https://introml.mit.edu/

6.390 Intro to Machine Learning

Lecture 3: Gradient Descent Methods

Shen Shen

❤️ Feb 14, 2025 ❤️

(11am, Room 10-250)

Outline

- Recap, motivation for gradient descent methods
- Gradient descent algorithm (GD)
  - The gradient vector
  - GD algorithm
  - Gradient decent properties
    - convex functions, local vs global min
- Stochastic gradient descent (SGD)
  - SGD algorithm and setup
  - GD vs SGD comparison

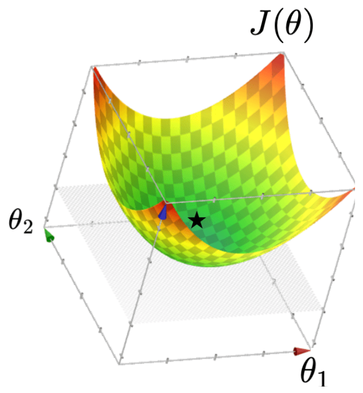
Outline

- Recap, motivation for gradient descent methods
- Gradient descent algorithm (GD)
  - The gradient vector
  - GD algorithm
  - Gradient decent properties
    - convex functions, local vs global min
- Stochastic gradient descent (SGD)
  - SGD algorithm and setup
  - GD vs SGD comparison

Recall

1. Typically, X is full column rank

- $J(\theta)$ looks like a bowl
- $\theta^* = \left(X^\top X\right)^{-1} X^\top Y$
- θ^* gives the unique optimal hyperplane



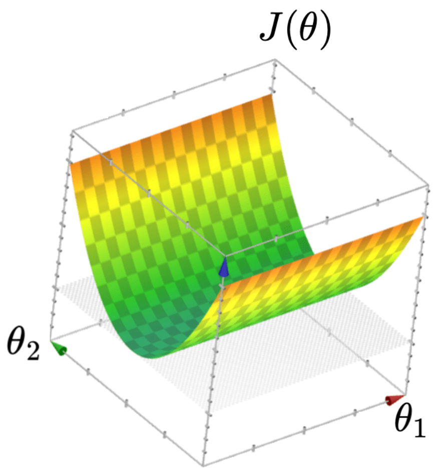
🥰

2. When X is not full column rank

a. either when $n<d$, or

b. columns (features) in X have linear dependency

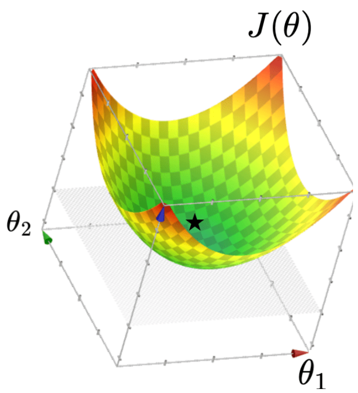
- $J(\theta)$ looks like a half-pipe
- This 👈 formula is not well-defined
- Infinitely many optimal hyperplanes



🥺

1. Typically, X is full column rank

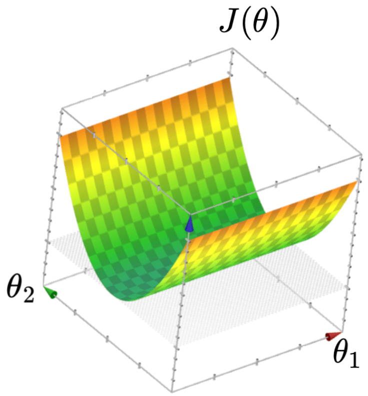
- $J(\theta)$ looks like a bowl
- $\theta^* = \left(X^\top X\right)^{-1} X^\top Y$
- θ^* gives the unique optimal hyperplane



- θ^* can be costly to compute (lab2, Q2.7)

2. When X is not full column rank

- $J(\theta)$ looks like a half-pipe
- This 👈 formula is not well-defined
- Infinitely many optimal hyperplanes



- No way yet to obtain an optimal parameter

Want a more efficient and general method => gradient descent methods

Outline

- Recap, motivation for gradient descent methods
- Gradient descent algorithm (GD)
  - The gradient vector
  - GD algorithm
  - Gradient decent properties
    - convex functions, local vs global min
- Stochastic gradient descent (SGD)
  - SGD algorithm and setup
  - GD vs SGD comparison

For $f : \mathbb{R}^m \to \mathbb{R}$, its ***gradient*** $\nabla f : \mathbb{R}^m \to \mathbb{R}^m$ is defined at the point $p = (x_1, \ldots, x_m)$ as:

$$\nabla f(p) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(p) \\ \vdots \\ \frac{\partial f}{\partial x_m}(p) \end{bmatrix}$$

1. The gradient generalizes the concept of a derivative to multiple dimensions.
2. By construction, the gradient's dimensionality always matches the function input.

Sometimes the gradient is undefined or ill-behaved, but today it is well-behaved unless stated otherwise.

3. The gradient can be symbolic or numerical.

$$\nabla f(p) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(p) \\ \vdots \\ \frac{\partial f}{\partial x_m}(p) \end{bmatrix}$$

example: $f(x, y, z) = x^2 + y^3 + z$

its *symbolic* gradient:

$$\nabla f(x, y, z) = \begin{bmatrix} 2x \\ 3y^2 \\ 1 \end{bmatrix}$$

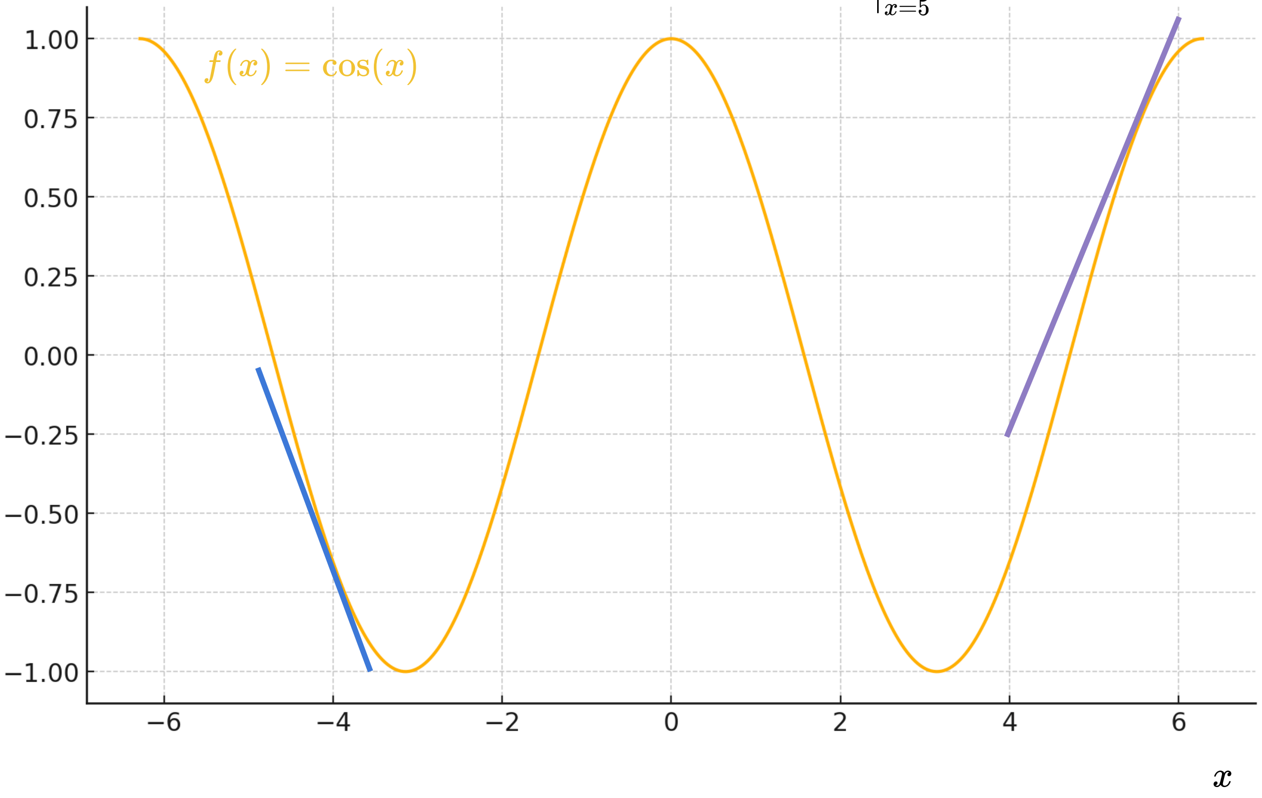
evaluating the symbolic gradient at a point gives a *numerical* gradient:

$$\nabla f(3, 2, 1) = \nabla f(x, y, z)\Big|_{(x,y,z)=(3,2,1)} = \begin{bmatrix} 6 \\ 12 \\ 1 \end{bmatrix}$$

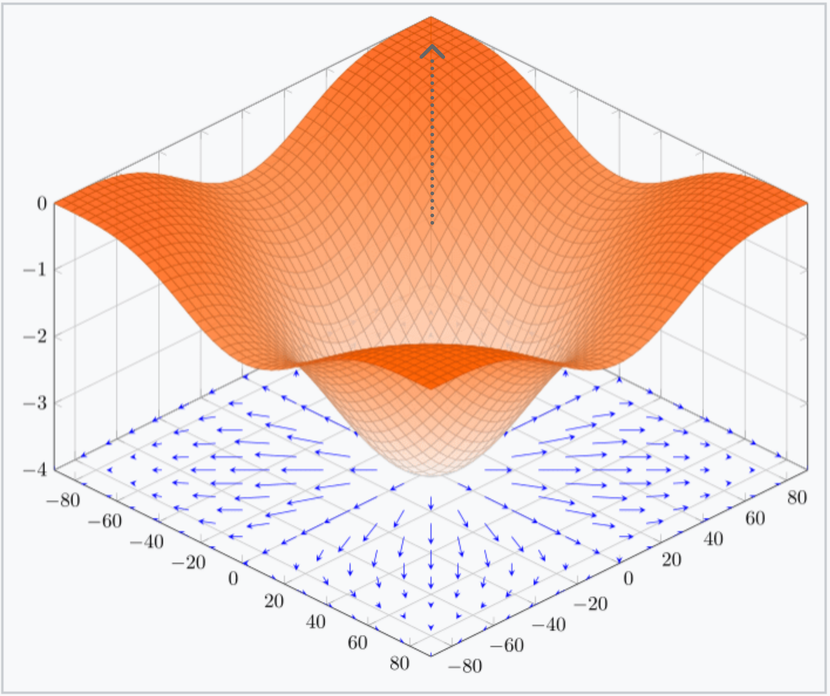
just like a derivative can be a function or a number.

4. The gradient points in the direction of the (steepest) *increase* in the function value.

$$\frac{d}{dx}\cos(x)\Big|_{x=5} = -\sin(5) \approx 0.9589$$



$$\frac{d}{dx}\cos(x)\Big|_{x=-4} = -\sin(-4) \approx -0.7568$$



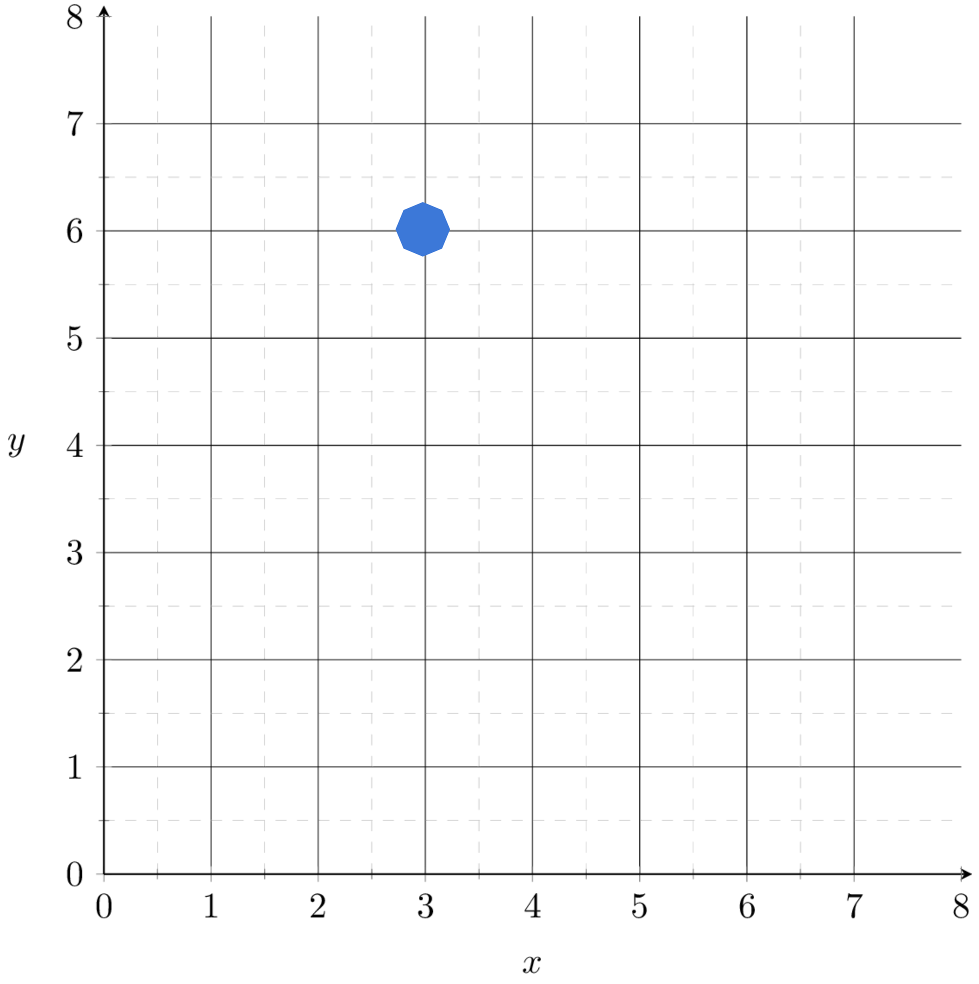
The gradient of the function $f(x,y) = -(\cos^2 x + \cos^2 y)^2$

Outline

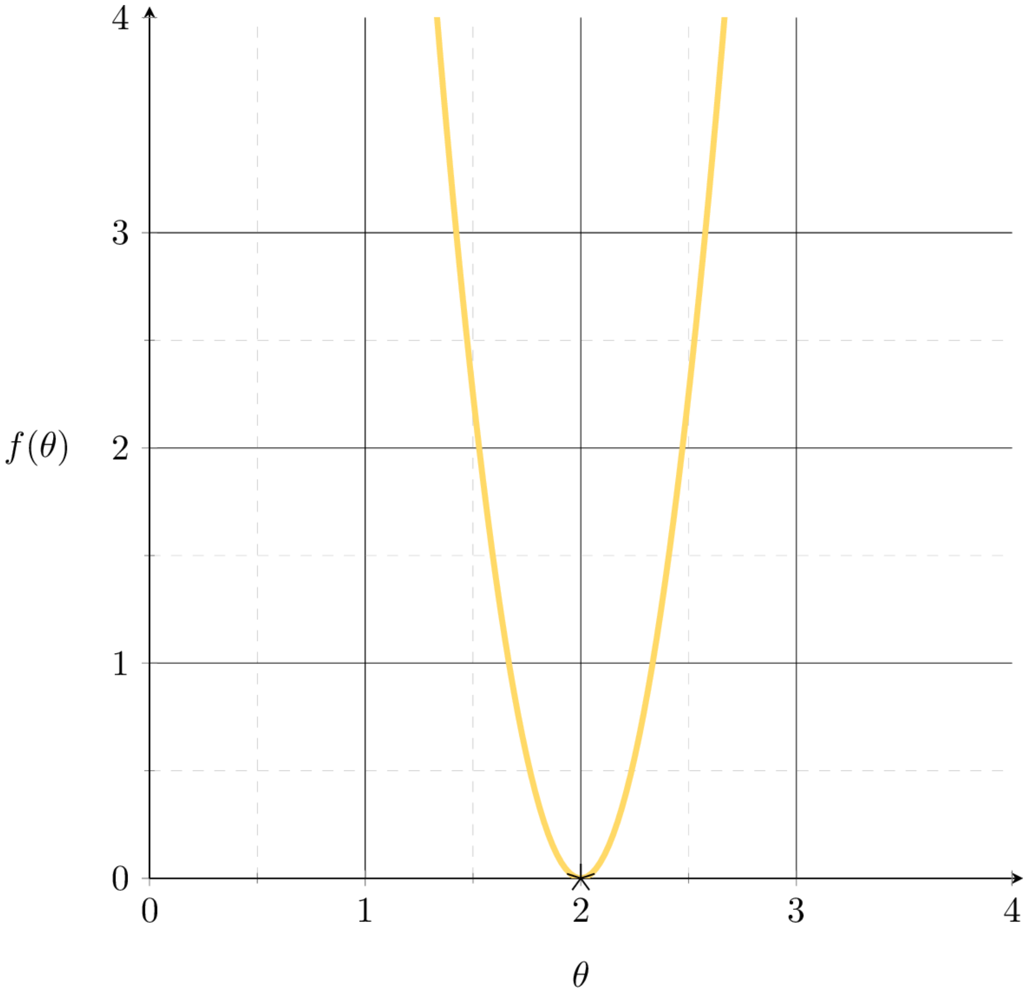
- Recap, motivation for gradient descent methods
- Gradient descent algorithm (GD)
  - The gradient vector
  - GD algorithm
  - Gradient decent properties
    - convex functions, local vs global min
- Stochastic gradient descent (SGD)
  - SGD algorithm and setup
  - GD vs SGD comparison

A single training data point

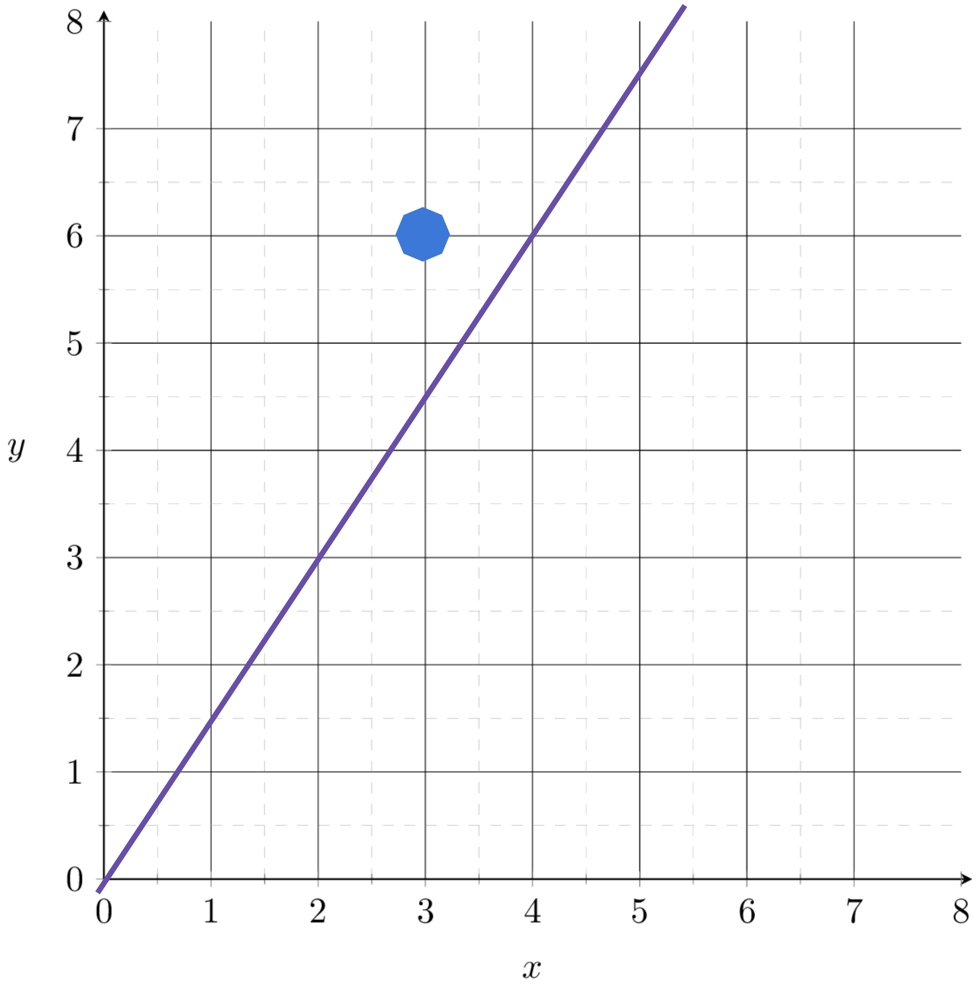
$(x, y) = (3, 6)$



Want to fit a line (without offset) to minimize the MSE: $f(\theta) = (3\theta - 6)^2$

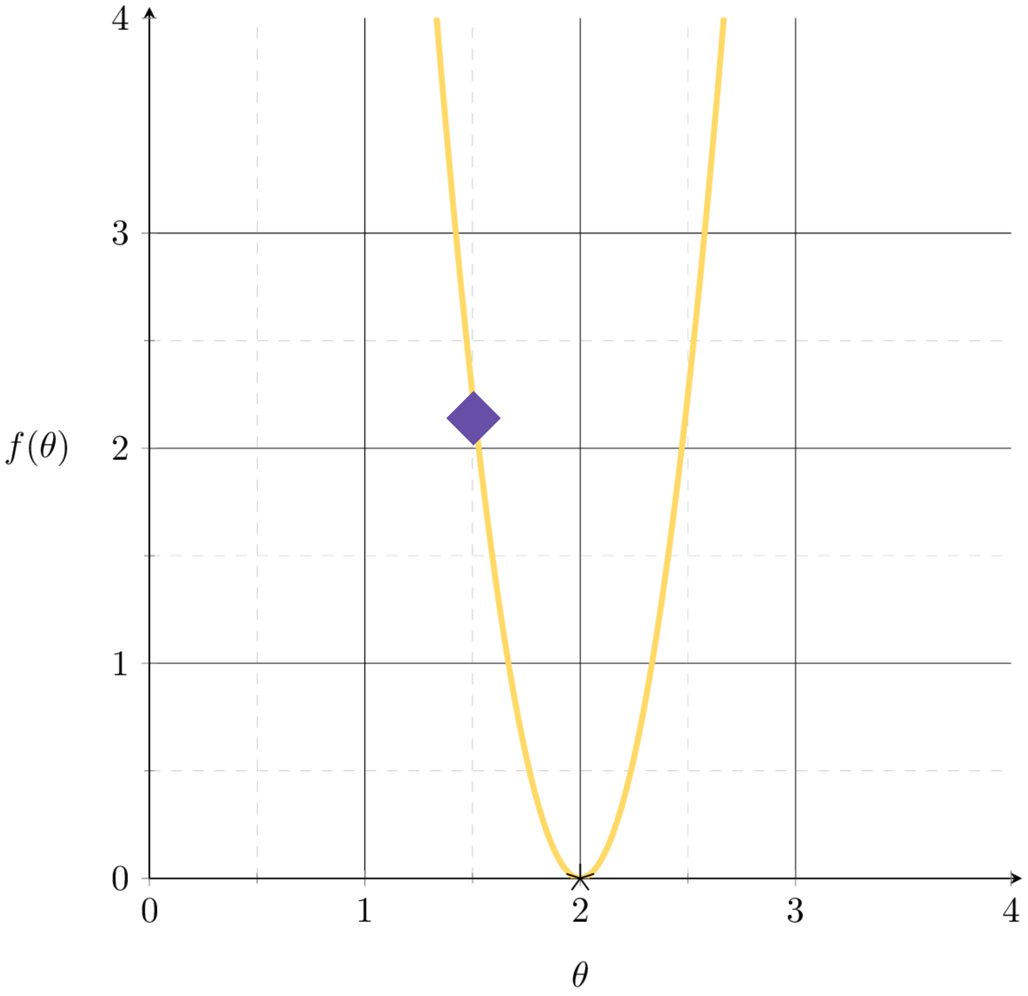


Suppose we fit a line $y = 1.5x$

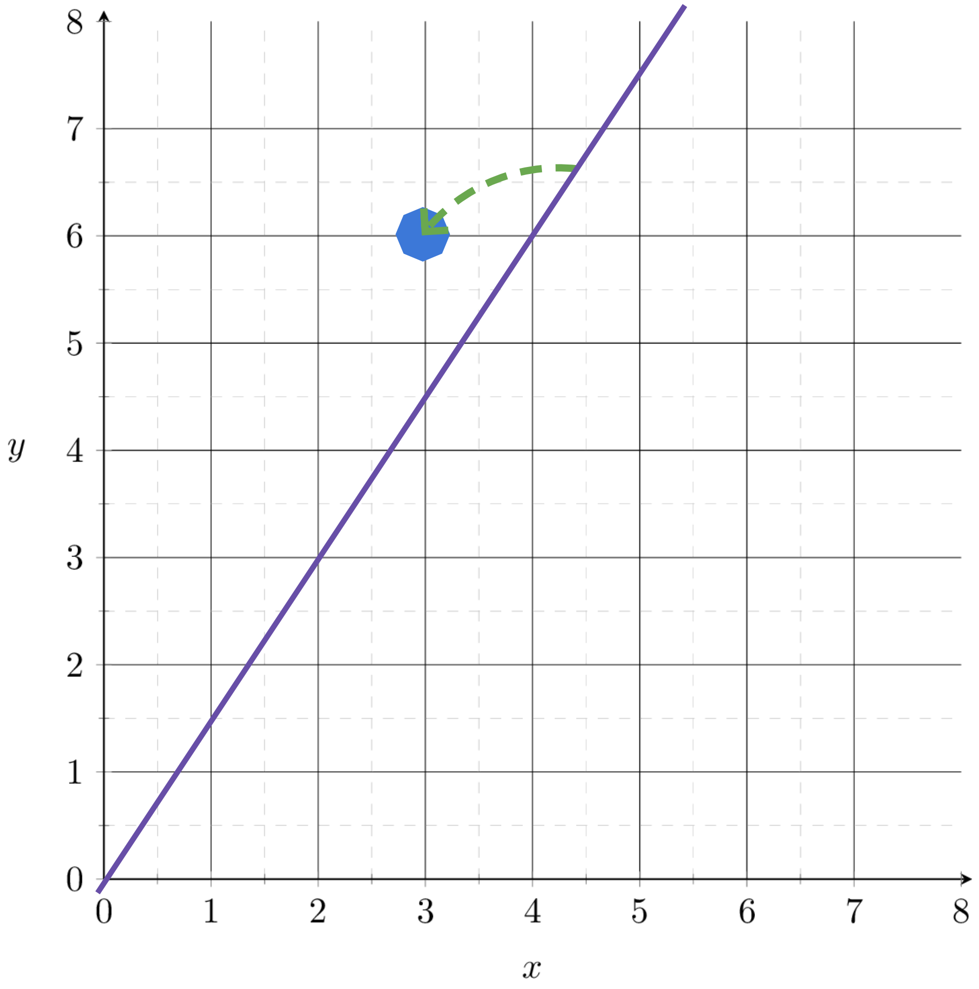


MSE could get better.

How to formalize this?

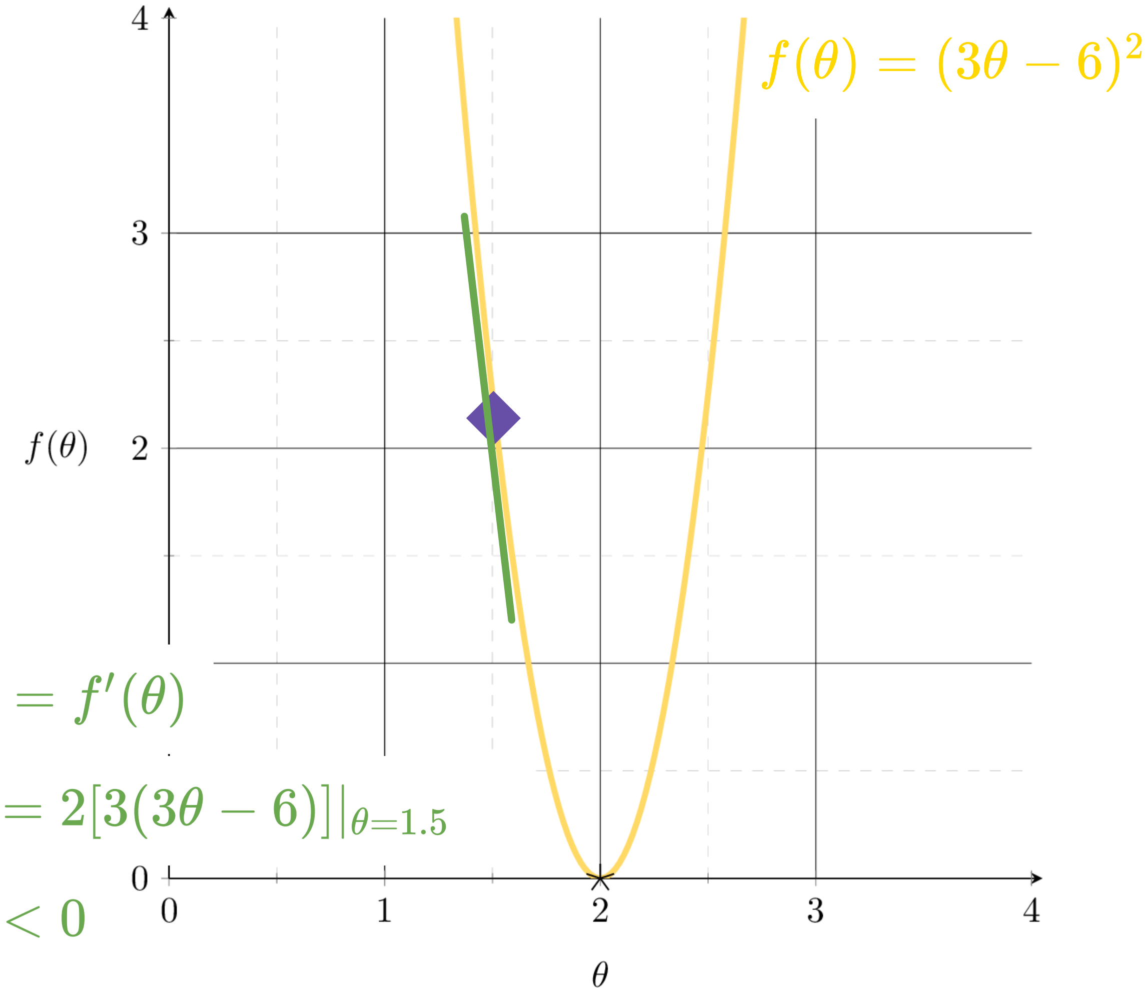


Suppose we fit a line $y = 1.5x$



MSE could get better. How to?

Leveraging the gradient.

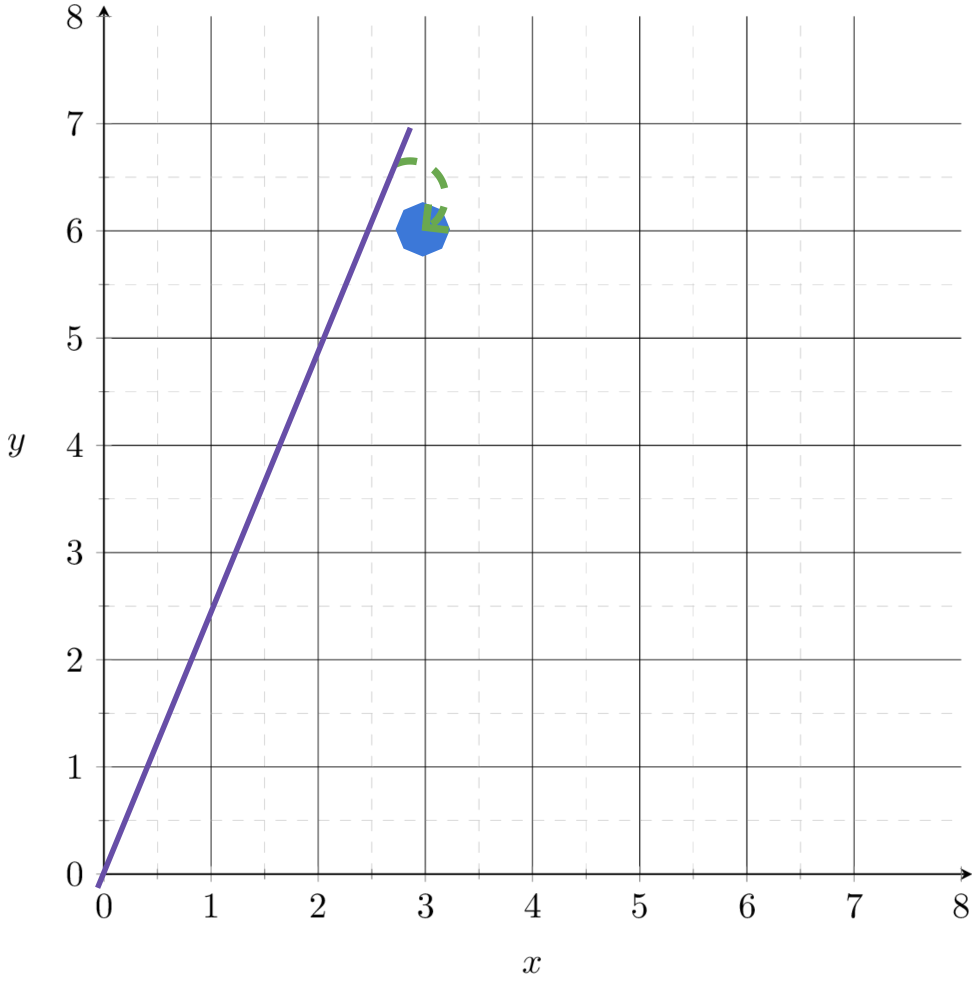


$$\nabla_\theta f = f'(\theta)$$

$$= 2[3(3\theta - 6)]|_{\theta=1.5}$$

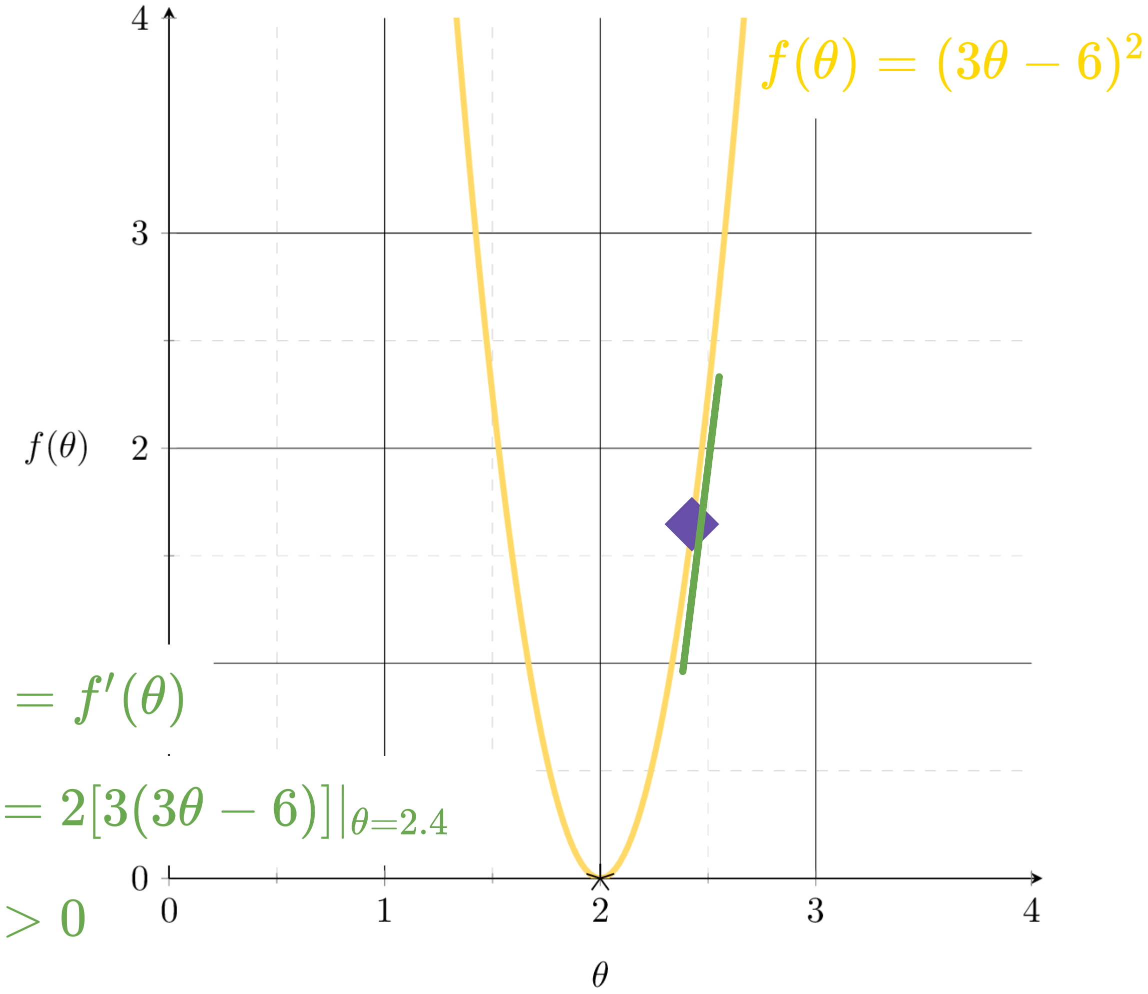
$$< 0$$

Suppose we fit a line $y = 2.4x$



MSE could get better. How to?

Leveraging the gradient.



$$\nabla_\theta f = f'(\theta)$$
$$= 2[3(3\theta - 6)]|_{\theta=2.4}$$
$$> 0$$

hyperparameters

initial guess of parameters

learning rate, aka, step size

precision

Gradient-Descent($\Theta_{\text{init}}, \eta, f, \nabla_\Theta f, \epsilon$)

  Initialize $\Theta^{(0)} = \Theta_{\text{init}}$

  Initialize t = 0

  repeat

    t = t + 1

    $\Theta^{(t)} = \Theta^{(t-1)} - \eta \nabla_\Theta f(\Theta^{(t-1)})$

  until $\left| f(\Theta^{(t)}) - f(\Theta^{(t-1)}) \right| < \epsilon$

  Return $\Theta^{(t)}$

Gradient-Descent($\Theta_{\text{init}}, \eta, f, \nabla_\Theta f, \epsilon$)

  Initialize $\Theta^{(0)} = \Theta_{\text{init}}$

  Initialize t = 0

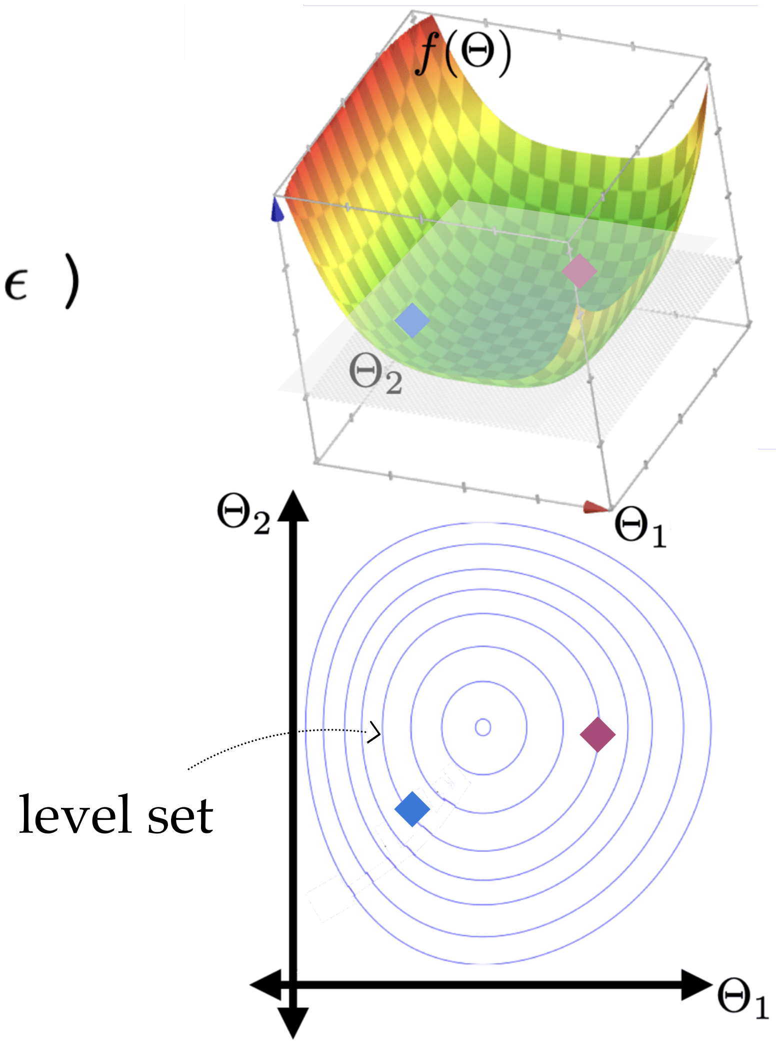
  repeat

    t = t + 1

    $\Theta^{(t)} = \Theta^{(t-1)} - \eta \nabla_\Theta f(\Theta^{(t-1)})$

  until $\left| f(\Theta^{(t)}) - f(\Theta^{(t-1)}) \right| < \epsilon$

  Return $\Theta^{(t)}$



Gradient-Descent($\Theta_{\text{init}}, \eta, f, \nabla_\Theta f, \epsilon$)

  Initialize $\Theta^{(0)} = \Theta_{\text{init}}$

  Initialize t = 0

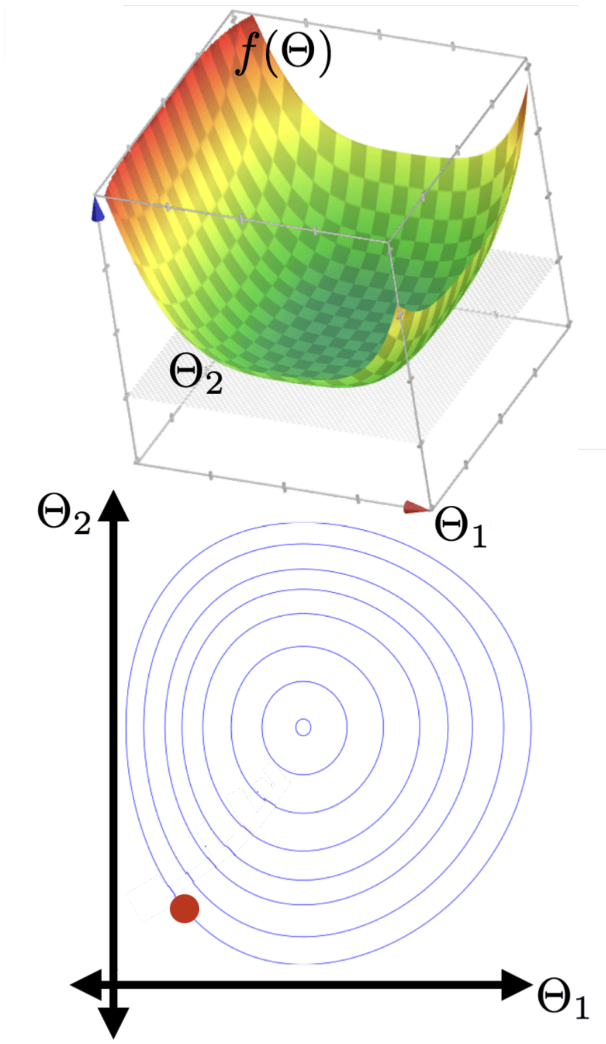
  repeat

    t = t + 1

    $\Theta^{(t)} = \Theta^{(t-1)} - \eta \nabla_\Theta f(\Theta^{(t-1)})$

  until $\left| f(\Theta^{(t)}) - f(\Theta^{(t-1)}) \right| < \epsilon$

  Return $\Theta^{(t)}$



Gradient-Descent($\Theta_{\text{init}}, \eta, f, \nabla_\Theta f, \epsilon$)

  Initialize $\Theta^{(0)} = \Theta_{\text{init}}$

  Initialize t = 0

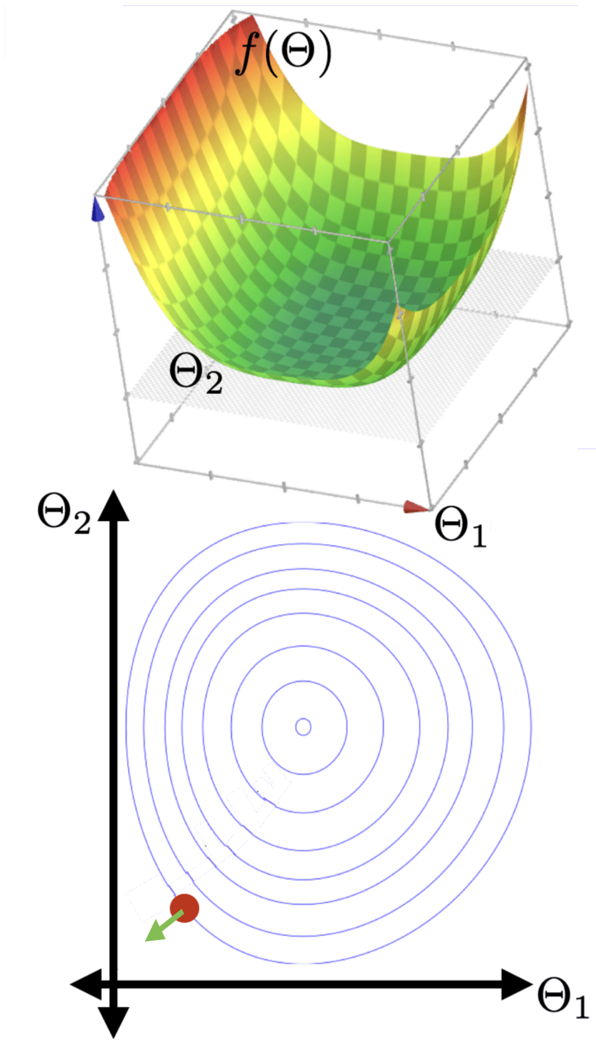
  repeat

    t = t + 1

    $\Theta^{(t)} = \Theta^{(t-1)} - \eta \nabla_\Theta f(\Theta^{(t-1)})$

  until $\left| f(\Theta^{(t)}) - f(\Theta^{(t-1)}) \right| < \epsilon$

  Return $\Theta^{(t)}$



Gradient-Descent($\Theta_{\text{init}}, \eta, f, \nabla_\Theta f, \epsilon$)

  Initialize $\Theta^{(0)} = \Theta_{\text{init}}$

  Initialize t = 0

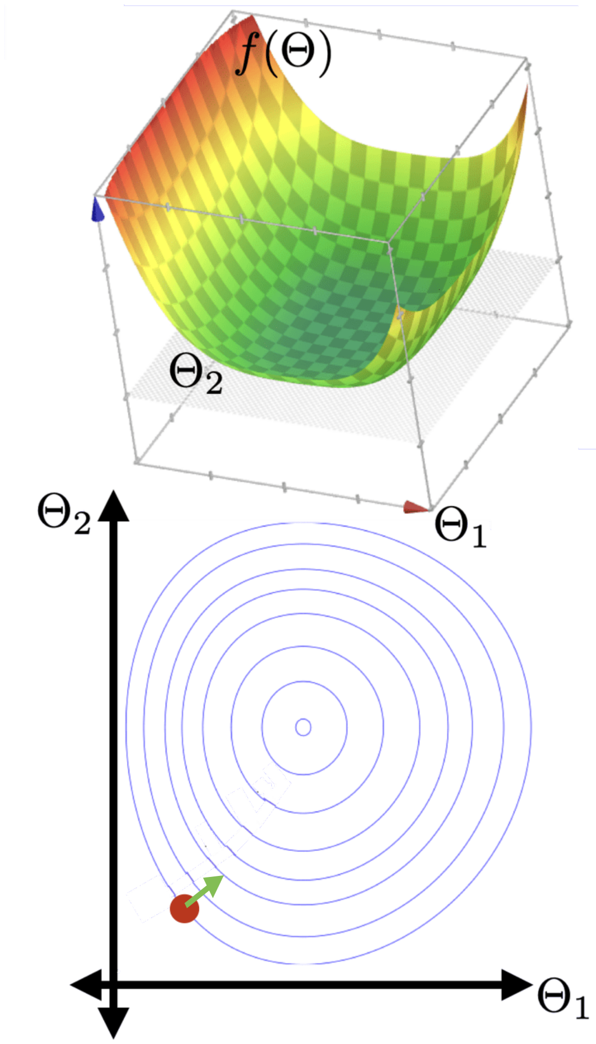
  repeat

    t = t + 1

    $\Theta^{(t)} = \Theta^{(t-1)} - \eta \nabla_\Theta f(\Theta^{(t-1)})$

  until $\left| f(\Theta^{(t)}) - f(\Theta^{(t-1)}) \right| < \epsilon$

  Return $\Theta^{(t)}$



Gradient-Descent($\Theta_{\text{init}}, \eta, f, \nabla_{\Theta} f, \epsilon$)

  Initialize $\Theta^{(0)} = \Theta_{\text{init}}$

  Initialize t = 0

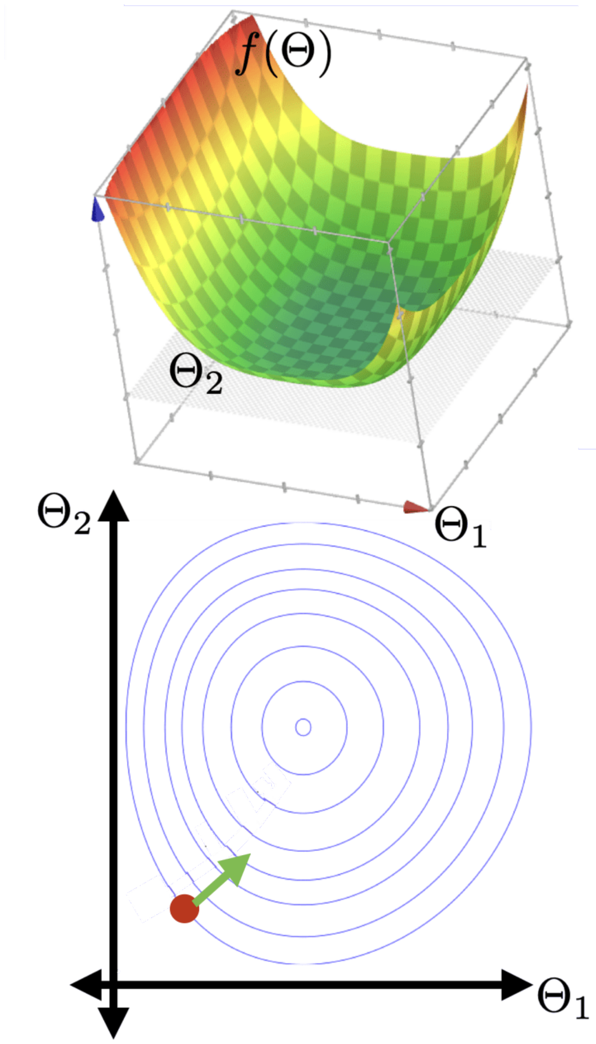
  repeat

    t = t + 1

    $\Theta^{(t)} = \Theta^{(t-1)} - \eta \nabla_{\Theta} f(\Theta^{(t-1)})$

  until $\left| f(\Theta^{(t)}) - f(\Theta^{(t-1)}) \right| < \epsilon$

  Return $\Theta^{(t)}$



Gradient-Descent($\Theta_{\text{init}}, \eta, f, \nabla_\Theta f, \epsilon$)

  Initialize $\Theta^{(0)} = \Theta_{\text{init}}$

  Initialize t = 0

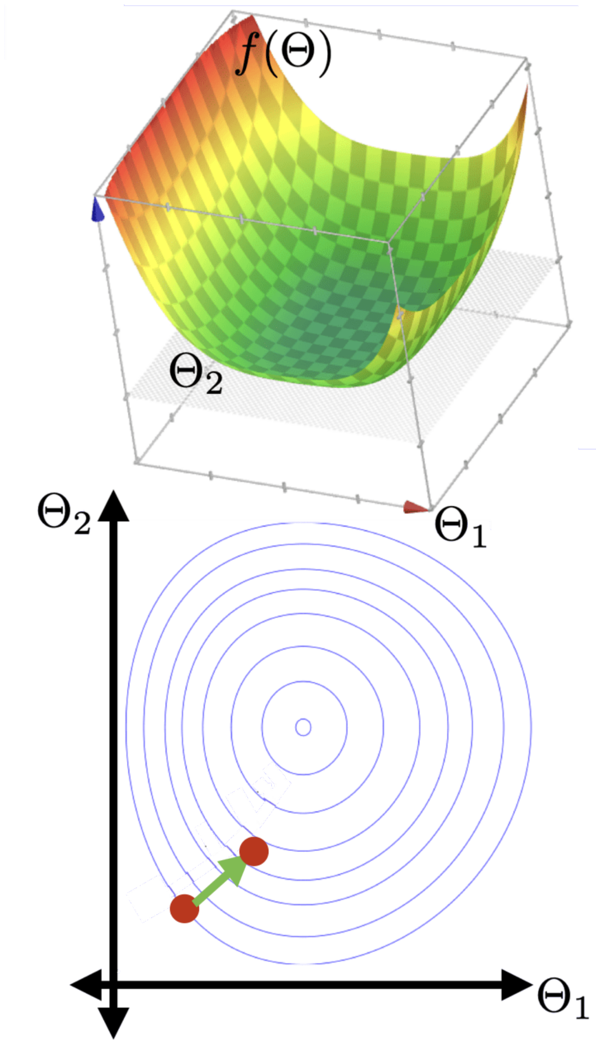
  repeat

    t = t + 1

    $\Theta^{(t)} = \Theta^{(t-1)} - \eta \nabla_\Theta f(\Theta^{(t-1)})$

  until $\left| f(\Theta^{(t)}) - f(\Theta^{(t-1)}) \right| < \epsilon$

  Return $\Theta^{(t)}$



Gradient-Descent($\Theta_{\text{init}}, \eta, f, \nabla_\Theta f, \epsilon$)

  Initialize $\Theta^{(0)} = \Theta_{\text{init}}$

  Initialize t = 0

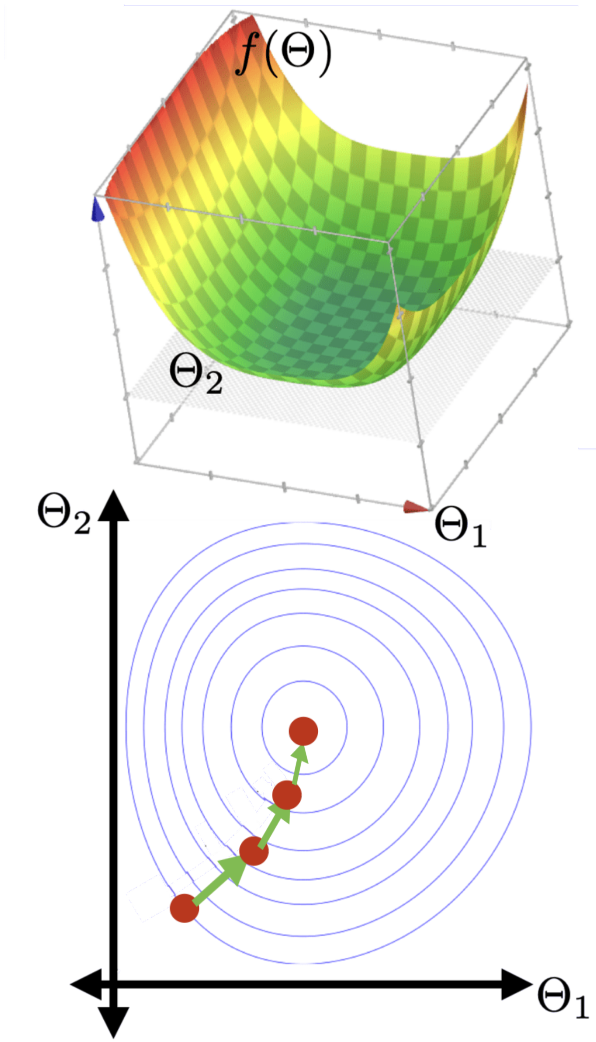
  repeat

    t = t + 1

    $\Theta^{(t)} = \Theta^{(t-1)} - \eta \nabla_\Theta f(\Theta^{(t-1)})$

  until $\left| f(\Theta^{(t)}) - f(\Theta^{(t-1)}) \right| < \epsilon$

  Return $\Theta^{(t)}$



```
Gradient-Descent( $\Theta_{\text{init}}, \eta, f, \nabla_\Theta f, \epsilon$ )
    Initialize $\Theta^{(0)} = \Theta_{\text{init}}$
    Initialize t = 0
    repeat
        t = t + 1
        $\Theta^{(t)} = \Theta^{(t-1)} - \eta \nabla_\Theta f(\Theta^{(t-1)})$
    until $\left| f(\Theta^{(t)}) - f(\Theta^{(t-1)}) \right| < \epsilon$
    Return $\Theta^{(t)}$
```

$$\textbf{Gradient-Descent}(\Theta_{\text{init}}, \eta, f, \nabla_\Theta f, \epsilon)$$

$$\text{Initialize } \Theta^{(0)} = \Theta_{\text{init}}$$

$$\text{Initialize } t = 0$$

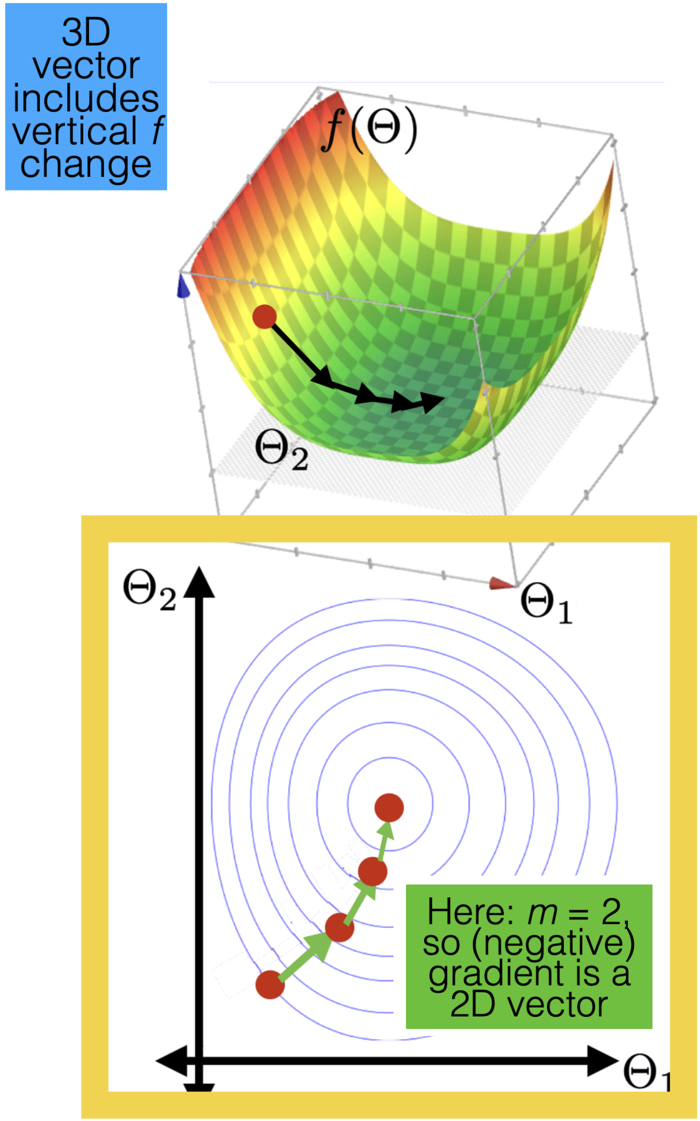
repeat

$$t = t + 1$$

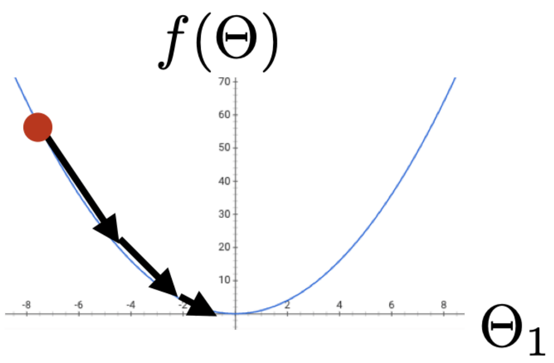
$$\Theta^{(t)} = \Theta^{(t-1)} - \eta \nabla_\Theta f(\Theta^{(t-1)})$$

until $\left| f(\Theta^{(t)}) - f(\Theta^{(t-1)}) \right| < \epsilon$

Return $\Theta^{(t)}$

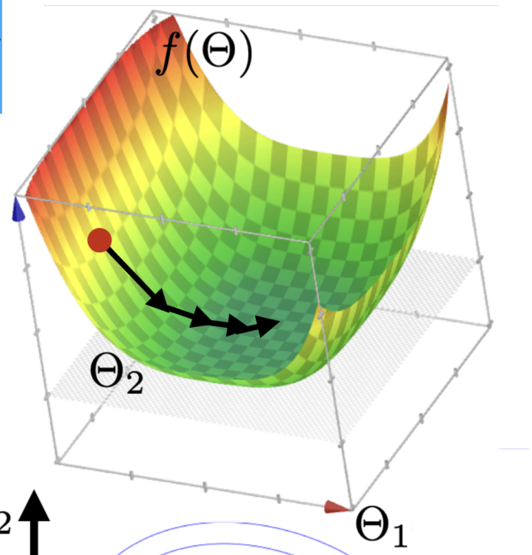


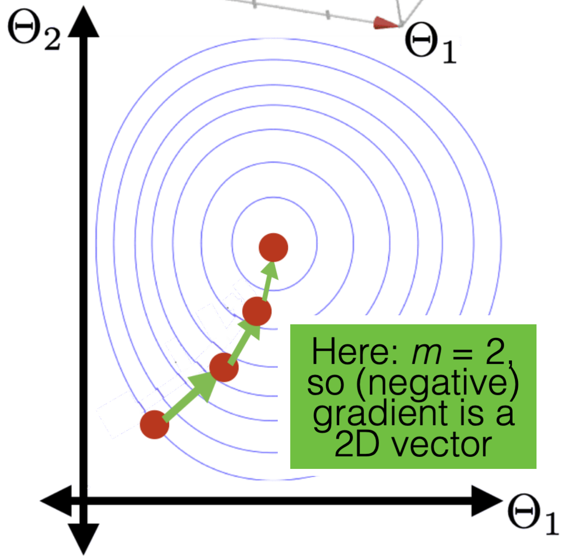
2D vector includes vertical f change



Here: m = 1, so (negative) gradient is a 1D vector

3D vector includes vertical f change





1 Gradient-Descent($\Theta_{\text{init}}, \eta, f, \nabla_\Theta f, \epsilon$)

2 Initialize $\Theta^{(0)} = \Theta_{\text{init}}$

3 Initialize t = 0

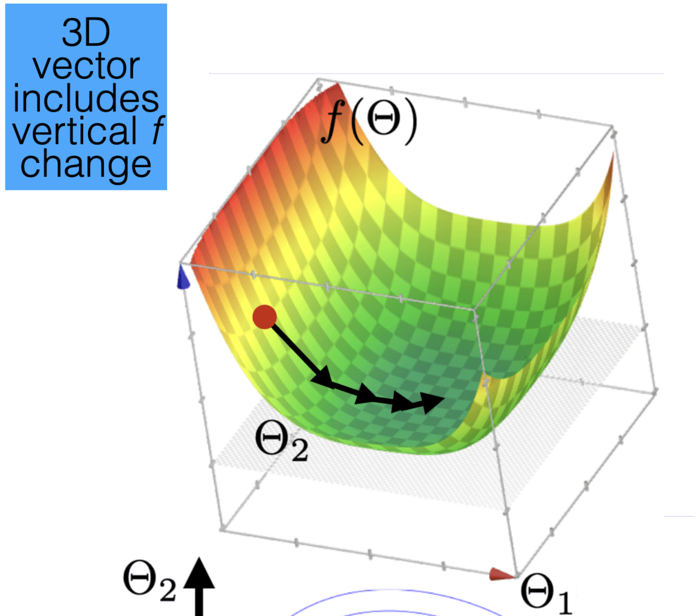
4 **repeat**

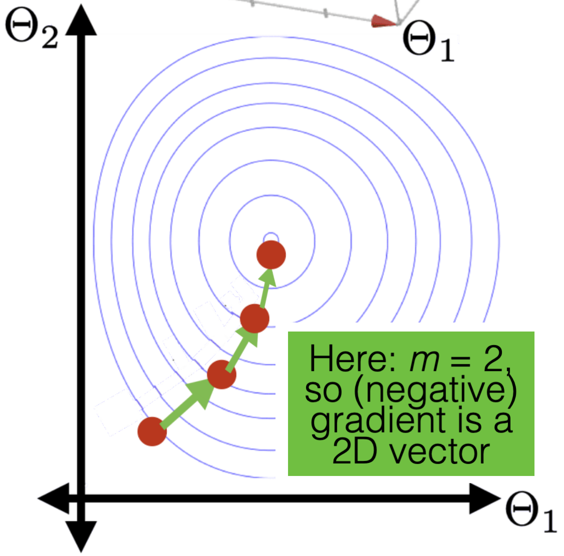
5 t = t + 1

6 $\Theta^{(t)} = \Theta^{(t-1)} - \eta \nabla_\Theta f(\Theta^{(t-1)})$

7 **until** $\left| f(\Theta^{(t)}) - f(\Theta^{(t-1)}) \right| < \epsilon$

8 **Return** $\Theta^{(t)}$





Q: what does this condition imply?

A: the gradient (at the current parameter nearly) is zero.

Other possible stopping criteria for line 7:

- Small parameter change: $\|\Theta^{(t)} - \Theta^{(t-1)}\| < \epsilon$, or
- Small gradient norm: $\|\nabla_\Theta f(\Theta^{(t)})\| < \epsilon$ also imply the same "gradient close to zero"

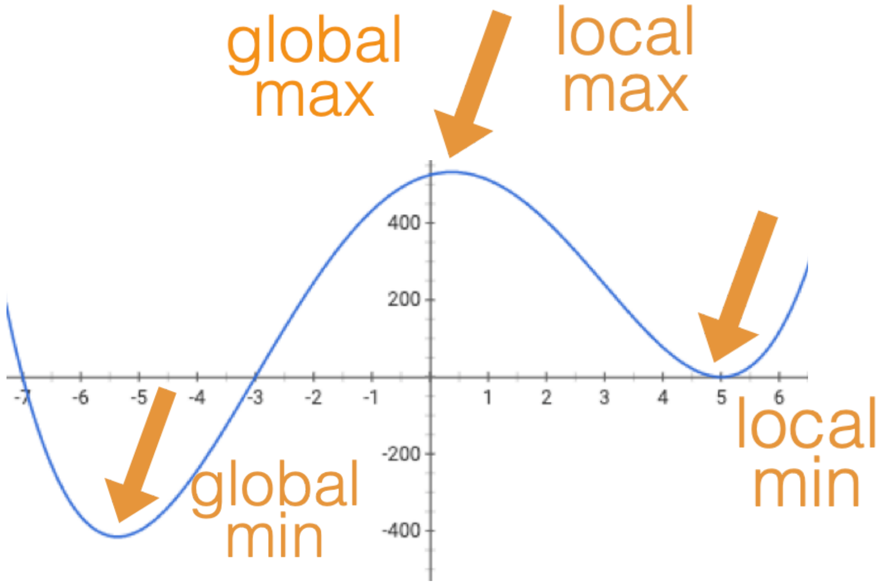
Outline

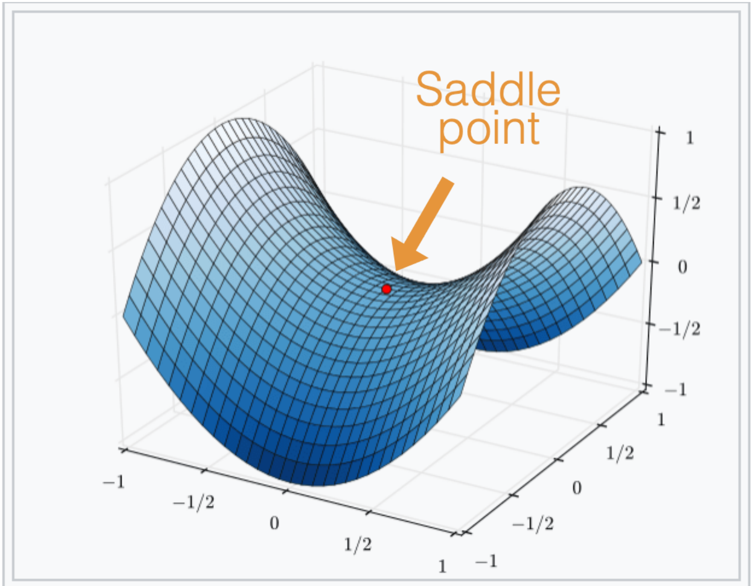
- Recap, motivation for gradient descent methods
- Gradient descent algorithm (GD)
  - The gradient vector
  - GD algorithm
  - Gradient decent properties
    - convex functions, local vs global min
- Stochastic gradient descent (SGD)
  - SGD algorithm and setup
  - GD vs SGD comparison

When minimizing a function, we aim for a global minimizer.

gradient descent can achieve this (to arbitrary precision)

At a global minimizer $\Longrightarrow$ / $\not\Longleftarrow$ the gradient vector is the zero





When minimizing a function, we aim for a global minimizer.

$$\text{At a global minimizer} \quad \Leftarrow \begin{cases} \text{the gradient vector is the zero} \\ \text{the objective function is convex} \end{cases}$$

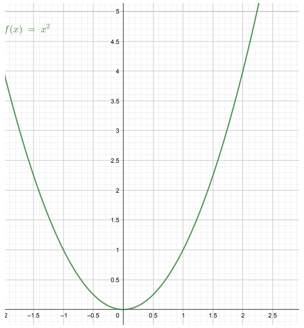
A function f is ***convex*** if any line segment connecting two points of the graph of f lies above or on the graph.

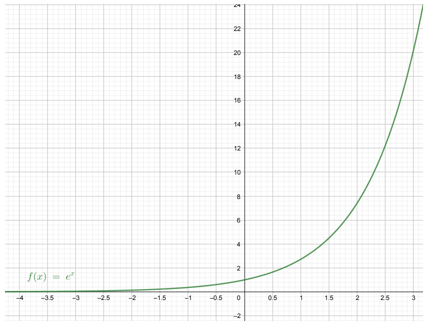
- f is concave if $-f$ is convex.
- one can say *a lot* about optimization convergence for convex functions.

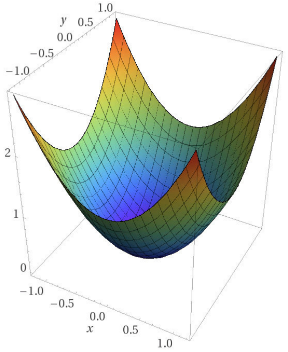
https://shenshen.mit.edu/demos/convex.html

Some examples

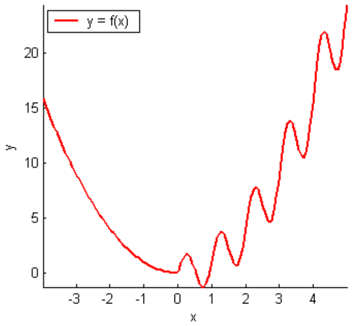
Convex functions

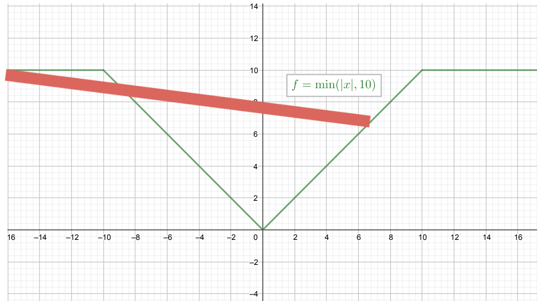


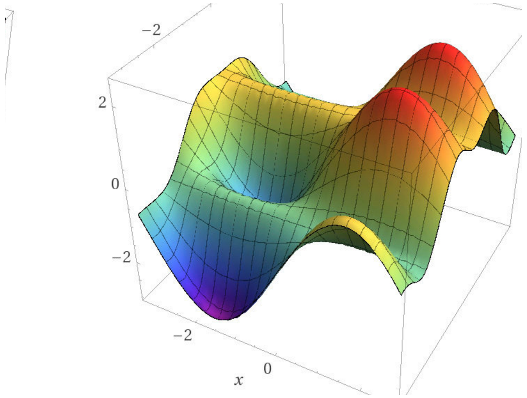




Non-convex functions







Gradient Descent Performance

- Assumptions:
  - f is sufficiently "smooth"
  - f is convex
  - f has at least one global minimum
  - Run gradient descent for sufficient iterations
  - η is sufficiently small
- Conclusion:
  - Gradient descent converges arbitrarily close to a global minimum of f.

Gradient Descent Performance

- Assumptions:
  - f is sufficiently "smooth"
  - f is convex
  - f has at least one global minimum
  - Run gradient descent for sufficient iterations
  - η is sufficiently small

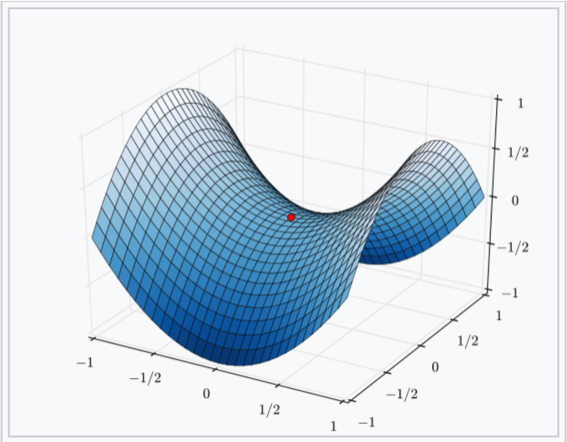
if violated, may not have gradient,
can't run gradient descent

- Conclusion:
  - Gradient descent converges arbitrarily close to a global minimum of f.

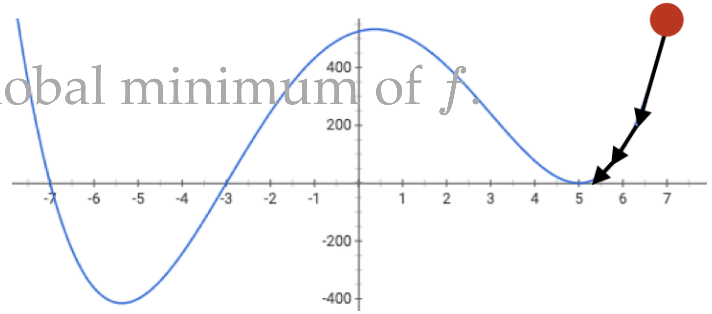
Gradient Descent Performance

- Assumptions:
  - f is sufficiently "smooth"
  - f is convex
  - f has at least one global minimum
  - Run gradient descent for sufficient iterations
  - η is sufficiently small
- Conclusion:
  - Gradient descent converges arbitrarily close to a global minimum of f.

if violated, may get stuck at a saddle point



or a local minimum

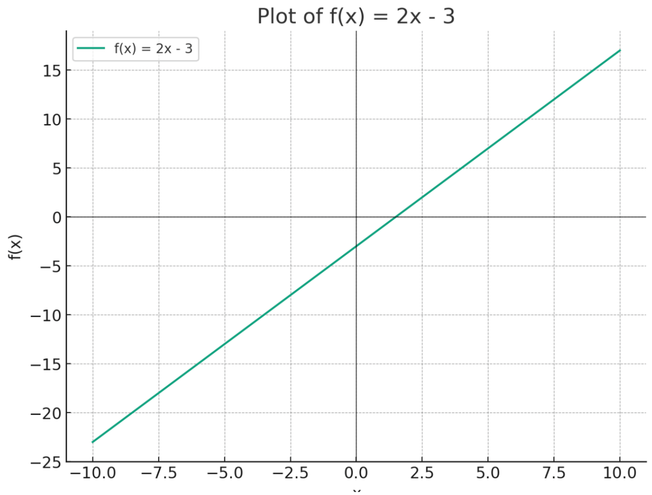


Gradient Descent Performance

- Assumptions:
  - f is sufficiently "smooth"
  - f is convex
  - f has at least one global minimum
  - Run gradient descent for sufficient iterations
  - η is sufficiently small
- Conclusion:
  - Gradient descent converges arbitrarily close to a global minimum of f.

if violated:

may not terminate/no minimum to converge to



Gradient Descent Performance

- Assumptions:
  - f is sufficiently "smooth"
  - f is convex
  - f has at least one global minimum
  - Run gradient descent for sufficient iterations
  - η is sufficiently small
- Conclusion:
  - Gradient descent converges arbitrarily close to a global minimum of f.

if violated:

see demo on next slide,
also lab/recitation/hw

https://shenshen.mit.edu/demos/gd.html

Outline

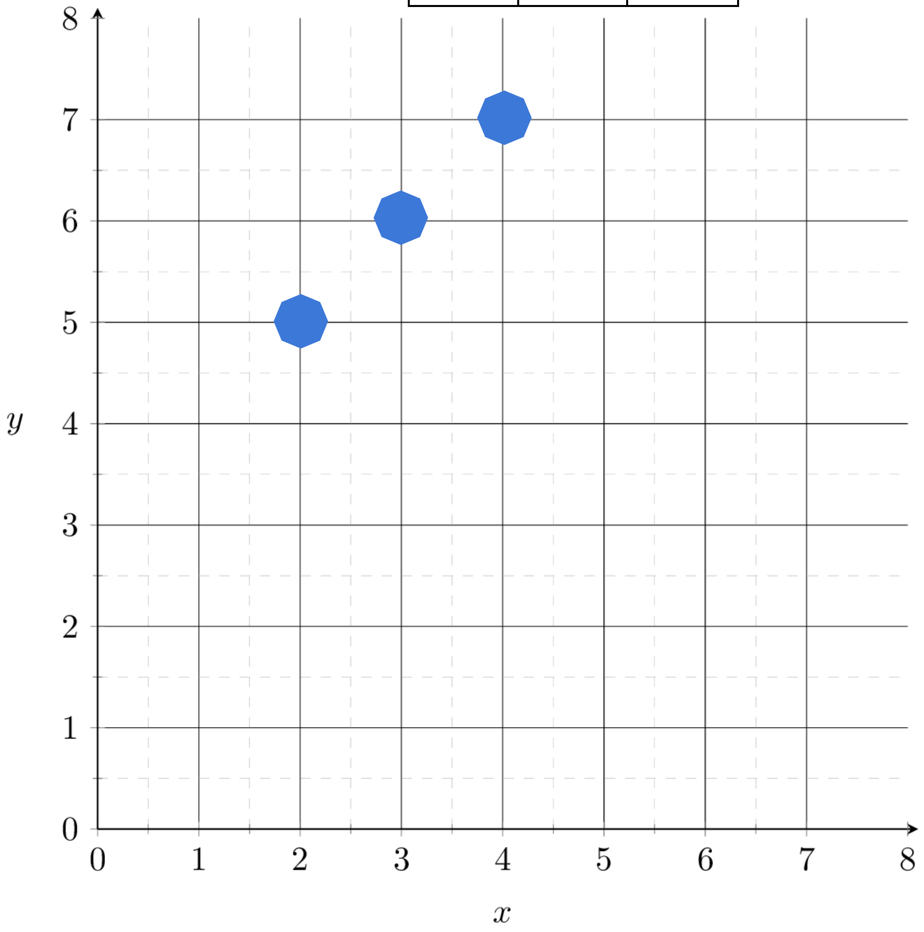
- Recap, motivation for gradient descent methods
- Gradient descent algorithm (GD)
  - The gradient vector
  - GD algorithm
  - Gradient decent properties
    - convex functions, local vs global min
- Stochastic gradient descent (SGD)
  - SGD algorithm and setup
  - GD vs SGD comparison

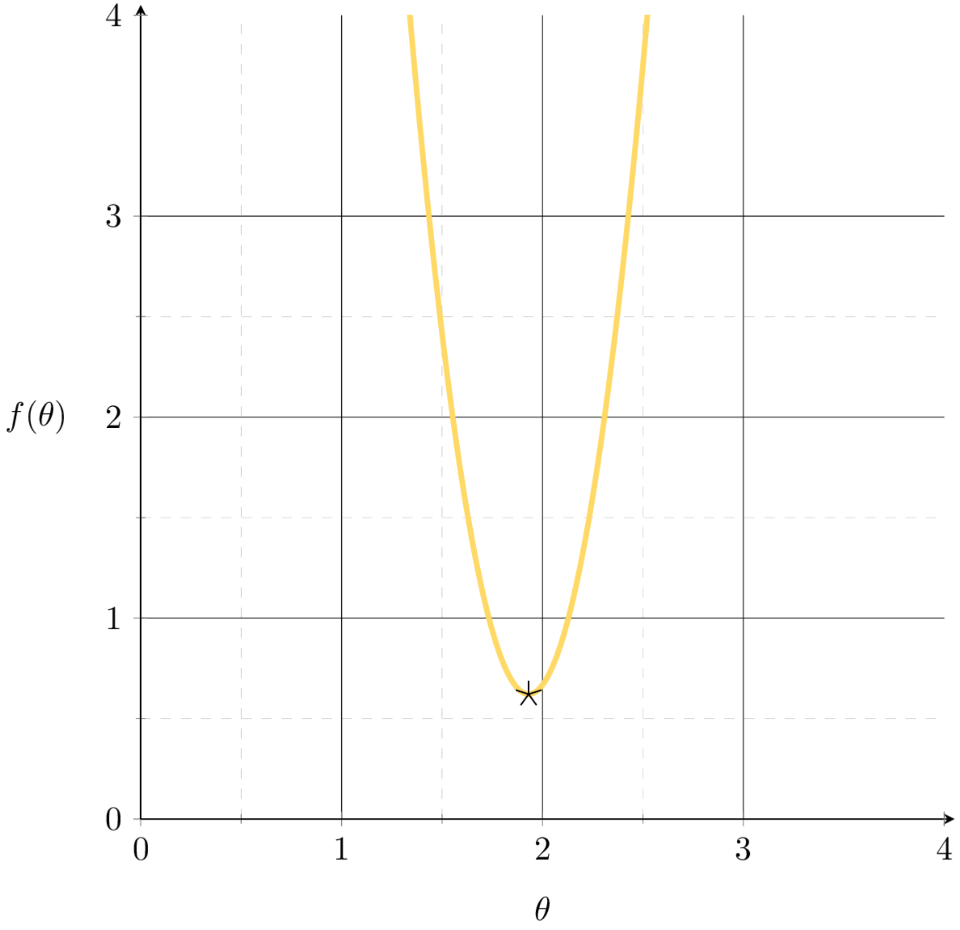
training data

	x	y
p1	2	5
p2	3	6
p3	4	7

Fit a line (without offset) to the dataset, the MSE:

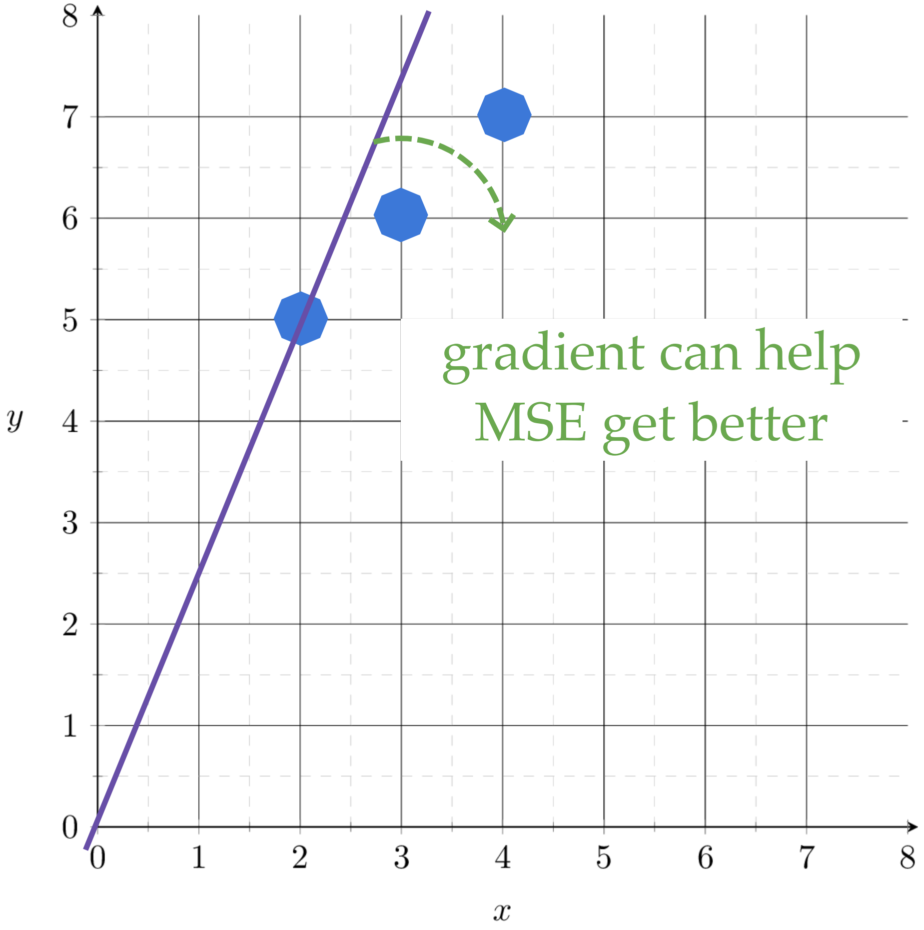
$$f(\theta) = \frac{1}{3}\left[(2\theta - 5)^2 + (3\theta - 6)^2 + (4\theta - 7)^2\right]$$

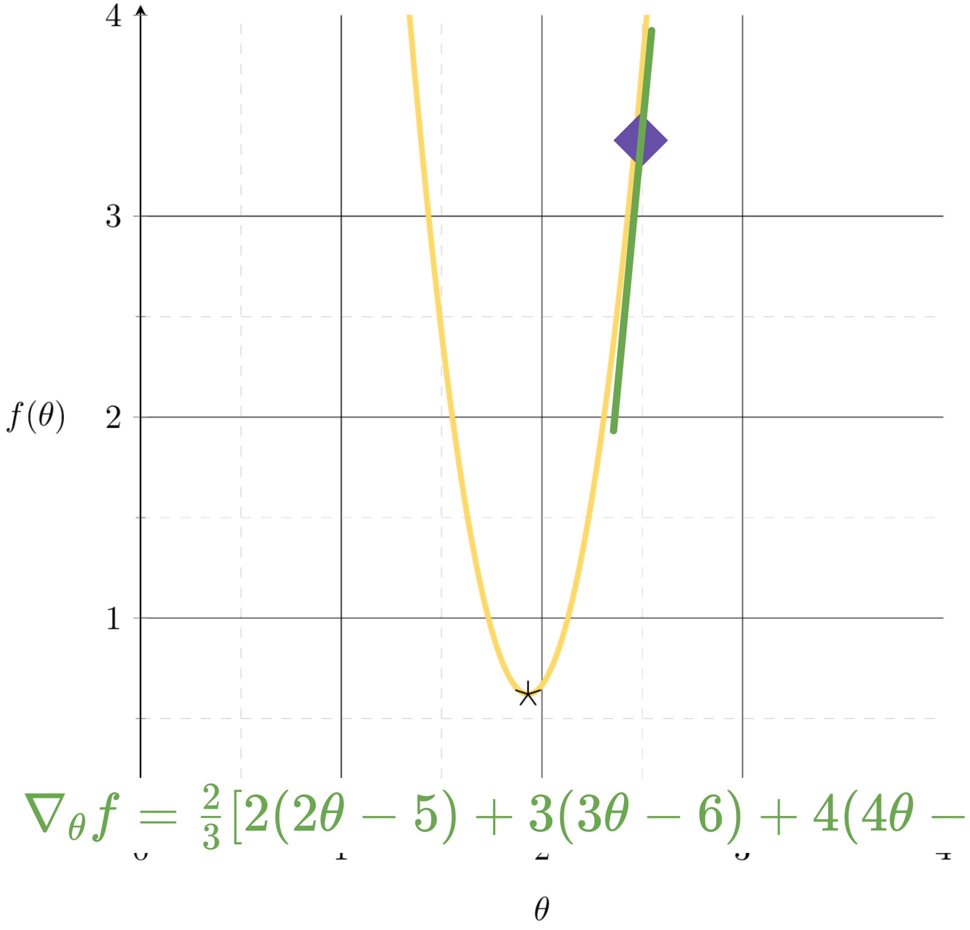




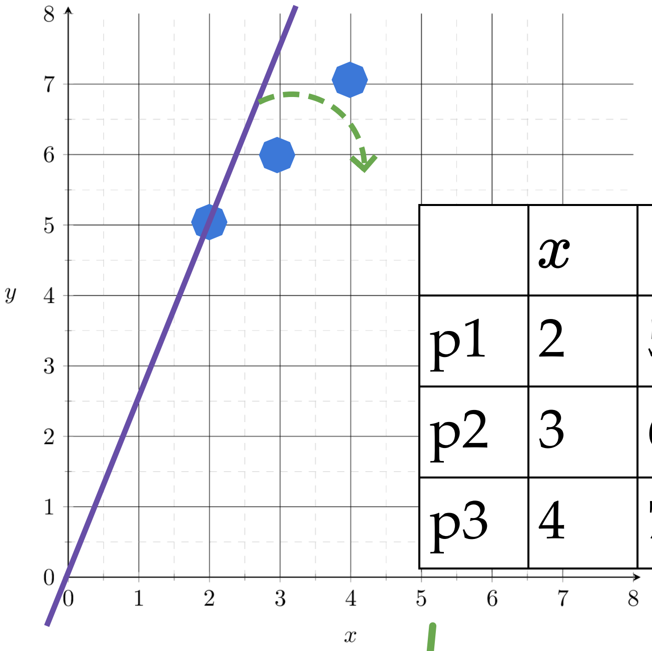
Suppose we fit a line $y = 2.5x$

$$f(\theta) = \frac{1}{3}\left[(2\theta-5)^2 + (3\theta-6)^2 + (4\theta-7)^2\right]$$

