https://introml.mit.edu/

6.390 Intro to Machine Learning

Lecture 3: Gradient Descent Methods

Shen Shen

Feb 16, 2024

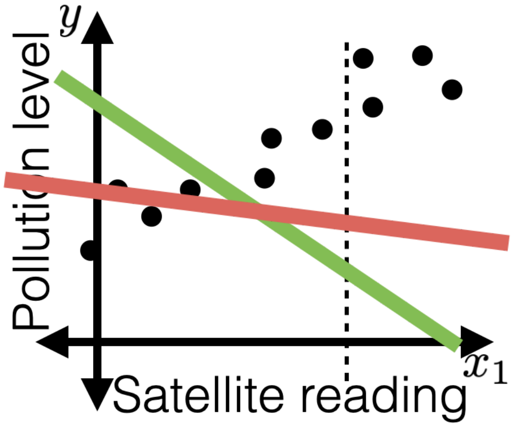
(many slides adapted from Tamara Broderick)

Outline

- Recall (Ridge regression) => Why care about GD
- Optimization primer
  - Gradient, optimality, convexity
- GD as an optimization algorithm for generic function
- GD as an optimization algorithm for ML applications
  - Loss function typically a finite sum
- Stochastic gradient descent (SGD) for ML applications
  - Pick one out of the finite sum

Recall

- A general ML approach
  - Collect data
  - Choose hypothesis class, hyperparameter, loss function
  - Train (optimize for) "good" hypothesis by minimizing loss. e.g. ridge regression
- Great when have analytical solutions
  - But don't always have them (recall, half-pipe)
  - Even when do have analytical solutions, can be expensive to compute (recall, lab2, Q2.8,)
- Want a more general, efficient way! => GD methods



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

$(\lambda > 0)$

Outline

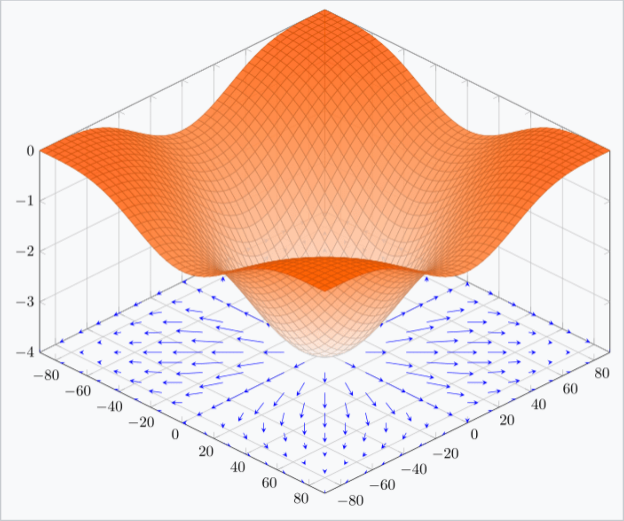
- Recall (Ridge regression) => Why care about GD
- Optimization primer
  - Gradient, optimality, convexity
- GD as an optimization algorithm for generic function
- GD as an optimization algorithm for ML applications
  - Loss function typically a finite sum
- Stochastic gradient descent (SGD) for ML applications
  - Pick one out of the finite sum

Gradient

- Def: 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

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

e.g.



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

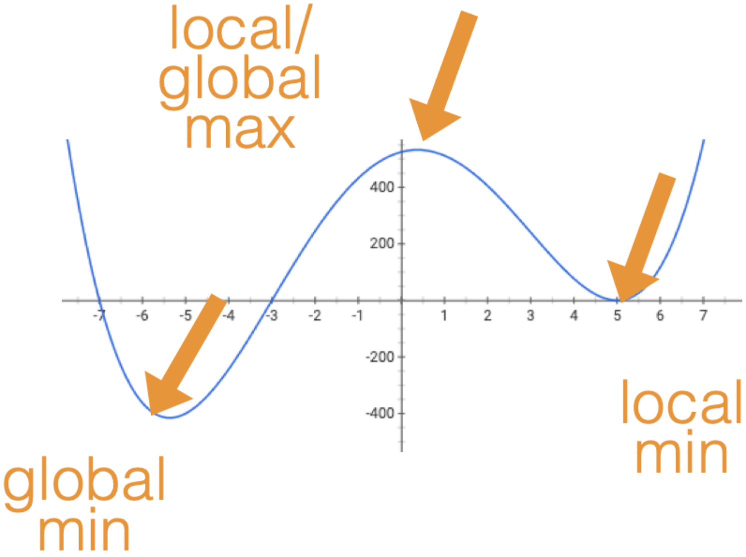
another example

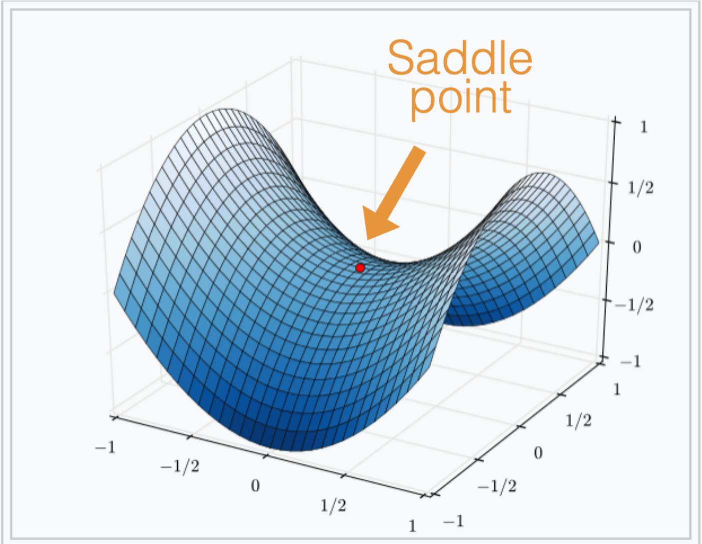
$$f(x, y, z) = x^2 + y^3 + z$$

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

When gradient is zero:

5 cases:





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

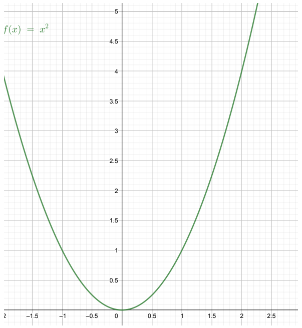
Convex Functions

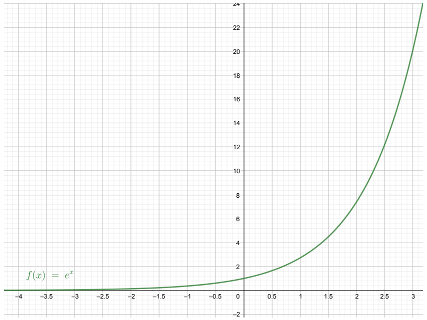
- A function f on $\mathbb{R}^m$ is **convex** if any line segment connecting two points of the graph of f lies above or on the graph.
- (f is concave if $-f$ is convex.)
- For convex functions, local minima are all global minima.

