CS182: Introduction to Machine Learning – Optimization for Machine Learning

Yujiao Shi
SIST, ShanghaiTech
Spring, 2025

Slides Courtesy of Matt Gormley & Henry Chai, CMU

Linear Regression as Function Approximation

Recall...

$\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^N$
where $\mathbf{x} \in \mathbb{R}^M$ and $y \in \mathbb{R}$

1. Assume $\mathcal{D}$ generated as:

$$\mathbf{x}^{(i)} \sim p^*(\cdot)$$
$$y^{(i)} = h^*(\mathbf{x}^{(i)})$$

2. Choose hypothesis space, $\mathcal{H}$:
all linear functions in M-dimensional space

$$\mathcal{H} = \{h_{\boldsymbol{\theta}} : h_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}^T\mathbf{x}, \boldsymbol{\theta} \in \mathbb{R}^M\}$$

3. Choose an objective function:
mean squared error (MSE)

$$\begin{aligned} J(\boldsymbol{\theta}) &= \frac{1}{N}\sum_{i=1}^N e_i^2 \\ &= \frac{1}{N}\sum_{i=1}^N \left(y^{(i)} - h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)})\right)^2 \\ &= \frac{1}{N}\sum_{i=1}^N \left(y^{(i)} - \boldsymbol{\theta}^T\mathbf{x}^{(i)}\right)^2 \end{aligned}$$

4. Solve the unconstrained optimization problem via favorite method:
   - *gradient descent*
   - *closed form*
   - *stochastic gradient descent*
   - $\ldots$

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\text{argmin}}\, J(\boldsymbol{\theta})$$

5. Test time: given a new $\mathbf{x}$, make prediction $\hat{y}$

$$\hat{y} = h_{\hat{\boldsymbol{\theta}}}(\mathbf{x}) = \hat{\boldsymbol{\theta}}^T\mathbf{x}$$

Gradient Calculation for Linear Regression

Recall...

Derivative of $J^{(i)}(\boldsymbol{\theta})$:

$$\begin{aligned}\frac{d}{d\theta_k} J^{(i)}(\boldsymbol{\theta}) &= \frac{d}{d\theta_k} \frac{1}{2} (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)})^2 \\ &= \frac{1}{2} \frac{d}{d\theta_k} (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)})^2 \\ &= (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) \frac{d}{d\theta_k} (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) \\ &= (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) \frac{d}{d\theta_k} \left(\sum_{j=1}^{K} \theta_j x_j^{(i)} - y^{(i)} \right) \\ &= (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) x_k^{(i)}\end{aligned}$$

Derivative of $J(\boldsymbol{\theta})$:

$$\begin{aligned}\frac{d}{d\theta_k} J(\boldsymbol{\theta}) &= \frac{1}{N} \sum_{i=1}^{N} \frac{d}{d\theta_k} J^{(i)}(\boldsymbol{\theta}) \\ &= \frac{1}{N} \sum_{i=1}^{N} (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) x_k^{(i)}\end{aligned}$$

Gradient of $J^{(i)}(\boldsymbol{\theta})$ [used by SGD]

$$\nabla_{\boldsymbol{\theta}} J^{(i)}(\boldsymbol{\theta}) = \begin{bmatrix} \frac{d}{d\theta_1} J^{(i)}(\boldsymbol{\theta}) \\ \frac{d}{d\theta_2} J^{(i)}(\boldsymbol{\theta}) \\ \vdots \\ \frac{d}{d\theta_M} J^{(i)}(\boldsymbol{\theta}) \end{bmatrix} = \begin{bmatrix} (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) x_1^{(i)} \\ (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) x_2^{(i)} \\ \vdots \\ (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) x_N^{(i)} \end{bmatrix}$$
$$= (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) \mathbf{x}^{(i)}$$

Gradient of $J(\boldsymbol{\theta})$ [used by Gradient Descent]

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \begin{bmatrix} \frac{d}{d\theta_1} J(\boldsymbol{\theta}) \\ \frac{d}{d\theta_2} J(\boldsymbol{\theta}) \\ \vdots \\ \frac{d}{d\theta_M} J(\boldsymbol{\theta}) \end{bmatrix} = \begin{bmatrix} \frac{1}{N} \sum_{i=1}^{N} (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) x_1^{(i)} \\ \frac{1}{N} \sum_{i=1}^{N} (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) x_2^{(i)} \\ \frac{1}{N} \vdots \\ \frac{1}{N} \sum_{i=1}^{N} (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) x_M^{(i)} \end{bmatrix}$$
$$= \frac{1}{N} \sum_{i=1}^{N} (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) \mathbf{x}^{(i)}$$

Recall: Gradient Descent for Linear Regression

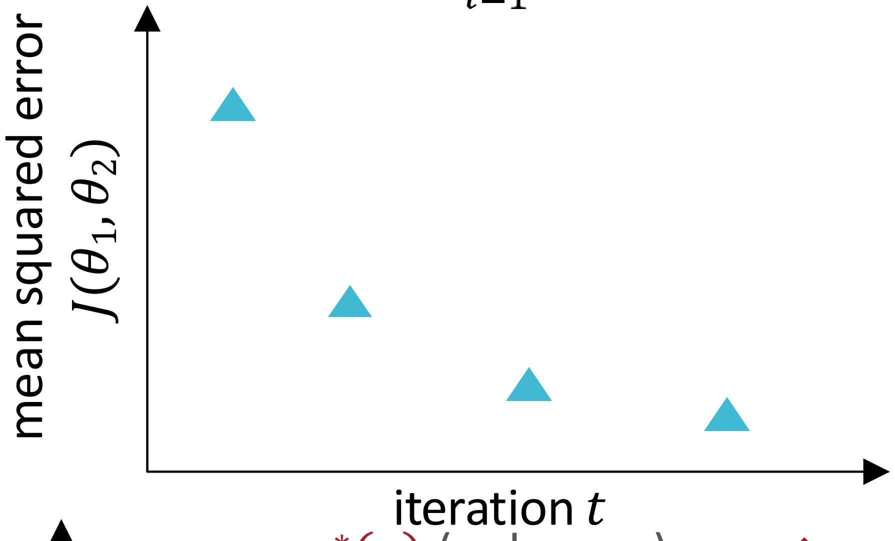
- Gradient descent for linear regression repeatedly takes steps opposite the gradient of the objective function

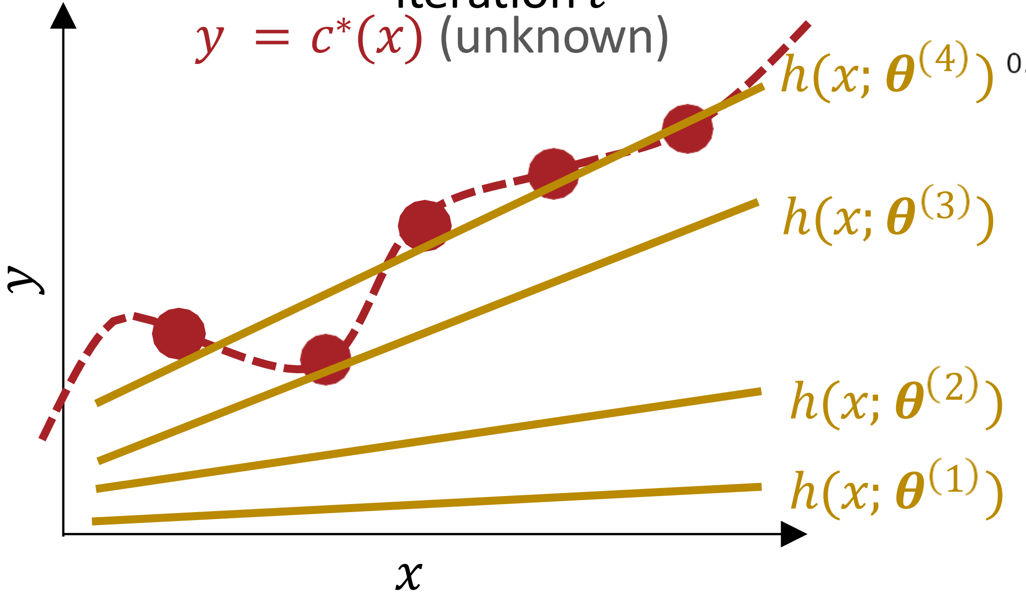
Algorithm 1 GD for Linear Regression

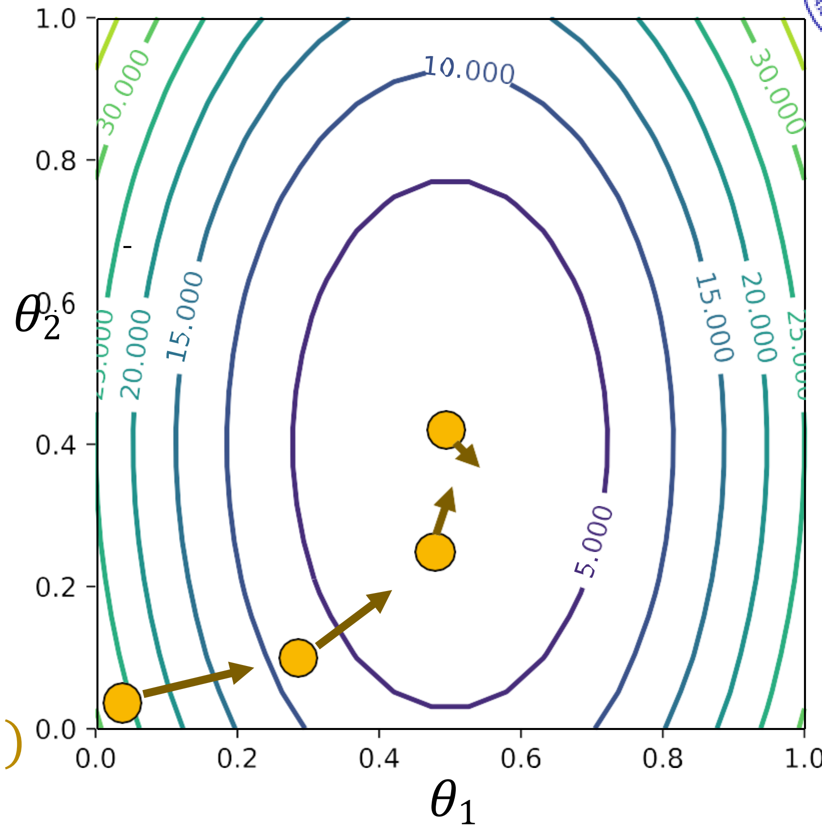
1: **procedure** GDLR($\mathcal{D}$, $\boldsymbol{\theta}^{(0)}$)
2: $\quad\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(0)}$ ▷ Initialize parameters
3: $\quad$**while** not converged **do**
4: $\quad\quad\mathbf{g} \leftarrow \sum_{i=1}^{N} (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)})\mathbf{x}^{(i)}$ ▷ Compute gradient
5: $\quad\quad\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \gamma \mathbf{g}$ ▷ Update parameters
6: $\quad$**return** $\boldsymbol{\theta}$

Recall: Gradient Descent for Linear Regression

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



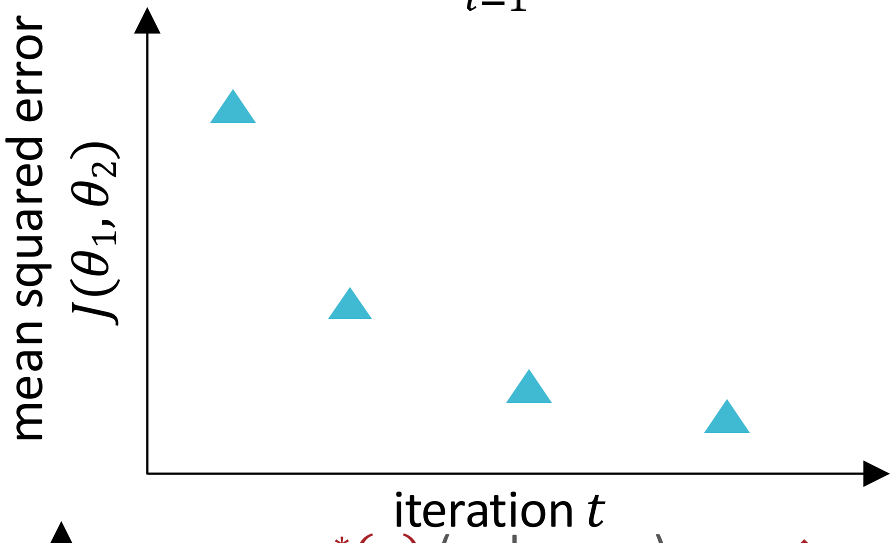


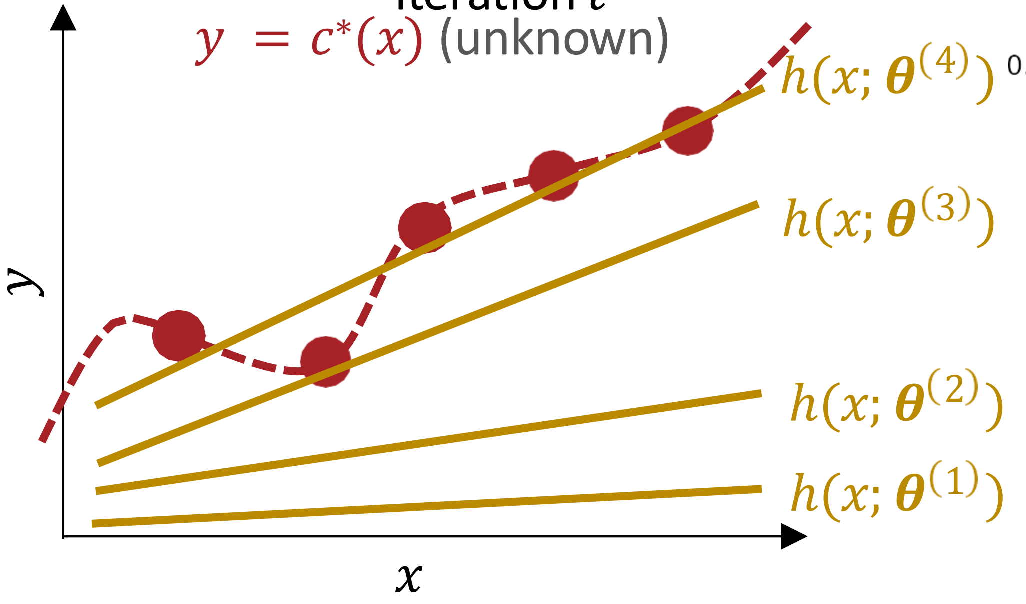


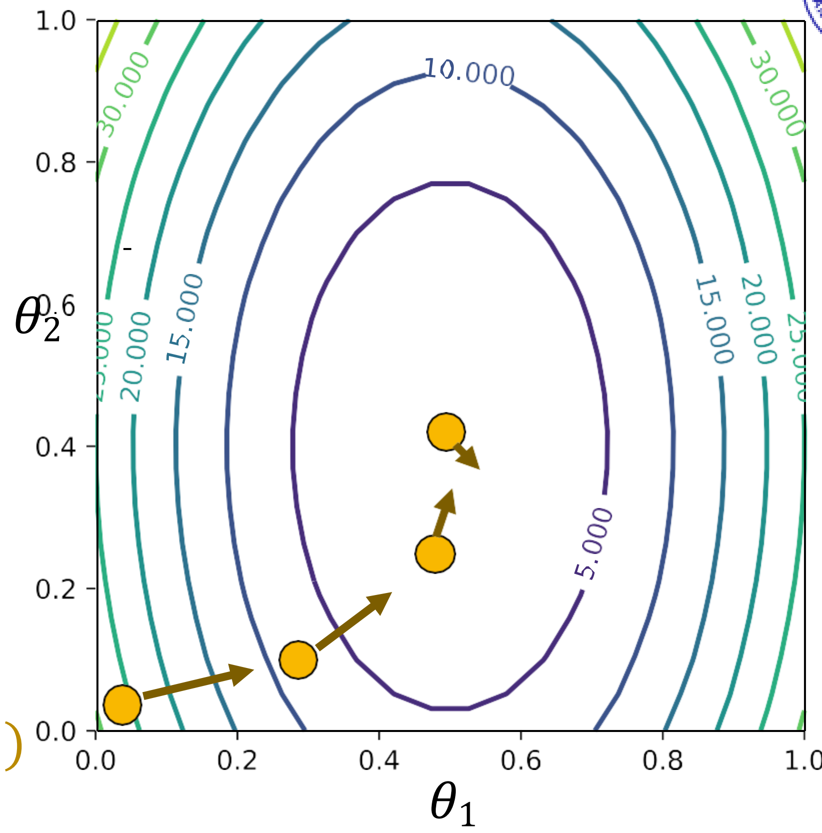
t	θ_1	θ_2	$J(\theta_1, \theta_2)$
1	0.01	0.02	25.2
2	0.30	0.12	8.7
3	0.51	0.30	1.5
4	0.59	0.43	0.2

Why Gradient Descent for Linear Regression?

