ORIE 4741: Learning with Big Messy Data

Linear Models and Linear Least Squares

Professor Udell
Operations Research and Information Engineering
Cornell

September 24, 2020
Announcements

▶ section this week: github tutorial + project ideas
▶ hw1 was due this morning, hw2 will be posted this weekend
▶ quiz 3 today 6:15pm - Friday 11:59pm; set a reminder!
▶ no collaboration is allowed on quizzes
▶ form project groups by this Sunday; submit at https://forms.gle/mFfRH18UqqZHuďXY7
▶ Zoom board game night Sunday 8pm (link on class calendar)

(All times ET)
Poll

How many Cornell students tested positive for COVID in the last week?

A. 0
B. 2
C. 6
D. 13
E. 27
Outline

Regression

Gradient descent

Least squares via gradient descent

Faster!

Proofs for GD

Least squares via normal equations
Supervised learning setup

- **input space** $\mathcal{X}$
  - $x \in \mathcal{X}$ is called the covariate, feature, or independent variable

- **output space** $\mathcal{Y}$
  - $y \in \mathcal{Y}$ is called the response, outcome, label, or dependent variable

- given $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$
  - $\mathcal{D}$ is called the data, examples, observations, samples or measurements

- we will find some $h \in \mathcal{H}$ so that (we hope!)

$$h(x_i) \approx y_i, \quad i = 1, \ldots, n$$
Supervised learning

different names for different $\mathcal{Y}$'s:

- **classification:** $\mathcal{Y} = \{-1, 1\}$
- **regression:** $\mathcal{Y} = \mathbb{R}$
- **multiclass classification:** $\mathcal{Y} = \{\text{car, pedestrian, bike}\}$
- **ordinal regression:** $\mathcal{Y} = \{\text{strongly disagree, \ldots, strongly agree}\}$
Regression

examples where $\mathcal{Y} = \mathbb{R}$:

- predict credit score of applicant
- predict temperature in Ithaca a year from today
- predict travel time at rush hour
- predict \# positive COVID cases at Cornell tomorrow
Regression

eamples where $\mathcal{Y} = \mathbb{R}$:

- predict credit score of applicant
- predict temperature in Ithaca a year from today
- predict travel time at rush hour
- predict $\#$ positive COVID cases at Cornell tomorrow

careful: are all real number valid predictions?
Linear model for regression

suppose $\mathcal{X} = \mathbb{R}^d$, $\mathcal{Y} = \mathbb{R}$

- predict $y$ using a linear function $h : \mathbb{R}^d \rightarrow \mathbb{R}$

  $h(x) = w^\top x$

- we want $h(x_i) \approx y_i$ for every $i = 1, \ldots, n$
suppose $\mathcal{X} = \text{anything}, \mathcal{Y} = \mathbb{R}$

- pick a transformation $\phi : \mathcal{X} \to \mathbb{R}^d$
- predict $y$ using a linear function of $\phi(x)$
  \[
  h(x) = w^\top \phi(x)
  \]
- we want $h(x_i) \approx y_i$ for every $i = 1, \ldots, n$
suppose $\mathcal{X} = \text{anything}, \mathcal{Y} = \mathbb{R}$

- pick a transformation $\phi : \mathcal{X} \rightarrow \mathbb{R}^d$
- predict $y$ using a linear function of $\phi(x)$

$$h(x) = w^\top \phi(x)$$

- we want $h(x_i) \approx y_i$ for every $i = 1, \ldots, n$

choices:

- how to pick $\phi$?
- how to pick $w$?
Linear model++

suppose $\mathcal{X} = \text{anything}$, $\mathcal{Y} = \mathbb{R}$

- pick a transformation $\phi : \mathcal{X} \rightarrow \mathbb{R}^d$
- predict $y$ using a linear function of $\phi(x)$

$$h(x) = w^\top \phi(x)$$

- we want $h(x_i) \approx y_i$ for every $i = 1, \ldots, n$

choices:

- how to pick $\phi$?
- how to pick $w$?

for now, assume $d$ and $\phi$ are fixed; we’ll return to these later...
Least squares fitting

- define **prediction error** or **residual**
  \[ r_i = y_i - h(x_i), \quad i = 1, \ldots, n \]

- choose \( w \) to minimize **sum of square residuals**
  \[ \sum_{i=1}^{n} (r_i)^2 = \sum_{i=1}^{n} (y_i - h(x_i))^2 = \sum_{i=1}^{n} (y_i - w^\top x_i)^2 \]
Poll

Why minimize the sum of square residuals?

A. the sum of square residuals is what I truly care about when predicting ≠ positive COVID cases
B. because it's easy to find the $w$ that minimizes it
Least squares fitting

rewrite using linear algebra:

- form vector $y \in \mathbb{R}^n$: each outcome $y_i$ is an entry of $y$
- form matrix $X \in \mathbb{R}^{n \times d}$: each example $x_i$ is a row of $X$
- rewrite error:

$$\sum_{i=1}^{n} (r_i)^2 = \sum_{i=1}^{n} (y_i - w^\top x_i)^2 = ||y - Xw||^2$$

interpretation:

- $Xw$ is a linear combination of the columns of $X$
- we seek the linear combination that best matches $y$
Evaluating least squares: computational complexity

Real numbers are generally represented as floating point numbers on a computer.

**Definition**

A **floating point operation** (flop) adds, multiplies, subtracts, or divides two floating point numbers.

