UMassAmherst
The Commonwealth's Flagship Campus

Gradient descent – using approximate gradient

Chaitra Gopalappa, PhD

Reference

- Chapter 8 – Murphy, 2023(Book 1)

Gradient descent (from previous slide sets)

$$Min\ f(\vec{x})$$
$$\vec{x} \in R^n$$

$$\vec{x}_{m+1} \leftarrow \vec{x}_m - \mu_m \nabla f(\vec{x}_m)$$

Complexity

Problems: What are key features related to these questions?

- What is optimal price for airline seat to maximize revenue?
- What should be testing frequency to minimize spread of a newly emergent infectious disease on a university campus?

- **Formulate problem**
  - Objective function?
  - Exogenous (decision) variables?
  - Endogenous (input) variables?
  - Constraints (if any)?

- Difficult to formulate analytically (no closed form equation), but value of function at any given point can be estimated (e.g., simulation/ system dynamics modeling)
- Difficult to find exact derivative
- Non-convex function- do not want to get stuck in global optima
- Stochastic or deterministic

Approximate gradient methods

- Finite difference derivative approximation; Finite difference stochastic approximation (FDSA)- or Kiefer-Wolfowitz SA
- Simultaneous perturbation stochastic approximation (SPSA) by Spall
  - SPSA in NN https://www.jhuapl.edu/spsa/PDF-SPSA/Vande_Wouwer_Simultaneous_Perturbation.PDF

Finite difference derivative approximation (1-D vector)

- Forward difference estimation of derivative using *Taylor's series expansion*
  - $f(x+h) = f(x) + \frac{h}{1!}\frac{df}{dx} + \frac{h^2}{2!}\frac{df^2}{dx} + \cdots + \frac{h^n}{n!}\frac{df^n}{dx}$
  - $f(x+h) = f(x) + \frac{h}{1!}\frac{df}{dx} + \delta$ (For a sufficiently small h, δ ~0)
  - $\frac{df}{dx} = \frac{f(x+h)-f(x)}{h}$ (forward difference estimation of derivative)
- Central difference estimation of derivative using *Taylor's series expansion*
  - $f(x+h) = f(x) + \frac{h}{1!}\frac{df}{dx} + \frac{h^2}{2!}\frac{df^2}{dx} + \cdots + \frac{h^n}{n!}\frac{df^n}{dx}$
  - $f(x+h) = f(x) + \frac{h}{1!}\frac{df}{dx} + \frac{h^2}{2!}\frac{df^2}{dx} + \delta_1$ (1)
  - $f(x-h) = f(x) + \frac{(-1)h}{1!}\frac{df}{dx} + \frac{(-1)^2h^2}{2!}\frac{df^2}{dx} + \cdots + \frac{(-1)^nh^n}{n!}\frac{df^n}{dx}$
  - $f(x-h) = f(x) + \frac{(-1)h}{1!}\frac{df}{dx} + \frac{(-1)^2h^2}{2!}\frac{df^2}{dx} + \delta_2$ (2)
  - (For a sufficiently small h, δ_1 ~0, δ_2 ~0)
  - (1)- (2) gives $\frac{df}{dx} = \frac{f(x+h)-f(x-h)}{2h}$ (central difference estimation of derivative)

Pointers:

- Central difference approximation is preferred as it maintains the second order derivative
- Minimize randomness between estimations of $f(x+h)$ and $f(x-h)$, e.g., if using stochastic simulation, use the same random numbers for evaluation of function at both points

Finite difference stochastic approximation

$$Min\ f(\vec{x})$$
$$\vec{x} \in R^n$$

Initialization: $m = 1$, $\vec{x}$ = vector of arbitrary starting point; set stopping conditions

1. Set $h[i] \leftarrow c$
2. Calculate $\mathrm{F}^+[i]$ and $\mathrm{F}^-[i]$ from simulation for each $i = 1,2,\ldots,n$
   (k is the number of variables, i.e., size of vector $\vec{x}$)
   1. $\mathrm{F}^+[i] = f(x_m[1] + h[1]\delta_{1i},\quad x_m[2] + h[2]\delta_{2i},\quad \ldots,\quad x_m[k] + h[k]\delta_{ki})$
   2. $\mathrm{F}^-[i] = f(x_m[1] - h[1]\delta_{1i},\quad x_m[2] - h[2]\delta_{2i},\quad \ldots,\quad x_m[k] - h[k]\delta_{ki})$
3. for each $i \in \{1,2,\ldots,n\}$
   1. $\dfrac{\partial f(\vec{x})}{\partial x[i]} \approx \dfrac{\mathrm{F}^+[i]-\mathrm{F}^-[i]}{2h[i]}$
4. Update
   1. $x_{m+1}[i] = x_m[i] - \mu_m \frac{\partial f(\vec{x})}{\partial x[i]}$ *for i=1,2,….n*
   2. $\vec{x}_{m+1} \leftarrow \vec{x}_m - \mu_m \nabla f(\vec{x}_m)$
5. Set $m = m + 1$; if stopping condition stop, else go to 1,

$$c = small\ value$$
$$c = f(m), e.g., c = \frac{1}{m}$$

Kronecker's delta

$$\delta_{ij} = \begin{cases} 0\ if\ i \neq j \\ 1\ if\ i = j \end{cases}$$

Drawbacks of FDSA

- Number of function evaluations = $2n, n$ = # of decision variables
- Computational burden high if n is high