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





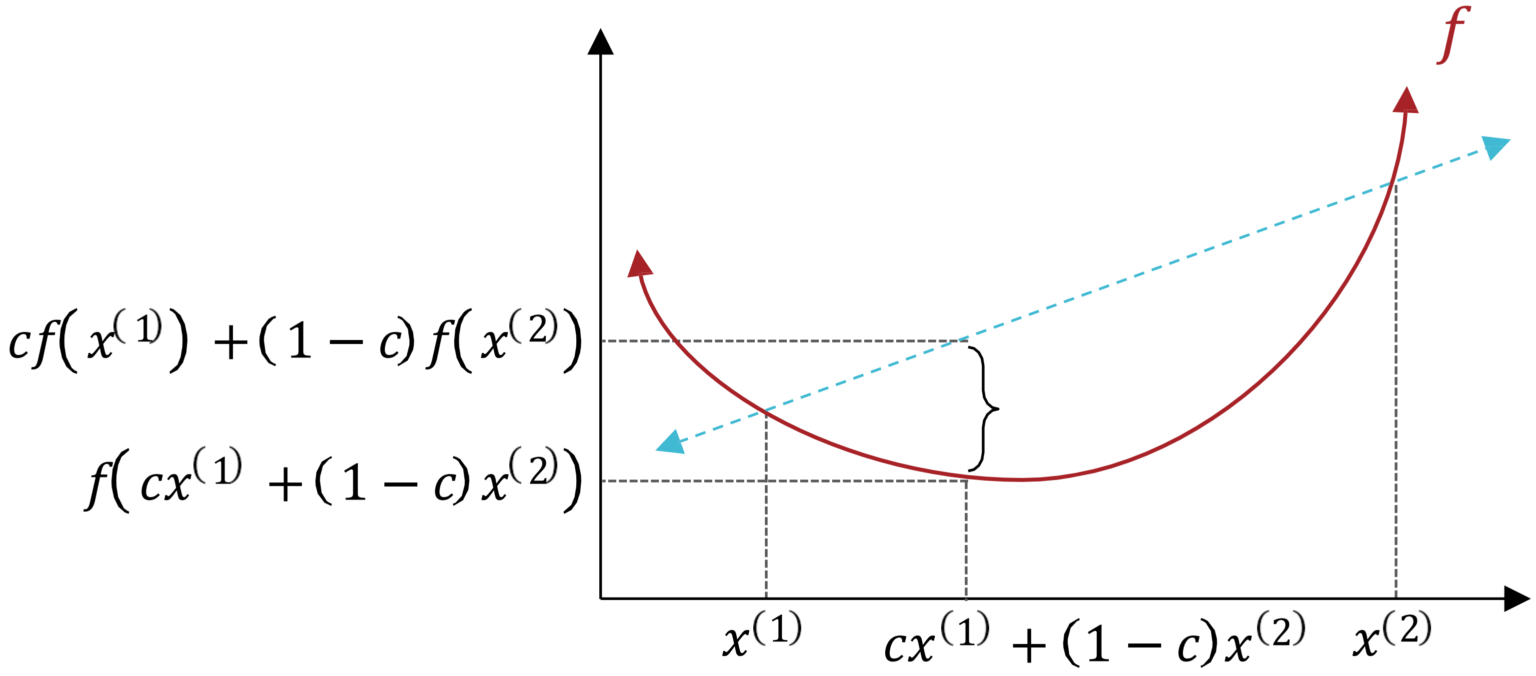


t	θ_1	θ_2	$J(\theta_1, \theta_2)$
1	0.01	0.02	25.2
2	0.30	0.12	8.7
3	0.51	0.30	1.5
4	0.59	0.43	0.2

Convexity

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

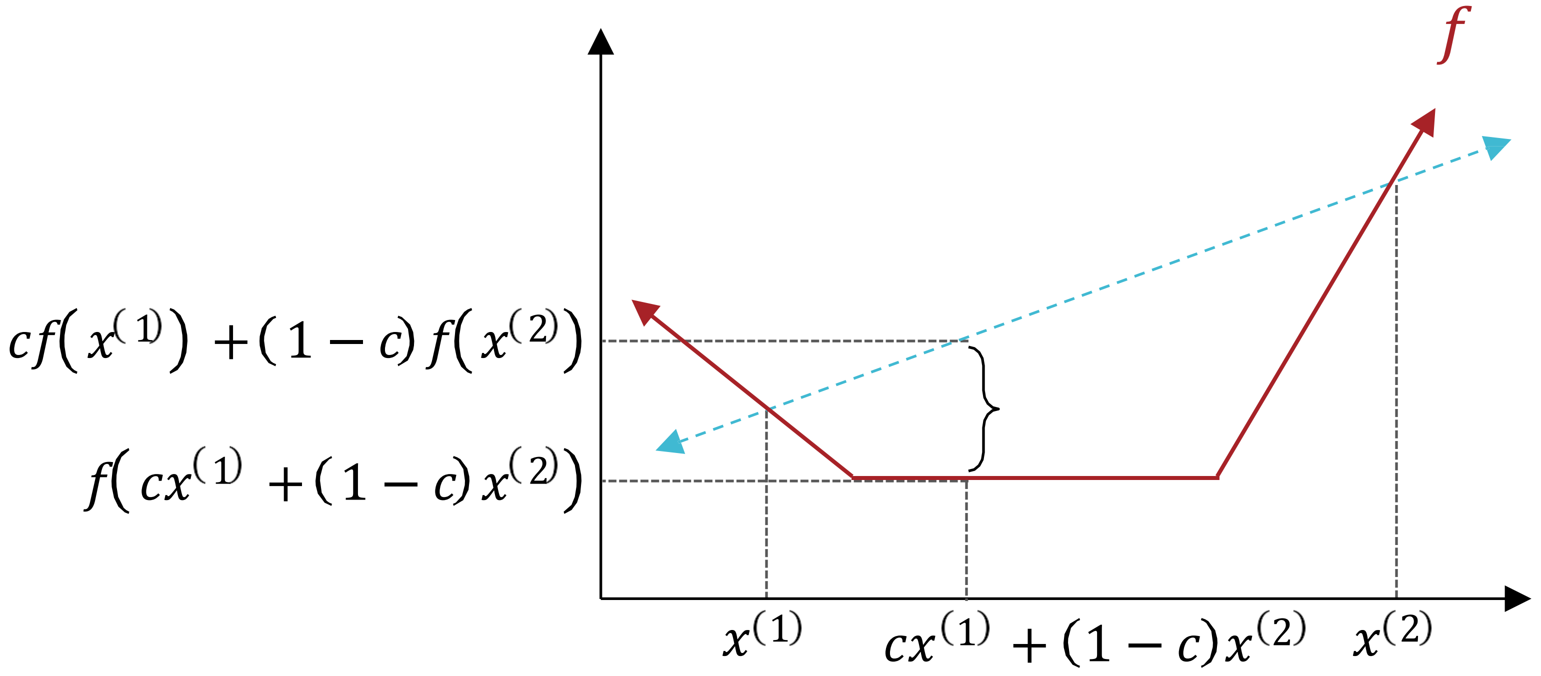
$$f\left(c\boldsymbol{x}^{(1)} + (1-c)\boldsymbol{x}^{(2)}\right) \le cf\left(\boldsymbol{x}^{(1)}\right) + (1-c)f\left(\boldsymbol{x}^{(2)}\right)$$



Convexity

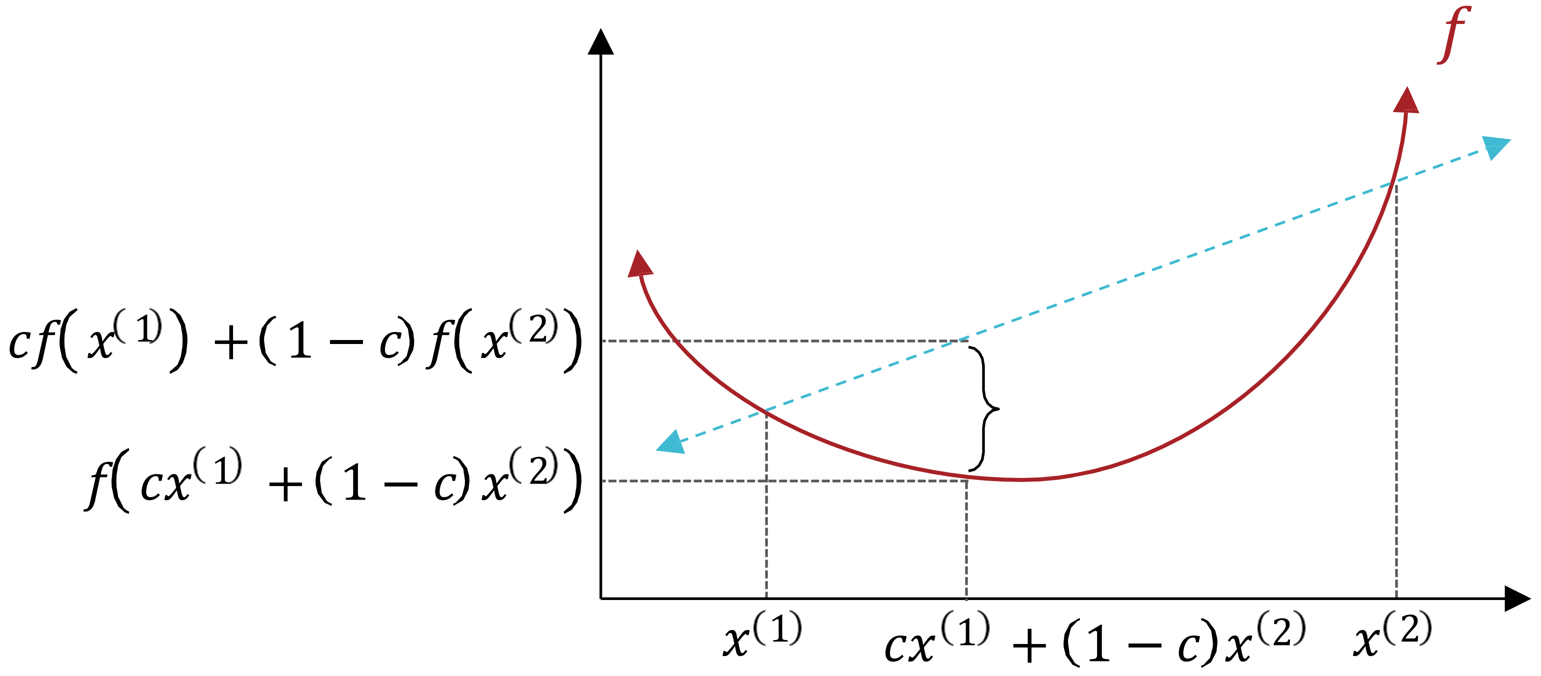
- A function $f: \mathbb{R}^D \to \mathbb{R}$ is convex if $\forall\, \boldsymbol{x}^{(1)} \in \mathbb{R}^D, \boldsymbol{x}^{(2)} \in \mathbb{R}^D$ and $0 \le c \le 1$

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



Convexity

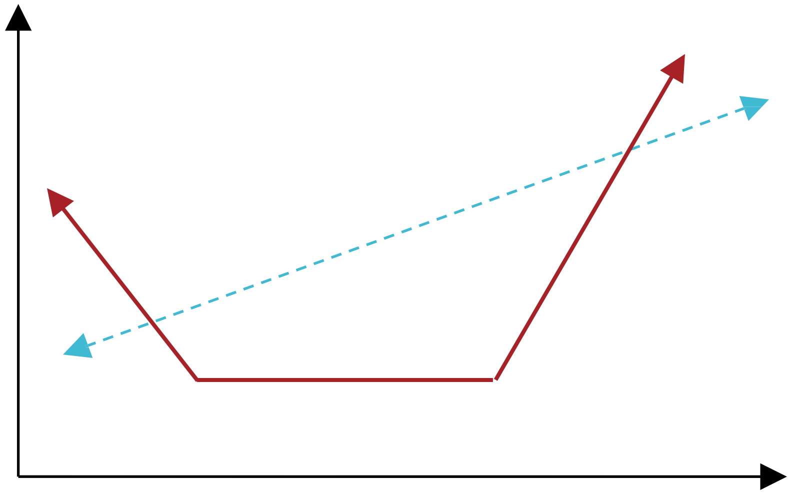
- A function $f: \mathbb{R}^D \rightarrow \mathbb{R}$ is *strictly* convex if
  $\forall\ \boldsymbol{x}^{(1)} \in \mathbb{R}^D, \boldsymbol{x}^{(2)} \in \mathbb{R}^D$ and $0 < c < 1$
  $f\left(c\boldsymbol{x}^{(1)} + (1-c)\boldsymbol{x}^{(2)}\right) < cf\left(\boldsymbol{x}^{(1)}\right) + (1-c)f\left(\boldsymbol{x}^{(2)}\right)$



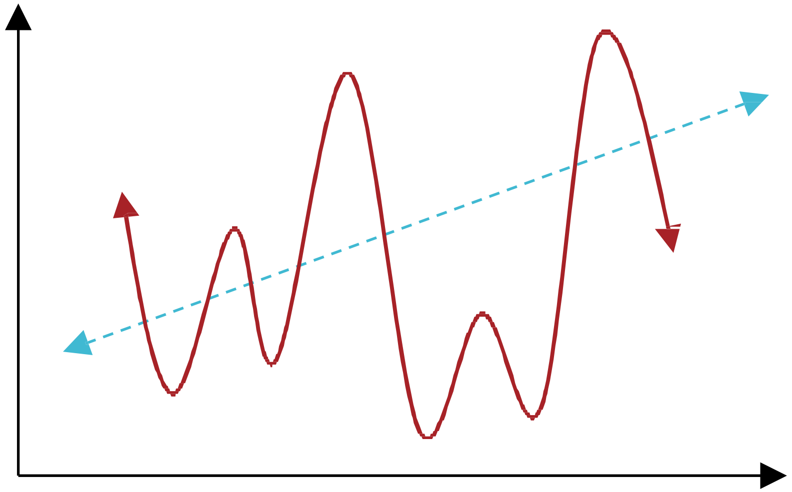
Poll Question 1

Convexity

(a)