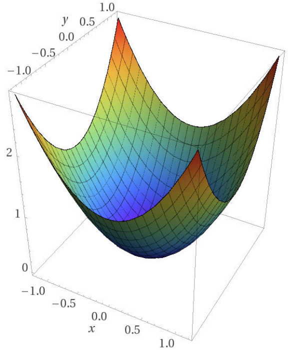
https://shenshen.mit.edu/demos/convex.html

Simple examples

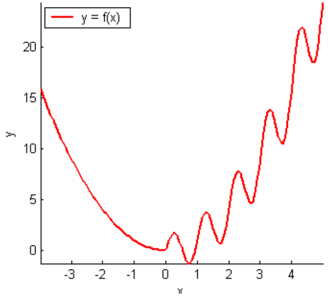
Convex functions

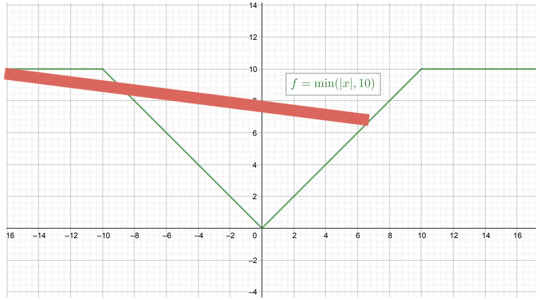


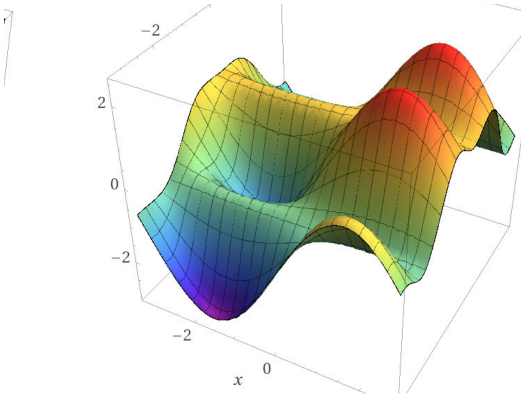




Non-convex functions







Convex Functions (cont'd)

What do we need to know:

- Intuitive understanding of the definition
- If given a function, can determine if it's convex or not. (We'll only ever give at most 2D, so visually is enough)
- Understand how (stochastic) gradient descent algorithms would behave differently depending on if convexity is satisfied.
- For this class, OLS loss function is convex, ridge regression loss is (strictly) convex, and later cross-entropy loss function is convex too.

Outline

- Recall (Ridge regression) => Why care about GD
- Optimization primer
  - Gradient, optimality, convexity
- GD as an optimization algorithm for generic function
- GD as an optimization algorithm for ML applications
  - Loss function typically a finite sum (over data)
- Stochastic gradient descent (SGD) for ML applications
  - Pick one data out of the finite sum