Reference

- Chapter 5.3 and 5.4 – Powell, 2019
- Chapter 8 – Murphy, 2023 (Book 1)

Simultaneous perturbation stochastic approximation

Initialization: $m = 1$, $\vec{x}$ = arbitrary starting point; set stopping conditions

1. Let $H[i] \sim Bernoulli(p = 0.5)$ with density function $f(n,p) = \begin{cases} p \; if \; n = 1 \\ 1 - p \; if \; n = -1 \end{cases}$
   1. Set $h[i] = H[i]c$; $i = 1,2,\ldots,n$
2. Calculate F^{+} and F^{-} from simulation
   1. $\mathrm{F}^{+} = f(\vec{x}_m + \vec{h})$
   2. $\mathrm{F}^{-} = f(\vec{x}_m - \vec{h})$
3. for each $i \in \{1,2,\ldots,n\}$
   1. $\frac{\partial f(\vec{x})}{\partial x[i]} \leftarrow \frac{\mathrm{F}^{+} - \mathrm{F}^{-}}{2h[i]}$ (notice, unlike in FDSA, numerator does not have [i], although denominator does)
4. Update
   1. $\vec{x}_{m+1} \leftarrow \vec{x}_m - \mu_m \nabla f(\vec{x}_m)$
5. Set $m = m + 1$; if stopping condition, stop, else goto 1

$$\boxed{\begin{array}{l} c = f(m), e.g., \\ c = \dfrac{Constant}{m} \end{array}}$$

Generalized: Stochastic gradient algorithm

- *Transformation*

$$\vec{x}_{m+1} \leftarrow \vec{x}_m - \mu_m Y_m(\vec{x}_m)$$

$$Y_m(\vec{x}_m) = \nabla f(\vec{x}_m) + noise$$

Projected gradient descent

$$Min\ f(\vec{x})$$
$$\vec{x} \in C^n$$

C^n *is a n-dimensional convex set, constrained problem*

$$\vec{x}_{m+1} \leftarrow P_C[\vec{x}_m - \mu_m \nabla f(\vec{x}_m)]$$

P_C *is the projection onto the feasible space.*

For example $$Min f(\vec{x}); \vec{a} \le \vec{x} \le \vec{b}$$

$$P[x[i], a[i], b[i]] = \begin{cases} a[i]\ if\ x[i] \le a[i] \\ b[i]\ if\ x[i] \ge b[i]\ ; \forall i \in \{1, .., n\} \\ x[i]\ o/w \end{cases}$$

Gradient descent for constrained problems

- $Minf(\vec{x}); \vec{a} \leq \vec{x} \leq \vec{b}$ $\quad P[x[i], a[i], b[i]] = \begin{cases} a[i] \; if \; x[i] \leq a[i] \\ b[i] \; if \; x[i] \geq b[i] \, ; \forall i \in \{1, .., n\} \\ x[i] \; o/w \end{cases}$
- Main transformation

$$x[i]_{m+1} \leftarrow x[i]_m - \frac{\mu_m \partial f(\vec{x})}{\partial x[i]} \; for \; i = 1,2, \dots, N$$

Same as gradient descent except for the points in red

Algorithm

1. Set m = 1, select μ_m (tiny step), set M (max iterations) to a sufficiently large value
2. Initialize $\vec{x}_m$ to an arbitrary feasible solution
3. Obtain $\frac{\partial f(\vec{x})}{\partial x[i]} \; for \; i = 1,2, \dots, N$
4. Update $x[i]_{m+1}$, Apply $x[i]_{m+1} = P[x[i], a[i], b[i]] \forall i$; update μ_{m+1}
5. If all $\frac{\partial f(\vec{x})}{\partial x[i]}$=0, or sufficiently close, or $m = M$ STOP. Else set $m = m + 1$, goto step 3

Pointers:

- Repeat above multiple times, with a different starting point in step 2;
- Here: μ_m is a hyperparameter
  - Try different μ_m (it can be a constant that does not change with m; or can be a function of m, e.g., $1/m$)
- Put a constraint on number of iterations

UMassAmherst

The Commonwealth's Flagship Campus