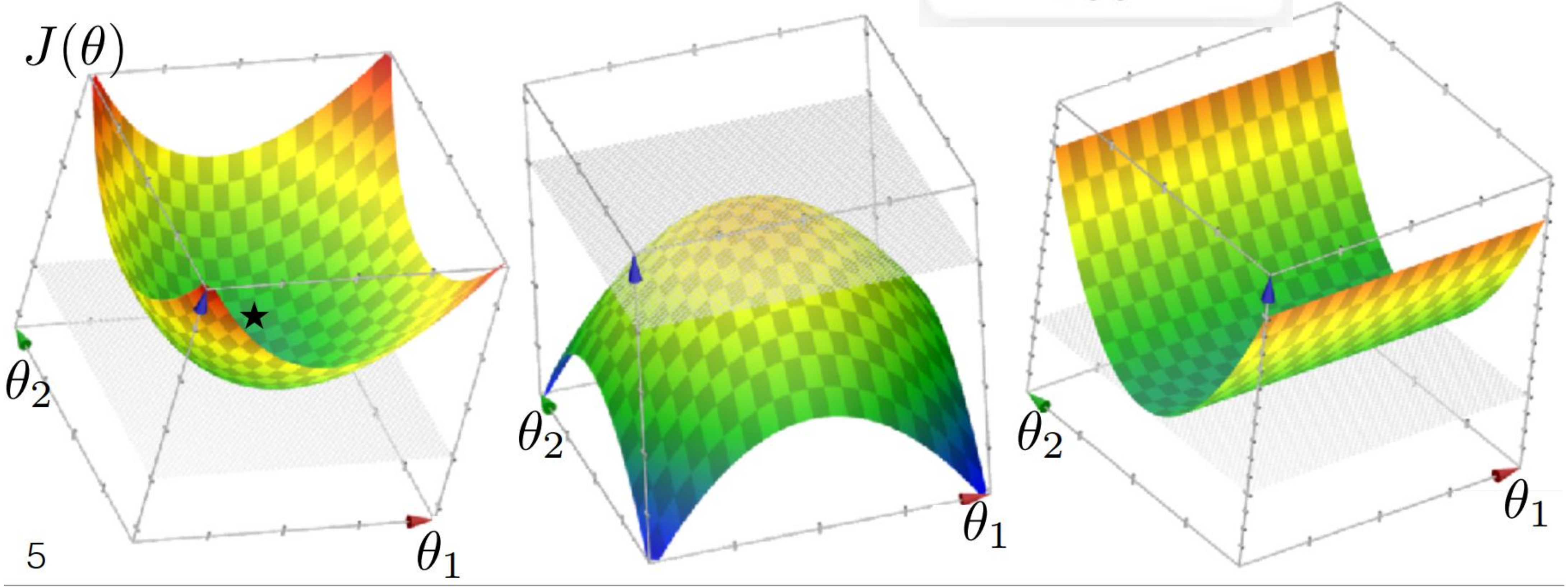
(b)



Convex or not?

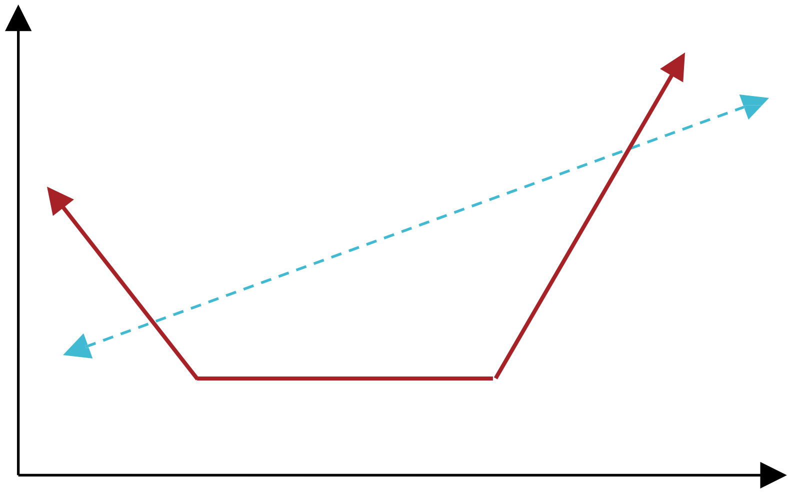
If convex, strictly convex or not?

Poll Question 2:
Convex or not?
If Convex, strictly convex or not?



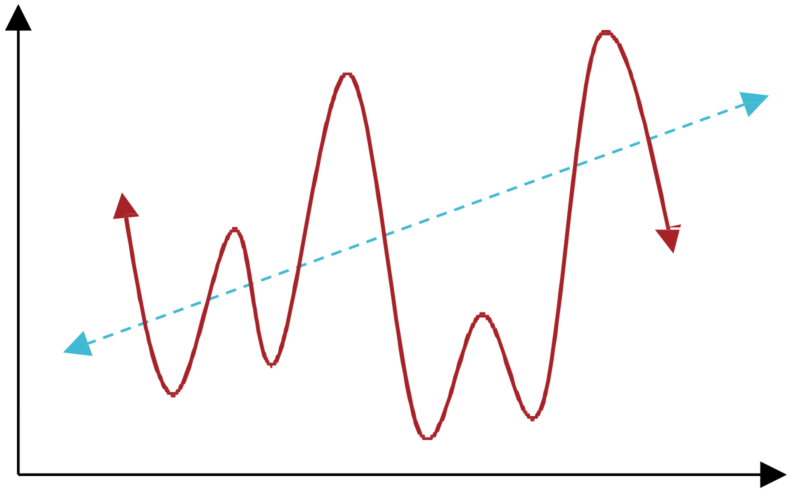
(a) (b) (c)

Convexity



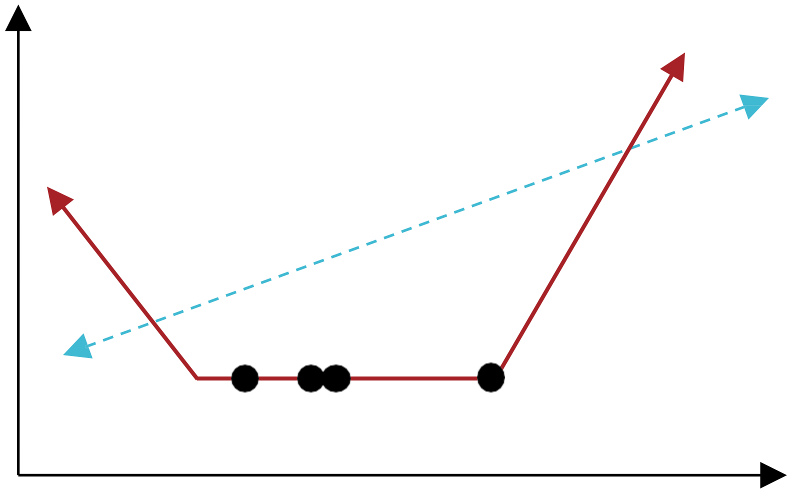
Given a function $f: \mathbb{R}^D \to \mathbb{R}$

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



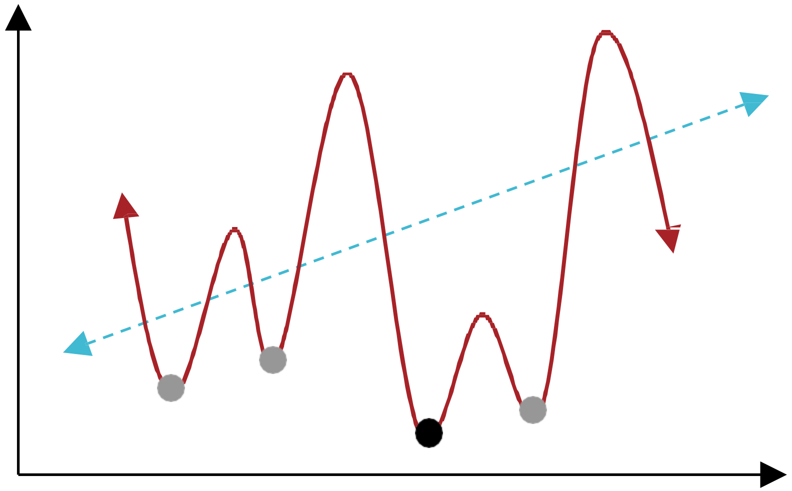
- $\boldsymbol{x}^*$ is a local minimum iff $\exists \; \epsilon$ s.t. $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x}) \; \forall \; \boldsymbol{x}$ s.t. $\|\boldsymbol{x} - \boldsymbol{x}^*\|_2 < \epsilon$

Convexity



Convex functions:

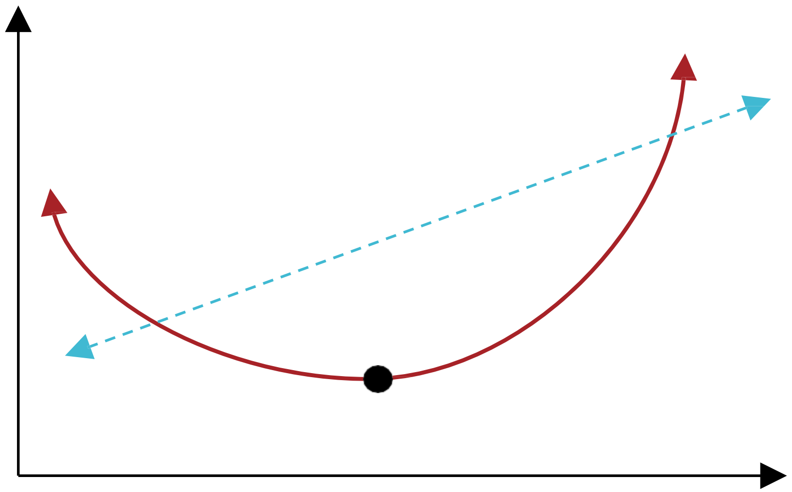
Each local minimum is a global minimum!



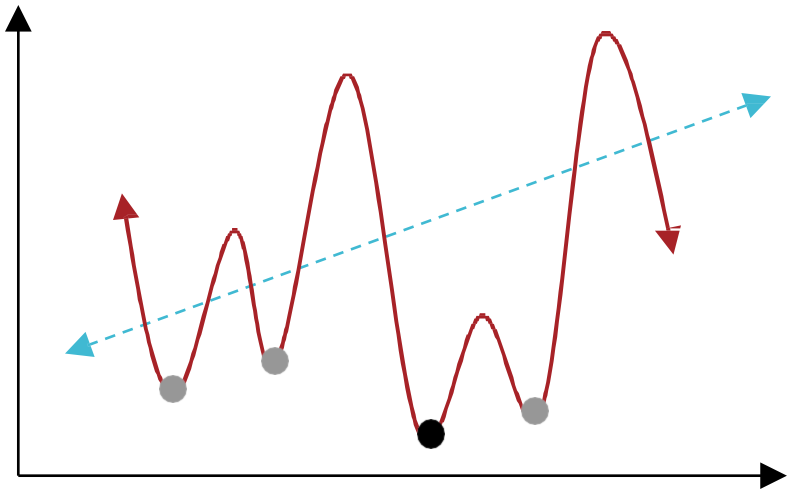
Non-convex functions:

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

Convexity



Strictly convex functions:
There exists a unique global minimum!

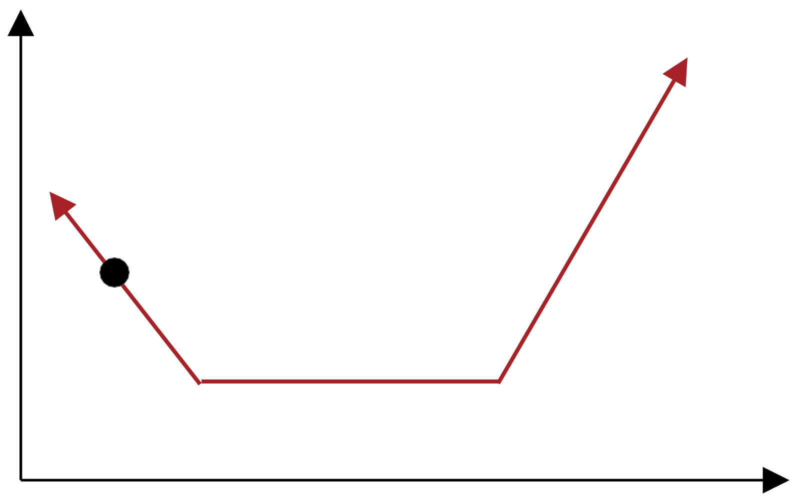


Non-convex functions:

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

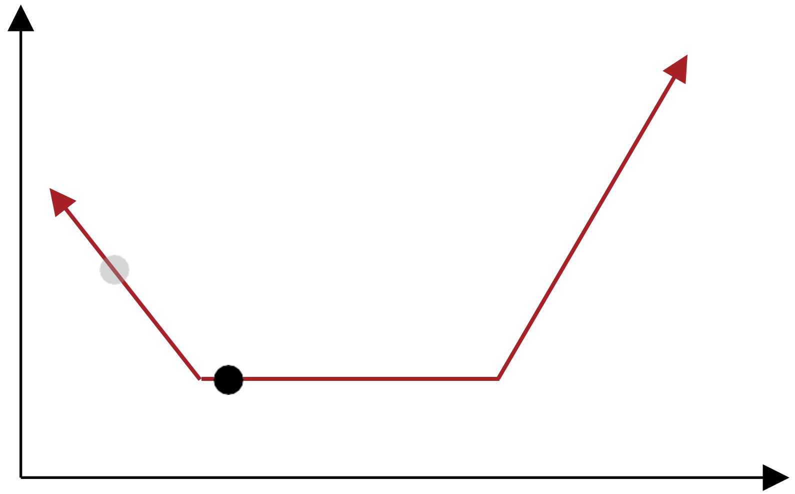
Gradient Descent & Convexity

- 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!



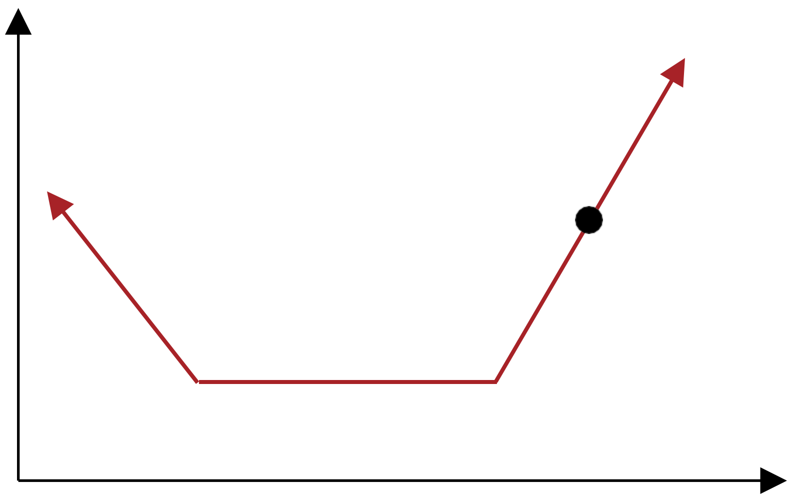
Gradient Descent & Convexity

- 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!