Gradient descent

hyperparameters

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

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

  Initialize t = 0

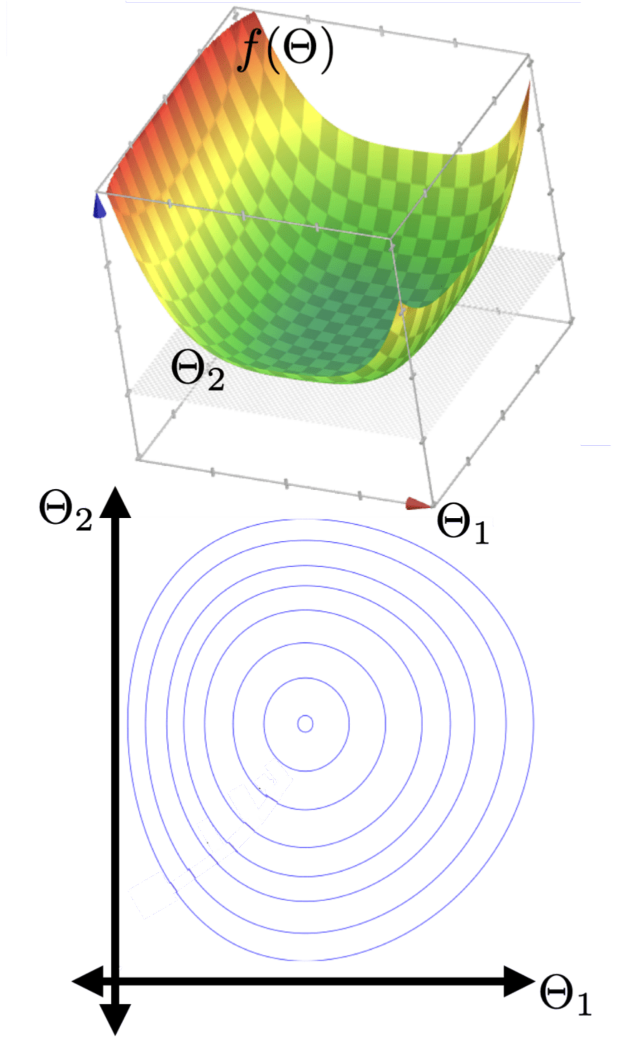
  repeat

    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

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

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

Initialize t = 0

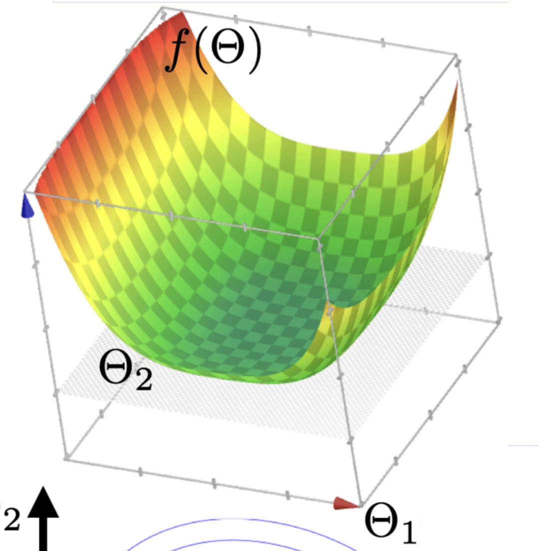
repeat

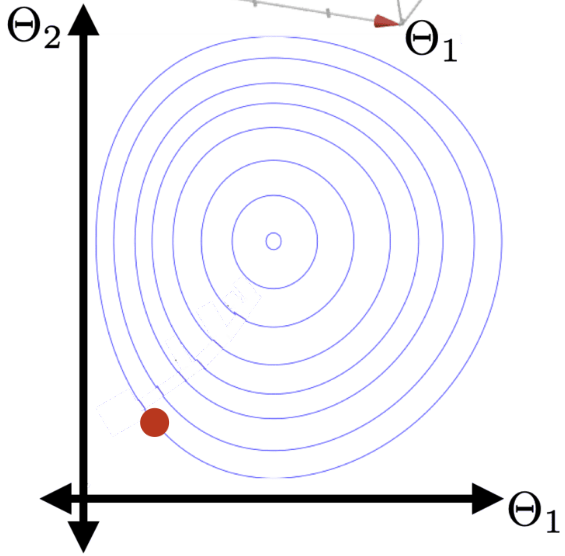
t = t + 1

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

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

Return $\Theta^{(t)}$





Gradient descent

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

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

  Initialize t = 0

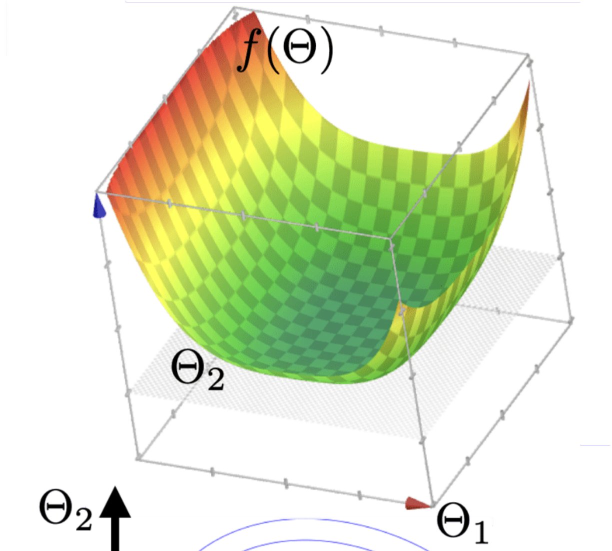
  repeat

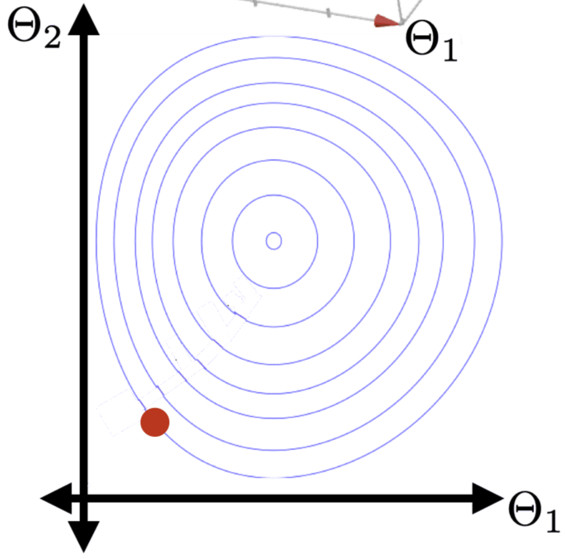
    t = t + 1

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

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

  Return $\Theta^{(t)}$





Gradient descent

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

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

Initialize t = 0

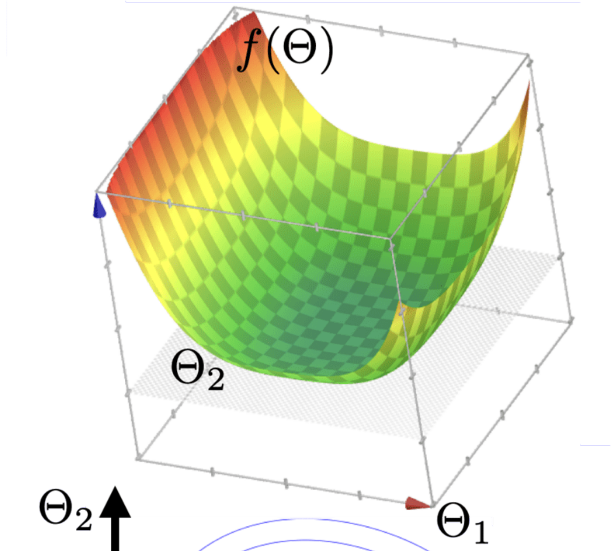
repeat

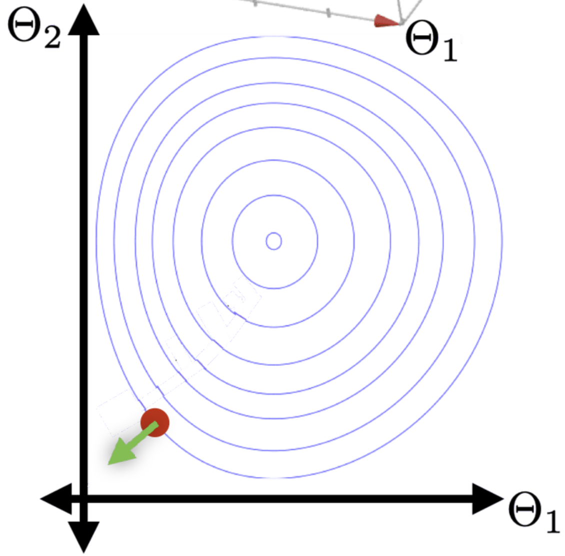
t = t + 1

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

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

Return $\Theta^{(t)}$





Gradient descent

```
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)
```

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

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

Initialize $t = 0$

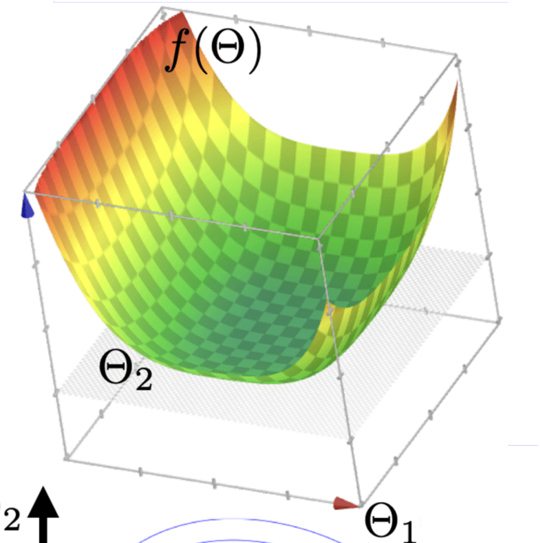
repeat

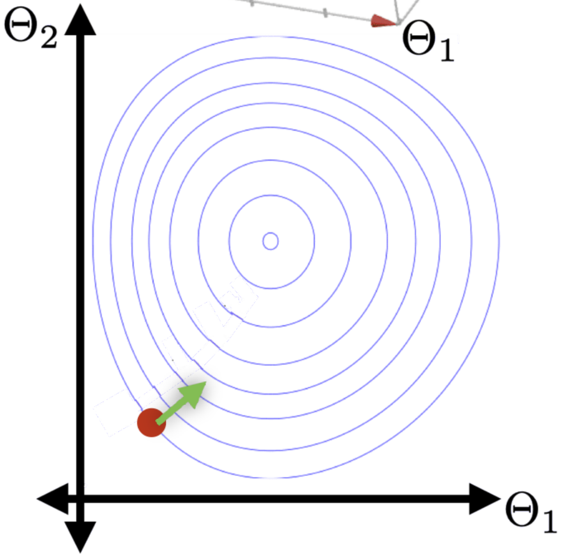
$t = t + 1$

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

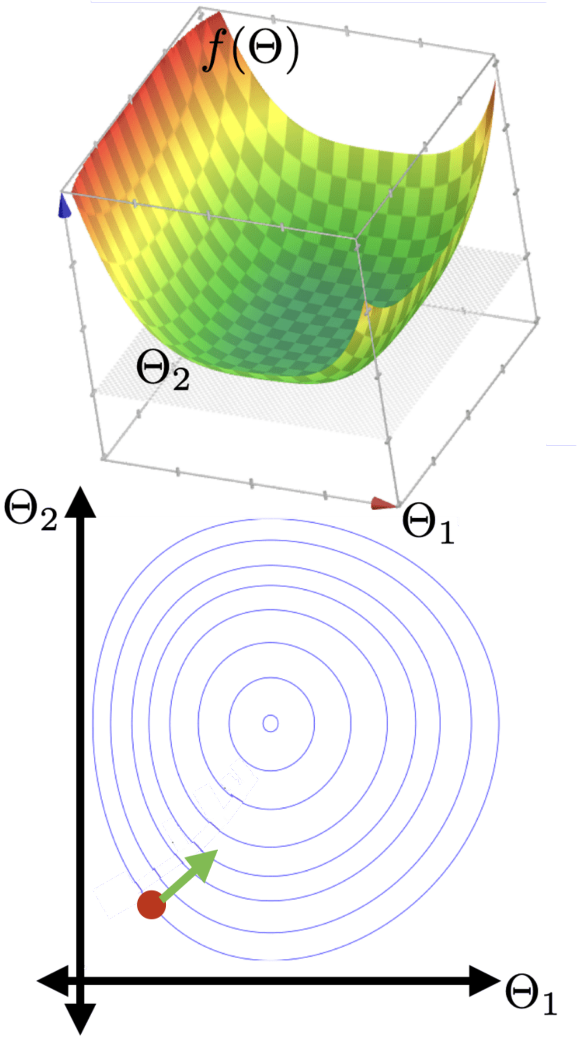
until $\left| f(\Theta^{(t)}) - f(\Theta^{(t-1)}) \right| < \epsilon$

