Optimalizáló algoritmusok

Baran Ágnes

Előadás

Sztochasztikus optimalizálás

A feladat:
Keresett

$$\min_{x\Omega} f(x),$$

ahol $f : \mathbb{R}^n \to \mathbb{R}$, $\Omega \subset \mathbb{R}^n$.
Sztochasztikus optimalizálásról beszélünk, ha

- az $f(x)$ függvény csak zajjal terhelten figyelhető meg, azaz csak $y(x) = f(x) + \varepsilon(x)$ ismert, ahol ε egy véletlen zaj,
- a keresési irányokat véletlenszerűen választjuk.

Determinisztikus esetben az f teljesen ismert és a keresési irányokat determinisztikusan választjuk.

Sztochasztikus optimalizálás. Egy egyszerű megoldás.

egyenletes eloszlás

Ha Ω korlátos, akkor generáljunk $u_1, \dots, u_m \sim U(\Omega)$ értékeket.
Keresett $f_m^* = \min\{f(u_1), \dots, f(u_m)\}$.
Ez tart $f^* = f(x^*)$-hoz, de nagyon lassan.

Példa. Keresett a következő függvény minimuma:

$f: \mathbb{R} \to \mathbb{R}$

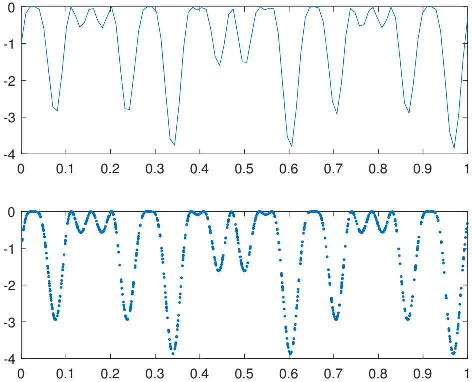
$$f(x) = -(\cos(10x) - \sin(60x))^2, \quad x \in [0, 1]$$

A valódi minimum: $f^* = -3.868428$ az 0.3396384 és 0.6028403 helyeken.
Közelítések:

m	100	500	1000	5000
f_m^*	-3.8303	-3.8667	-3.8680	-3.8684

Példa.

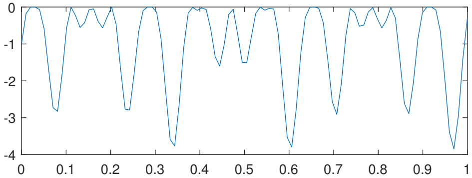
$$f(x) = -(\cos(10x) - \sin(60x))^2, \quad x \in [0, 1]$$

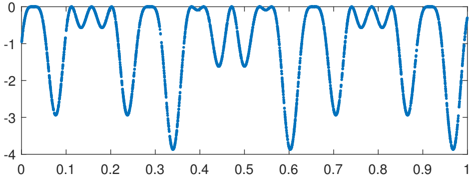


Az $f(x)$ és $f(u)$ függvények, ahol $m = 1000$

Példa.

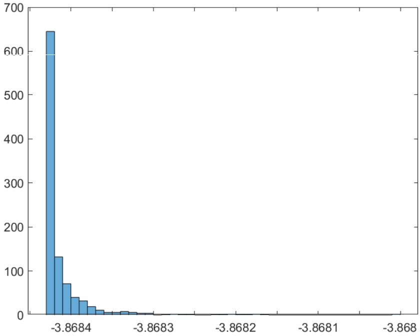
$$f(x) = -(\cos(10x) - \sin(60x))^2, \quad x \in [0,1]$$





Az $f(x)$ és $f(u)$ függvények, ahol $m = 5000$

$$f(x) = -(\cos(10x) - \sin(60x))^2, \quad x \in [0,1]$$



1000 tesztfutás eredménye ($m = 5000$). A valódi minimum -3.868428.

