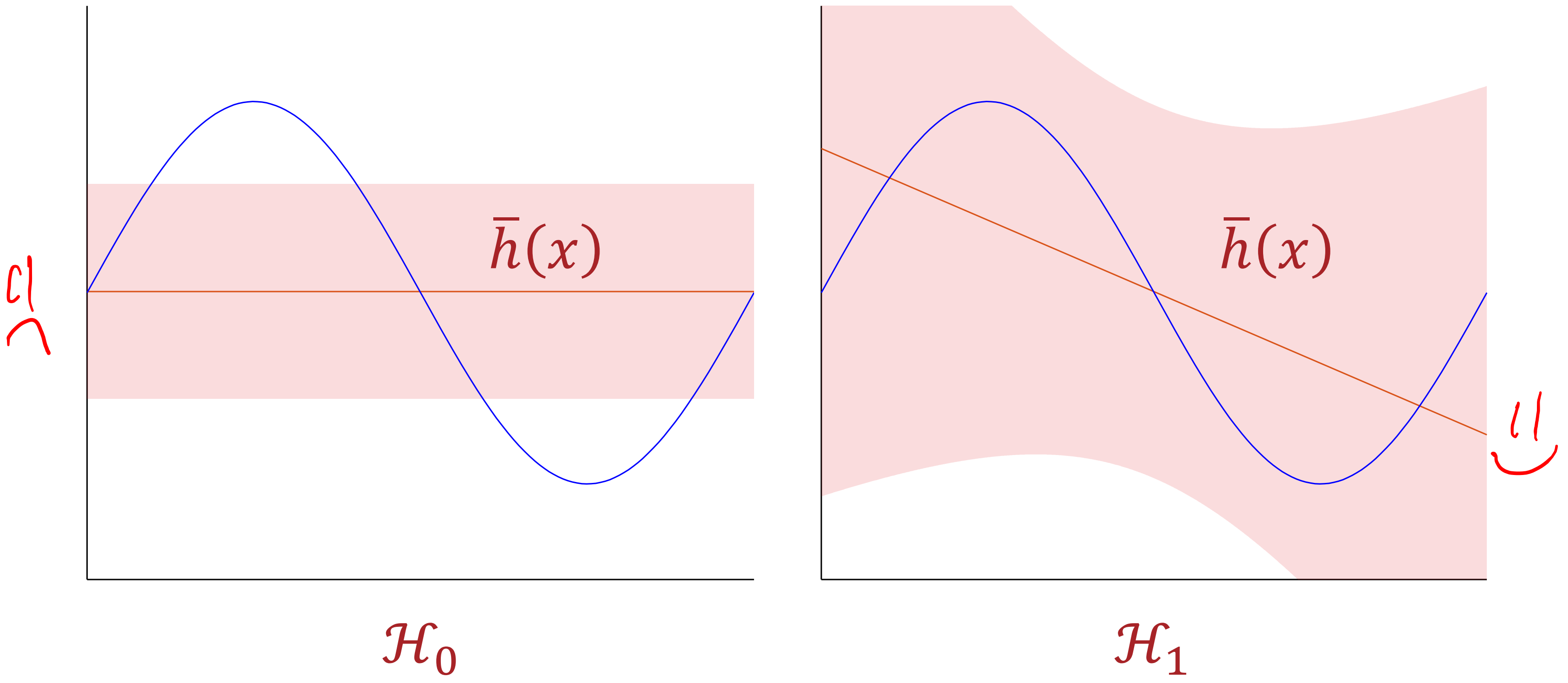
Bias-Variance Tradeoff (Example)



