Neurális hálók tanítása során alkalmazott optimalizáció

Neurális hálózatok – 2016 tavasz

Háló paramétereinek tanulása

- Lényegében egy szélsőérték keresési feladat:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \left\{ L\left\{ f\left\{ \boldsymbol{\theta}, \mathbf{X} \right\}, \mathbf{d} \right\} \right\} = \arg\min_{\boldsymbol{\theta}} \left\{ J\left\{ \boldsymbol{\theta} \right\} \right\}$$

  - $\boldsymbol{\theta}$: háló paramétereinek vektora
  - $\mathbf{X}$: tanító minták bemeneteiből képzett mátrix
  - $\mathbf{d}$: tanító mintákhoz tartozó elvárt kimenet
  - $f\left\{\cdot,\cdot\right\}$: háló által megvalósított leképezés
  - $L\left\{\cdot,\cdot\right\}$: úgynevezett Loss function
  - $J\left\{\boldsymbol{\theta}\right\} = L\left\{ f\left\{ \boldsymbol{\theta}, \mathbf{X} \right\}, \mathbf{d} \right\}$

Optimalizációs algoritmusok

- Hibafelület jellege szerint:
  - **Konvex:** egyszerűbb eset – a célfüggvény konvex függvénye a keresett paramétervektornak
  - **Globális:** lassabb algoritmusok – ha nem konvex célfüggvény szélsőértékének helyét keressük
- Neurális hálók esete:
  - Nemlinearitások miatt nem konvex hibafelület
  - Általában konvex optimalizációs eljárások
    - Lokális optimumot képesek csak megtalálni
    - Mély hálóknál ez kevésbé jelent problémát

Konvex optimalizációs algoritmusok

- Továbbiakban konvex optimalizáció:
  - Iteratív algoritmusok (kivéve pár egyszerű eset)
- Iterációk során javítóirányt keresünk:
  - $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \lambda \cdot \boldsymbol{\Delta\theta}_k$
  - $\boldsymbol{\Delta\theta}_k$ becslése $J\{\boldsymbol{\theta}\}$ Taylor polinomja alapján
- Csoportosítás Taylor polinom fokszáma szerint:
  - 0-ad rendű: pl. Genetikus algoritmus
  - Elsőrendű: pl. SGD, Momentum, AdaGrad …
  - Másodrendű: Newton / kvázi Newton módszerek

Newton módszer

- $J^{(2)}_{\boldsymbol{\theta}_0}\left\{\boldsymbol{\theta}+\boldsymbol{\theta}_0\right\} = J\left\{\boldsymbol{\theta}_0\right\} + \boldsymbol{\theta}^T \cdot \nabla J\left\{\boldsymbol{\theta}_0\right\} + \left(1/2\right)\cdot \boldsymbol{\theta}^T \cdot \mathbf{H}\left\{\boldsymbol{\theta}_0\right\} \cdot \boldsymbol{\theta}$
  - $\mathbf{H}\left\{\boldsymbol{\theta}_0\right\}_{(i,j)} = \dfrac{\partial^2 J\left\{\boldsymbol{\theta}_0\right\}}{\partial \boldsymbol{\theta}_{(i)} \partial \boldsymbol{\theta}_{(j)}}$ jelöli J Hesse-mátrixát
  - $\nabla J\left\{\boldsymbol{\theta}_0\right\}_{(i)} = \dfrac{\partial J\left\{\boldsymbol{\theta}_0\right\}}{\partial \boldsymbol{\theta}_{(i)}}$ jelöli a gradienst $\boldsymbol{\theta}_0$ -ban
  - Ha $\mathbf{H}\left\{\boldsymbol{\theta}_0\right\}$ pozitív definit, akkor

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \left\{ J^{(2)}_{\boldsymbol{\theta}_0}\left\{\boldsymbol{\theta}\right\} \right\} \quad \Leftrightarrow \quad \nabla J^{(2)}_{\boldsymbol{\theta}_0}\left\{\boldsymbol{\theta}^*\right\} = \mathbf{0}$$

    - Tehát $\boldsymbol{\theta}^* = -\mathbf{H}\left\{\boldsymbol{\theta}_0\right\}^{-1} \cdot \nabla J\left\{\boldsymbol{\theta}_0\right\}$

Newton módszer

- $J^{(2)}_{\boldsymbol{\theta}_0}\{\boldsymbol{\theta}\}$ csak approximálja J-t:
  - Viszont J $\boldsymbol{\theta}_0$ pontbeli kvadratikus közelítésének t.f.h. a minimumhelye: $\boldsymbol{\theta}_0 + \boldsymbol{\theta}^*$
  - Tehát $\Delta\boldsymbol{\theta} = \alpha \cdot \boldsymbol{\theta}^*$
  - α meghatározása line-search algoritmusokkal. Az alábbi trade-off problémát oldják meg:
    - $\left| J^{(2)}_{\boldsymbol{\theta}_0}\{\boldsymbol{\theta}_0 + \alpha \cdot \Delta\boldsymbol{\theta}\} - J\{\boldsymbol{\theta}_0 + \alpha \cdot \Delta\boldsymbol{\theta}\} \right| < \varepsilon$
    - $J^{(2)}_{\boldsymbol{\theta}_0}\{\boldsymbol{\theta}_0\} - J^{(2)}_{\boldsymbol{\theta}_0}\{\boldsymbol{\theta}_0 + \alpha \cdot \Delta\boldsymbol{\theta}\} > \delta$

Newton módszer

- Interpretációja:
  1. Aktuális paramétervektor környezetében egy kvadratikus felülettel közelítjük $J\left\{\boldsymbol{\theta}\right\}$ -t.
  2. Megkeressük a közelítés minimumhelyét, majd annak irányába lépünk „elegendően” nagyot.
  3. Ha nem teljesül a leállási feltétel, akkor GOTO 1.
- Előnyei:
  - Pár 10 / 100 iteráció gyakorlatban elég
  - $J\left\{\boldsymbol{\theta}\right\}$ affin transzformációjára érzéketlen

Newton módszer

- Hátrányai:
  - Minden iterációban újra kell számolni, és invertálni a Hesse mtx.-ot (mely akár 10^8×10^8 méretű):
    - Kvázi Newton módszerek ezt próbálják kikerülni
  - Batch-elt tanításhoz illeszkedik jól – túl nagy lépéseket tesz ahhoz, hogy sztochasztikus (pl. mini batch) optimalizációra alkalmas legyen:
    - Így nem is lehet jól párhuzamosítani
- Klasszikus Neurális hálóknál alkalmazható inkább (lsd. Levenberg-Marquadt)