Vakkeresés

Az algoritmus:

- Válasszunk egy $x_0 \in \Omega$ kezdőpontot (véletlenszerűen vagy determinisztikusan)
- $x_k \to x_{k+1}$: generáljunk egy $x_n \in \Omega$ véletlen, független pontot egy adott eloszlás segítségével. Ha $f(x_n) < f(x_k)$, akkor legyen $x_{k+1} = x_n$, egyébként $x_{k+1} = x_k$.
- Leállás: ha elértük a maximális iterációszámot (vagy elégedettek vagyunk az aktuális közelítőértékkel)

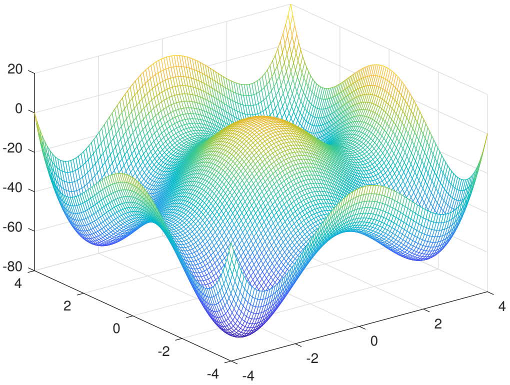
Az algoritmus elegendően általános feltételek mellett majdnem biztosan konvergál x^*-hoz, de csak alacsony dimenzióban hatékony.

Vak: az új pont generálásánál teljesen figyelmen kívül hagyja az x^* korábbi közelítéseit.

Tesztfutások

$f: \mathbb{R}^2 \to \mathbb{R}$

$$f(x) = (x_1^4 - 16x_1^2 + 5x_1)/2 + (x_2^4 - 16x_2^2 + 5x_2)/2;$$

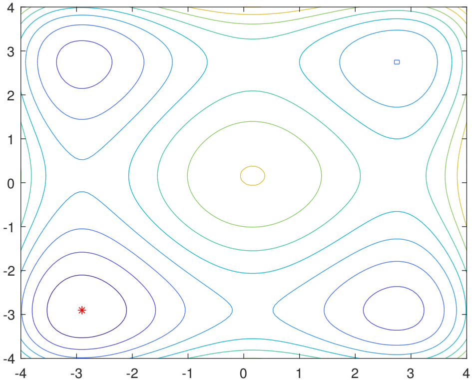


$[-4, 4] \times [-4, 4]$

$x^* = [-2.9035, -2.9035]$

Tesztfutások

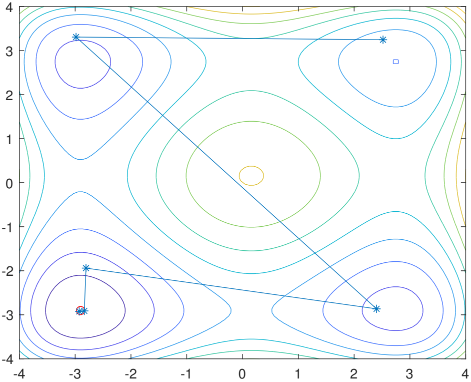
$$f(x) = (x_1^4 - 16x_1^2 + 5x_1)/2 + (x_2^4 - 16x_2^2 + 5x_2)/2;$$



$x^* = [-2.9035, -2.9035]$

Tesztfutások

$$f(x) = (x_1^4 - 16x_1^2 + 5x_1)/2 + (x_2^4 - 16x_2^2 + 5x_2)/2;$$



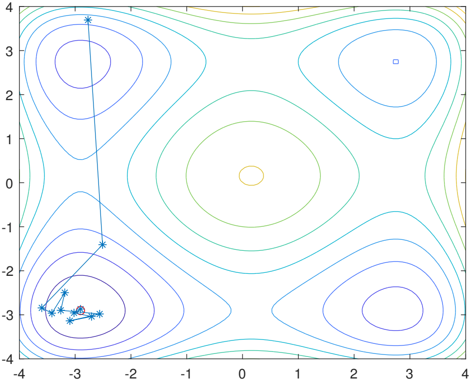
(5000 iteráció)

