Limited-memory BFGS
(Broyden-Fletcher-Goldfarb-Shanno)

Ovaj dokument predstavlja kratak opis implementiranog optimizacionog algoritma, Limited-memory BFGS. Na početku će biti reči o samom algoritmu, a potom će biti izneti koraci algoritma. Sam algoritam je implementiran u programskom jeziku Python. Na kraju rada će biti dati primeri funkcija koje su testirane i sa referentnom postojećom C bibliotekom.

Uvod

BFGS (**B**royden-**F**letcher-**G**oldfarb-**S**hanno) je optimizacioni algoritam iz porodice kvazi-Njutnovih[1] metoda koji predstavlja iterativnu metodu za rešavanje nelinearnih optimizacionih problema. U kvazi-Njutnovim metodama Hesian[2] matrica se ne računa direktno, već se aproksimira koristeći evaluacije gradijenata.

L-BFGS algoritam je takodje optimizacioni algoritam koji aproksimira BFGS algoritam koristeći ograničenu količinu memorije. Cilj ovog algoritma jeste da minimizira funkciju $f(x)$, gde je f diferencijabilna funkcija, a x element višedimenzionog prostora bez ograničenja.

Numeričke optimizacije igraju veliku ulogu u mašinskom učenju. Kada smo definisali naš model i imamo spreman skup podataka, estimacija parametara za naš model svodi se na minimiziranje neke funkcije. Zato je L-BFGS algoritam često korišćen u mašinskom učenju.

Kao i originalni BFGS, L-BFGS koristi procenu inverzne Hesian matrice, kako bi usmerio svoju pretragu. Umesto čuvanja velike količine aproksimacija Hesian matrice, L-BFGS skladišti samo nekoliko vektora. Zbog svojih potreba za linearnom memorijom, L-BFGS metoda je posebno pogodna za problem optimizacije sa velikim brojem promenljivih. Umesto inverzne Hesian matrice, L-BFGS čuva prethodnih m ažuriranja pozicije x i njegovog gradijenta $\nabla f(x)$, gde je generalno vrednost m mala ($m < 10$).

[1] Kvazi-Njutnove metode su metode koje se koriste za računanje nula funkcije ili minimuma i maksimuma funkcije, kao alternativa Njutnovim metodama. Obično se koriste ako je skupo računati Jakobi ili Hesian matrice.
[2] Hesian matrica je kvadratna matrica drugih izvoda funkcije koja opisuje lokalnu zakrivljenost funkcije.

Algoritam i impelementacija

Algoritam počinje sa inicijalizacijom optimalne vrednosti x_0, i nastavlja iterativno da bi je pročistio nizom tačnijih aproksimacija $x_1, x_2, \dots$. Izvodi funkcije $g_k \coloneqq \nabla f(x_k)$ se koriste za identifikaciju pravca najbržeg spuštanja, kao i za formiranje aproksimacija Hesian matrice.

L-BFGS ima dosta zajedničkih tačaka sa drugim kvazi-Njutonovim algoritmima, ali je veoma različit u načinu množenja vektora $d_k = -H_k g_k$, gde je d_k aproksimiran Njutnov smer, g_k trenutni gradijent, a H_k je inverz Hesian matrice. Postoji više poznatih pristupa koji koriste istoriju ažuriranja za formiranje vektora pravca. Ovde ćemo koristiti uobičajen pristup, takozvanu "rekurziju dve petlje".

Neka je dato x_k kao pozicija na k-toj iteraciji i gradijent $g_k \equiv \nabla f(x_k)$, gde je f funkcija koja se minimizuje, a svi vektori su vektori kolone. Takođe pretpostavljamo da smo sačuvali poslednjih m ažuriranja sledećih jednačina:

$$s_k = x_{k+1} - x_k$$
$$y_k = g_{k+1} - g_k$$

Definišemo $\boldsymbol{\rho}_k = \frac{1}{y_k^T s_k}$, a H_k^0 će biti inicijalna aproksimacija inverza Hesian matrice sa kojom počinje naša k-ta iteracija. Algoritam je zasnovan na BFGS rekurziji za inverz Hesian matrice:

$$H_{k+1} = \left(I - \boldsymbol{\rho}_k s_k y_k^T\right) H_k \left(I - \boldsymbol{\rho}_k y_k s_k^T\right) + \boldsymbol{\rho}_k s_k s_k^T$$

Pravac spuštanja, z, možemo računati na sledeći način:

$q = g_k$

$For\ i = k-1, k-2, \dots, k-m$

$\quad \alpha_i = \boldsymbol{\rho}_i s_i^T q$

$\quad q = q - \alpha_i y_i$

$\gamma_k = \dfrac{s_{k-1}^T y_{k-1}}{y_{k-1}^T y_{k-1}}$

$H_k^0 = \gamma_k I$

$z = H_k^0 q$

$For\ i = k-m, k-m+1, \dots, k-1$

$\quad \beta_i = \boldsymbol{\rho}_i y_i^T z$

$\quad z = z + s_i(\alpha_i - \beta_i)$

Ova formulacija daje smer za pretragu problema maksimizacije, tj. $z = H_k g_k$. Za problem minimizacije treba uzimati $-z$. Treba primetiti da se za inicijalnu aproksimaciju inverza Hesian matrice uzima dijagonalna matrica jer je to numerički efikasno.

Skaliranje matrice γ_k osigurava da se smer pretrage dobro promeni. Koristi se Armijo linijska pretraga[3] da bi se garantovao pad vrednosti ciljne funkcije dovoljan za kovergenciju. Ova metoda računa maksimalni korak koji zadovoljava tzv. Armijov uslov:

$$f(x + \alpha p) \leq f(x) + \beta_1 \alpha p \, \nabla f(x), \beta_1 \in (0,1)$$

Armijo uslov garantuje samo dužinu koraka. Koristi se takodje i linijska pretraga sa Wolf-ovim uslovima kako bi se osiguralo da je uslov zakrivljenosti ispunjen i BFGS ažuriranje stabilno. Wolf uslovi predstavljaju skup nejednačina za linijsku pretragu, posebno u kvazi-Njutnovim metodama, koji se koriste za minimizaciju problema bez ograničenja. Uslov zakrivljenosti glasi:

$$\nabla f(x + \alpha p)^T p \geq f(x) + \beta_2 p \, \nabla f(x), \beta_2 \in (0,1), \beta_1 < \beta_2$$

BFGS algoritam, pa i L-BFGS, su dizajnirani tako da rade samo za glatke funkcije bez ograničenja. Da bi radilo i za nediferencijabilne funkcije ili za funkcije sa ograničenjima, trebalo bi ih modifikovati. Postoji već nekoliko varijanti za ovaj problem.

Jedna od njih je L-BFGS-B algoritam, koji predstavlja proširenje L-BFGS sa mogućnošću dodatnih uslova za promenljive. Uslovi su oblika $l_i \leq x_i \leq u_i$, gde su l_i i u_i gornje i donje granice, za svako x_i. Ovaj metod radi tako što pronalazi fiksirane i slobodne promenljive u svakom koraku i primenjuje L-BFGS algoritam na slobodne promenljive da bi se dobila veća tačnost.

[3] Linijska pretraga predstavlja nalaženje maksimalne vrednosti parametra α za koju će biti izvršena minimizacija funkcije duž nekog pravca p. Preciznije, za neki početni vektor x i funkciju $f: D \to R$, traži se vrednost $f(x + \alpha p)$ koja se minimizuje.

Testne instance

Funkcije koje su korišćene za proveru rada algoritma su diferencijabilne i bez dodatnih ograničenja po parametrima. Primeri funkcija:

- $f(x_1, \dots, x_n) = \sum_{i=1,2}^{n-1} (1 - x_i)^2 + \ (10 * (x_{i+1} + x_i{}^2))^2$
- $f(x_1, x_2) = 100 * (x_2 - \ x_1{}^2)^2 + (1 - x_1)^2$
- $f(x_1, x_2) = (\ x_1 \ + 2 * \ x_2 \ - 7)^2 + (2 * \ x_1 \ + \ \ x_2 \ - 5)^2$
- $f(x_1, x_2) = 20 * \ x_1{}^2 + 100 * \ x_2{}^2$
- $f(x_1, x_2) = \ x_1{}^2 + x_2{}^3 + x_1 * x_2.$

Literatura

- http://en.wikipedia.org/wiki/Limited-memory_BFGS/

- http://ni.matf.bg.ac.rs/

- http://aria42.com/blog/2014/12/understanding-lbfgs