Kvázi Newton módszerek – BFGS

- A Hesse mátrix inverzét közelítik
- BFGS algoritmus ú.n. secant feltétele:
  - $\nabla J^{(2)}_{\boldsymbol{\theta}_k}\left\{\boldsymbol{\theta}_k\right\} = \nabla J\left\{\boldsymbol{\theta}_k\right\}$ és $\nabla J^{(2)}_{\boldsymbol{\theta}_k}\left\{\boldsymbol{\theta}_{k-1}\right\} = \nabla J\left\{\boldsymbol{\theta}_{k-1}\right\}$
  - Első feltétel $J^{(2)}_{\boldsymbol{\theta}_k}\left\{\cdot\right\}$ definíciója miatt teljesül
  - Második feltétel kifejtését követően:

$$\nabla J\left\{\boldsymbol{\theta}_{k-1}\right\} = \nabla J\left\{\boldsymbol{\theta}_k\right\} + \mathbf{H}\left\{\boldsymbol{\theta}_k\right\}\cdot\left(\boldsymbol{\theta}_{k-1} - \boldsymbol{\theta}_k\right)$$

  - Tehát a secant feltétel igaz, ha

$$\mathbf{H}\left\{\boldsymbol{\theta}_k\right\}^{-1}\cdot\left(\nabla J\left\{\boldsymbol{\theta}_{k-1}\right\} - \nabla J\left\{\boldsymbol{\theta}_k\right\}\right) = \left(\boldsymbol{\theta}_{k-1} - \boldsymbol{\theta}_k\right)$$

Kvázi Newton módszerek – BFGS

- Mit tudunk eddig $\mathbf{H}\left\{\boldsymbol{\theta}_k\right\}^{-1}$-ről:
  - Szimmetrikus, hiszen szimmetrikus mátrix inverze
  - $\mathbf{H}\left\{\boldsymbol{\theta}_k\right\}^{-1} \cdot \left(\nabla J\left\{\boldsymbol{\theta}_{k-1}\right\} - \nabla J\left\{\boldsymbol{\theta}_k\right\}\right) = \left(\boldsymbol{\theta}_{k-1} - \boldsymbol{\theta}_k\right)$
  - De ilyenből végtelen van – „regularizáció” kell:
    - Éljünk a simasági feltételezéssel ($\mathbf{H}\left\{\cdot\right\}$ folytonos)
    - $\mathbf{H}\left\{\boldsymbol{\theta}_k\right\}^{-1} = \arg\min_{\mathbf{H}^{-1}} \left\{ \left\| \mathbf{H}^{-1} - \mathbf{H}\left\{\boldsymbol{\theta}_{k-1}\right\}^{-1} \right\|_F^2 \middle| \ldots \right\}$
- Szerencsére analitikusan megoldható

Kvázi Newton módszerek – BFGS

- BFGS frissítés tulajdonságai:
  - Ha $\mathbf{H}\left\{\boldsymbol{\theta}_0\right\}$ -t diagonálisnak választjuk, akkor $\Delta\boldsymbol{\theta}_k$ számítása $O\left(k \cdot size\left(\boldsymbol{\theta}\right)\right)$ komplexitású.
  - Ehhez viszont szükség van $\nabla J\left\{\boldsymbol{\theta}_k\right\}$ -re és $\boldsymbol{\theta}_k$ -ra, minden $i = 0,\ldots,k$ esetén
    - Nagyméretű problémák esetén ez már problémát jelent (memória).
  - Memória használata túl nagy:
    - Pl. 1E5 iteráció, és 1E7 paraméter esetén ~1 TB

Kvázi Newton módszerek – L-BFGS

- Belátható, hogy konvergál az optimumhoz
  - De esetenként jóval lassabban, mint a Newton módszer – tipikusan pár tíz/százezer iteráció kell
  - Skálázásra érzékeny (sok numerikus művelet)
  - Mini Batch módszerek problémásak
- L(imitált memóriahasználatú)-BFGS:
  - Bár a közelített inverz Hesse kiszámításához szükség lenne az összes addigi gradiensre és paraméterre, ezt csak az utolsó *m* db alapján becsli.
  - Matematikai bizonyítása a konvergenciának nincs, de praktikusan konvergens (itt már jelentősen függ a szükséges iterációk száma a változók számától).
  - Mini Batch-es tanítást ez sem igazán kedveli

Kvázi Newton módszerek – Konjugált Gradiens módszer

- Lineáris egyenletek megoldására találták ki, most $\boldsymbol{\theta}^* = -\mathbf{H}^{-1} \cdot \nabla J\left\{\boldsymbol{\theta}_0\right\}$-t keressük iteratívan.
- Alapötlet – lineáris egyenlet megoldása:
  - $\alpha^* = \arg\min_{\alpha}\left\{\left\|\mathbf{H} \cdot \left(\boldsymbol{\theta}_{k-1} + \alpha \cdot \Delta\boldsymbol{\theta}_{k-1}\right)\right\|_2\right\}$ esetén

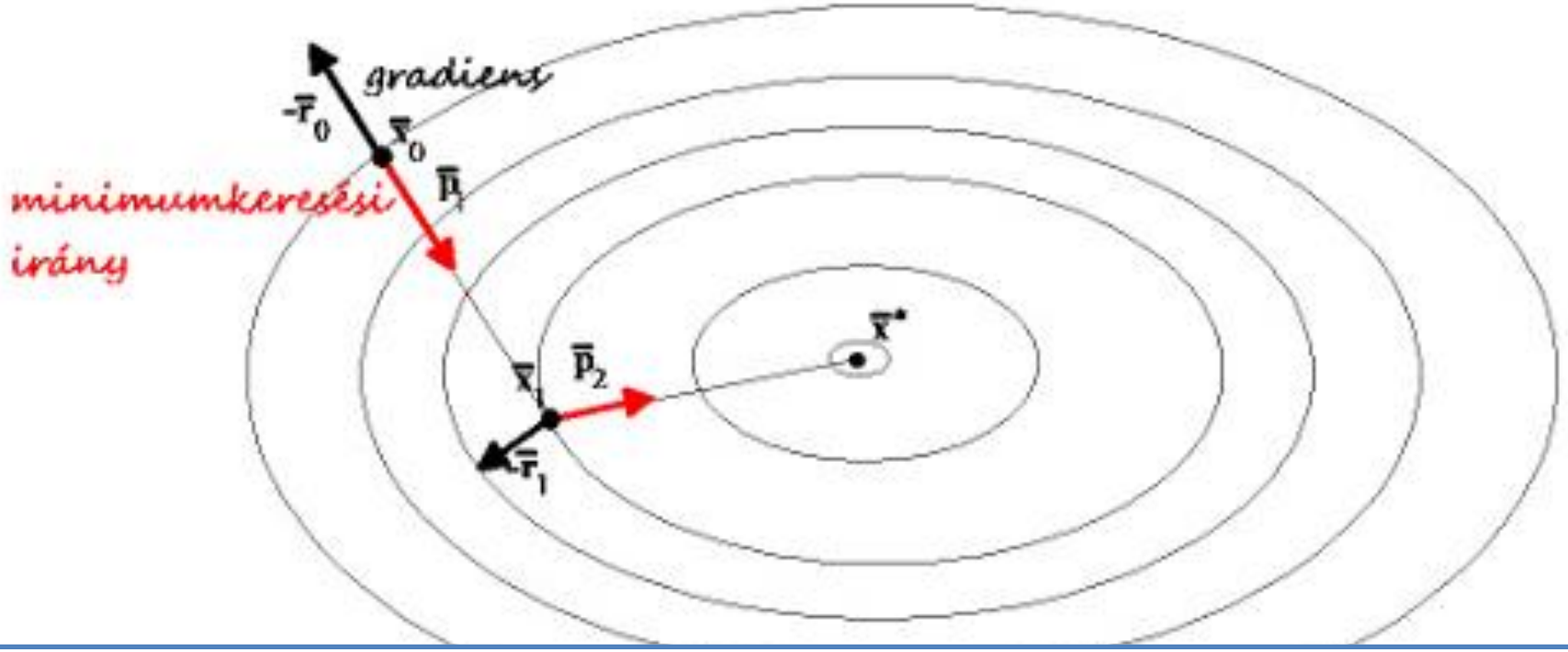
$$\Delta\boldsymbol{\theta}_{k-1}{}^T \cdot \left(\nabla J\left\{\boldsymbol{\theta}_0\right\} - \mathbf{H} \cdot \left(\boldsymbol{\theta}_{k-1} + \alpha^* \cdot \Delta\boldsymbol{\theta}_{k-1}\right)\right) =$$