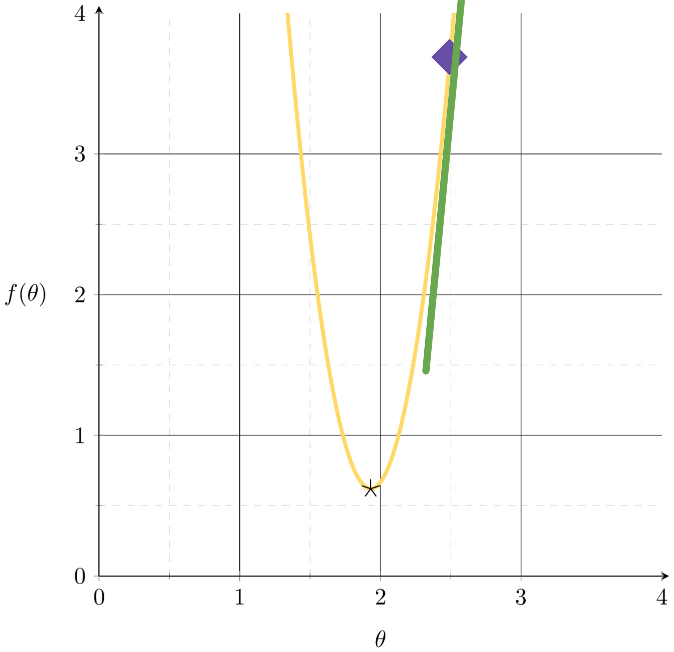




$$\nabla_\theta f = \frac{2}{3}[2(2\theta-5) + 3(3\theta-6) + 4(4\theta-7)]$$



	x	y
p1	2	5
p2	3	6
p3	4	7

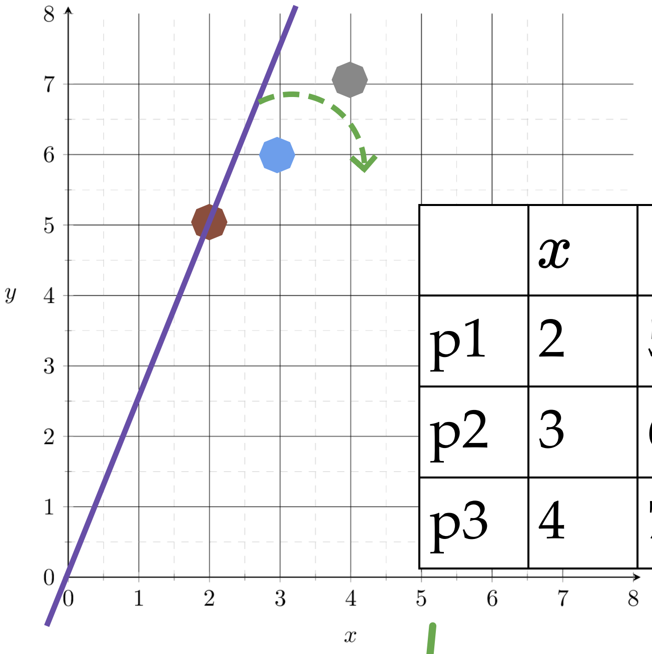


- the MSE of a linear hypothesis:

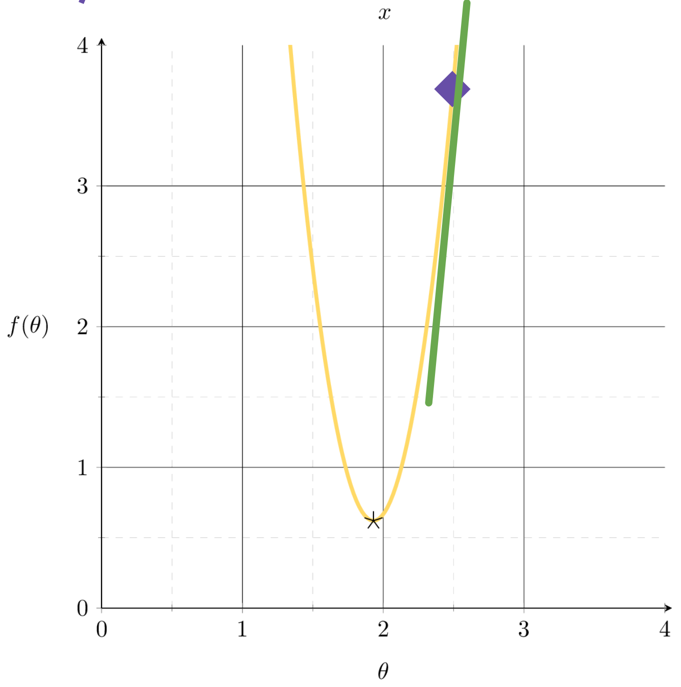
$$f(\theta) = \frac{1}{3}\left[(2\theta-5)^2 + (3\theta-6)^2 + (4\theta-7)^2\right]$$

- and its gradient w.r.t. θ:

$$\nabla_\theta f = \frac{2}{3}[2(2\theta-5) + 3(3\theta-6) + 4(4\theta-7)]$$



	x	y
p1	2	5
p2	3	6
p3	4	7



- the MSE of a linear hypothesis:

$$f(\theta) = \frac{1}{3}\left[(2\theta - 5)^2 + (3\theta - 6)^2 + (4\theta - 7)^2\right]$$

- and its gradient w.r.t. θ:

$$\nabla_\theta f = \frac{2}{3}\left[2(2\theta - 5) + 3(3\theta - 6) + 4(4\theta - 7)\right]$$

Gradient of an ML objective

Using our example data set,

- the MSE of a linear hypothesis:

$$f(\theta) = \tfrac{1}{3}\left[(2\theta - 5)^2 + (3\theta - 6)^2 + (4\theta - 7)^2\right]$$

- and its gradient w.r.t. θ:

$$\nabla_\theta f = \tfrac{2}{3}[2(2\theta - 5) + 3(3\theta - 6) + 4(4\theta - 7)]$$

Using any dataset,

- the MSE of a linear hypothesis:

$$f(\theta) = \frac{1}{n}\sum_{i=1}^{n}\left(\theta^\top x^{(i)} - y^{(i)}\right)^2$$

- and its gradient w.r.t. θ:

$$\nabla f(\theta) = \frac{2}{n}\sum_{i=1}^{n}\left(\theta^\top x^{(i)} - y^{(i)}\right)x^{(i)}$$

Gradient of an ML objective

- the MSE of a linear hypothesis:

$$f(\theta) = \frac{1}{n}\sum_{i=1}^{n} \left(\theta^\top x^{(i)} - y^{(i)}\right)^2$$

- and its gradient w.r.t. θ:

$$\nabla f(\theta) = \frac{2}{n}\sum_{i=1}^{n} \left(\theta^\top x^{(i)} - y^{(i)}\right) x^{(i)}$$

In general,

- An ML objective function is a finite sum

$$f(\theta) = \frac{1}{n}\sum_{i=1}^{n} f_i(\theta)$$

- and its gradient w.r.t. θ:

$$\nabla f(\theta) = \nabla\left(\frac{1}{n}\sum_{i=1}^{n} f_i(\theta)\right) = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\theta)$$

👆

👋 (gradient of the sum) = (sum of the gradient)

Gradient of an ML objective

In general,

- An ML objective function is a finite sum

$$f(\theta) = \frac{1}{n}\sum_{i=1}^{n} f_i(\theta)$$

- and its gradient w.r.t. θ:

need to add n of them

$$\nabla f(\theta) = \nabla\left(\frac{1}{n}\sum_{i=1}^{n} f_i(\theta)\right) = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\theta)$$

each of these $\nabla f_i(\theta) \in \mathbb{R}^d$

Costly!

Let's do stochastic gradient descent (on the board).

Stochastic gradient descent

```
Gradient-Descent( $\Theta_{\text{init}}, \eta, f, \nabla_\Theta f, \epsilon$ )
  Initialize  $\Theta^{(0)} = \Theta_{\text{init}}$
  Initialize t = 0
  repeat
    t = t + 1
    $\Theta^{(t)} = \Theta^{(t-1)} - \eta \nabla_\Theta f(\Theta^{(t-1)})$
  until $\left| f(\Theta^{(t)}) - f(\Theta^{(t-1)}) \right| < \epsilon$
  Return  $\Theta^{(t)}$
```

Stochastic

```
Gradient-Descent( $\Theta_{\text{init}}, \eta, f, \nabla_\Theta f, \epsilon$ )
  Initialize  $\Theta^{(0)} = \Theta_{\text{init}}$
  Initialize t = 0
  repeat
    t = t + 1
    randomly select i from {1,…,n}
    $\Theta^{(t)} = \Theta^{(t-1)} - \eta(t) \nabla_\Theta f_i(\Theta^{(t-1)})$
  until $\left| f(\Theta^{(t)}) - f(\Theta^{(t-1)}) \right| < \epsilon$
  Return  $\Theta^{(t)}$
```

$$\nabla f(\Theta) = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(\Theta) \approx \nabla f_i(\Theta)$$

for a randomly picked data point i

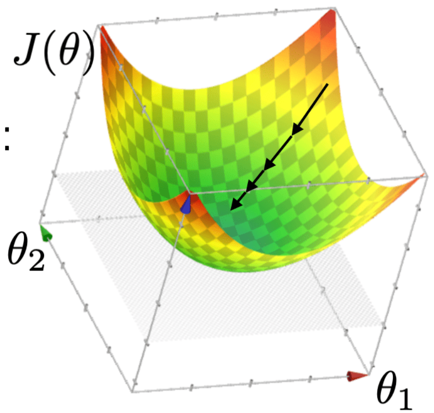
Stochastic gradient descent performance

- Assumptions:
  - f is sufficiently "smooth"
  - f is convex
  - f has at least one global minimum
  - Run gradient descent for sufficient iterations
  - η is sufficiently small and satisfies additional "scheduling" condition
- Conclusion:

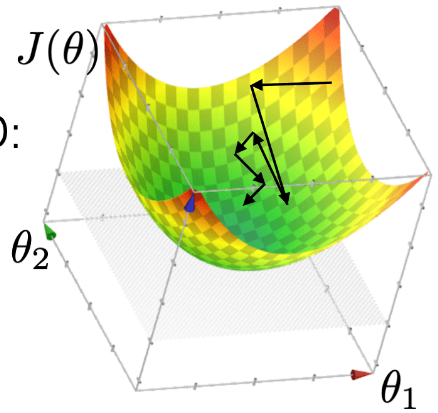
$\sum_{t=1}^{\infty} \eta(t) = \infty$ and $\sum_{t=1}^{\infty} \eta(t)^2 < \infty$

  - Stochastic gradient descent converges arbitrarily close to a global minimum of f with probability 1.

- GD:

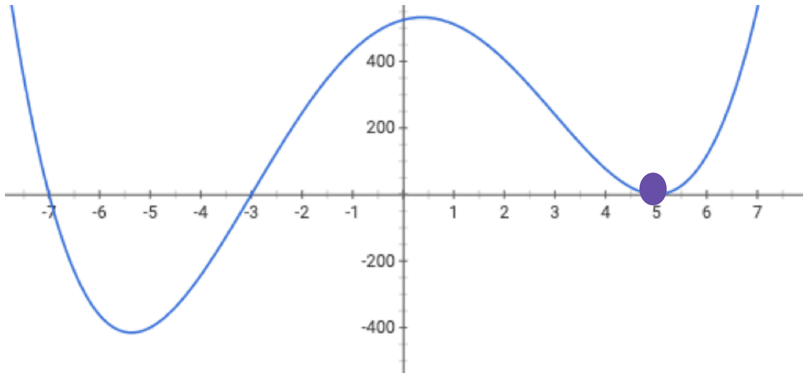


- SGD:



Compared with GD, SGD

is more "random"



may get us out of a local min

$$\nabla f(\Theta) = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\Theta) \approx \nabla f_i(\Theta)$$

is more efficient

Summary

- Most ML methods can be formulated as optimization problems.
- We won't always be able to solve optimization problems analytically (in closed-form).
- We won't always be able to solve (for a global optimum) efficiently.
- We can still use numerical algorithms to good effect. Lots of sophisticated ones available.
- Introduce the idea of gradient descent in 1D: only two directions! But magnitude of step is important.
- In higher dimensions the direction is very important as well as magnitude.
- GD, under appropriate conditions (most notably, when objective function is convex), can guarantee convergence to a global minimum.
- SGD: approximated GD, more efficient, more random, and less guarantees.

https://docs.google.com/forms/d/e/1FAIpQLScj9i83AI8TuhWDZXSjiWzX6gZpnPugjGsH-i3RdrBCtF-opg/viewform?embedded=true

We'd love to hear your thoughts.

Thanks!