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

Henry Chai & Matt Gormley

9/25/23

Exam 1 Logistics

- Exam 1 on 2/19 (next Monday!) from 7 PM – 9 PM
- 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

- Review the exam practice problems (released 2/12 on the course website, under Coursework)
- Attend the dedicated exam 1 review OH (in lieu of recitation on 2/16)
- 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

$\mathcal{D} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1}^{N}$

where $\mathbf{x} \in \mathbb{R}^M$ and $y \in \mathbb{R}$

Linear Regression as Function Approximation

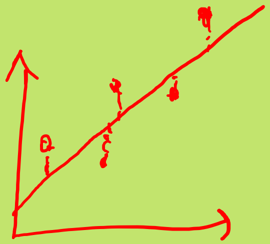
1. Assume $\mathcal{D}$ generated as:

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

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 $\hat{y}$

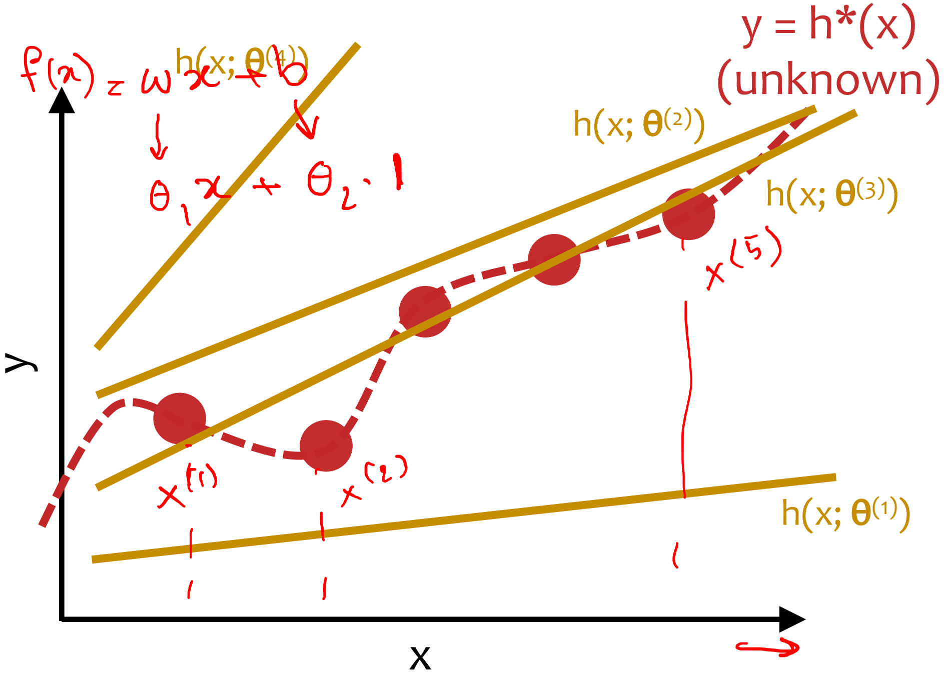
$$\hat{y} = h_{\hat{\boldsymbol{\theta}}}(\mathbf{x}) = \hat{\boldsymbol{\theta}}^T \mathbf{x}$$

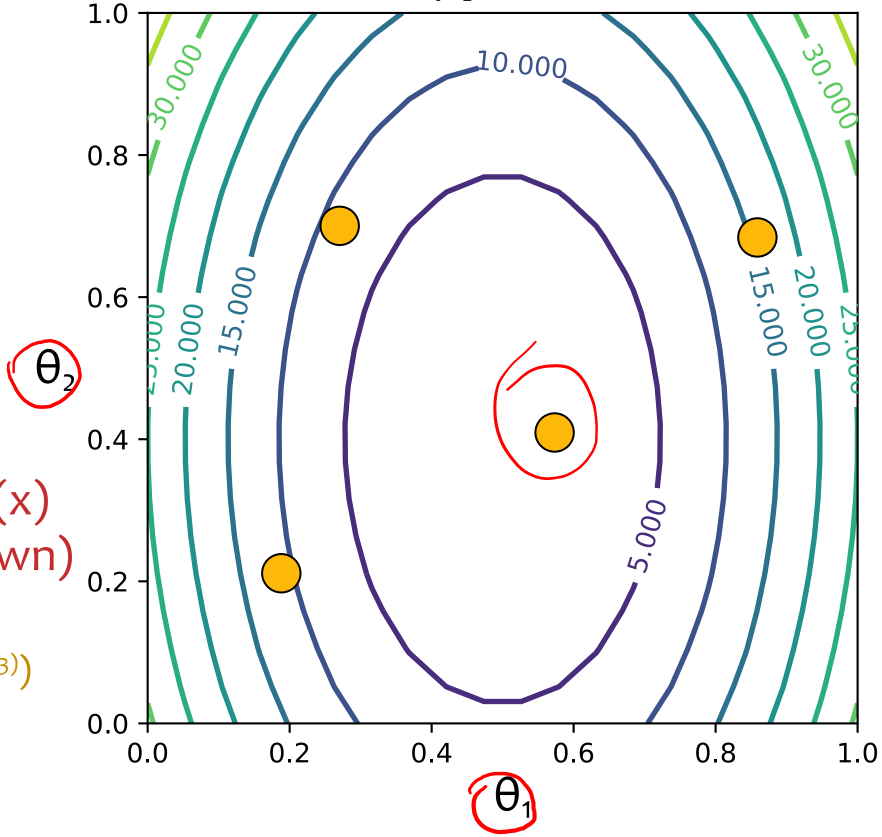
Linear Regression by Rand. Guessing

Optimization Method #0: Random Guessing

1. Pick a random $\boldsymbol{\theta}$
2. Evaluate $J(\boldsymbol{\theta})$
3. Repeat steps 1 and 2 many times
4. Return $\boldsymbol{\theta}$ that gives smallest $J(\boldsymbol{\theta})$

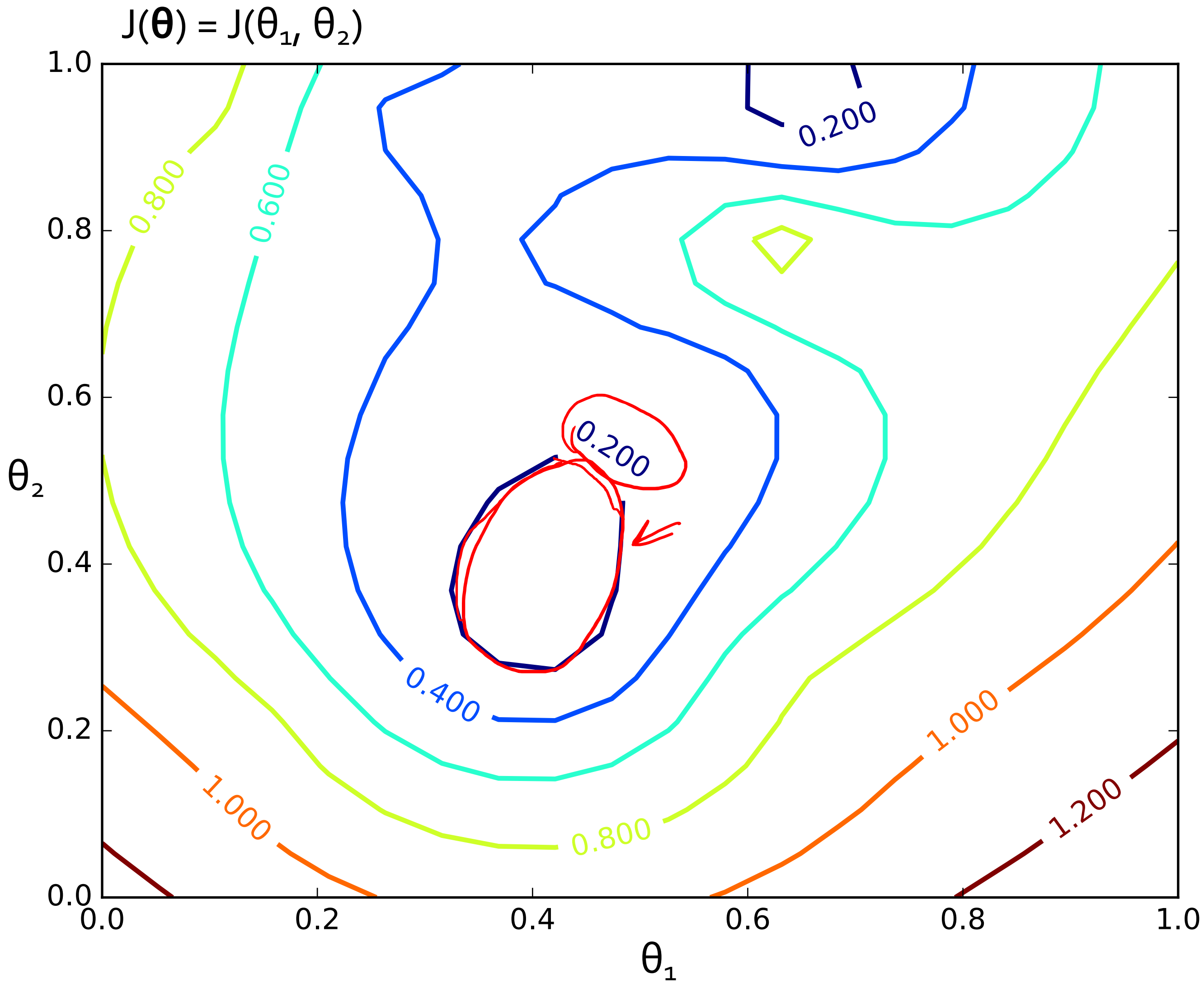
$$J(\boldsymbol{\theta}) = J(\theta_1, \theta_2) = \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - \boldsymbol{\theta}^T \mathbf{x}^{(i)}\right)^2$$



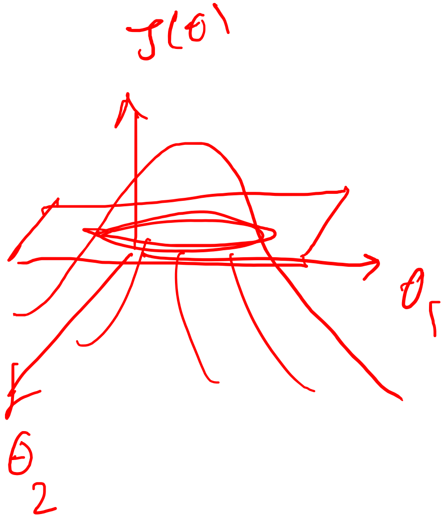


t	θ_1	θ_2	$J(\theta_1, \theta_2)$
1	0.2	0.2	10.4
2	0.3	0.7	7.2
3	0.6	0.4	1.0
4	0.9	0.7	16.2

Gradients

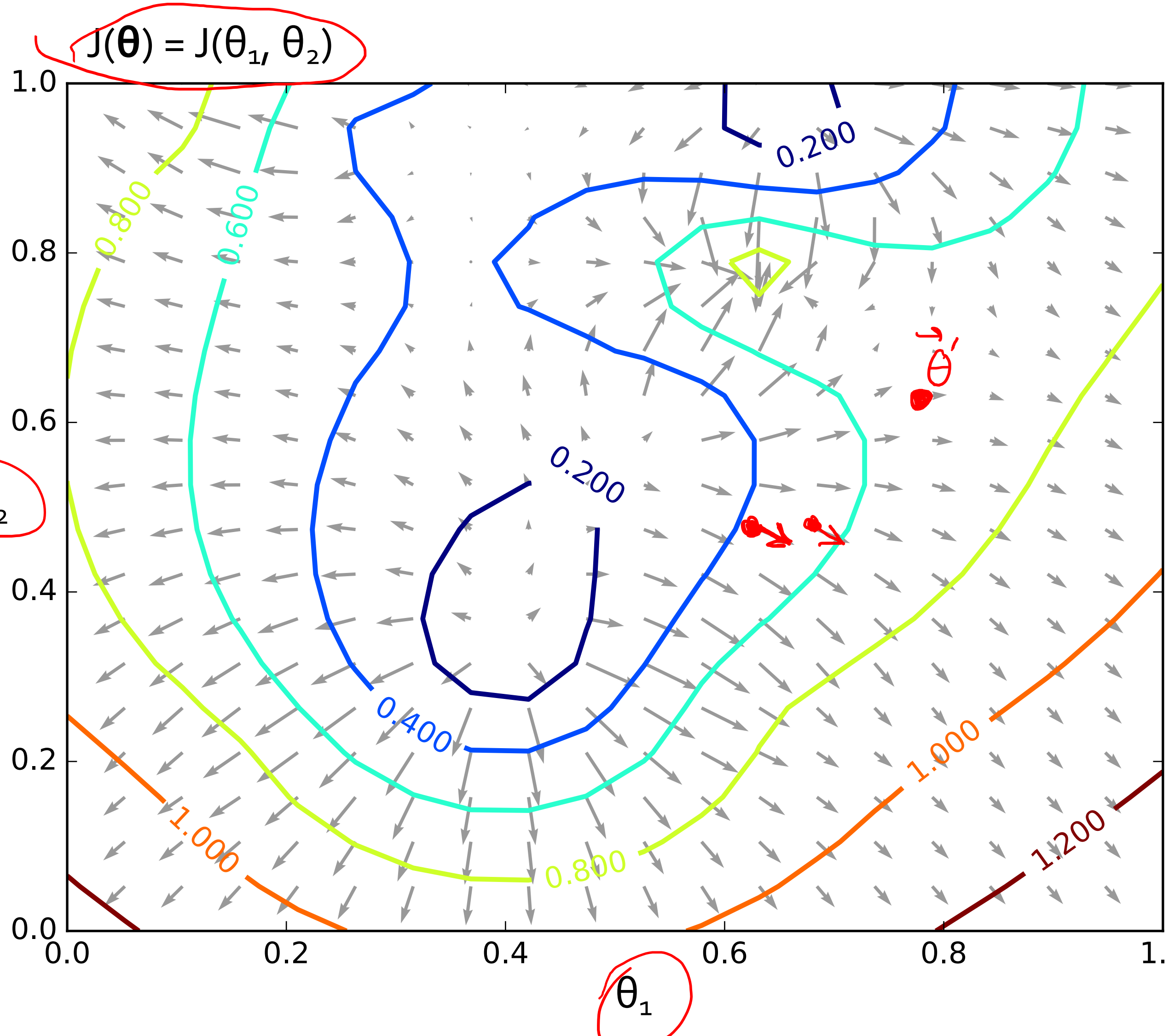


J(θ) = J(θ1, θ2)
θ2
θ1
0.200
0.200
0.400
0.600
0.800
0.800
1.000
1.000
1.200
0.0
0.2
0.4
0.6
0.8
1.0



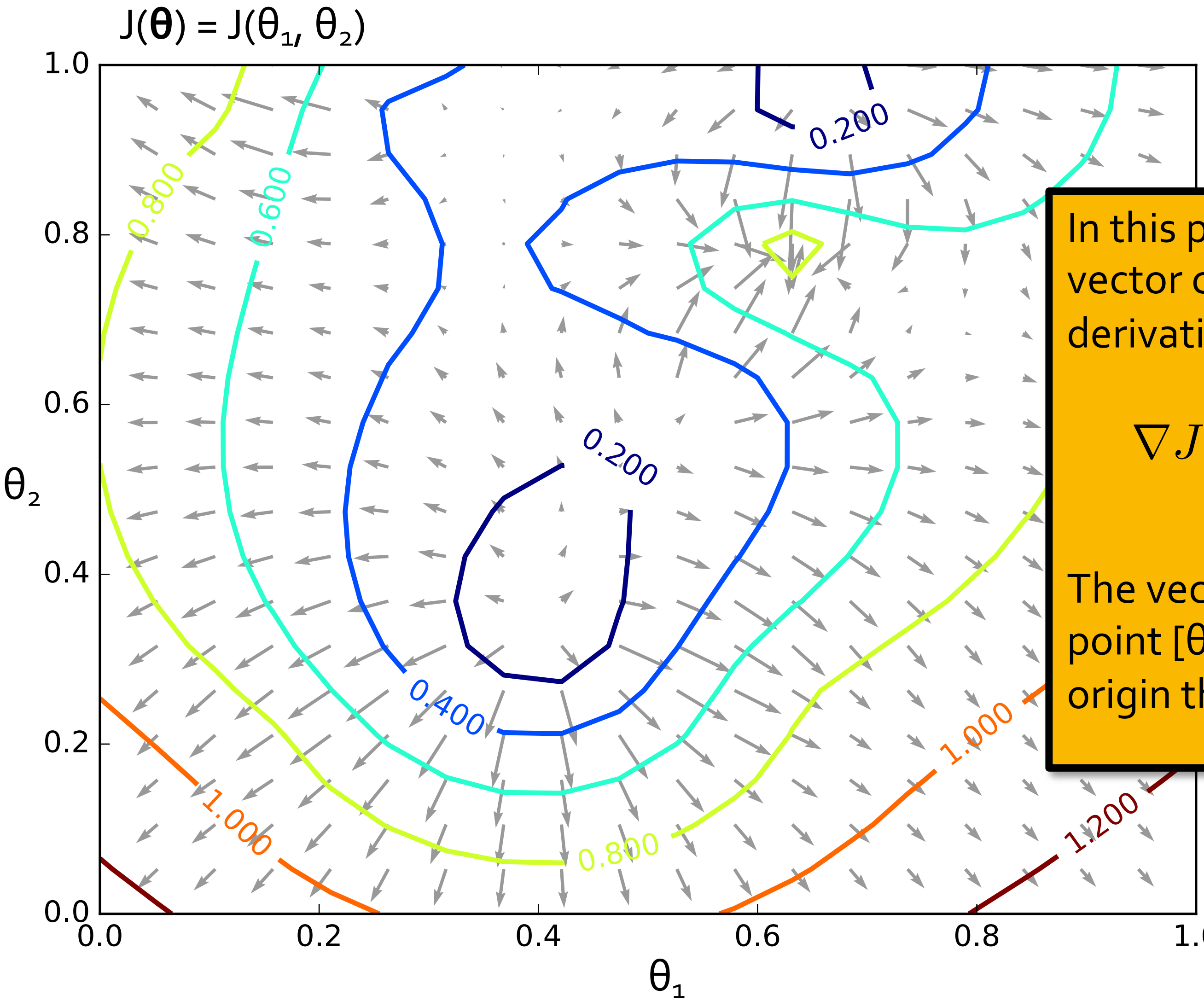
J(θ)
θ1
θ2

Gradients



These are the **gradients** that Gradient **Ascent** would follow.

Gradients



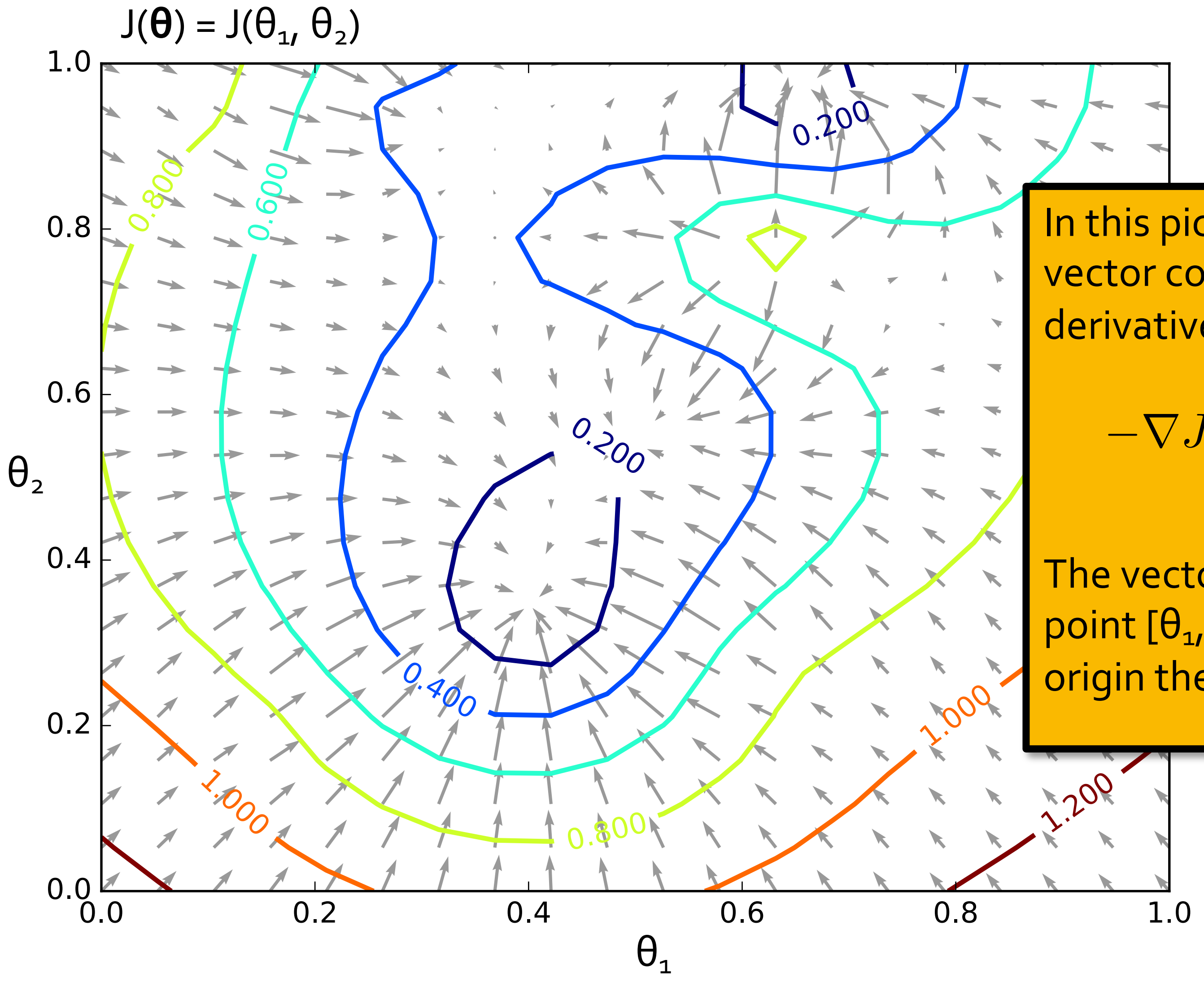
In this picture, each arrow is a 2D vector consisting of two partial derivatives.

$$\nabla J(\theta_1, \theta_2) = \begin{bmatrix} \frac{\partial J}{\partial \theta_1} \\ \\ \frac{\partial J}{\partial \theta_2} \end{bmatrix}$$

The vector is evaluated at the point $[\theta_1, \theta_2]^T$ and plotted with its origin there as well.

These are the **gradients** that Gradient **Ascent** would follow.

(Negative) Gradients



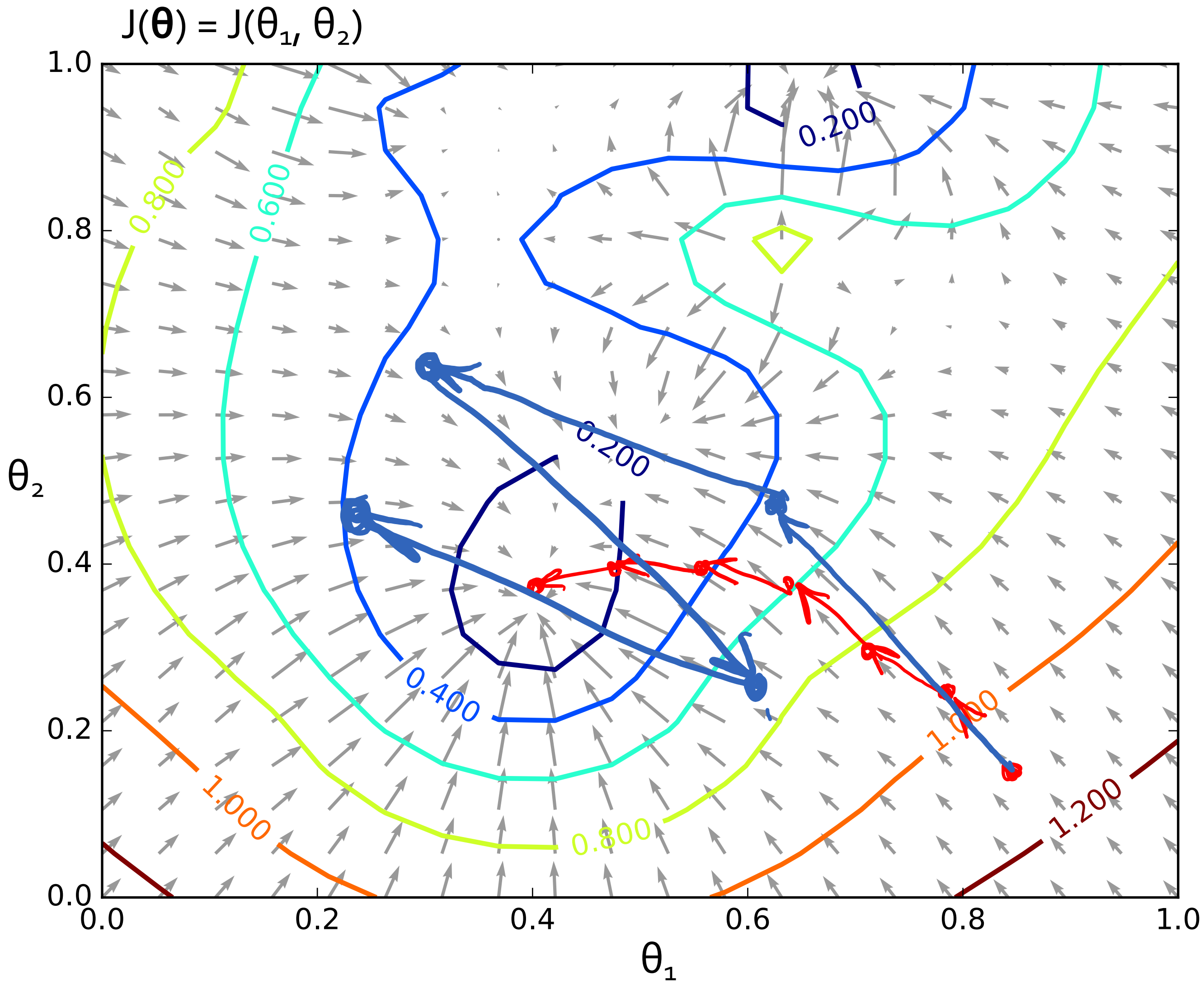
In this picture, each arrow is a 2D vector consisting of two partial derivatives.

$$-\nabla J(\theta_1, \theta_2) = \begin{bmatrix} -\frac{\partial J}{\partial \theta_1} \\ -\frac{\partial J}{\partial \theta_2} \end{bmatrix}$$

The vector is evaluated at the point $[\theta_1, \theta_2]^T$ and plotted with its origin there as well.

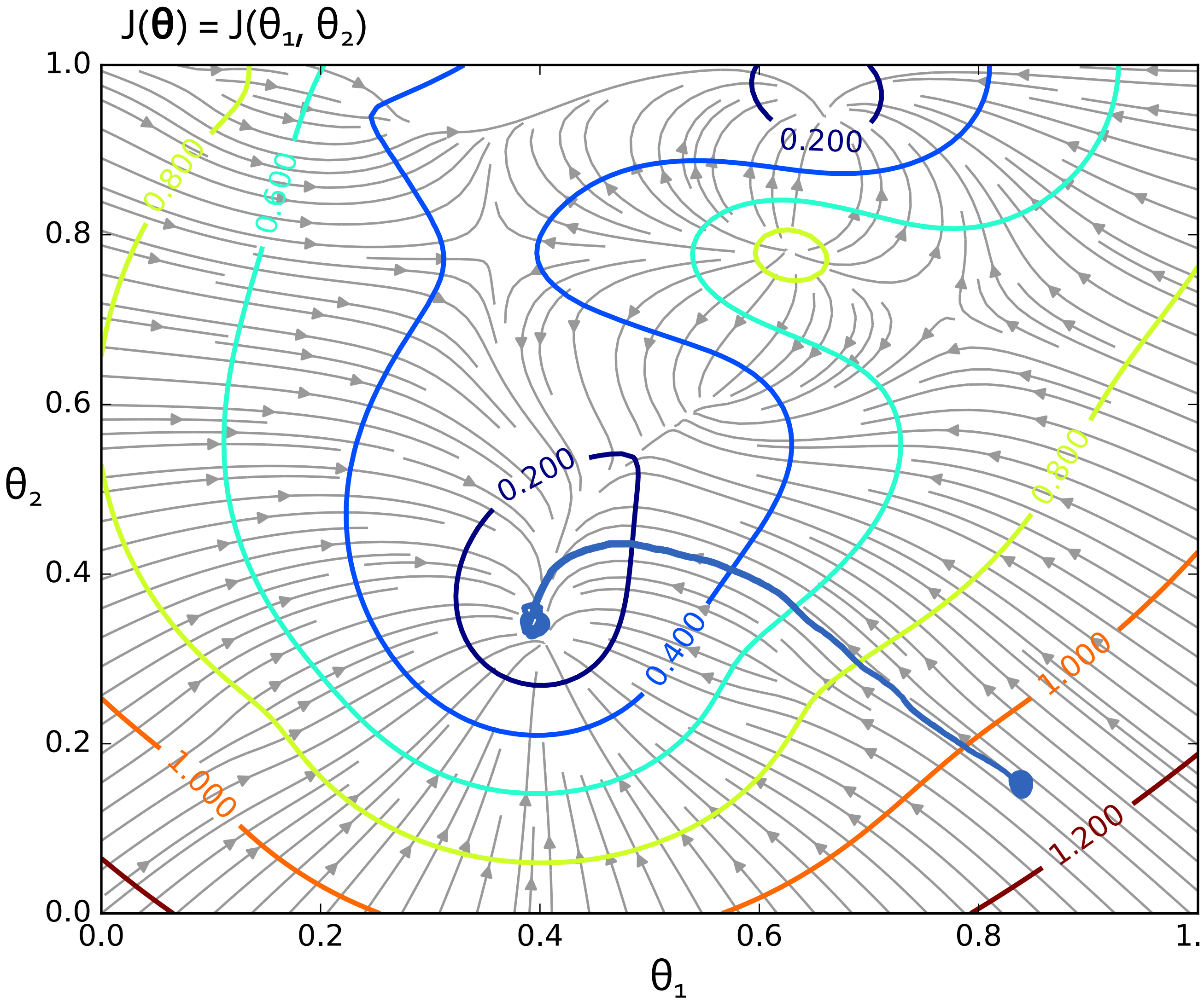
These are the **negative** gradients that Gradient **Descent** would follow.

(Negative) Gradients



These are the **negative** gradients that Gradient **Descent** would follow.

(Negative) Gradient *Pa*



Shown are the **paths** that Gradient Descent would follow if it were making **infinitesimally small steps**.

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 − γ g                                    ▷ Update parameters
    return θ
```

$\nabla J(\theta)$

step size

learning rate

$\nabla J = \vec{0}$

$\|\nabla J\| \approx 0$

Gradient Calculation for Linear Regression

i → training instance i ∈ 1,…,N
k → feature k ∈ 1,…,M

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

(4)

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

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

(3)

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

(2)

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

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

$\frac{\partial J}{\partial \theta_k}$, k=1,…,M

(1) $MSE = J(\vec{\theta}) = \frac{1}{2N} \sum_{i=1}^{N} (y^{(i)} - \vec{\theta} \cdot \vec{x}^{(i)})^2$

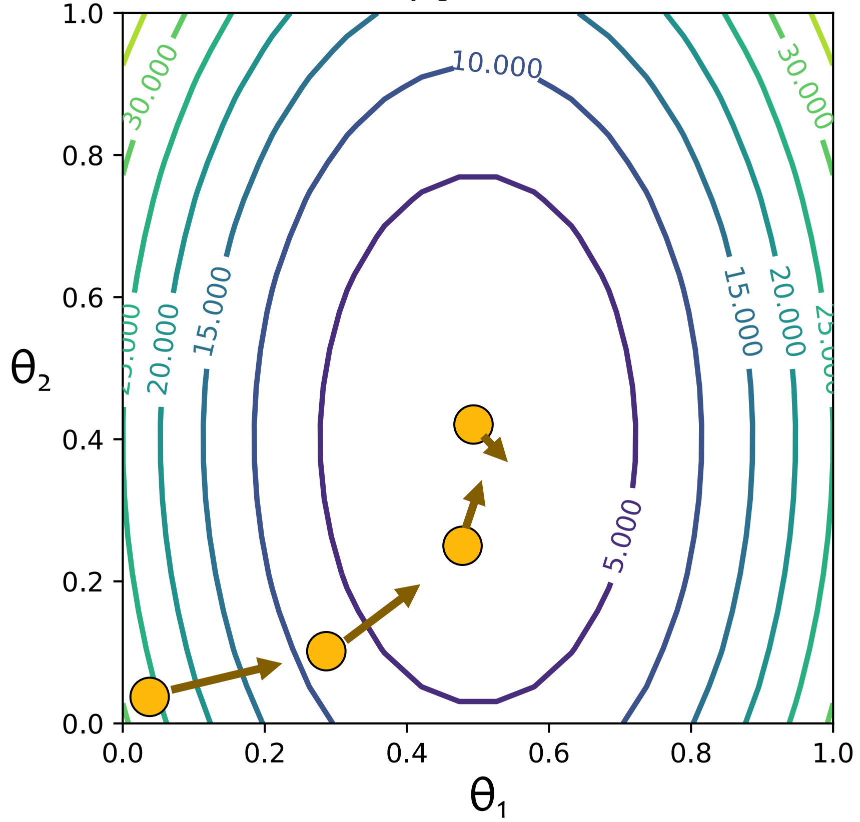
$\forall i = 1, \ldots, N: J(\vec{\theta}) = \frac{1}{2N} \sum_{i=1}^{N} J^{(i)}(\theta)$ $J^{(i)}(\theta)$

Linear Regression by Gradient Desc.

Optimization Method #1: Gradient Descent

1. Pick a random $\boldsymbol{\theta}$
2. Repeat:
   a. Evaluate gradient $\nabla J(\boldsymbol{\theta})$
   b. Step opposite gradient
3. Return $\boldsymbol{\theta}$ that gives smallest $J(\boldsymbol{\theta})$

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

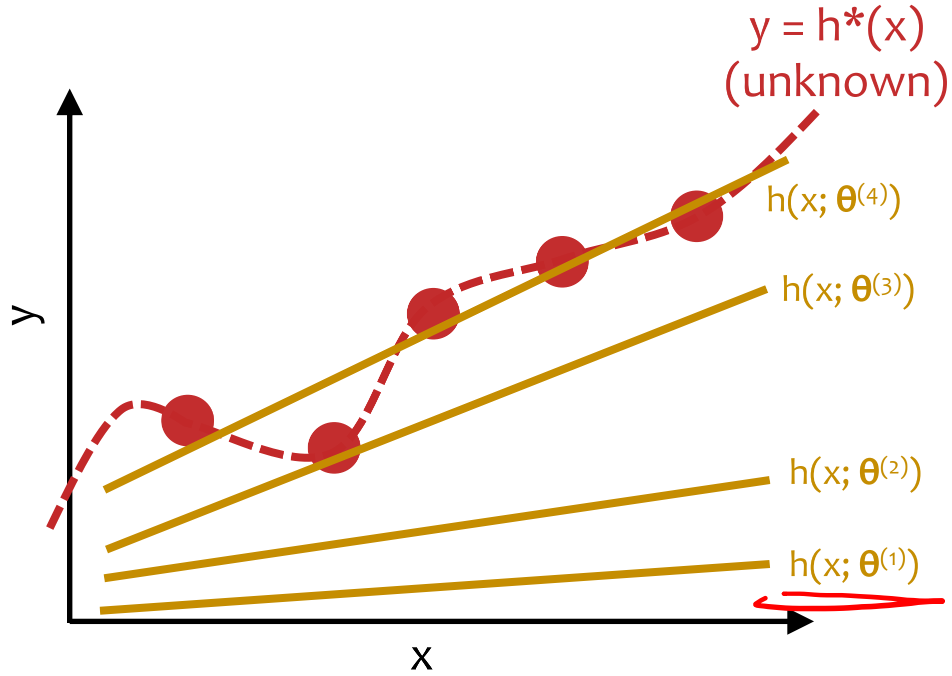


t	θ_1	θ_2	$J(\theta_1, \theta_2)$
1	0.01	0.02	25.2
2	0.30	0.12	8.7
3	0.51	0.30	1.5
4	0.59	0.43	0.2

Linear Regression by Gradient Desc.

Optimization Method #1: Gradient Descent

1. Pick a random $\boldsymbol{\theta}$
2. Repeat:
   a. Evaluate gradient $\nabla J(\boldsymbol{\theta})$
   b. Step opposite gradient
3. Return $\boldsymbol{\theta}$ that gives smallest $J(\boldsymbol{\theta})$



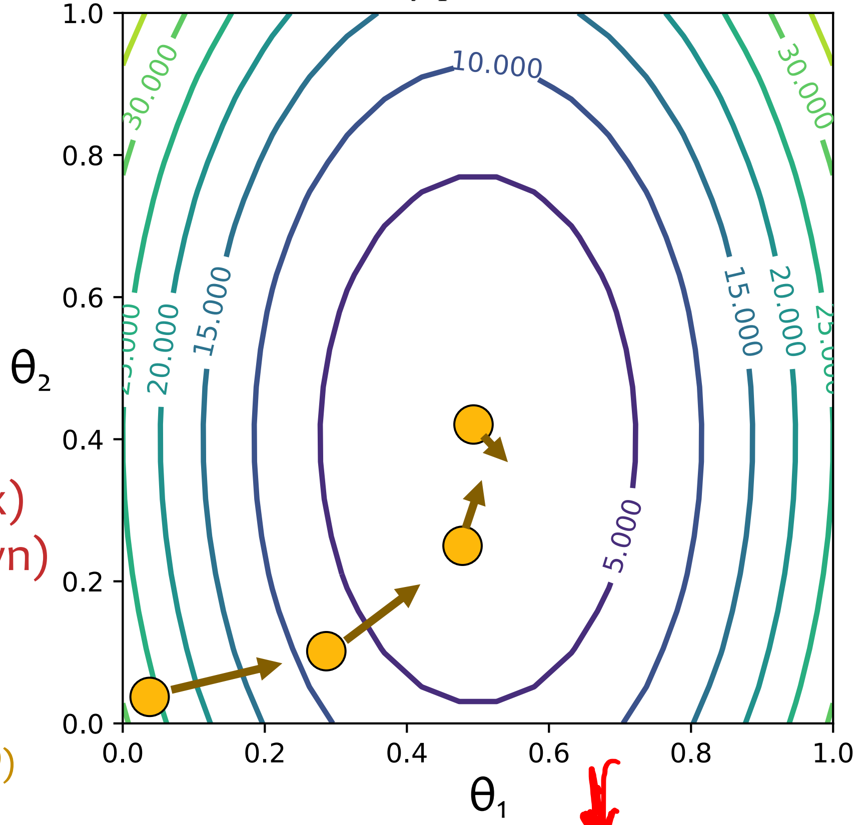
t	θ_1	θ_2	$J(\theta_1, \theta_2)$
1	0.01	0.02	25.2
2	0.30	0.12	8.7
3	0.51	0.30	1.5
4	0.59	0.43	0.2

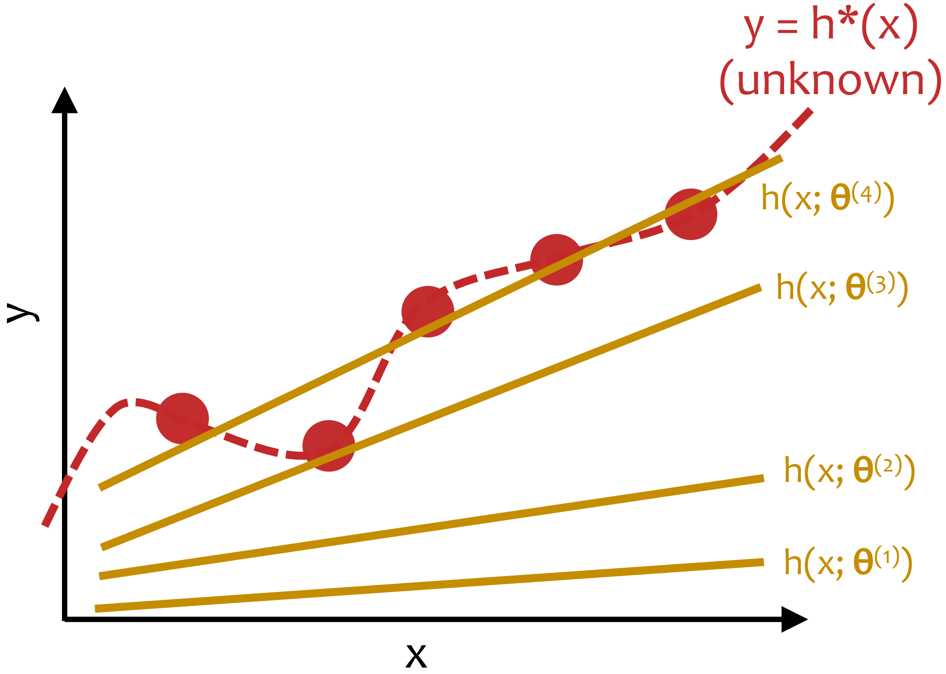
Linear Regression by Gradient Desc.

Optimization Method #1: Gradient Descent

1. Pick a random $\boldsymbol{\theta}$
2. Repeat:
   a. Evaluate gradient $\nabla J(\boldsymbol{\theta})$
   b. Step opposite gradient
3. Return $\boldsymbol{\theta}$ that gives smallest $J(\boldsymbol{\theta})$

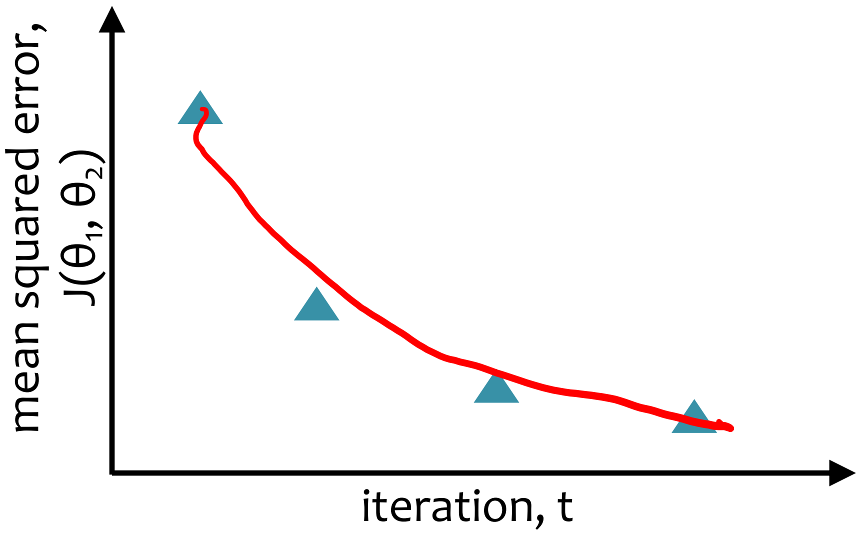
$$J(\boldsymbol{\theta}) = J(\theta_1, \theta_2) = \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - \boldsymbol{\theta}^T\mathbf{x}^{(i)}\right)^2$$

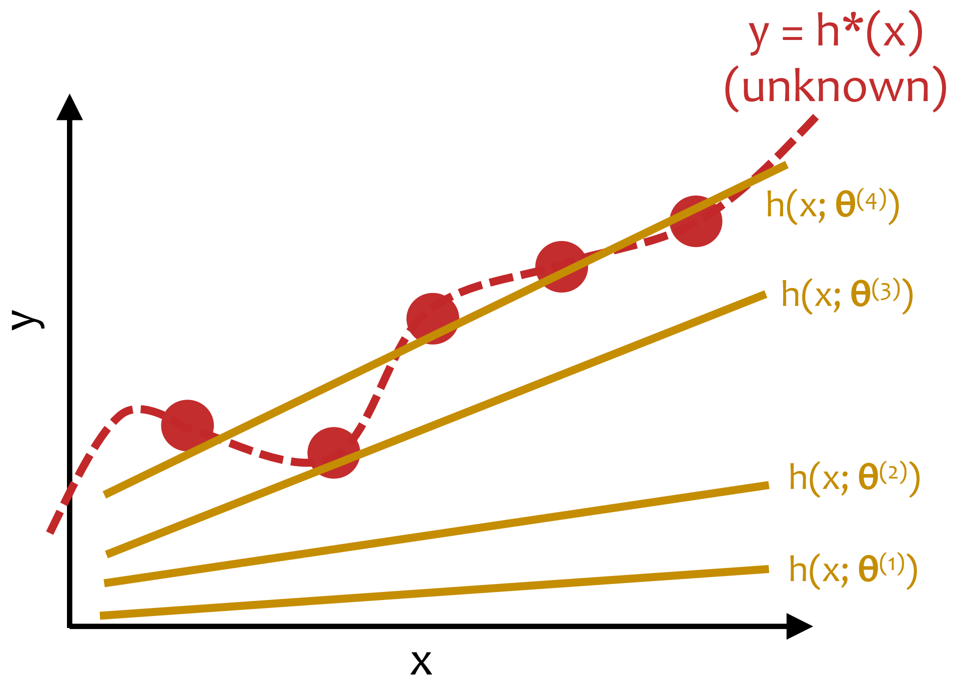




t	θ_1	θ_2	$J(\theta_1, \theta_2)$
1	0.01	0.02	25.2
2	0.30	0.12	8.7
3	0.51	0.30	1.5
4	0.59	0.43	0.2

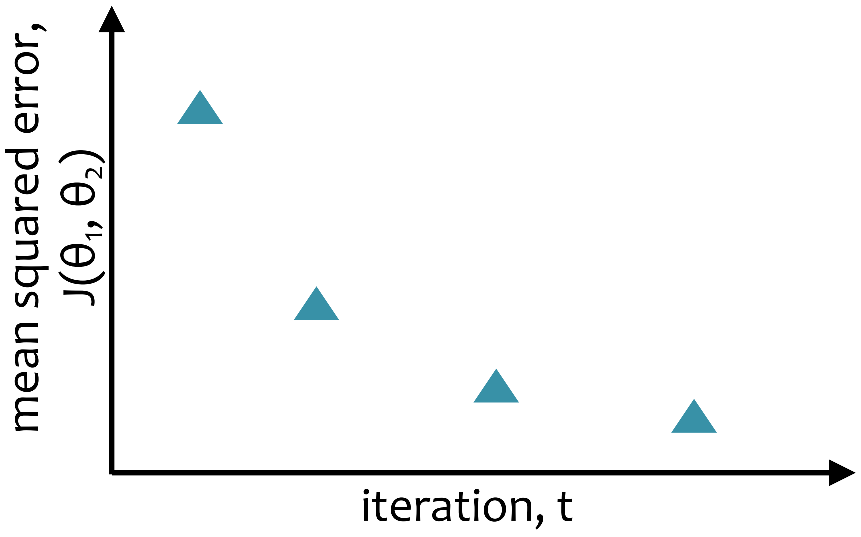
Linear Regression by Gradient Desc.

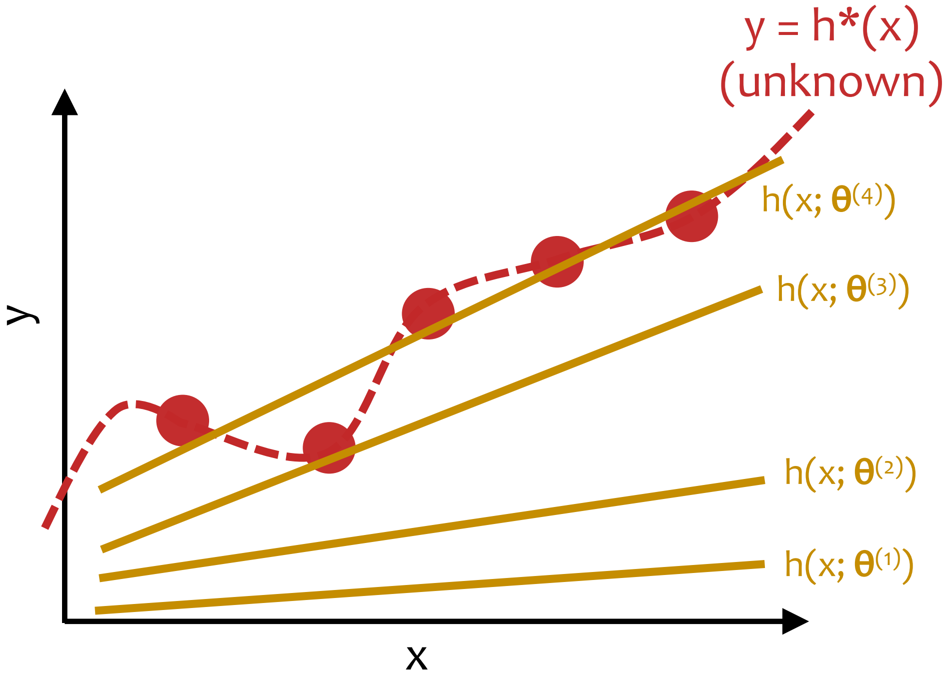




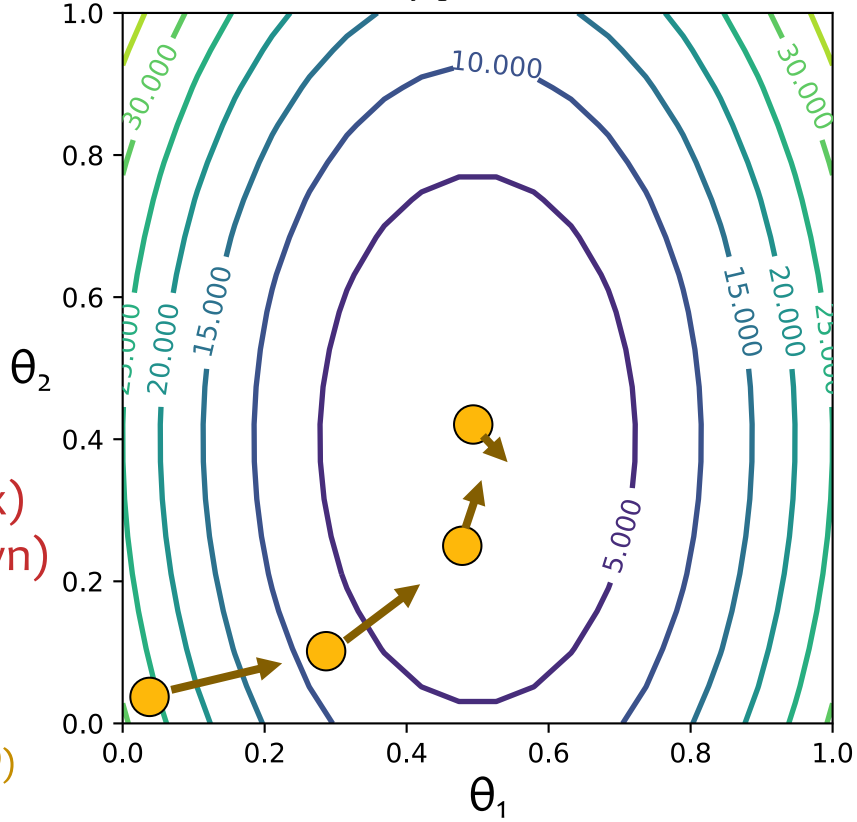
t	θ_1	θ_2	$J(\theta_1, \theta_2)$
1	0.01	0.02	25.2
2	0.30	0.12	8.7
3	0.51	0.30	1.5
4	0.59	0.43	0.2

Linear Regression by Gradient Desc.





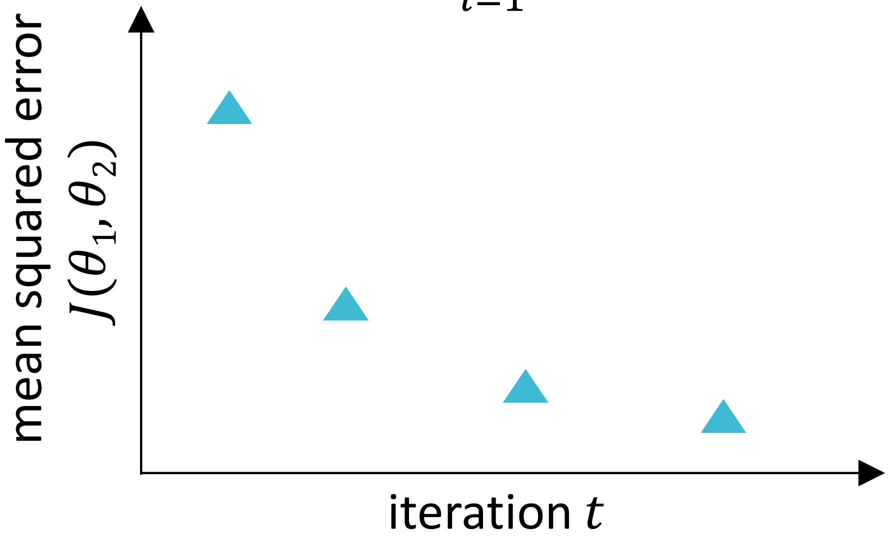
$$J(\boldsymbol{\theta}) = J(\theta_1, \theta_2) = \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - \boldsymbol{\theta}^T \mathbf{x}^{(i)}\right)^2$$

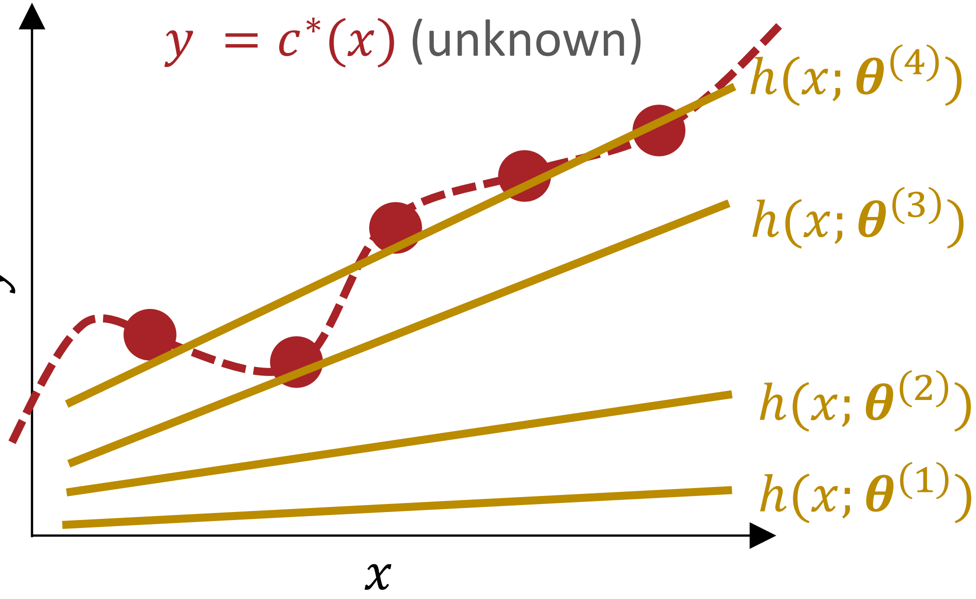


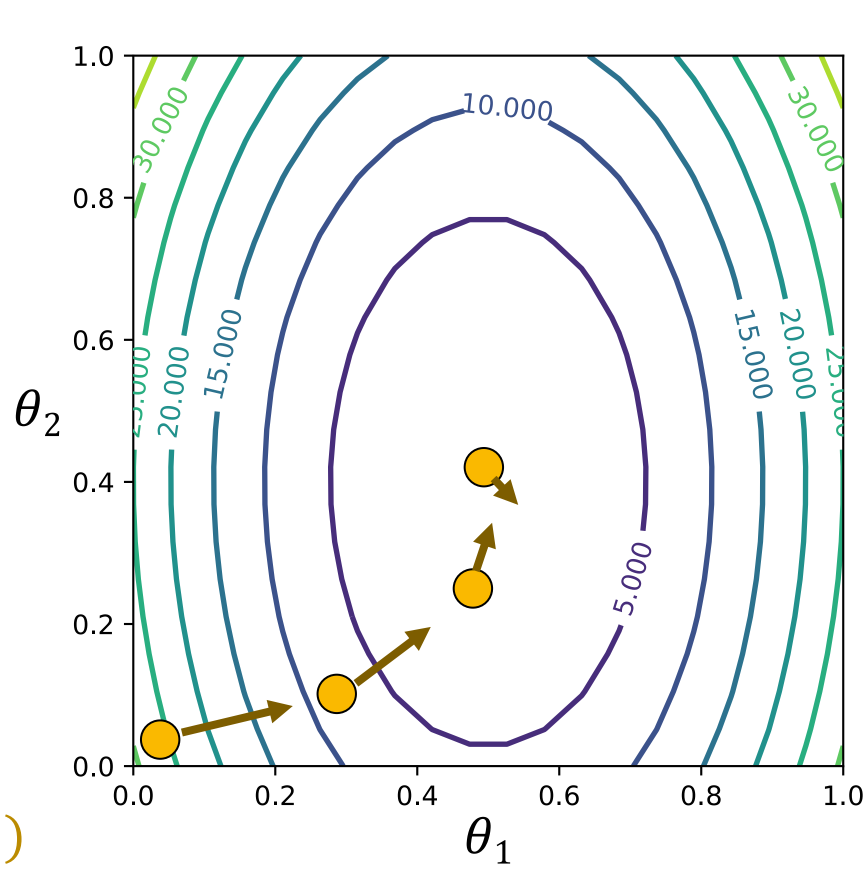
t	θ_1	θ_2	$J(\theta_1, \theta_2)$
1	0.01	0.02	25.2
2	0.30	0.12	8.7
3	0.51	0.30	1.5
4	0.59	0.43	0.2

Why Gradient Descent for Linear Regression?

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





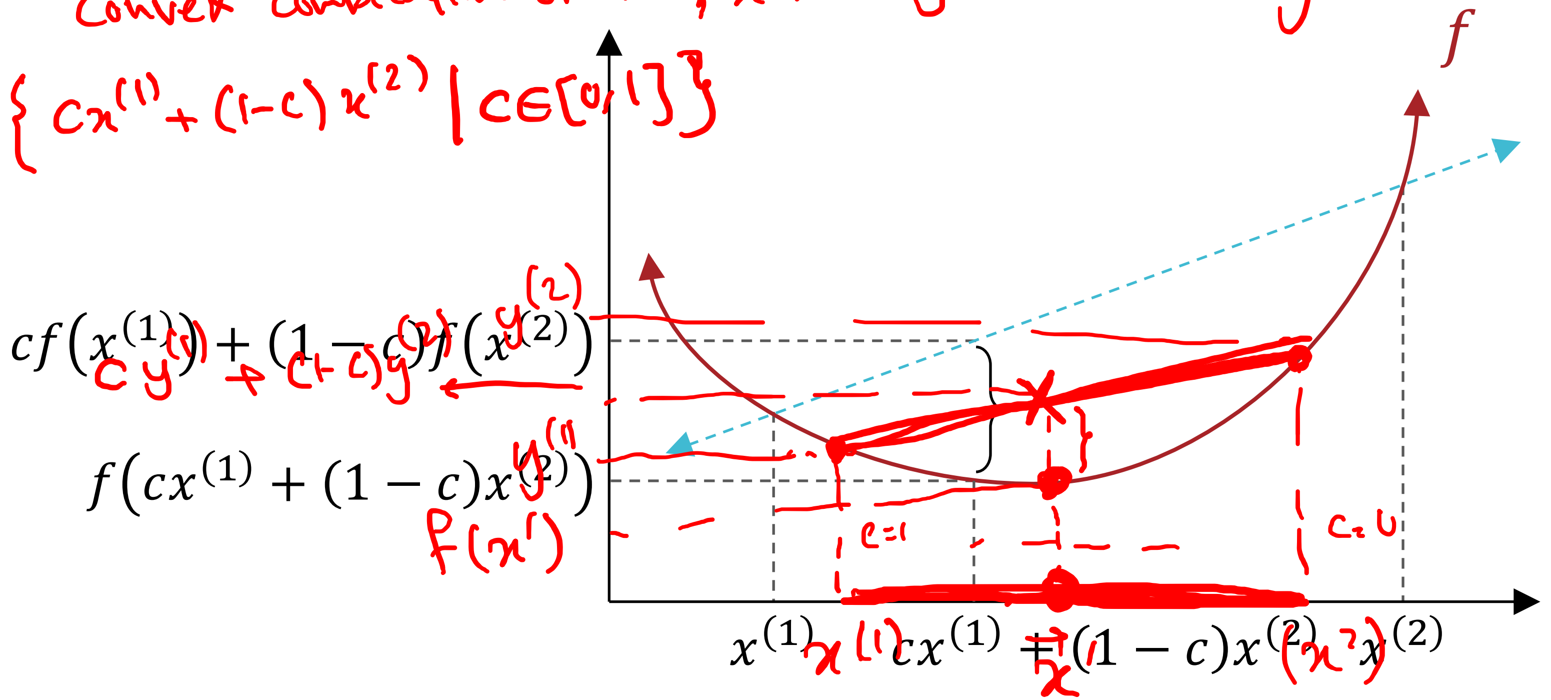


t	θ_1	θ_2	$J(\theta_1, \theta_2)$
1	0.01	0.02	25.2
2	0.30	0.12	8.7
3	0.51	0.30	1.5
4	0.59	0.43	0.2

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 $0 \le c \le 1$

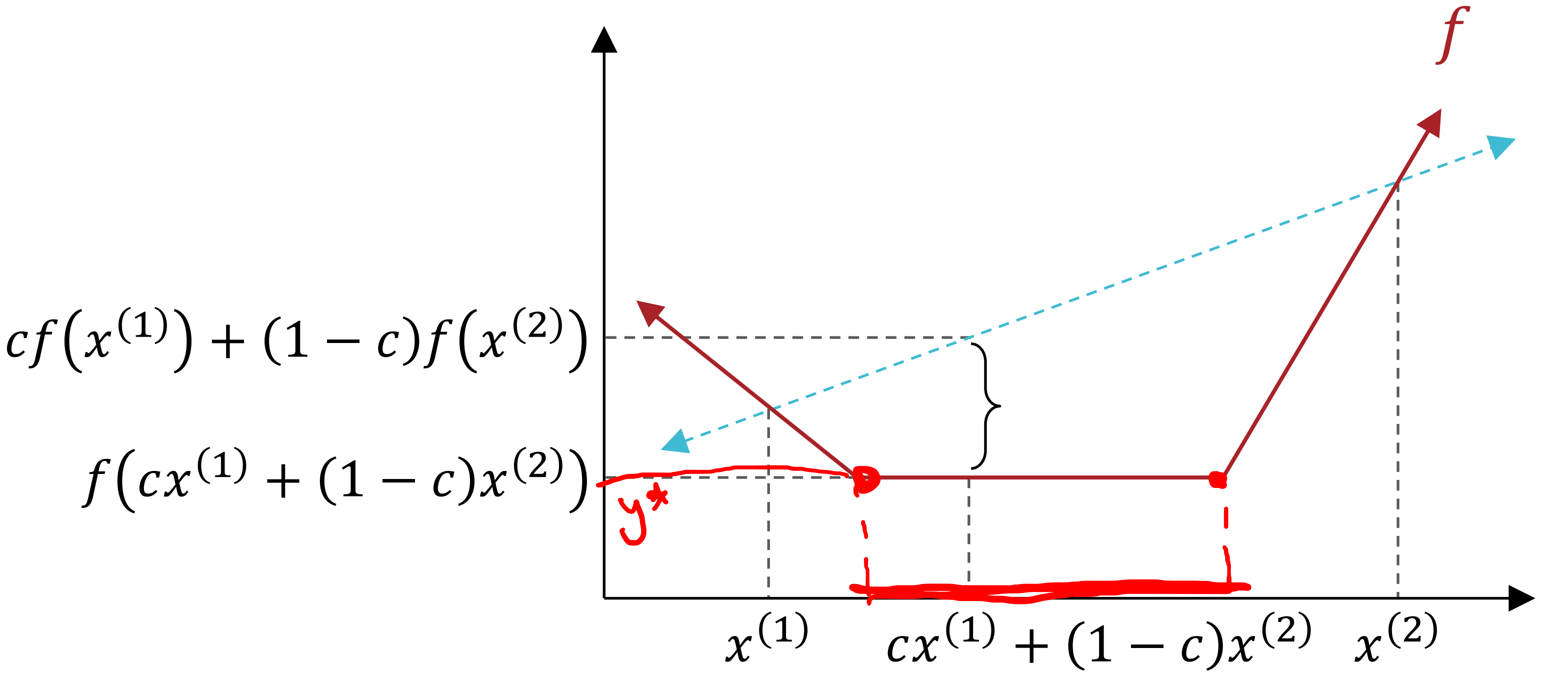
$$f\left(c\boldsymbol{x}^{(1)} + (1-c)\boldsymbol{x}^{(2)}\right) \le c f\left(\boldsymbol{x}^{(1)}\right) + (1-c) f\left(\boldsymbol{x}^{(2)}\right)$$



Convexity

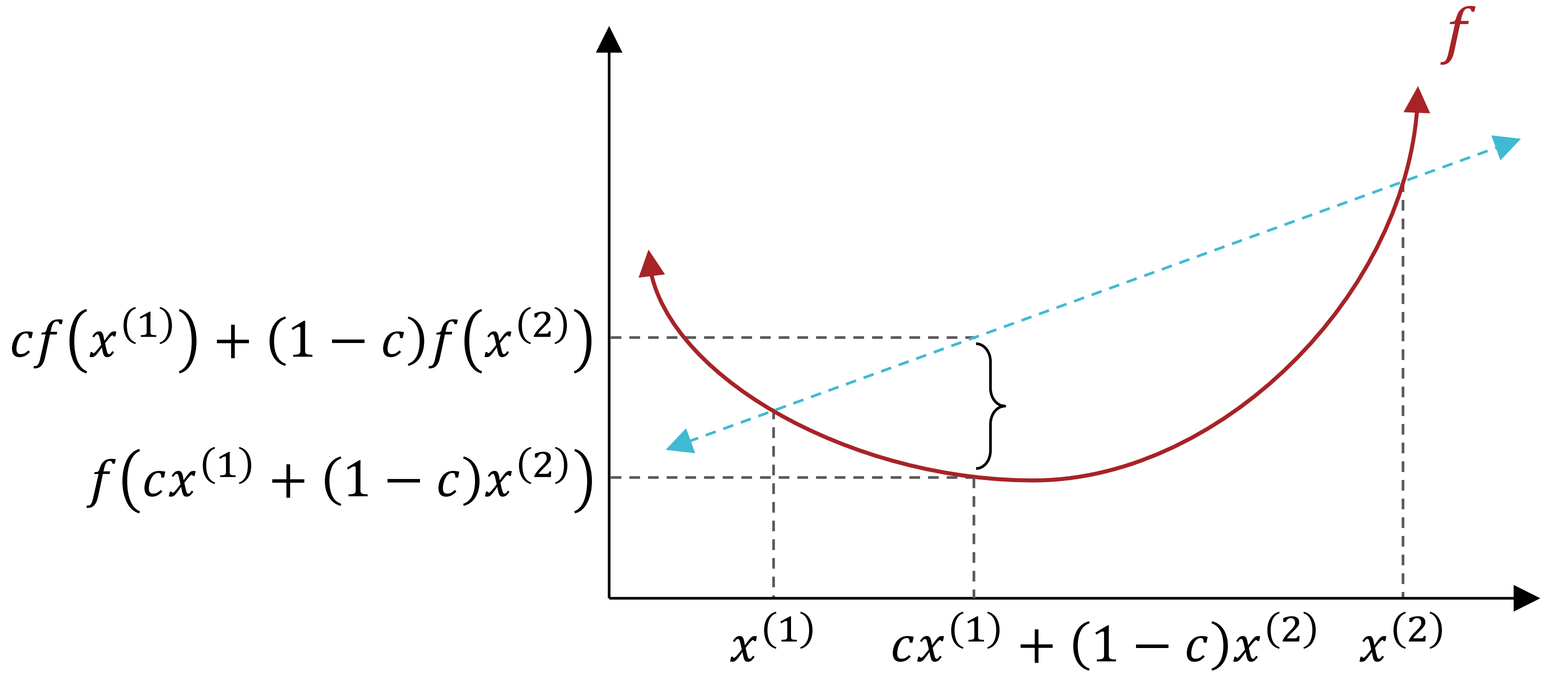
- A function $f: \mathbb{R}^D \rightarrow \mathbb{R}$ is convex if $\forall\, \boldsymbol{x}^{(1)} \in \mathbb{R}^D, \boldsymbol{x}^{(2)} \in \mathbb{R}^D$ and $0 \leq c \leq 1$

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

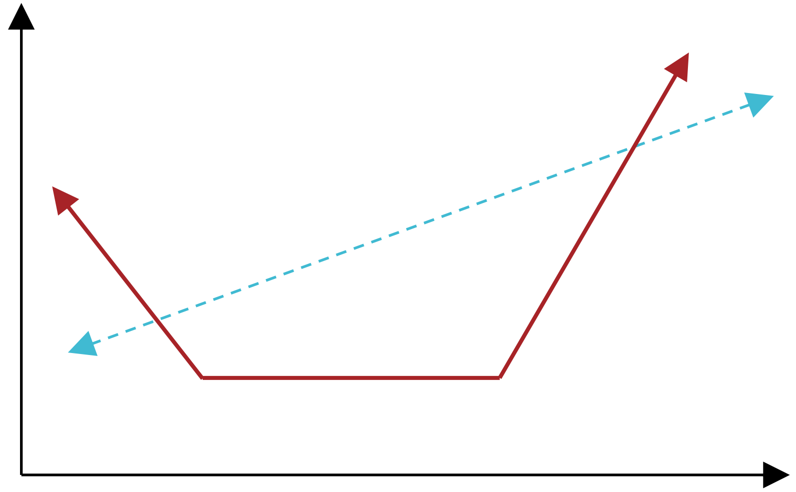


Convexity

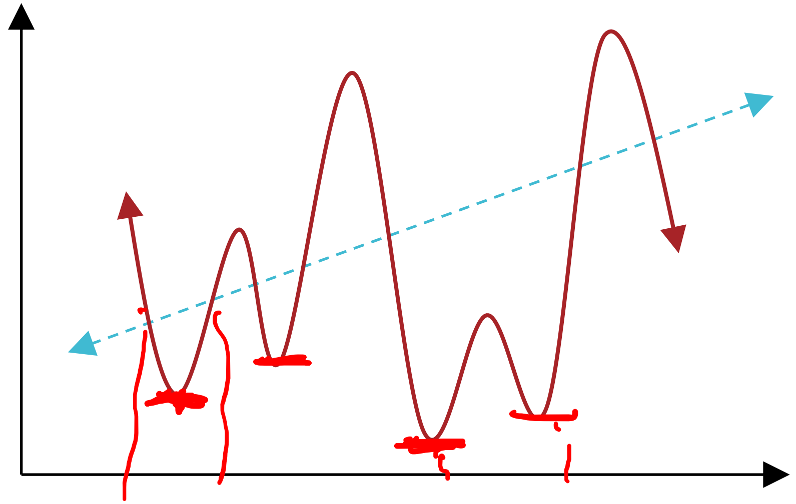
- A function $f: \mathbb{R}^D \rightarrow \mathbb{R}$ is *strictly* convex if
  $\forall\, \boldsymbol{x}^{(1)} \in \mathbb{R}^D, \boldsymbol{x}^{(2)} \in \mathbb{R}^D$ and $0 < c < 1$
  $f\left(c\boldsymbol{x}^{(1)} + (1-c)\boldsymbol{x}^{(2)}\right) < cf\left(\boldsymbol{x}^{(1)}\right) + (1-c)f\left(\boldsymbol{x}^{(2)}\right)$



Convexity

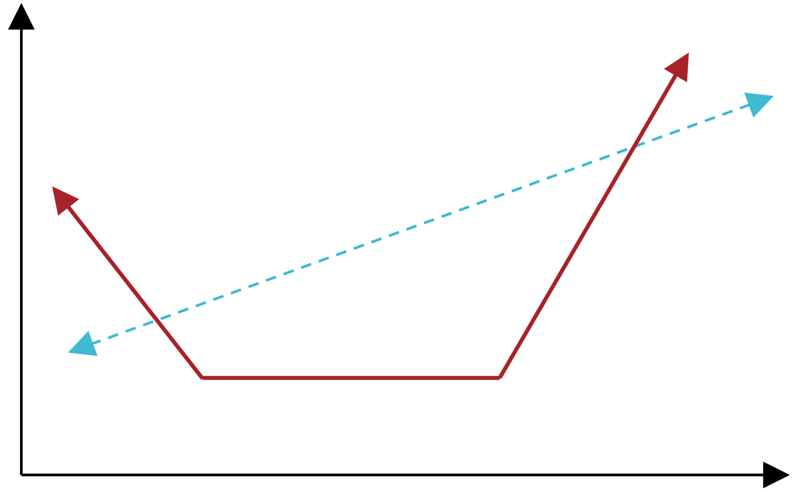


Convex functions



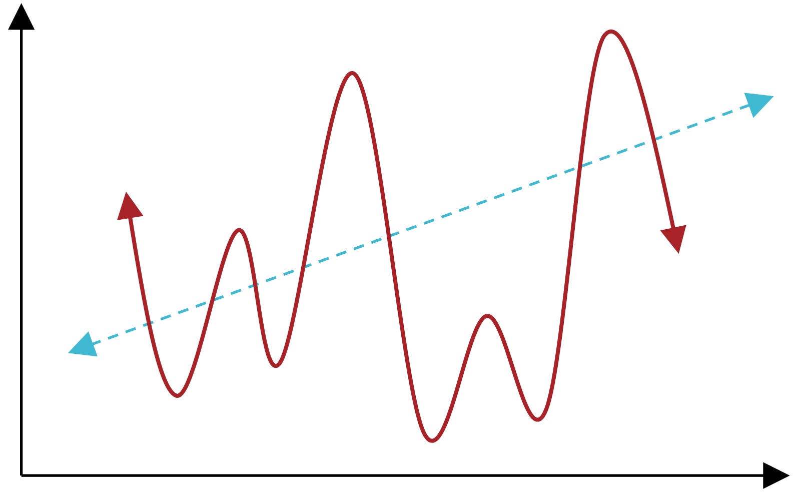
Non-convex functions

Convexity



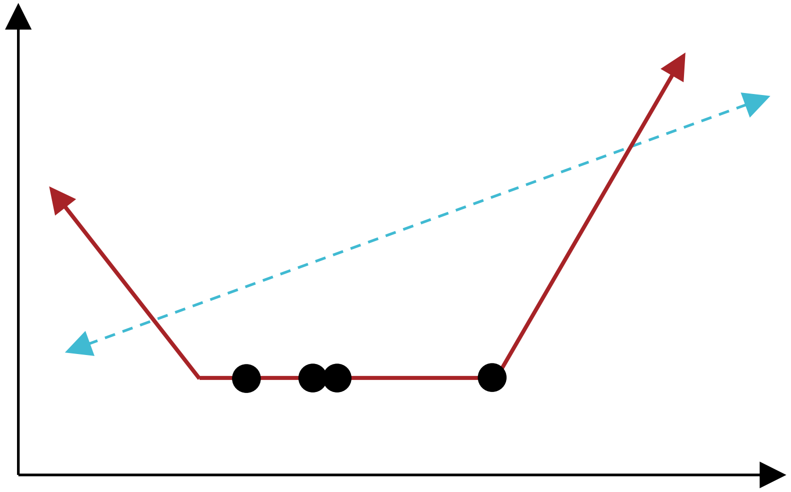
Given a function $f: \mathbb{R}^D \rightarrow \mathbb{R}$

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



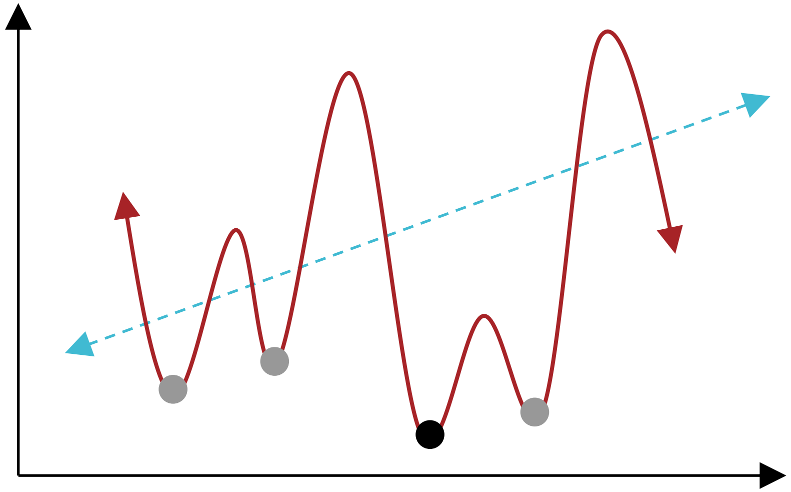
- $\boldsymbol{x}^*$ is a local minimum iff $\exists \; \epsilon$ s.t. $f(\boldsymbol{x}^*) \leq f(\boldsymbol{x}) \; \forall \; \boldsymbol{x}$ s.t. $\|\boldsymbol{x} - \boldsymbol{x}^*\|_2 < \epsilon$

Convexity



Convex functions:

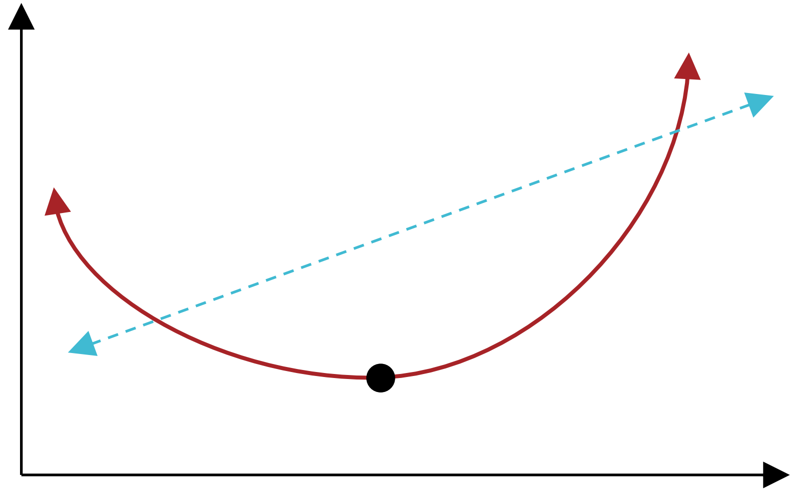
Each local minimum is a global minimum!



Non-convex functions:

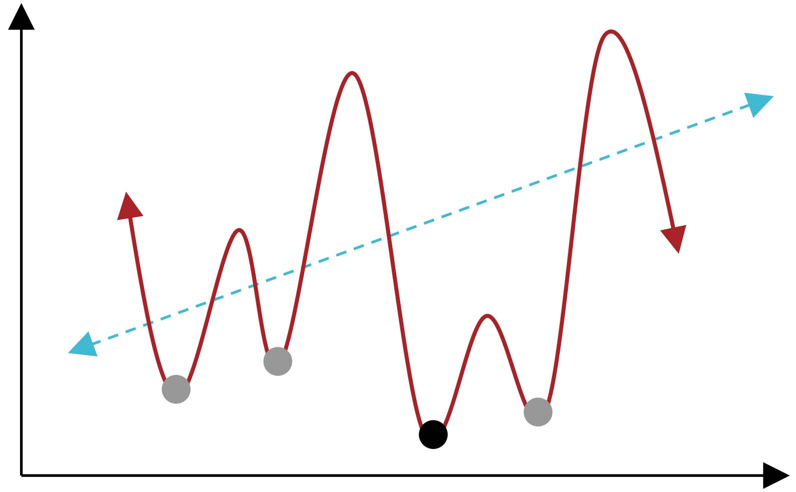
A local minimum may or may not be a global minimum…

Convexity



Strictly convex functions:

There exists a unique global minimum!

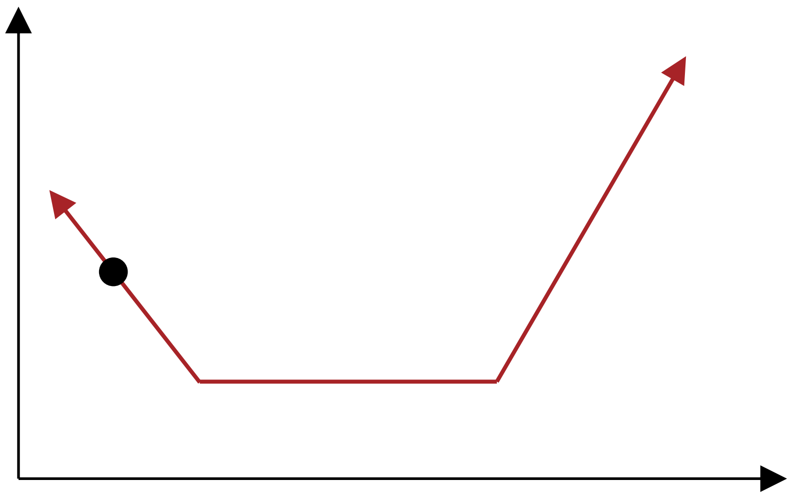


Non-convex functions:

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

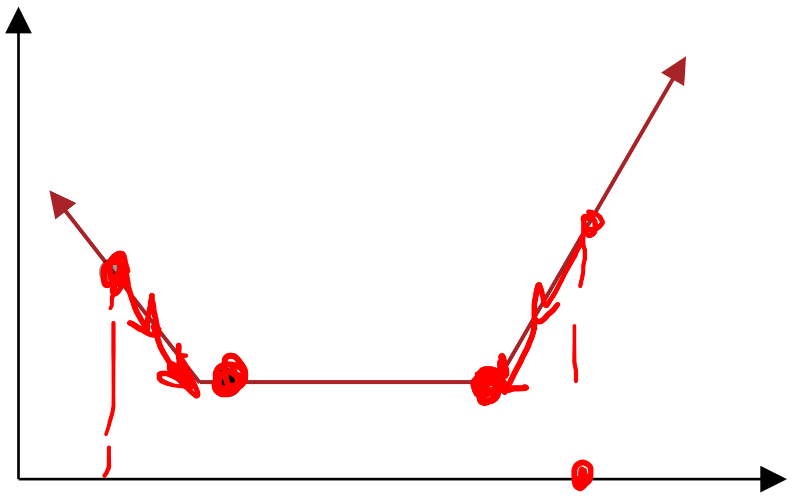
Gradient Descent & Convexity

- 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!



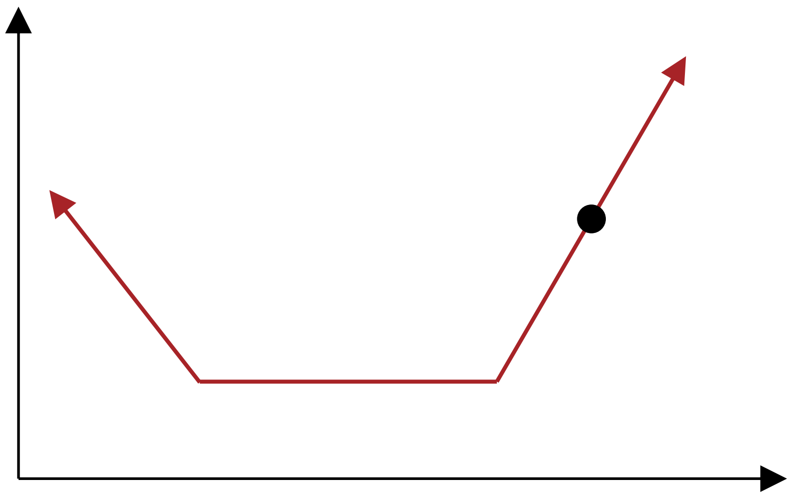
Gradient Descent & Convexity

- 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!



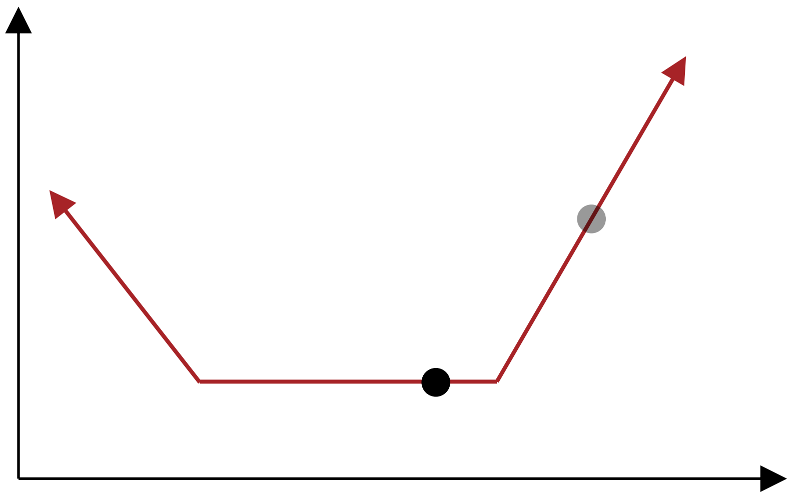
Gradient Descent & Convexity

- 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!



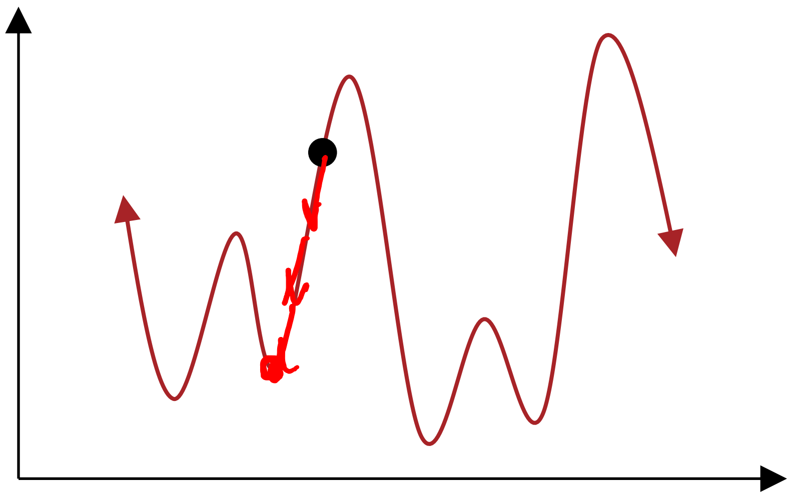
Gradient Descent & Convexity

- 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!



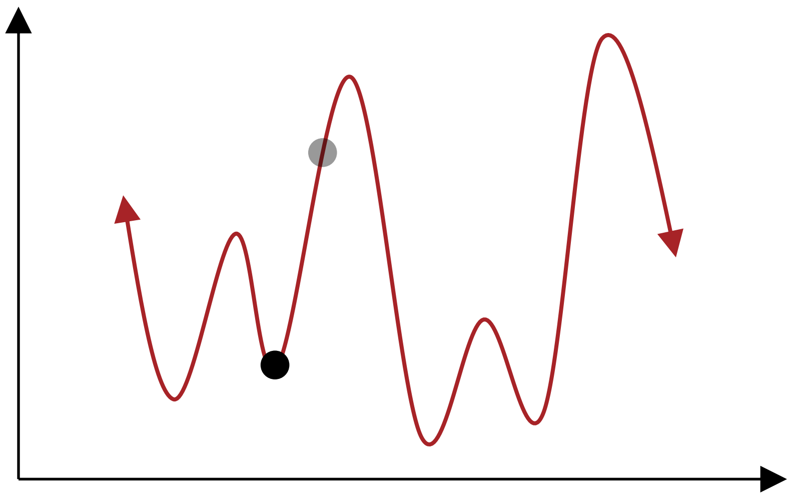
Gradient Descent & Convexity

- 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…



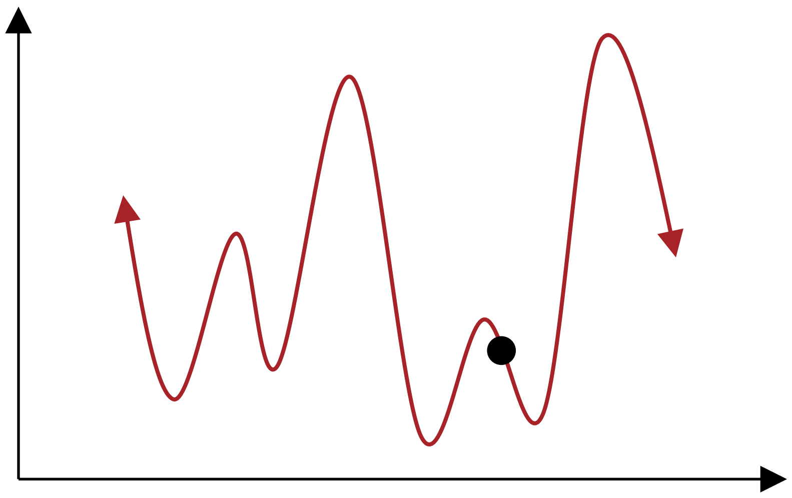
Gradient Descent & Convexity

- 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…



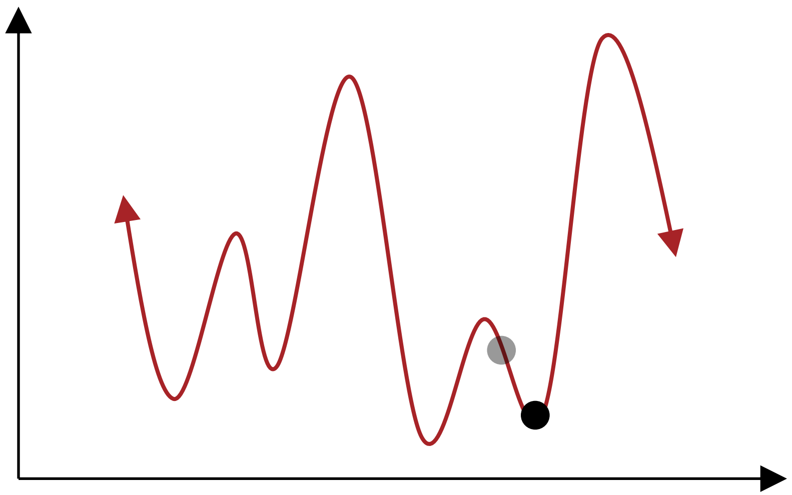
Gradient Descent & Convexity

- 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…



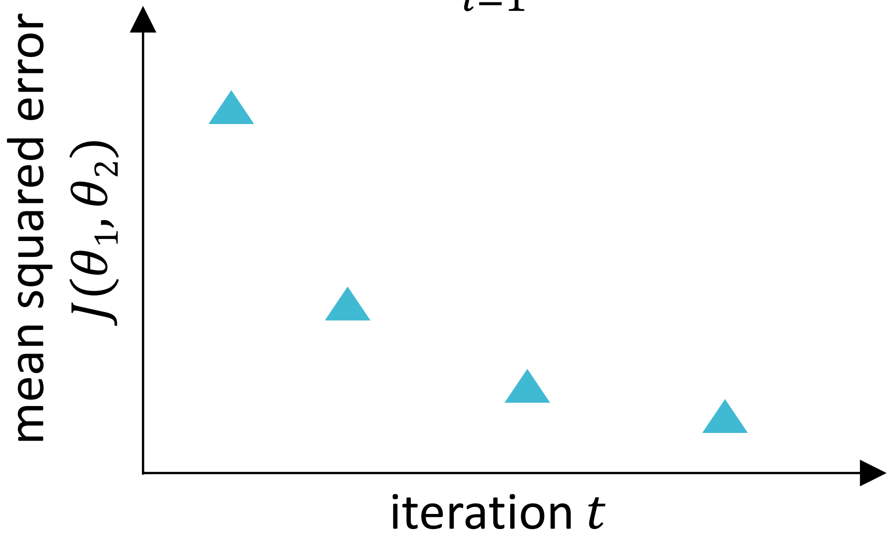
Gradient Descent & Convexity

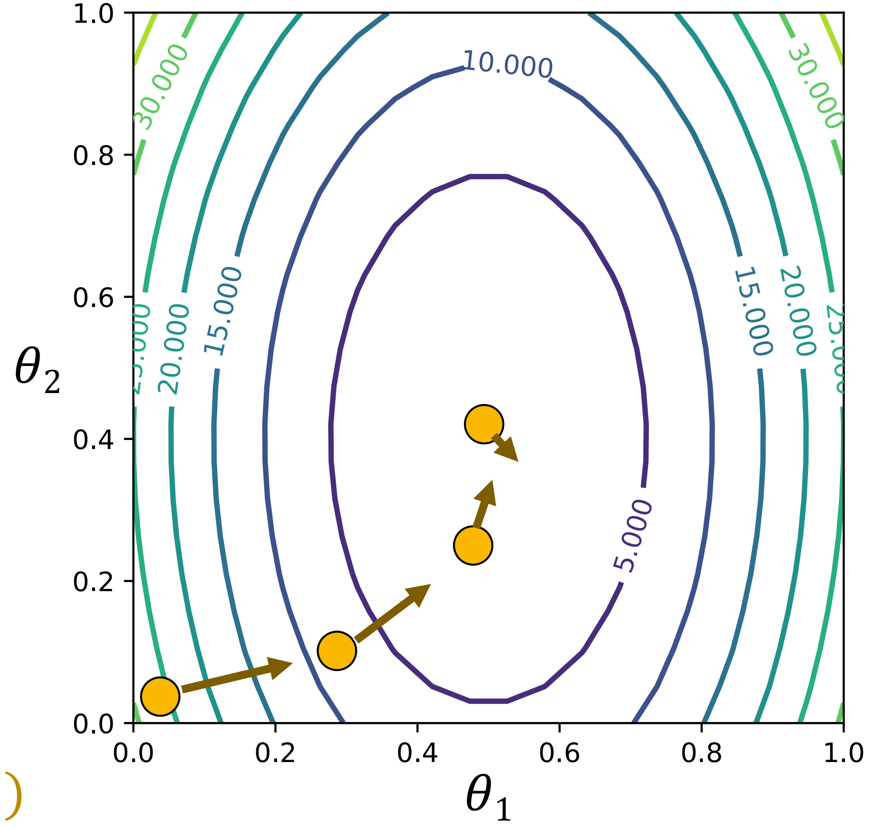
- Gradient descent is a local optimization algorithm – it will converge to a local minimum (if it converges)
  - Not ideal if the objective function is non-convex…

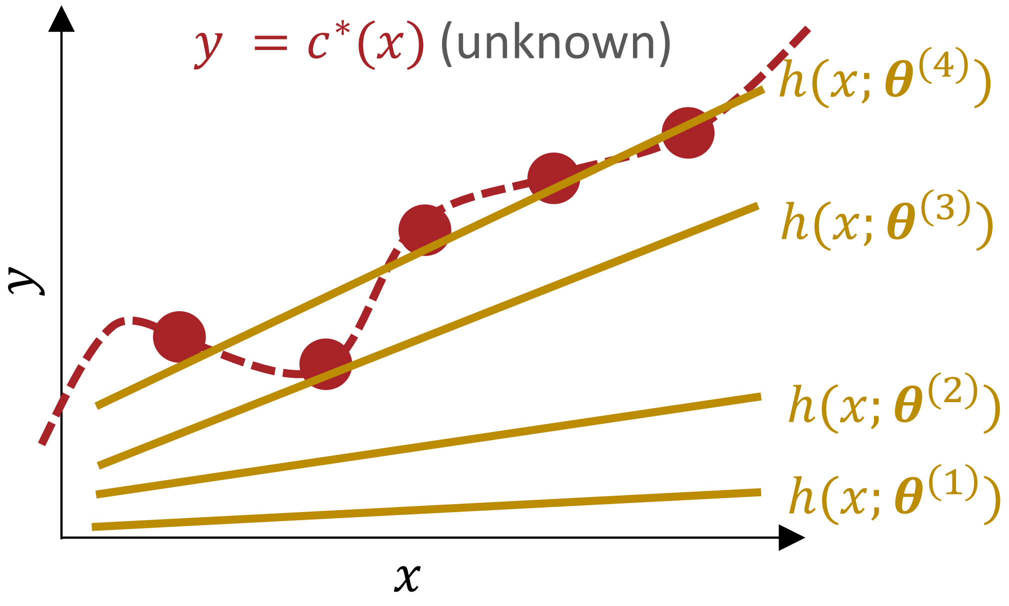


Why Gradient Descent for Linear Regression?

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



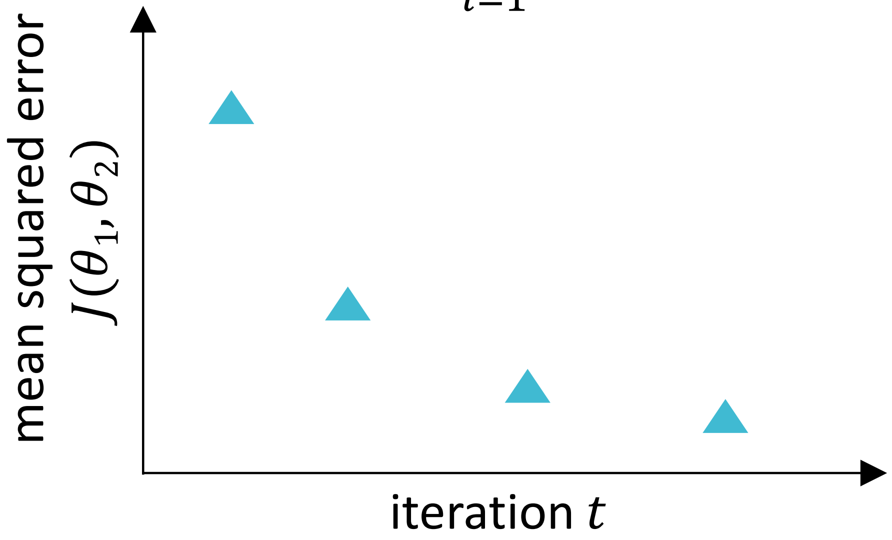


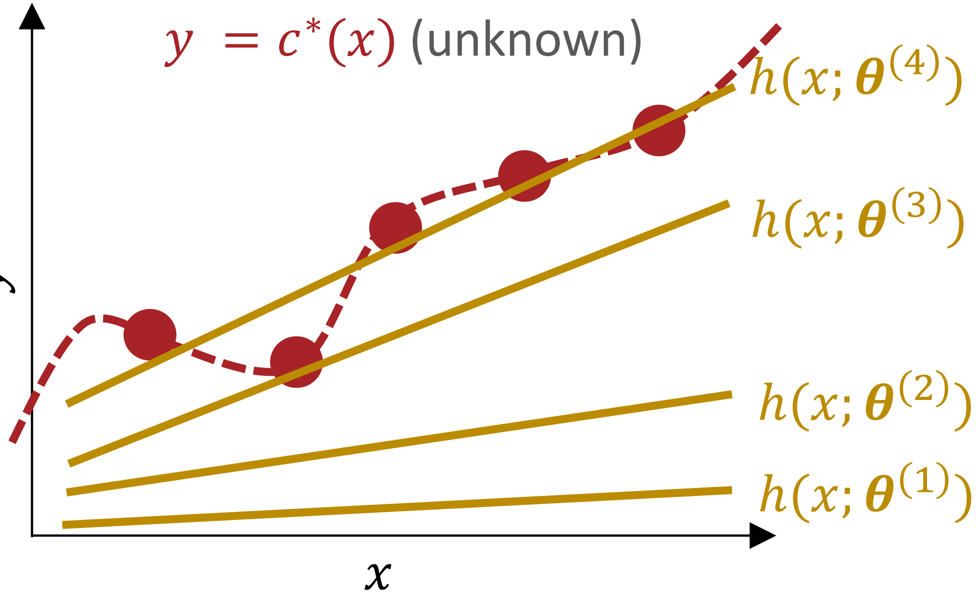


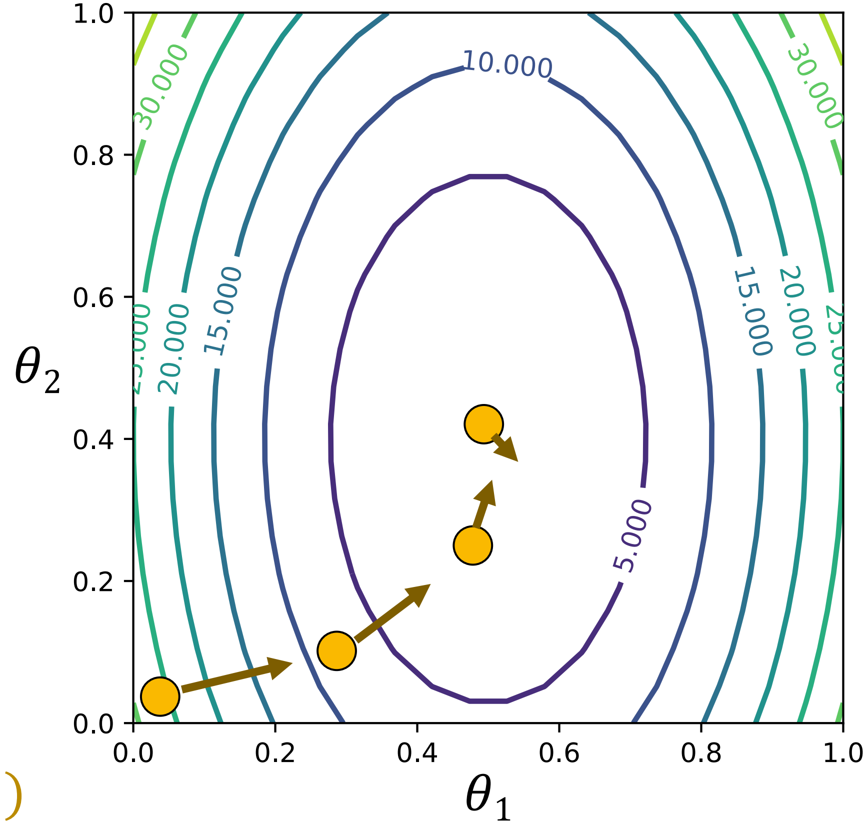
t	θ_1	θ_2	$J(\theta_1, \theta_2)$
1	0.01	0.02	25.2
2	0.30	0.12	8.7
3	0.51	0.30	1.5
4	0.59	0.43	0.2

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

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



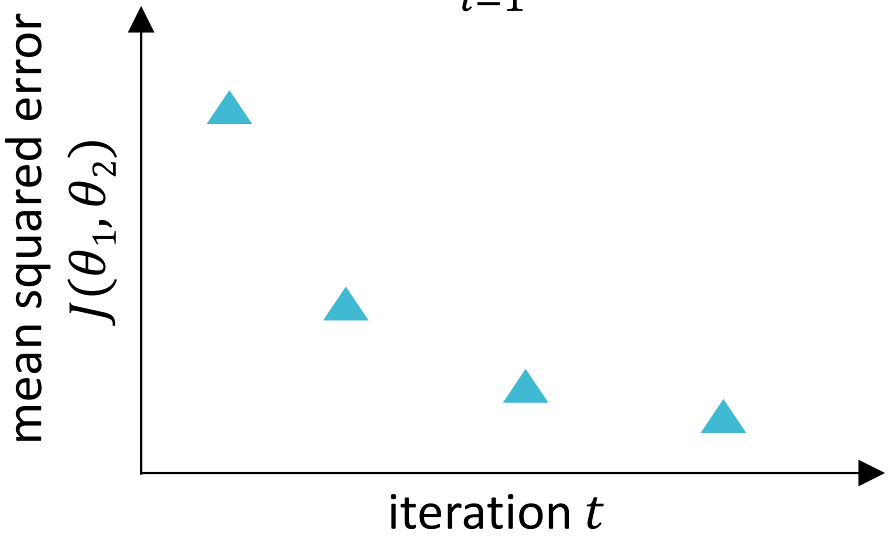


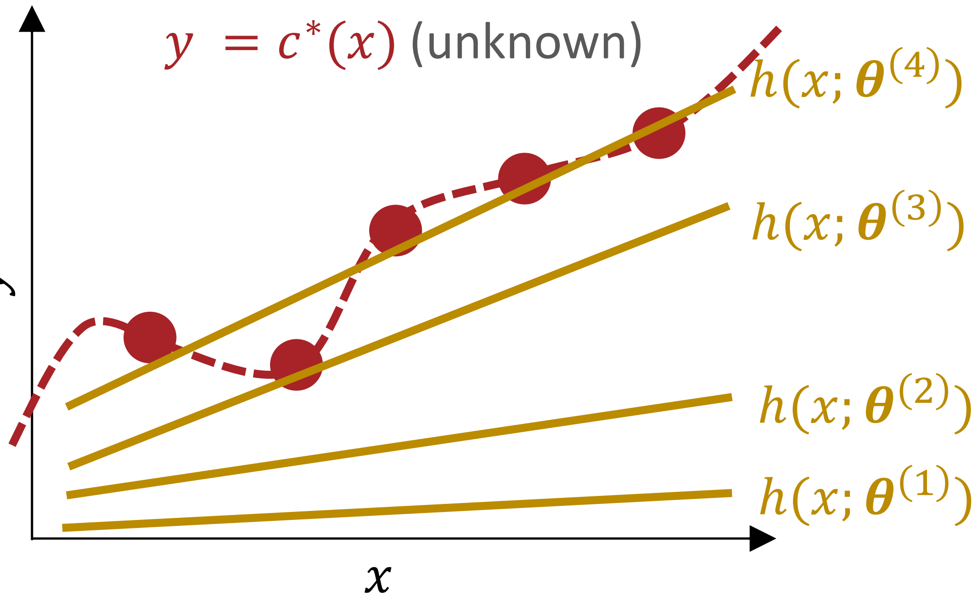


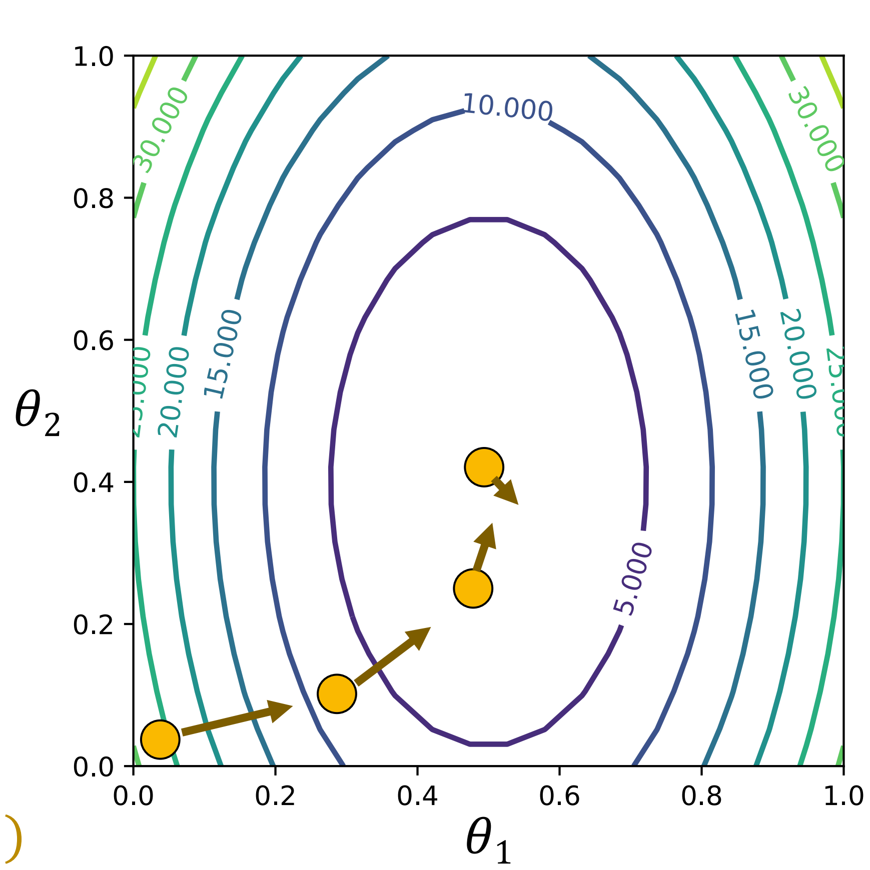
t	θ_1	θ_2	$J(\theta_1, \theta_2)$
1	0.01	0.02	25.2
2	0.30	0.12	8.7
3	0.51	0.30	1.5
4	0.59	0.43	0.2

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

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







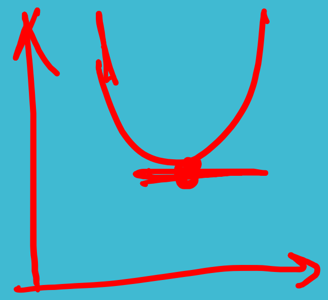
t	θ_1	θ_2	$J(\theta_1, \theta_2)$
1	0.01	0.02	25.2
2	0.30	0.12	8.7
3	0.51	0.30	1.5
4	0.59	0.43	0.2

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 $\mathcal{D} = \{(\boldsymbol{x}^{(n)}, y^{(n)})\}_{n=1}^{N}$
  - $$X = \begin{bmatrix} 1 & \boldsymbol{x}^{(1)^T} \\ 1 & \boldsymbol{x}^{(2)^T} \\ \vdots & \vdots \\ 1 & \boldsymbol{x}^{(N)^T} \end{bmatrix} = \begin{bmatrix} 1 & x_1^{(1)} & \cdots & x_D^{(1)} \\ 1 & x_1^{(2)} & \cdots & x_D^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(N)} & \cdots & x_D^{(N)} \end{bmatrix} \in \mathbb{R}^{N \times D+1}$$
    M

    is the *design matrix*
  - $\boldsymbol{y} = \left[y^{(1)}, \dots, y^{(N)}\right]^T \in \mathbb{R}^N$ is the *target vector*



Minimizing the Mean Squared Error

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

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

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

$\iff \vec{X}^T\vec{X}\vec{\theta} - \vec{X}^T\vec{y} = 0$

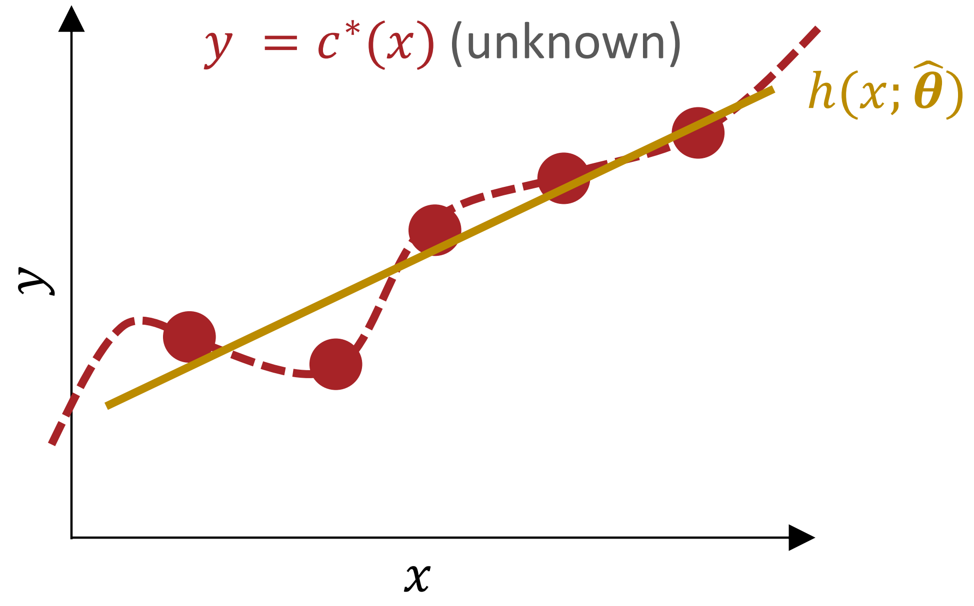
$\iff \vec{X}^T\vec{X}\vec{\theta} = \vec{X}^T\vec{y}$

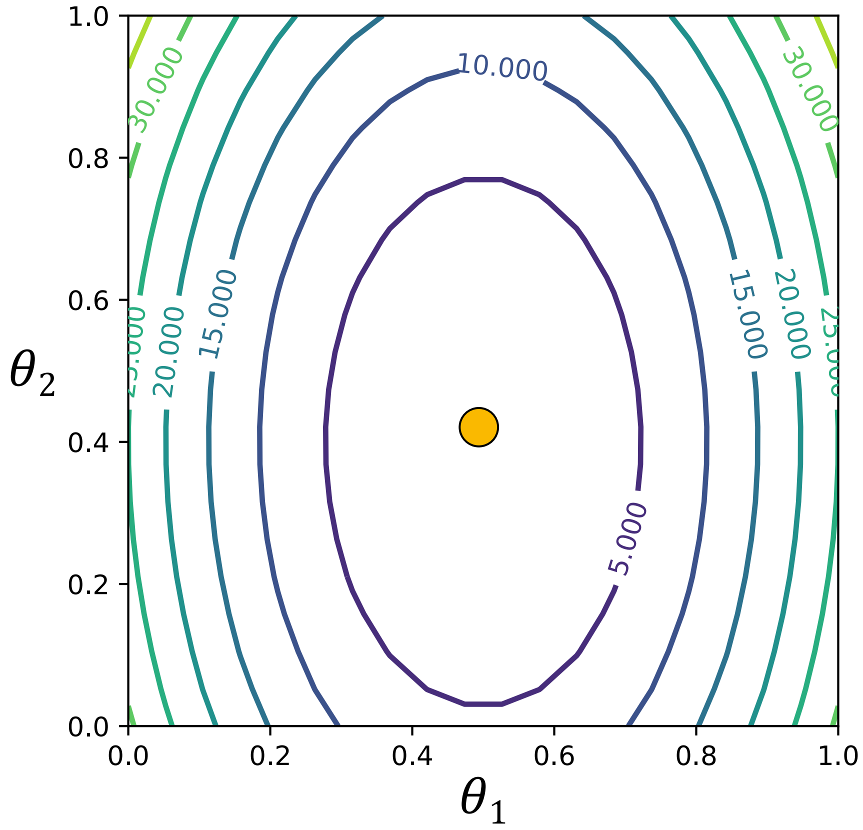
$(X^TX)^{-1}X^TX\vec{\theta} = (X^TX)^{-1}\vec{X}^T\vec{y}$

Hessian of J must be PSD.

Closed Form Optimization

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





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

Closed Form Solution

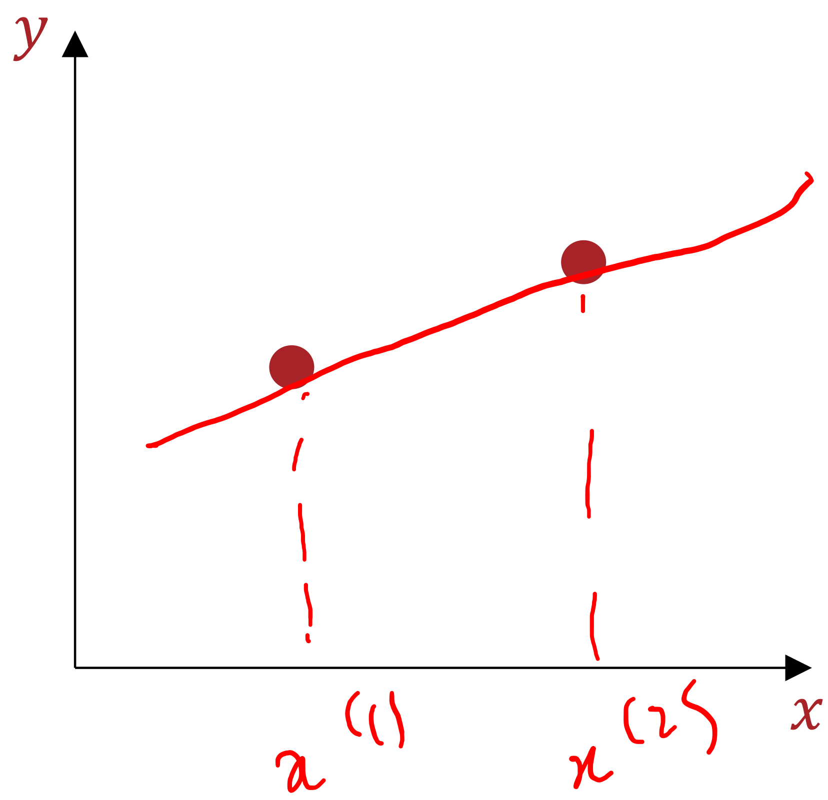
$$\hat{\boldsymbol{\theta}} = (X^T X)^{-1} X^T \boldsymbol{y}$$

1. Is $X^T X$ invertible?

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

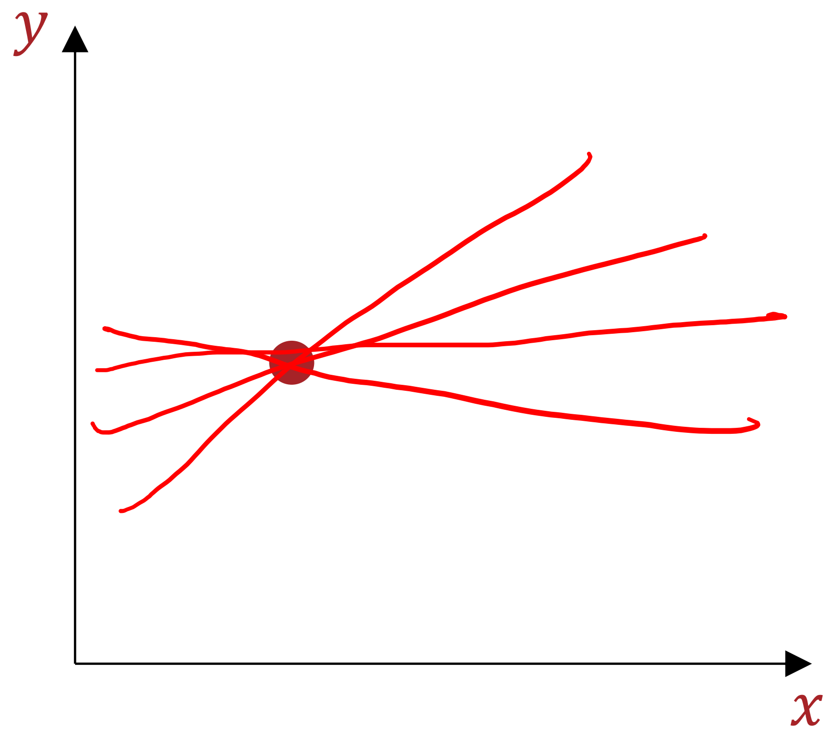
Linear Regression: Uniqueness

- 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?



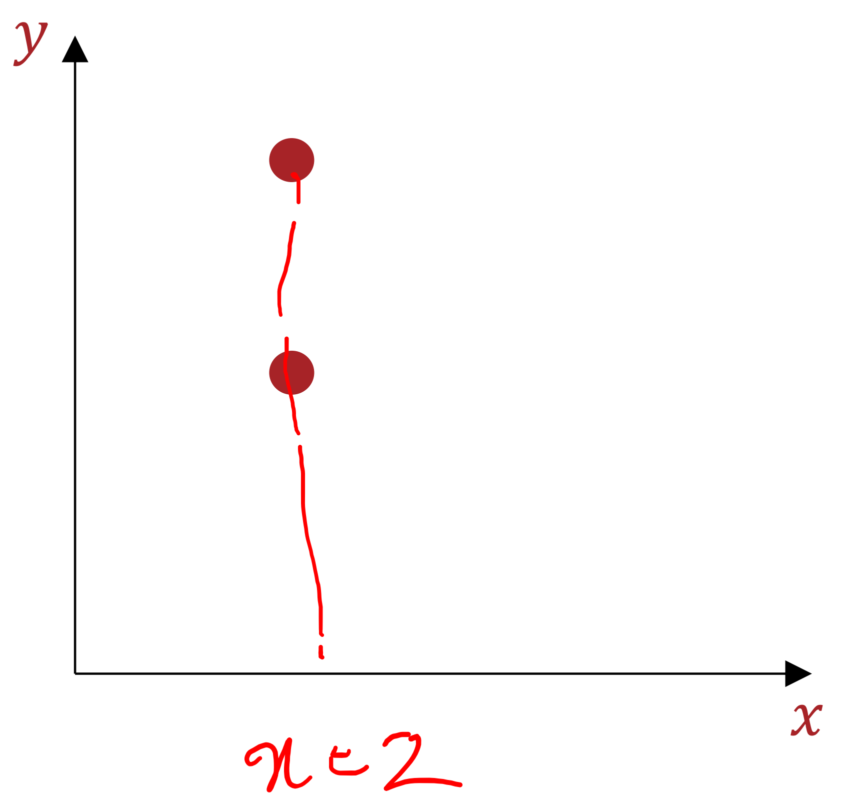
Linear Regression: Uniqueness

- 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?



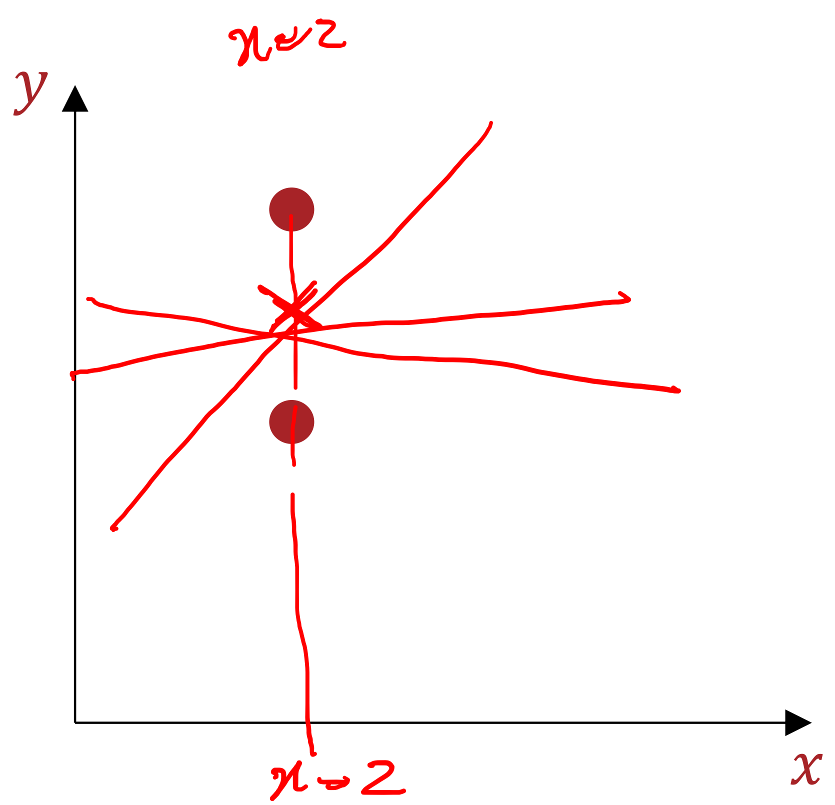
Linear Regression: Uniqueness

- 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

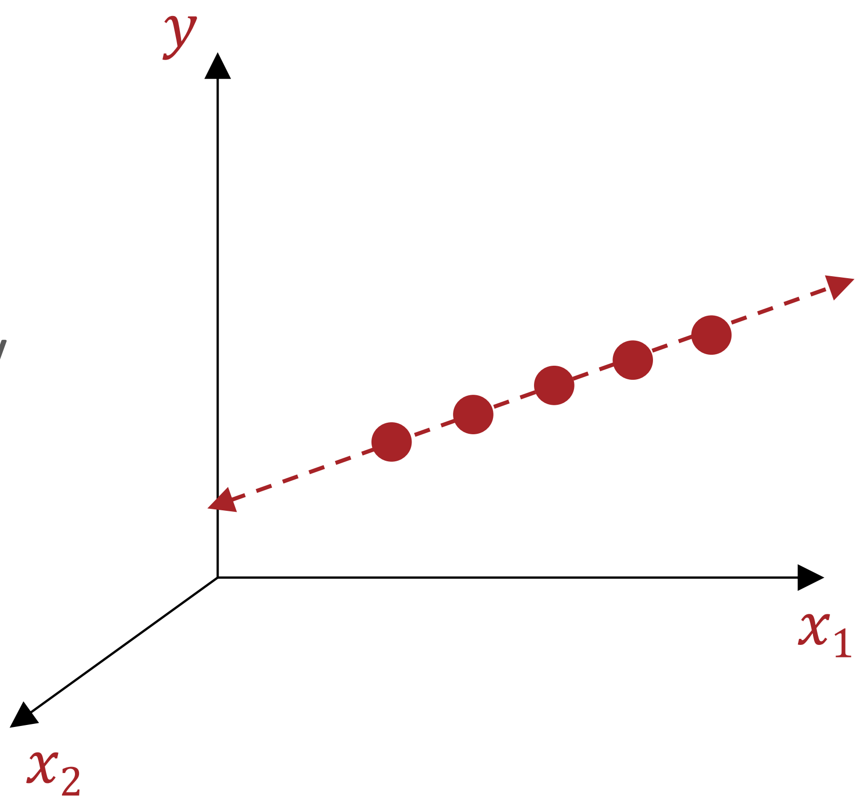
- Consider a 1D linear regression model trained to minimize the mean squared error: how many optimal solutions (i.e., sets of parameters θ) are there for the given dataset?



A. -1 **(TOXIC)** B. 0 C. 1 D. 2 E. ∞

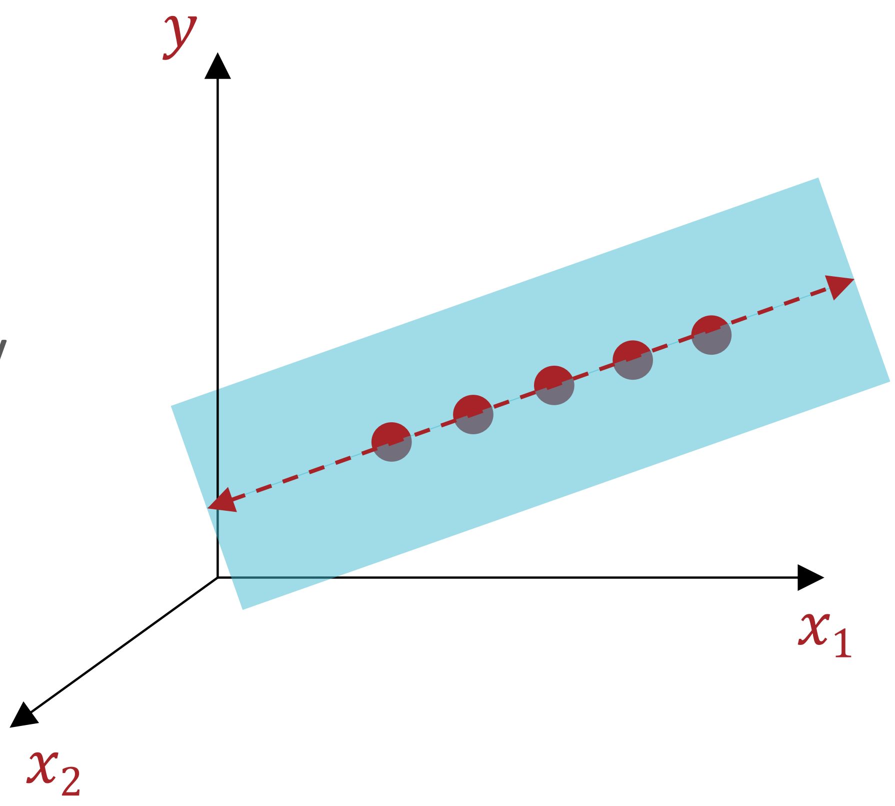
Linear Regression: Uniqueness

- 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?



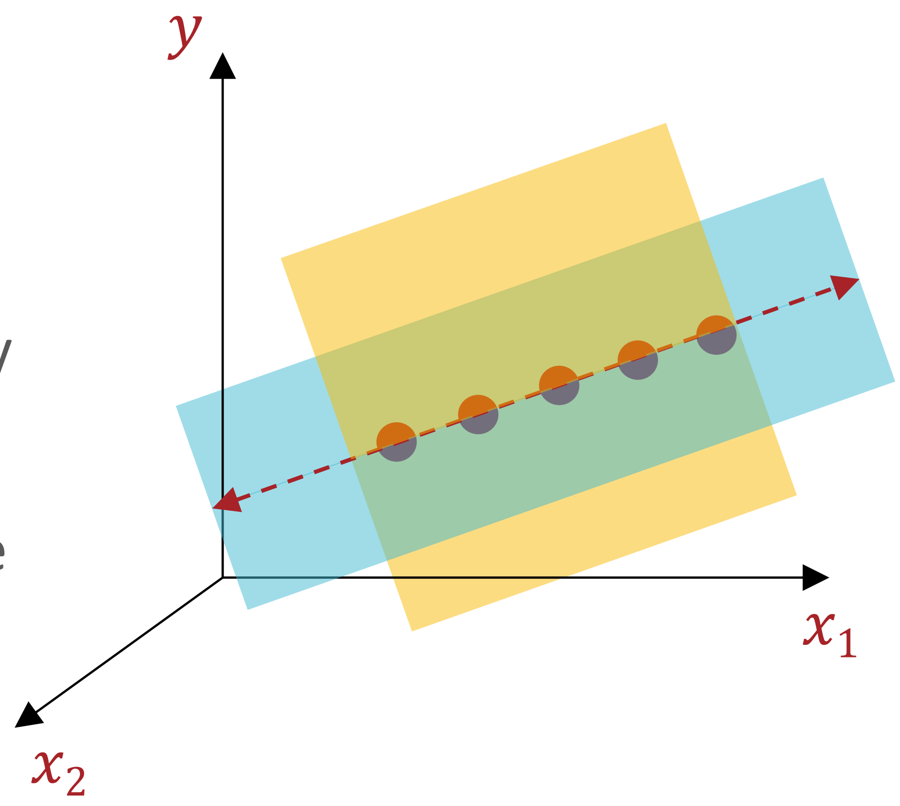
Linear Regression: Uniqueness

- 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?



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 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