$$= \Delta\boldsymbol{\theta}_{k-1}{}^T \cdot \mathbf{H} \cdot \left(\boldsymbol{\theta}^* - \boldsymbol{\theta}_k\right) = \mathbf{0}$$

Tehát az *k*-adik javítóirány $\mathbf{H}$ feletti konjugáltja a (*k-1*)-ediknek, azaz $\Delta\boldsymbol{\theta}_{k-1}{}^T \cdot \mathbf{H} \cdot \Delta\boldsymbol{\theta}_k = \mathbf{0}$

Kvázi Newton módszerek – Konjugált Gradiens módszer

- Mint lineáris egyenlet megoldó: $\mathbf{x}^* = \mathbf{A}^{-1} \cdot \mathbf{b}$

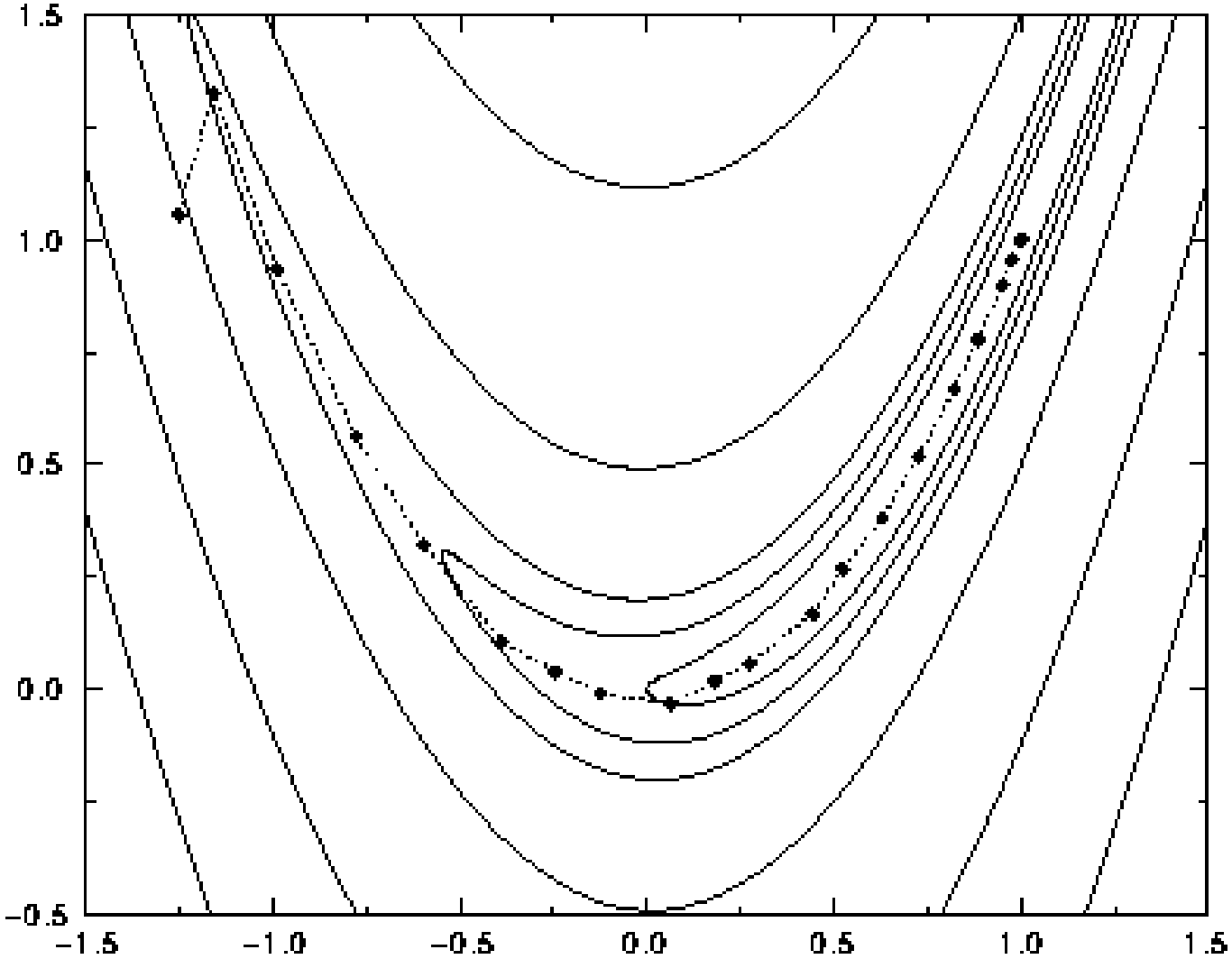


$\mathbf{p}_i$: i-edik keresési irány, $\mathbf{x}_i$: i-edik becslése $\mathbf{x}^*$-nak

