ORIE 4741: Learning with Big Messy Data

Linear Models and Linear Least Squares

Professor Udell

Operations Research and Information Engineering
Cornell

October 16, 2021

Announcements 9/7/21

- section this week: python for data science (seaborn and pandas)
- hw1 is out, due next week
- missed the quiz? set a reminder for this week!
- private question? ask @staff on Zulip
- I will announce when you can register physical iClicker on Canvas. . .

Announcements 9/9/21

- column $(3, 4)$ vs row $[3, 4]$ vectors
- start looking for project groups: post your idea on zulip in the #project channel

Announcements 9/14/21

- section this week: github + jupyter tutorial
- bonus section from last year: linear algebra review
- hw1 is out, due this Thursday at 9:15am
- form project groups by this Sunday. see `https://people.orie.cornell.edu/mru8/orie4741/projects.html`
- looking for a project group? post your idea on zulip in the #project channel

Poll

How many Cornell students tested positive for COVID yesterday?

A. 2
B. 6
C. 13
D. 27
E. 233

Poll

Did your iClicker record your participation for last lecture 9/9/21?

A. yes
B. no

Poll

Questions about homework 1 should be posted on Zulip

A. in the #general channel, with a topic like "homework"
B. in the #homework 1 channel, with a topic like "q3c ambiguous wording"

Poll

Can we see examples of good projects from previous years?

A. yes
B. no

Outline

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:** $\mathcal{Y} = \mathbf{R}$
- **multiclass classification:** $\mathcal{Y} = \{\text{car}, \text{pedestrian}, \text{bike}\}$
- **ordinal regression:**
  $\mathcal{Y} = \{\text{strongly disagree}, \dots, \text{strongly agree}\}$

Regression

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

- 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

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

- 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 $\mathcal{X} = \mathbf{R}^d$, $\mathcal{Y} = \mathbf{R}$

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

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

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

Linear model++

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

- pick a transformation $\phi : \mathcal{X} \to \mathbf{R}^d$
- 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 $i = 1, \ldots, n$

Linear model++

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

- pick a transformation $\phi : \mathcal{X} \to \mathbf{R}^d$
- 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 ϕ?
- how to pick w?

Linear model++

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

- pick a transformation $\phi : \mathcal{X} \to \mathbf{R}^d$
- 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 ϕ?
- how to pick w?

for now, assume d and ϕ are fixed; we'll return to these later...

Least squares fitting

- define **prediction error** or **residual**

$$r_i = y_i - h(x_i), \qquad i = 1, \ldots, n$$

- 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 $y \in \mathbf{R}^n$: each outcome y_i is an entry of y
- form matrix $X \in \mathbf{R}^{n \times d}$: 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 $3 * 2 + 4 * 6$?

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

Poll

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, $v \in \mathbf{R}^d$?

A. d-1
B. d
C. d+1
D. 2d-1
E. 2d

Poll

How many flops to compute Xw, where $X \in \mathbf{R}^{n \times d}$, $w \in \mathbf{R}^{d}$?

A. n+2d-1
B. 2n+2d-1
C. 2nd
D. n(2d-1)
E. 2n(2d-1)

Poll

How many flops to compute $y - z$, where y, $z \in \mathbf{R}^n$?

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

Poll

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

A. n-1

B. n

C. n+1

D. 2n-1

E. 2n

Poll

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

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

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

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

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

Optimization

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

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

with $w \in \mathbf{R}^n$ when f is **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

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

with $w \in \mathbf{R}^n$ when f is **differentiable**

1. gradient descent
2. solve normal equations

when f is convex, both methods provably find the solution

example: for least squares, $f(w) = \|y - Xw\|^2$

The gradient

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

Definition

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

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

The gradient

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

Definition

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

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

- allows easy computation of directional derivatives:
  for fixed $v \in \mathbf{R}^d$, let $w^+(\alpha) = w + \alpha v$. then

$$\begin{aligned}\frac{d}{d\alpha} f(w^+(\alpha)) &= \frac{\partial f}{\partial w_1^+}\frac{dw_1^+}{d\alpha} + \cdots + \frac{\partial f}{\partial w_d^+}\frac{dw_d^+}{d\alpha} \\ &= (\nabla f(w))^\top v\end{aligned}$$

The gradient

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

Definition

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

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

- allows easy computation of directional derivatives: for fixed $v \in \mathbf{R}^d$, let $w^+(\alpha) = w + \alpha v$. then

$$\begin{aligned} \frac{d}{d\alpha} f(w^+(\alpha)) &= \frac{\partial f}{\partial w_1^+}\frac{dw_1^+}{d\alpha} + \cdots + \frac{\partial f}{\partial w_d^+}\frac{dw_d^+}{d\alpha} \\ &= (\nabla f(w))^\top v \end{aligned}$$