Return $\Theta^{(t)}$





Gradient descent



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

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

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

Initialize t = 0

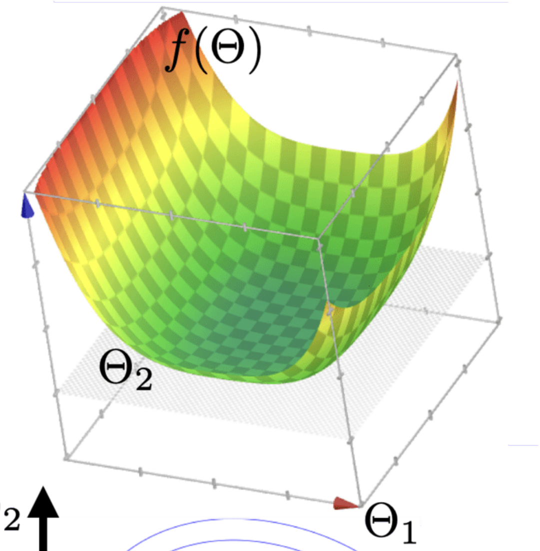
repeat

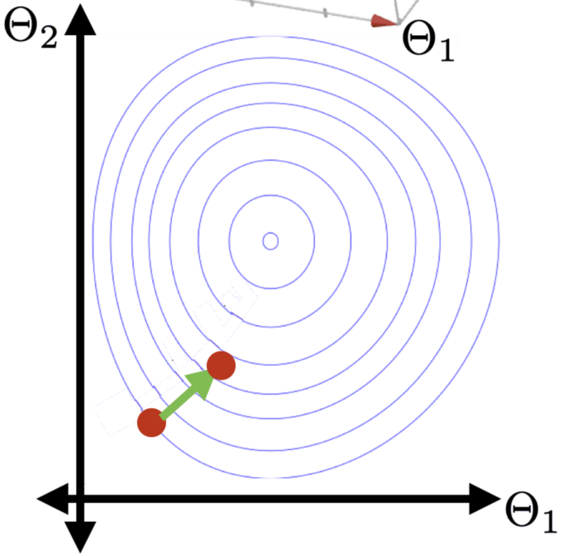
t = t + 1

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

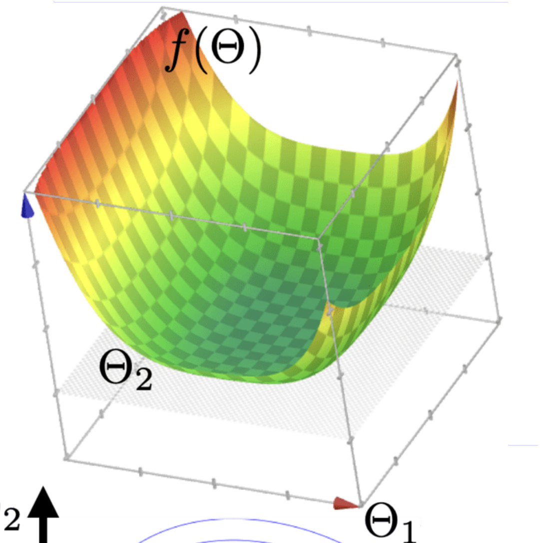
until $\left| f(\Theta^{(t)}) - f(\Theta^{(t-1)}) \right| < \epsilon$

Return $\Theta^{(t)}$





Gradient descent



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

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

  `Initialize t = 0`

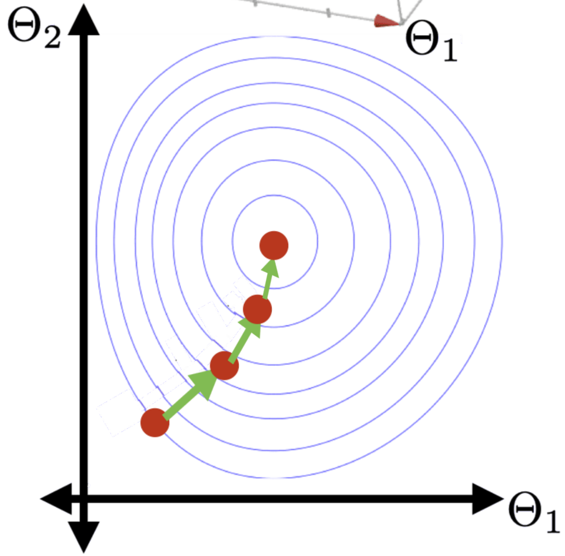
  repeat

    `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

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

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

Initialize $t = 0$

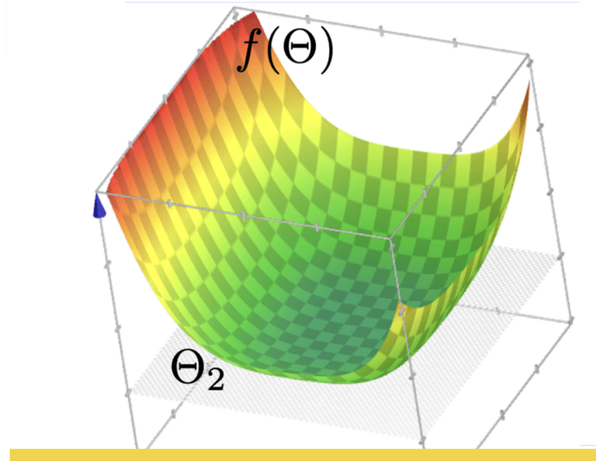
repeat

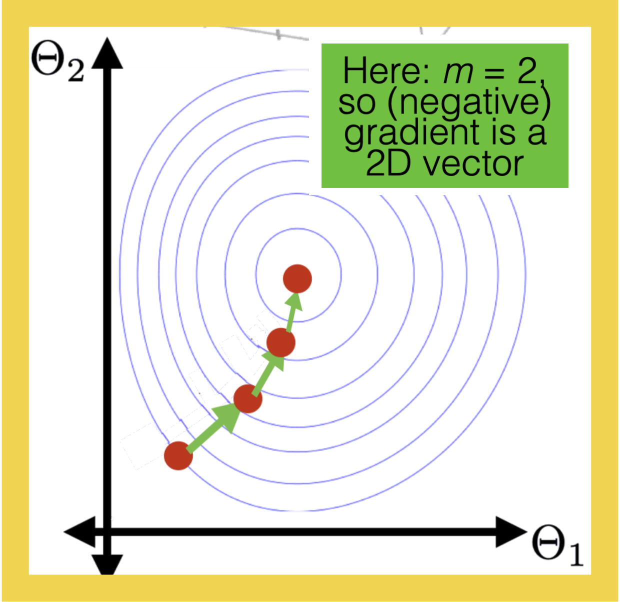
$t = t + 1$

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

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

Return $\Theta^{(t)}$





Gradient descent

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

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

Initialize t = 0

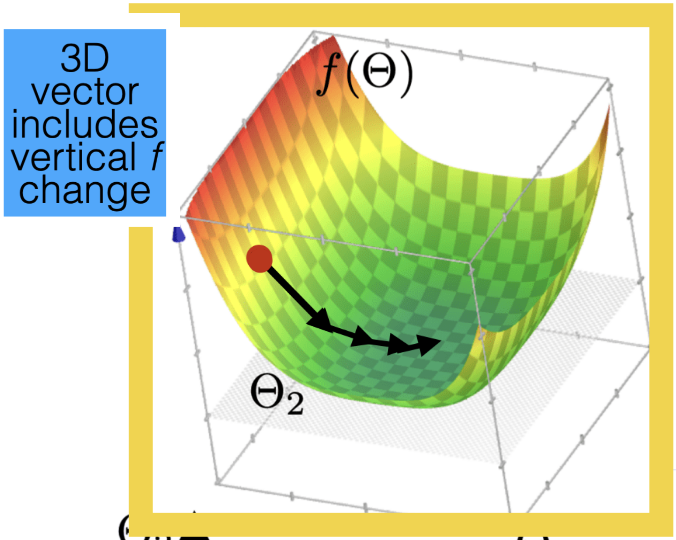
repeat

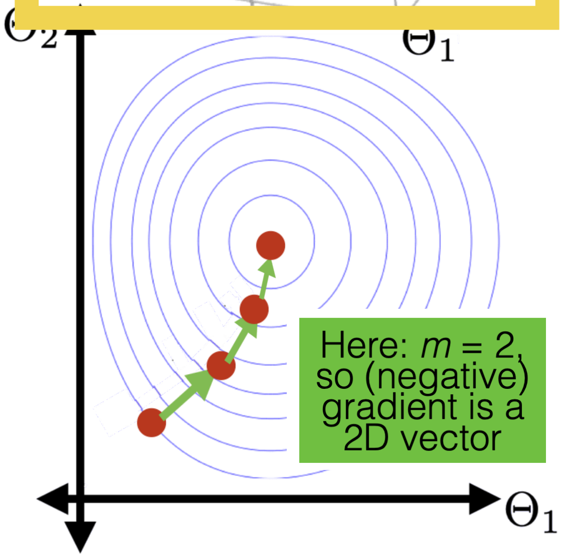
t = t + 1

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

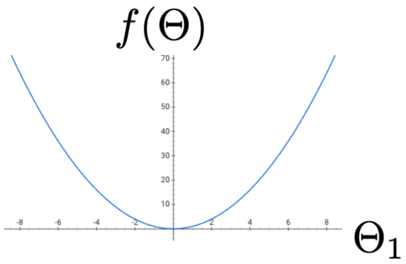
until $\left| f(\Theta^{(t)}) - f(\Theta^{(t-1)}) \right| < \epsilon$

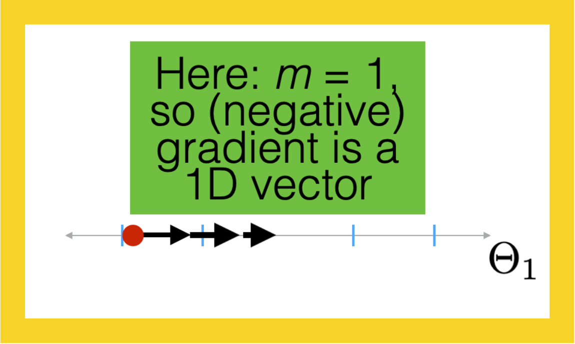
Return $\Theta^{(t)}$

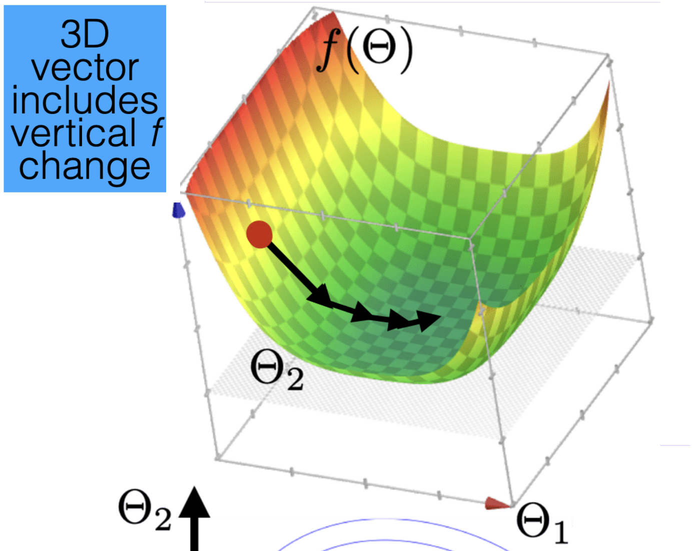


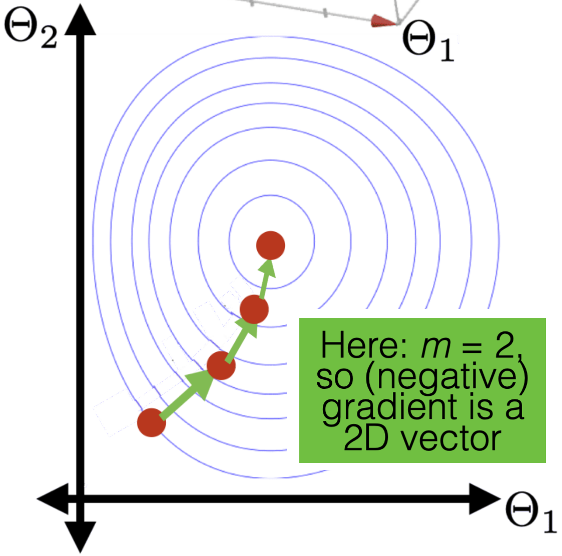


Gradient descent



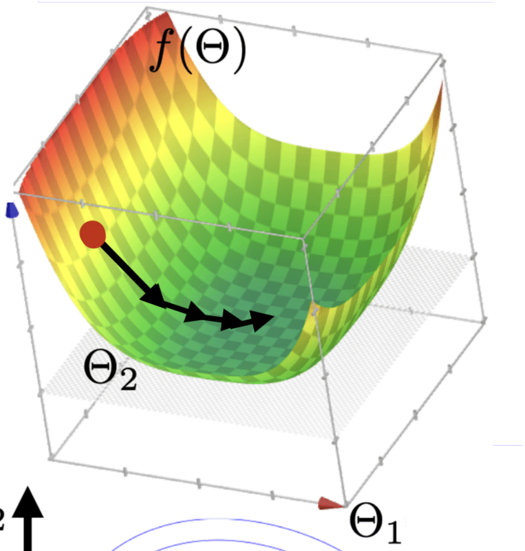


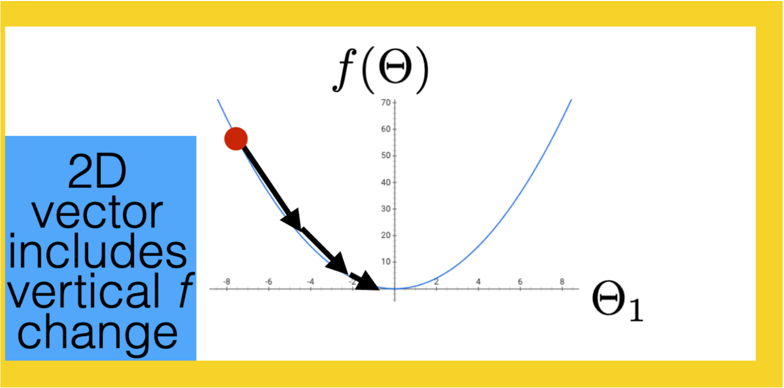


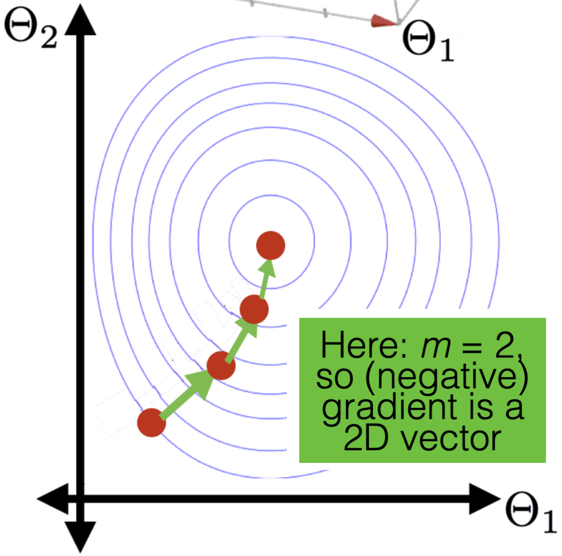


Gradient descent

3D vector includes vertical f change







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

Gradient descent

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

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

  Initialize t = 0

  repeat

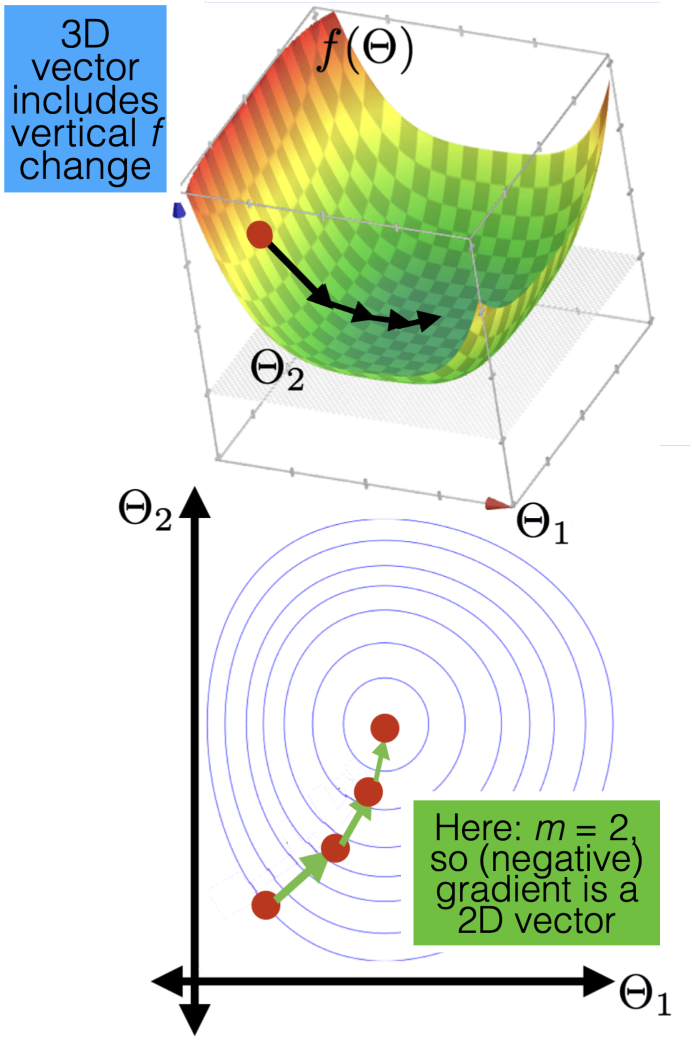
    t = t + 1

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

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

  Return $\Theta^{(t)}$

- Other possible stopping criteria:
  - Max number of iterations T
  - $\|\Theta^{(t)} - \Theta^{(t-1)}\| < \epsilon$
  - $\|\nabla_\Theta f(\Theta^{(t)})\| < \epsilon$



Gradient descent properties

Theorem: Gradient descent performance

- **Assumptions**: (Choose any $\tilde{\epsilon} > 0$)
  - *f* $\boxed{\text{is sufficiently “smooth”}}$ and convex
  - *f* has at least one global optimum
  - η is sufficiently small
- **Conclusion**: If run long enough, gradient descent will return a value within $\tilde{\epsilon}$ of a global optimum Θ

if violated:

can't run gradient descent

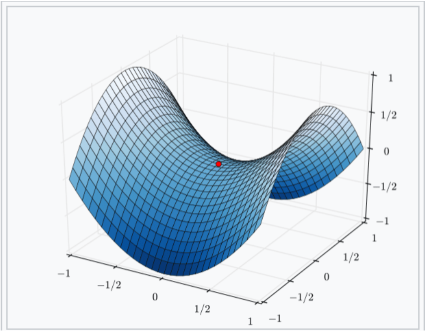
Gradient descent properties

Theorem: Gradient descent performance

- **Assumptions**: (Choose any $\tilde{\epsilon} > 0$)
  - f is sufficiently "smooth" and convex
  - f has at least one global optimum
  - η is sufficiently small
- **Conclusion**: If run long enough, gradient descent will return a value within $\tilde{\epsilon}$ of a global optimum Θ

if violated:

e.g. get stuck at a saddle point



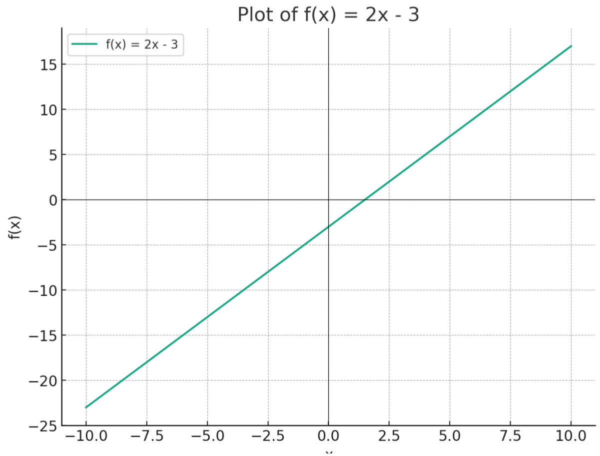
Gradient descent properties

Theorem: Gradient descent performance

- **Assumptions**: (Choose any $\tilde{\epsilon} > 0$)
  - f is sufficiently "smooth" and convex
  - f has at $\boxed{\text{least one global optimum}}$
  - η is sufficiently small
- **Conclusion**: If run long enough, gradient descent will return a value within $\tilde{\epsilon}$ of a global optimum Θ

if violated:

e.g. may not terminate



Gradient descent properties

Theorem: Gradient descent performance

- **Assumptions**: (Choose any $\tilde{\epsilon} > 0$)
  - *f* is sufficiently "smooth" and convex
  - *f* has at least one global optimum
  - η $\boxed{\text{is sufficiently small}}$
- **Conclusion**: $\boxed{\text{If run long enough,}}$ gradient descent will return a value within $\tilde{\epsilon}$ of a global optimum Θ

if violated:

see demo, and lab

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

Recall: need step-size sufficiently small
run long enough

Outline

- Recall (Ridge regression) => Why care about GD
- Optimization primer
  - Gradient, optimality, convexity
- GD as an optimization algorithm for generic function
- GD as an optimization algorithm for ML applications
  - Loss function typically a finite sum
- Stochastic gradient descent (SGD) for ML applications
  - Pick one out of the finite sum

Outline

- Recall (Ridge regression) => Why care about GD
- Optimization primer
  - Gradient, optimality, convexity
- GD as an optimization algorithm for generic function
- GD as an optimization algorithm for ML applications
  - Loss function typically a finite sum
- Stochastic gradient descent (SGD) for ML applications
  - Pick one out of the finite sum

Gradient descent on ML objective

- ML objective functions has typical form: finite sum

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

- Because (gradient of sum) = (sum of gradient), gradient of an ML objective :

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

- For instance, MSE we've seen so far:

$$\frac{1}{n}\sum_{i=1}^{n}(\theta^\top x^{(i)} + \theta_0 - y^{(i)})^2$$

- gradient of that MSE w.r.t. θ:

$$\frac{2}{n}\sum_{i=1}^{n}\left(\theta^\top x^{(i)} + \theta_0 - y^{(i)}\right) x^{(i)}$$

Outline

- Recall (Ridge regression) => Why care about GD
- Optimization primer
  - Gradient, optimality, convexity
- GD as an optimization algorithm for generic function
- GD as an optimization algorithm for ML applications
  - Loss function typically a finite sum
- Stochastic gradient descent (SGD) for ML applications
  - Pick one out of the finite sum

Stochastic gradient descent

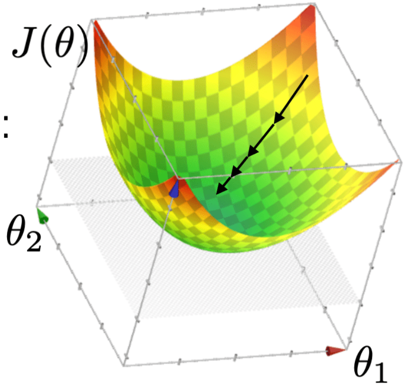
```
Gradient-Descent( Θ_init, η, f, ∇_Θ f, ε )
  Initialize  Θ^(0) = Θ_init
  Initialize t = 0
  repeat
    t = t + 1
    Θ^(t) = Θ^(t-1) - η ∇_Θ f(Θ^(t-1))
  until |f(Θ^(t)) - f(Θ^(t-1))| < ε
  Return  Θ^(t)
```

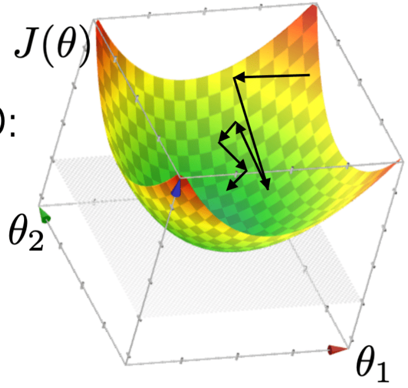
```
Stochastic-
Gradient-Descent( Θ_init, η, f, ∇_Θ f, ε )
  Initialize  Θ^(0) = Θ_init
  Initialize t = 0
  repeat
    t = t + 1
    randomly select i from {1,…,n}
    Θ^(t) = Θ^(t-1) - η(t) ∇_Θ f_i(Θ^(t-1))
  until |f(Θ^(t)) - f(Θ^(t-1))| < ε
  Return  Θ^(t)
```

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

for a randomly picked i

Stochastic gradient descent (SGD) properties

- GD: 

- SGD: 

More "random"

Theorem: SGD performance

- **Assumptions**: (Choose any $\tilde{\epsilon} > 0$)
  - *f* is "nice" & convex, has a unique global minimizer
  - $\sum_{t=1}^{\infty} \eta(t) = \infty, \sum_{t=1}^{\infty} (\eta(t))^2 < \infty$
    - e.g. $\eta(t) = \alpha(\tau_0 + t)^{-\kappa} (\kappa \in (0.5, 1])$
- **Conclusion**: If run long enough, stochastic gradient descent will return a value within $\tilde{\epsilon}$ of the global minimizer

More "demanding"

We'd love it for you to share some lecture feedback.

Thanks!