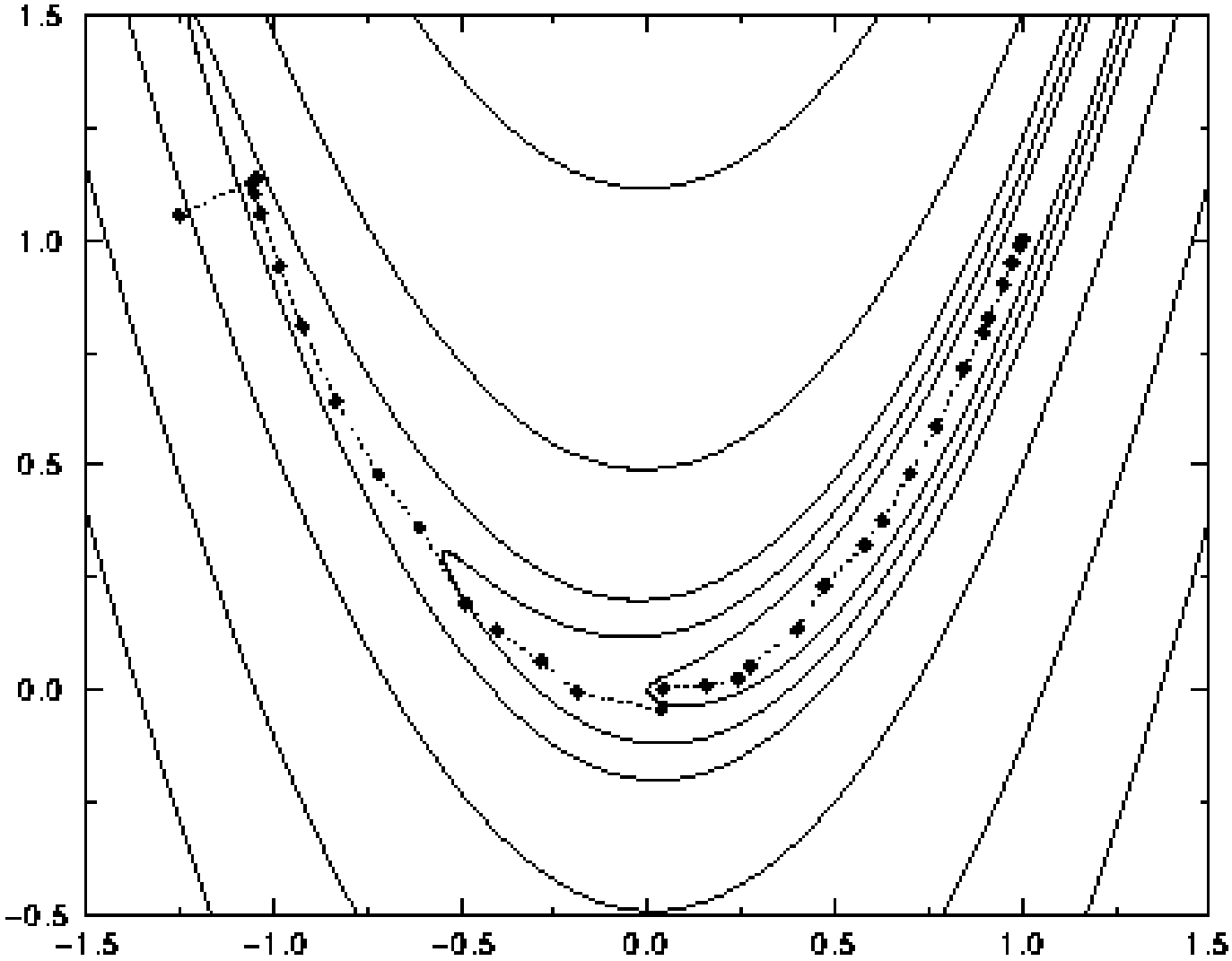
Kvázi Newton módszerek – Konjugált Gradiens módszer

- Mint veszteségfüggvény minimalizáló eljárás:
  - Aktuális pontbeli gradiensből az előző javítóiránnyal párhuzamos komponenst levonva áll elő az új javítóirány: $\Delta\boldsymbol{\theta}_k = -\nabla J\left(\boldsymbol{\theta}_{k-1}\right) - \beta \cdot \Delta\boldsymbol{\theta}_{k-1}$
  - Javító lépés nagysága: $\arg\min_{\alpha}\left\{ J\left(\boldsymbol{\theta}_{k-1} + \alpha \cdot \Delta\boldsymbol{\theta}_k\right)\right\}$
  - Nem igényli a Hesse mtx. kiszámítását
- Konvergencia:
  - Nagyságrendileg Newton iterációk száma≈60 × változók száma iterációra van szükség

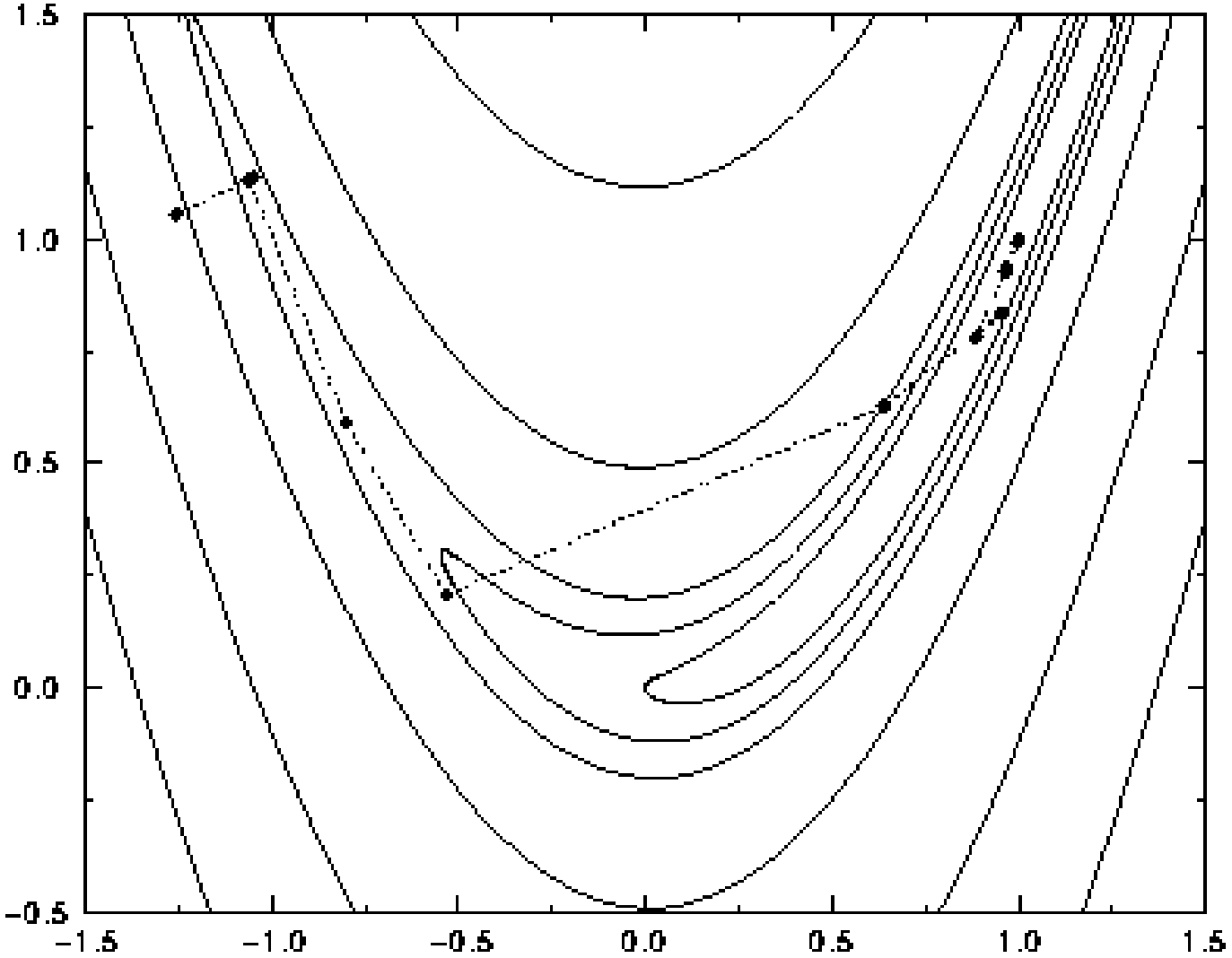
Futási példák – Csonkolt Newton



Futási példák – BFGS



Futási példák – CG



Newton / kvázi Newton módszerek

- CG-t főleg viselkedése és komplexitása miatt soroltuk ide (mindenhol u.a. $\mathbf{H}$ feltételezése).
- Általában bonyolultabb iterációk:
  - De jóval kevesebb elég belőle Batch tanításnál
  - Mini-batch/ LMS esetén viszont nem éri meg
- Klasszikus neurális hálóknál fontos szerep:
  - Levenberg Marquadt algoritmus – „Newton módszer és elsőrendű módszer keveréke”

Elsőrendű módszerek

- Gradiens irányába lépnek: $\Delta\boldsymbol{\theta} = -\nabla J\left\{\boldsymbol{\theta}_{k-1}\right\}$
- Mikor pontosabb a másodfokú közelítés?

  $$\mathbf{0} = \nabla J^{(2)}_{\boldsymbol{\theta}_{k-1}}\left\{\boldsymbol{\theta}_{k-1} + \Delta\boldsymbol{\theta}\right\} \Leftrightarrow \Delta\boldsymbol{\theta} = -\nabla J\left\{\boldsymbol{\theta}_{k-1}\right\} \cdot \mathbf{H}\left\{\boldsymbol{\theta}_{k-1}\right\}^{-1}$$

  – Ha $\mathbf{H}\left\{\boldsymbol{\theta}_{k-1}\right\}^{-1} = \mathbf{I} \Leftrightarrow \Delta\boldsymbol{\theta} = -\nabla J\left\{\boldsymbol{\theta}_{k-1}\right\}$

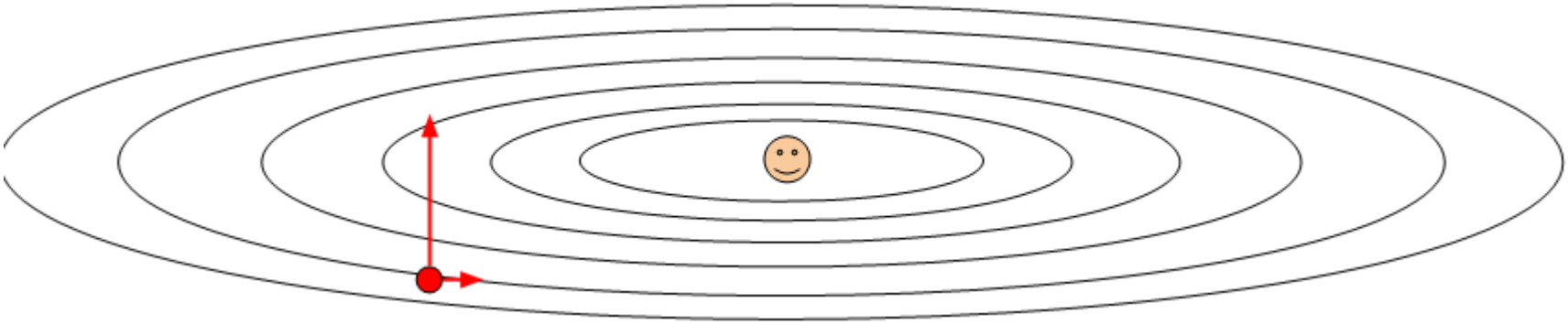
  – Általánosságban is igaz, hogy „minél izotropikusabb” a hibafelület, annál jobbak az elsőrendű módszerek
- Kérdés, mi a helyzet, ha $\mathbf{H}\left\{\boldsymbol{\theta}_{k-1}\right\}$ egy forgatás?

Gradiens módszer

- Gradienssel ellentétes irányba lépünk:
  - $\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} - \alpha \cdot \nabla J\left\{\boldsymbol{\theta}_{k-1}\right\},\ \alpha \in \mathbb{R}_+$ bátorsági tényező
  - Konvex függvények esetén meghatározható az α :
    - Polyak módszer : $\alpha_k = \left(J\left\{\boldsymbol{\theta}_k\right\} - J\left\{\boldsymbol{\theta}^*\right\}\right) \Big/ \left\|\nabla J\left\{\boldsymbol{\theta}_k\right\}\right\|_2^2$
    - Léteznek line search algoritmusok (pl. Armijo)
    - Gyakori az $\alpha_k = \alpha_0 \cdot \beta^{k-1}$ választás (pl. $\beta = 0.99$)
- Problémás, ahol $J\left\{\cdot\right\}$ rosszul kondicionált:
  - Pl. szűk völgyek, elnyújtott hibafelület
  - Azaz kb. ahol $\min_{\beta}\left\{\left\|\mathbf{H} - \beta \cdot \mathbf{I}\right\|_F\right\}$ túl nagy

Gradiens módszer szemléltetés

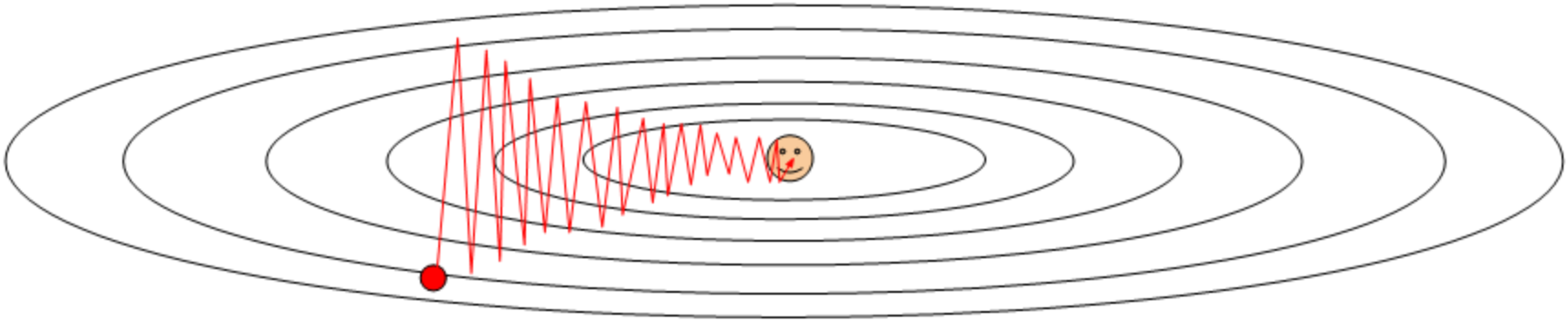
- Pl. egy vízszintesen elnyújtott, kvadratikus $J\{\cdot\}$