**example:** to check objective value of $w$

$$\|y - Xw\|^2$$

requires 2nd flops
Poll

How many flops to compute $3 \times 2 + 4 \times 6$?

A. 2
B. 3
C. 4
D. 5
How many flops to compute $u^T v$, where $u = [3, 4]$ and $v = [2, 6]$?

A. 2  
B. 3  
C. 4  
D. 5
Poll

How many flops to compute $u^T v$, where $u, v \in \mathbb{R}^d$?

A. $d-1$
B. $d$
C. $d+1$
D. $2d-1$
E. $2d$
How many flops to compute \( Xw \), where \( X \in \mathbb{R}^{n \times d} \), \( w \in \mathbb{R}^d \)?

A. \( n+2d-1 \)
B. \( 2n+2d-1 \)
C. \( 2nd \)
D. \( n(2d-1) \)
E. \( 2n(2d-1) \)
How many flops to compute $y - z$, where $y, z \in \mathbb{R}^n$?

A.  $n-1$
B.  $n$
C.  $n+1$
D.  $2n-1$
E.  $2n$
Poll

How many flops to compute $\|y\|^2$, where $y \in \mathbb{R}^n$?

A. $n-1$
B. $n$
C. $n+1$
D. $2n-1$
E. $2n$
How many flops to compute $\|y\|^2$, where $y \in \mathbb{R}^n$?

A. $n-1$
B. $n$
C. $n+1$
D. $2n-1$
E. $2n$

note $\|y\|^2 = y^T y$
Add it up!

To compute $\|y - Xw\|^2$, 
Add it up!

To compute \( \| y - Xw \|^2 \),

\[ n(2d - 1) = \mathcal{O}(nd) \text{ flops to compute } Xw \]
Add it up!

To compute $\|y - Xw\|^2$,

- $n(2d - 1) = \mathcal{O}(nd)$ flops to compute $Xw$
- $n = \mathcal{O}(n)$ flops to compute $y - Xw$
Add it up!

To compute $\|y - Xw\|^2$,

- $n(2d - 1) = O(nd)$ flops to compute $Xw$
- $n = O(n)$ flops to compute $y - Xw$
- $2n - 1 = O(n)$ flops to compute $\|y - Xw\|^2$
Add it up!

To compute $\|y - Xw\|^2$,

- $n(2d - 1) = \mathcal{O}(nd)$ flops to compute $Xw$
- $n = \mathcal{O}(n)$ flops to compute $y - Xw$
- $2n - 1 = \mathcal{O}(n)$ flops to compute $\|y - Xw\|^2$

$$= 2nd - n + n + 2n - 1 = 2nd + 2n - 1 = \mathcal{O}(nd)$$
Outline

Regression

Gradient descent

Least squares via gradient descent

Faster!

Proofs for GD

Least squares via normal equations
Optimization

in this lecture, we will see two methods to solve the problem

$$\text{minimize } f(w)$$

with $w \in \mathbb{R}^n$ when $f$ is \textbf{differentiable}

1. gradient descent
2. solve normal equations

when $f$ is convex, both methods provably find the solution
Optimization

in this lecture, we will see two methods to solve the problem

\[ \text{minimize } f(w) \]

with \( w \in \mathbb{R}^n \) when \( f \) is \textbf{differentiable}

1. gradient descent
2. solve normal equations

when \( f \) is convex, both methods provably find the solution

\textbf{example:} for least squares, \( f(w) = \|y - Xw\|^2 \)
The gradient

the gradient $\nabla f(w)$ generalizes the derivative.

**Definition**

for $w \in \mathbb{R}^d$, $f : \mathbb{R}^d \to \mathbb{R}$ differentiable,

$$\nabla f(w) = \left( \frac{\partial f}{\partial w_1}, \ldots, \frac{\partial f}{\partial w_d} \right) \in \mathbb{R}^d$$
The gradient

the **gradient** $\nabla f(w)$ generalizes the derivative.

**Definition**

for $w \in \mathbb{R}^d$, $f : \mathbb{R}^d \to \mathbb{R}$ differentiable,

$$\nabla f(w) = \left( \frac{\partial f}{\partial w_1}, \ldots, \frac{\partial f}{\partial w_d} \right) \in \mathbb{R}^d$$

allows easy computation of directional derivatives:

for fixed $v \in \mathbb{R}^d$, let $w^+(\alpha) = w + \alpha v$. then

$$\frac{d}{d\alpha} f(w^+(\alpha)) = \frac{\partial f}{\partial w_1^+} \frac{dw_1^+}{d\alpha} + \cdots + \frac{\partial f}{\partial w_d^+} \frac{dw_d^+}{d\alpha}$$

$$= (\nabla f(w))^\top v$$
The gradient

the **gradient** $\nabla f(w)$ generalizes the derivative.