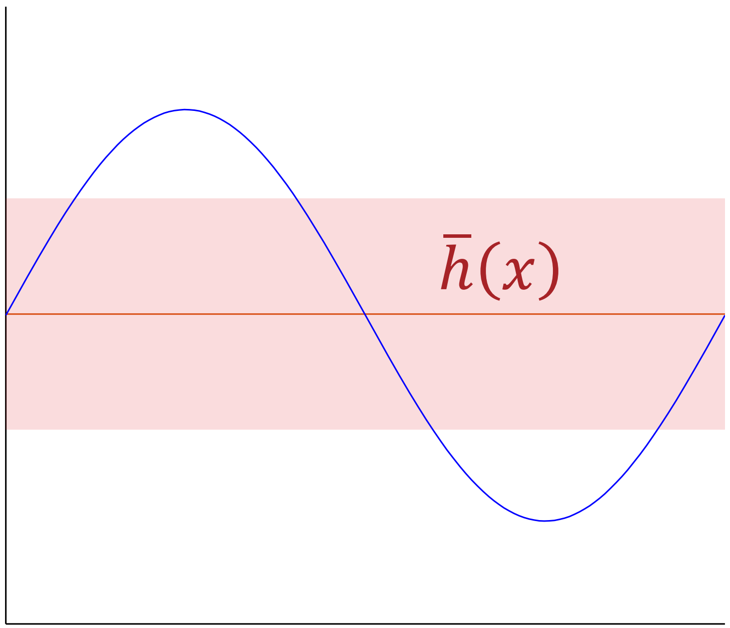
Bias-Variance Tradeoff (Example)



Bias-Variance Tradeoff (Example)



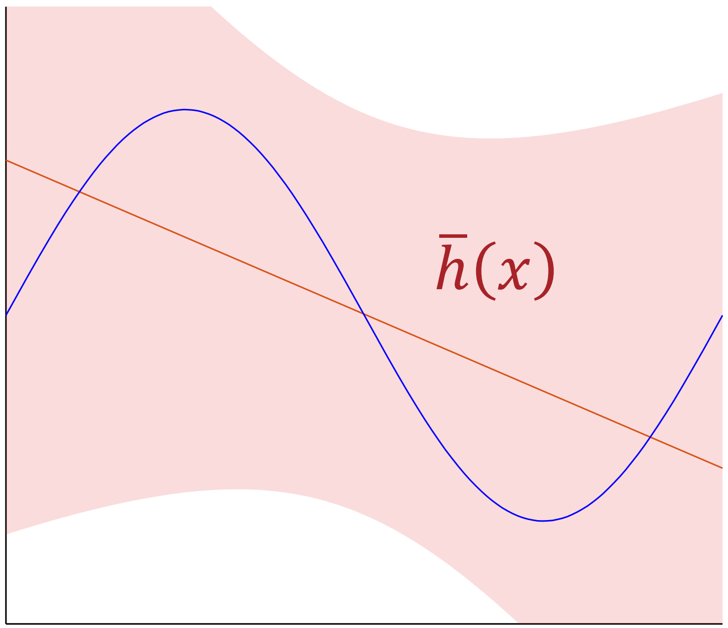
Bias-Variance Tradeoff $(N = 2)$



Bias of $\bar{h}(x) \approx 0.50$

Variance of $h_{\mathcal{D}}(x) \approx 0.25$

$\mathbb{E}_{\mathcal{D}}[err(h_{\mathcal{D}})] \approx 0.75$

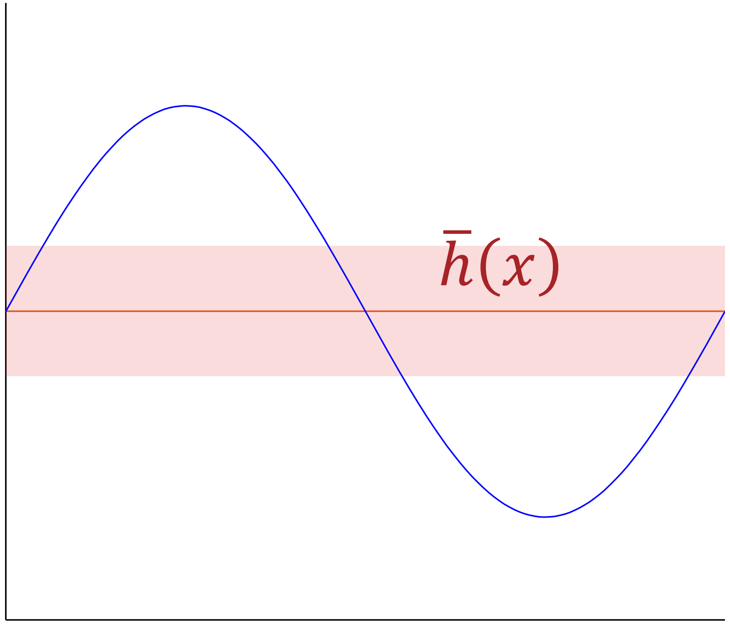


Bias of $\bar{h}(x) \approx 0.21$

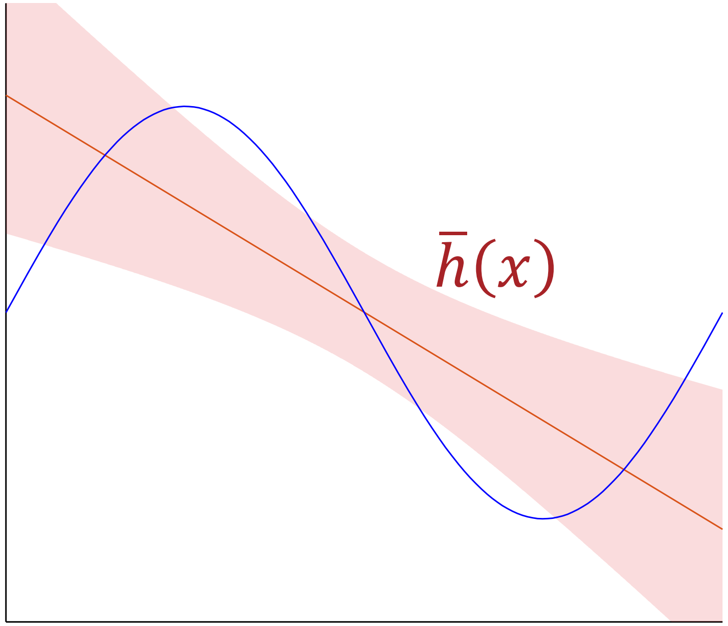
Variance of $h_{\mathcal{D}}(x) \approx 1.74$

$\mathbb{E}_{\mathcal{D}}[err(h_{\mathcal{D}})] \approx 1.95$

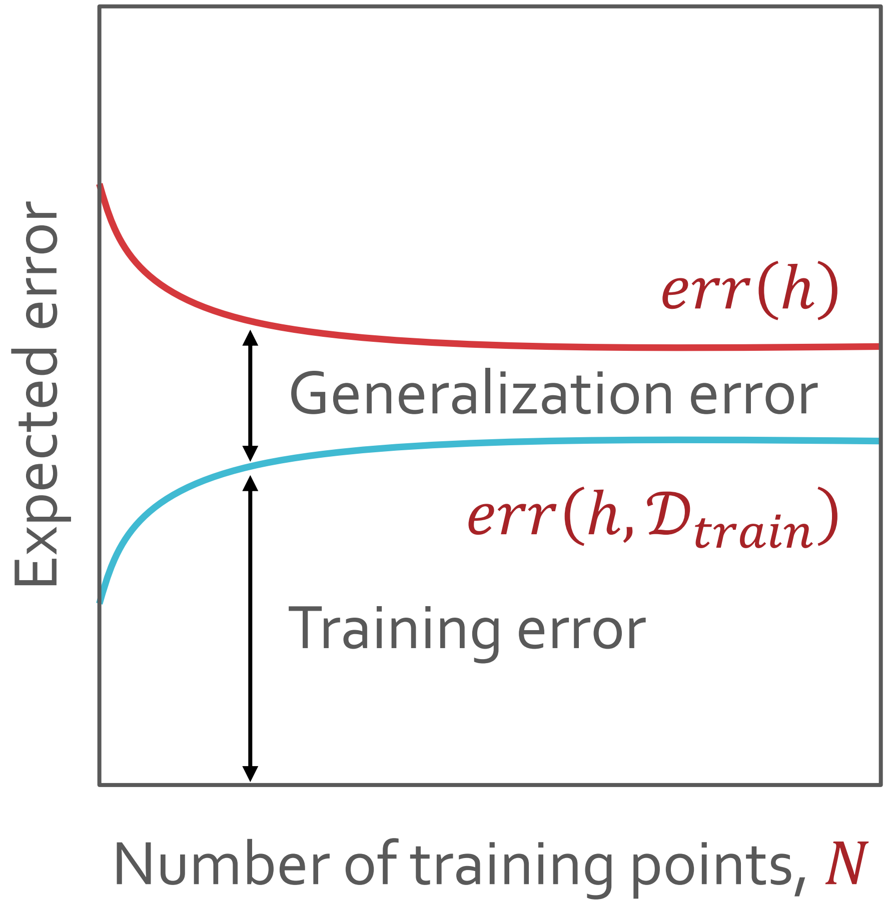
Bias-Variance Tradeoff ($N = 5$)

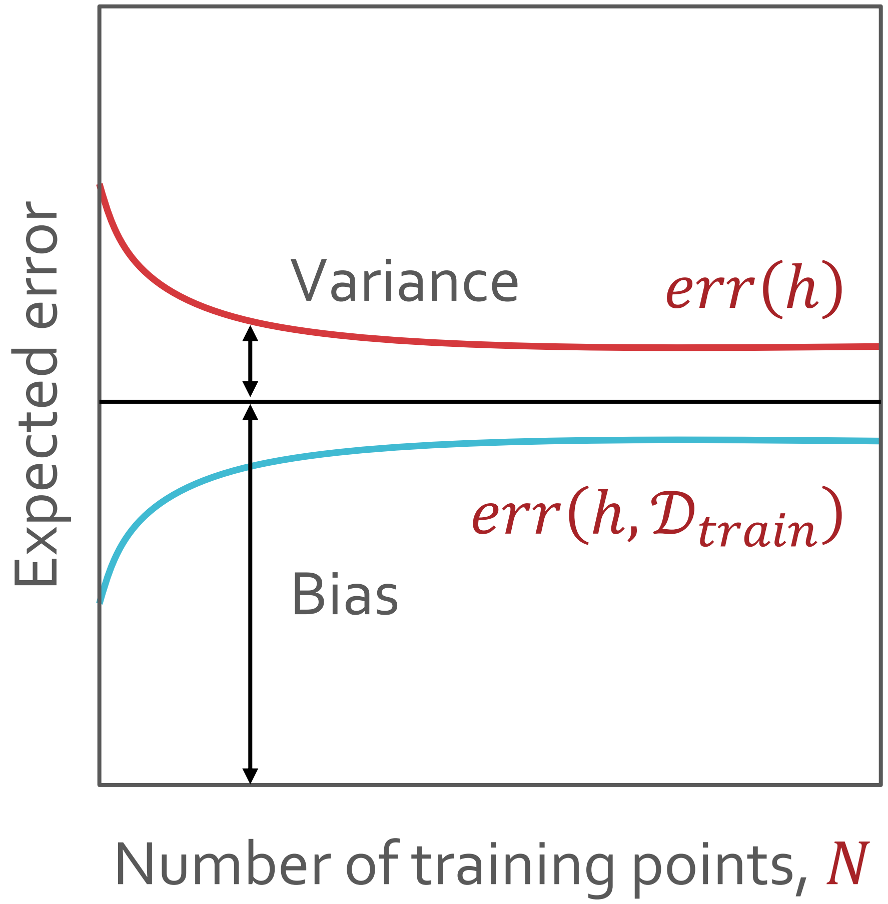


Bias of $\bar{h}(x) \approx 0.50$
Variance of $h_{\mathcal{D}}(x) \approx 0.10$
$\mathbb{E}_{\mathcal{D}}[err(h_{\mathcal{D}})] \approx 0.60$



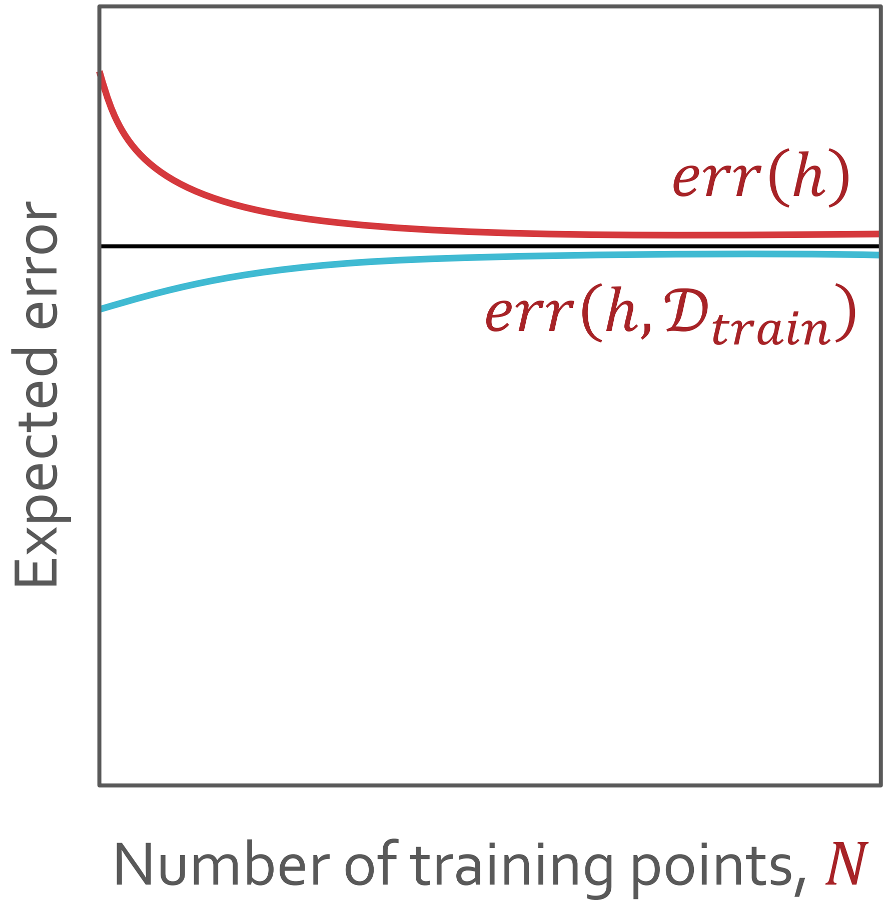
Bias of $\bar{h}(x) \approx 0.21$
Variance of $h_{\mathcal{D}}(x) \approx 0.21$
$\mathbb{E}_{\mathcal{D}}[err(h_{\mathcal{D}})]$ $\boxed{\approx 0.42}$

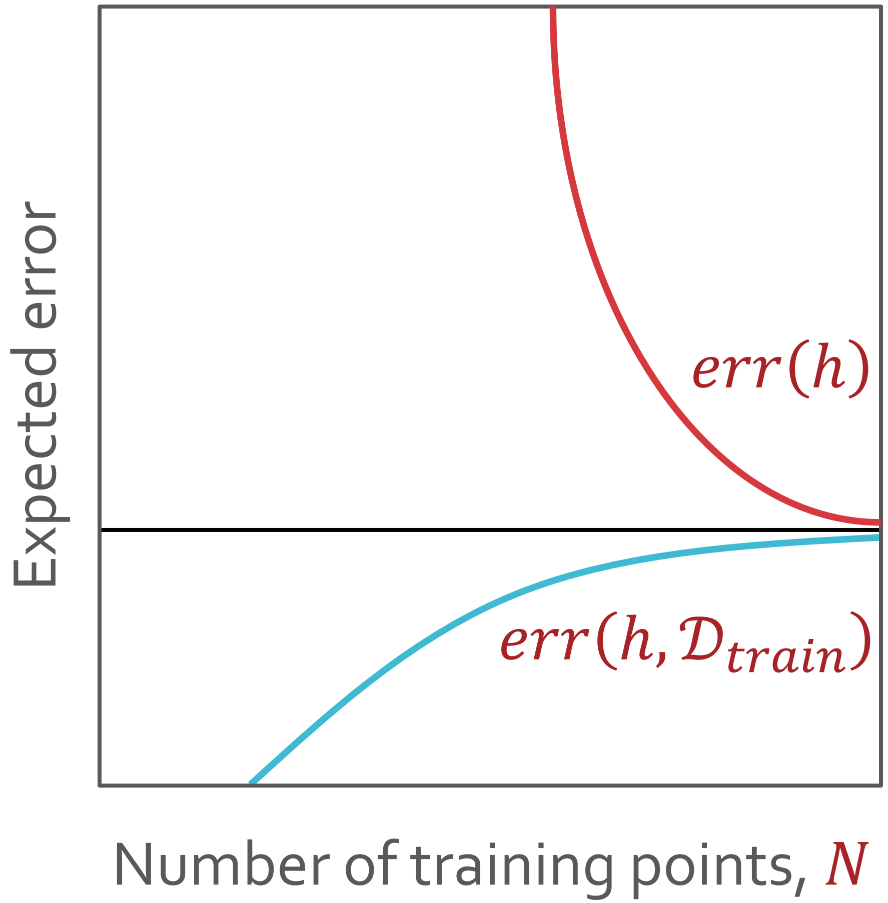




Generalization

Bias-Variance analysis





Simple model

Complex model