  - Gradiens vektor függőleges tengelyre eső vetülete jóval nagyobb, mint a vízszintes tengelyre eső vetülete
  - pedig vízszintes irányban vagyunk messze a minimumtól

Gradiens módszer szemléltetés

- Pl. egy vízszintesen elnyújtott, kvadratikus $J\{\cdot\}$

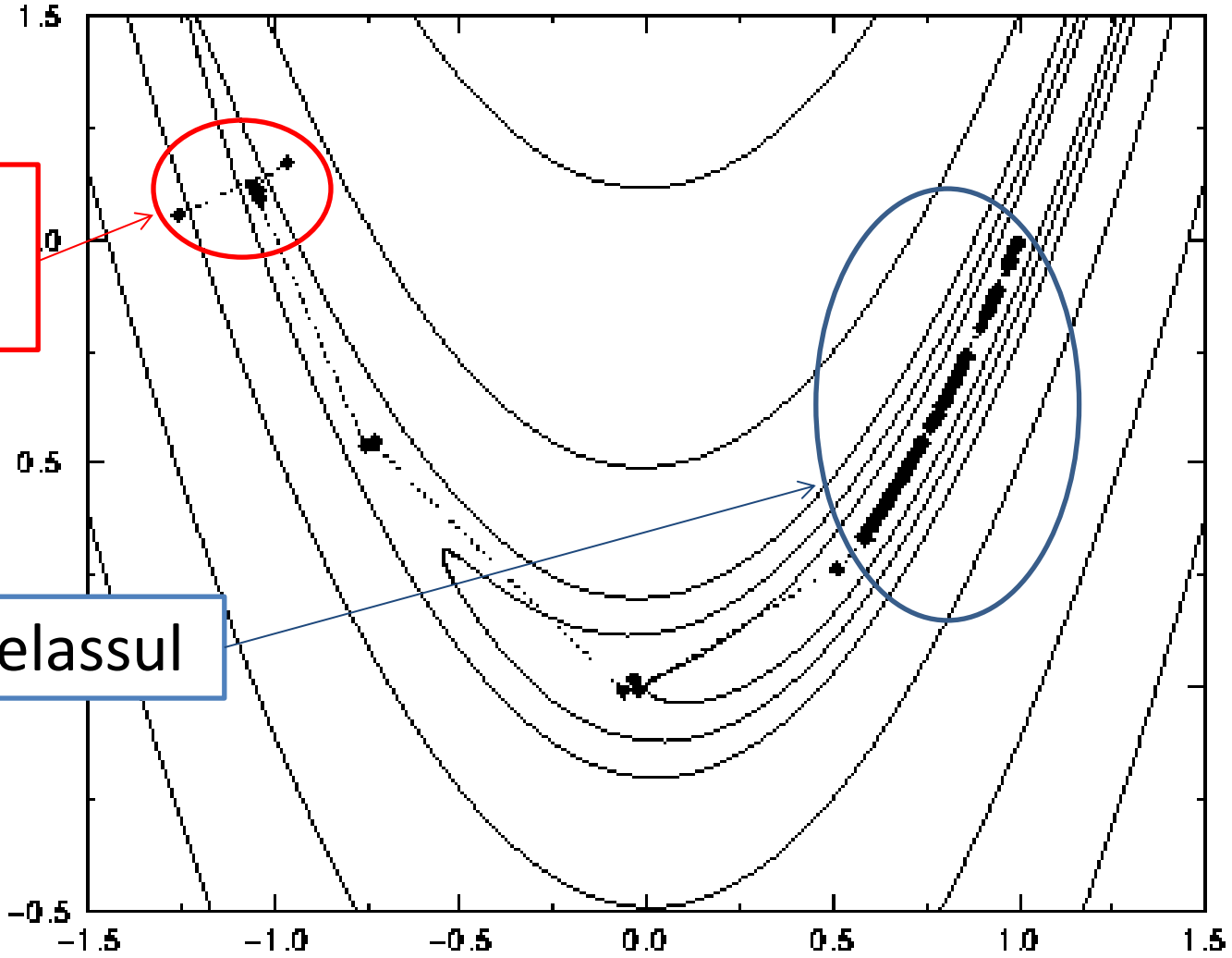


  - Lassú, kis amplitúdójú lépések vízszintesen
  - Nagy lépések függőlegesen, ami cikk-cakk jellegű becslővektor trajektóriához vezet

Gradiens módszer szemléltetés

Szűk völgy körül oszcillál

Lapos felületen lelassul



Gradiens módszer konvergenciája

- Belátható, hogy minden esetben létezik olyan bátorsági tényező, mellyel konvergál
  - Akár fix bátorsági tényező esetén is
  - De a konvergencia sebessége nagyon függ a hiba-felülettől (ellentétben a Newton módszerekkel)
  - A paramétertér affin transzformációja érzékeny
- Jól tűri a sztochasztikus jelleget is
  - Alkalmas mini batch-re (pl. LMS)
  - 1 valószínűséggel konvergál, de jóval több lépés kell

Momentum módszerek

- Első interpretáció – alul-áteresszük $\boldsymbol{\theta}_k$-at:
  - Ezzel a cikk-cakkos ugrásokat kiküszöbölhetjük
  - $\Delta\boldsymbol{\theta}_k{}' = \beta \cdot \Delta\boldsymbol{\theta}_{k-1}{}' + (1-\beta) \cdot \nabla J\left\{\boldsymbol{\theta}_{k-1}\right\} \quad \beta \in (0,1)$
  - $\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} - \alpha \cdot \Delta\boldsymbol{\theta}_k{}'$
- Másik interpretáció – fizikai modell:
  - $\mathbf{v}_k = \mu \cdot \mathbf{v}_{k-1} - \alpha \cdot \nabla J\left\{\boldsymbol{\theta}_{k-1}\right\} \quad \mu \in (0.5\,, 1)$

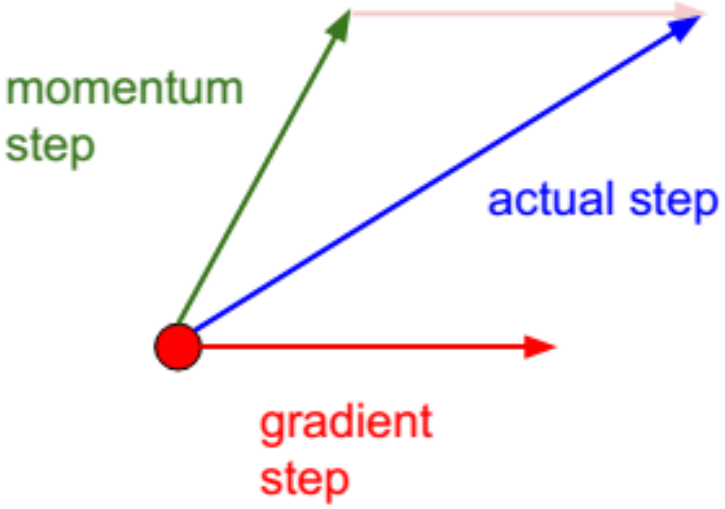
    Integráljuk a gyorsulást ($-\alpha \cdot \nabla J\left\{\boldsymbol{\theta}_{k-1}\right\}$), μ súrlódással
  - $\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} + \mathbf{v}_k$