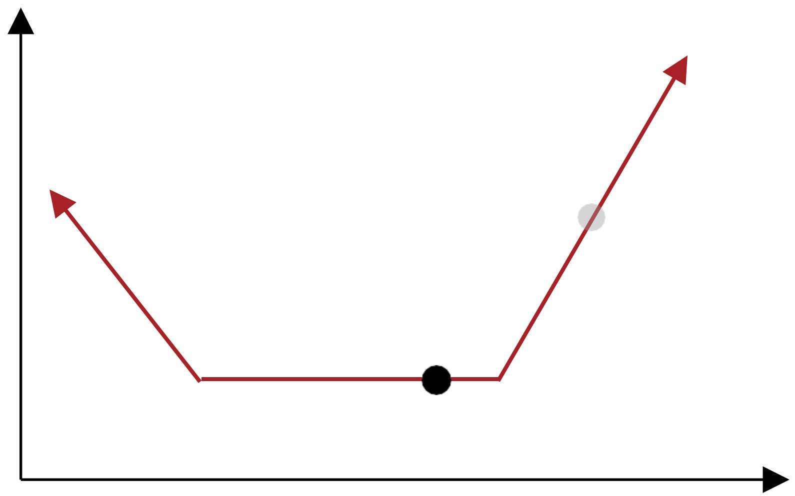
Gradient Descent & Convexity

- 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!



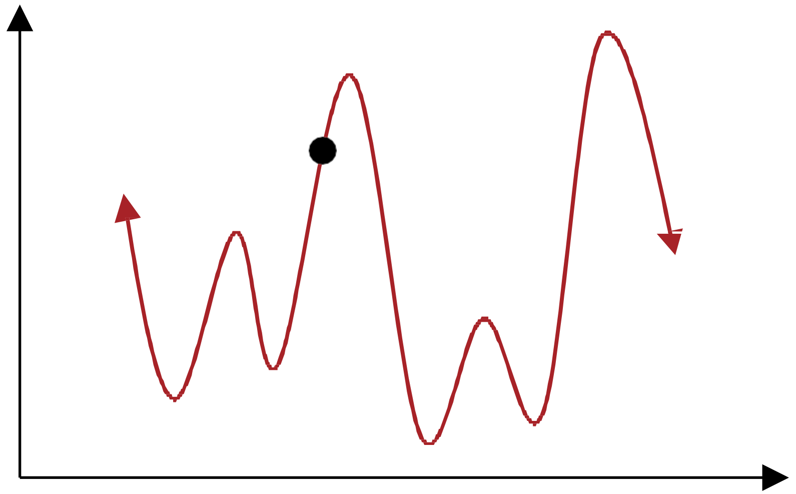
Gradient Descent & Convexity

- 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!



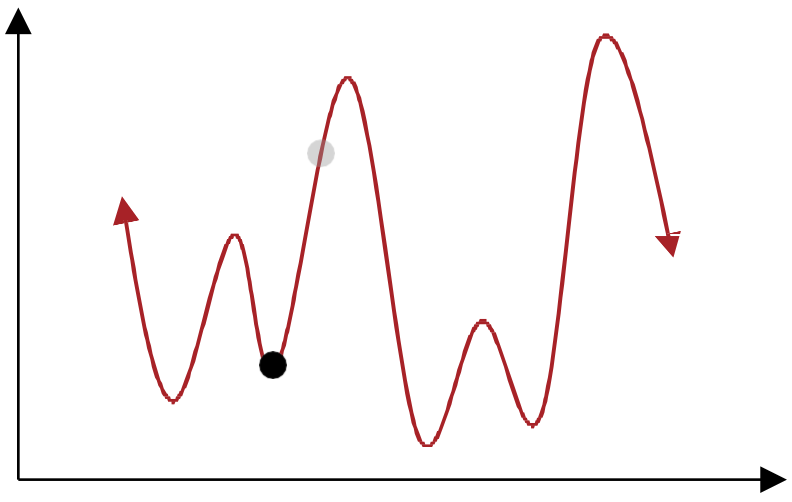
Gradient Descent & Convexity

- 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…



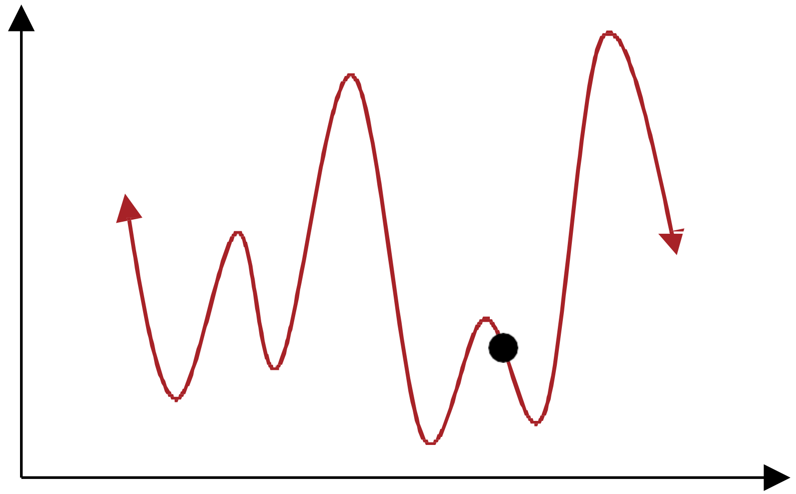
Gradient Descent & Convexity

- 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…



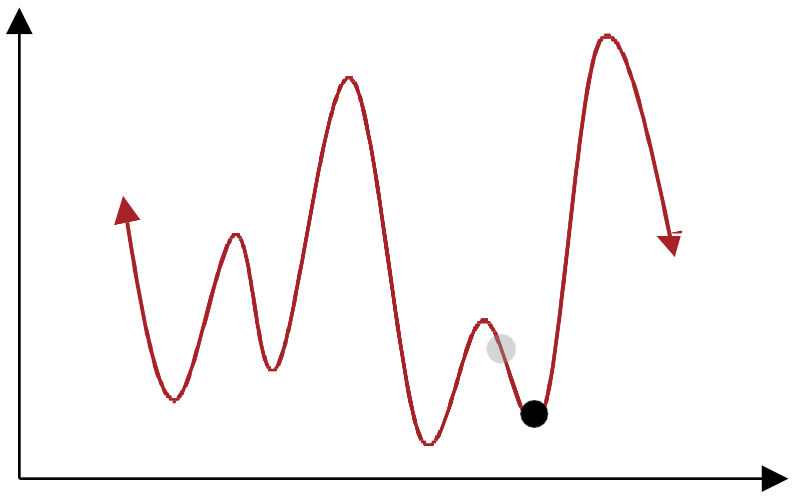
Gradient Descent & Convexity

- 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…



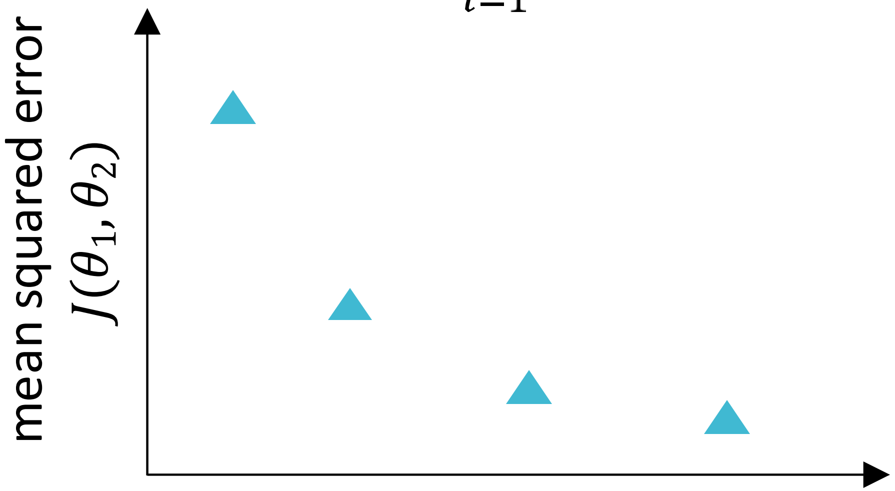
Gradient Descent & Convexity

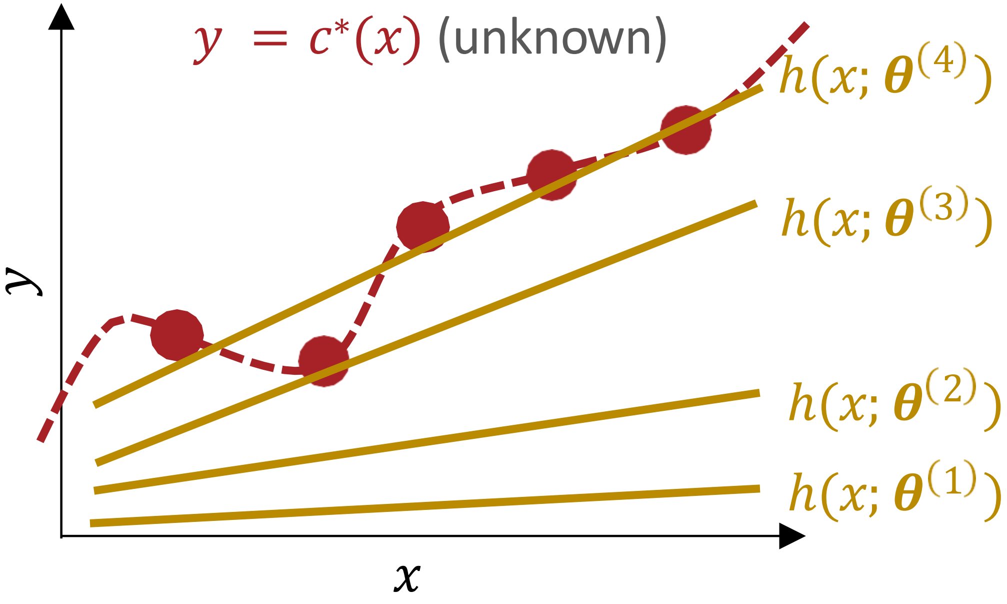
- Gradient descent is a local optimization algorithm – it will converge to a local minimum (if it converges)
  - Not ideal if the objective function is non-convex…

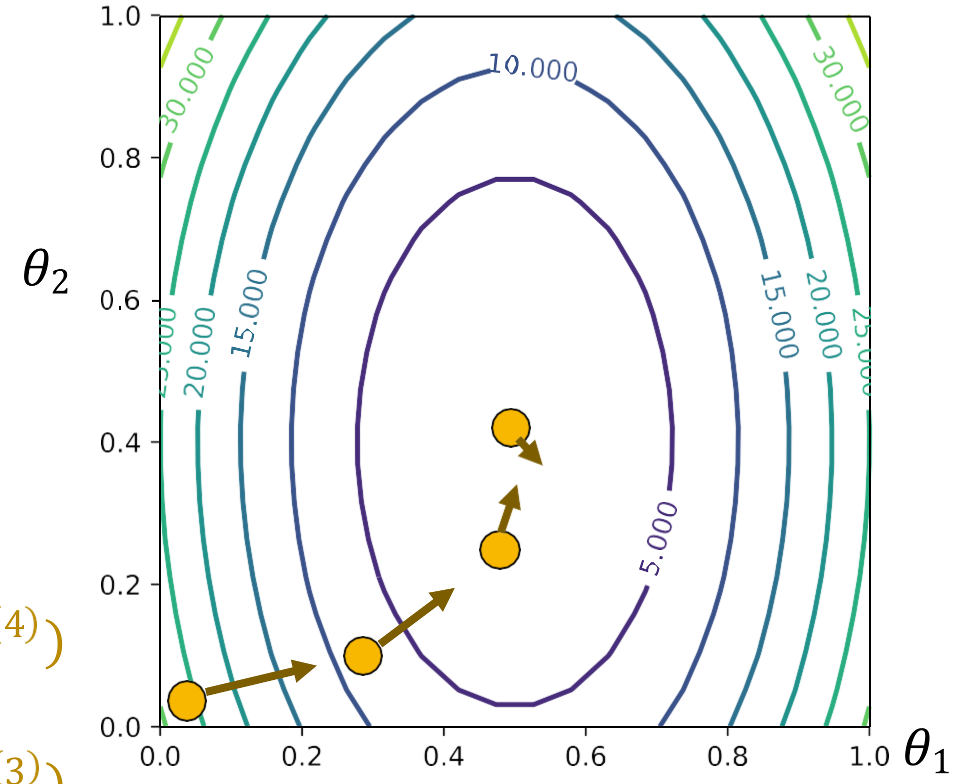


Why Gradient Descent for Linear Regression ?