az elfogadott pontok száma

$x^* = [-2.9035, -2.9035]$, $x_{opt} = [-2.9276, -2.9233]$ $\boxed{k = 6}$

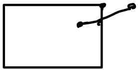
Tesztfutások

$$f(x) = (x_1^4 - 16x_1^2 + 5x_1)/2 + (x_2^4 - 16x_2^2 + 5x_2)/2;$$



$$x^* = [-2.9035, -2.9035],\ x_{opt} = [-2.9028, -2.8707], \boxed{k = 11}$$

Lokalizált véletlen keresés



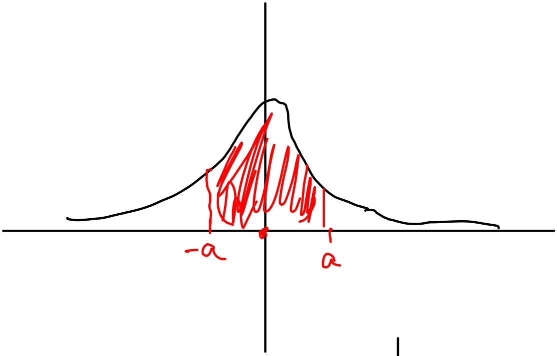
Az új pontot az eddigi legjobb közelítés egy környezetéből választjuk.

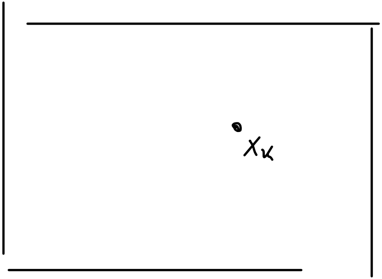
Az algoritmus:

- Válasszunk egy $x_0 \in \Omega$ kezdőpontot (véletlenszerűen vagy determinisztikusan)
- $x_k \to x_{k+1}$: generáljunk egy véletlen irányt $d_k \in \mathbb{R}^n$, ha $x_k + d_k \notin \Omega$ akkor generáljunk egy új d_k irányt, ezt ismételjük (vagy válasszuk az Ω legközelebbi pontját). Ha $x_k + d_k \in \Omega$ akkor legyen $x_n = x_k + d_k$.
- Ha $f(x_n) < f(x_k)$, akkor legyen $x_{k+1} = x_n$, egyébként $x_{k+1} = x_k$.
- Leállás: ha elértük a maximális iterációszámot (vagy elégedettek vagyunk az aktuális közelítőértékkel)

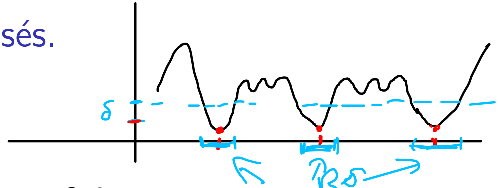
A d_k irány generálásánál egy 0 várható értékű normális eloszlást használjunk. A szórásokat válasszuk az aktuális x_k komponenseinek nagyságával összhangban.

Az 1 dim normális eloszlás sűrűségfv.
0 várható értékű)





Lokalizált véletlen keresés.



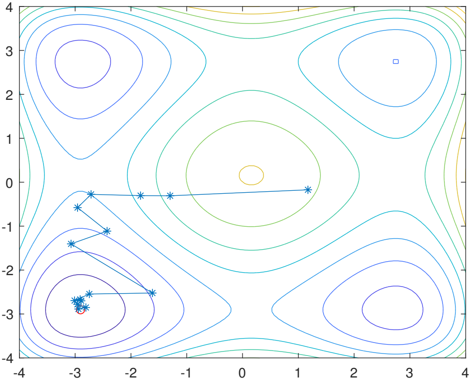
Tétel. Legyen Ω^* az f függvény Ω feletti globális minimumhelyeinek a halmaza. Tegyük fel, hogy f folytonos, és ha $x_k + d_k \notin \Omega$ akkor az új d_k irányt véletlenszerűen választjuk. Adott $\delta > 0$ esetén legyen

$$R_\delta = \bigcup_{x \in \Omega^*} \{x \in \Omega : |f(x) - f(x^*)| < \delta\}.$$

Ekkor ha a d_k sorozat elemei független, azonos $\mathcal{N}(0, I)$ eloszlásúak, akkor $\lim\limits_{k\to\infty} P(x_k \in R_\delta) = 1$.