    Integráljuk a sebességet

Nesterov Momentum

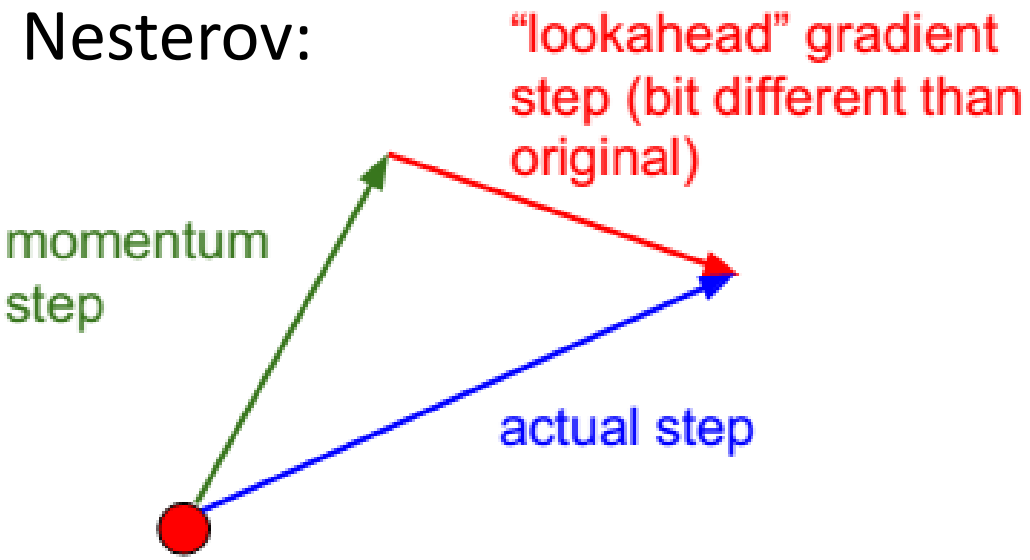
- Ha úgyis ellépünk $\mu \cdot \mathbf{v}_{k-1}$ irányba, akkor az ottani gradiens irányába lépjünk tovább:

Egyszerű momentum:



$$\mathbf{v}_k = \mu \cdot \mathbf{v}_{k-1} - \alpha \cdot \nabla J\left\{\boldsymbol{\theta}_{k-1}\right\}$$

Nesterov:



$$\mathbf{v}_k = \mu \cdot \mathbf{v}_{k-1} - \alpha \cdot \nabla J\left\{\boldsymbol{\theta}_{k-1} + \mu \cdot \mathbf{v}_{k-1}\right\}$$

Momentum módszerek

- Sima momentum hasonló a CG-vel:
  - Csak itt hasból választjuk α-t, μ-t
- Gyakorlatban jobbak a gradiens módszernél
  - Elméletileg ez nehezen látható be
  - A Nesterov momentum viszont elméletileg is gyorsabb a sima momentumnál
- Jellemző az állapot-trajektóriára a túllövés:
  - Nagy amplitúdójú gradiensek sokáig magukkal húzzák az azt követő javító lépéseket

Adaptive Gradient (AdaGrad)

- A hibafelület alakjához próbálunk adaptálódni / dimenziónként skálázunk:
  - k-adik lépésnél $\nabla J\left\{\boldsymbol{\theta}_i\right\} \quad i=1,2,\ldots,k-1$ alapján becsüljük a dimenziónkénti elnyújtottságot:
    $$\left(\boldsymbol{\tau}_k\right)_{(i)} = \sum_{j=1}^{k-1} \nabla J\left\{\boldsymbol{\theta}_j\right\}_{(i)}^2$$
  - Majd ezzel normálunk (elkerülve a 0-val osztást):
    $$\Delta\boldsymbol{\theta}_{k(i)} = -\nabla J\left\{\boldsymbol{\theta}_{k-1}\right\}_{(i)} \Big/ \left(\sqrt{\left(\boldsymbol{\tau}_k\right)_{(i)}} + \varepsilon\right)$$
  - Sok iteráció után már túl kicsik a lépések:
    - Ez sima felületeknél általában nem baj, de neurális hálókra tipikusan nem ez a jellemző

RMSProp

- Ötlet: összegzés helyett mozgó átlag alapján becsüljük a hibafelületet
  - $\left(\boldsymbol{\tau}_k\right)_{(i)} = \beta \cdot \left(\boldsymbol{\tau}_{k-1}\right)_{(i)} + \left(1-\beta\right) \cdot \nabla J\left\{\boldsymbol{\theta}_k\right\}_{(i)}^{\ 2} \qquad \beta \in \left(0\,,1\right)$
  - Mozgó átlagolás miatt elfelejti a hibafelület távolabbi részeit
  - Ráadásul nem nő minden határon túl
  - $\left(\Delta\boldsymbol{\theta}_k\right)_{(i)} = -\nabla J\left\{\boldsymbol{\theta}_{k-1}\right\}_{(i)} \Big/ \left(\sqrt{\left(\boldsymbol{\tau}_k\right)_{(i)}} + \varepsilon\right)$
    - Ebből jön az elnevezésbeli RMS
  - Mély NN esetén általában jobb az AdaGrad-nál

Adaptive Momentum (Adam)

- Kombináljuk az RMSProp-ot a Momentummal:
  1. $\mathbf{v}_k = \beta_1 \cdot \mathbf{v}_{k-1} + (1-\beta_1) \cdot \nabla J\{\boldsymbol{\theta}_k\} \qquad \beta_1 \in (0,1)$
  2. $(\boldsymbol{\tau}_k)_{(i)} = \beta_2 \cdot (\boldsymbol{\tau}_{k-1})_{(i)} + (1-\beta_2) \cdot \nabla J\{\boldsymbol{\theta}_k\}_{(i)}^2 \quad \beta_2 \in (0,1)$
  3. $(\boldsymbol{\theta}_k)_{(i)} = (\boldsymbol{\theta}_{k-1})_{(i)} - \alpha \cdot (\mathbf{v}_k)_{(i)} \Big/ \left(\sqrt{(\boldsymbol{\tau}_k)_{(i)}} + \varepsilon\right) \quad \alpha \in \mathbb{R}_{++}$
- Ötvözi a két módszer előnyét:
  - Simítja a lépési irányt, mint a Momentum
  - Adaptálódik a hibafelülethez, mint az RMSprop
  - Mély hálók tanításának standard módszere

Módszerek szemléltetése

- Kvadratikus hibafelület:

http://home.mit.bme.hu/~hadhazi/Oktatas/NN/1.gif

- Összetettebb hibafelület:

http://home.mit.bme.hu/~hadhazi/Oktatas/NN/2.gif

- Nyeregpont:

http://home.mit.bme.hu/~hadhazi/Oktatas/NN/4.gif

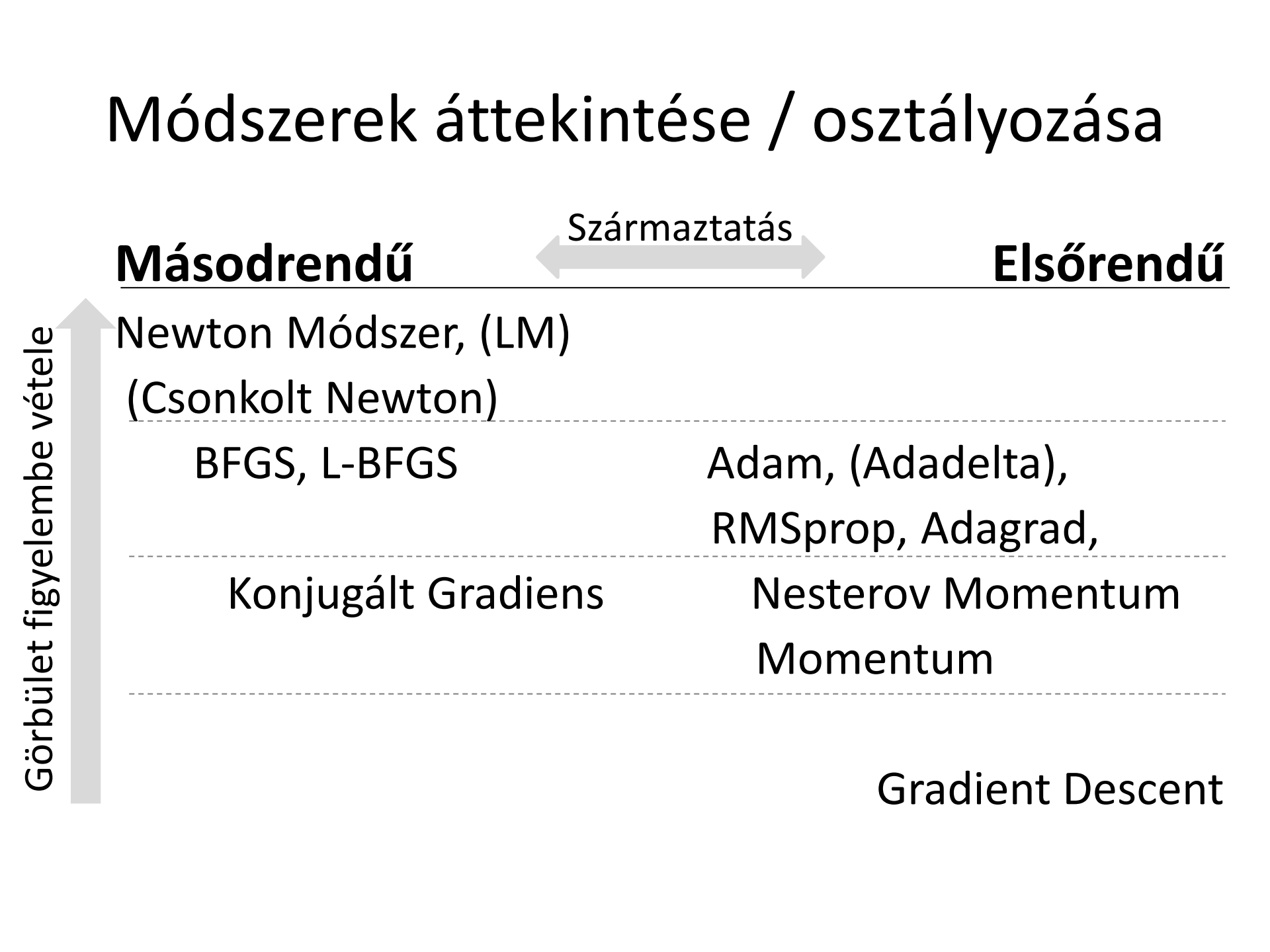
- Elnyúló völgy:

http://home.mit.bme.hu/~hadhazi/Oktatas/NN/3.gif

Elsőrendű módszerek összegzés

- Milyen az ideális optimalizáló módszer?
  - Azon irány mentén mozog gyorsan, melyre eső gradiens vetület stabil (még akkor is, ha ez kicsi)
  - Az instabil gradiens vetületű irány mentén megfontoltabb (nagy amplitúdó esetén is)
- Különböző változatok a hibafelület lokális görbületét próbálják becsülni a gradiens vektorokból
  - Jelentős hasonlóság a BFGS / CG módszerekkel
- Sokat segíthet az előfeldolgozás – cél az izotrópia
  - Bemenetenkénti normalizálás
  - Bemenetek de-korrelációja (pl. PCA, igaz sok változó esetén nem egyszerű)

Módszerek áttekintése / osztályozása



Általános megfontolások -1-

- Klasszikus NN-ek esete:
  - Általában kevésbé „redundáns” tanítóminták
  - Egy epoch-ban minden mintát felhasználunk
  - Másodrendű módszerek jól használhatóak
- Mély NN-ek esete:
  - Sok „redundáns” tanítóminta
  - Érdemes a mintákat mini-batch-ekre partícionálni
    - Egy partícióba kevésbé korrelált minták essenek
    - Fontos, hogy kiegyensúlyozottak legyenek a bacth-ek

Általános megfontolások -2-

- Mini-batch mérete általában nagy legyen
  - Szélsőséges eset (1 minta) on-line tanulás
- Másodrendű módszerek kevésbé hatékonyak:
  - Bonyolultabb a hibafelület, kvadratikus approx.-ból származó javítóirányok csak kicsit jobbak a gradiensnél
  - Gyakorlatban tetszőlegesen kis hibát a tanítómintákon el lehet velük érni (de ez gépi tanulásnál kb. lényegtelen)

- Hibafelület alakja:
  - Lineáris neuron esetén kvadratikus
  - NN / DNN esetén lokálisan kvadratikus (lsd. RELU)

Felhasznált források

- Felhasznált források:

1. Stanford CS231n - http://cs231n.stanford.edu/syllabus.html
2. Denzi Yuret - Alec Radford's animations for optimization algorithms http://www.denizyuret.com/2015/03/alec-radfords-animations-for.html
3. Horváth Gábor (szerk.) : Neurális hálózatok - 2.5.1 alfejezet https://www.mit.bme.hu/books/neuralis/ch02s05
4. Faragó István – Horváth Róbert : Numerikus módszerek, Typotex (2013) http://tankonyvtar.ttk.bme.hu/pdf/30.pdf
5. Stephen Boyd and Lieven Vandenberghe : Convex Optimization, Cambridge University Press (2009) http://stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf
6. Toronto University CSC321 – Lect6: http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf