Legmeredekebb ereszkedés (steepest descent)

Kormányos Andor

Komplex Rendszerek Fizikája Tanszék

2020. október 19.

Függvényextrémum keresés: többváltozós eset

Többváltozós $f(\mathbf{x})$ függvény esetén nem tudjuk bekeretezni a minimumot!

Függvényextrémum keresés: többváltozós eset

Többváltozós $f(\mathbf{x})$ függvény esetén nem tudjuk bekeretezni a minimumot!

A módszerek most is két csoportra oszthatóak:

- csak függvénykiértékelések szükségesek, pl Nelder–Mead-módszer
- a függvény $\nabla f(\mathbf{x})$ gradiensét is fel tudjuk használni, pl legmeredekebb ereszkedés módszer

A legmeredekebb ereszkedés

Induljunk ki egy tetszőleges $\mathbf{P}_0$ pontból

- határozzuk meg a gradiensvektort: $\nabla f(\mathbf{P}_0)$
- $-\nabla f(\mathbf{P}_0)$ által kijelölt irány mentén csökken a leggyorsabban a függvény
- a következő lépés az irány által kijelölt **egyenes mentén** történő minimalizáció
- ehhez a lépéshez használhatjuk az egydimenziós esetben kifejlesztett módszereket
- így érünk el a $\mathbf{P}_1$ minimum pontba, ez általában egy lokális minimum
- a lépéssor ismétlése $-\nabla f(\mathbf{P}_i)$ irányba
- ez a legmeredekebb ereszkedés (steepest descent) módszer

A legmeredekebb ereszkedés

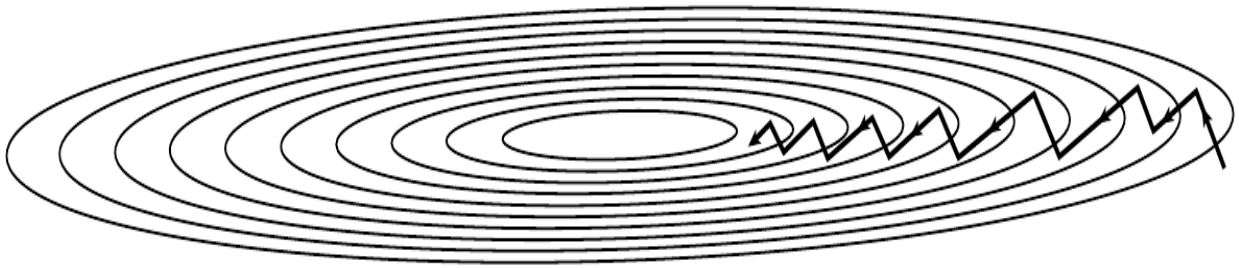

Figure: Legmeredekebb ereszkedés iteráció. ©Numerical Recipies

Probléma

- az újabb minimum pontban a gradiens mindig merőleges a előző irányra
- ez abból következik, hogy minimumba léptünk (ha nem ez lenne a helyzet, akkor az azt jelentené, hogy az előző irány menti derivált nem nulla a minimumban)
- "szűk völgyekben" nagyon lassan halad az iteráció
- erre lehet megoldás az ún. **konjugált gradiens módszer**

Legmeredekebb ereszkedés regressziós feladatokban[1]

Gépi tanulással kapcsolatos feladatokban gyakran a legmeredekebb ereszkedés egy egyszerűbb változatát használják

- adott költségfüggvény $J(\mathbf{a}; \mathbf{x}^{(i)}, y^{(i)})$, ahol $\mathbf{a}$ paramétervektor
- keressük $\min_{\mathbf{a}}[J(\mathbf{a}; \mathbf{x}^{(i)}, y^{(i)})]$

[1] Andrew Ng, Coursera alapján

Legmeredekebb ereszkedés regressziós feladatokban[1]

Gépi tanulással kapcsolatos feladatokban gyakran a legmeredekebb ereszkedés egy egyszerűbb változatát használják

- adott költségfüggvény $J(\mathbf{a}; \mathbf{x}^{(i)}, y^{(i)})$, ahol $\mathbf{a}$ paramétervektor
- keressük $\min_{\mathbf{a}}[J(\mathbf{a}; \mathbf{x}^{(i)}, y^{(i)})]$

Stratégia (gradient descent)

- válasszunk egy tetszőleges kezdőértéket a $a_0, a_1 \ldots a_k$-nak
- `repeat until convergence` $\{$
  $a_j := a_j - \alpha \frac{\partial}{\partial a_j} J(\mathbf{a}; \mathbf{x}^{(i)}, y^{(i)})$
  $\}$
- $\alpha > 0$: *learning parameter*, nekünk kell megadni

[1] Andrew Ng, Coursera alapján

Legmeredekebb ereszkedés regressziós feladatokban

Stratégia:

$\mathbf{a} = \mathbf{a}^{(init)}$
repeat until convergence $\{$
$a_j := a_j - \alpha \frac{\partial}{\partial a_j} J(\mathbf{a}; \mathbf{x}^{(i)}, y^{(i)})$
$\}$

- a legmeredekebb ereszkedés algoritmussal ellentétben nincs irány menti minimalizáció a legközelebbi lokális minimumig
- viszont bevezetjük α paramétert
- ha α túl nagy, akkor elvétjük a minimumot, ha túl kicsi, lassú a konvergencia
- mit jelent a konvergencia figyelése? később tárgyaljuk

Legmeredekebb ereszkedés regressziós feladatokban

Stratégia:

$\mathbf{a} = \mathbf{a}^{(init)}$
repeat until convergence $\{$
$a_j := a_j - \alpha \frac{\partial}{\partial a_j} J(\mathbf{a}; \mathbf{x}^{(i)}, y^{(i)})$
$\}$

- a legmeredekebb ereszkedés algoritmussal ellentétben nincs irány menti minimalizáció a legközelebbi lokális minimumig
- viszont bevezetjük α paramétert
- ha α túl nagy, akkor elvétjük a minimumot, ha túl kicsi, lassú a konvergencia
- mit jelent a konvergencia figyelése? később tárgyaljuk

Kérdés: függ-e az interáció $\mathbf{a}^{(final)}$ eredménye attól, hogy milyen $\mathbf{a}^{(init)}$ kezdőértéket választunk?

- általában függhet, ha a $J(\mathbf{a}; x^{(i)}, y^{(i)})$ költségfüggvény olyan, hogy több lokális minimuma van
- az iteráció beragadhat egy közeli lokális minimumba

Példa: egyenesillesztés legkisebb négyzetek módszerrel

- adathalmaz: $\{(x^{(i)}, y^{(i)})\}$, $i = 0 \ldots N$ számpárok
- erre szeretnénk egy $h(x; \mathbf{a}) = a_0 + a_1 x$ függvényt illeszteni

Példa: egyenesillesztés legkisebb négyzetek módszerrel

- adathalmaz: $\{(x^{(i)}, y^{(i)})\}$, $i = 0 \dots N$ számpárok
- erre szeretnénk egy $h(x; \mathbf{a}) = a_0 + a_1 x$ függvényt illeszteni
- legkisebb négyzetek módszere: minimalizáljuk a

$$J(a_0, a_1; x^{(i)}, y^{(i)}) = \frac{1}{2N} \sum_{i=1}^{N} \left(y^{(i)} - (a_0 + a_1 x^{(i)}) \right)^2$$

költségfüggvényt!

Példa: egyenesillesztés legkisebb négyzetek módszerrel

- adathalmaz: $\{(x^{(i)}, y^{(i)})\}$, $i = 0 \ldots N$ számpárok
- erre szeretnénk egy $h(x; \mathbf{a}) = a_0 + a_1 x$ függvényt illeszteni
- legkisebb négyzetek módszere: minimalizáljuk a

$$J(a_0, a_1; x^{(i)}, y^{(i)}) = \frac{1}{2N} \sum_{i=1}^{N} \left(y^{(i)} - (a_0 + a_1 x^{(i)}) \right)^2$$

  költségfüggvényt!
- a parciális deriváltak:

$$\frac{\partial}{\partial a_0} J(a_0, a_1) = \frac{1}{N} \sum_{i=1}^{N} \left(y^{(i)} - (a_0 + a_1 x^{(i)}) \right)$$

$$\frac{\partial}{\partial a_1} J(a_0, a_1) = \frac{1}{N} \sum_{i=1}^{N} \left(y^{(i)} - (a_0 + a_1 x^{(i)}) \right) x^{(i)}$$

Példa: egyenesillesztés legkisebb négyzetek módszerrel

Tehát az algoritmus:

$a_0 = a_0^{(init)}$, $a_1 = a_1^{(init)}$ /*kezdőérték adás*/

`repeat until convergence`

{

$a_0 := a_0 - \alpha \frac{1}{N} \sum_{i=1}^{N} \left(y^{(i)} - (a_0 + a_1 x^{(i)}) \right)$

$a_1 := a_1 - \alpha \frac{1}{N} \sum_{i=1}^{N} \left(y^{(i)} - (a_0 + a_1 x^{(i)}) \right) x^{(i)}$

}

- megmutatható, hogy egyenesillesztés esetén csak egy minimum van, az algoritmus oda konvergál

Legkisebb négyzetek illesztés: kitekintés

Megjegyzések:

- az egyenesillesztés esetén az a_0, a_1 kiszámítására létezik egy egzakt képlet
- **általánosított legkisebb négyzetek illesztés**: pl $a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \ldots$
- ennek létezik egy nem iteratív megoldása is: normál egyenletek
- ez egy lineáris egyenletrendszer megoldására vezet vissza
- ha olyan a probléma, hogy sok paramétert kell illeszteni vagy nagyon sok az adat, akkor a `gradient descent` módszer előnyösebb lehet (jobb skálázódás)