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



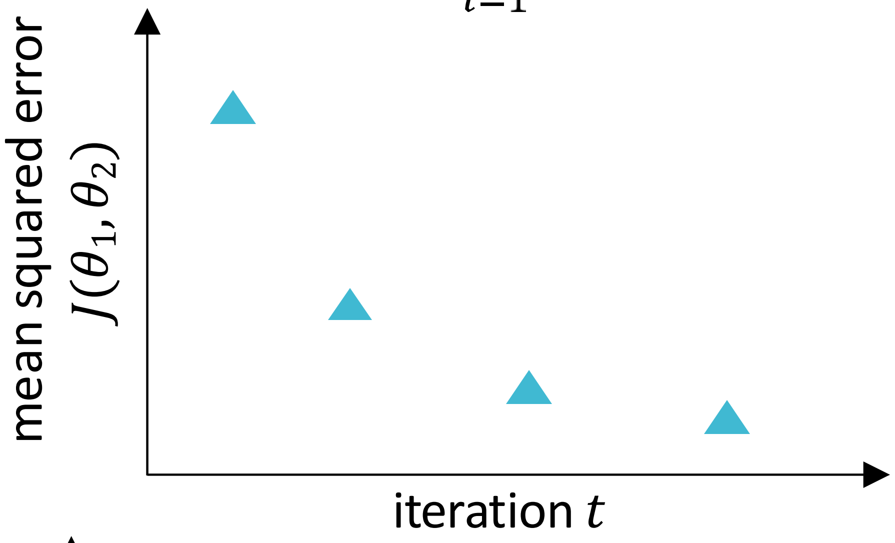


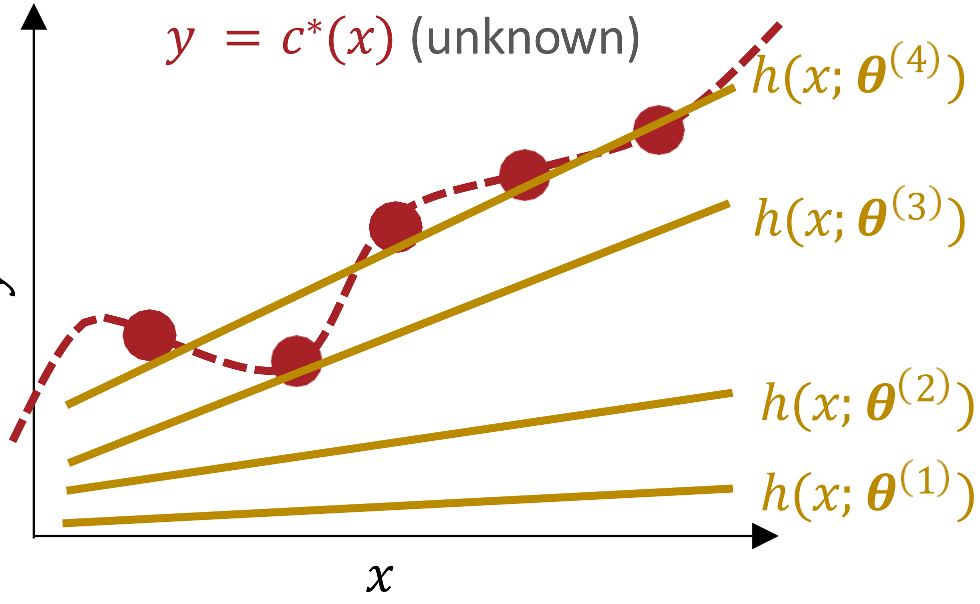


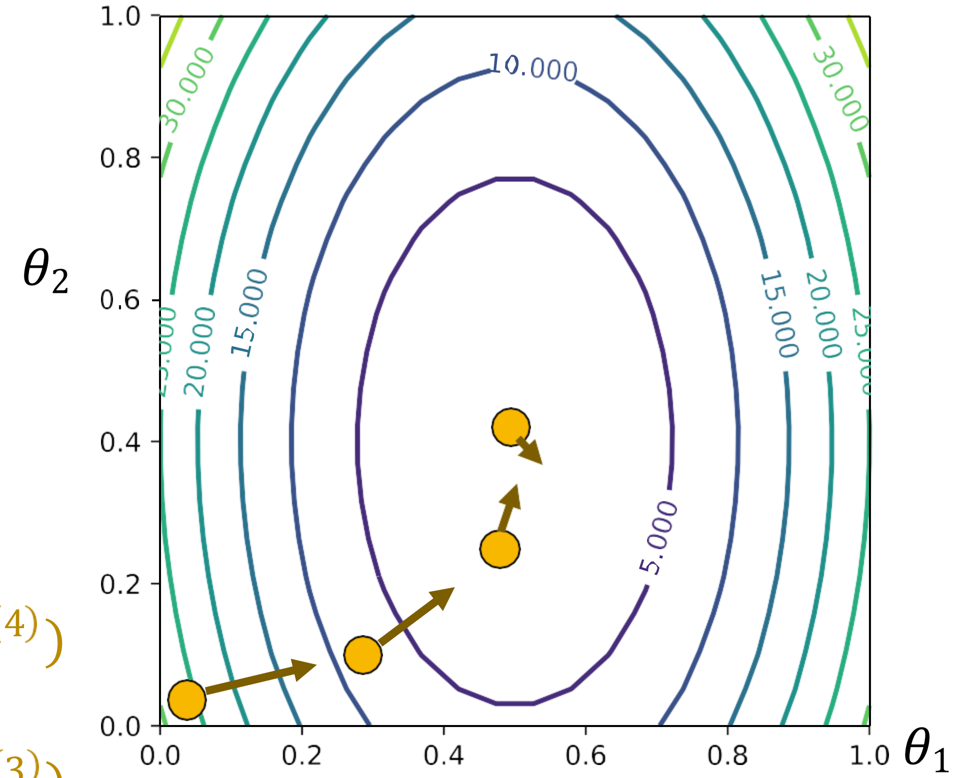
t	θ_1	θ_2	$J(\theta_1, \theta_2)$
1	0.01	0.02	25.2
2	0.30	0.12	8.7
3	0.51	0.30	1.5
4	0.59	0.43	0.2

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

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



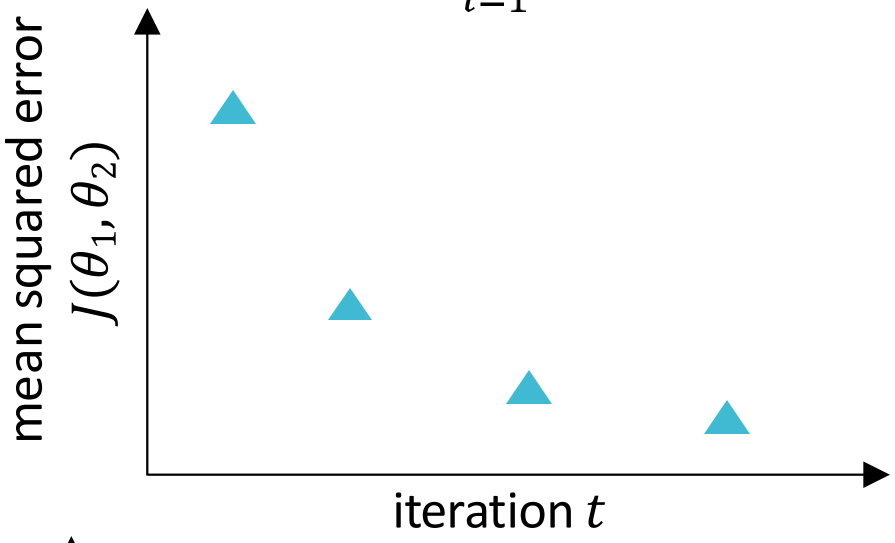


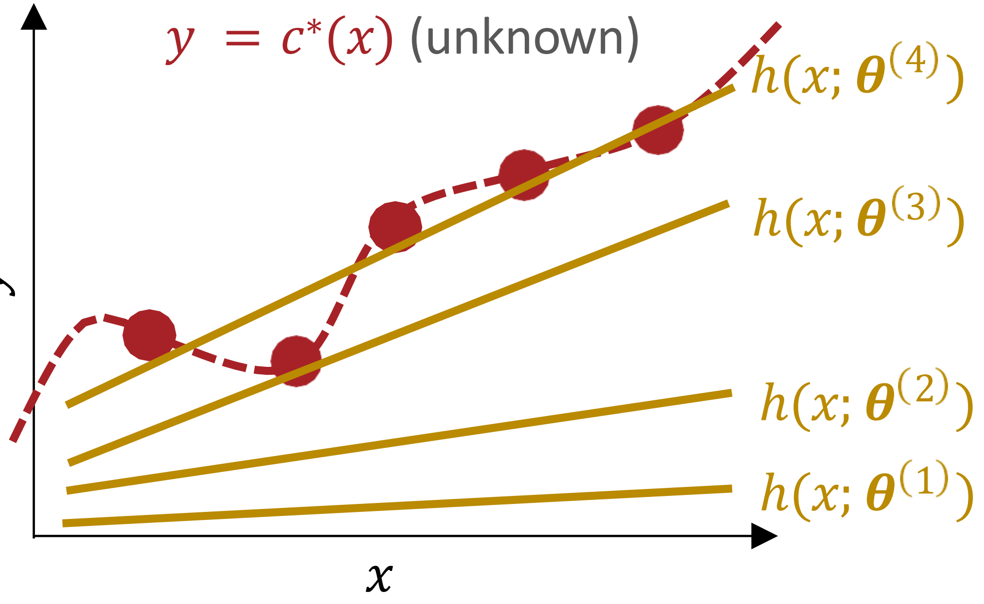


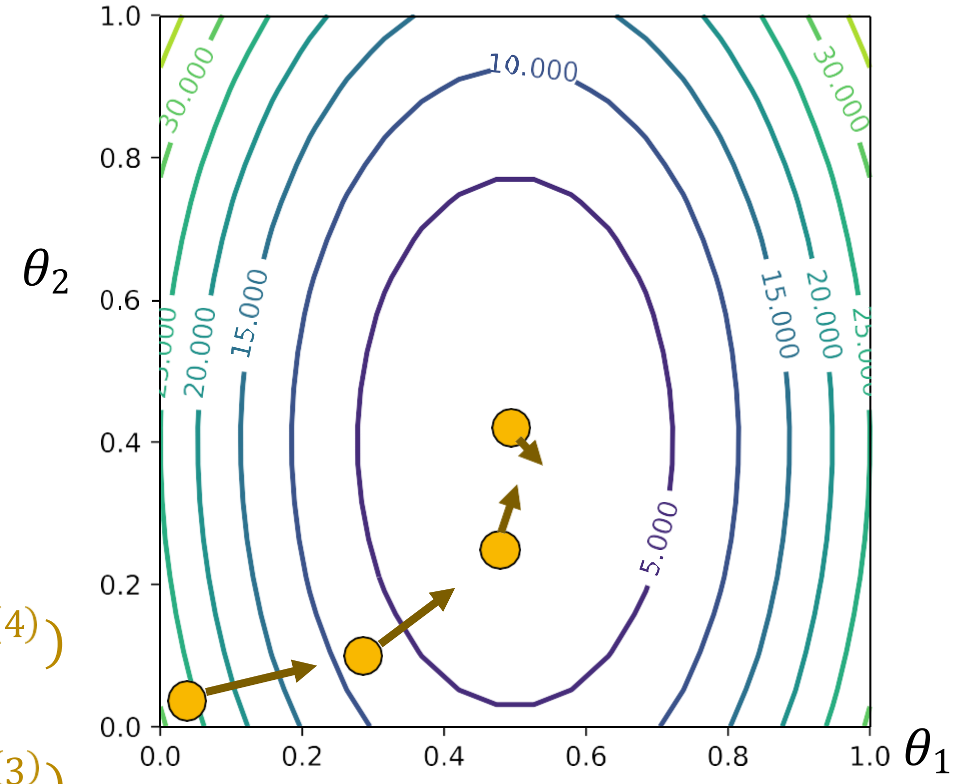
t	θ_1	θ_2	$J(\theta_1, \theta_2)$
1	0.01	0.02	25.2
2	0.30	0.12	8.7
3	0.51	0.30	1.5
4	0.59	0.43	0.2

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	θ_1	θ_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

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

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

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

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

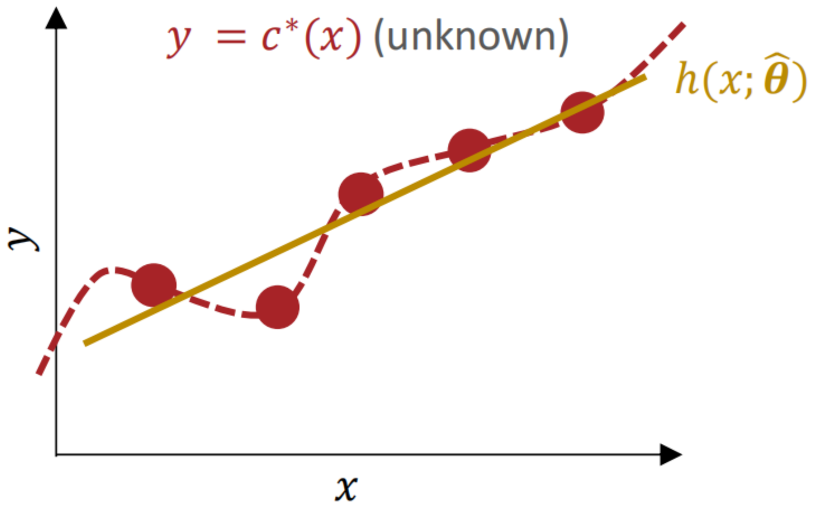
$$\nabla_{\boldsymbol{\theta}} J(\widehat{\boldsymbol{\theta}}) = \frac{1}{2N}\left(2X^T X\widehat{\boldsymbol{\theta}} - 2X^T\boldsymbol{y}\right) = 0$$

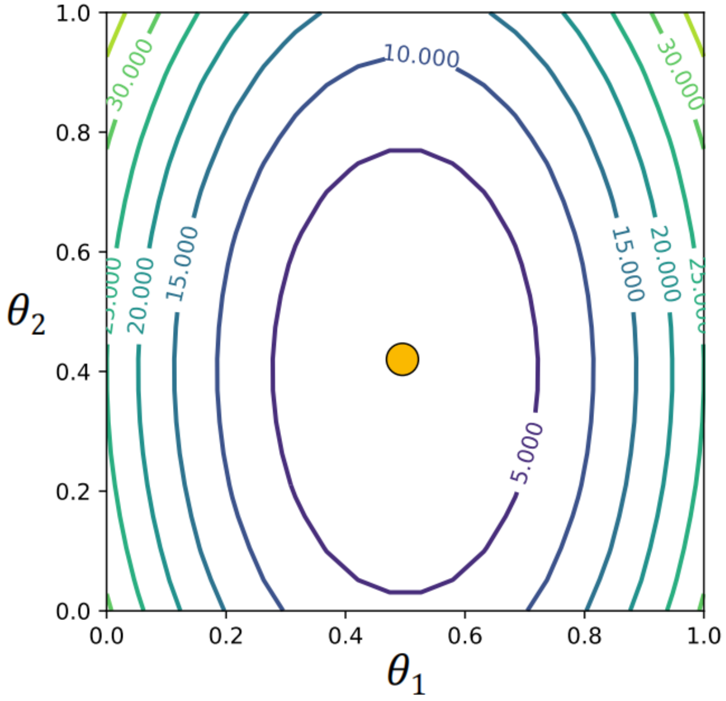
$$\rightarrow X^T X\widehat{\boldsymbol{\theta}} = X^T\boldsymbol{y}$$

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

Closed Form Optimization

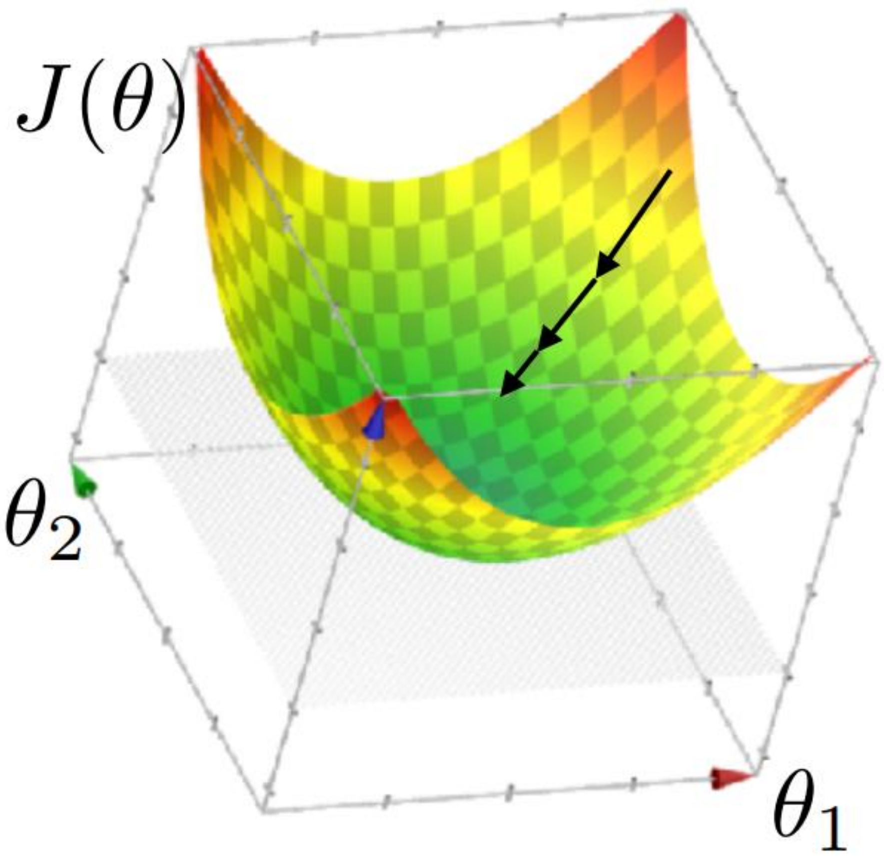
$$\widehat{\boldsymbol{\theta}} = (X^T X)^{-1} X^T \boldsymbol{y}$$

