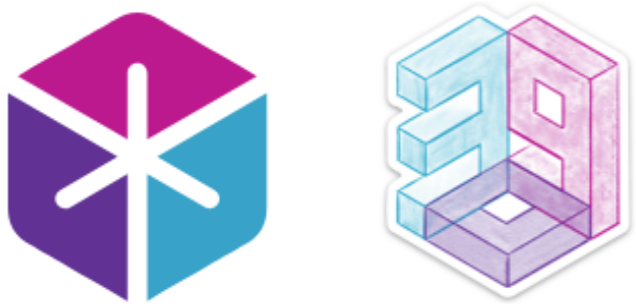
ChatGPT 4.0

Gradient Descent

Rina BUOY, PhD

Disclaimer

Adopted from



6.390

Introduction to Machine Learning

(Fall 2024)

https://introml.mit.edu/fall24

Outline

- Recap of last (content) week.
- Ordinary least-square regression
  - Analytical solution (when exists)
  - Cases when analytical solutions don't exist
    - Practically, visually, mathamtically
- Regularization
- Hyperparameter, cross-validation

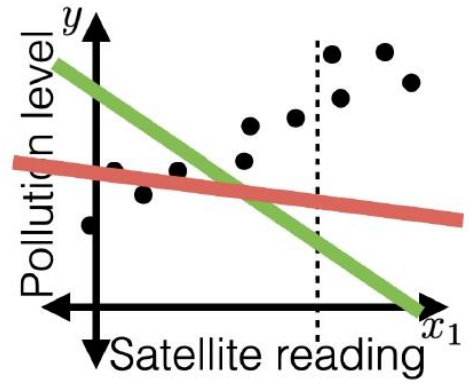
Outline

- 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

$$\frac{1}{n}\sum_{i=1}^{n} L(h(x^{(i)};\Theta), y^{(i)}) + \lambda R(\Theta)$$

- A general ML approach
  - Collect data
  - Choose hypothesis class, hyperparameter, loss function
  - Train (optimize for) "good" hypothesis by minimizing loss.



- Limitations of a closed-form solution for objective minimizer
  - Don't always have closed-form solutions. (Recall, half-pipe cases.)
  - Ridge came to the rescue, but we don't always have such "savior".
  - Even when closed-form solutions exist, can be expensive to compute (recall, lab2, Q2.8)
- Want a more general, efficient way! (=> GD methods today)

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

Gradient

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

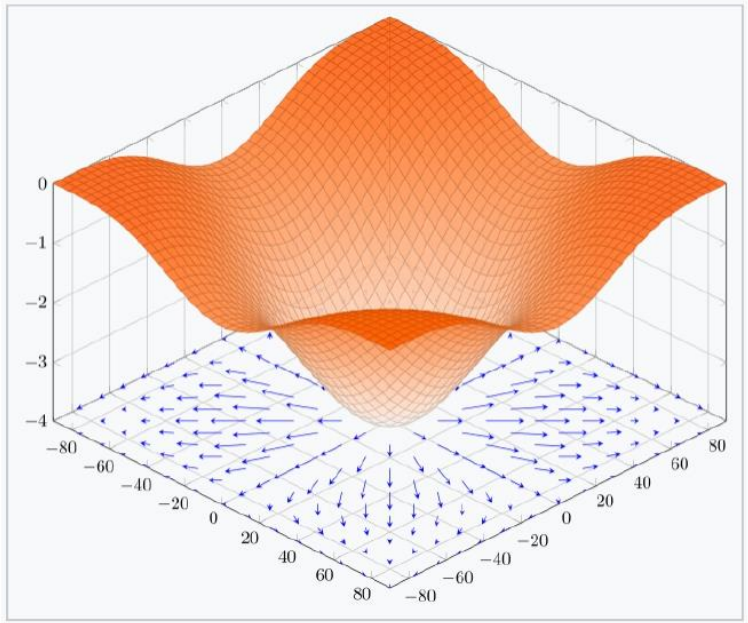
$$\nabla f(p) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(p) \\ \vdots \\ \frac{\partial f}{\partial x_m}(p) \end{bmatrix}$$

1. Generalizes 1-dimensional derivatives.
2. By construction, always has the same dimensionality as the function input.

(Aside: sometimes, the gradient doesn't exist, or doesn't behave nicely, as we'll see later in this course. For today, we have well-defined, nice, gradients.)

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

one cute example:



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

another example

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

a gradient can be the (symbolic) function

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

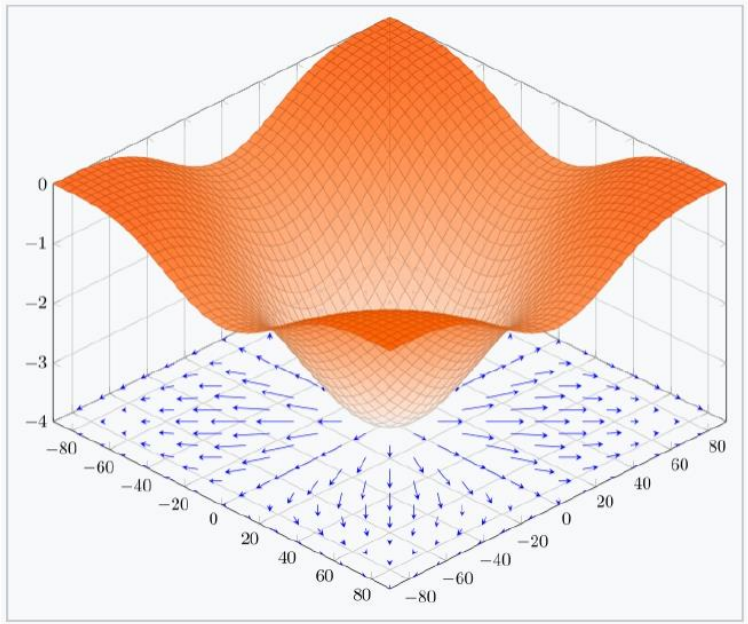
or,

we can also evaluate the gradient function at a point and get (numerical) gradient vectors

$$\nabla f(3, 2, 1) = \begin{bmatrix} 6 \\ 12 \\ 1 \end{bmatrix}$$

exactly like how derivative can be both a function and a number.

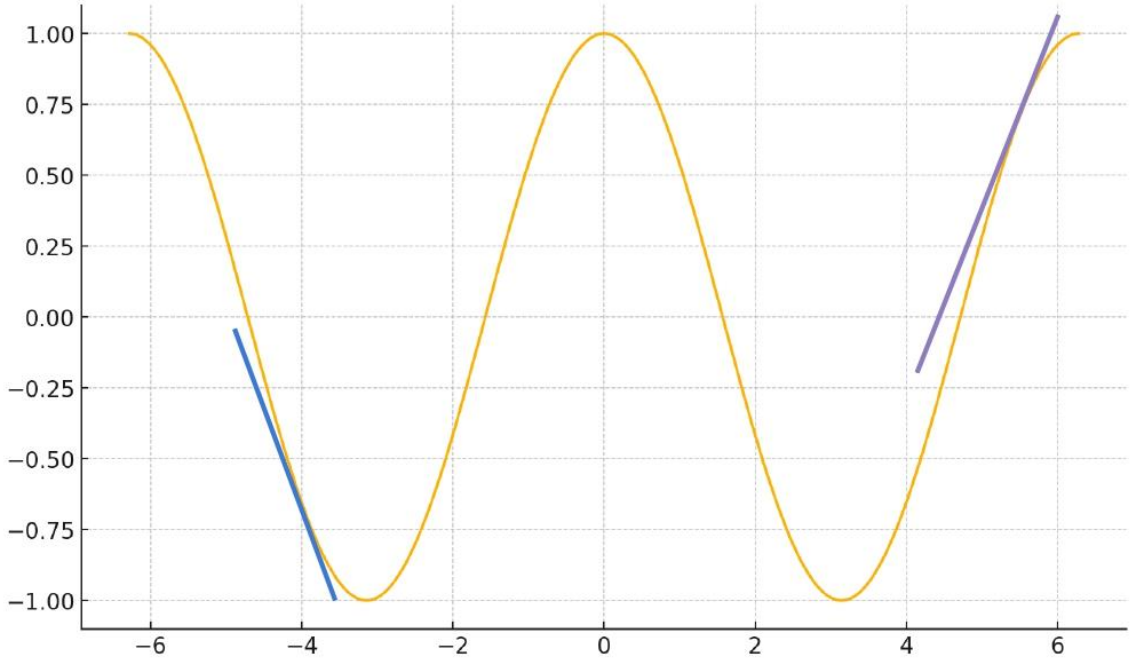
$$\nabla f(p) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(p) \\ \vdots \\ \frac{\partial f}{\partial x_m}(p) \end{bmatrix}$$



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

3. the gradient points in the direction of the (steepest) increase in the function value.

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



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

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

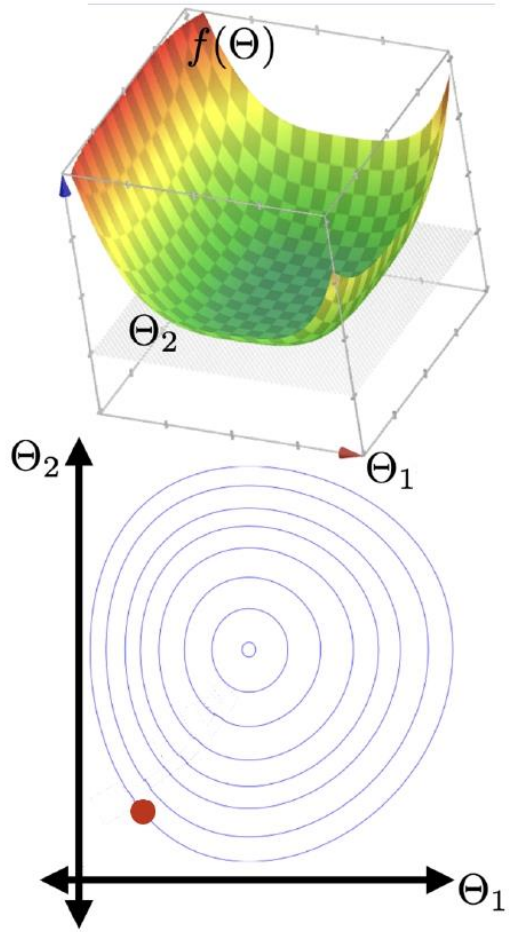
initial guess — learning rate,

hyperparameters — of parameters — aka, step size — precision

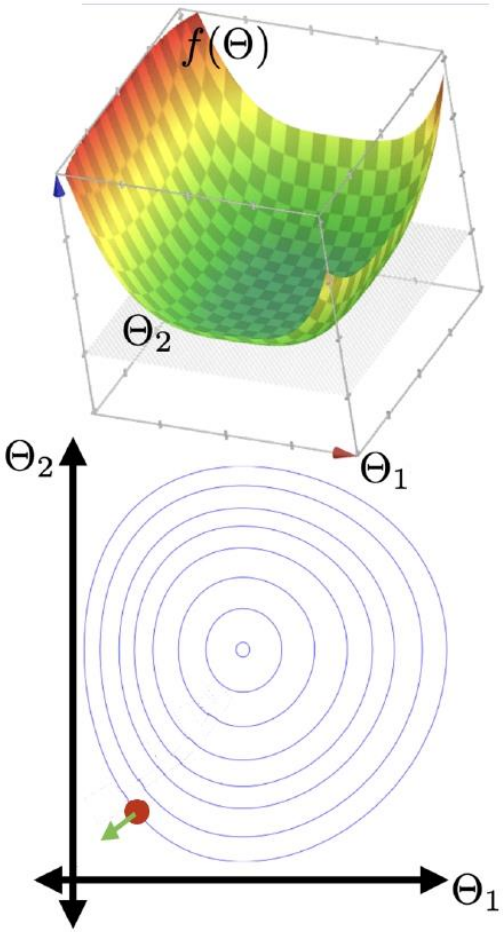
```
Gradient-Descent( Θ_init, η, f, ∇_Θ f, ε )
  Initialize  Θ^(0) = Θ_init
  Initialize t = 0
  repeat
    t = t + 1
    Θ^(t) = Θ^(t−1) − η ∇_Θ f(Θ^(t−1))
  until  |f(Θ^(t)) − f(Θ^(t−1))| < ε
  Return  Θ^(t)
```

Adopted from : https://introml.mit.edu

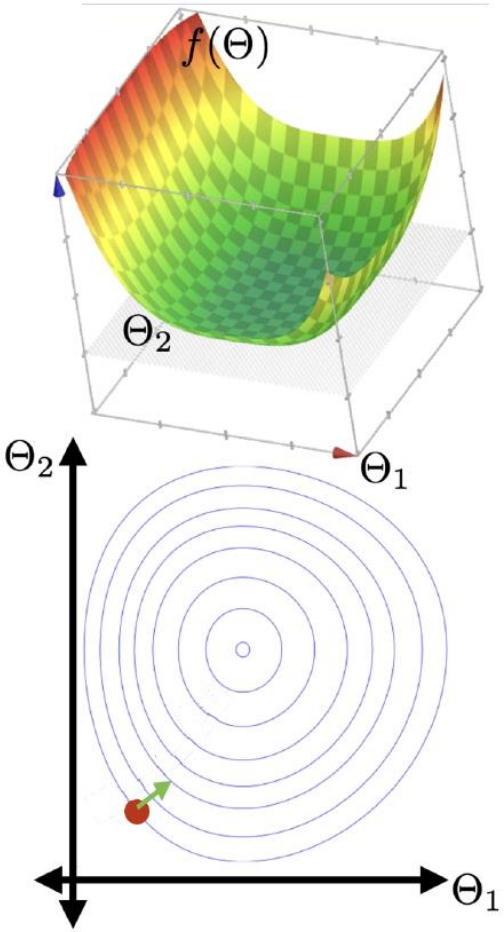
1 Gradient-Descent($\Theta_{\text{init}}, \eta, f, \nabla_\Theta f, \epsilon$)
2 Initialize $\Theta^{(0)} = \Theta_{\text{init}}$
3 Initialize t = 0
4 **repeat**
5 t = t + 1
6 $\Theta^{(t)} = \Theta^{(t-1)} - \eta \nabla_\Theta f(\Theta^{(t-1)})$
7 **until** $\left| f(\Theta^{(t)}) - f(\Theta^{(t-1)}) \right| < \epsilon$
8 **Return** $\Theta^{(t)}$



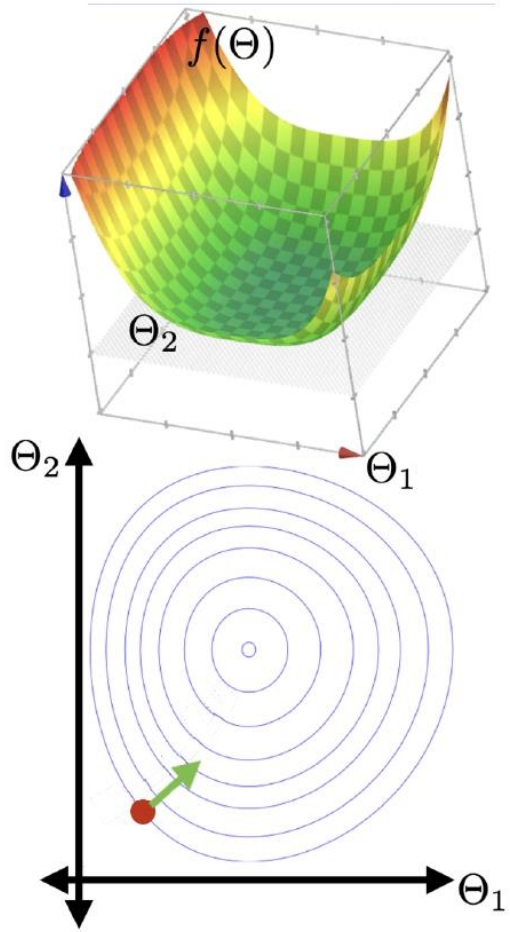
1 Gradient-Descent ($\Theta_{\text{init}}, \eta, f, \nabla_\Theta f, \epsilon$)
2 Initialize $\Theta^{(0)} = \Theta_{\text{init}}$
3 Initialize t = 0
4 **repeat**
5 t = t + 1
6 $\Theta^{(t)} = \Theta^{(t-1)} - \eta \nabla_\Theta f(\Theta^{(t-1)})$
7 **until** $\left| f(\Theta^{(t)}) - f(\Theta^{(t-1)}) \right| < \epsilon$
8 **Return** $\Theta^{(t)}$



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

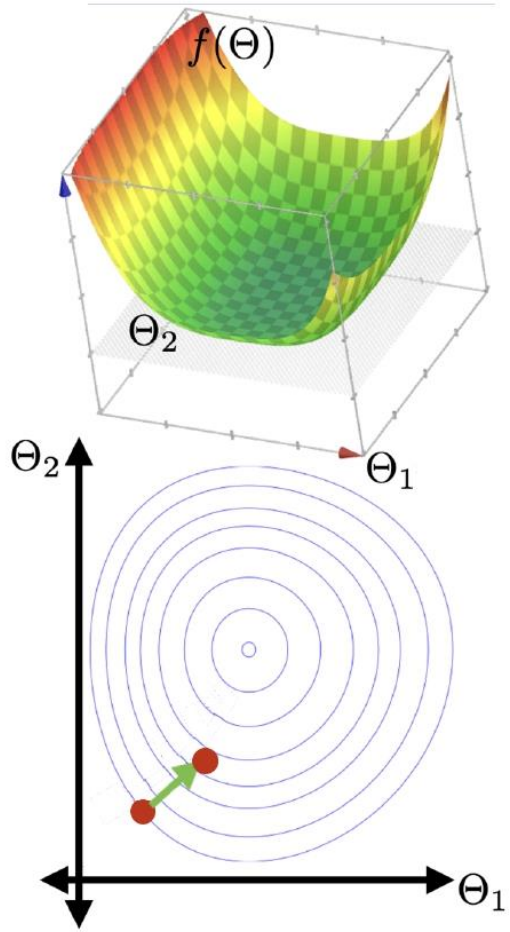


```
1 Gradient-Descent( Θ_init, η, f, ∇_Θ f, ε )
2   Initialize  Θ^(0) = Θ_init
3   Initialize t = 0
4   repeat
5     t = t + 1
6     Θ^(t) = Θ^(t−1) − η∇_Θ f(Θ^(t−1))
7   until  |f(Θ^(t)) − f(Θ^(t−1))| < ε
8   Return  Θ^(t)
```

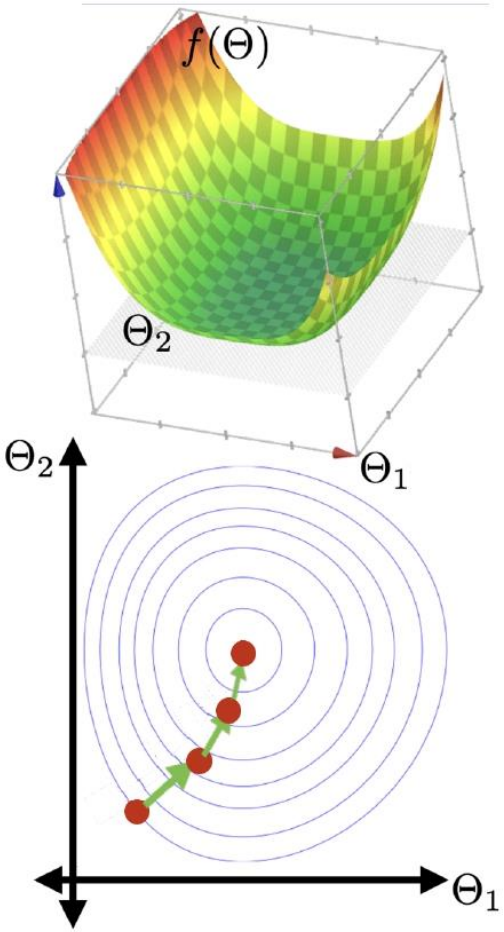


Adopted from : https://introml.mit.edu

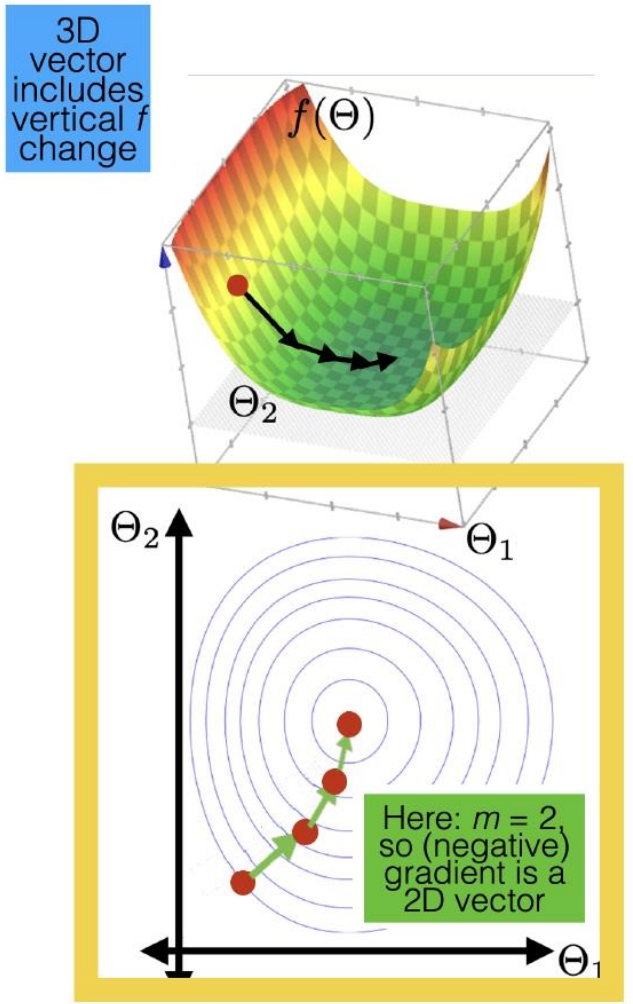
1 Gradient-Descent($\Theta_{\text{init}}, \eta, f, \nabla_\Theta f, \epsilon$)
2 Initialize $\Theta^{(0)} = \Theta_{\text{init}}$
3 Initialize t = 0
4 **repeat**
5 t = t + 1
6 $\Theta^{(t)} = \Theta^{(t-1)} - \eta \nabla_\Theta f(\Theta^{(t-1)})$
7 **until** $\left| f(\Theta^{(t)}) - f(\Theta^{(t-1)}) \right| < \epsilon$
8 **Return** $\Theta^{(t)}$



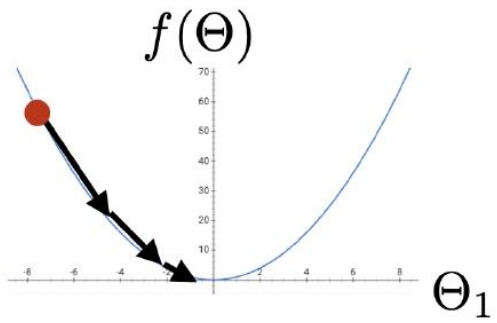
1 Gradient-Descent ($\Theta_{\text{init}}, \eta, f, \nabla_\Theta f, \epsilon$)
2 Initialize $\Theta^{(0)} = \Theta_{\text{init}}$
3 Initialize t = 0
4 **repeat**
5 t = t + 1
6 $\Theta^{(t)} = \Theta^{(t-1)} - \eta \nabla_\Theta f(\Theta^{(t-1)})$
7 **until** $\left| f(\Theta^{(t)}) - f(\Theta^{(t-1)}) \right| < \epsilon$
8 **Return** $\Theta^{(t)}$



```
1 Gradient-Descent( Θ_init, η, f, ∇_Θ f, ε )
2   Initialize  Θ^(0) = Θ_init
3   Initialize t = 0
4   repeat
5     t = t + 1
6     Θ^(t) = Θ^(t−1) − η ∇_Θ f(Θ^(t−1))
7   until  |f(Θ^(t)) − f(Θ^(t−1))| < ε
8   Return  Θ^(t)
```

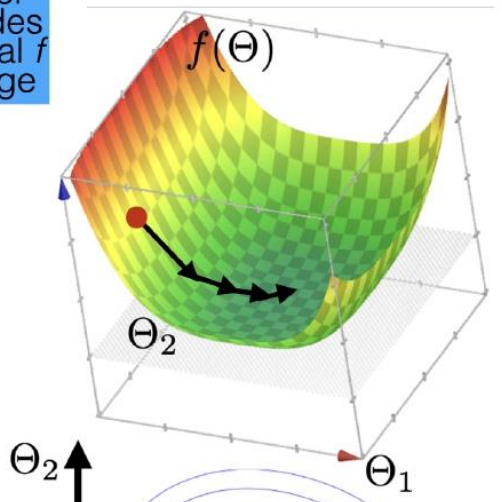


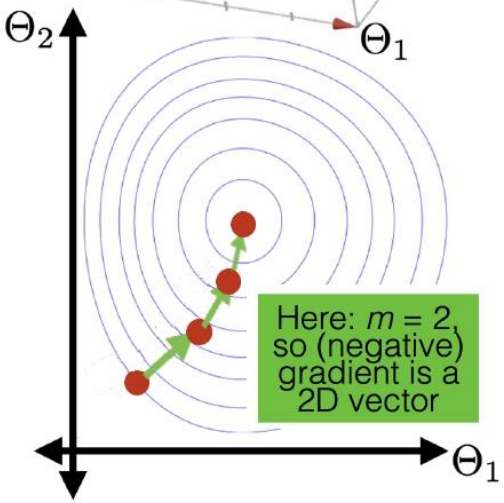
2D vector includes vertical *f* change



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

3D vector includes vertical *f* change

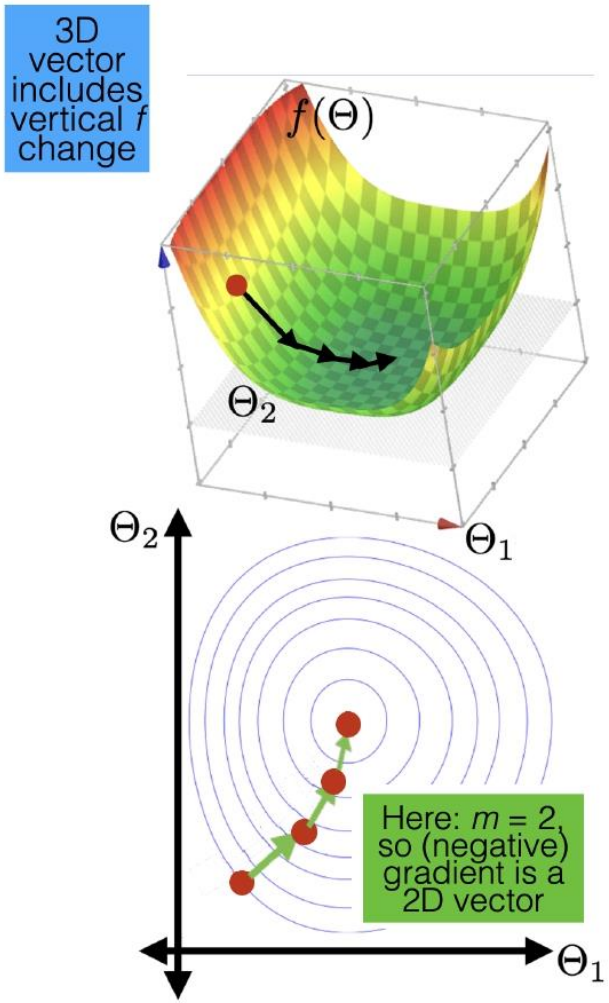




```
1 Gradient-Descent( Θ_init, η, f, ∇_Θ f, ε )
2   Initialize  Θ^(0) = Θ_init
3   Initialize t = 0
4   repeat
5     t = t + 1
6     Θ^(t) = Θ^(t−1) − η ∇_Θ f(Θ^(t−1))
7   until  |f(Θ^(t)) − f(Θ^(t−1))| < ε
8   Return  Θ^(t)
```

Q: if this condition is satisfied, what does it imply?

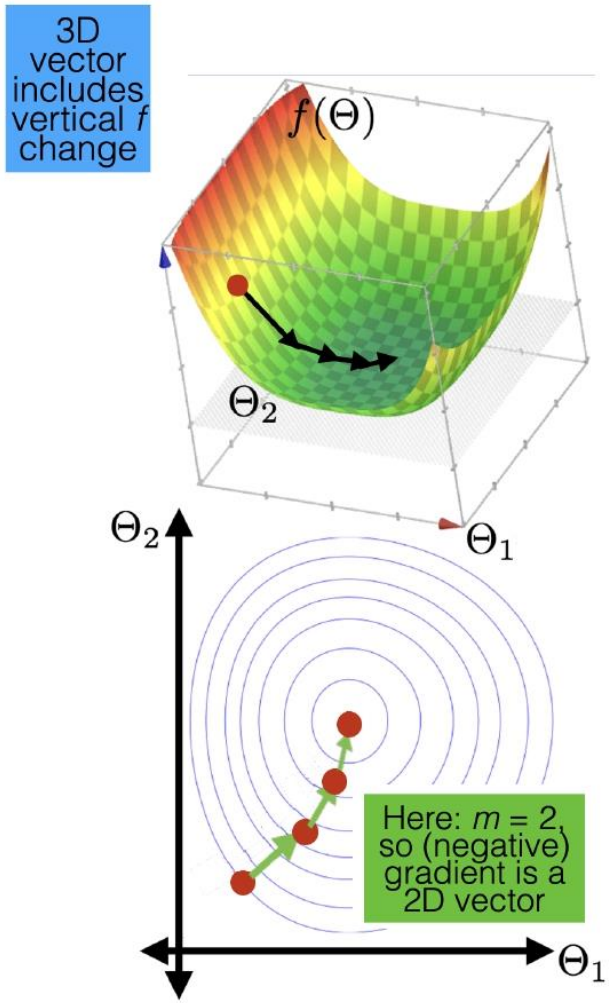
A: the gradient at the current parameter is almost zero.



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

Other possible stopping criteria for line 7:

- Parameter norm change between iteration $\left\| \Theta^{(t)} - \Theta^{(t-1)} \right\| < \epsilon$
- Gradient norm close to zero $\left\| \nabla_\Theta f\left(\Theta^{(t)}\right) \right\| < \epsilon$
- Max number of iterations T



Outline

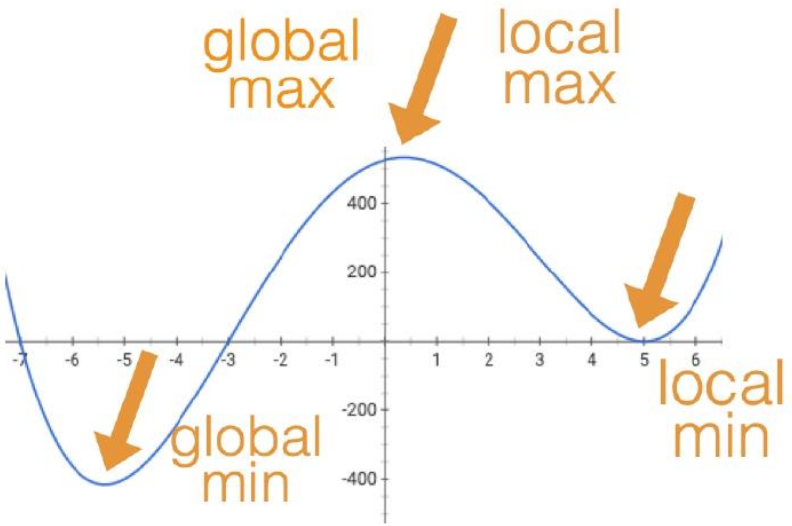
- Recap, motivation for gradient descent methods
- Gradient descent algorithm (GD)
  - The gradient vector
  - GD algorithm
  - Gradient decent properties
    - convex functions, local vs global min
- Stochastic gradient descent (SGD)
  - SGD algorithm and setup
  - GD vs SGD comparison

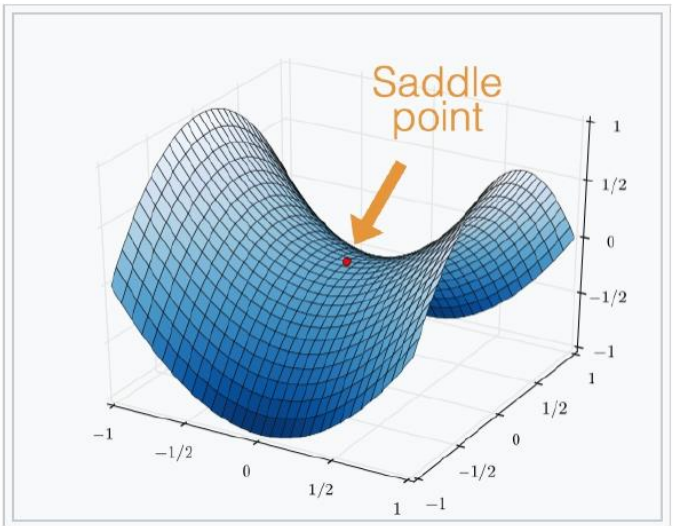
When minimizing a function, we'd hope to get to a global minimizer

$\Rightarrow$

At a global minimizer the gradient vector is the zero vector

$\not\Leftarrow$





When minimizing a function, we'd hope to get to a global minimizer

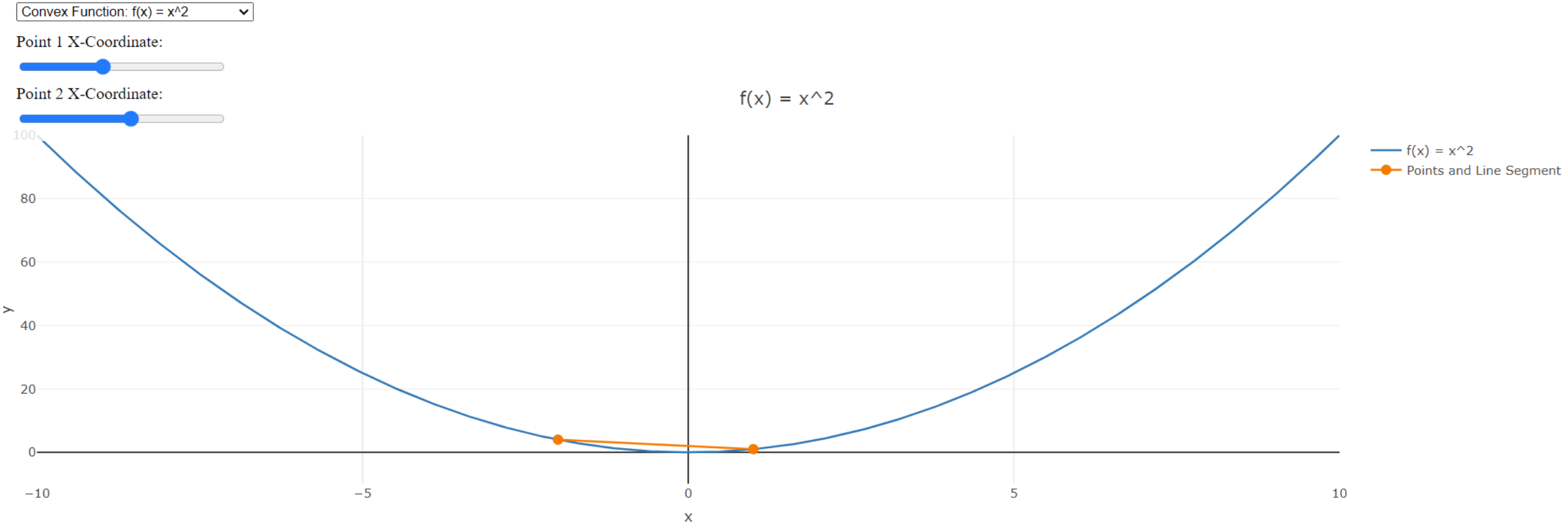
At a global minimizer $\Leftarrow$ { the gradient vector is the zero vector ; the function is a convex function

A function f is ***convex***

if any line segment connecting two points of the graph of f lies above or on the graph.

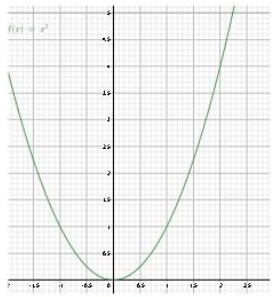
- (f is concave if $-f$ is convex.)
- (one can say a lot about optimization convergence for convex functions.)

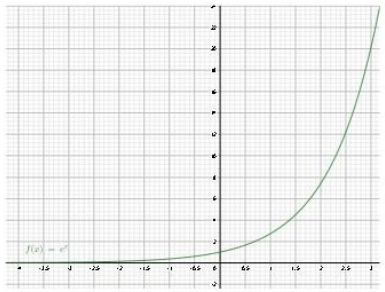
Demo - https://shenshen.mit.edu/demos/convex.html

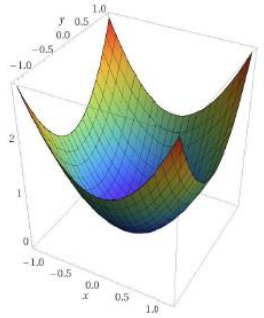


Some examples

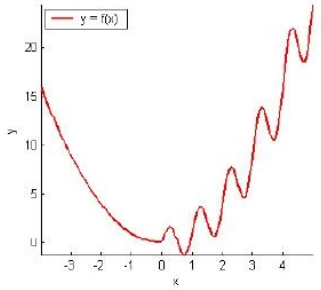
Convex functions

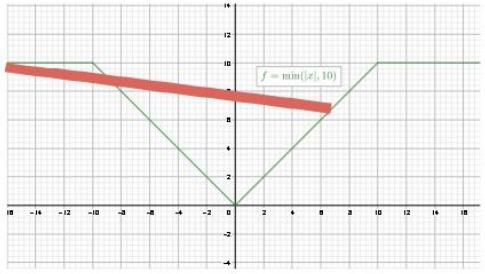


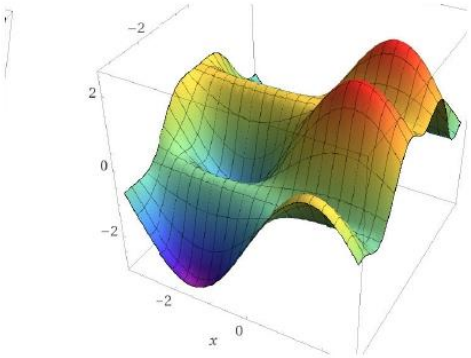




Non-convex functions







Gradient Descent Performance

- Assumptions:
  - f is sufficiently "smooth"
  - f has at least one global minimum
  - Run the algorithm long enough
  - η is sufficiently small
  - f is convex
- Conclusion:
  - Gradient descent will return a parameter value within $\tilde{\epsilon}$ of a global minimum (for any chosen $\tilde{\epsilon} > 0$)

Gradient Descent Performance

- Assumptions:
  - f is sufficiently "smooth"
  - f has at least one global minimum
  - Run the algorithm long enough
  - η is sufficiently small
  - f is convex
- Conclusion:
  - Gradient descent will return a parameter value within $\tilde{\epsilon}$ of a global minimum (for any chosen $\tilde{\epsilon} > 0$)

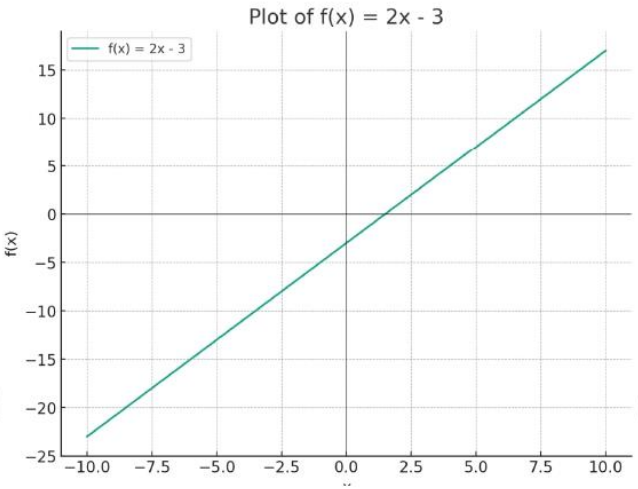
if violated, may not have gradient, can't run gradient descent

Gradient Descent Performance

- Assumptions:
  - f is sufficiently "smooth"
  - f has at least one global minimum
  - Run the algorithm long enough
  - η is sufficiently small
  - f is convex
- Conclusion:
  - Gradient descent will return a parameter value wit … r any chosen $\tilde{\epsilon} > 0$

if violated:

may not terminate/no minimum to converge to



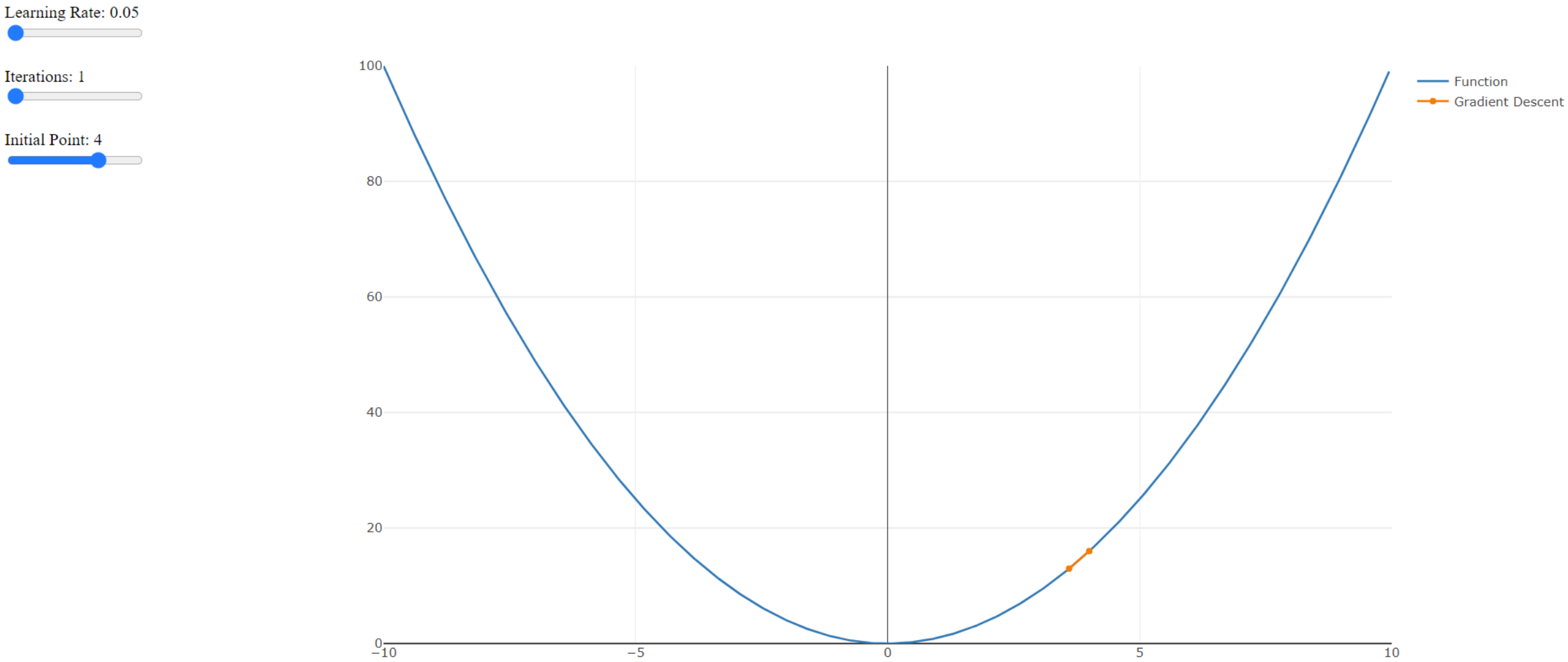
Gradient Descent Performance

- Assumptions:
  - f is sufficiently "smooth"
  - f has at least one global minimum
  - Run the algorithm long enough
  - η is sufficiently small
  - f is convex
- Conclusion:
  - Gradient descent will return a parameter value within $\tilde{\epsilon}$ of a global minimum (for any chosen $\tilde{\epsilon} > 0$)

if violated:

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

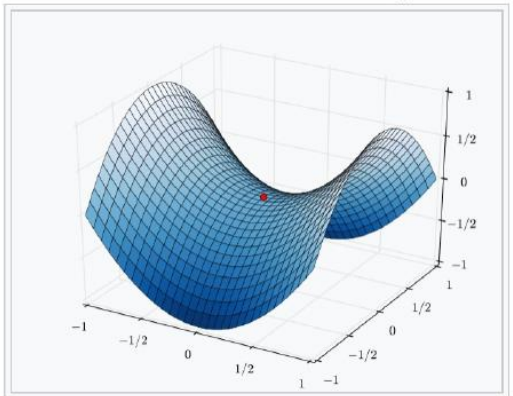
Demo - https://shenshen.mit.edu/demos/gd.html



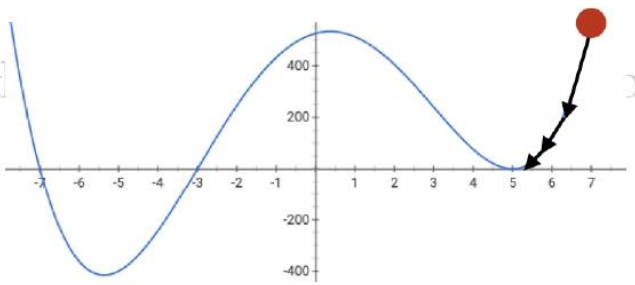
Gradient descent performance

- Assumptions:
  - f is sufficiently "smooth"
  - f has at least one global minimum
  - Run the algorithm sufficiently "long"
  - η is sufficiently small
  - f is convex
- Conclusion:
  - Gradient descent will return a parameter value wit or any chosen $\tilde{\epsilon} > 0$)

if violated, may get stuck at a saddle point



or a local minimum



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

Gradient of an ML objective

In general,

- An ML objective function is a finite sum

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

- The gradient of an ML objective :

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

👆

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

For instance,

- the MSE of a linear hypothesis:

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

- and its gradient w.r.t. θ:

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

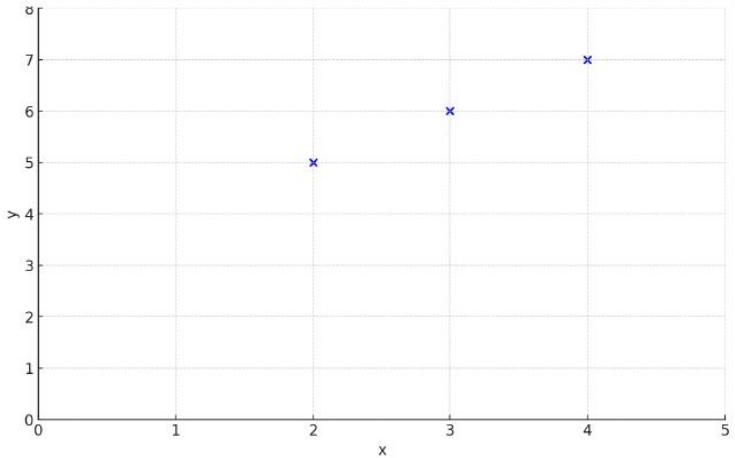
Concrete example

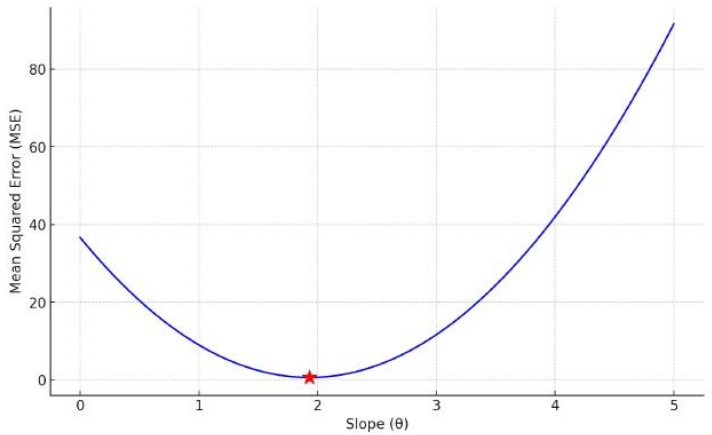
Three data points:

{(2,5), (3,6), (4,7)}

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

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





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

First data point's "pull"

Second data point 's "pull"

Third data point's "pull"

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

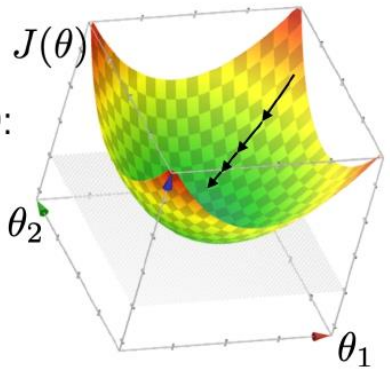
Stochastic gradient descent performance

- Assumptions:
  - f is sufficiently "smooth"
  - f has at least one global minimum
  - Run the algorithm long enough
  - η is sufficiently small and **satisfies additional "scheduling" condition**
  - f is convex

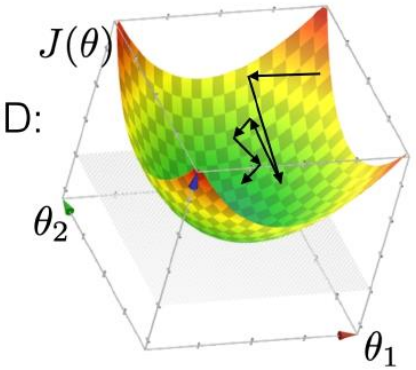
$$\sum_{t=1}^{\infty} \eta(t) = \infty \text{ and } \sum_{t=1}^{\infty} \eta(t)^2 < \infty$$

- Conclusion:
  - **Stochastic** gradient descent will return a parameter value within $\tilde{\epsilon}$ of a global minimum **with probability 1** (for any chosen $\tilde{\epsilon} > 0$)

- GD:

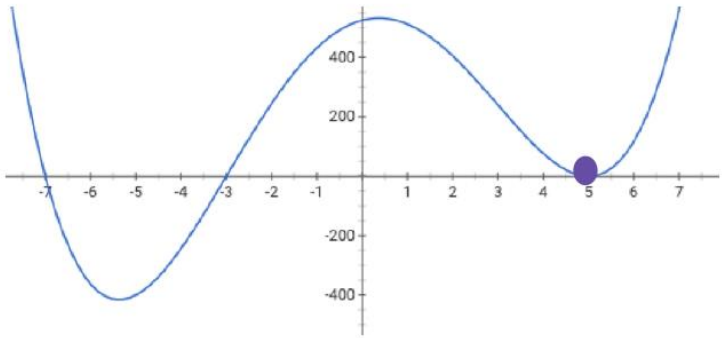


- SGD:



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.