# 10-301/601: Introduction to Machine Learning Lecture 8 – Optimization for Machine Learning

Matt Gormley & Henry Chai

9/18/24

# Front Matter

- Announcements:
  - HW3 released 9/16, due 9/23 at 11:59 PM
    - **Only two grace days allowed on HW3**
  - Exam 1 on 9/30 from 6:30 PM - 8:30 PM
    - If you have a conflict, you must complete the Exam conflict form by 9/23 at 1 PM

# Exam 1 Logistics

- Location & Seats: You all will be split across multiple (large) rooms.
  - Everyone will have an assigned seat
  - Please watch Piazza carefully for more details
  - If you have exam accommodations through ODR, they will be proctoring your exam on our behalf; **you are responsible for submitting the exam proctoring request through your student portal.**

# Exam 1 Logistics

- Format of questions:
  - Multiple choice
  - True / False (with justification)
  - Derivations
  - Short answers
  - Drawing & Interpreting figures
  - Implementing algorithms on paper
- No electronic devices (you won't need them!)
- You are allowed to bring one letter-size sheet of notes; you can put *whatever* you want on *both sides*

# Exam 1 Topics

- Covered material: Lectures 1 – 7
  - Foundations
    - Probability, Linear Algebra, Geometry, Calculus
    - Optimization
  - Important Concepts
    - Overfitting
    - Model selection / Hyperparameter optimization
  - Decision Trees
  - $k$-NN
  - Perceptron
  - Regression
    - Decision Tree and $k$-NN Regression
    - Linear Regression

# Exam 1 Preparation

- Attend the midterm review lecture (right now!)
- Review the exam practice problems (to be released on 9/20, under the Coursework tab)
- Review HWs 1 - 3
- Consider whether you have achieved the “learning objectives” for each lecture / section
- Write your one-page cheat sheet (back and front)

# Exam 1 Tips

- Solve the easy problems first
- If a problem seems extremely complicated, you might be missing something
- If you make an assumption, write it down
- Don’t leave any answer blank
  - If you look at a question and don’t know the answer:
    - just start trying things
    - consider multiple approaches
    - imagine arguing for some answer and see if you like it

# Practice Problem 1a: Decision Trees

Consider the problem of predicting whether the university will be closed on a particular day. We will assume that the factors which decide this are whether there is a snowstorm, whether it is a weekend or an official holiday. Suppose we have the training examples described in the Table 5.2.

| Snowstorm | Holiday | Weekend | Closed |
|---|---|---|---|
| T | T | F | F |
| T | T | F | T |
| F | T | F | F |
| T | T | F | F |
| F | F | F | F |
| F | F | F | T |
| T | F | F | T |
| F | F | F | T |

Table 1: Training examples for decision tree

- What would be the effect of the “Weekend” attribute on the decision tree if we made it the root node? Explain your answer in terms of mutual information

# Practice Problem 1b: Decision Trees

Consider the problem of predicting whether the university will be closed on a particular day. We will assume that the factors which decide this are whether there is a snowstorm, whether it is a weekend or an official holiday. Suppose we have the training examples described in the Table 5.2.

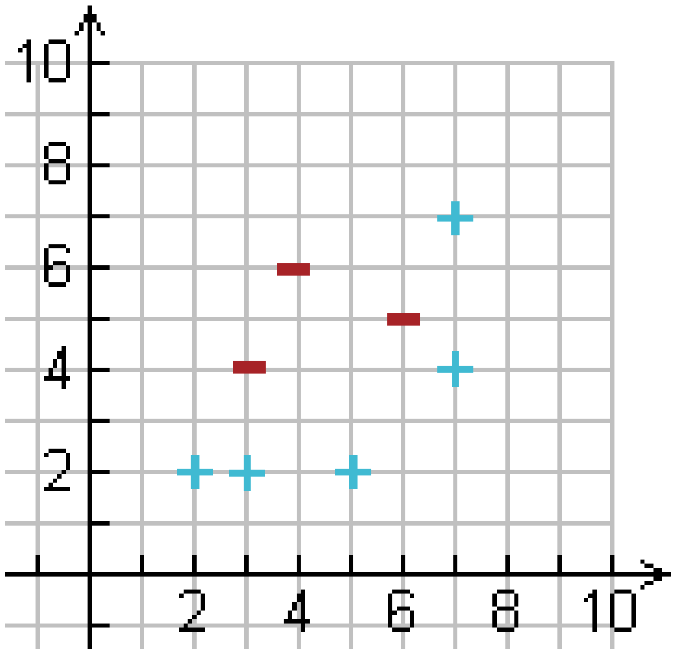
| Snowstorm | Holiday | Weekend | Closed |
|---|---|---|---|
| T | T | F | F |
| T | T | F | T |
| F | T | F | F |
| T | T | F | F |
| F | F | F | F |
| F | F | F | T |
| T | F | F | T |
| F | F | F | T |

Table 1: Training examples for decision tree

- Which attribute would we split on first if we used mutual information as the splitting criterion? You may use $\log_2\left(\frac{3}{4}\right) \approx -0.4$ and $\log_2\left(\frac{1}{4}\right) = -2$

# Practice Problem 2: $k$-NN

- Consider the dataset below:




- What is the leave-one-out cross-validation error for a 1-NN model using the Euclidean distance?

# Practice Problem 3: Perceptron

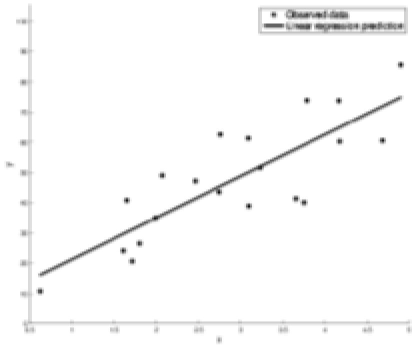
- True or False: Consider two datasets $\mathcal{D}_1 = \left\{\left(\boldsymbol{x}_1^{(1)}, y_1^{(1)}\right), \left(\boldsymbol{x}_1^{(2)}, y_1^{(2)}\right), \dots, \left(\boldsymbol{x}_1^{(N_1)}, y_1^{(N_1)}\right)\right\}$ and $\mathcal{D}_2 = \left\{\left(\boldsymbol{x}_2^{(1)}, y_2^{(1)}\right), \left(\boldsymbol{x}_2^{(2)}, y_2^{(2)}\right), \dots, \left(\boldsymbol{x}_2^{(N_2)}, y_2^{(N_2)}\right)\right\}$ where $\boldsymbol{x}_1^{(i)} \in \mathbb{R}^{d_1}$ and $\boldsymbol{x}_2^{(i)} \in \mathbb{R}^{d_2}$. Suppose $N_1 > N_2$ and $d_1 > d_2$. The maximum number of mistakes the Perceptron learning algorithm will make on $\mathcal{D}_1$ is higher than the maximum number of mistakes it will make on $\mathcal{D}_2$.

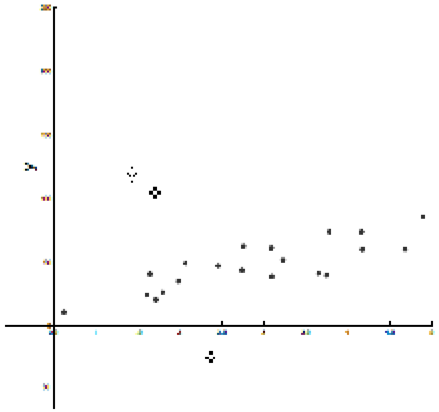
# Poll Question 1

- True or False: Consider two datasets $\mathcal{D}_1 = \left\{\left(\boldsymbol{x}_1^{(1)}, y_1^{(1)}\right), \left(\boldsymbol{x}_1^{(2)}, y_1^{(2)}\right), \dots, \left(\boldsymbol{x}_1^{(N_1)}, y_1^{(N_1)}\right)\right\}$ and $\mathcal{D}_2 = \left\{\left(\boldsymbol{x}_2^{(1)}, y_2^{(1)}\right), \left(\boldsymbol{x}_2^{(2)}, y_2^{(2)}\right), \dots, \left(\boldsymbol{x}_2^{(N_2)}, y_2^{(N_2)}\right)\right\}$ where $\boldsymbol{x}_1^{(i)} \in \mathbb{R}^{d_1}$ and $\boldsymbol{x}_2^{(i)} \in \mathbb{R}^{d_2}$. Suppose $N_1 > N_2$ and $d_1 > d_2$. The maximum number of mistakes the Perceptron learning algorithm will make on $\mathcal{D}_1$ is higher than the maximum number of mistakes it will make on $\mathcal{D}_2$.
  - True
  - False
  - True and False **(TOXIC)**

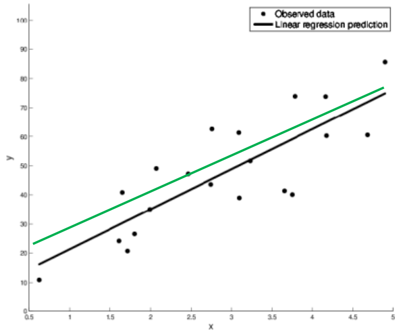
# Practice Problem 4a: Linear Regression

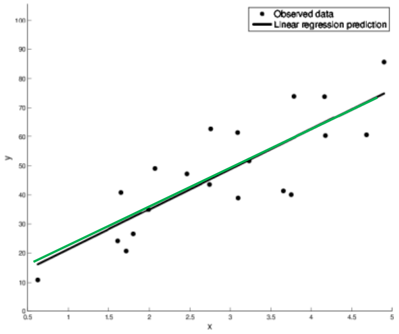
Consider the dataset plotted in the figure below along with the line learned by linear regression.

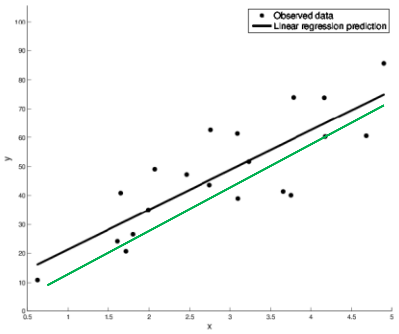




Now suppose we slightly alter the dataset in different ways: for each new dataset, select the option below that best approximates the new line linear regression would learn

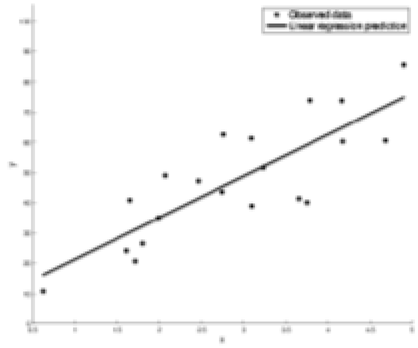


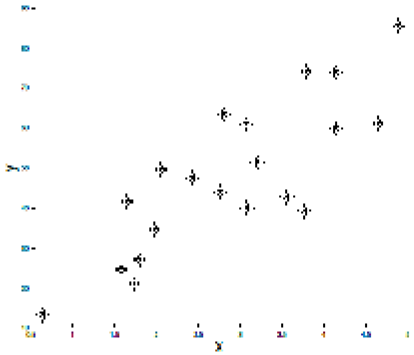



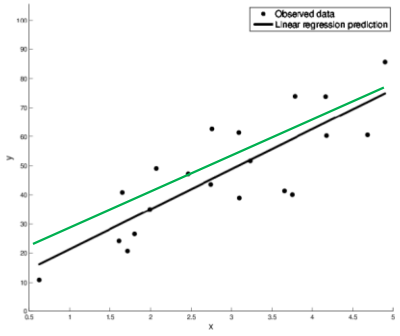
# Practice Problem 4b: Linear Regression

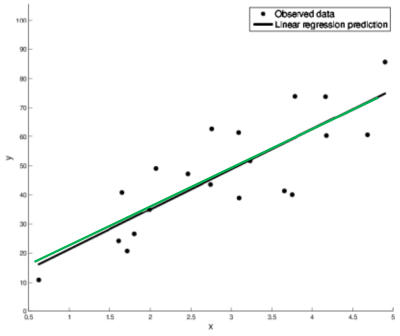
Consider the dataset plotted in the figure below along with the line learned by linear regression.