**Definition**

for $w \in \mathbb{R}^d$, $f : \mathbb{R}^d \to \mathbb{R}$ differentiable,

$$\nabla f(w) = \left( \frac{\partial f}{\partial w_1}, \ldots, \frac{\partial f}{\partial w_d} \right) \in \mathbb{R}^d$$

- allows easy computation of directional derivatives:
  for fixed $v \in \mathbb{R}^d$, let $w^+(\alpha) = w + \alpha v$. then
  $$\frac{d}{d\alpha} f(w^+(\alpha)) = \frac{\partial f}{\partial w_1^+} \frac{dw_1^+}{d\alpha} + \cdots + \frac{\partial f}{\partial w_d^+} \frac{dw_d^+}{d\alpha}$$
  $$= (\nabla f(w))^\top v$$

- locally approximates $f(w)$:
  $$f(w + \alpha v) \approx f(w) + \alpha (\nabla f(w))^\top v$$
The gradient

\[ f(w + \alpha v) \approx f(w) + \alpha (\nabla f(w))^\top v \]

**Q:** From the point \( w \), which direction \( v \) should we travel in to make \( f(w) \) **increase** as fast as possible?
The gradient

\[ f(w + \alpha v) \approx f(w) + \alpha (\nabla f(w))^\top v \]

**Q:** From the point \( w \), which direction \( v \) should we travel in to make \( f(w) \) **increase** as fast as possible?

**A:** In the direction \( v = \nabla f(w) \), to maximize \( (\nabla f(w))^\top v \)
The gradient

\[ f(w + \alpha v) \approx f(w) + \alpha (\nabla f(w))^\top v \]

**Q:** From the point \( w \), which direction \( v \) should we travel in to make \( f(w) \) **increase** as fast as possible?

**A:** In the direction \( v = \nabla f(w) \), to maximize \( (\nabla f(w))^\top v \)

**Q:** From the point \( w \), which direction \( v \) should we travel in to make \( f(w) \) **decrease** as fast as possible?
The gradient

\[ f(w + \alpha v) \approx f(w) + \alpha (\nabla f(w))^\top v \]

**Q:** From the point \( w \), which direction \( v \) should we travel in to make \( f(w) \) **increase** as fast as possible?  
**A:** In the direction \( v = \nabla f(w) \), to maximize \( (\nabla f(w))^\top v \)

**Q:** From the point \( w \), which direction \( v \) should we travel in to make \( f(w) \) **decrease** as fast as possible?  
**A:** In the direction \( v = -\nabla f(w) \)
Demo: gradient descent

let’s verify these properties of gradients numerically
https://github.com/ORIE4741/demos/blob/master/Gradient%20descent.ipynb
Gradient descent

\[
\text{minimize } f(w)
\]

idea: go downhill to get to a (the?) minimum!

---

**Algorithm** Gradient descent

**Given:** \( f : \mathbb{R}^d \rightarrow \mathbb{R} \), stepsize \( \alpha \), maxiters

**Initialize:** \( w = 0 \) (or anything you’d like)

**For:** \( k = 1, \ldots, \text{maxiters} \)

- update \( w \):

\[
w \leftarrow w - \alpha \nabla f(w)
\]
Gradient descent

minimize $f(w)$

Algorithm Gradient descent

Given: $f : \mathbb{R}^d \rightarrow \mathbb{R}$, maxiters
Initialize: $w = 0$ (or anything you’d like)
For: $k = 1, \ldots, \text{maxiters}$
  ▶ choose stepsize $\alpha^{(k)}$
  ▶ update $w$:

$$w^{(k)} = w^{(k-1)} - \alpha^{(k)} \nabla f(w^{(k-1)})$$

nomenclature

▶ $w^{(k)} \in \mathbb{R}^d$ are called iterates
▶ $\alpha^{(k)} \in \mathbb{R}$ are called step-sizes
Gradient descent: choosing a step-size

- **constant step-size.** \( \alpha^{(k)} = \alpha \) (constant)
- **decreasing step-size.** \( \alpha^{(k)} = 1/k \)
- **line search.** try different possibilities for \( \alpha^{(k)} \) until objective at new iterate

\[
f(w^{(k)}) = f(w^{(k-1)} - \alpha^{(k)} \nabla f(w^{(k-1)}))
\]
decreases enough.

tradeoff: evaluating \( f(w) \) takes \( \mathcal{O}(nd) \) flops each time . . .
Line search

define $w^+ = w - \alpha \nabla f(w)$

- exact line search: find $\alpha$ to minimize $f(w^+)$

- the Armijo rule requires $\alpha$ to satisfy

$$f(w^+) \leq f(w) - c\alpha \|\nabla f(w)\|^2$$

for some $c \in (0,1)$, e.g., $c = .01$. 
Line search

define \( w^+ = w - \alpha \nabla f(w) \)

- exact line search: find \( \alpha \) to minimize \( f(w^+) \)
- the Armijo rule requires \( \alpha \) to satisfy
  \[
  f(w^+) \leq f(w) - c\alpha \|\nabla f(w)\|^2
  \]
  for some \( c \in (0, 1) \), e.g., \( c = .01 \).

A simple backtracking line search algorithm:

- set \( \alpha = 1 \)
- if step decreases objective value sufficiently, accept \( w^+ \):
  \[
  f(w^+) \leq f(w) - c\alpha \|\nabla f(w)\|^2 \quad \Rightarrow \quad w \leftarrow w^+
  \]
  otherwise, halve the stepsize \( \alpha \leftarrow \alpha/2 \) and try again
Line search

define $w^+ = w - \alpha \nabla f(w)$

- exact line search: find $\alpha$ to minimize $f(w^+)$
- the Armijo rule requires $\alpha$ to satisfy

$$f(w^+) \leq f(w) - c\alpha \|\nabla f(w)\|^2$$

for some $c \in (0, 1)$, e.g., $c = .01$.

a simple backtracking line search algorithm:

- set $\alpha = 1$
- if step decreases objective value sufficiently, accept $w^+$:

$$f(w^+) \leq f(w) - c\alpha \|\nabla f(w)\|^2 \implies w \leftarrow w^+$$

otherwise, halve the stepsize $\alpha \leftarrow \alpha/2$ and try again

Q: can we always satisfy the Armijo rule for some $\alpha$?
Line search

define $w^+ = w - \alpha \nabla f(w)$

- exact line search: find $\alpha$ to minimize $f(w^+)$
- the Armijo rule requires $\alpha$ to satisfy

$$f(w^+) \leq f(w) - c\alpha \| \nabla f(w) \|^2$$

for some $c \in (0, 1)$, e.g., $c = .01$.