- 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_descent.ipynb`

Gradient descent

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

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

Algorithm Gradient descent

Given: $f : \mathbf{R}^d \to \mathbf{R}$, stepsize α, maxiters
Initialize: $w = 0$ (or anything you'd like)
For: $k = 1, \ldots,$ maxiters

- update w:

$$w \leftarrow w - \alpha \nabla f(w)$$

Gradient descent

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

Algorithm Gradient descent

Given: $f : \mathbf{R}^d \to \mathbf{R}$, maxiters

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

For: $k = 1, \ldots,$ maxiters

- choose stepsize $\alpha^{(k)}$
- update w:

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

---

nomenclature

- $w^{(k)} \in \mathbf{R}^d$ are called **iterates**
- $\alpha^{(k)} \in \mathbf{R}$ 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 α to minimize $f(w^+)$
- the **Armijo rule** requires α 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 α to minimize $f(w^+)$
- the **Armijo rule** requires α 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^+:

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

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

Line search

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

- exact line search: find α to minimize $f(w^+)$
- the **Armijo rule** requires α 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^+:

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

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

Q: can we can always satisfy the Armijo rule for some α?

Line search

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

- exact line search: find α to minimize $f(w^+)$
- the **Armijo rule** requires α 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^+:

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

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

Q: can we can always satisfy the Armijo rule for some α?
A: yes! see gradient descent demo

Outline

Some matrix calculus identities

two useful identities: let $w,\ b \in \mathbf{R}^d$, $A \in \mathbf{R}^{d \times d}$ symmetric

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

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

2. let $f(w) = w^\top A w$. 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 $\nabla f(w)$:

$$\begin{aligned}
\nabla f(w) &= \sum_{i=1}^{n} \nabla (y_i - w^T x_i)^2 \\
&= \sum_{i=1}^{n} -2(y_i - w^T x_i)x_i
\end{aligned}$$

Gradient of the least squares problem (matrix version)

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

compute $\nabla f(w)$:

$$\begin{aligned}
\nabla f(w) &= \nabla (y - Xw)^\top (y - Xw) \\
&= \nabla (y^\top y - w^\top X^\top y - y^\top Xw + w^\top X^\top Xw) \\
&= -\nabla (w^\top X^\top y + w^\top X^\top y) + \nabla (w^\top X^\top Xw) \\
&= -2X^\top y + 2X^\top Xw
\end{aligned}$$

Solving the least squares problem: gradient descent

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

Algorithm Gradient descent for least squares

Given: $X : \mathbf{R}^{n \times d}$, $y \in \mathbf{R}^n$, stepsize α, maxiters
Initialize: $w = 0$ (or anything you'd like)
For: $k = 1, \ldots,$ maxiters

- update w:

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

Poll

Gradient descent update:

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

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

A. $\mathcal{O}(d)$
B. $\mathcal{O}(n)$
C. $\mathcal{O}(nd)$
D. $\mathcal{O}(nd^2)$
E. $\mathcal{O}(n^2 d^2)$

Poll

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. $\mathcal{O}(d)$
B. $\mathcal{O}(n)$
C. $\mathcal{O}(nd)$
D. $\mathcal{O}(nd^2)$
E. $\mathcal{O}(n^2d^2)$

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

Speeding up gradient descent when $n \gg d$

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

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** $G = X^\top X = \sum_{i=1}^n x_i x_i^\top$ $\quad$ ($2nd^2$ flops)

Speeding up gradient descent when $n \gg d$

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

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)

Speeding up gradient descent when $n \gg d$

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

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)

$\mathcal{O}(nd^2)$ flops to start, plus $\mathcal{O}(d^2)$ per iteration

Parallel computation

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

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?

Parallelism: gradient descent

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

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

Parallelism: gradient descent

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

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

- form the **Gram matrix** $G = X^\top X = \sum_{p=1}^{P}(\sum_{i \in \mathcal{N}_p} x_i x_i^\top)$
  ($2nd^2/P$ flops per proc)

Parallelism: gradient descent

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

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

- form the **Gram matrix** $G = X^\top X = \sum_{p=1}^P (\sum_{i \in \mathcal{N}_p} x_i x_i^\top)$
  ($2nd^2/P$ flops per proc)
- form $b = X^\top y = \sum_{p=1}^P (\sum_{i \in \mathcal{N}_p} y_i x_i)$
  ($2nd/P$ flops per proc)

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 $\{1,\dots,n\}$.

- form the **Gram matrix** $G = X^\top X = \sum_{p=1}^P (\sum_{i\in\mathcal{N}_p} x_i x_i^\top)$ ($2nd^2/P$ flops per proc)
- form $b = X^\top y = \sum_{p=1}^P (\sum_{i\in\mathcal{N}_p} y_i x_i)$ ($2nd/P$ flops per proc)
- for $k = 1,\dots$
  - update $w^+ = w - 2\alpha(Gw - b)$ ($2d^2 + 3d$ 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

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

  (often, $|S| = 50$ or so.)

Stochastic gradient method for least squares

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

Algorithm Stochastic gradient method for least squares

Given: $X : \mathbf{R}^{n \times d}$, $y \in \mathbf{R}^n$, stepsize α, maxiters
Initialize: $w = 0$ (or anything you'd like)
For: $k = 1, \ldots,$ 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: $X : \mathbf{R}^{n \times d}$, $y \in \mathbf{R}^n$, stepsize α, maxiters
Initialize: $w = 0$ (or anything you'd like)
For: $k = 1, \dots,$ maxiters

- pick a random subset S from $\{1, \dots, n\}$
- update w:

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

Poll

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. $\mathcal{O}(d^2)$
B. $\mathcal{O}(|S|^2)$
C. $\mathcal{O}(dn)$
D. $\mathcal{O}(d|S|)$
E. $\mathcal{O}(nd^2)$

Demo: SGD

https://github.com/ORIE4741/demos/blob/master/gradient_descent.ipynb

Outline

Convexity: definitions

Q: Define convexity?

Convexity: definitions

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

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

Convexity: definitions

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

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

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

$$f(v) - f(w) \geq \nabla f(w)^\top (v - w) \qquad \forall w, v \in \mathbf{R}^n$$

Convexity: definitions

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

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

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

$$f(v) - f(w) \geq \nabla f(w)^\top (v - w) \qquad \forall w, v \in \mathbf{R}^n$$

- A twice differentiable function $f : \mathbf{R}^n \to \mathbf{R}$ is convex iff
  its Hessian is always **positive semidefinite**: $\lambda_{\min}(\nabla^2 f) \geq 0$

Poll: Convexity examples

Is this function convex?

A. yes
B. no

Convex function: global proof of optimality

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

$$f(y) - f(x) \geq \nabla f(x)^\top (y - x) = 0 \qquad \forall x, y \in \mathbf{R}^n$$

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

$$f(y) - f(x) \geq \nabla f(x)^\top (y - x) = 0 \qquad \forall x, y \in \mathbf{R}^n$$

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

$$\begin{aligned}
&f(w') - f(w) = \|y - Xw'\|^2 - \|y - Xw\|^2 \\
&= y^\top y - 2y^\top Xw' + w'^\top X^\top Xw' - y^\top y + 2y^\top Xw - w^\top X^\top Xw \\
&= -2y^\top X(w' - w) + w'^\top X^\top X(w' - w) + w^\top X^\top X(w' - w) \\
&= -2y^\top X(w' - w) + (w' - w)^\top X^\top X(w' - w) + 2w^\top X^\top X(w' - w) \\
&= -2y^\top X(w' - w) + \|X(w' - w)\|^2 + 2w^\top X^\top X(w' - w) \\
&\geq (-2y^\top X + 2w^\top X^\top X)(w' - w) \\
&= (\nabla f(w))^\top (w' - w)
\end{aligned}$$

Least squares is smooth

Definition

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

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

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 $f : \mathbf{R} \to \mathbf{R}$ is L-**smooth** if, for all $w,\ w' \in \mathbf{R}$,

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

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

proof:

$$\begin{aligned} f(w') &= \|Xw' - y\|^2 \\ &= \|X(w' - w) + Xw - y\|^2 \\ &= \|Xw - y\|^2 + 2(Xw - y)^T X(w' - w) + \|X(w' - w)\|^2 \\ &= f(w) + (\nabla f(w))^T (w' - w) + \|X(w' - w)\|^2 \\ &\leq f(w) + (\nabla f(w))^T (w' - w) + \|X\|^2 \|(w' - w)\|^2 \end{aligned}$$

so $L = 2\|X\|^2$, where $\|X\|$ is the maximum singular value of X

Gradient descent converges when $\alpha \le 2/L$

claim: gradient descent converges for an L-smooth function $f : \mathbf{R} \to \mathbf{R}$ if the step size $\alpha \le 2/L$.

proof: f is L-smooth, so

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

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

$$\begin{aligned} f(w^+) &\le f(w) + (\nabla f(w))^T(-\alpha \nabla f(w)) + \frac{L}{2}\|-\alpha \nabla f(w)\|^2 \\ &\le f(w) - \alpha\|\nabla f(w)\|^2 + \frac{L\alpha^2}{2}\|\nabla f(w)\|^2 \end{aligned}$$

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

$$-\alpha + \frac{L\alpha^2}{2} < 0 \quad \Longrightarrow \quad \alpha < 2/L$$

Outline

Solving least squares: straight to the bottom

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

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

$$\begin{aligned} 0 &= \nabla \|y - Xw\|^2 \\ &= -2X^\top y + 2X^\top Xw \\ X^\top Xw &= X^\top y \end{aligned}$$

- $X^\top X$ is called the **Gram matrix**
- $X^\top Xw = X^\top y$ is called the **normal equations**

Solving least squares: straight to the bottom

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

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

$$\begin{aligned} 0 &= \nabla\|y - Xw\|^2 \\ &= -2X^\top y + 2X^\top Xw \\ X^\top Xw &= X^\top y \end{aligned}$$

- $X^\top X$ is called the **Gram matrix**
- $X^\top Xw = X^\top y$ 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: $X^\top X w = X^\top y \iff w$ 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',

$$\|y - Xw'\|^2 - \|y - Xw\|^2 \geq 0$$

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

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

$$\begin{aligned} 0 &= \nabla_w \|y - Xw\|^2 \\ &= -2X^\top y + 2X^\top X w \\ X^\top X w &= X^\top y \end{aligned}$$

Outline

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 `inv(X'*X)` or `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** $X = QR$

- $Q \in \mathbf{R}^{n\times d}$ has orthogonal columns: $Q^\top Q = I_d$
- $R \in \mathbf{R}^{d\times d}$ is upper triangular: $R_{ij} = 0$ for $i > j$
- diagonal of $R \in \mathbf{R}^{d\times d}$ 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** $X = QR$

- $Q \in \mathbf{R}^{n \times d}$ has orthogonal columns: $Q^\top Q = I_d$
- $R \in \mathbf{R}^{d \times d}$ is upper triangular: $R_{ij} = 0$ for $i > j$
- diagonal of $R \in \mathbf{R}^{d \times d}$ 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

use `scipy.linalg.qr`:

```
Q,R = qr(X)
```

advantage of QR: it's easy to invert R!

QR for least squares

use QR to solve least squares: if $X = QR$,

$$
\begin{aligned}
X^\top X w &= X^\top y \\
(QR)^\top QR w &= (QR)^\top y \\
R^\top Q^\top QR w &= R^\top Q^\top y \\
R^\top R w &= R^\top Q^\top y \\
R w &= Q^\top y \\
w &= R^{-1} Q^\top y
\end{aligned}
$$

Computational considerations

never form the inverse explicitly: numerically unstable!

instead, use QR factorization:

- compute QR factorization of X ($2nd^2$ flops)
- to compute $w = R^{-1}Q^\top y$
  - form $b = Q^\top y$ ($2nd$ flops)
  - compute $w = R^{-1}b$ by back-substitution (d^2 flops)

Computational considerations

never form the inverse explicitly: numerically unstable!

instead, use QR factorization:

- compute QR factorization of X ($2nd^2$ flops)
- to compute $w = R^{-1}Q^\top y$
  - form $b = Q^\top y$ ($2nd$ flops)
  - compute $w = R^{-1}b$ by back-substitution (d^2 flops)

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

```
w = X \ y
```

in python, use `numpy.lstsq`

Demo: QR

https://github.com/ORIE4741/demos/QR.ipynb

Computational speed comparison

	GD	SGM	Gram GD	Parallel GD	QR
initial	0	0	nd^2	nd^2/P	nd^2
per iter	nd	$\|S\|d$	d^2	d^2	0

(numbers in flops, omitting constants)

References

- Stanford EE103: "Least squares" and "Least squares data fitting". Boyd, 2016.
- Learning from Data: Chapter 3. Abu-Mostafa, Magdon-Ismail, and Lin, 2012.
- Gradient descent: `https://www.cs.cmu.edu/~ggordon/10725-F12/slides/05-gd-revisited.pdf`. Gordon and Tibshirani, CMU.
- QR factorization: `https://en.wikipedia.org/wiki/QR_decomposition`