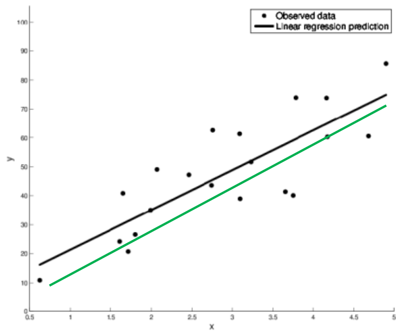




Now suppose we slightly alter the dataset in different ways: for each new dataset, select the option below that best approximates the new line linear regression would learn

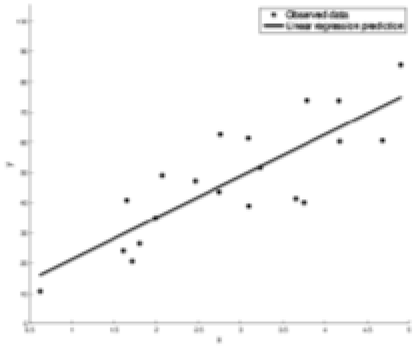


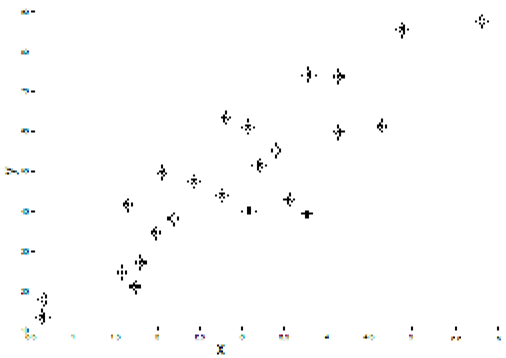



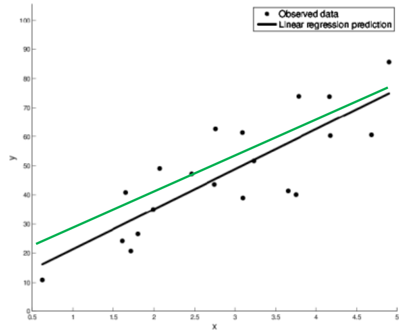
# Practice Problem 4c: Linear Regression

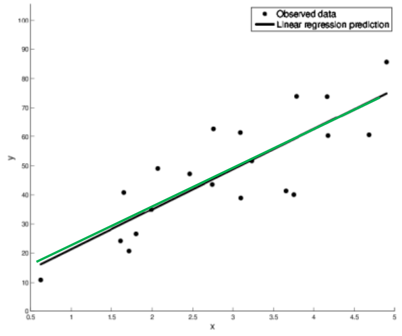
Consider the dataset plotted in the figure below along with the line learned by linear regression.

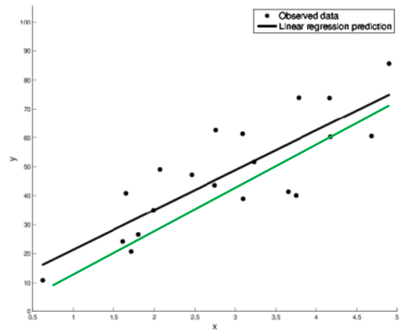




Now suppose we slightly alter the dataset in different ways: for each new dataset, select the option below that best approximates the new line linear regression would learn



# Poll Question 2

What questions do you have?

# Recall: Gradient Descent for Linear Regression

- Gradient descent for linear regression repeatedly takes steps opposite the gradient of the objective function

**Algorithm 1** GD for Linear Regression

```
procedure GDLR(𝒟, θ^(0))
    θ ← θ^(0)                                   ▷ Initialize parameters
    while not converged do
        g ← Σ_{i=1}^N (θ^T x^(i) − y^(i)) x^(i)   ▷ Compute gradient
        θ ← θ − γg                              ▷ Update parameters
    return θ
```

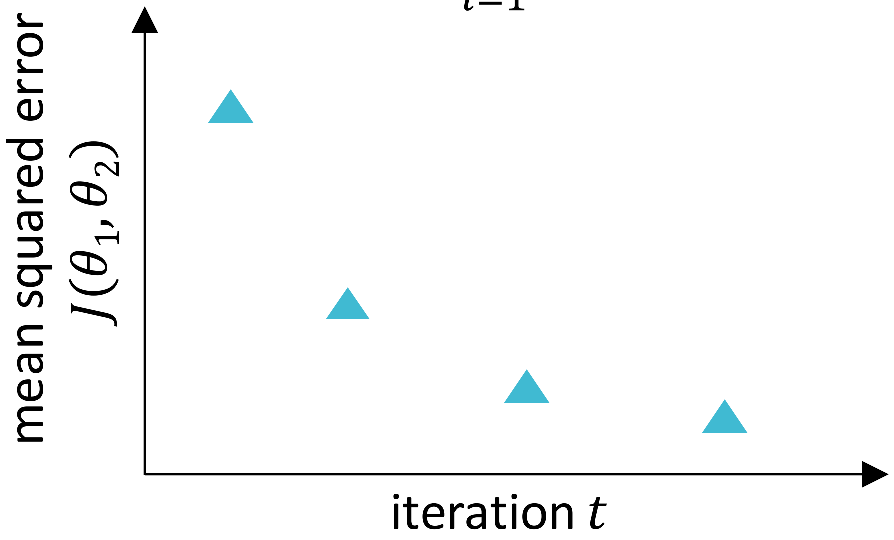
Where:

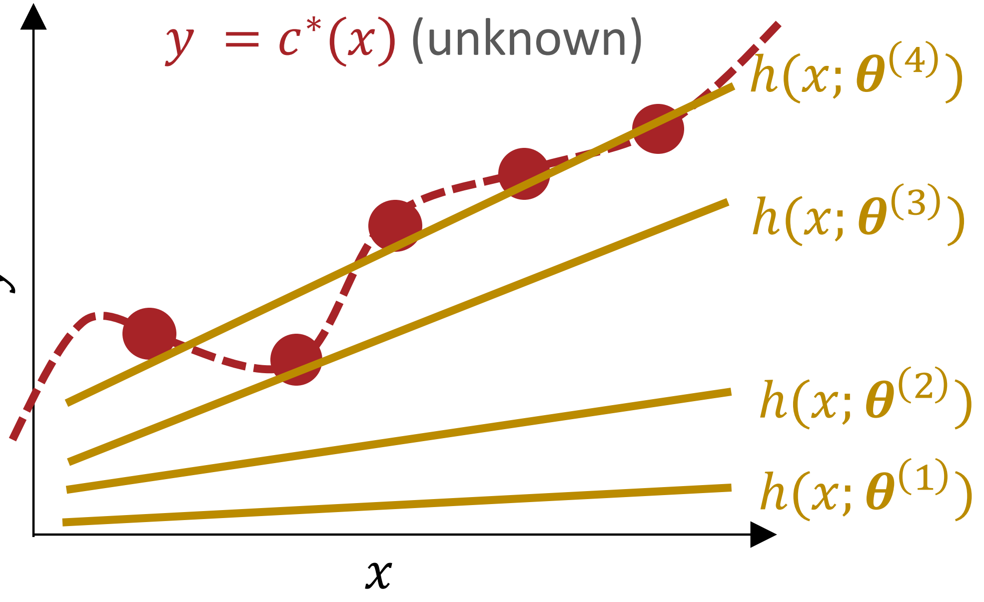
$$\mathbf{g} \leftarrow \sum_{i=1}^{N} (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)})\mathbf{x}^{(i)}$$

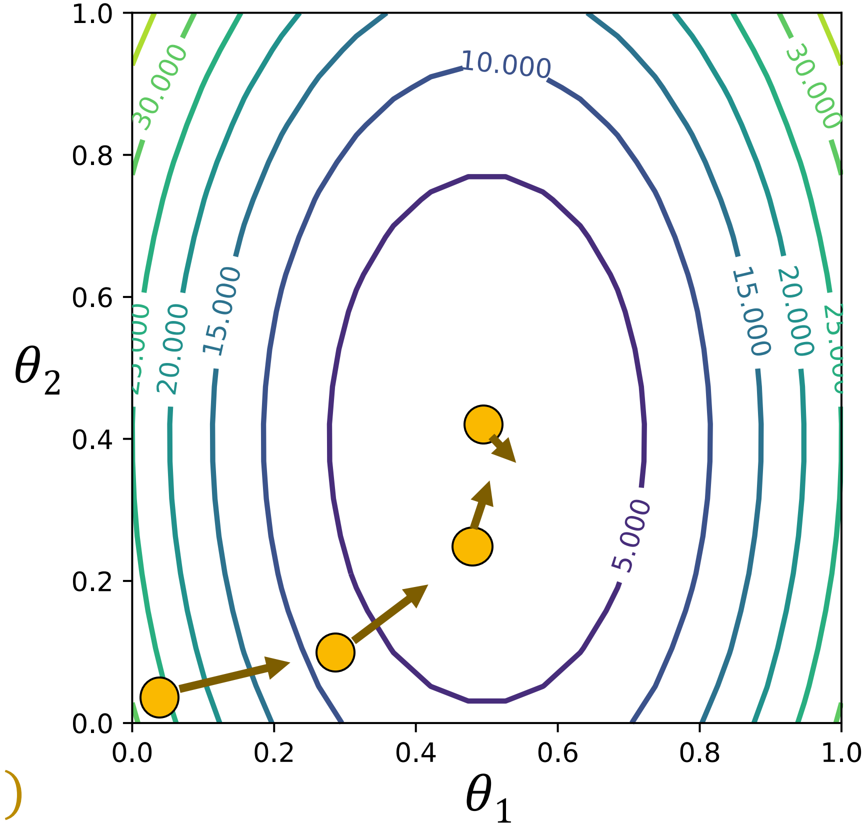
$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \gamma \mathbf{g}$$

# Recall: Gradient Descent for Linear Regression

$$J(\theta_1, \theta_2) = \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - \boldsymbol{\theta}^T \boldsymbol{x}^{(i)}\right)^2$$



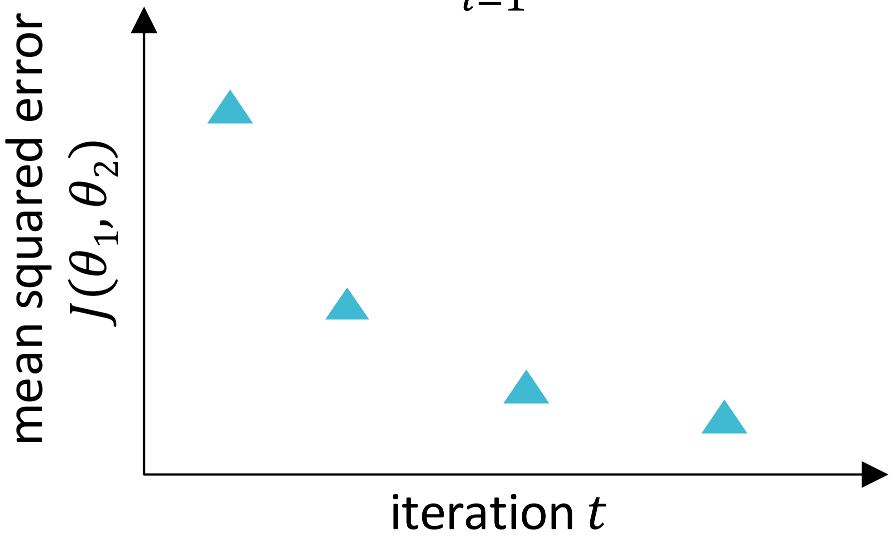



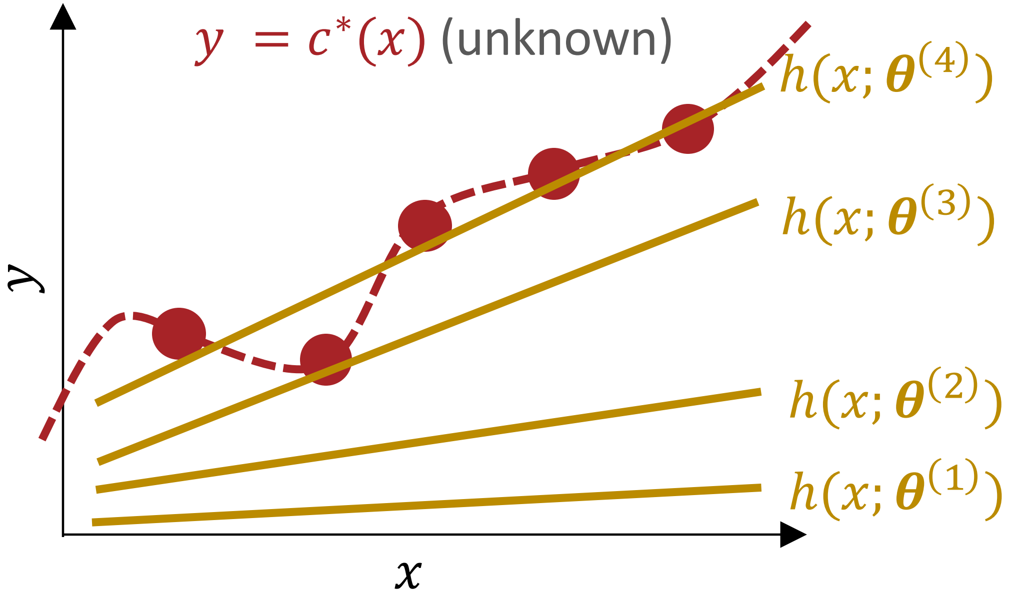



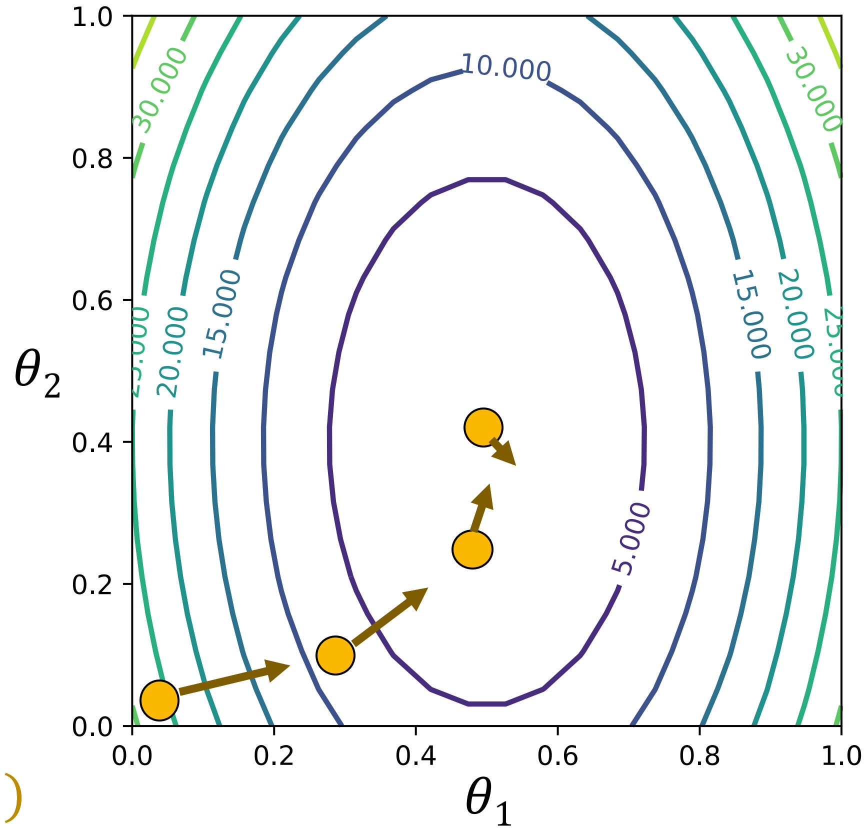

| $t$ | $\theta_1$ | $\theta_2$ | $J(\theta_1, \theta_2)$ |
|---|---|---|---|
| 1 | 0.01 | 0.02 | 25.2 |
| 2 | 0.30 | 0.12 | 8.7 |
| 3 | 0.51 | 0.30 | 1.5 |
| 4 | 0.59 | 0.43 | 0.2 |

# Why Gradient Descent for Linear Regression?

$$J(\theta_1, \theta_2) = \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - \boldsymbol{\theta}^T \boldsymbol{x}^{(i)}\right)^2$$



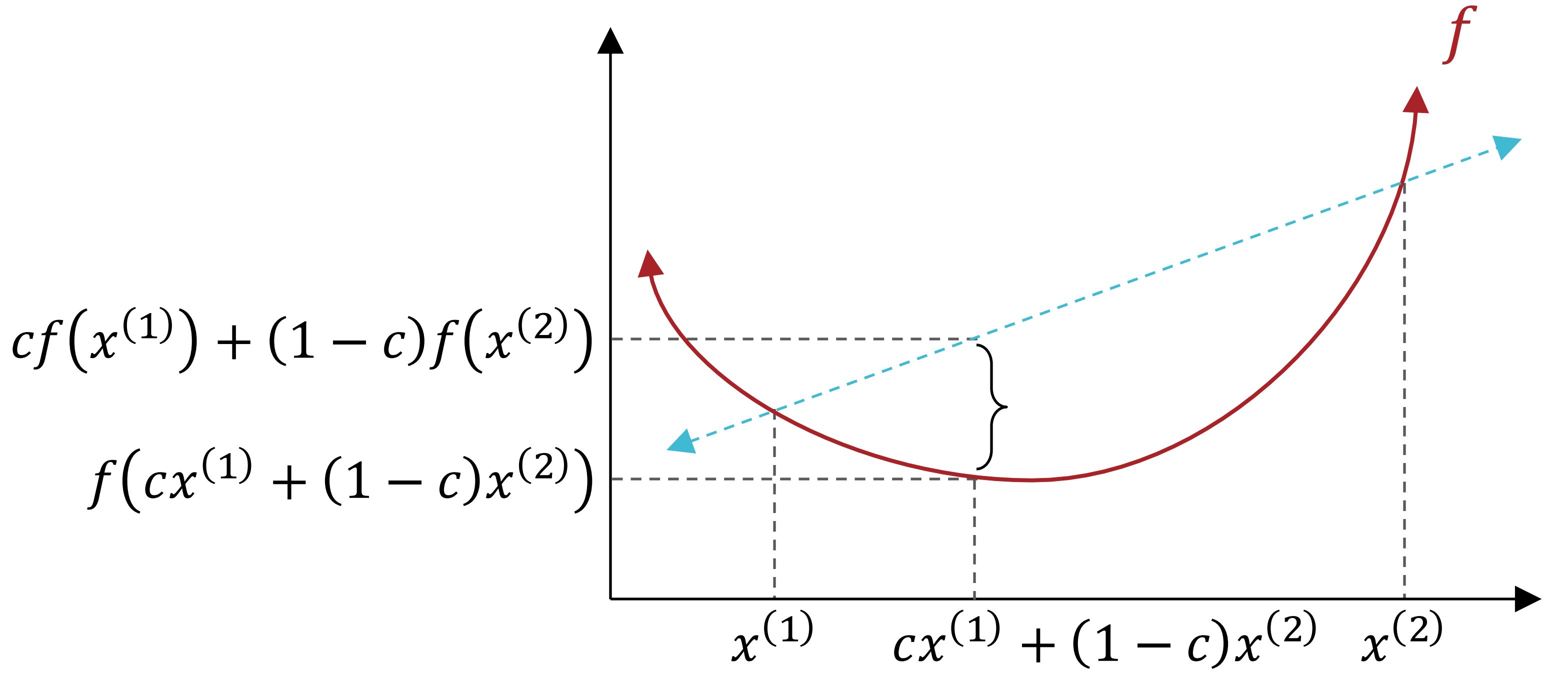




| $t$ | $\theta_1$ | $\theta_2$ | $J(\theta_1, \theta_2)$ |
|---|---|---|---|
| 1 | 0.01 | 0.02 | 25.2 |
| 2 | 0.30 | 0.12 | 8.7 |
| 3 | 0.51 | 0.30 | 1.5 |
| 4 | 0.59 | 0.43 | 0.2 |

# Convexity

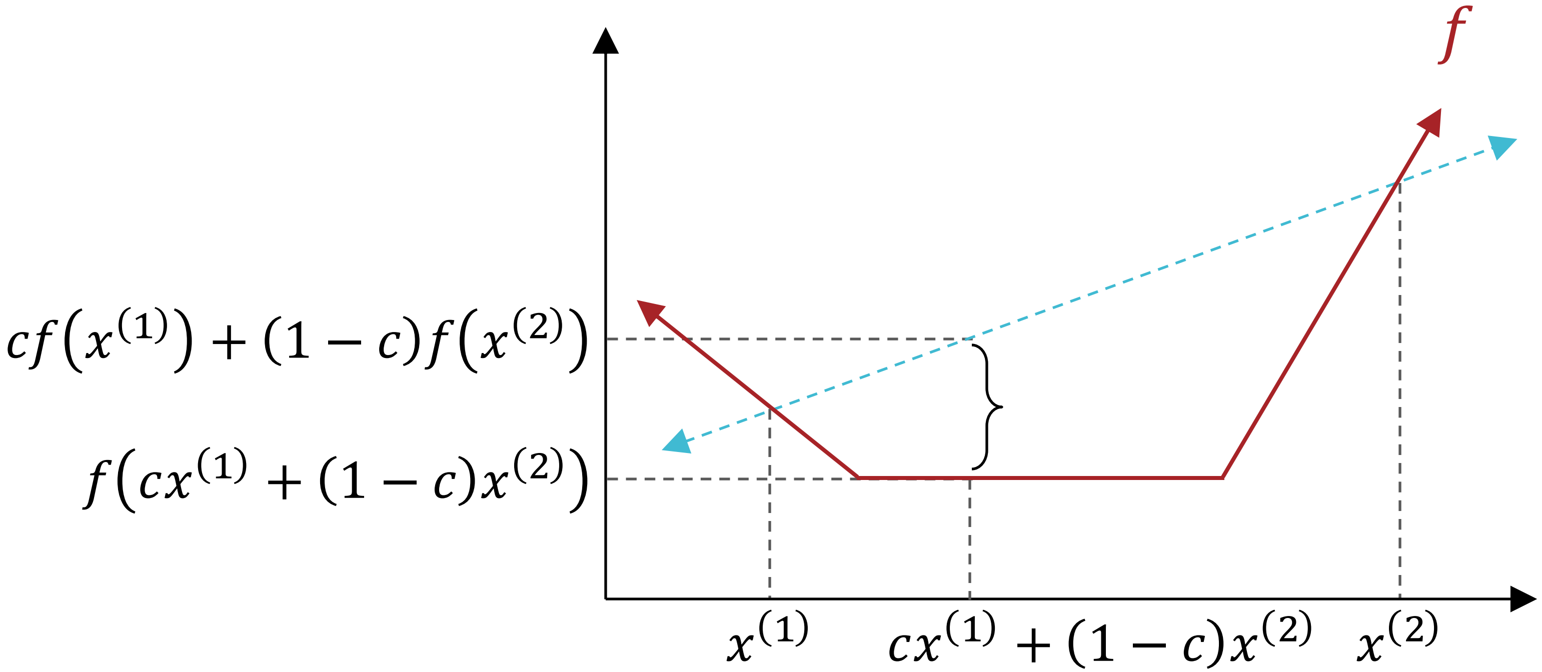
- A function $f: \mathbb{R}^D \rightarrow \mathbb{R}$ is convex if $\forall\, \boldsymbol{x}^{(1)} \in \mathbb{R}^D, \boldsymbol{x}^{(2)} \in \mathbb{R}^D$ and $0 \leq c \leq 1$

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



# Convexity

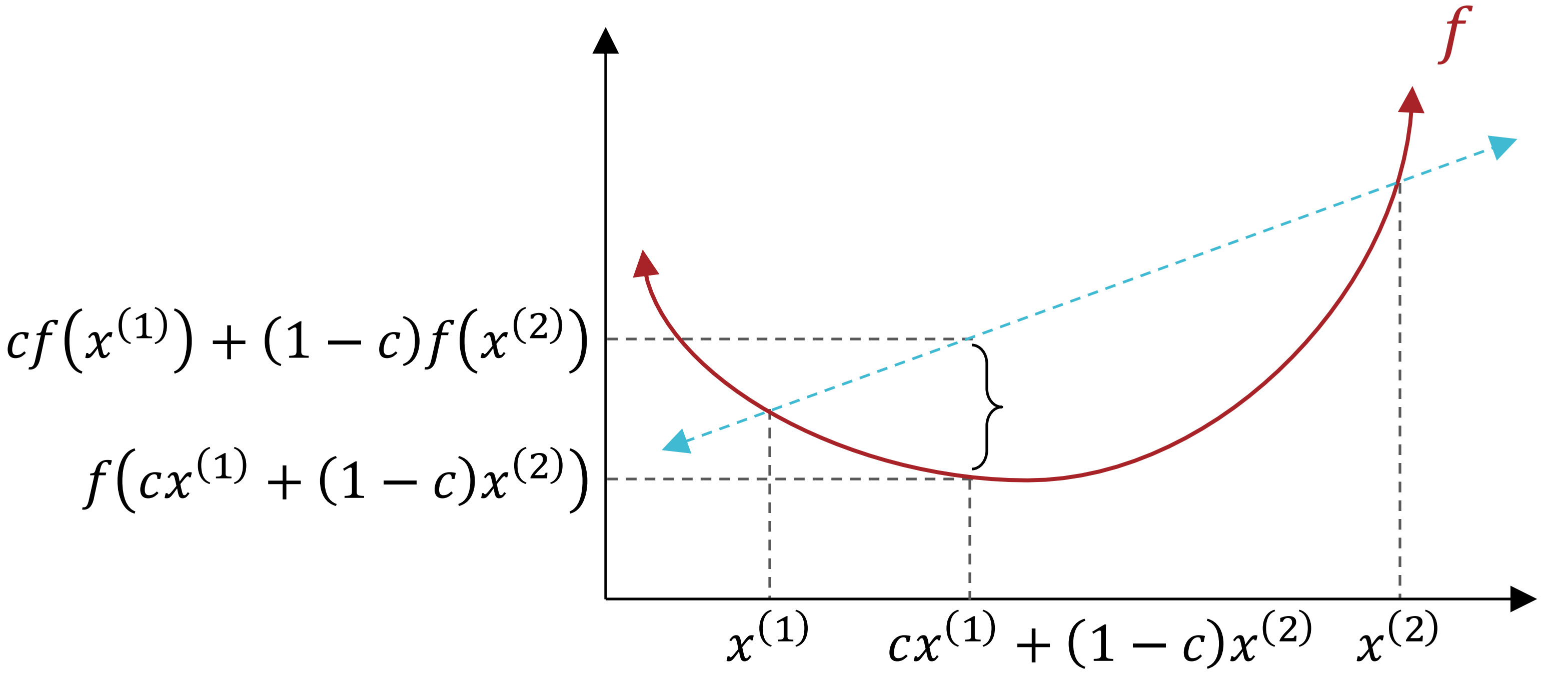
- A function $f: \mathbb{R}^D \to \mathbb{R}$ is convex if
  $\forall\, \boldsymbol{x}^{(1)} \in \mathbb{R}^D, \boldsymbol{x}^{(2)} \in \mathbb{R}^D$ and $0 \le c \le 1$
  $$f\left(c\boldsymbol{x}^{(1)} + (1-c)\boldsymbol{x}^{(2)}\right) \le cf\left(\boldsymbol{x}^{(1)}\right) + (1-c)f\left(\boldsymbol{x}^{(2)}\right)$$



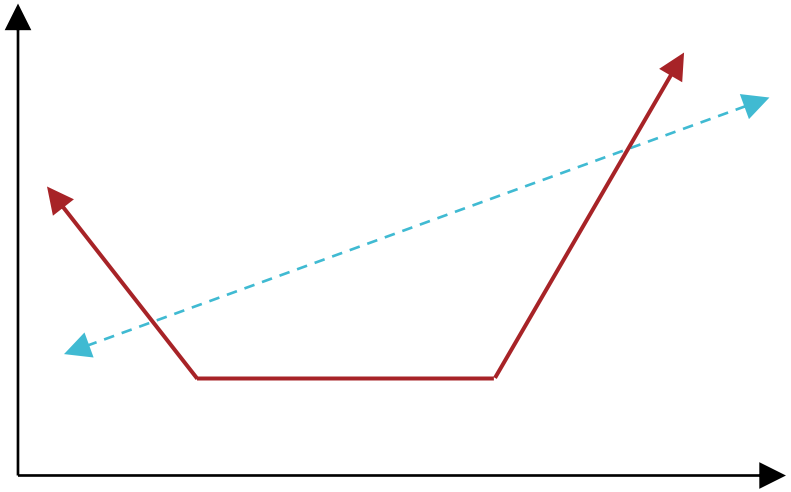
# Convexity

- 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 $0 < c < 1$

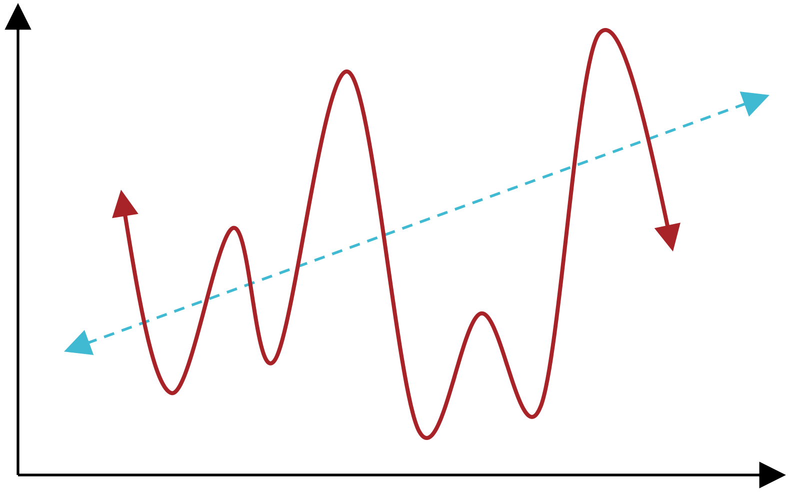
$$f\left(c\boldsymbol{x}^{(1)} + (1-c)\boldsymbol{x}^{(2)}\right) < cf\left(\boldsymbol{x}^{(1)}\right) + (1-c)f\left(\boldsymbol{x}^{(2)}\right)$$



# Convexity

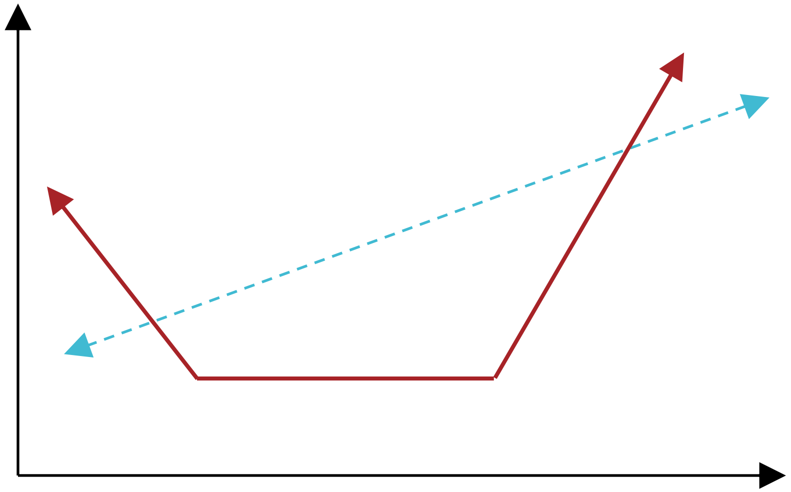


Convex functions



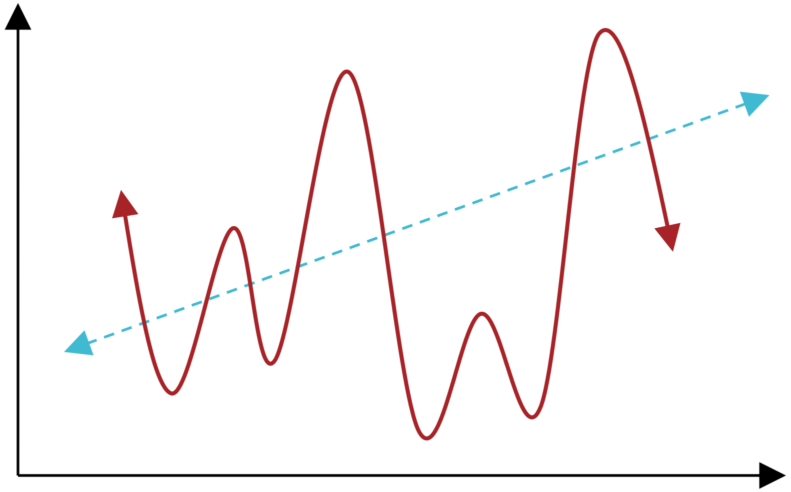
Non-convex functions

# Convexity



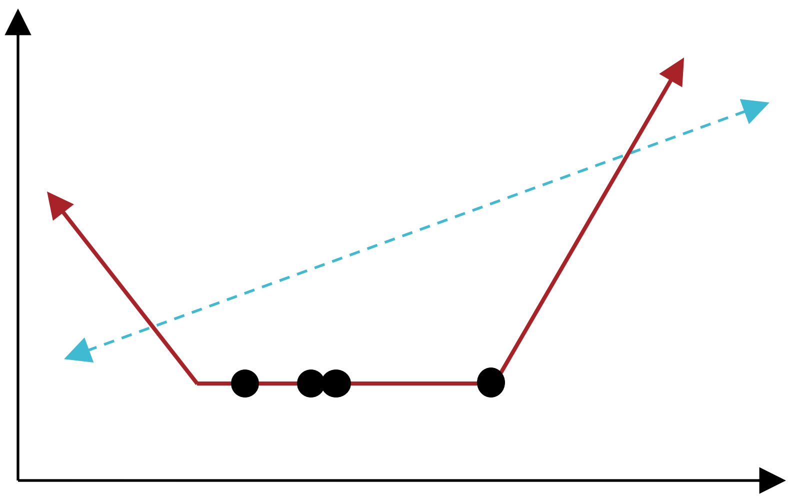
Given a function $f: \mathbb{R}^D \to \mathbb{R}$

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



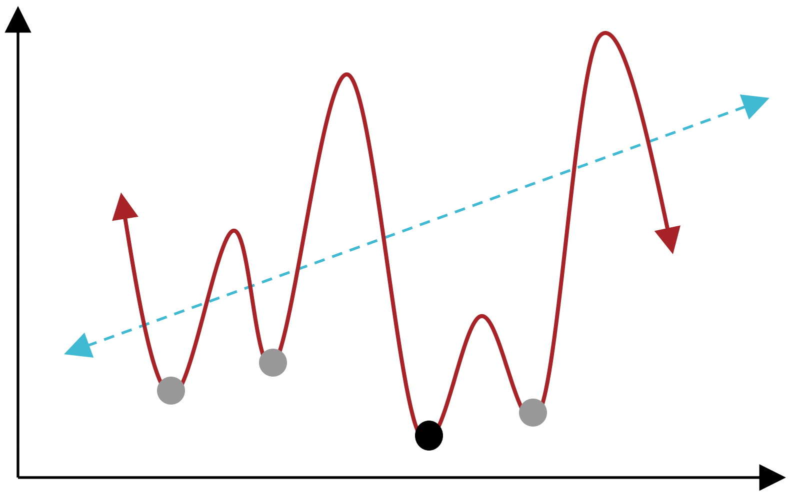
- $\boldsymbol{x}^*$ is a local minimum iff $\exists \; \epsilon$ s.t. $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x}) \; \forall \; \boldsymbol{x}$ s.t. $\|\boldsymbol{x} - \boldsymbol{x}^*\|_2 < \epsilon$

# Convexity



Convex functions:

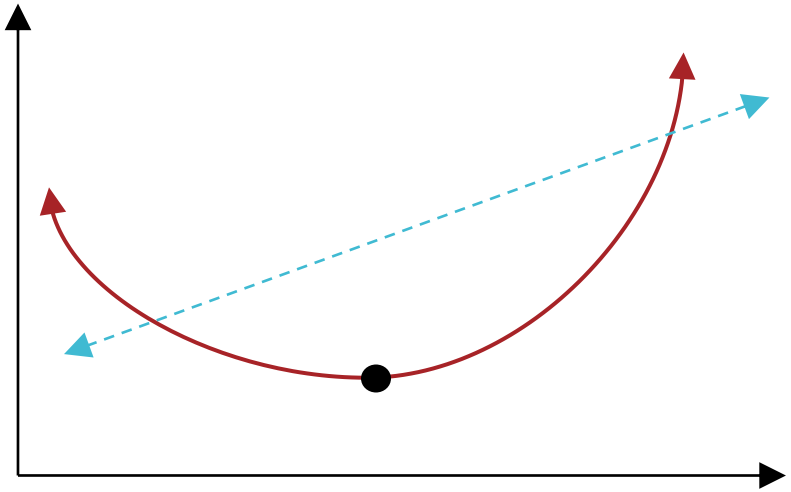
Each local minimum is a global minimum!



Non-convex functions:

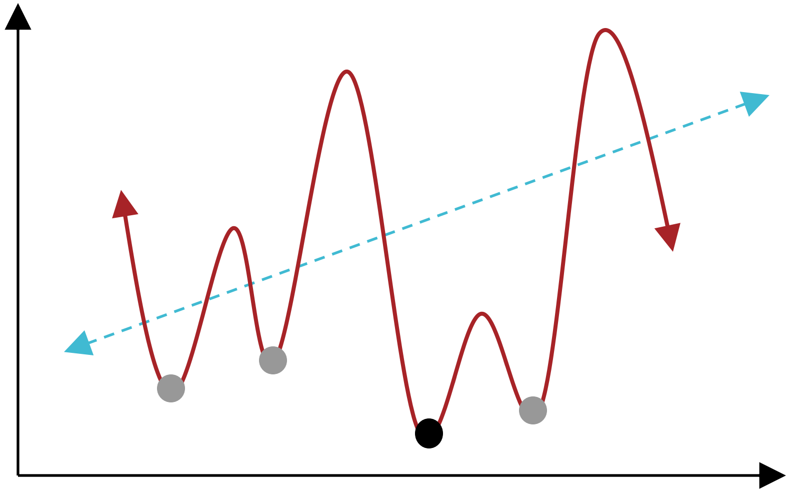
A local minimum may or may not be a global minimum…

# Convexity



Strictly convex functions:

There exists a unique global minimum!

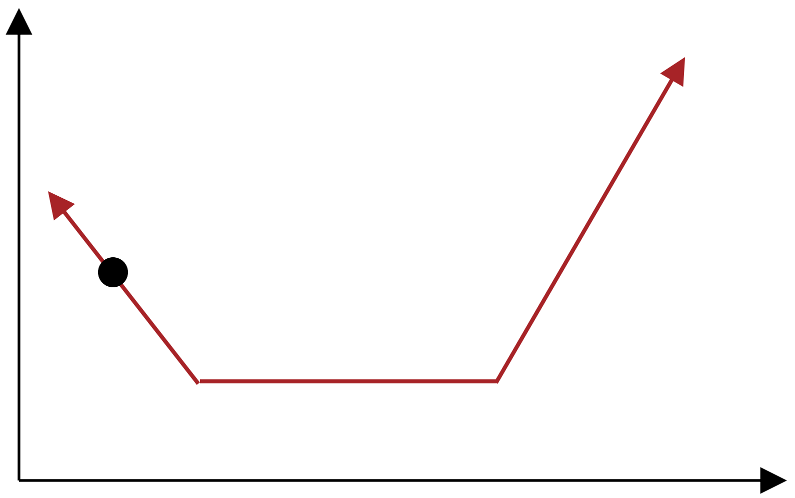


Non-convex functions:

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

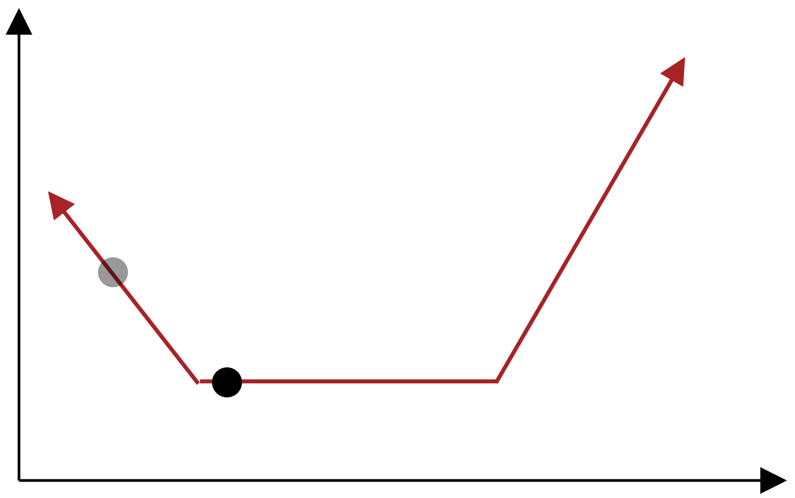
# Gradient Descent & Convexity

- 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!



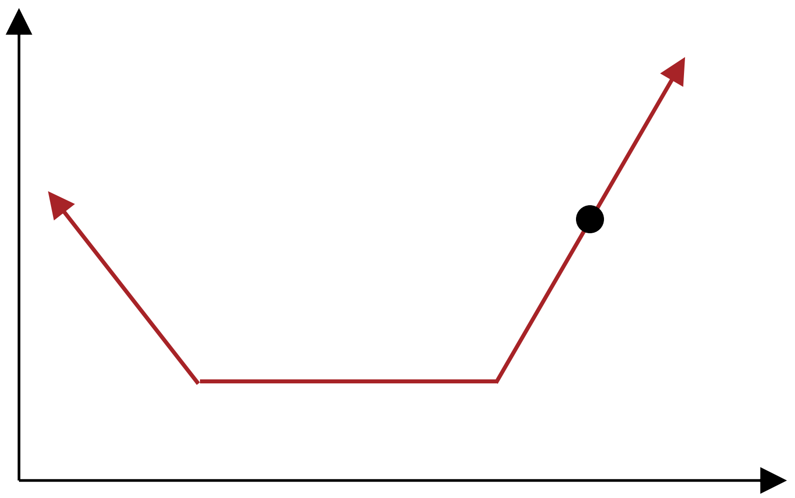
# Gradient Descent & Convexity

- 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!



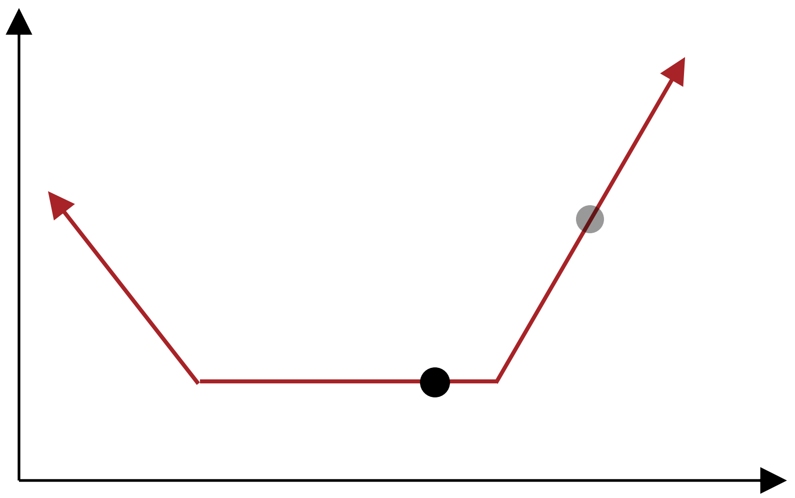
# Gradient Descent & Convexity

- 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!



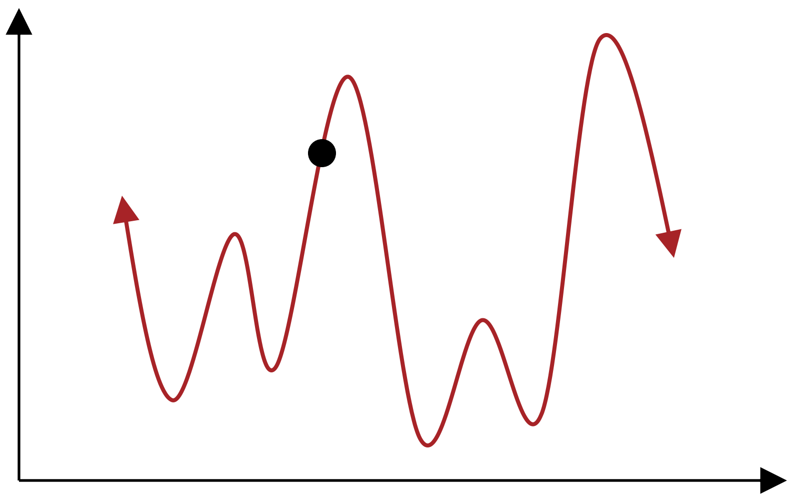
# Gradient Descent & Convexity

- 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!



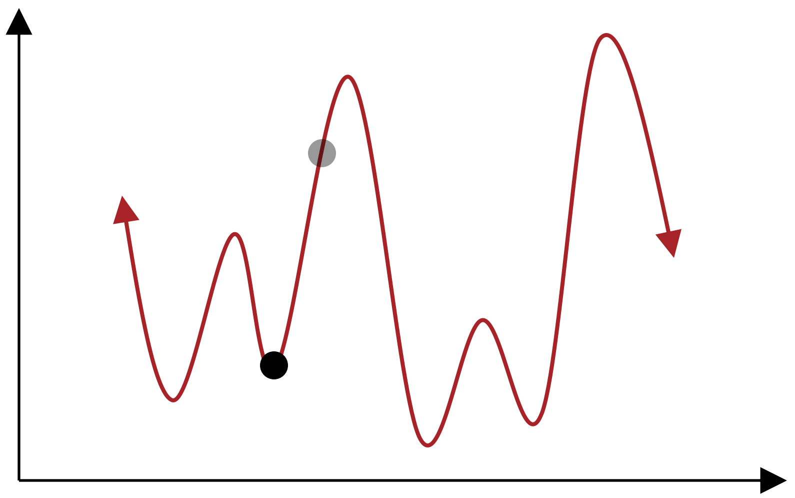
# Gradient Descent & Convexity

- 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…



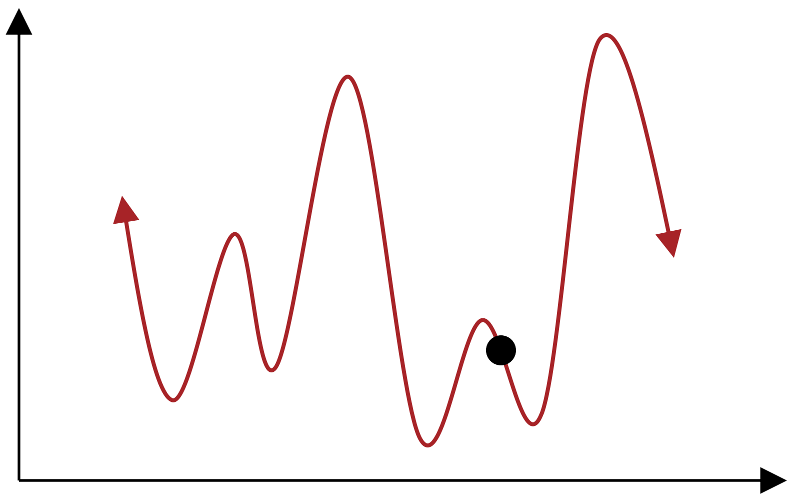
# Gradient Descent & Convexity

- 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…



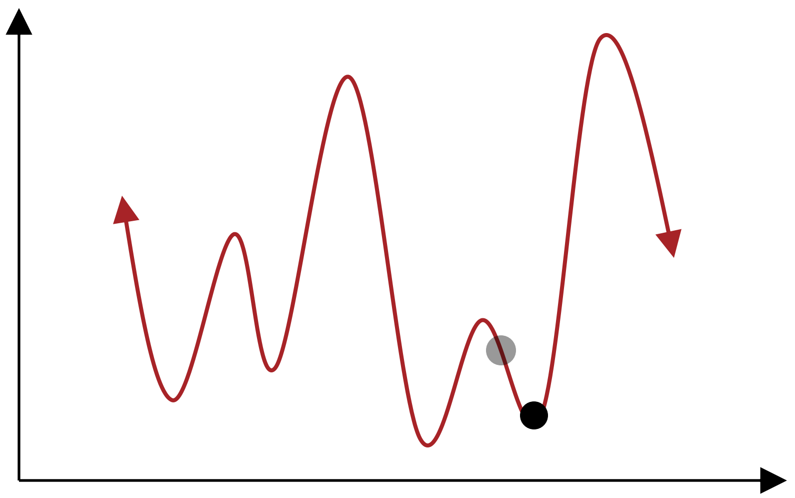
# Gradient Descent & Convexity

- 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…



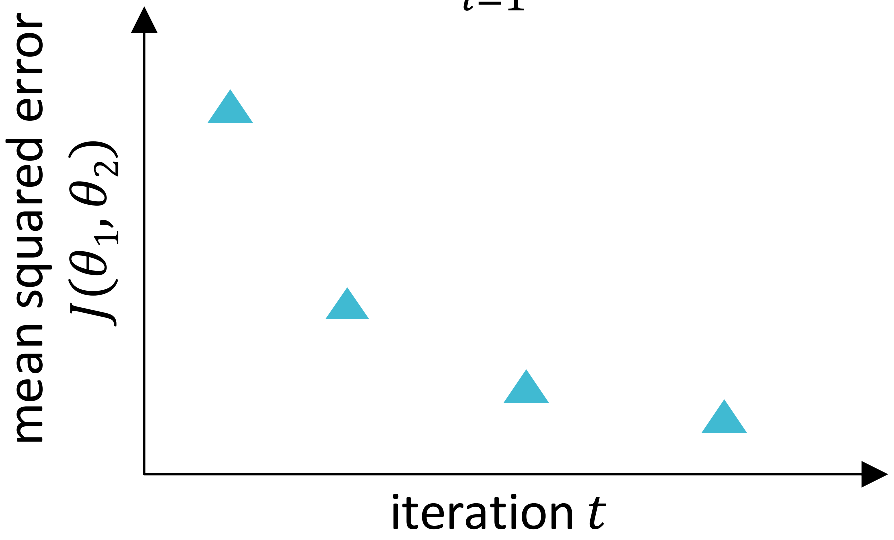
# Gradient Descent & Convexity

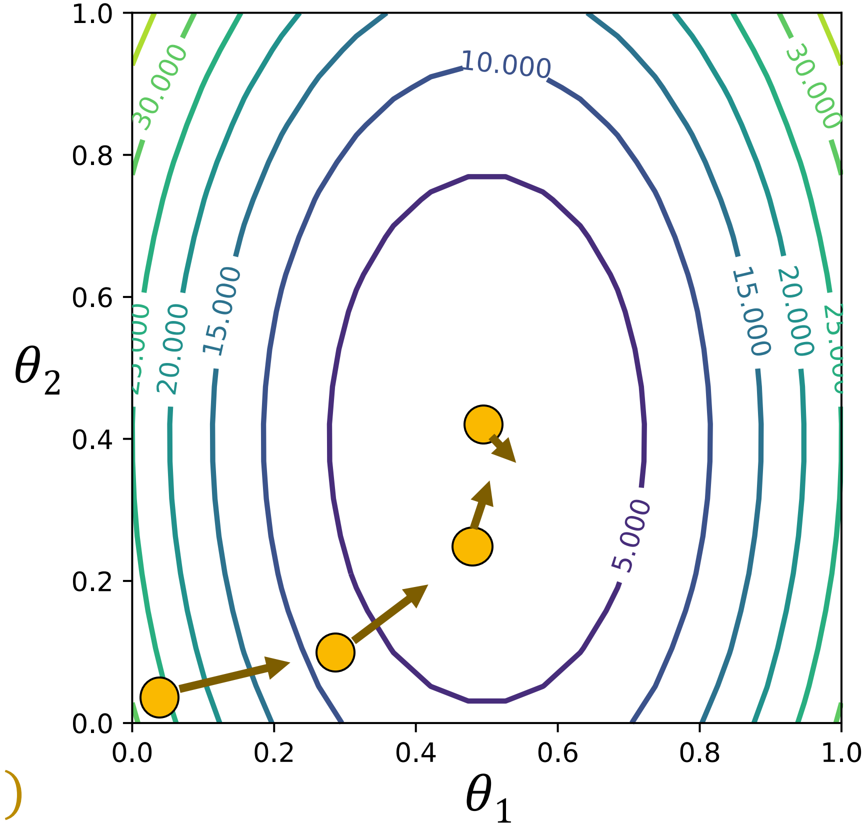
- Gradient descent is a local optimization algorithm – it will converge to a local minimum (if it converges)
  - Not ideal if the objective function is non-convex…

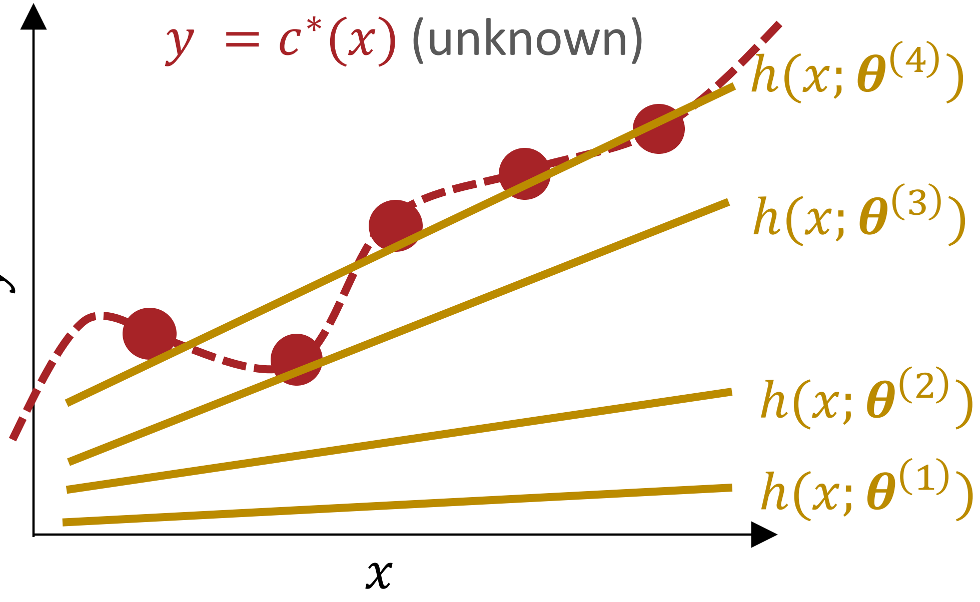


# Why Gradient Descent for Linear Regression?

$$J(\theta_1, \theta_2) = \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - \boldsymbol{\theta}^T \boldsymbol{x}^{(i)}\right)^2$$



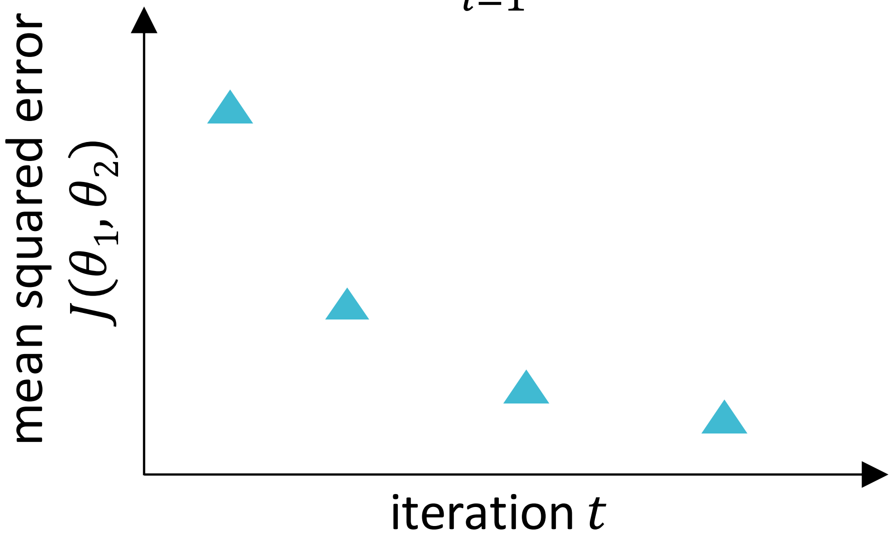



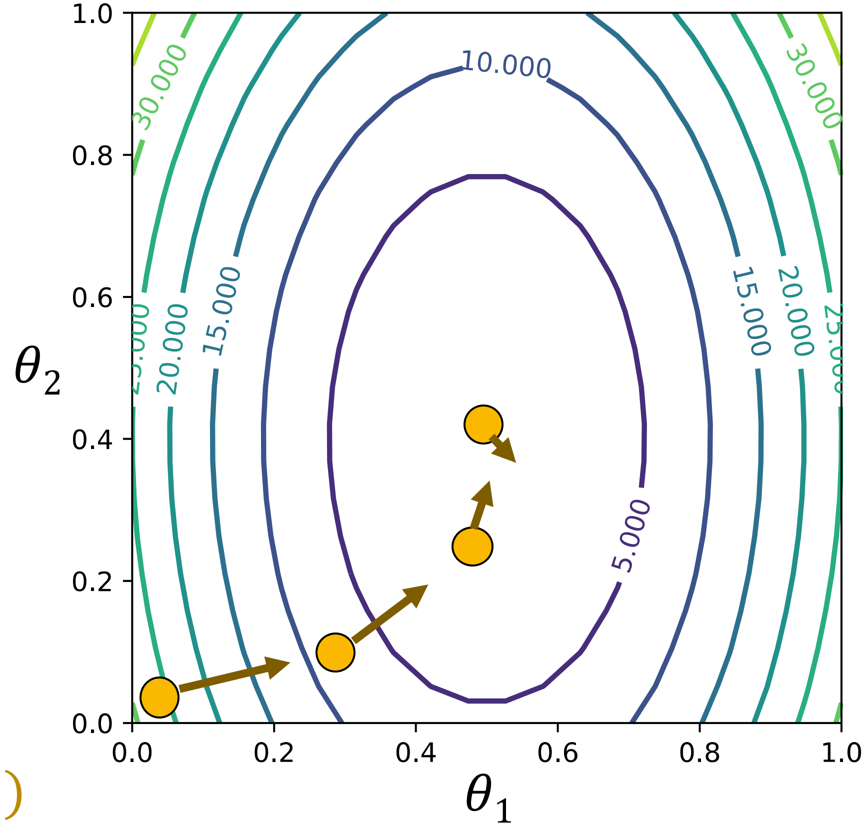



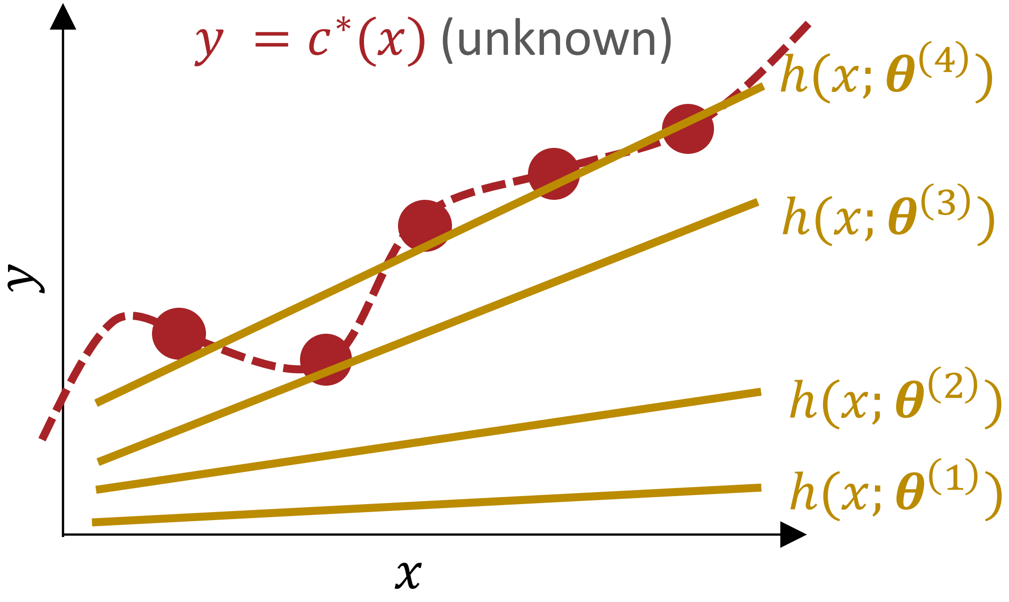

| $t$ | $\theta_1$ | $\theta_2$ | $J(\theta_1, \theta_2)$ |
|---|---|---|---|
| 1 | 0.01 | 0.02 | 25.2 |
| 2 | 0.30 | 0.12 | 8.7 |
| 3 | 0.51 | 0.30 | 1.5 |
| 4 | 0.59 | 0.43 | 0.2 |

# The mean squared error is convex (but not always strictly convex)

$$J(\theta_1, \theta_2) = \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - \boldsymbol{\theta}^T \boldsymbol{x}^{(i)}\right)^2$$



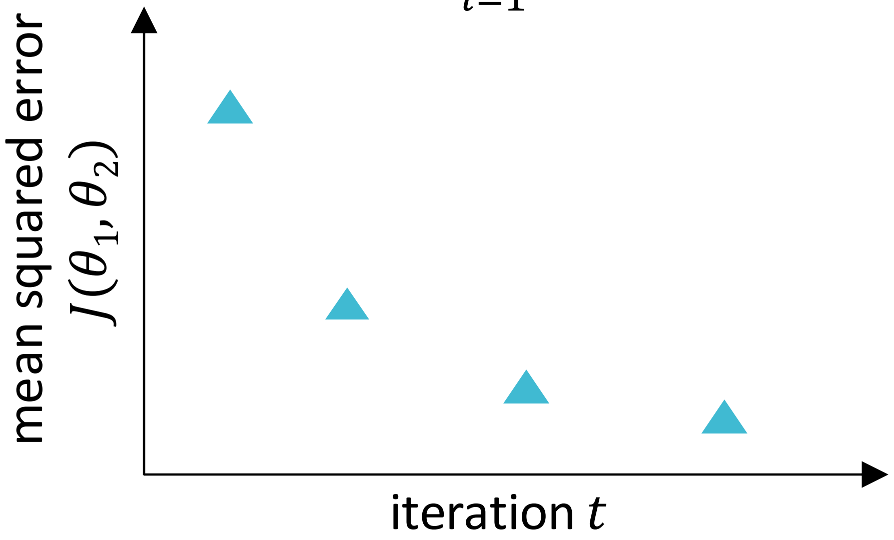



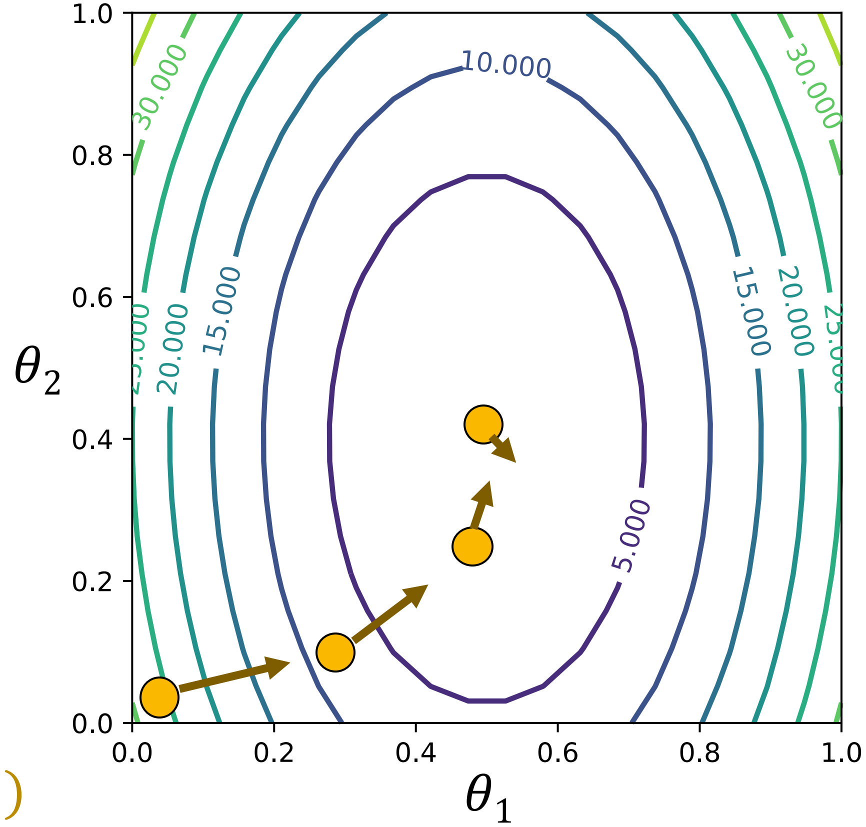

| $t$ | $\theta_1$ | $\theta_2$ | $J(\theta_1, \theta_2)$ |
|---|---|---|---|
| 1 | 0.01 | 0.02 | 25.2 |
| 2 | 0.30 | 0.12 | 8.7 |
| 3 | 0.51 | 0.30 | 1.5 |
| 4 | 0.59 | 0.43 | 0.2 |

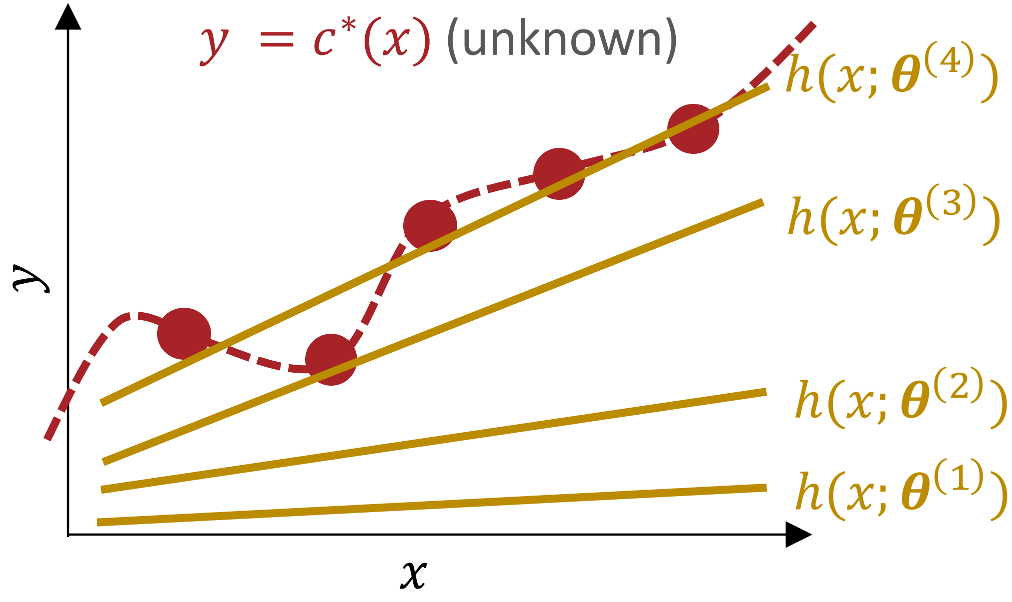
Okay, fine but couldn't we do something simpler?

Yes! (sometimes)

$$J(\theta_1, \theta_2) = \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - \boldsymbol{\theta}^T \boldsymbol{x}^{(i)}\right)^2$$




| $t$ | $\theta_1$ | $\theta_2$ | $J(\theta_1, \theta_2)$ |
|---|---|---|---|
| 1 | 0.01 | 0.02 | 25.2 |
| 2 | 0.30 | 0.12 | 8.7 |
| 3 | 0.51 | 0.30 | 1.5 |
| 4 | 0.59 | 0.43 | 0.2 |

# Closed Form Optimization

- Idea: find the *critical points* of the objective function, specifically the ones where $\nabla J(\theta) = \mathbf{0}$ (the vector of all zeros), and check if any of them are local minima

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

# Minimizing the Mean Squared Error

$$J(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}\left(y^{(i)} - \boldsymbol{\theta}^T\boldsymbol{x}^{(i)}\right)^2 = \frac{1}{2N}\sum_{i=1}^{N}\left({\boldsymbol{x}^{(i)}}^T\boldsymbol{\theta} - y^{(i)}\right)^2$$

$$= \frac{1}{2N}(X\theta - \boldsymbol{y})^T(X\theta - \boldsymbol{y})$$

$$= \frac{1}{2N}(\boldsymbol{\theta}^T X^T X\boldsymbol{\theta} - 2\boldsymbol{\theta}^T X^T\boldsymbol{y} + \boldsymbol{y}^T\boldsymbol{y})$$

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{1}{2N}(2X^T X\boldsymbol{\theta} - 2X^T\boldsymbol{y})$$

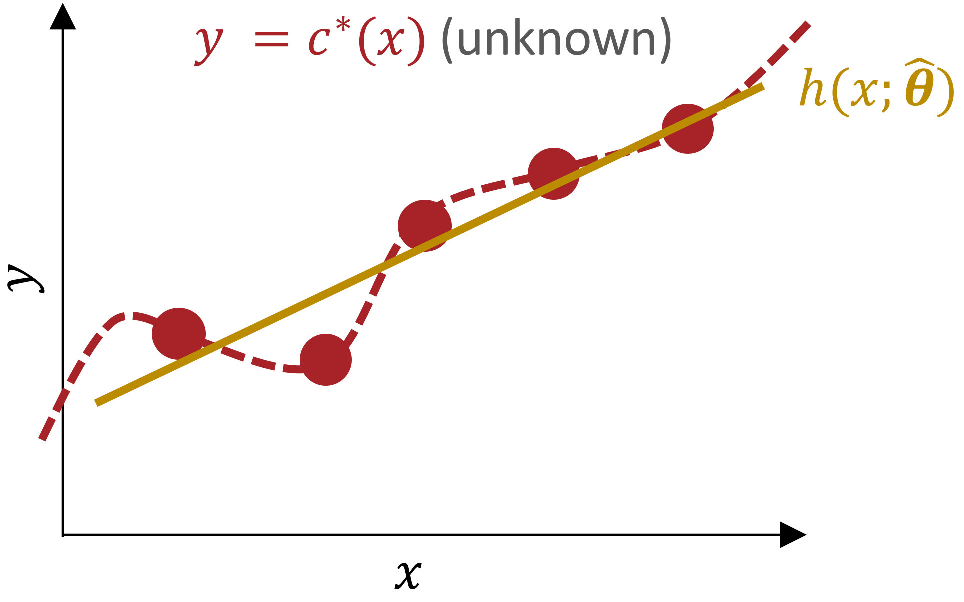
$$\nabla_{\boldsymbol{\theta}} J(\widehat{\boldsymbol{\theta}}) = \frac{1}{2N}\left(2X^T X\widehat{\boldsymbol{\theta}} - 2X^T\boldsymbol{y}\right) = 0$$

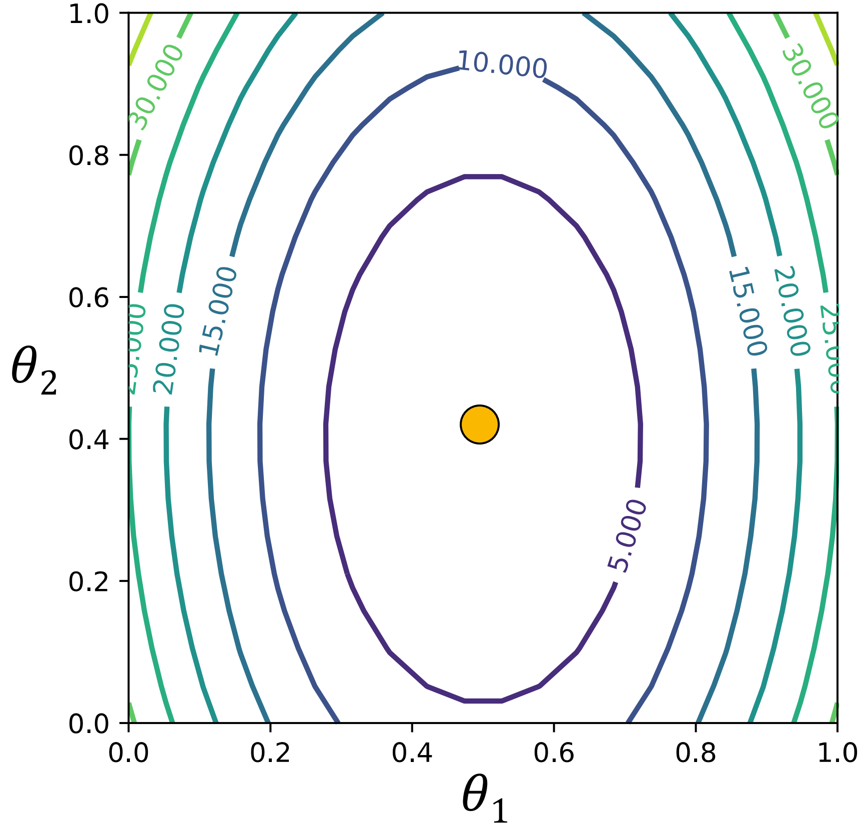
$$\rightarrow X^T X\widehat{\boldsymbol{\theta}} = X^T\boldsymbol{y}$$

$$\rightarrow \widehat{\boldsymbol{\theta}} = (X^T X)^{-1}X^T\boldsymbol{y}$$

# Closed Form Optimization

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




| $t$ | $\theta_1$ | $\theta_2$ | $J(\theta_1, \theta_2)$ |
|---|---|---|---|
| 1 | 0.59 | 0.43 | 0.2 |

# Closed Form Solution

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

1. Is $X^T X$ invertible?

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

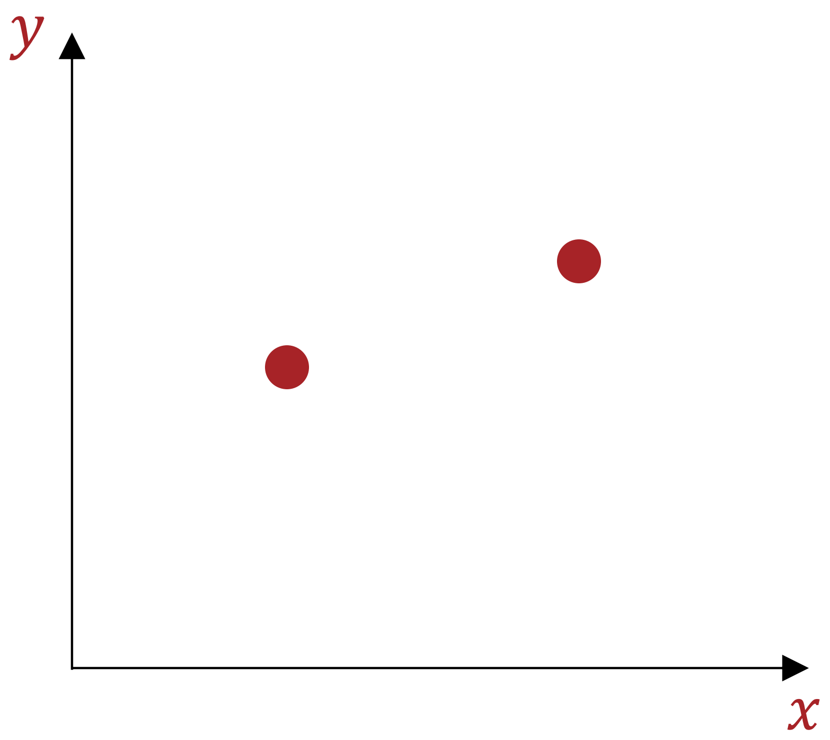
# Closed Form Solution

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

1. Is $X^T X$ invertible?
   - When $N \gg D + 1$, $X^T X$ is (almost always) full rank and therefore, invertible!
   - If $X^T X$ is not invertible (occurs when one of the features is a linear combination of the others) then there are either 0 or infinitely many solutions!
2. If so, how computationally expensive is inverting $X^T X$?

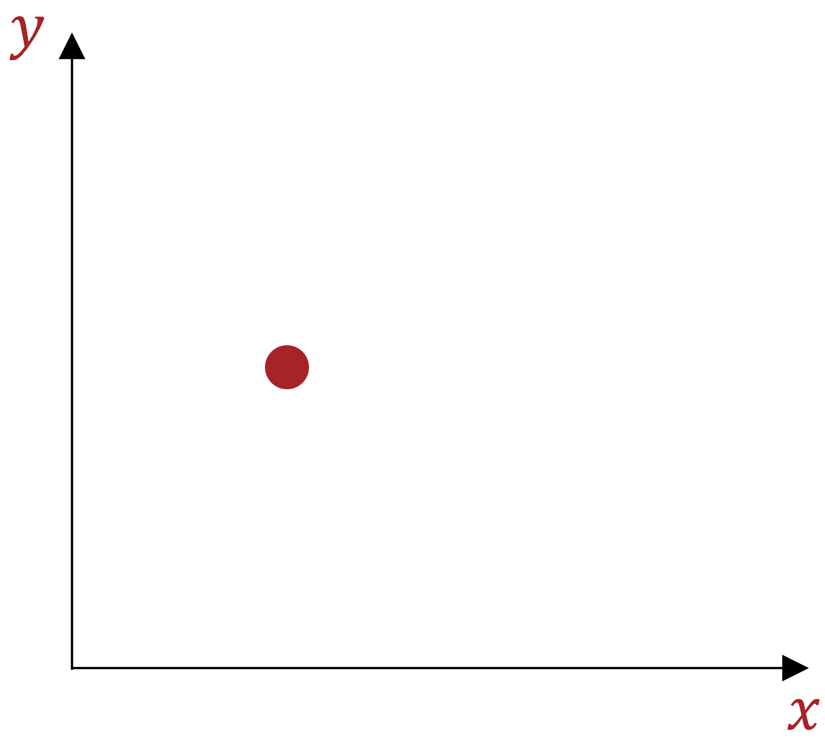
# Linear Regression: Uniqueness

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



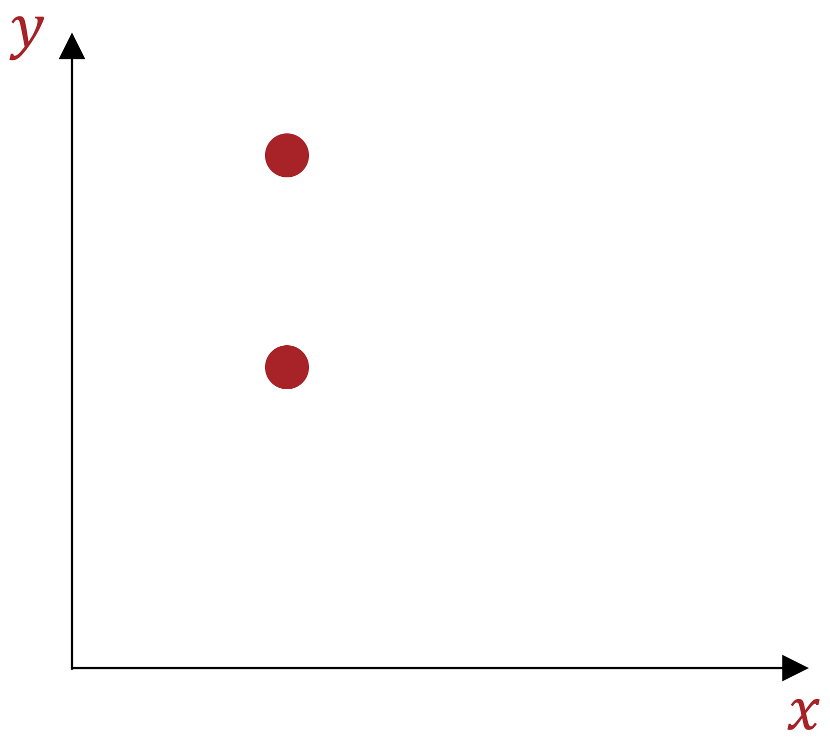
# Linear Regression: Uniqueness

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



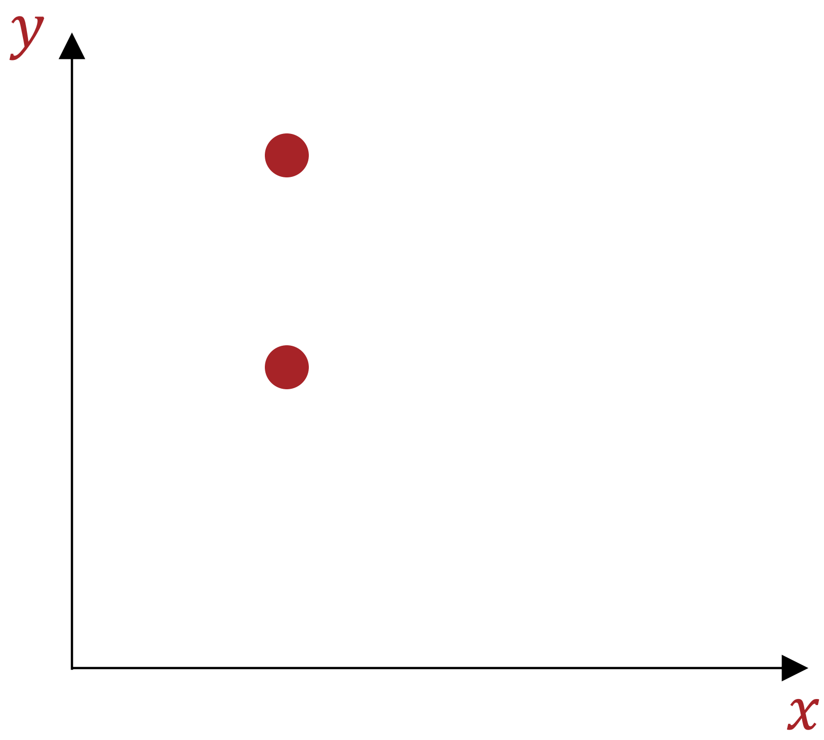
# Linear Regression: Uniqueness

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



# Poll Question 3

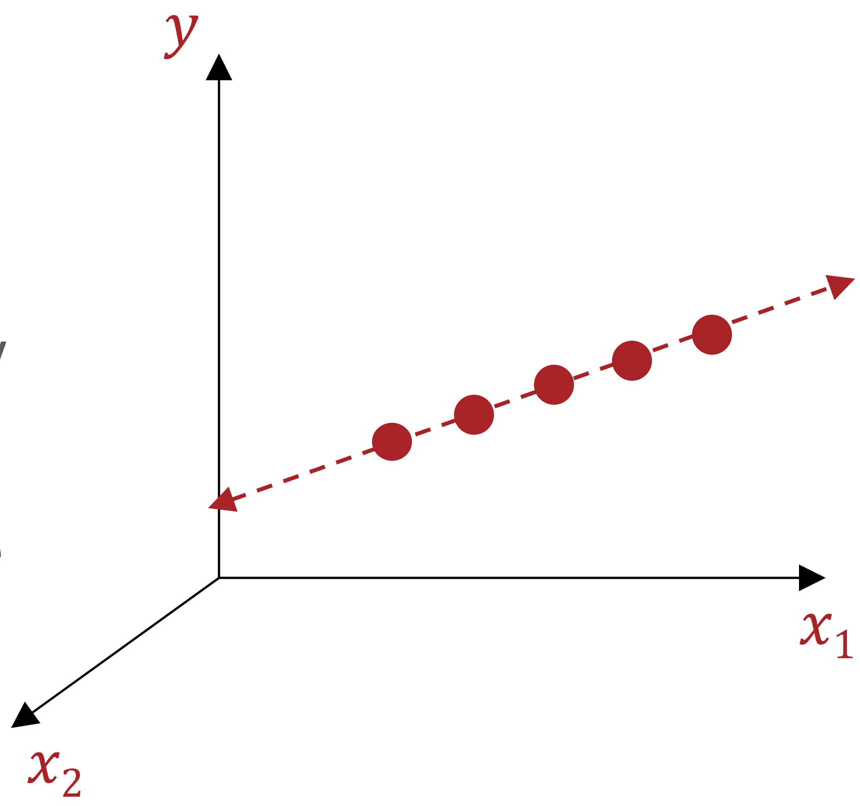
- Consider a 1D linear regression model trained to minimize the mean squared error: how many optimal solutions (i.e., sets of parameters $\theta$) are there for the given dataset?




A. -1 **(TOXIC)** B. 0 C. 1 D. 2 E. $\infty$

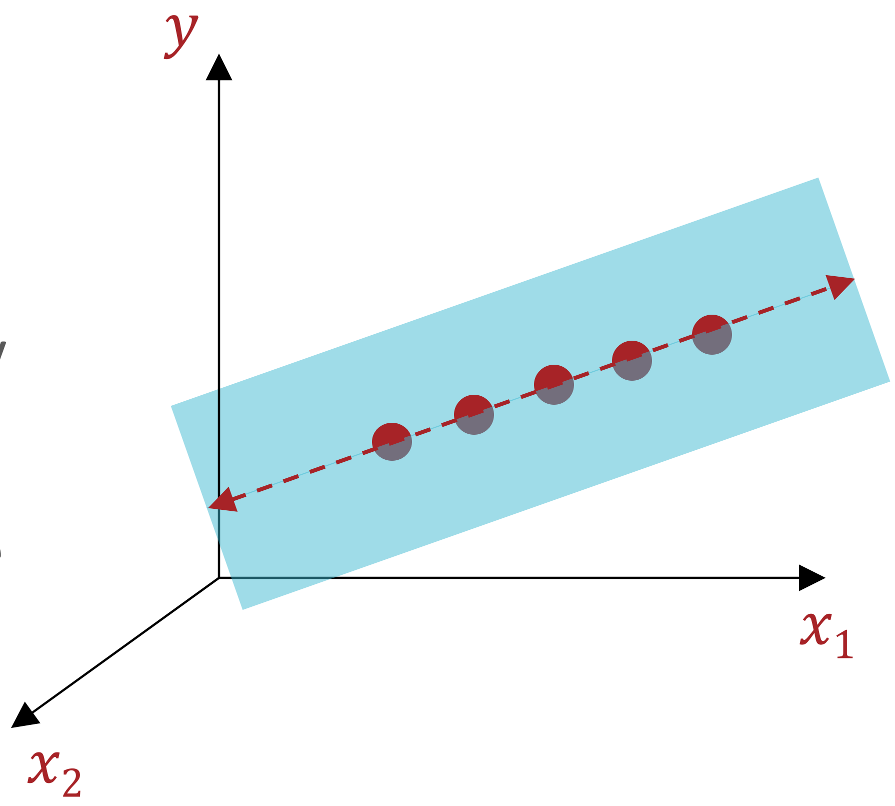
# Linear Regression: Uniqueness

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



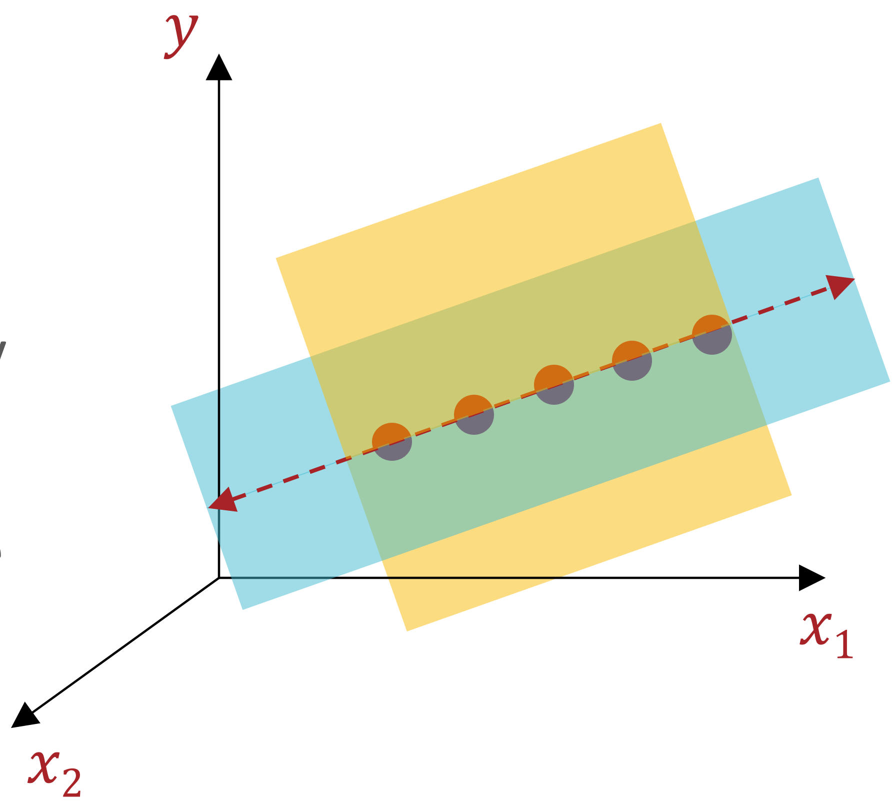
# Linear Regression: Uniqueness

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



# Closed Form Solution

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

1. Is $X^T X$ invertible?
   - When $N \gg D + 1$, $X^T X$ is (almost always) full rank and therefore, invertible!
   - If $X^T X$ is not invertible (occurs when one of the features is a linear combination of the others) then there are either 0 or infinitely many solutions
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 Learning Objectives

You should be able to…

- Design k-NN Regression and Decision Tree Regression
- Implement learning for Linear Regression using gradient descent or closed form optimization
- Choose a Linear Regression optimization technique that is appropriate for a particular dataset by analyzing the tradeoff of computational complexity vs. convergence speed
- Identify situations where least squares regression has exactly one solution or infinitely many solutions