A simple backtracking line search algorithm:

- set $\alpha = 1$
- if step decreases objective value sufficiently, accept $w^+$:

$$f(w^+) \leq f(w) - c\alpha \| \nabla f(w) \|^2 \quad \Rightarrow \quad w \leftarrow w^+$$

otherwise, halve the stepsize $\alpha \leftarrow \alpha/2$ and try again

Q: can we always satisfy the Armijo rule for some $\alpha$?

A: yes! see gradient descent demo
Outline

Regression

Gradient descent

Least squares via gradient descent

Faster!

Proofs for GD

Least squares via normal equations
two useful identities: let $w, b \in \mathbb{R}^d$, $A \in \mathbb{R}^{d \times d}$ symmetric

1. let $f(w) = w^\top b$. Then

$$\nabla f(w) = b$$

2. let $f(w) = w^\top Aw$. Then

$$\nabla f(w) = 2Aw$$

verify:

- take partial derivatives wrt each entry of $w$
- concatenate to get the matrix calculus result
Gradient of the least squares problem

\[ f(w) = \sum_{i=1}^{n} (y_i - w^T x_i)^2 \]

compute \( \nabla f(w) \):

\[ \nabla f(w) = \sum_{i=1}^{n} \nabla (y_i - w^T x_i)^2 \]
\[ = \sum_{i=1}^{n} -2(y_i - w^T x_i)x_i \]
Gradient of the least squares problem (matrix version)

\[ f(w) = \| y - Xw \|^2 \]

compute \( \nabla f(w) \):

\[
\begin{align*}
\nabla f(w) &= \nabla (y - Xw)^\top (y - Xw) \\
&= \nabla (y^\top y - w^\top X^\top y - y^\top Xw + w^\top X^\top Xw) \\
&= -\nabla (w^\top X^\top y + w^\top X^\top y) + \nabla (w^\top X^\top Xw) \\
&= -2X^\top y + 2X^\top Xw
\end{align*}
\]
Solving the least squares problem: gradient descent

minimize \( \|y - Xw\|^2 \)

**Algorithm**  Gradient descent for least squares

**Given:** \( X : \mathbb{R}^{n \times d}, y \in \mathbb{R}^n \), stepsize \( \alpha \), maxiters

**Initialize:** \( w = 0 \) (or anything you’d like)

**For:** \( k = 1, \ldots, \) maxiters

- update \( w \):
  \[
  w \leftarrow w + 2\alpha(X^T y - X^T Xw)
  \]
Gradient descent update:

\[ w \leftarrow w + 2\alpha(X^\top y - X^\top Xw) \]

How many flops does gradient descent require per iteration, as a function of the number of examples \( n \) and number of features \( d \)?

A. \( O(d) \)
B. \( O(n) \)
C. \( O(nd) \)
D. \( O(nd^2) \)
E. \( O(n^2d^2) \)
Gradient descent update:

\[ w \leftarrow w + 2\alpha (X^\top y - X^\top Xw) \]

How many flops does gradient descent require per iteration, as a function of the number of examples \( n \) and number of features \( d \)?

A. \( O(d) \)
B. \( O(n) \)
C. \( O(nd) \)
D. \( O(nd^2) \)
E. \( O(n^2d^2) \)

compute it as \( w + 2\alpha (X^\top y - X^\top (Xw)) \)
Demo: gradient descent for least squares

https://github.com/ORIE4741/demos/blob/master/Gradient%20descent.ipynb
Outline

Regression

Gradient descent

Least squares via gradient descent

Faster!

Proofs for GD

Least squares via normal equations
Speeding up gradient descent when \( n \gg d \)

\[
w^{+} = w + 2\alpha (X^\top y - X^\top Xw)
\]

to compute this quickly when \( n \gg d \):
Speeding up gradient descent when \( n \gg d \)

\[
w^+ = w + 2\alpha(X^\top y - X^\top Xw)
\]

to compute this quickly when \( n \gg d \):

- form **Gram matrix** \( G = X^\top X = \sum_{i=1}^n x_i x_i^\top \) (\( 2nd^2 \) flops)
Speeding up gradient descent when $n \gg d$

$$w^+ = w + 2\alpha(X^Ty - X^TXw)$$

to compute this quickly when $n \gg d$:

- Form **Gram matrix** $G = X^TX = \sum_{i=1}^{n} x_ix_i^\top$ (2nd$^2$ flops)
- Form $b = X^Ty = \sum_{i=1}^{n} y_ix_i$ (2nd flops)
Speeding up gradient descent when $n \gg d$

$$w^+ = w + 2\alpha(X^\top y - X^\top Xw)$$

to compute this quickly when $n \gg d$:

- form **Gram matrix** $G = X^\top X = \sum_{i=1}^{n} x_i x_i^\top$ (2nd$^2$ flops)
- form $b = X^\top y = \sum_{i=1}^{n} y_i x_i$ (2nd flops)
- for $k = 1, \ldots$
  - update $w^+ = w - 2\alpha(Gw - b)$ (2d$^2 + 3d$ flops)

$O(nd^2)$ flops to start, plus $O(d^2)$ per iteration
Parallel computation

- flops/core is constant over the last decade
- clock speed is roughly 1GHz: $10^9$ cycles per second
- processors do 2–32 flops per cycle
- cores/$\$ and cores/computer are still increasing
- your laptop: 4–16 cores
- my server: 80 cores
- NVIDIA GPUs: 1000s of cores

Q: Can we use parallelism to speed up gradient descent?
Parallel computation

- flops/core is constant over the last decade
  - clock speed is roughly 1GHz: $10^9$ cycles per second
  - processors do 2–32 flops per cycle
Parallel computation

- flops/core is constant over the last decade
  - clock speed is roughly 1GHz: $10^9$ cycles per second
  - processors do 2–32 flops per cycle
- cores/$ and cores/computer are still increasing
  - your laptop: 4–16 cores
  - my server: 80 cores
  - NVIDIA GPUs: 1000s of cores

Q: Can we use parallelism to speed up gradient descent?
Parallel computation

- flops/core is constant over the last decade
  - clock speed is roughly 1GHz: $10^9$ cyles per second
  - processors do 2–32 flops per cycle
- cores/$ and cores/computer are still increasing
  - your laptop: 4–16 cores
  - my server: 80 cores
  - NVIDIA GPUs: 1000s of cores

Q: Can we use parallelism to speed up gradient descent?
Parallelism: gradient descent

\[ w^+ = w + 2\alpha(X^\top y - X^\top Xw) \]

suppose we have \( P \) processors. let \( \{\mathcal{N}_j\}_{j=1}^P \) partition \( \{1, \ldots, n\} \).
Parallelism: gradient descent

\[ w^+ = w + 2\alpha(X^Ty - X^TXw) \]

suppose we have \( P \) processors. let \( \{\mathcal{N}_j\}_{j=1}^P \) partition \( \{1, \ldots, n\} \).

- form the **Gram matrix** \( G = X^TX = \sum_{p=1}^{P} (\sum_{i \in \mathcal{N}_p} x_i x_i^T) \)
  
  \( (2nd^2/P \) flops per proc)
Parallelism: gradient descent

\[ w^+ = w + 2\alpha(X^T y - X^T Xw) \]

suppose we have \( P \) processors. let \( \{N_j\}_{j=1}^P \) partition \( \{1, \ldots, n\} \).

- form the **Gram matrix** \( G = X^T X = \sum_{p=1}^P (\sum_{i \in N_p} x_i x_i^T) \)
  
  \( (2nd^2/P \text{ flops per proc}) \)

- form \( b = X^T y = \sum_{p=1}^P (\sum_{i \in N_p} y_i x_i) \)
  
  \( (2nd/P \text{ flops per proc}) \)
Parallelism: gradient descent

\[ w^+ = w + 2\alpha(X^Ty - X^TXw) \]

suppose we have \( P \) processors. let \( \{N_j\}_{j=1}^P \) partition \( \{1, \ldots, n\} \).

▶ form the **Gram matrix** \( G = X^TX = \sum_{p=1}^P (\sum_{i \in N_p} x_i x_i^\top) \)

\( (2nd^2/P \text{ flops per proc}) \)

▶ form \( b = X^Ty = \sum_{p=1}^P (\sum_{i \in N_p} y_i x_i) \)

\( (2nd/P \text{ flops per proc}) \)

▶ for \( k = 1, \ldots \)

▶ update \( w^+ = w - 2\alpha(Gw - b) \)

\( (2d^2 + 3d \text{ flops}) \)

\( O(nd^2) \) flops per proc to start, plus \( O(d^2) \) per iteration
Stochastic gradients?

- computing the gradient is slow
- idea: approximate the gradient!

A stochastic gradient $\tilde{\nabla} f(w)$ is a random variable with

$$\mathbb{E}\tilde{\nabla} f(w) = \nabla f(w)$$
Stochastic gradient: examples

stochastic gradient obeys $\mathbb{E} \tilde{\nabla} f(w) = \nabla f(w)$

**Examples:** for $f(w) = \sum_{i=1}^{n} (y_i - w^T x_i)^2$, 
Stochastic gradient: examples

stochastic gradient obeys \( \mathbb{E} \tilde{\nabla} f(w) = \nabla f(w) \)

**examples:** for \( f(w) = \sum_{i=1}^{n} (y_i - w^T x_i)^2 \),

- **single stochastic gradient.** pick a random example \( i \). set
  \[
  \tilde{\nabla} f(w) = n \nabla (y_i - w^T x_i)^2 = -2n(y_i - w^T x_i)x_i
  \]
Stochastic gradient: examples

stochastic gradient obeys $\mathbb{E} \tilde{\nabla} f(w) = \nabla f(w)$

**examples:** for $f(w) = \sum_{i=1}^{n} (y_i - w^T x_i)^2$,

- **single stochastic gradient.** pick a random example $i$. set

  $$\tilde{\nabla} f(w) = n \nabla (y_i - w^T x_i)^2 = -2n(y_i - w^T x_i)x_i$$

- **minibatch stochastic gradient.**
  pick a random set of examples $S$. set

  $$\tilde{\nabla} f(w) = \frac{n}{|S|} \nabla \left( \sum_{i \in S} (y_i - w^T x_i)^2 \right)$$

  $$= \frac{n}{|S|} \left( -2 \sum_{i \in S} (y_i - w^T x_i)x_i \right)$$

  (often, $|S| = 50$ or so.)
Stochastic gradient method for least squares

minimize $\|y - Xw\|^2$

Algorithm

Given: $X : \mathbb{R}^{n \times d}$, $y \in \mathbb{R}^n$, stepsize $\alpha$, maxiters

Initialize: $w = 0$ (or anything you’d like)

For: $k = 1, \ldots, \text{maxiters}$

▶ pick $i$ at random from $\{1, \ldots, n\}$

▶ update $w$:

$$w \leftarrow w + 2\alpha n(y_i - w^T x_i)x_i$$

▶ not a descent method; objective can increase!

▶ can’t use linesearch

▶ converges to **ball around** optimum;
  bigger $\alpha \implies$ larger ball
Stochastic gradient method for least squares

\[
\text{minimize} \quad \| y - Xw \|^2
\]

**Algorithm**  Stochastic gradient method for least squares

**Given:** \( X : \mathbb{R}^{n \times d}, \ y \in \mathbb{R}^n, \) stepsize \( \alpha, \) maxiters

**Initialize:** \( w = 0 \) (or anything you’d like)

**For:** \( k = 1, \ldots, \) maxiters

1. pick a random subset \( S \) from \{1, \ldots, n\}
2. update \( w: \)

\[
w \leftarrow w + \frac{2\alpha n}{|S|} \sum_{i \in |S|} (y_i - w^T x_i)x_i
\]
Stochastic gradient update:

\[ w \leftarrow w + \frac{2\alpha n}{|S|} \sum_{i \in |S|} (y_i - w^T x_i)x_i \]

How many flops does stochastic gradient require per iteration, as a function of the number of examples \( n \) and number of features \( d \)?

A. \( O(d^2) \)
B. \( O(|S|^2) \)
C. \( O(dn) \)
D. \( O(d|S|) \)
E. \( O(nd^2) \)
Outline

Regression

Gradient descent

Least squares via gradient descent

Faster!

Proofs for GD

Least squares via normal equations
**Convexity: definitions**

**Q:** Define convexity?
Convexity: definitions

A function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is convex iff it never lies above its chord: for all $\theta \in [0, 1]$, $w, v \in \mathbb{R}^n$

$$f(\theta w + (1 - \theta)v) \leq \theta f(w) + (1 - \theta)f(v)$$
Convexity: definitions

- A function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is convex iff it never lies above its chord: for all $\theta \in [0, 1]$, $w, v \in \mathbb{R}^n$

  $$f(\theta w + (1 - \theta)v) \leq \theta f(w) + (1 - \theta)f(v)$$

- A differentiable function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is convex iff it satisfies the first order condition

  $$f(v) - f(w) \geq \nabla f(w)^\top(v - w) \quad \forall w, v \in \mathbb{R}^n$$
Convexity: definitions

▶ A function \( f : \mathbb{R}^n \to \mathbb{R} \) is convex iff it never lies above its chord: for all \( \theta \in [0, 1] \), \( w, v \in \mathbb{R}^n \)

\[
f(\theta w + (1 - \theta)v) \leq \theta f(w) + (1 - \theta)f(v)
\]

▶ A differentiable function \( f : \mathbb{R}^n \to \mathbb{R} \) is convex iff it satisfies the first order condition

\[
f(v) - f(w) \geq \nabla f(w)^\top (v - w) \quad \forall w, v \in \mathbb{R}^n
\]

▶ A twice differentiable function \( f : \mathbb{R}^n \to \mathbb{R} \) is convex iff its Hessian is always positive semidefinite: \( \lambda_{\text{min}}(\nabla^2 f) \geq 0 \)
Poll: Convexity examples

Is this function convex?

A. yes

B. no
Theorem

For a convex and differentiable function,

\[ \nabla f(w) = 0 \iff w \text{ minimizes } f. \]

proof:
Convex function: global proof of optimality

**Theorem**

For a convex and differentiable function,

\[ \nabla f(w) = 0 \iff w \text{ minimizes } f. \]

**proof:** if \( \nabla f(x) \), then the first order condition says

\[ f(y) - f(x) \geq \nabla f(x)^\top (y - x) = 0 \quad \forall x, y \in \mathbb{R}^n \]
Convex function: global proof of optimality

**Theorem**

For a convex and differentiable function,

\[ \nabla f(w) = 0 \iff w \text{ minimizes } f. \]

**proof:** if \( \nabla f(x) \), then the first order condition says

\[
    f(y) - f(x) \geq \nabla f(x)^\top (y - x) = 0 \quad \forall x, y \in \mathbb{R}^n
\]

**Q:** Counterexample for nonconvex function?
Least squares objective is convex

**Theorem**

The least squares objective $f(w) = \|y - Xw\|^2$ is convex.

**proof:** consider any two models $w$ and $w'$. use the first order condition for convexity:

$$f(w') - f(w) \geq (\nabla f(w))^\top (w' - w)$$

compute

$$f(w') - f(w) = \|y - Xw'\|^2 - \|y - Xw\|^2$$

$$= y^\top y - 2y^\top Xw' + w'^\top X^\top Xw' - y^\top y + 2y^\top Xw - w^\top X^\top Xw$$

$$= -2y^\top X(w' - w) + w'^\top X^\top X(w' - w) + w^\top X^\top X(w' - w)$$

$$= -2y^\top X(w' - w) + (w' - w)^\top X^\top X(w' - w) + 2w^\top X^\top X(w' - w)$$

$$= -2y^\top X(w' - w) + \|X(w' - w)\|^2 + 2w^\top X^\top X(w' - w)$$

$$\geq (-2y^\top X + 2w^\top X^\top X)(w' - w)$$

$$= (\nabla f(w))^\top (w' - w)$$
Least squares is smooth

Definition

A continuously differentiable function $f : \mathbb{R} \to \mathbb{R}$ is $L$-smooth if, for all $w, w' \in \mathbb{R}$,

$$f(w') \leq f(w) + (\nabla f(w))^T (w' - w) + \frac{L}{2} \|w' - w\|^2.$$ 

claim: the least squares objective $f(w) = \|Xw - y\|^2$ is $L$-smooth for $L = 2\|X\|^2$
Least squares is smooth

**Definition**

A continuously differentiable function $f : \mathbb{R} \to \mathbb{R}$ is $L$-smooth if, for all $w, w' \in \mathbb{R}$,

$$f(w') \leq f(w) + (\nabla f(w))^T (w' - w) + \frac{L}{2} ||w' - w||^2.$$ 

**claim:** the least squares objective $f(w) = ||Xw - y||^2$ is $L$-smooth for $L = 2||X||^2$

**proof:**

$$f(w') = ||Xw' - y||^2$$

$$= ||X(w' - w) + Xw - y||^2$$

$$= ||Xw - y||^2 + 2(Xw - y)^T X(w' - w) + ||X(w' - w)||^2$$

$$= f(w) + (\nabla f(w))^T (w' - w) + ||X||^2 ||w' - w||^2$$

$$\leq f(w) + (\nabla f(w))^T (w' - w) + ||X||^2 ||w' - w||^2$$

so $L = 2||X||^2$, where $||X||$ is the maximum singular value of $X$.
Gradient descent converges when $\alpha \leq 2/L$

**Claim:** gradient descent converges for an $L$-smooth function $f : \mathbb{R} \to \mathbb{R}$ if the step size $\alpha \leq 2/L$.

**Proof:** $f$ is $L$-smooth, so

$$f(w^+) \leq f(w) + (\nabla f(w))^T (w^+ - w) + \frac{L}{2} \|w^+ - w\|^2.$$ 

Now use $w^+ - w = -\alpha \nabla f(w)$:

$$f(w^+) \leq f(w) + (\nabla f(w))^T (-\alpha \nabla f(w)) + \frac{L}{2} \| - \alpha \nabla f(w) \|^2$$

$$\leq f(w) - \alpha \|\nabla f(w)\|^2 + \frac{L \alpha^2}{2} \|\nabla f(w)\|^2$$

so $f(w^+) < f(w)$ when

$$-\alpha + \frac{L \alpha^2}{2} < 0 \quad \implies \quad \alpha < \frac{2}{L}$$
Outline

Regression

Gradient descent

Least squares via gradient descent

Faster!

Proofs for GD

Least squares via normal equations
Solving least squares: straight to the bottom

minimize $\|y - Xw\|^2$

solve by setting the gradient to 0: optimal $w$ satisfies

$$0 = \nabla \|y - Xw\|^2 = -2X^\top y + 2X^\top Xw$$

$X^\top Xw = X^\top y$

$X^\top X$ is called the **Gram matrix**

$X^\top Xw = X^\top y$ is called the **normal equations**
Solving least squares: straight to the bottom

minimize $\|y - Xw\|^2$

- solve by setting the gradient to 0: optimal $w$ satisfies

$$0 = \nabla \|y - Xw\|^2 = -2X^Ty + 2X^TXw$$

$$X^TXw = X^Ty$$

- $X^TX$ is called the **Gram matrix**
- $X^TXw = X^Ty$ is called the **normal equations**

Normal equations are very useful for understanding solution of least squares;
when $d$ is small, they are also useful for solving least squares.
Any solution to normal equations solves least squares

**Claim:** $X^\top X w = X^\top y \iff w$ is optimal

**Proof:** using first order condition,

$$
||y - Xw'||^2 - ||y - Xw||^2 \geq (\nabla_w ||y - Xw||^2)^\top (w' - w)
$$
Any solution to normal equations solves least squares

**claim:** $X^\top Xw = X^\top y \iff w$ is optimal

**proof:** using first order condition,

$$
\|y - Xw'\|^2 - \|y - Xw\|^2 \geq (\nabla_w \|y - Xw\|^2)^\top (w' - w)
$$

- if $\nabla_w \|y - Xw\|^2 = 0$, then for any $w'$,

$$
\|y - Xw'\|^2 - \|y - Xw\|^2 \geq 0
$$
Any solution to normal equations solves least squares

claim: $X^TXw = X^Ty \iff w$ is optimal

proof: using first order condition,

$$\|y - Xw'\|^2 - \|y - Xw\|^2 \geq (\nabla_w \|y - Xw\|^2)^\top (w' - w)$$

- if $\nabla_w \|y - Xw\|^2 = 0$, then for any $w'$,
  $$\|y - Xw'\|^2 - \|y - Xw\|^2 \geq 0$$

- so $w$ minimizes $\|y - Xw\|^2$!
Any solution to normal equations solves least squares

claim: \( X^\top X w = X^\top y \iff w \) is optimal

proof: using first order condition,

\[
\| y - Xw' \|^2 - \| y - Xw \|^2 \geq (\nabla_w \| y - Xw \|^2)^\top (w' - w)
\]

▶ if \( \nabla_w \| y - Xw \|^2 = 0 \), then for any \( w' \),

\[
\| y - Xw' \|^2 - \| y - Xw \|^2 \geq 0
\]

▶ so \( w \) minimizes \( \| y - Xw \|^2 \)!

▶ rewrite \( \nabla_w \| y - Xw \|^2 = 0 \) to get normal equations

\[
0 = \nabla_w \| y - Xw \|^2
= -2X^\top y + 2X^\top Xw
\]

\[
X^\top Xw = X^\top y
\]
Outline

Regression

Gradient descent

Least squares via gradient descent

Faster!

Proofs for GD

Least squares via normal equations
The fundamental theorem of numerical analysis

**Theorem**

*Never form the inverse (or pseudoinverse) of a matrix explicitly.*

(Numerically unstable.)

Corollary: never type `inv(X'*X)` or `pinv(X'*X)` to solve the normal equations.
The fundamental theorem of numerical analysis

**Theorem**

*Never form the inverse (or pseudoinverse) of a matrix explicitly.*

(Numerically unstable.)

Corollary: never type \( \text{inv}(X'X) \) or \( \text{pinv}(X'X) \) to solve the normal equations.

Instead: compute the inverse using easier matrices to invert, like

- **Orthogonal matrices** \( Q \):
  \[
  a = Qb \iff Q^T a = b
  \]

- **Triangular matrices** \( R \):
  if \( a = Rb \), can find \( b \) given \( R \) and \( a \) by solving sequence of simple, stable equations.
The QR factorization

rewrite $X$ in terms of **QR decomposition** $X = QR$

- $Q \in \mathbb{R}^{n \times d}$ has orthogonal columns: $Q^\top Q = I_d$
- $R \in \mathbb{R}^{d \times d}$ is upper triangular: $R_{ij} = 0$ for $i > j$
- diagonal of $R \in \mathbb{R}^{d \times d}$ is positive: $R_{ii} > 0$ for $i = 1, \ldots, d$
- this factorization always exists and is unique (proof by Gram-Schmidt construction)

can compute $QR$ factorization of $X$ in $2nd^2$ flops
The QR factorization

rewrite $X$ in terms of **QR decomposition** $X = QR$

- $Q \in \mathbb{R}^{n \times d}$ has orthogonal columns: $Q^\top Q = I_d$
- $R \in \mathbb{R}^{d \times d}$ is upper triangular: $R_{ij} = 0$ for $i > j$
- diagonal of $R \in \mathbb{R}^{d \times d}$ is positive: $R_{ii} > 0$ for $i = 1, \ldots, d$
- this factorization always exists and is unique (proof by Gram-Schmidt construction)

can compute $QR$ factorization of $X$ in $2nd^2$ flops

in julia (or matlab), use the QR function

$$Q, R = \text{qr} (X)$$

advantage of QR: it’s easy to invert $R$!
use QR to solve least squares: if $X = QR$,

\[
X^TXw = X^Ty \\
(QR)^TQRw = (QR)^Ty \\
R^TQ^TQRw = R^TQ^Ty \\
R^TRw = R^TQ^Ty \\
Rw = Q^Ty \\
w = R^{-1}Q^Ty
\]
Computational considerations

never form the inverse explicitly: numerically unstable!

instead, use QR factorization:

- compute QR factorization of $X$  
  $(2nd^2 \text{ flops})$
- to compute $w = R^{-1}Q^\top y$
  - form $b = Q^\top y$  
    $(2nd \text{ flops})$
  - compute $w = R^{-1}b$ by back-substitution  
    $(d^2 \text{ flops})$
never form the inverse explicitly: numerically unstable!

instead, use QR factorization:

▶ compute QR factorization of $X$  
  \[ (2nd^2 \text{ flops}) \]

▶ to compute $w = R^{-1}Q^\top y$
  \[ (2nd \text{ flops}) \]
  
  ▶ form $b = Q^\top y$
  \[ (d^2 \text{ flops}) \]

  ▶ compute $w = R^{-1}b$ by back-substitution

in julia (or matlab), the backslash operator solves least-squares efficiently (usually, using QR)

$$w = X \backslash y$$
Demo: QR

https://github.com/ORIE4741/demos/QR.ipynb
## Computational speed comparison

<table>
<thead>
<tr>
<th></th>
<th>GD</th>
<th>SGM</th>
<th>Gram GD</th>
<th>Parallel GD</th>
<th>QR</th>
</tr>
</thead>
<tbody>
<tr>
<td>initial</td>
<td>0</td>
<td>0</td>
<td>$nd^2$</td>
<td>$nd^2/P$</td>
<td>$nd^2$</td>
</tr>
<tr>
<td>per iter</td>
<td>$nd$</td>
<td>$</td>
<td>S</td>
<td>d$</td>
<td>$d^2$</td>
</tr>
</tbody>
</table>

(numbers in flops, omitting constants)
References

- QR factorization: https://en.wikipedia.org/wiki/QR_decomposition