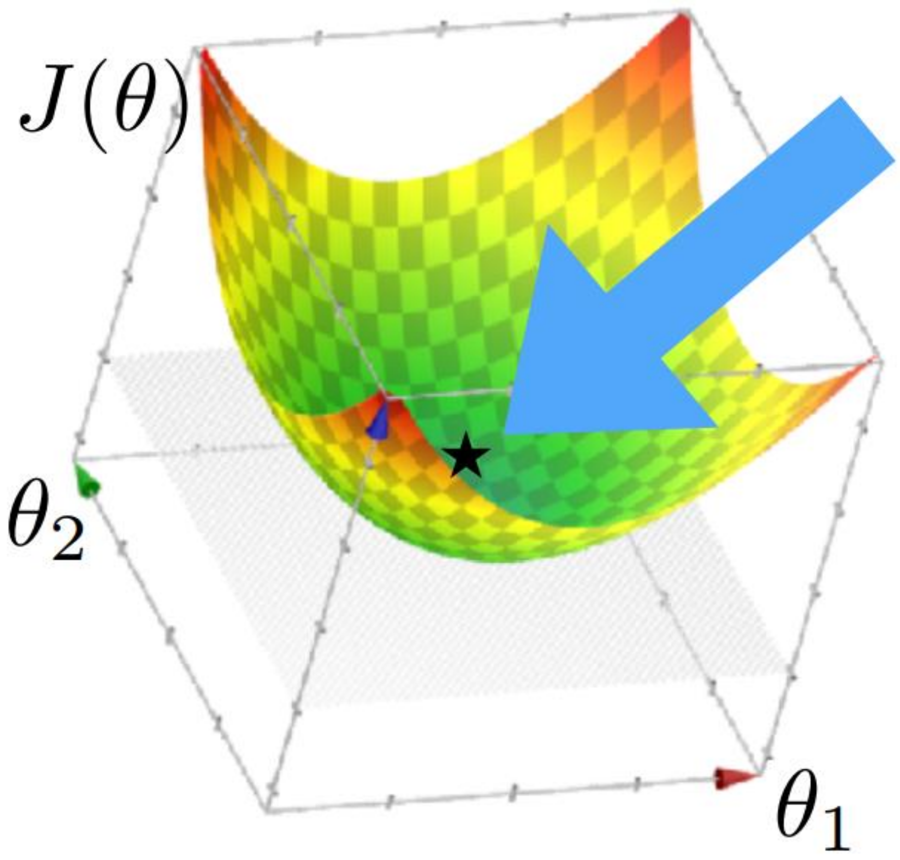




t	θ_1	θ_2	$J(\theta_1, \theta_2)$
1	0.59	0.43	0.2

Gradient Descent Vs. Analytical/Closed-form/Direct Solution





Closed Form Solution

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

1. Is $X^T X$ invertible?

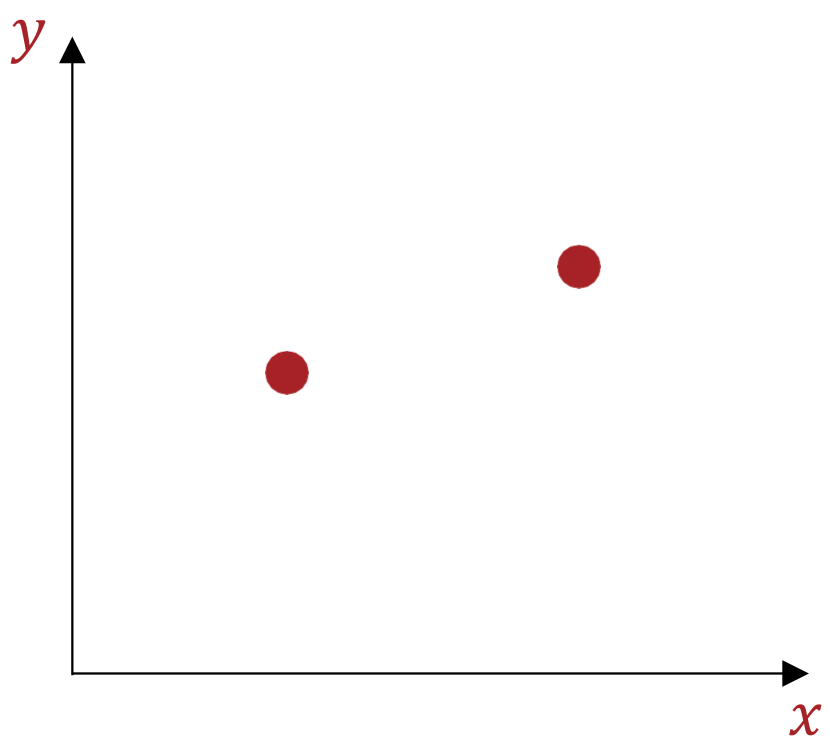
2. If so, how computationally expensive is inverting $X^T X$?

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

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 $X^T X$?

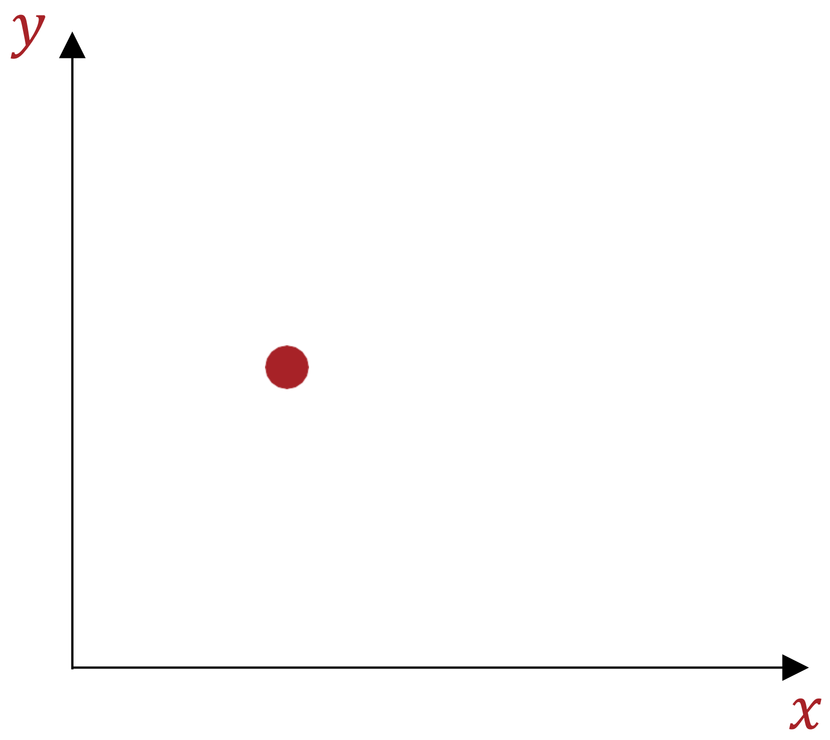
Linear Regression: Uniqueness

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



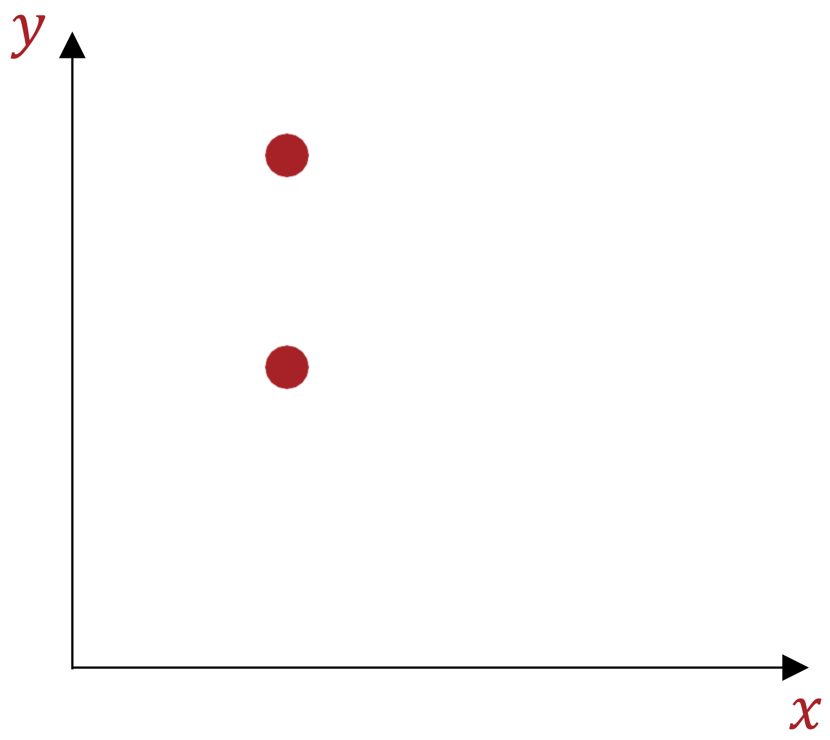
Linear Regression: Uniqueness

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



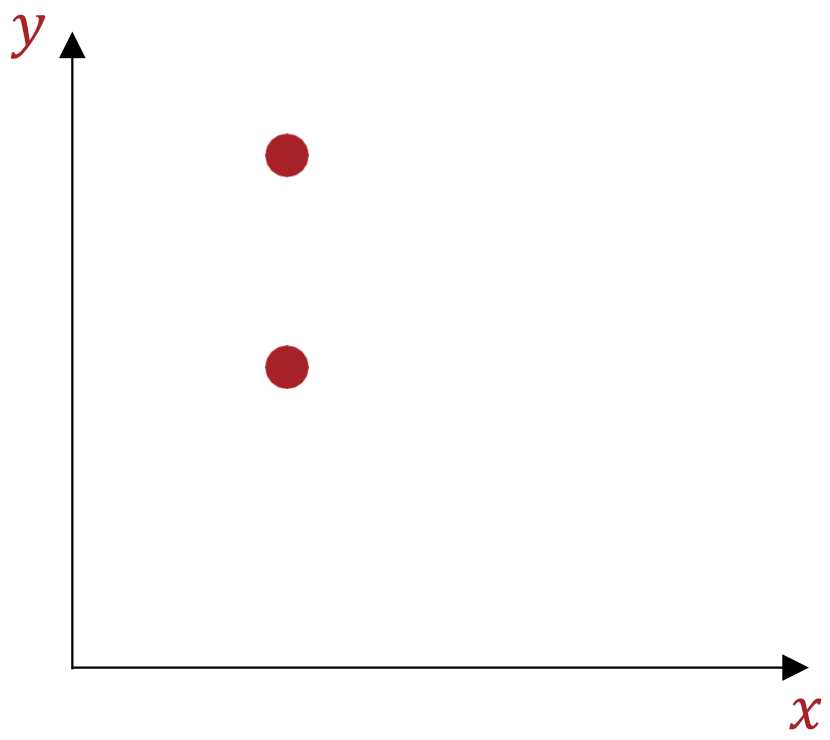
Linear Regression: Uniqueness

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



Poll Question 3

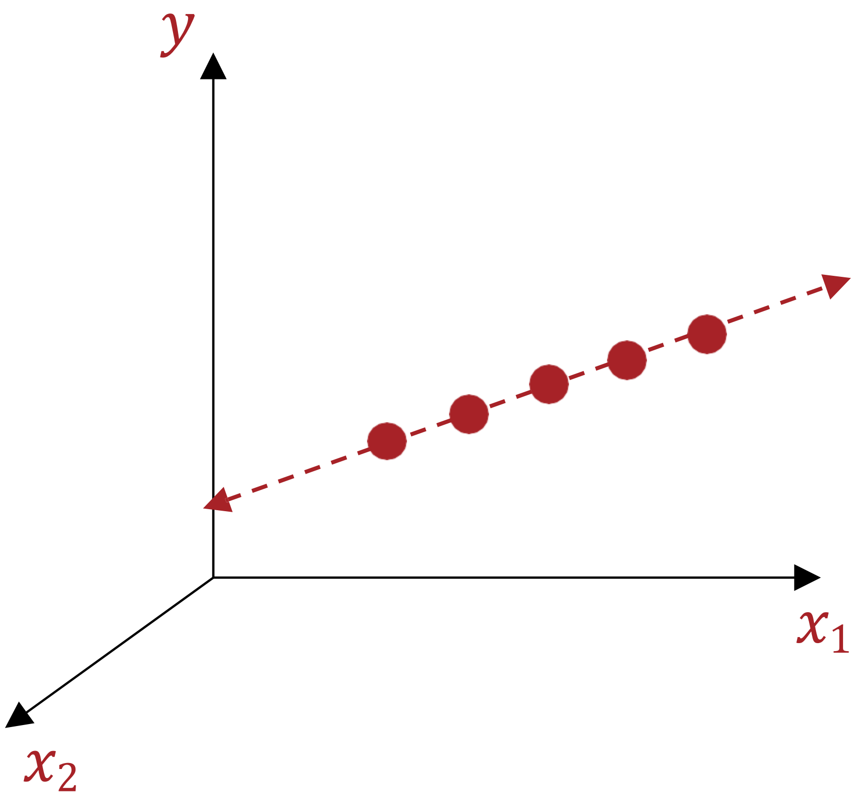
- Consider a 1D linear regression model trained to minimize the mean squared error: how many optimal solutions (i.e., sets of parameters θ) are there for the given dataset?



A. 0　　B. 1　　C. 2　　D. 3　　E. ∞

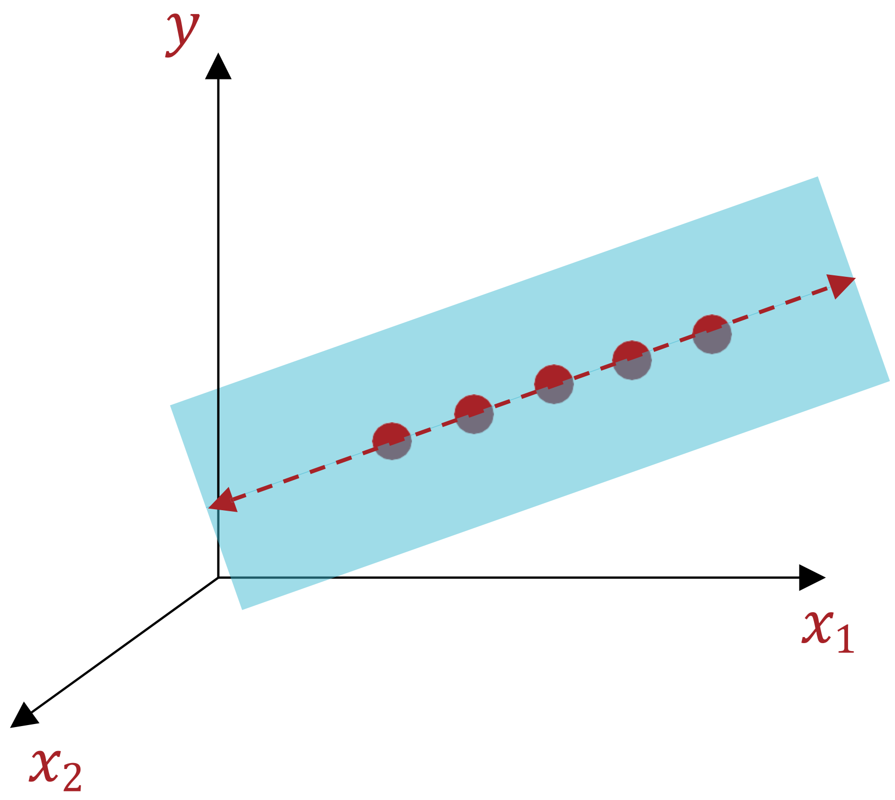
Linear Regression: Uniqueness

- 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?



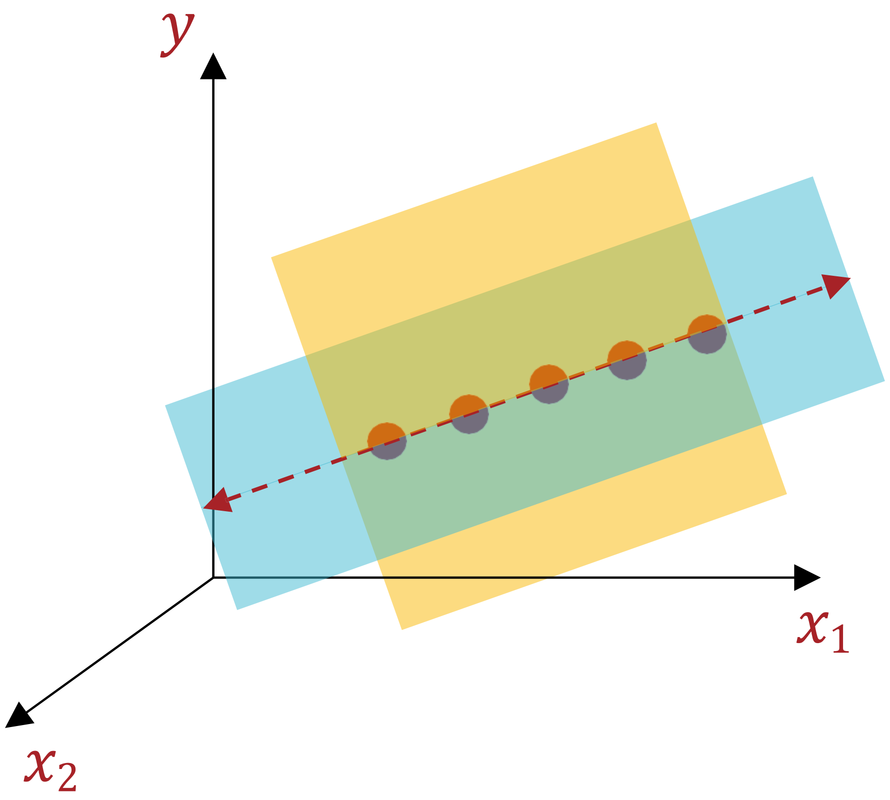
Linear Regression: Uniqueness

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



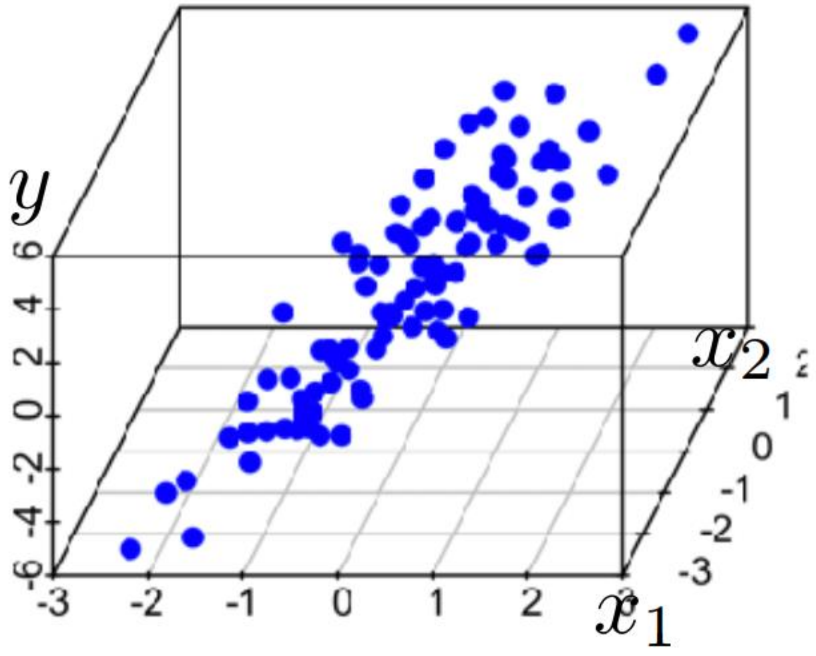
Linear Regression: Uniqueness

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



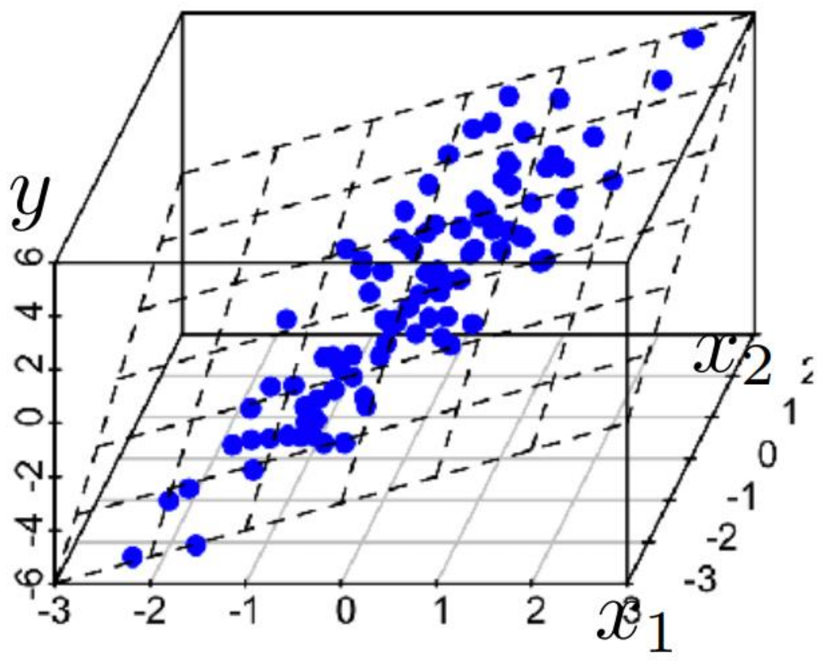
Linear Regression: Uniqueness

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



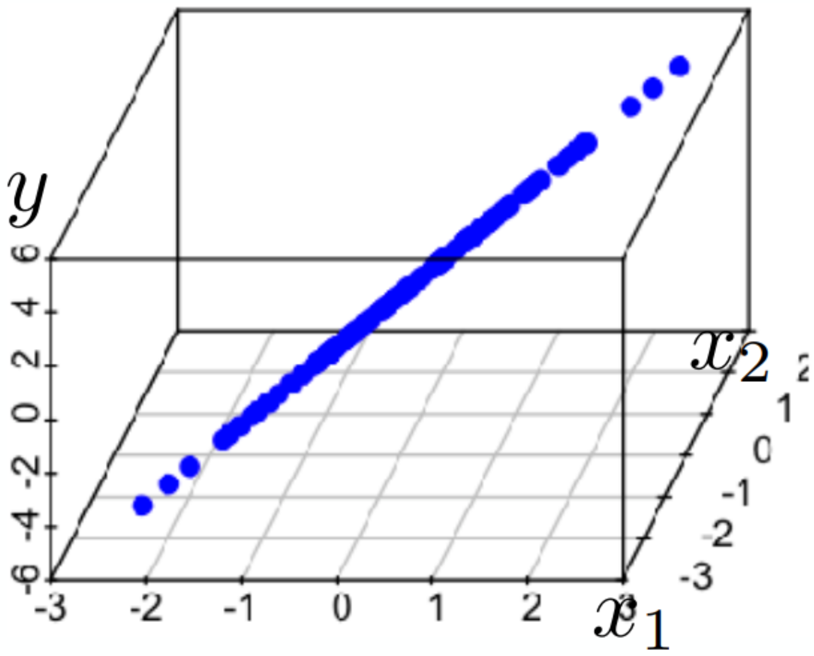
Linear Regression: Uniqueness

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

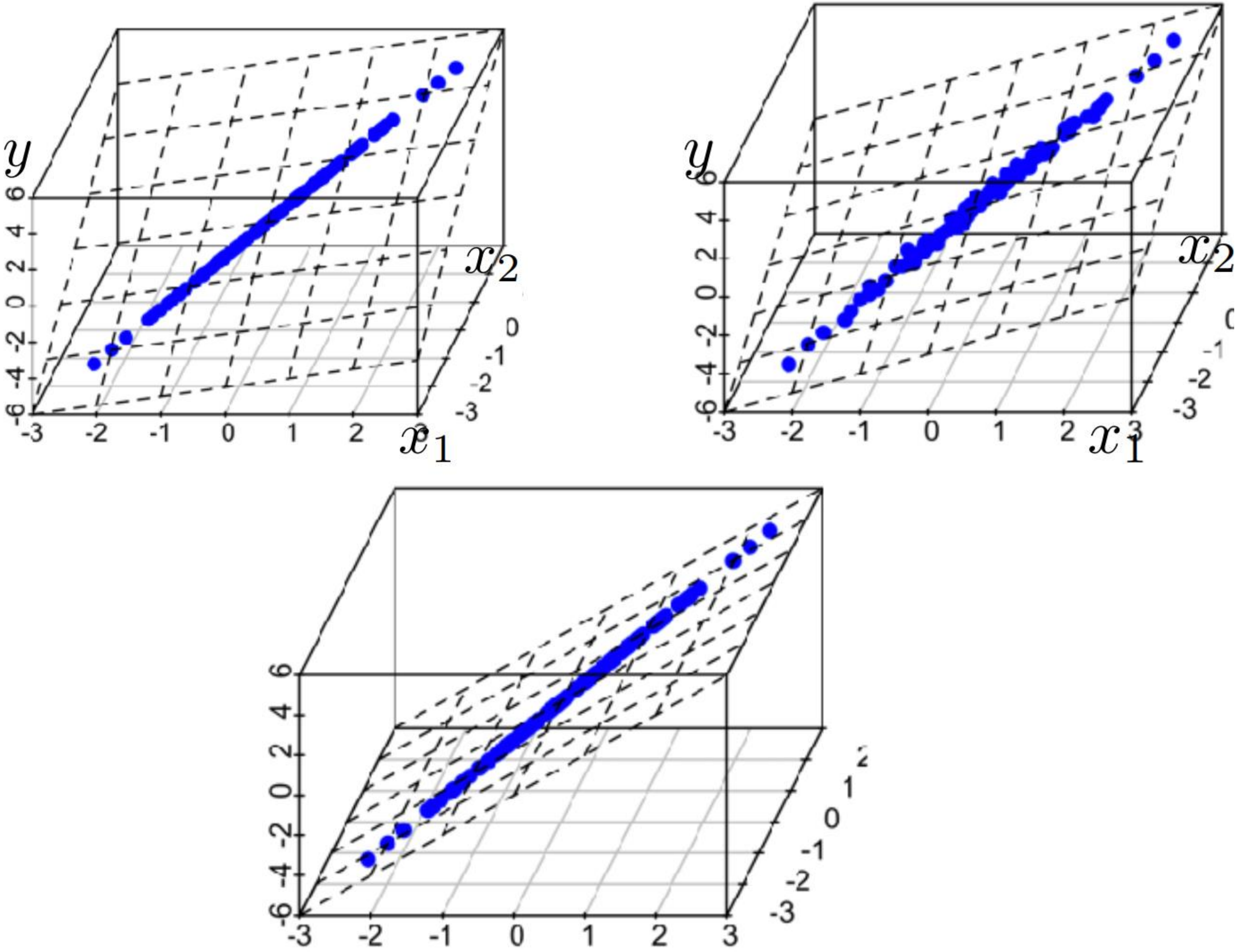


Linear Regression: Uniqueness

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



Linear Regression: Uniqueness



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...

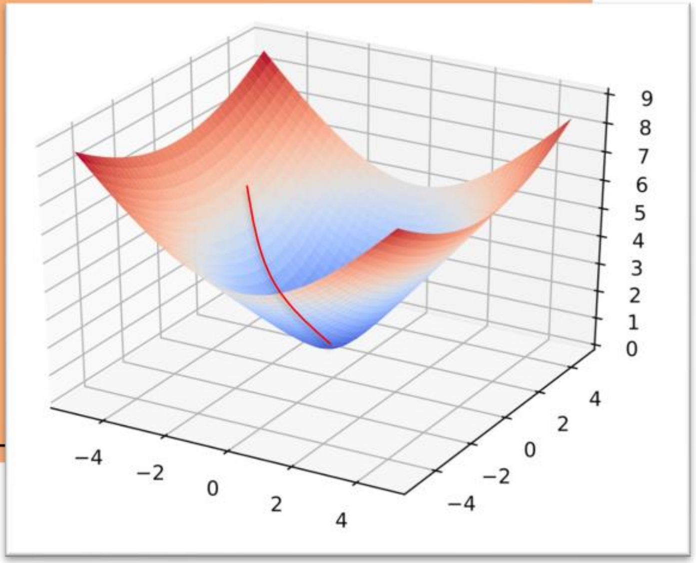
- Design k-NN Regression and Decision Tree Regression
- Implement learning for Linear Regression using gradient descent or closed form optimization
- Choose a Linear Regression optimization technique that is appropriate for a particular dataset by analyzing the tradeoff of computational complexity vs.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

Stochastic Gradient Descent

Gradient Descent

Algorithm 1 Gradient Descent

1: **procedure** GD($\mathcal{D}, \boldsymbol{\theta}^{(0)}$)
2: $\quad\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(0)}$
3: $\quad$**while** not converged **do**
4: $\quad\quad\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \gamma \nabla J(\boldsymbol{\theta})$
5: $\quad$**return** $\boldsymbol{\theta}$



per-example objective:

$$J^{(i)}(\boldsymbol{\theta})$$

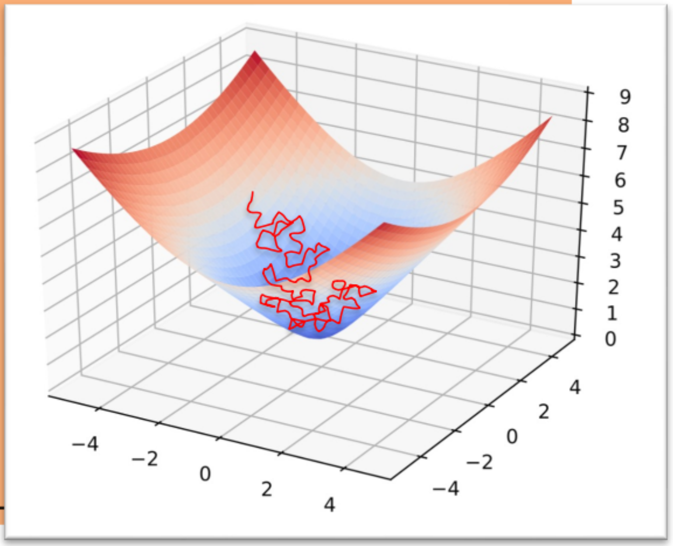
original objective:

$$J(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} J^{(i)}(\boldsymbol{\theta})$$

Stochastic Gradient Descent

Algorithm 2 Stochastic Gradient Descent (SGD)

procedure SGD($\mathcal{D}, \boldsymbol{\theta}^{(0)}$)
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(0)}$
  while not converged **do**
    $i \sim \text{Uniform}(\{1, 2, \ldots, N\})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \gamma \nabla_{\boldsymbol{\theta}} J^{(i)}(\boldsymbol{\theta})$
  return $\boldsymbol{\theta}$



per-example objective:

$$J^{(i)}(\boldsymbol{\theta})$$

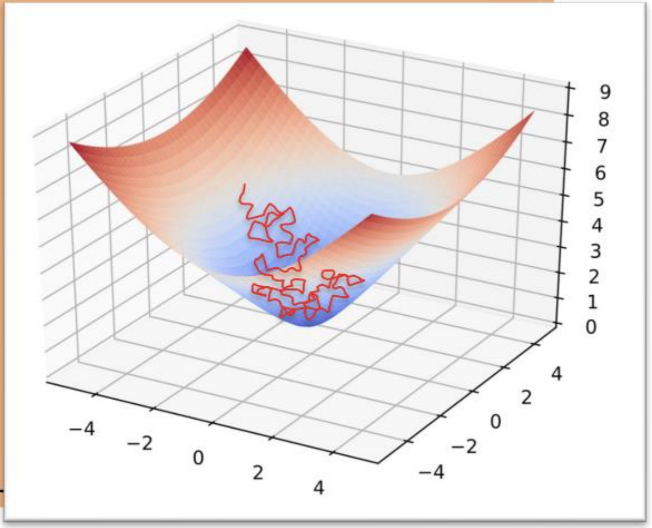
original objective:

$$J(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} J^{(i)}(\boldsymbol{\theta})$$

Stochastic Gradient Descent

Algorithm 2 Stochastic Gradient Descent (SGD)

procedure SGD($\mathcal{D}, \boldsymbol{\theta}^{(0)}$)
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(0)}$
  while not converged **do**
    for $i \in \text{shuffle}(\{1, 2, \ldots, N\})$ **do**
      $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \gamma \nabla_{\boldsymbol{\theta}} J^{(i)}(\boldsymbol{\theta})$
  return $\boldsymbol{\theta}$



per-example objective:

$$J^{(i)}(\boldsymbol{\theta})$$

original objective:

$$J(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} J^{(i)}(\boldsymbol{\theta})$$

In practice, it is common to implement SGD using sampling **without** replacement (i.e. shuffle({1,2,… N}), even though most of the theory is for sampling **with** replacement (i.e. Uniform({1,2,… N}).

Why does SGD work?

Background: *Expectation of a function of a random variable*

For any discrete random variable X

$$E_X[f(X)] = \sum_{x \in \mathcal{X}} P(X = x) f(x)$$

Objective Function for SGD

We assume the form to be:

$$J(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} J^{(i)}(\boldsymbol{\theta})$$

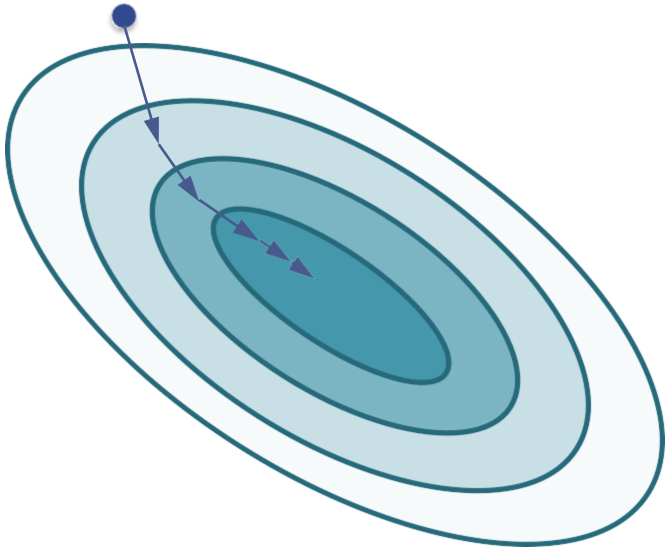
Expectation of a Stochastic Gradient:

- If the example is sampled uniformly at random, the expected value of the pointwise gradient is the same as the full gradient!

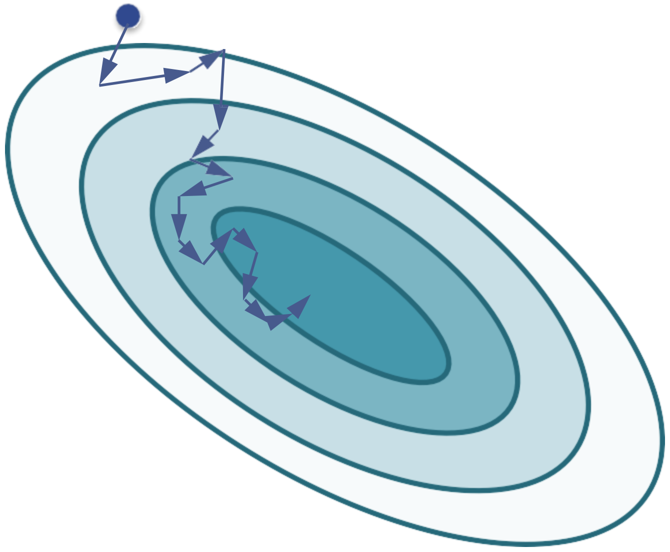
$$\begin{aligned} E[\nabla_{\boldsymbol{\theta}} J^{(i)}(\boldsymbol{\theta})] &= \sum_{i=1}^{N} (\text{probability of selecting } \boldsymbol{x}^{(i)}, y^{(i)}) \nabla_{\boldsymbol{\theta}} J^{(i)}(\boldsymbol{\theta}) \\ &= \sum_{i=1}^{N} \left(\frac{1}{N}\right) \nabla_{\boldsymbol{\theta}} J^{(i)}(\boldsymbol{\theta}) \\ &= \frac{1}{N} \sum_{i=1}^{N} \nabla_{\boldsymbol{\theta}} J^{(i)}(\boldsymbol{\theta}) \\ &= \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \end{aligned}$$

- In practice, the data set is randomly shuffled then looped through so that each data point is used equally often

SGD Vs. Gradient Descent



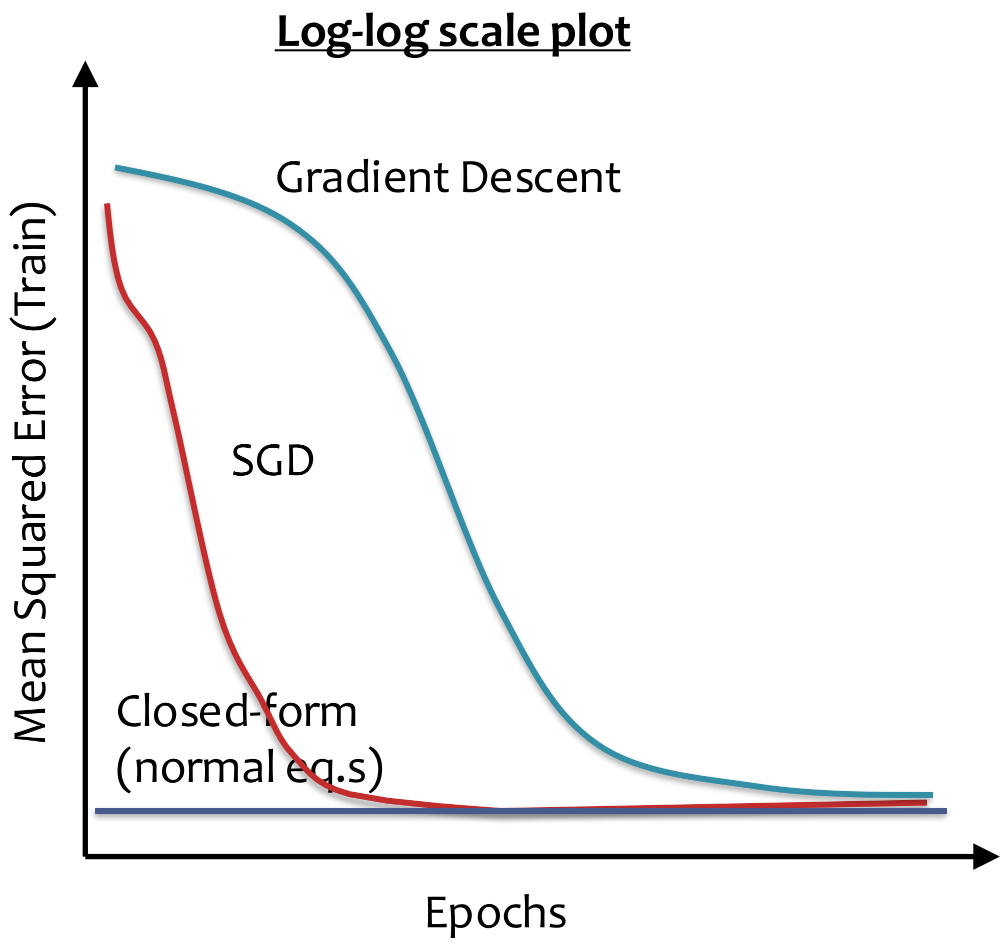
Gradient Descent



Stochastic Gradient Descent

SGD Vs. Gradient Descent

- Empirical comparison:



- *Def*: an **epoch** is a single pass through the training data

1. For GD, only **one update** per epoch
2. For SGD, ***N* updates** per epoch
   N = (# train examples)

- SGD reduces MSE much more rapidly than GD
- For GD / SGD, training MSE is initially large due to uninformed initialization

SGD vs. Gradient Descent

- Theoretical comparison:

Define convergence to be when $J\left(\boldsymbol{\theta}^{(t)}\right) - J(\boldsymbol{\theta}^*) < \epsilon$

Method	Steps to Convergence	Computation per Step
Gradient descent	$O\left(\log {}^{1}/_{\epsilon}\right)$	$O(NM)$
SGD	$O\left({}^{1}/_{\epsilon}\right)$	$O(M)$

(with high probability under certain assumptions)

Main Takeaway: SGD has much slower asymptotic convergence (i.e. it's slower in theory), but is often much faster in practice.

SGD FOR
LINEAR REGRESSION

Linear Regression as Function Approximation

Recall...

$\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^{N}$
where $\mathbf{x} \in \mathbb{R}^M$ and $y \in \mathbb{R}$

1. Assume $\mathcal{D}$ generated as:

$$\mathbf{x}^{(i)} \sim p^*(\cdot)$$
$$y^{(i)} = h^*(\mathbf{x}^{(i)})$$

2. Choose hypothesis space, $\mathcal{H}$:
all linear functions in M-dimensional space

$$\mathcal{H} = \{h_{\boldsymbol{\theta}} : h_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}^T \mathbf{x}, \boldsymbol{\theta} \in \mathbb{R}^M\}$$

3. Choose an objective function:
mean squared error (MSE)

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

4. Solve the unconstrained optimization problem via favorite method:
   - *gradient descent*
   - *closed form*
   - *stochastic gradient descent*
   - ...

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, J(\boldsymbol{\theta})$$

5. Test time: given a new $\mathbf{x}$, make prediction $\hat{y}$

$$\hat{y} = h_{\hat{\boldsymbol{\theta}}}(\mathbf{x}) = \hat{\boldsymbol{\theta}}^T \mathbf{x}$$

Gradient Calculation for Linear Regression

Recall...

Derivative of $J^{(i)}(\boldsymbol{\theta})$:

$$\frac{d}{d\theta_k} J^{(i)}(\boldsymbol{\theta}) = \frac{d}{d\theta_k} \frac{1}{2}(\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)})^2$$
$$= \frac{1}{2} \frac{d}{d\theta_k}(\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)})^2$$
$$= (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) \frac{d}{d\theta_k}(\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)})$$
$$= (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) \frac{d}{d\theta_k}\left(\sum_{j=1}^{K} \theta_j x_j^{(i)} - y^{(i)}\right)$$
$$= (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) x_k^{(i)}$$

Derivative of $J(\boldsymbol{\theta})$:

$$\frac{d}{d\theta_k} J(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} \frac{d}{d\theta_k} J^{(i)}(\boldsymbol{\theta})$$
$$= \frac{1}{N} \sum_{i=1}^{N} (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) x_k^{(i)}$$

Gradient of $J^{(i)}(\boldsymbol{\theta})$ [used by SGD]

$$\nabla_{\boldsymbol{\theta}} J^{(i)}(\boldsymbol{\theta}) = \begin{bmatrix} \frac{d}{d\theta_1} J^{(i)}(\boldsymbol{\theta}) \\ \frac{d}{d\theta_2} J^{(i)}(\boldsymbol{\theta}) \\ \vdots \\ \frac{d}{d\theta_M} J^{(i)}(\boldsymbol{\theta}) \end{bmatrix} = \begin{bmatrix} (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) x_1^{(i)} \\ (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) x_2^{(i)} \\ \vdots \\ (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) x_N^{(i)} \end{bmatrix}$$
$$= (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) \mathbf{x}^{(i)}$$

Gradient of $J(\boldsymbol{\theta})$ [used by Gradient Descent]

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \begin{bmatrix} \frac{d}{d\theta_1} J(\boldsymbol{\theta}) \\ \frac{d}{d\theta_2} J(\boldsymbol{\theta}) \\ \vdots \\ \frac{d}{d\theta_M} J(\boldsymbol{\theta}) \end{bmatrix} = \begin{bmatrix} \frac{1}{N} \sum_{i=1}^{N} (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) x_1^{(i)} \\ \frac{1}{N} \sum_{i=1}^{N} (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) x_2^{(i)} \\ \frac{1}{N} \vdots \\ \frac{1}{N} \sum_{i=1}^{N} (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) x_M^{(i)} \end{bmatrix}$$
$$= \frac{1}{N} \sum_{i=1}^{N} (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) \mathbf{x}^{(i)}$$

SGD for Linear Regression

SGD applied to Linear Regression is called the "Least Mean Squares" algorithm

Algorithm 1 Least Mean Squares (LMS)

procedure LMS($\mathcal{D}$, $\boldsymbol{\theta}^{(0)}$)
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(0)}$ ▷ Initialize parameters
    while not converged **do**
        for $i \in$ shuffle($\{1, 2, \ldots, N\}$) **do**
            $\mathbf{g} \leftarrow (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)})\mathbf{x}^{(i)}$ ▷ Compute gradient
            $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \gamma \mathbf{g}$ ▷ Update parameters
    return $\boldsymbol{\theta}$

GD 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($\mathcal{D}$, $\boldsymbol{\theta}^{(0)}$)
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(0)}$ ▷ Initialize parameters
    while not converged **do**
        $\mathbf{g} \leftarrow \frac{1}{N} \sum_{i=1}^{N} (\boldsymbol{\theta}^T \mathbf{x}^{(i)} - y^{(i)}) \mathbf{x}^{(i)}$ ▷ Compute gradient
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \gamma \mathbf{g}$ ▷ Update parameters
    return $\boldsymbol{\theta}$

Solving Linear Regression

Question:

True or False: If Mean Squared Error (i.e. $\frac{1}{N}\sum_{i=1}^{N}(y^{(i)} - h(\mathbf{x}^{(i)}))^2$) has a unique minimizer (i.e. argmin), then Mean Absolute Error (i.e. $\frac{1}{N}\sum_{i=1}^{N}|y^{(i)} - h(\mathbf{x}^{(i)})|$) must also have a unique minimizer.

Answer:

Optimization Objectives

You should be able to…

- Apply gradient descent to optimize a function
- Apply stochastic gradient descent (SGD) to optimize a function
- Apply knowledge of zero derivatives to identify a closed-form solution (if one exists) to an optimization problem
- Distinguish between convex, concave, and nonconvex functions
- Obtain the gradient (and Hessian) of a (twice) differentiable function