Tesztfutások

$$f(x) = (x_1^4 - 16x_1^2 + 5x_1)/2 + (x_2^4 - 16x_2^2 + 5x_2)/2;$$



elfogadott pontok száma

$$x^* = [-2.9035, -2.9035],\ x_{opt} = [-2.9416, -2.8864],\ \boxed{k = 15}$$

Gradiens módszer

Determinisztikus esetben: $x_{k+1} = x_k - \alpha_k \nabla f(x_k)$

Ha f nem differenciálható, vagy az f-et nem ismerjük analitikus alakban, akkor

$$x_{k+1} = x_k - \underline{\alpha_k} \widehat{\nabla} f(x_k).$$

Legyen most

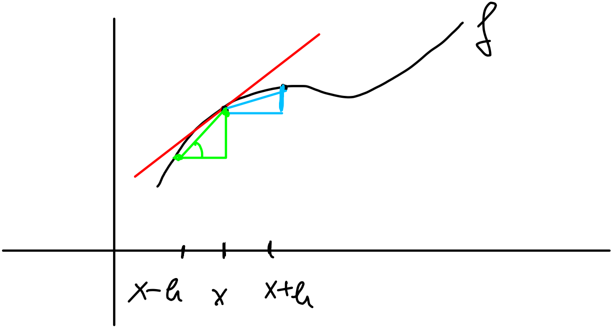
$$\widehat{\nabla} f(x_k) = \begin{pmatrix} \frac{f(x_k+\beta_k\xi_1)-f(x_k-\beta_k\xi_1)}{2\beta_k} \\ \vdots \\ \frac{f(x_k+\beta_k\xi_n)-f(x_k-\beta_k\xi_n)}{2\beta_k} \end{pmatrix},$$

$$\begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

ahol ξ_i jelöli az i-edik egységvektort, és $\beta_k > 0$.
Az α_k és β_k egy tipikus megválasztása:

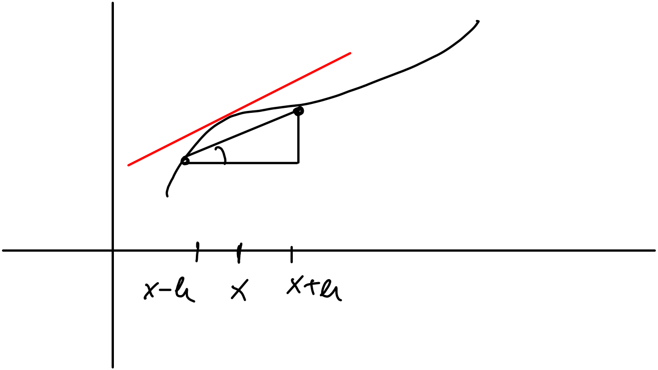
$$\alpha_k = \frac{\alpha}{(k+1+A)^a}, \quad \beta_k = \frac{\beta}{(k+1)^\gamma},$$

ahol A nemnegatív, a többi konstans pozitív.



$$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$ haladó diff.

$$f'(x) \approx \frac{f(x) - f(x-h)}{h}$$ retrográd diff.



$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}$$

centrál diff.

A gradiens módszer egy sztochasztikus módosítása

$$\nabla f(x_k) \approx \frac{f(x_k + \beta_k \zeta_k) - f(x_k - \beta_k \zeta_k)}{2\beta_k} \zeta_k = \frac{\Delta f(x_k, \beta_k \zeta_k)}{2\beta_k} \zeta_k$$

és

$$x_{k+1} = x_k - \frac{\alpha_k}{2\beta_k} \Delta f(x_k, \beta_k \zeta_k) \zeta_k,$$

gömb

ahol (β_k) egy csökkenő sorozat, míg ζ_k az $\|\zeta\| = 1$ egység~~körön~~n egyenletes eloszlású.

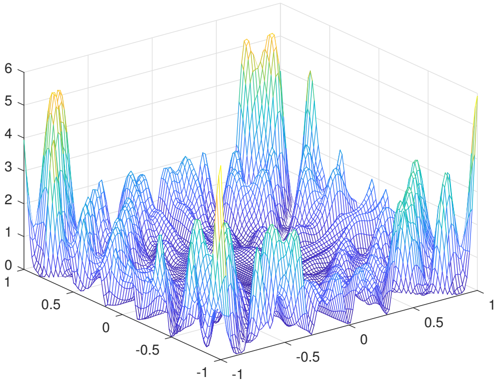
A gyakorlatban ajánlott egy skálázási lépés használata, mellyel elkerülhető a túl nagy lépés: legyen

$$g := \frac{\alpha_k}{2\beta_k} \Delta f(x_k, \beta_k \zeta_k) \zeta_k,$$

ha $\|g\| > 1$ akkor generáljunk egy új ζ_k-t.

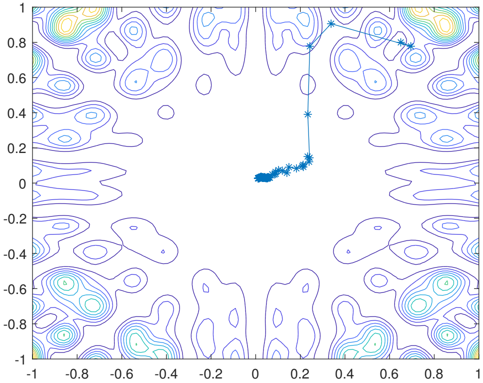
Példa.

$$f(x) = \left(x_1 \sin(20x_2) + x_2 \sin(20x_1)\right)^2 \cosh\left(\sin(10x_1)x_1\right)$$
$$+ \left(x_1 \cos(10x_2) - x_2 \sin(10x_1)\right)^2 \cosh\left(\cos(20x_2)x_2\right).$$



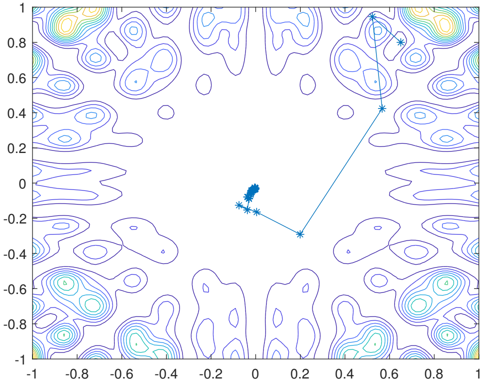
Az $f(x)$ függvény a $[-1,1]^2$ tartomány felett. Minimumhely: $(0,0)$.

Tesztfutások



$\alpha_k = \frac{1}{k+1}$, $\beta_k = \frac{1}{(1+k)^{0.5}}$, $x_{opt} = [0.0109, 0.0268]$, $k = 110$

Tesztfutások



$\alpha_k = \frac{1}{k+1}$, $\beta_k = \frac{1}{(1+k)^{0.5}}$, $x_{opt} = [-0.0042, 0.0260]$, $k = 53$

Sztochasztikus gradiens módszer

Az

$$x_{k+1} = x_k - \alpha_k \boxed{\nabla f(x_k)}$$

iteráció helyett az

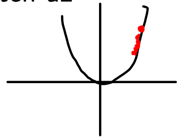
$$x_{k+1} = x_k - \alpha_k g(x_k, \xi_k),$$

közelítést használjuk, ahol ξ_k egy valószínűségi változó, és $\mathbb{E} g(x_k, \xi_k) = \nabla f(x_k)$.

($g(x_k, \xi_k)$ a $\nabla f(x_k)$ egy torzítatlan becslése)

Gépi tanulásban (SGD: stochastic gradient descent)

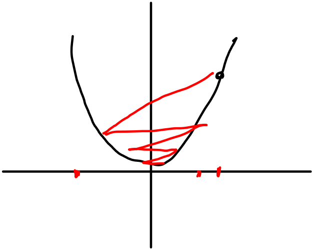
Adott $\{(x^{(i)}, d^{(i)}), i = 1, \ldots, M\}$ cimkézett tanulóhalmaz esetén az



$$L(\Theta) = \frac{1}{M} \sum_{i=1}^{M} L_i(\Theta),$$

veszteségfüggvényt akarjuk minimalizálni, ahol Θ a hálózat paramétereit tartalmazó vektor, míg L_i az i-edik tanulóadatnak megfelelő veszteség.

Gradiens módszer esetén a paraméter módosítása:



$$\Theta := \Theta - \frac{\eta}{M} \sum_{i=1}^{M} \nabla L_i(\Theta),$$

ahol η a tanulási paraméter (a lépéshossz).

MSE

$$L(\Theta) = \frac{1}{M} \sum_{i=1}^{M} \underbrace{\left(y^{(i)} - d^{(i)}\right)^2}_{L_i(\Theta)}$$

ahol $y^{(i)}$ az $x^{(i)}$-hez a háló által rendelt érték

Az **SGD** formula:

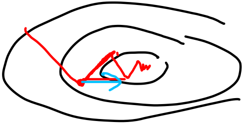
$$\Theta := \Theta - \eta \nabla L_i(\Theta).$$

1 epoch: amikor i minden értéket felvesz $\{1, ..., M\}$-ből (a tanulóhalmaz elemeit véletlenszerűen keverjük minden epoch előtt).

Mini-batch SGD:
az $\{1, \ldots, M\}$ indexhalmazt véletlenszerűen m elemű $M_1, \ldots, M_K$ részhalmazokra osztjuk, és

$$\Theta := \Theta - \frac{\eta}{m} \sum_{i \in M_j} \nabla L_i(\Theta).$$

1 epoch: amikor j minden értéket felvesz $\{1, ..., K\}$-ból.



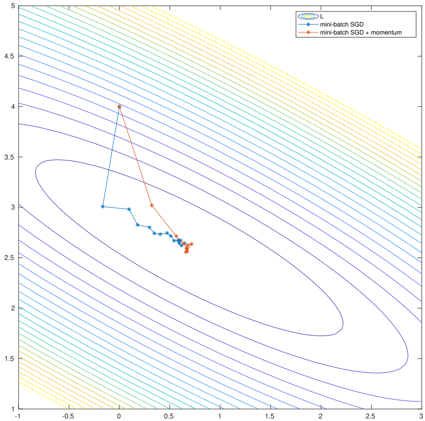
Momentum módszer

Az aktuális lépés~~hossz~~ az előző lépés~~hossz~~ és az aktuális gradiens lineáris kombinációja:

$$
\begin{aligned}
\Delta\Theta &:= \varepsilon \Delta\Theta - \eta G(\Theta), \\
\Theta &:= \Theta + \Delta\Theta,
\end{aligned}
$$

ahol $G(\Theta)$ az aktuális gradiens (a módszernek megfelelően).

“Játékpélda”



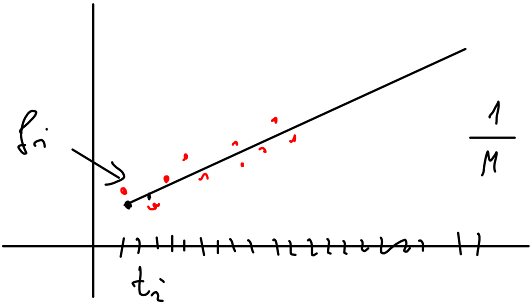
6 elemű
mini-batch

30 tanulóadat

⇒ epochonként

5 iteráció

Egyenes illesztése 30 generált, zajos, síkbeli pontra, a veszteségfüggvény az átlagos négyzetes hiba. A mini-batchek mérete 6, a momentum módszernél $\varepsilon = 0.9$. A tanulási paraméter állandó (nem ideális!) A közelítéseket az epochok végén ábrázoltuk.



$$y(t) = \underbrace{\boxed{a}}_{x_1} + \underbrace{\boxed{b}}_{x_2} \cdot t$$

$$\frac{1}{M} \sum_{i=1}^{M} \left(y(t_i) - f_i \right)^2$$

Példa

10 osztályos osztályozási feladat, 10x1000 kép, ennek 75%-ával tanítunk.

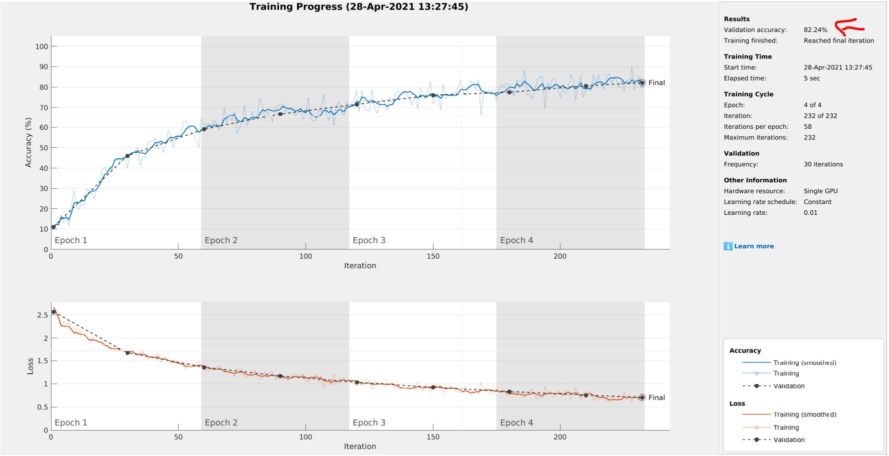
Veszteségfüggvény: keresztentrópia

(a) mini-batch SGD, momentum nélkül, mini-batch mérete 128, tanulási paraméter állandó (0.01)

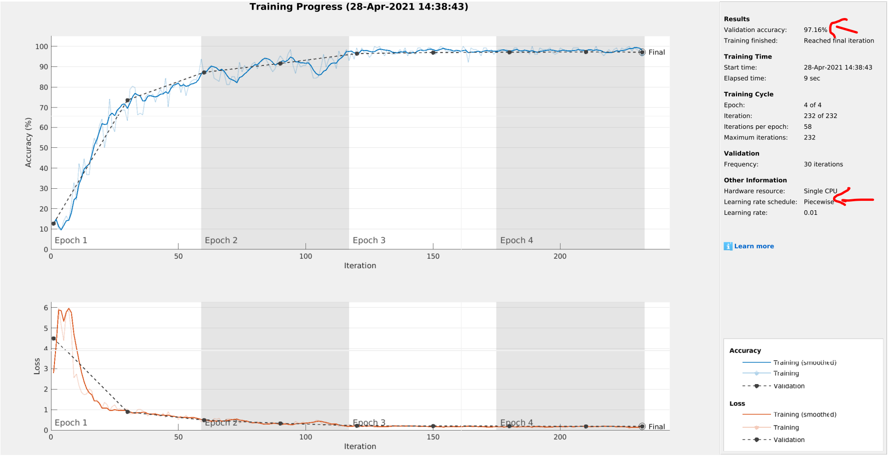
(b) mini-batch SGD, momentum nélkül, mini-batch mérete 128, tanulási paraméter változik (0.1 két epochon át, majd 0.01)

(c) mini-batch SGD, momentummal ($\varepsilon = 0.9$), mini-batch mérete 128, tanulási paraméter állandó (0.01)

(a) eset:


Training Progress (28-Apr-2021 13:27:45)
Accuracy (%)
Iteration
Epoch 1
Epoch 2
Epoch 3
Epoch 4
Final
Loss
Results
Validation accuracy: 82.24%
Training finished: Reached final iteration
Training Time
Start time: 28-Apr-2021 13:27:45
Elapsed time: 5 sec
Training Cycle
Epoch: 4 of 4
Iteration: 232 of 232
Iterations per epoch: 58
Maximum iterations: 232
Validation
Frequency: 30 iterations
Other Information
Hardware resource: Single GPU
Learning rate schedule: Constant
Learning rate: 0.01
Learn more
Accuracy
Training (smoothed)
Training
Validation
Loss
Training (smoothed)
Training
Validation

(b) eset:



Training Progress (28-Apr-2021 14:38:43)
Accuracy (%)
Epoch 1
Epoch 2
Epoch 3
Epoch 4
Final
Iteration
Loss
Results
Validation accuracy: 97.16%
Training finished: Reached final iteration
Training Time
Start time: 28-Apr-2021 14:38:43
Elapsed time: 9 sec
Training Cycle
Epoch: 4 of 4
Iteration: 232 of 232
Iterations per epoch: 58
Maximum iterations: 232
Validation
Frequency: 30 iterations
Other Information
Hardware resource: Single CPU
Learning rate schedule: Piecewise
Learning rate: 0.01
Learn more
Accuracy
Training (smoothed)
Training
Validation
Loss
Training (smoothed)
Training
Validation

momentum módszer

$\varepsilon = 0.9$

(c) eset: