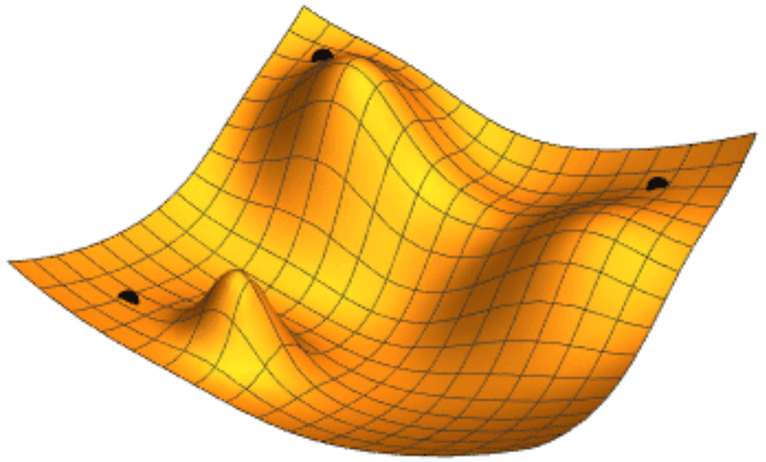
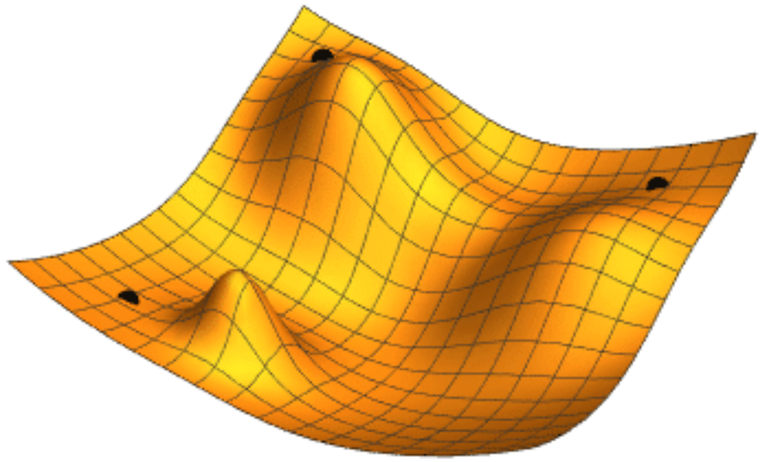


2. Előadás: Optimalizáció

Generatív AI és Inverz Módszerek a Képszintézisben

BME-VIK IIT, 2026

Dr. Vaitkus Márton



2. Előadás: Optimalizáció

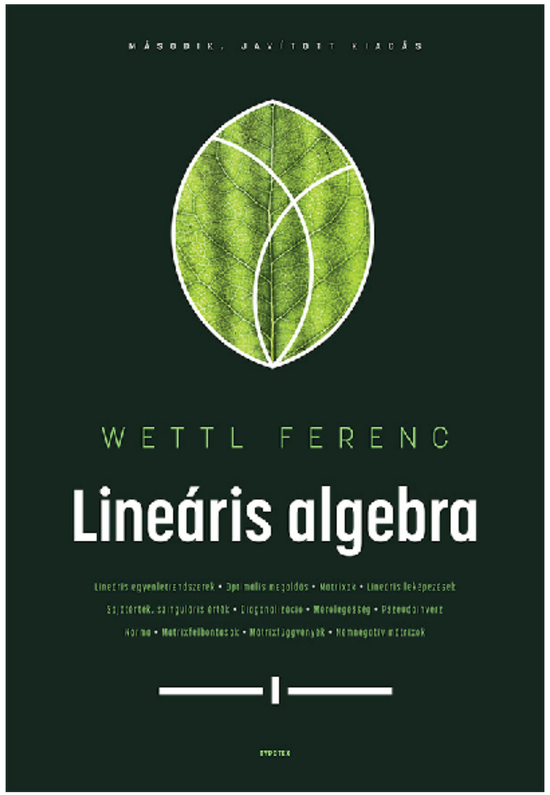
Generatív AI és Inverz Módszerek a Képszintézisben

BME-VIK IIT, 2026

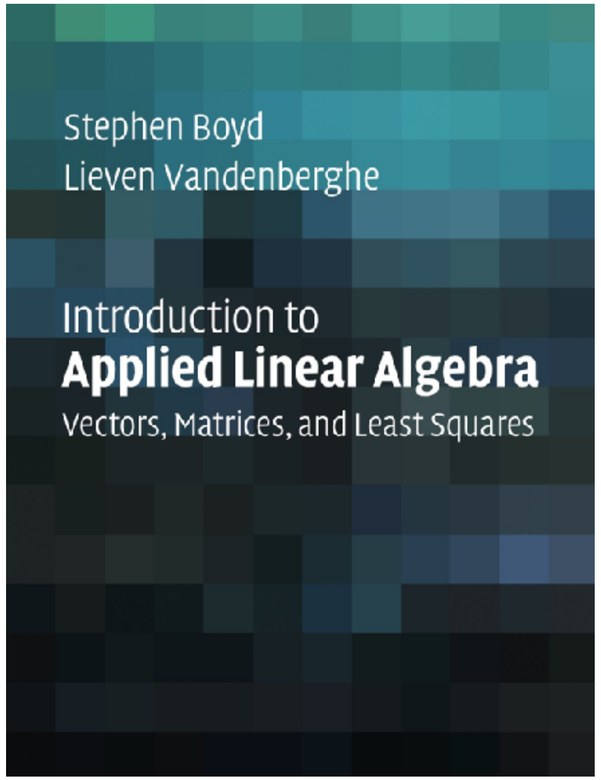
Dr. Vaitkus Márton

Lineáris Algebra Alapok

Könyvajánló



https://math.bme.hu/~wettl/okt/Jegyzet/00la.pdf



https://web.stanford.edu/~boyd/vmls/

Jelölések

$$\mathbf{x} = x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

A vektorokat / mátrixokat nem mindig különböztetjük meg a skalároktól!

$$\frac{\partial f(x)}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} \\ \frac{\partial f(x)}{\partial x_2} \\ \vdots \\ \frac{\partial f(x)}{\partial x_n} \end{bmatrix} = \nabla f(x) = f'(x) = f_x(x) = \cdots$$

$$\frac{\partial^2 f(x)}{\partial \mathbf{x}^2} = \begin{bmatrix} \frac{\partial^2 f(x)}{\partial x_1^2} & \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f(x)}{\partial x_2 \partial x_1} & \frac{\partial^2 f(x)}{\partial x_2^2} & \cdots & \frac{\partial^2 f(x)}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} & \frac{\partial^2 f(x)}{\partial x_2 \partial x_n} & \cdots & \frac{\partial^2 f(x)}{\partial x_n^2} \end{bmatrix} = \nabla^2 f(x) = f''(x) = f_{xx}(x) = \cdots$$

Gradiens (1. derivált)

- A lokálisan max. növekedés irányába mutat
- Merőleges (normális) az $f(x) = C$ szintfelületre

Többkimenetű függvényre: **Jacobi-mátrix**

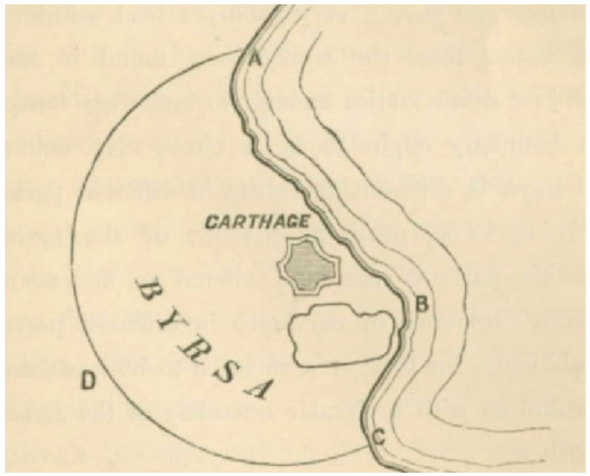
Hesse-mátrix (2. derivált)

- A függvény lokális görbületét definiálja
- Másodrendben simul az $f(x) = C$ szintfelülethez

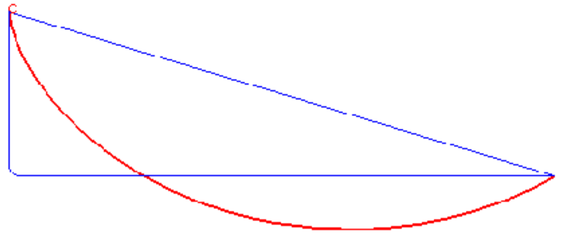
Többkimenetű függvényre: magasabb rendű tenzor

Optimalizációs Feladatok

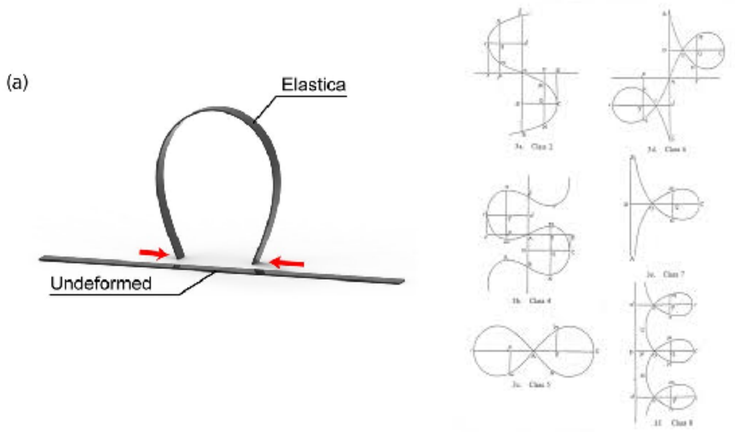
Történelem



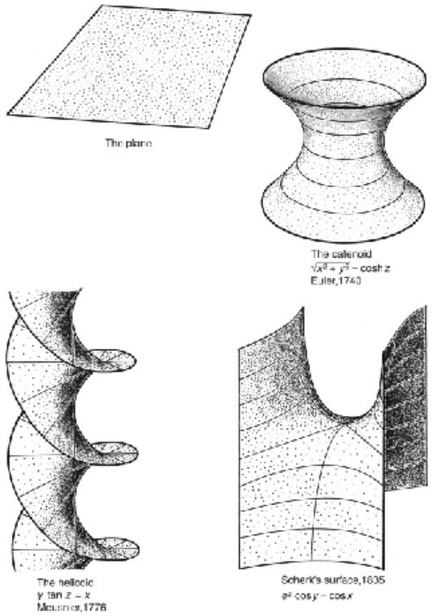
Izoperimetrikus probléma
[Dido, i.e. 9. Sz.]



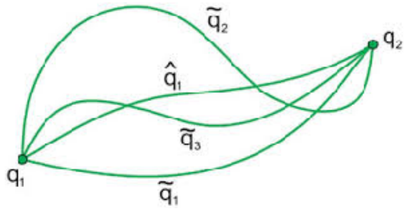
Brachisztochron-probléma
[Bernoulli, 1696]



“Elasztikus” görbék
[Lagrange, 1744]



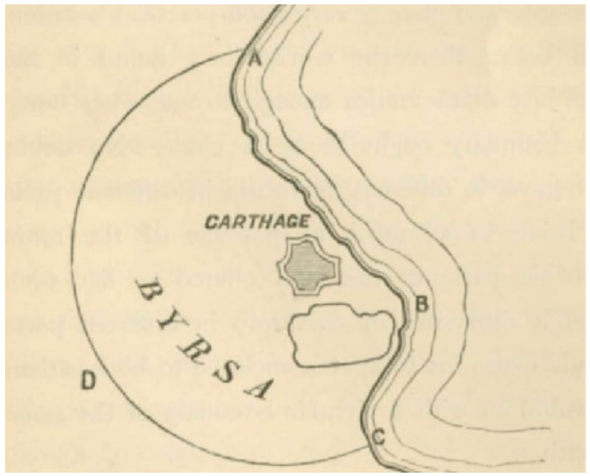
“Minimál” felületek
[Lagrange, 1762]



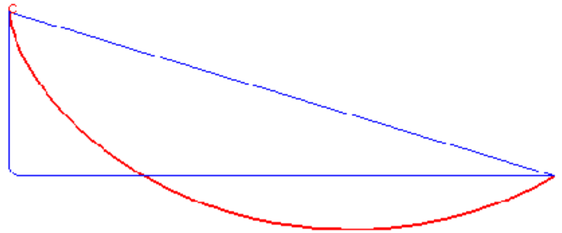
Mechanika (“Variációs elv”)
[Euler-Lagrange-Hamilton, 18 sz.]

Optimalizációs Feladatok

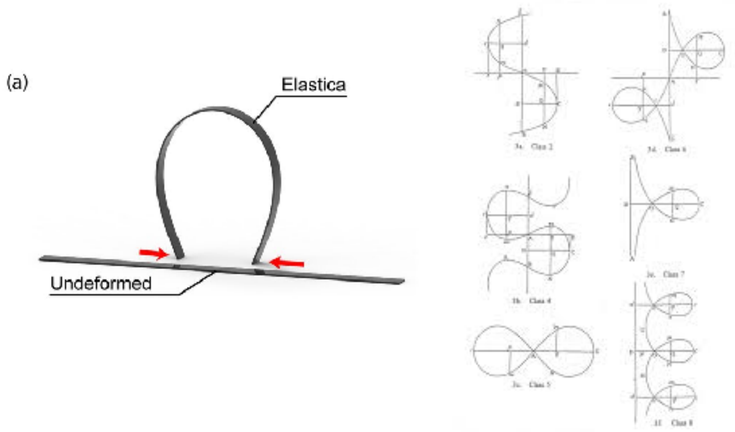
Történelem



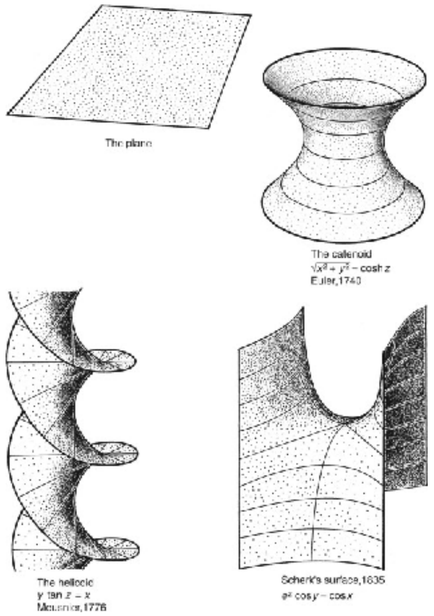
Izoperimetrikus probléma
[Dido, i.e. 9. Sz.]



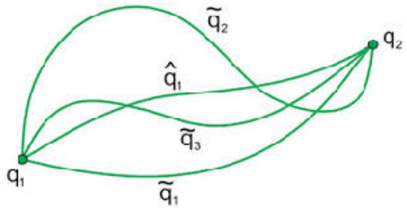
Brachisztochron-probléma
[Bernoulli, 1696]



“Elasztikus” görbék
[Lagrange, 1744]



“Minimál” felületek
[Lagrange, 1762]



Mechanika (“Variációs elv”)
[Euler-Lagrange-Hamilton, 18 sz.]

Optimalizációs Feladatok

- Diszkrét optimalizáció:
  - diszkrét ismeretlenek — pl. Boolean, véges halmaz, egész számok, stb.
  - Pl. kombinatorikus problémák (gráfok, keresés, SAT, stb.) — Tipikusan NP-Nehéz!
  - “Ügyes” algoritmusok (Dijkstra, stb.), heurisztikák szükségesek!
- Folytonos optimalizáció:
  - Folytonos ismeretlenek (valós számok)
  - Szintén lehet (NP-)nehéz, de sok gyakorlati probléma “könnyebb”

Optimalizációs Feladatok
Általános definíció

Költségfüggvény (célfüggvény/veszteségfüggvény/loss)

$$\min_{x} \quad f(x)$$

$$\text{s.t.} \quad g_i(x) \geq 0, \quad i = 1,\ldots,m$$
$$h_j(x) = 0, \quad j = 1,\ldots,p$$

} **Kényszerek**

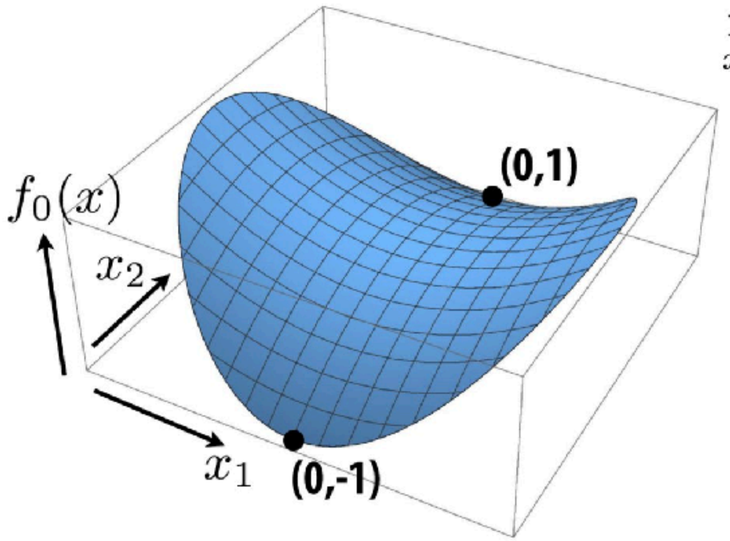
Alternatív jelölés: $f(x) \rightarrow \min$

Optimalizációs feladat (Standard alakban)

Gondoljunk bele:
Létezik egyáltalán megoldás?
A megoldás egyértelmű?
Tényleg a “valódi” optimum érdekel minket?

Optimalizációs Feladatok

Megoldás egyértelműsége



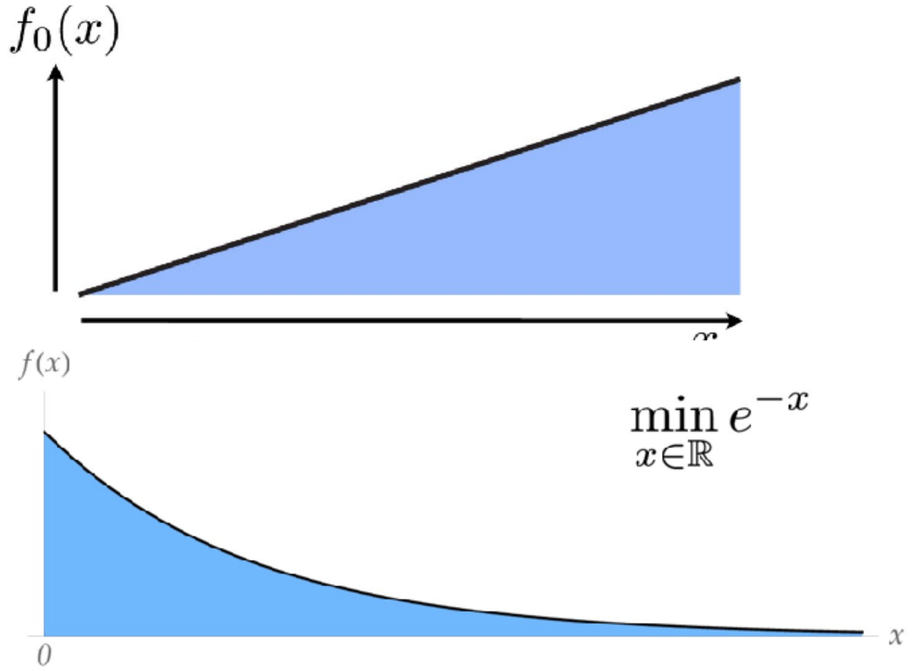
$$\begin{aligned} \min_{x \in \mathbb{R}^2} \quad & x_1^2 - x_2^2 \\ \text{s.t.} \quad & x_1^2 + x_2^2 - 1 \leq 0 \end{aligned}$$

Forrás: K. Crane

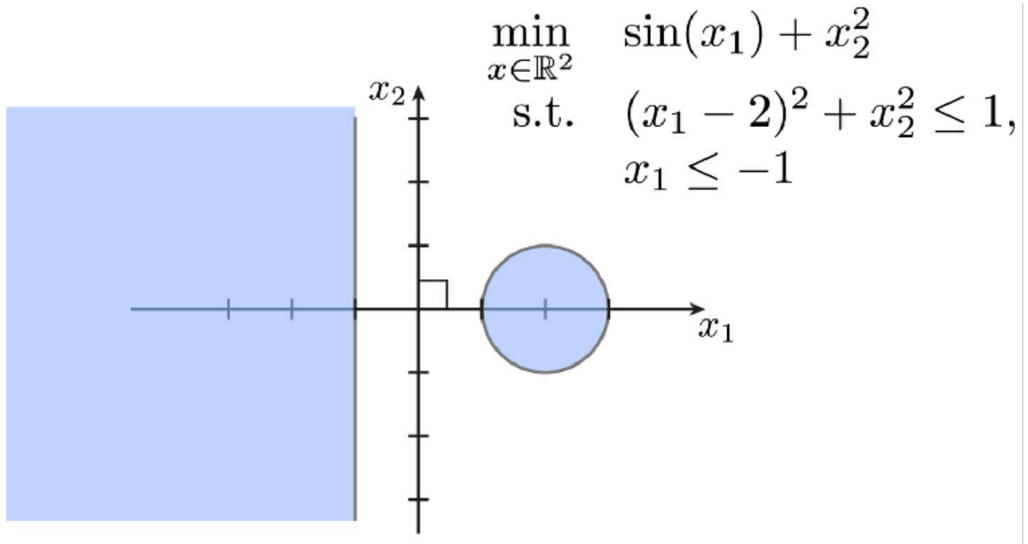
Lehet, hogy a megoldás nem egyértelmű!

Optimalizációs Feladatok

Megoldás létezése



<u>**Lehet, hogy nem létezik optimális megoldás!**</u>



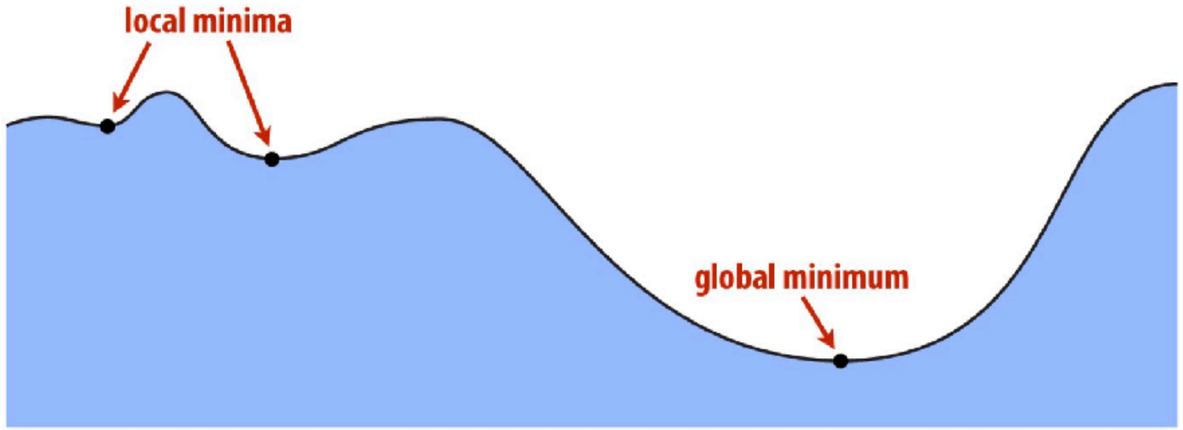
<u>**Lehet, hogy nem létezik “helyes” megoldás!**</u>

Forrás: K. Crane

Optimalizációs Feladatok

Lokális vs. Globális Optimum

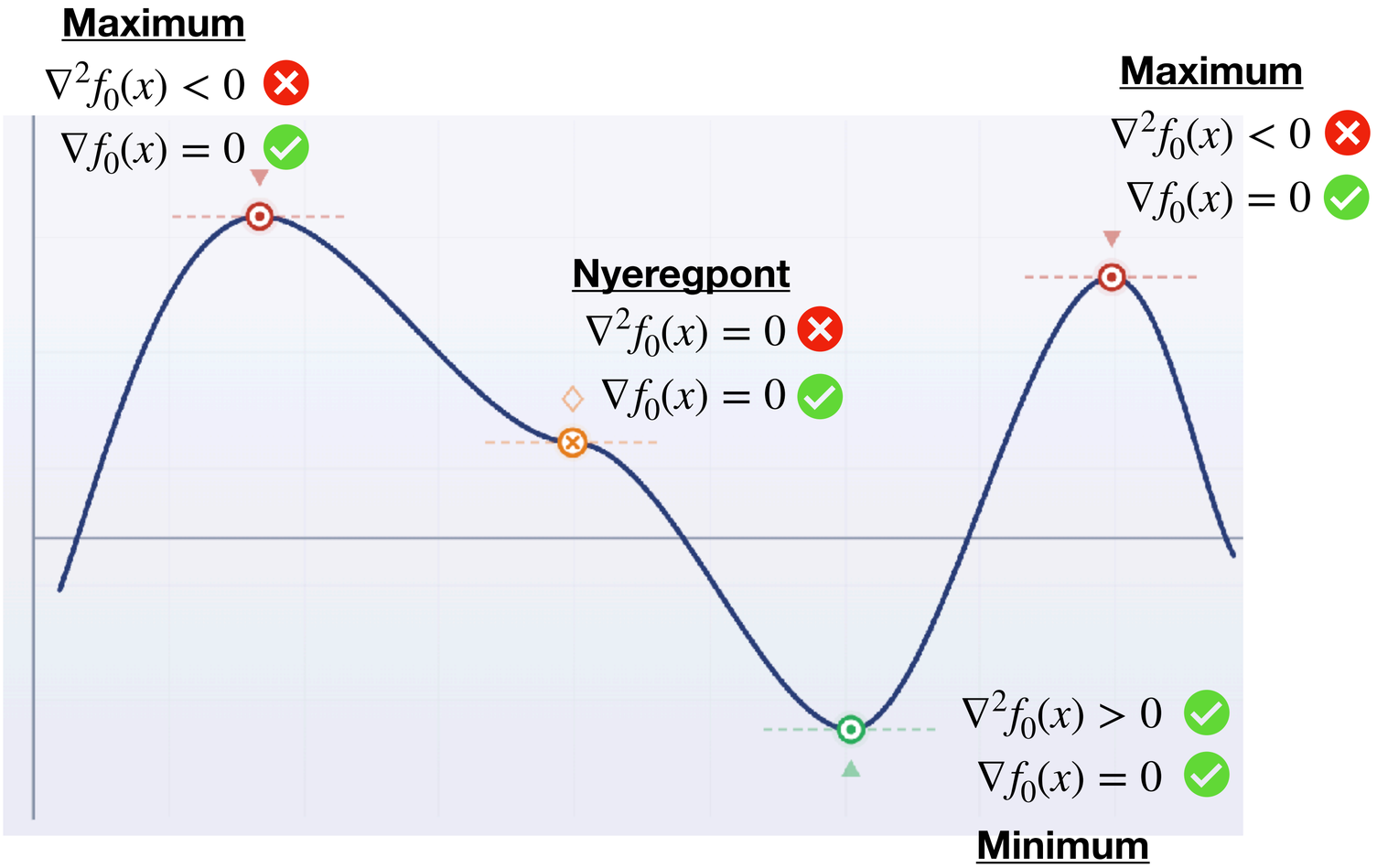
- **Globális minimum**: a legkisebb lehetséges érték a megengedett tartományon
  - Általában nincs reményünk megtalálni!
- **Lokális minimum**: közvetlen környezetében nem tudunk jobbat találni
  - Általában könnyebb megtalálni…



Forrás: K. Crane

Optimalizációs Feladatok

Lokális optimum feltétele



Optimalizációs Feladatok

Kitérő: Kvadratikus függvények

$$f(x) = \frac{1}{2}\sum_i \sum_j a_{ij} x_i x_j = \frac{1}{2} x^T A x \underset{x=Sz}{=} \sum_i \lambda_i z_i^2$$

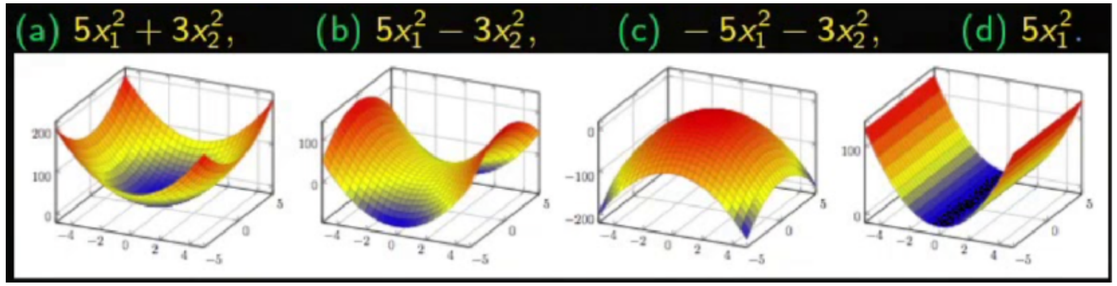
A sajátértékei

főtengely transzformáció

$\forall \lambda_i > 0$: pozitív definit (*konvex elliptikus*)

$\forall \lambda_i < 0$: negatív definit (*konkáv elliptikus*)

Egyébként: indefinit (*hiperbolikus*)



Konvex fv.: konstans szintfelületek ellipszoid alakúak, főtengelyek hossza $1/\lambda_i$-el arányos

Kvadratikus függvény minimumának szükséges feltétele: $\frac{\partial f(x)}{\partial x} = 0 \quad \Rightarrow \quad \boxed{Ax = b}$

Lineáris egyenletrendszer!

Optimalizációs Feladatok

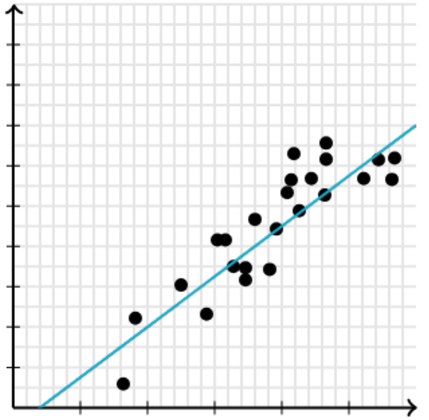
Példák — Lineáris regresszió

- Adat párok: független változók $\mathbf{x}_i = (x_{i1}, \ldots x_{id}) \in \mathbb{R}^d$, függő változók $y_i \in \mathbb{R}$
- Változók között posztulált összefüggés (“lineáris modell”):

Függvénybázis

$$y = w_1\beta_1(\mathbf{x}) + \ldots + w_N\beta_N(\mathbf{x})$$

Együtthatók (ismeretlenek!)



példa: egyenes illesztés pontokra $y = w_1 x + w_0$

- Lineáris egyenletrendszer az adat párok alapján:

$$w_1\varphi_1(\mathbf{x}_i) + \ldots + w_N\varphi_N(\mathbf{x}_i) = y_i,\ i = 1,\ldots,M$$

- Túlhatározott $(N > M)$ — általában nem oldható meg minden egyenlet!
- Minimalizáljuk a négyzetes hibaösszeget (legkisebb négyzetek / **least-squares**):

$$f(w_1, \ldots w_N) = \sum_{i=1}^{M} (w_1\varphi_1(\mathbf{x}_i) + \ldots + w_N\varphi_N(\mathbf{x}_i) - y_i)^2 \to \min$$

Mátrixos alak:

$$\mathbf{Bw} = \mathbf{y}$$

$$\| \mathbf{Bw} - \mathbf{y} \|^2 \to \min$$

Optimalizációs Feladatok

Példák — Lineáris regresszió

- Megoldás zárt alakban (pszeudoinverz): $\mathbf{w} = (\mathbf{B}^{\mathbf{T}}\mathbf{B})^{-1}\mathbf{B}^{T}\mathbf{y}$
  - Ritka kivétel!

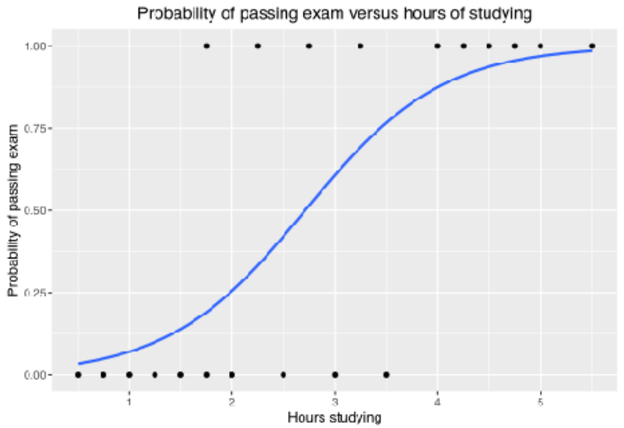
Optimalizációs Feladatok

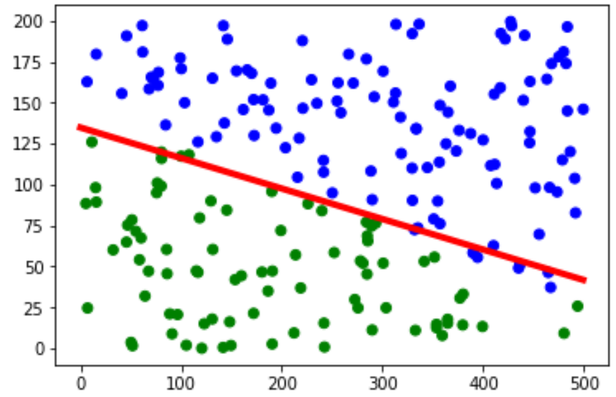
Példák — Lineáris osztályozás / Logisztikus regresszió

- Adat párok: független változók
  $\mathbf{x}_i = (x_{i1}, \ldots x_{id}) \in \mathbb{R}^d$, bináris címkék $y_i \in \{0,1\}$
  (pl. Nem Spam/Spam)
- Cél: helyesen osztályozni az adatpontokat
- Valószínűségi modell: $p(x) = \frac{1}{1+e^{-z(x)}}$ $z(x) = \sum_i w_i \beta_i(x)$

  Együtthatók (ismeretlenek)
- Logisztikus loss (**kereszt-entrópia**):

$$L(x) = \sum_{i=1}^{M} y_i \log(p(x_i)) + (1-y_i)\log(1-p(x_i))$$



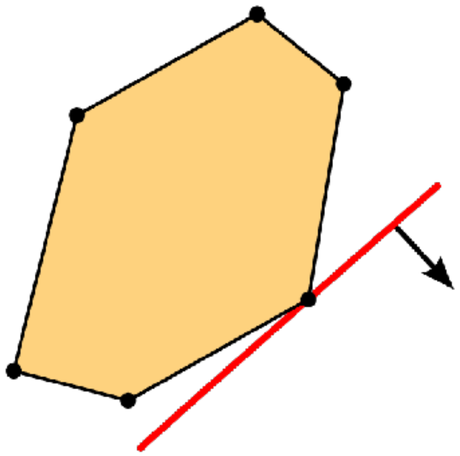


Optimalizációs Feladatok

Példák — Lineáris Programozás

- Feladat: erőforrások (pl. pénz, élelem, áru, stb.) optimális szétosztása
- Minimalizálandó összköltség: $\sum_{i=1}^{n} q_i c_i = \mathbf{c^T q}$
- Teljesítendő kényszerek: $\sum a_{i,j} q_j \leq b_i \rightarrow \mathbf{Aq} \preceq \mathbf{b}$
- Lineáris Programozási feladat (LP):

$$\min_{\mathbf{q}} \quad \mathbf{c}^T\mathbf{q}$$
$$\text{s.t.} \quad \mathbf{Aq} \preceq \mathbf{b}$$

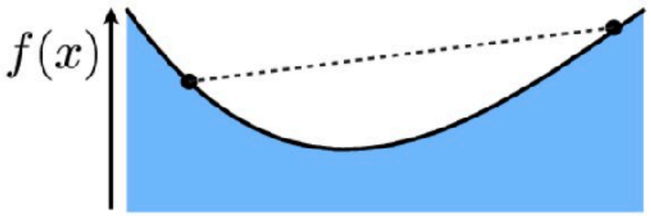


- Geometriai értelmezés: keressük egy politóp legszélsőbb pontját, adott irányban
- Nincs zárt alakú megoldása — iteratív módszerek (szimplex, belsőpont, stb.)

Optimalizációs Feladatok

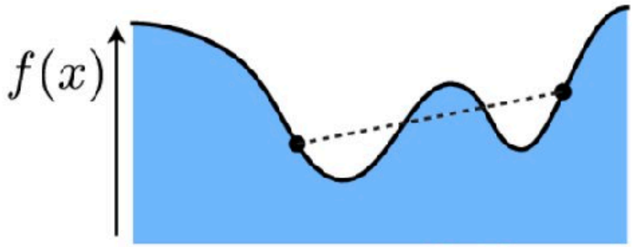
Konvexitás

Konvex függvény

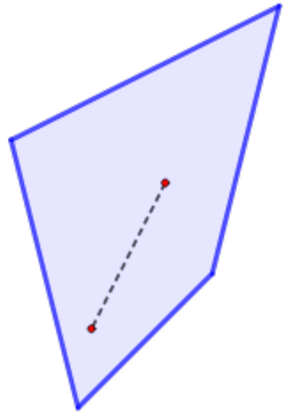


Forrás: K. Crane

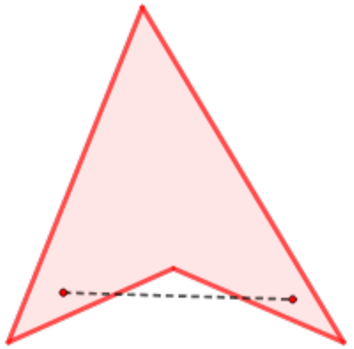
Nem-konvex függvény



Konvex halmaz



Nem-konvex halmaz



Konvex célfv. + Konvex kényszerhalmaz = “**konvex optimalizáció**”

Konvex optimalizáció:
- globális optimum
- polinomiális időn belül
- garanciákkal

Nem-konvex optimalizáció:
- csak lokális optimum
- nincs garancia…

Optimalizációs Módszerek

Hogyan találjuk meg az optimumot?

Optimalizációs Módszerek

Optimalizáció kényszerek nélkül

- Optimalizációs feladat: $f(\mathbf{x}) \rightarrow \min$
- Hogyan találjuk meg az $\mathbf{x}^* = \text{argmin}_{\mathbf{x}} f(\mathbf{x})$ optimumot?
- Egy “általános” optimalizációs probléma NP-Nehéz — egzakt optimumot találni praktikusan nem lehetséges!
- Gyakran lehetséges “jó” (lokális) optimumot számítani, *numerikus módszerekkel*
- Nulladrendű módszerek: csak az $f(\mathbf{x})$ kiértékelését igénylik
  - Előny: a minimalizált függvény lehet “fekete doboz”, nem kell deriválni
  - Hátrány: lassú, csak kevés ismeretlen esetén praktikus (“dimenziók átka”)

Optimalizációs Módszerek

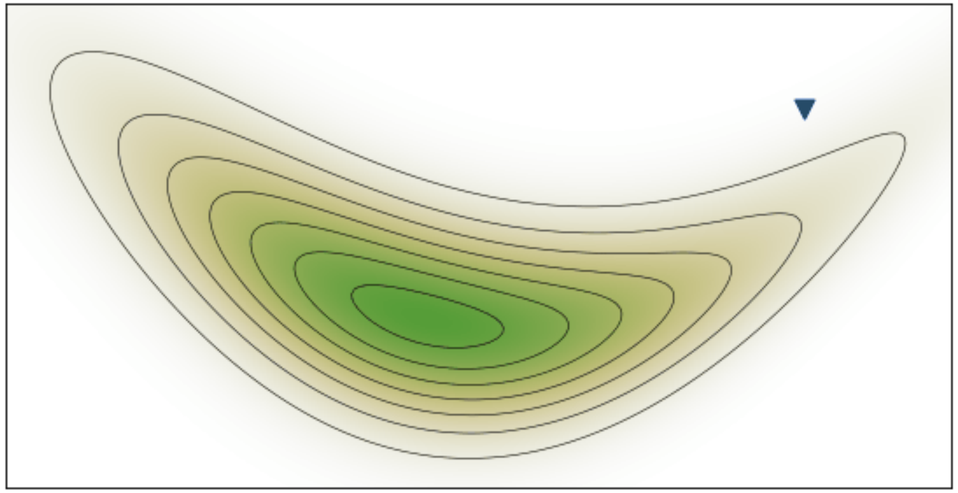
Gradiens módszer

- Lokális optimum szükséges feltétele: $\frac{\partial f(\mathbf{x}^*)}{\partial \mathbf{x}} = \mathbf{0}$
- Gradiens módszer (**gradient descent**):

$$x_{k+1} = x_k - \tau \cdot \frac{\partial f(x_k)}{\partial x}$$

  - τ: lépésköz / “**learning rate**”

Csak lokális minimum — függ a kezdőponttól!



Optimalizációs Módszerek

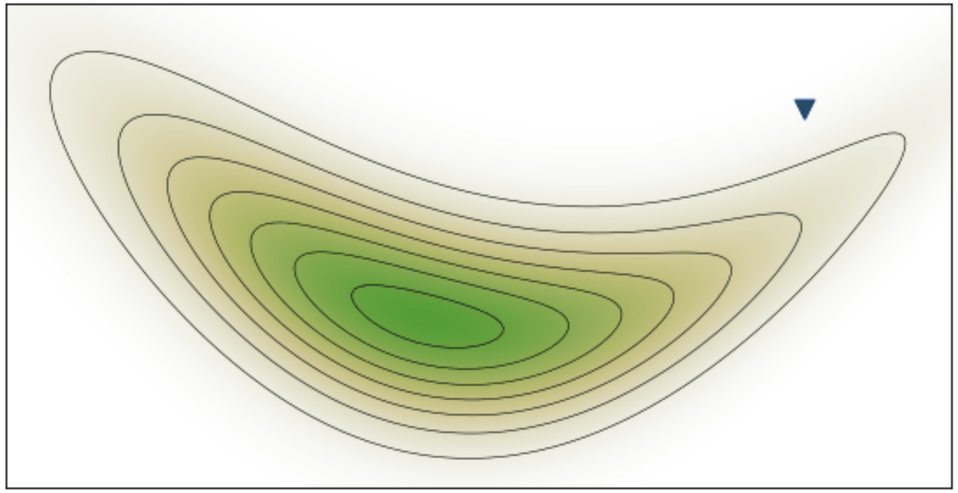
Gradiens módszer

- Lokális optimum szükséges feltétele: $\frac{\partial f(\mathbf{x}^*)}{\partial \mathbf{x}} = \mathbf{0}$
- Gradiens módszer (**gradient descent**):

$$x_{k+1} = x_k - \tau \cdot \frac{\partial f(x_k)}{\partial x}$$

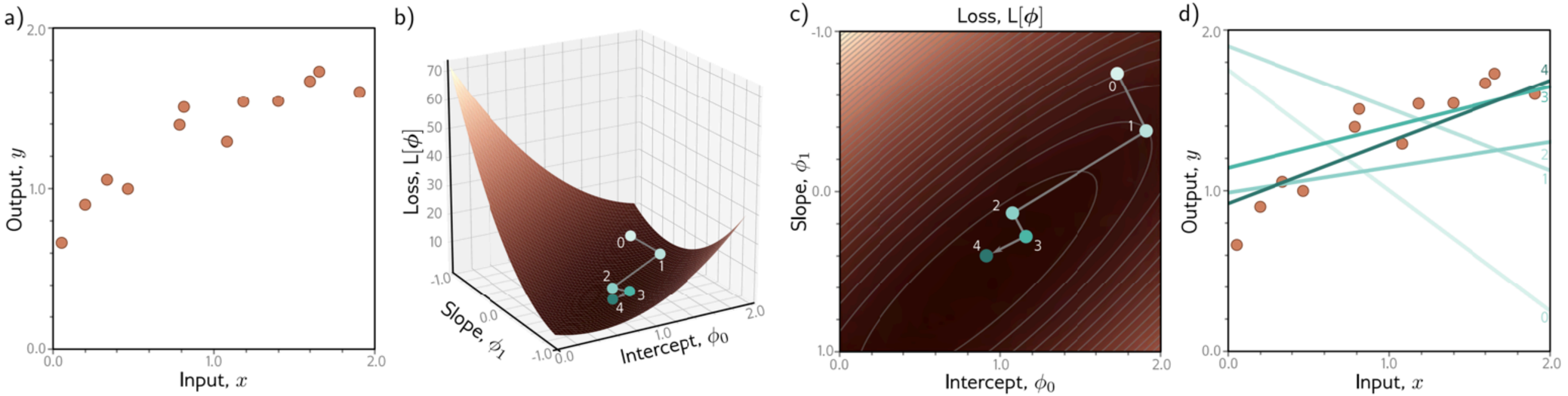
  - τ: lépésköz / “**learning rate**”

Csak lokális minimum — függ a kezdőponttól!



Optimalizációs Módszerek

Gradiens módszer — Példa



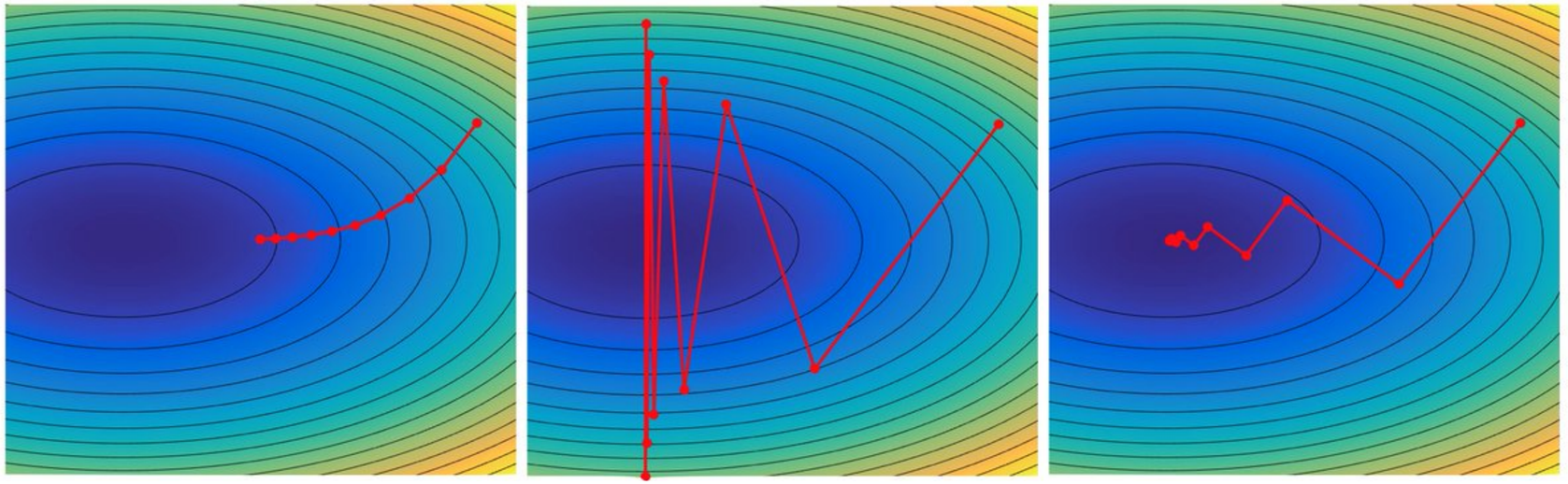
Forrás: S.J.D. Prince

Egyenes illesztése gradiens módszerrel

Optimalizációs Módszerek

Gradiens módszer — A lépésköz szerepe

$$x_{k+1} = x_k - \tau \cdot \frac{\partial f(x_k)}{\partial x}$$



túl kicsi τ

túl nagy τ

ideális τ (“line search”)

A gyakorlatban érdemes *adaptív* lépésközt használni!

Optimalizációs Módszerek

Gradiens módszer — Kondíciószám

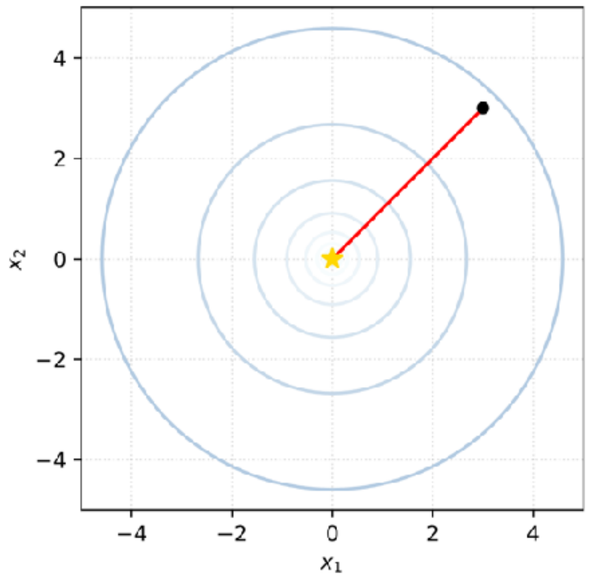
- Másodfokú közelítés:

$$f(x+\Delta x) \approx f(x) + \frac{\partial f(x)}{\partial x}\Delta x + \frac{1}{2}\Delta x^T \underbrace{\frac{\partial^2 f(x)}{\partial x^2}}_{\text{Hesse mátrix}} \Delta x$$

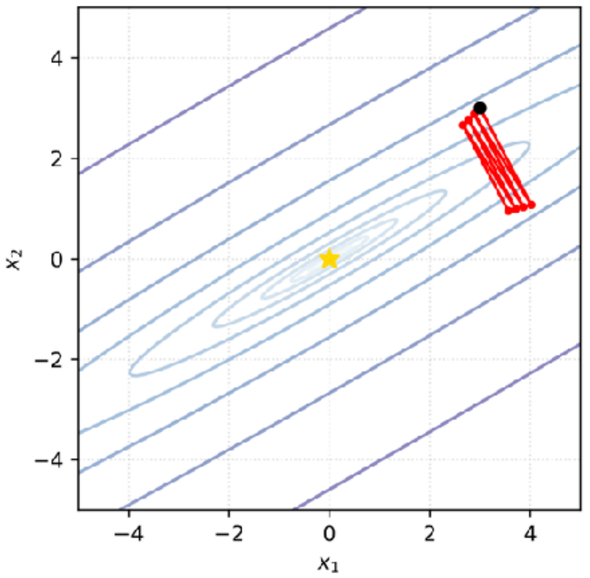
- A gradiens-módszer konvergenciáját a másodfokú tagok határozzák meg.

- A Hesse-mátrix legnagyobb / legkisebb sajátértékének aránya: (lokális) **kondíciószám**

$$\kappa = \frac{\lambda_{max}}{\lambda_{min}}$$



$\kappa = 1$

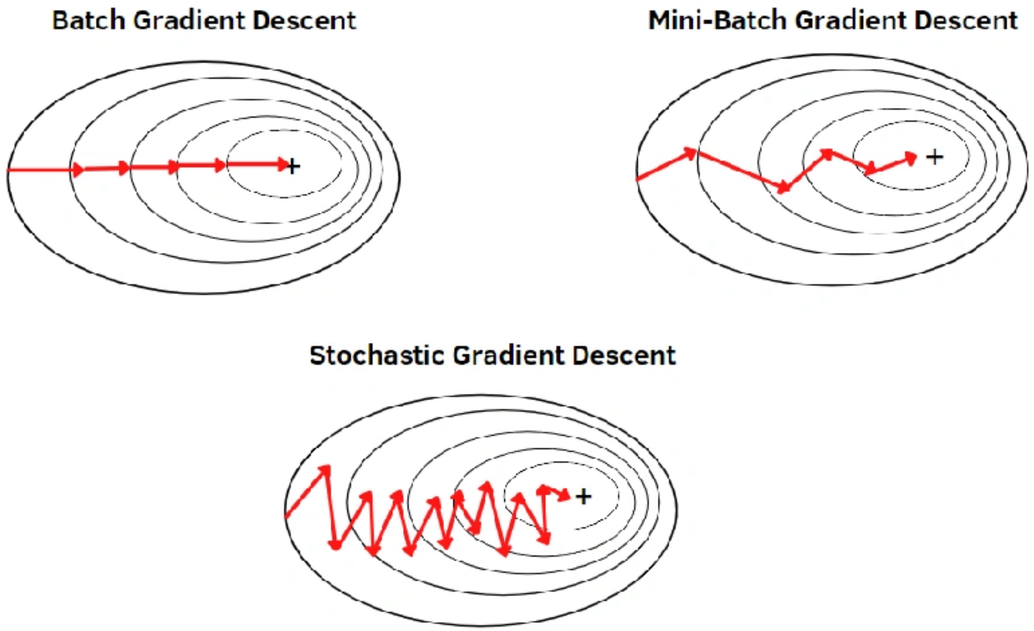


$\kappa = 100$

Optimalizációs Módszerek

Sztochasztikus gradiens módszer

- Adatillesztési / tanulási problémáknál a gradiens kiértékeléséhez minden iterációban az *összes* adatpontra szükség volna…
- Praktikus kompromisszum: a gradienst véletlenül mintavételezett részhalmaz alapján becsüljük (**mini-batch**)
  - batch méret = 1 — **stochastic gradient descent (SGD)**
  - Az SGD zajos, ezért nagyobb batch méretet használunk (de pongyolán ezt is SGD-nek szokás nevezni)!
- **Epoch**: minden adatpont egyszer “érintettünk”.

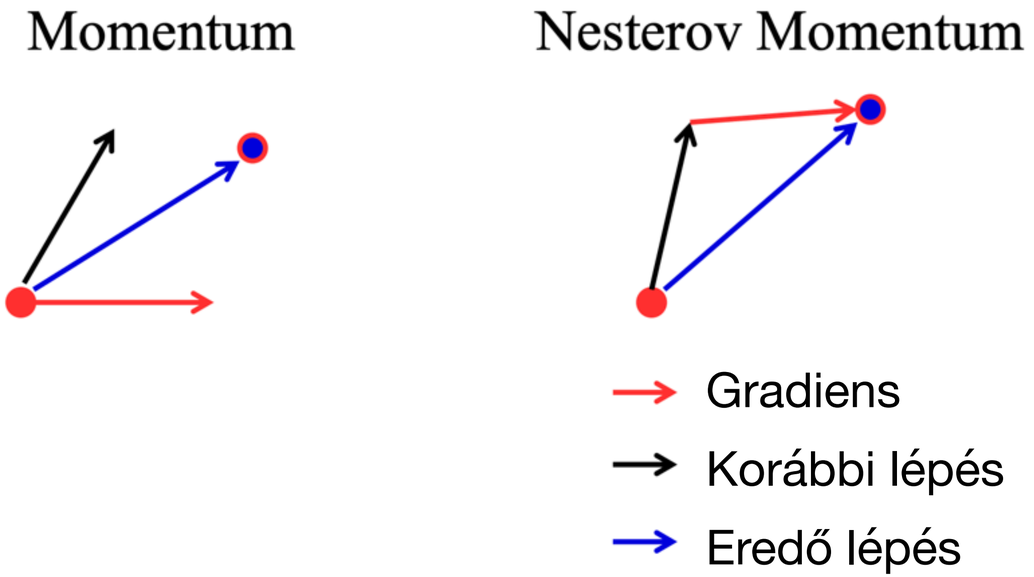


Optimalizációs Módszerek

Momentum

- Az (S)GD könnyen beragad lokális minimumokba…
- Ötlet: legyen a mozgásnak “tehetetlensége” (guruló golyó analógia) — a lokális minimumból ki tudunk “gurulni”!
- Használjuk fel a korábbi iteráció gradiensét (mozgóátlag, β_1 “felejtési faktor”):

$$M_{k+1} = \beta_1 \cdot M_k + (1 - \beta_1) \cdot \nabla L(x_k)$$



Forrás: Szemenyei M.

Optimalizációs Módszerek

Adaptív Momentum (Adam)

- Probléma: az (S)GD nagyobbat lép ahol meredek a függvény, kisebbet ahol lapos…
- 1. ötlet: skálázzuk adaptívan (normalizáljuk) a gradienst

$$x_{k+1} = x_k - \tau \cdot \frac{\nabla L(x_k)}{\| \nabla L(x_k) \|}$$

  - Újabb probléma: nem konvergál…

- 2. ötlet: momentum a gradiens normára (**RMSProp**)

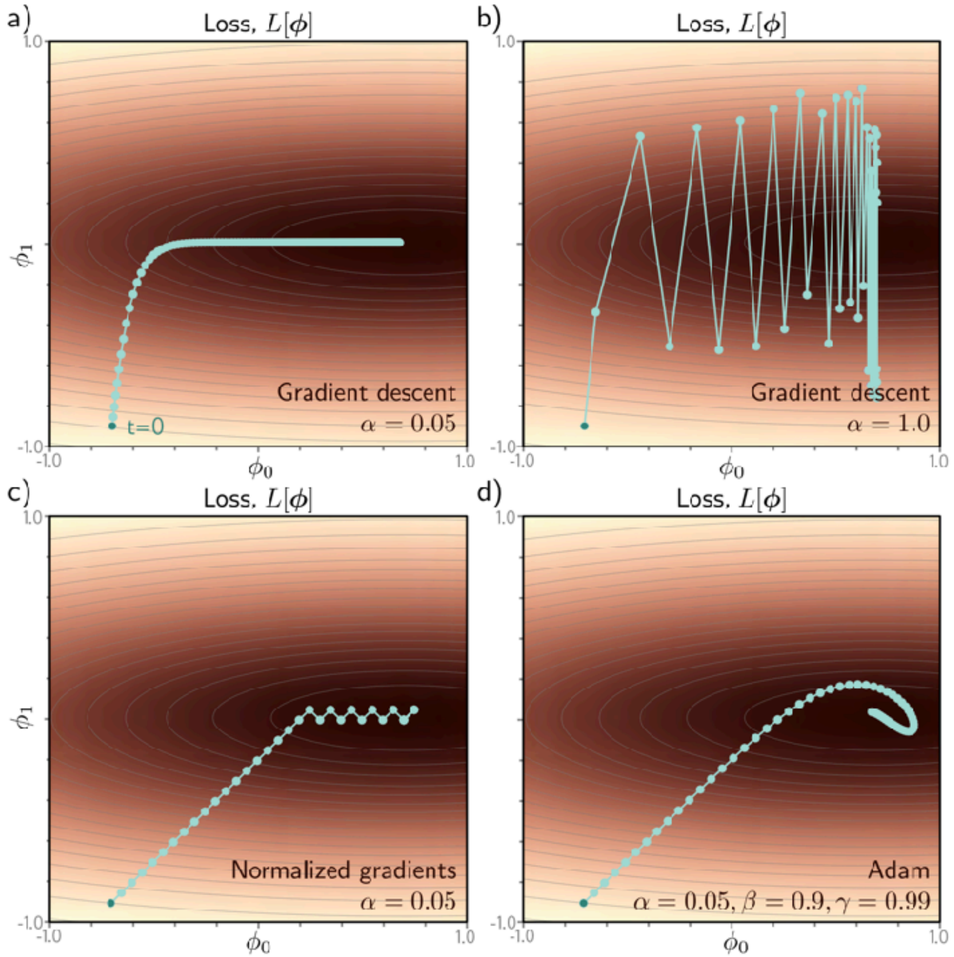
$$x_{k+1} = x_k - \frac{\tau}{\sqrt{G_k} + \epsilon} \cdot \nabla L(x_k)$$

$$G_{k+1} = \beta_2 \cdot G_k + (1 - \beta_2) \cdot \| \nabla L(x_k) \|^2$$

“2. momentum”

$\epsilon \approx 10^{-8}$

Biztonsági faktor



Forrás: S.J.D. Prince

Optimalizációs Módszerek

Adaptív Momentum (Adam)

- 3. ötlet: Momentum a gradiensre és a normájára

$$x_{k+1} = x_k - \frac{\tau}{\sqrt{G_k} + \epsilon} \cdot M_k$$

$$M_{k+1} = \beta_1 \cdot M_k + (1 - \beta_1) \cdot \nabla L(x_k)$$

$$G_{k+1} = \beta_2 \cdot G_k + (1 - \beta_2) \cdot \| \nabla L(x_k) \|^2$$

  - Probléma: M_k, G_k 0-ra van inicializálva — lassú indulás…

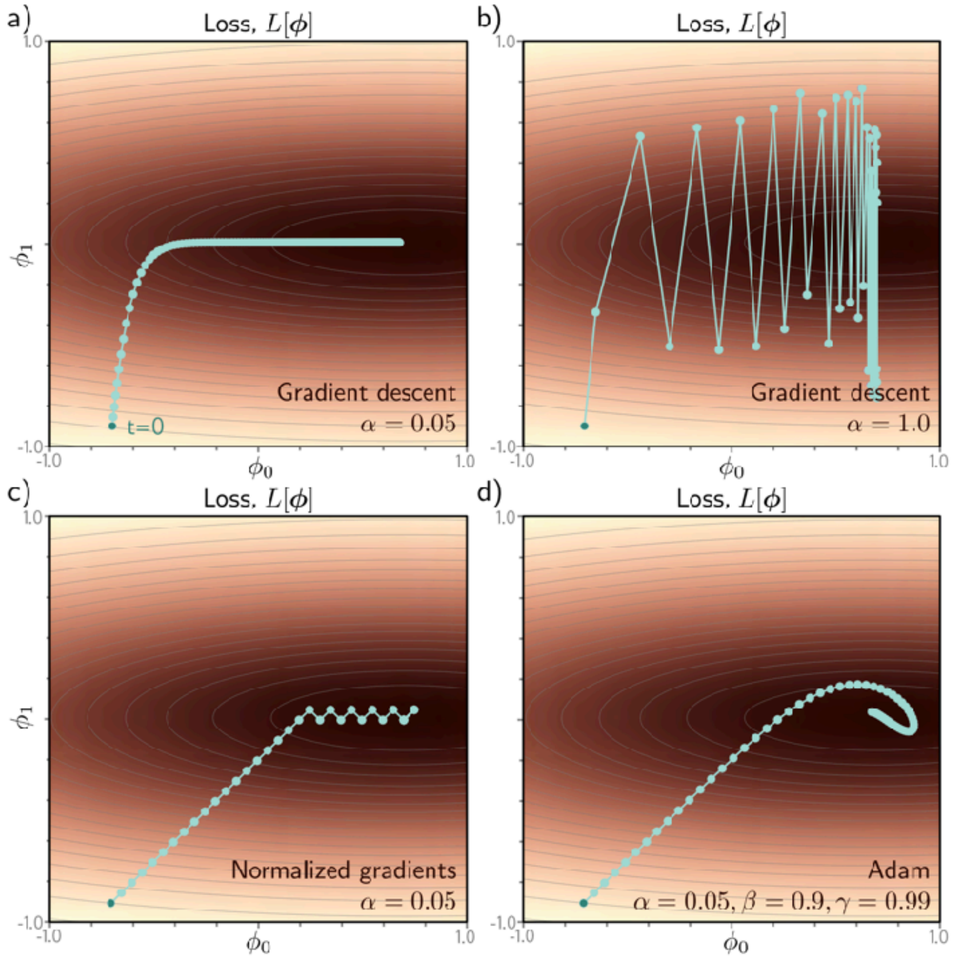
- 4. ötlet: adaptív β_1, β_2

$$x_{k+1} = x_k - \frac{\tau}{\sqrt{G_k} + \epsilon} \cdot M_k$$

$$M_{k+1} = \frac{1}{1 - \beta_1^k} \cdot \left(\beta_1 \cdot M_k + (1 - \beta_1) \cdot \nabla L(x_k)\right)$$

$$G_{k+1} = \frac{1}{1 - \beta_1^k} \cdot \left(\beta_2 \cdot G_k + (1 - \beta_2) \cdot \| \nabla L(x_k) \|^2\right)$$

Adam



Forrás: S.J.D. Prince

Optimalizációs Módszerek

Adaptív Momentum (Adam)

- $\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \eta = 10^{-8}$ (Defaults)

$m_0 \leftarrow 0$ (Initialize 1st moment vector)

$v_0 \leftarrow 0$ (Initialize 2nd moment vector)

$i \leftarrow 0$ (Initialize step)

while Θ_i not converged **do**

$i \leftarrow i+1$

$g_i \leftarrow \nabla_\Theta f_i(\Theta_{i-1})$ (Get gradients at step i)

$m_i \leftarrow \beta_1 \cdot m_{i-1} + (1-\beta_1) \cdot g_i$ (Update biased first moment estimate)

$v_i \leftarrow \beta_2 \cdot v_{i-1} + (1-\beta_2) \cdot g_i^2$ (Update biased second raw moment estimate)

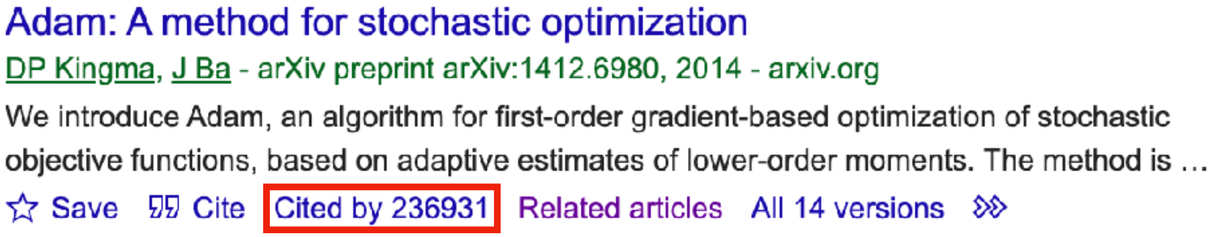
$\hat{m}_i \leftarrow m_i / (1-\beta_1^i)$ (Compute bias-corrected first moment estimate)

$\hat{v}_i \leftarrow v_i / (1-\beta_2^i)$ (Compute bias-corrected second raw moment estimate)

$\Theta_i \leftarrow \Theta_{i-1} - \alpha \cdot \hat{m}_i / (\sqrt{\hat{v}_t} + \eta)$ (Update parameters)

end while

return Θ_i (resulting parameters)



Adam: 2014 óta de facto standard a gradiens-alapú módszerek között!

Optimalizációs Módszerek

Newton módszer

- Ötlet: minimalizáljuk a függvény másodfokú közelítését

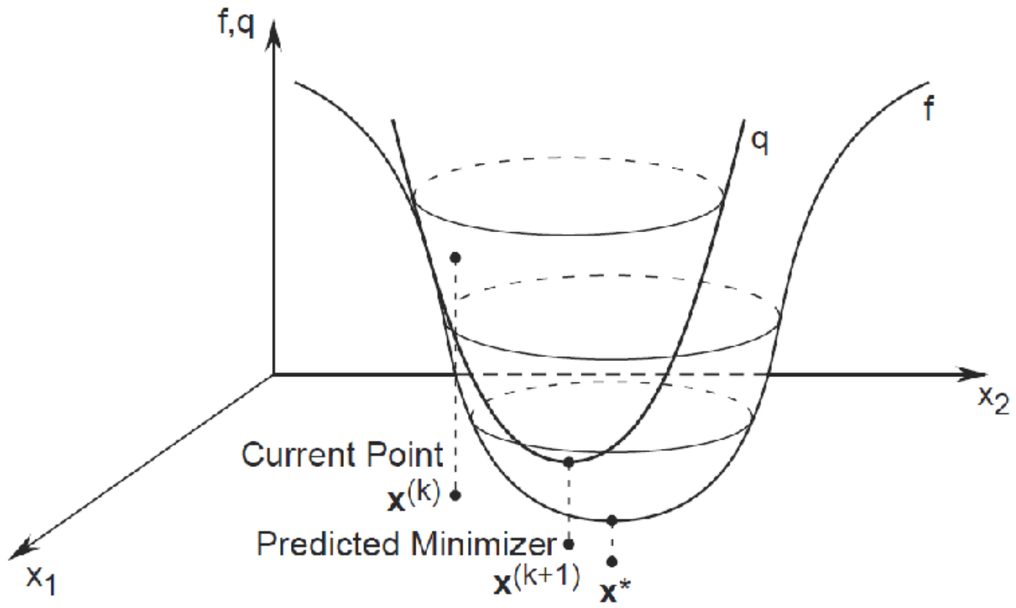
$$f(\mathbf{x}+\mathbf{d}) \approx f(\mathbf{x}) + \underbrace{\left(\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}\right)^T}_{\mathbf{g}(\mathbf{x})}\mathbf{d} + \frac{1}{2}\mathbf{d}^T \underbrace{\left(\frac{\partial^2 f(\mathbf{x})}{\partial \mathbf{x}^2}\right)}_{\mathbf{H}(\mathbf{x})}\mathbf{d}$$

$$\mathbf{H}(\mathbf{x})\mathbf{d} = -\mathbf{g}(\mathbf{x})$$

$$\mathbf{d} = -\mathbf{H}^{-1}(\mathbf{x})\mathbf{g}(\mathbf{x})$$

- Lineáris egyenletrendszer
- Interpretáció: kompenzáljuk a függvény görbületét (prekondícionálás)

Másik értelmezés: Newton-Raphson gyök keresés a $\nabla f(x) = 0$ egyenletre!



Az optimum közelében másodrendű konvergencia!
Konvex függvények esetén ideális választás!

Optimalizációs Módszerek

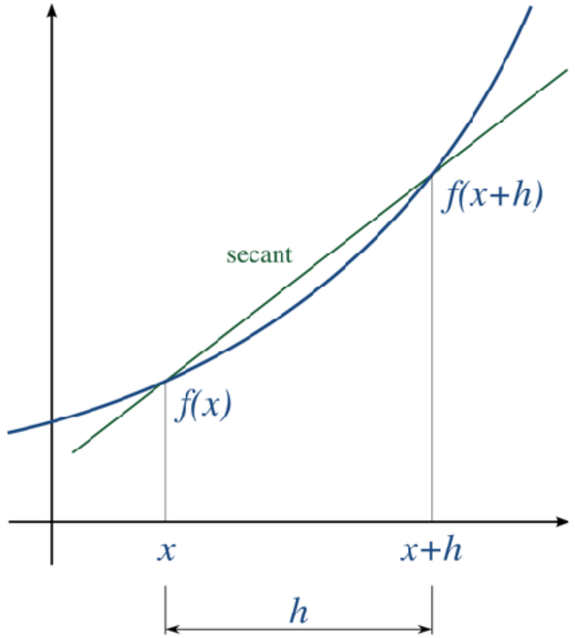
Kvázi-Newton módszerek*

- Newton-módszerben a tényleges Hesse-mátrix kiszámítása költséges
- Ötlet: közelítsük a Hesse-mátrixot — **Kvázi-Newton** módszerek
- Gyakran alkalmazott közelítés: szelő módszer

$$\mathbf{g}(\mathbf{x}+\mathbf{d}) \approx \mathbf{g}(\mathbf{x}) + \tilde{\mathbf{H}}\mathbf{d}$$

  - Népszerű variáns: **L-BFGS**
- Speciális esetek:
  - $\tilde{\mathbf{H}} = \mathbf{I}$ — GD
  - $\tilde{\mathbf{H}} = \mathbf{H}$ — Newton-módszer

Forrás: Wikipedia



- Kisebb méretű problémák (<1M változó) esetén nagyon gyakori választás, nagyobb problémákra kevés (fő akadály: batching)

Optimalizációs Módszerek

Gauss-Newton*

- Speciális, de nagyon gyakori loss alak — $f_i(x)$ nemlineáris függvény-rendszer:

$$L(x) = \sum_i (f_i(x))^2$$ *“Nem-lineáris least-squares”*

- Hesse mátrix közelítése (**Gauss-Newton** módszer):

$$\frac{\partial^2 (f_i(x))^2}{\partial x^2} = \frac{\partial f_i(x)}{\partial x}\frac{\partial f_i(x)}{\partial x}^T + \cancel{\frac{\partial^2 f_i(x)}{\partial x^2}}$$

- **Levenberg-Marquardt** módszer — adaptív λ paraméter

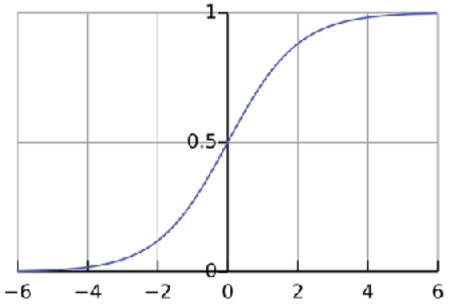
$$\left(\sum_i \frac{\partial f_i(x)}{\partial x}\frac{\partial f_i(x)}{\partial x}^T + \lambda \mathbf{I} \right) \mathbf{d} = \sum_i \frac{\partial f_i(x)}{\partial x} f_i(x)$$

- De facto standard kis/közepes nemlineáris least-squares problémákra.

Optimalizációs Módszerek

Kényszerek érvényesítése — Paraméterezés

- Néha lehet úgy formalizálni (paraméterezni) a feladatot, hogy egyes kényszerek automatikusan teljesüljenek
- Néhány példa:
  - $x_i \geq 0 : x_i = z_i^2$
  - $x_i > 0 : x_i = e^{z_i}$
  - $0 < x_i < 1 : x = \frac{1}{1 + e^x}$ (szigmoid)
  - $\sum_i x_i = 1 : x_i = \frac{z_i}{\sum_i z_i}$



- “Nincs ingyen ebéd” — az optimalizáció gyakran nehezebbé válik!
- Sajnos nem minden kényszerrel tehető meg…

Optimalizációs Módszerek

Kényszerek érvényesítése — Büntető Tagok

- A kényszer “hibáját” hozzáadhatjuk a minimalizált Loss-hoz.
- Például:

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & h_i(x) = 0, \quad i = 1,\dots,m \end{aligned} \quad \Rightarrow \quad \min_x \quad f(x) + \sum_{i=1}^{m} \mu_i h_i(x)^2$$

- Az optimum függ a μ_i súlyok választásától!
  - Túl kicsi μ_i: a kényszerek csak nagy hibával teljesülnek
  - Túl nagy μ_i: az eredeti loss-t ignoráljuk (+ numerikus problémák)
- Ellenben: univerzális módszer, egyszerű implementáció — gyakran alkalmazzuk!

Optimalizációs Módszerek

Kényszerek érvényesítése — Lagrange multiplikátor

Lagrange multiplikátor (“duális változó”)

$$\min_x \quad f(x)$$
$$\text{s.t.} \quad g(x) = 0$$

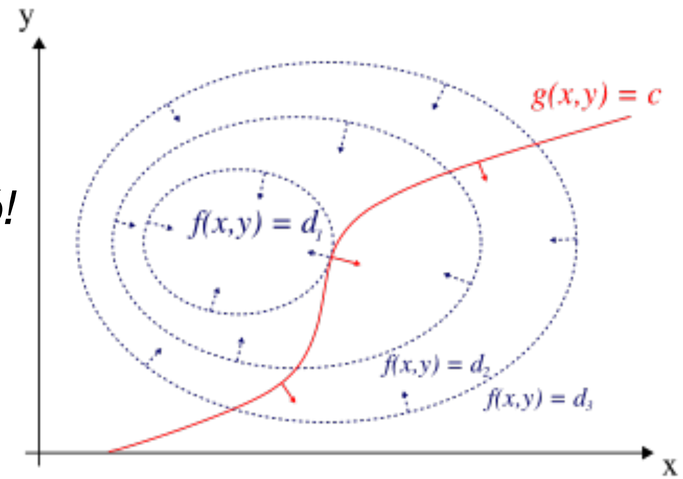
$$\min_x \max_\lambda \quad f(x) - \lambda g(x)$$

Lagrange duális (“nyeregponti”) probléma

$$\frac{\partial}{\partial x} = 0$$
$$\frac{\partial}{\partial \lambda} = 0$$

$$\nabla f(x) - \lambda \nabla g(x) = 0$$
$$g(x) = 0$$

Geometriai interpretáció!



Optimalizációs Módszerek

Kényszerek érvényesítése — Lagrange multiplikátor — Példa

- Másodfokú függvény + lineáris kényszerek:

$$\begin{aligned} \min_x \quad & \frac{1}{2}x^T A x - b^T x + c \\ \text{s.t.} \quad & Cx = d \end{aligned}$$

$$\min_x \max_\lambda \quad \frac{1}{2}x^T A x - b^T x + c - \lambda(Cx - d)$$

$$\begin{aligned} Ax - C^T\lambda &= b \\ Cx &= d \end{aligned}$$

$$\begin{bmatrix} A & C^T \\ C & 0 \end{bmatrix} \begin{bmatrix} x \\ \lambda \end{bmatrix} = \begin{bmatrix} b \\ d \end{bmatrix}$$

Optimalizációs Módszerek

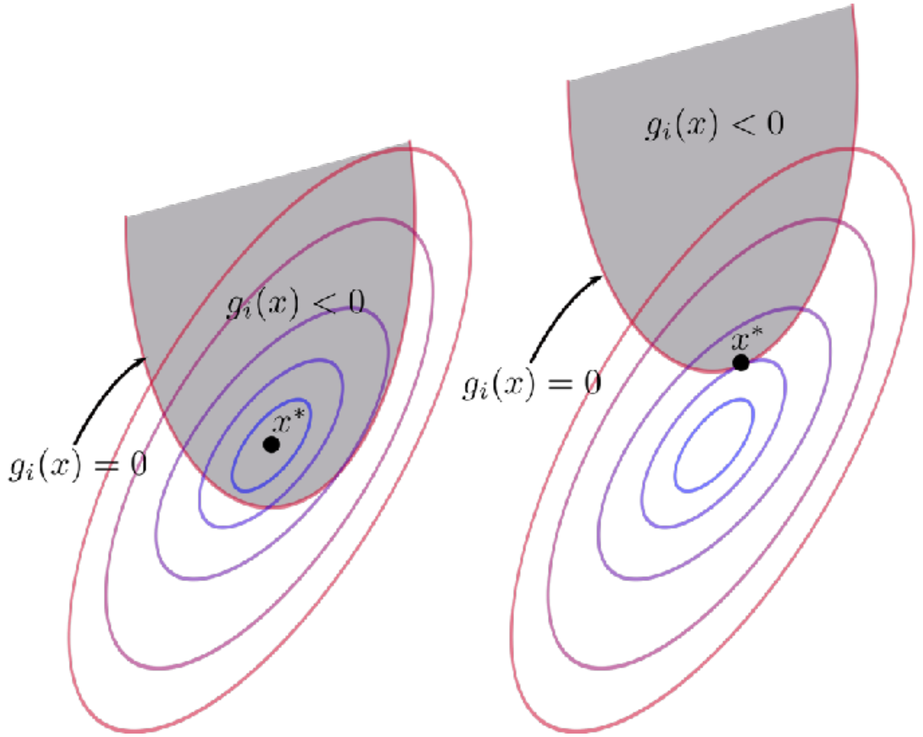
Optimalizáció kényszerekkel — KKT feltételek*

$$\min_x \quad f(x)$$
$$\text{s.t.} \quad g(x) \leq 0$$

$$\min_x \max_\lambda \quad f(x) - \lambda g(x)$$

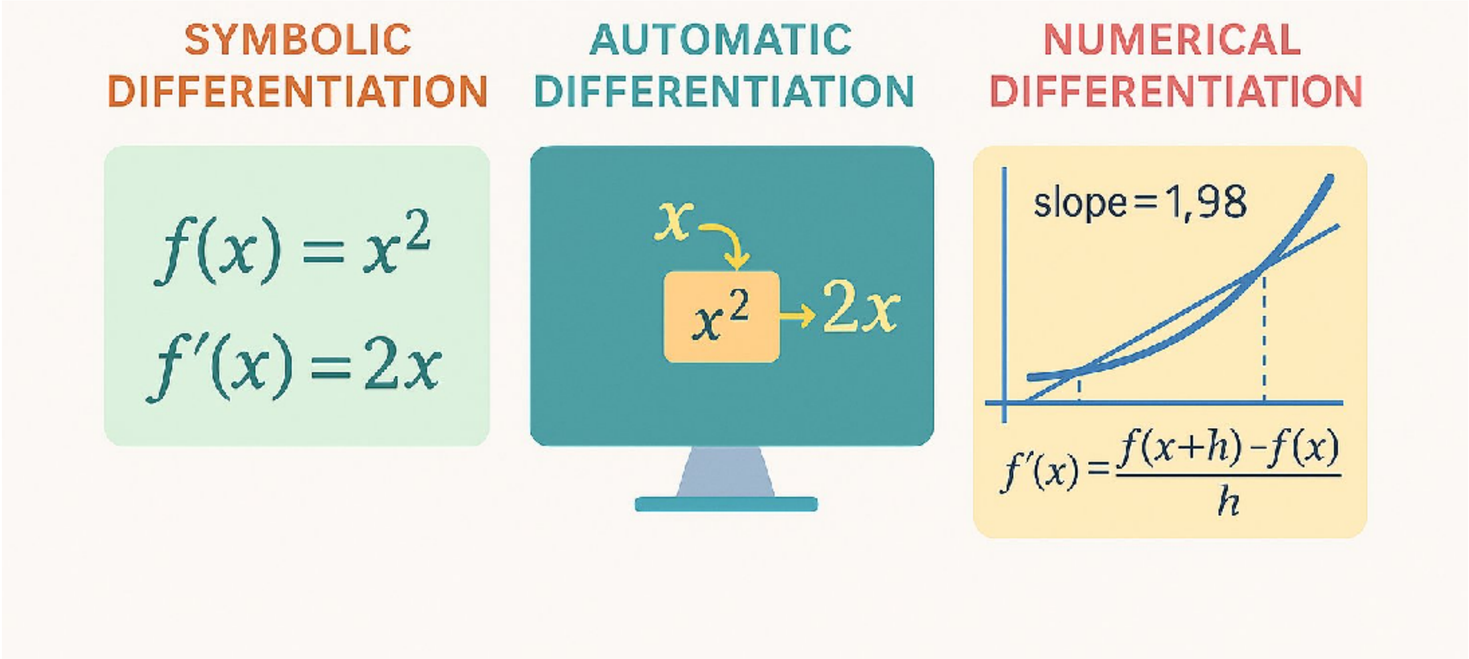
$$\nabla f(x) - \lambda \nabla g(x) = 0$$
$$g(x) \leq 0$$
$$\lambda \geq 0$$
$$g(x)\lambda = 0$$

Forrás: Wikipedia



Deriváltak Számítása

Hogyan számoljuk ki a deriváltakat?



Deriváltak Számítása

Szimbolikus Differenciálás

- A deriválás szabályait jól ismerjük…
- Elvileg bármilyen szokásos műveletekből álló függvény deriváltját le lehet vezetni szimbolikusan!
- Szimbolikus algebra szoftverekkel megtehető, de általában nem túl praktikus…
  - Nagyon számításigényes lehet
  - A kifejezések komplexitása rohamosan nő, egyszerűsítési szabályokat is alkalmazni kell!

$$\frac{d}{dx}(a) = 0$$
$$\frac{d}{dx}(x) = 1$$
$$\frac{d}{dx}(au) = a\frac{du}{dx}$$
$$\frac{d}{dx}(u+v-w) = \frac{du}{dx}+\frac{dv}{dx}-\frac{dw}{dx}$$
$$\frac{d}{dx}(uv) = u\frac{dv}{dx}+v\frac{du}{dx}$$
$$\frac{d}{dx}\left[\frac{u}{v}\right] = \frac{1}{v}\frac{du}{dx}-\frac{u}{v^2}\frac{dv}{dx}$$
$$\frac{d}{dx}(u^n) = nu^{n-1}\frac{du}{dx}$$
$$\frac{d}{dx}(\sqrt{u}) = \frac{1}{2\sqrt{u}}\frac{du}{dx}$$
$$\frac{d}{dx}\left[\frac{1}{u}\right] = -\frac{1}{u^2}\frac{du}{dx}$$
$$\frac{d}{dx}\left[\frac{1}{u^n}\right] = -\frac{n}{u^{n+1}}\frac{du}{dx}$$

$$\frac{d}{dx}[\ln u] = \frac{d}{dx}[\log_e u] = \frac{1}{u}\frac{du}{dx}$$
$$\frac{d}{dx}[\log_a u] = \log_a e\frac{1}{u}\frac{du}{dx}$$
$$\frac{d}{dx}e^u = e^u\frac{du}{dx}$$
$$\frac{d}{dx}a^u = a^u \ln a\frac{du}{dx}$$
$$\frac{d}{dx}(u^v) = vu^{v-1}\frac{du}{dx}+\ln u\ u^v\frac{dv}{dx}$$
$$\frac{d}{dx}\sin u = \cos u\frac{du}{dx}$$
$$\frac{d}{dx}\cos u = -\sin u\frac{du}{dx}$$
$$\frac{d}{dx}\tan u = \sec^2 u\frac{du}{dx}$$
$$\frac{d}{dx}\cot u = -\csc^2 u\frac{du}{dx}$$
$$\frac{d}{dx}\sec u = \sec u\tan u\frac{du}{dx}$$
$$\frac{d}{dx}\csc u = -\csc u\cot u\frac{du}{dx}$$

WOLFRAM MATHEMATICA

Maple

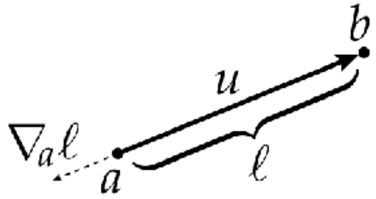
Maxima

SymPy

Deriváltak Számítása

Geometriai Differenciálás

- Van, hogy a derivált szemléletes jelentéssel bír, geometriai megfontolásokból levezethető
- Tipikus példák:
  - Él hossza
  - Háromszög területe
  - Továbbiak: LINK
- Szimbolikus módszerek sokkal bonyolultabb formulát találnak…



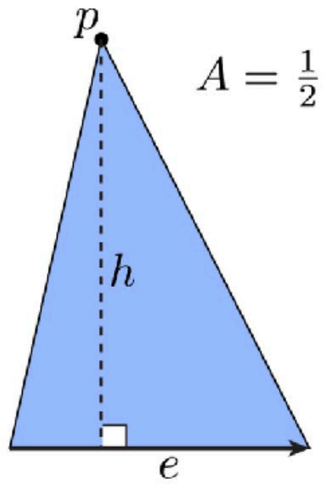
Mathematica output:

```
(2 (b2 - c2) (-b2 c1 + a2 (-b1 + c1) + a1 (b2 - c2) +
b1 c2) + 2 (b3 - c3) (-b3 c1 + a3 (-b1 + c1) + a1 (b3
- c3) + b1 c3))/(4 Sqrt((a2 b1 - a1 b2 - a2 c1 + b2 c1
+ a1 c2 - p b1 c2)^2 + (a3 b1 - a1 b3 - a3 c1 + b3 c1
+ a1 c3 - b1 c3)^2 + (a3 b2 - a2 b3 - a3 c2 + b3 c2 +
a2 c3 - b2 c3)^2)), (2 (b1 - c1) (a2 (b1 - c1) + b2 c1
- b1 c2 + a1 (-b2 + c2)) + 2 (b3 - c3) (-b3 c2 + a3 (-
b2 + c2) + a2 (b3 - c3) + b2 c3))/(4 Sqrt((a2 b1 - a1
b2 - a2 c1 + b2 c1 + a1 c2 - b1 c2)^2 + (a3 b1 - a1 b3
- a3 c1 + b3 c1 + a1 c3 - b1 c3)^2 + (a3 b2 - a2 b3 -
a3 c2 + b3 c2 + a2 c3 - b2 c3)^2)), (2 (b1 - c1) (a3
(b1 - c1) + b3 c1 - b1 c3 + a1 (-b3 + c3)) + 2 (b2 -
c2) (a3 (b2 - c2) + b3 c2 - b2 c3 + a2 (-b3 + c3)))/(4
Sqrt((a2 b1 - a1 b2 - a2 c1 + b2 c1 + a1 c2 - b1 c2)^2
+ (a3 b1 - a1 b3 - a3 c1 + b3 c1 + a1 c3 - b1 c3)^2 +
(a3 b2 - a2 b3 - a3 c2 + b3 c2 + a2 c3 - b2 c3)^2))
```

"Geometric" derivative:

$$\nabla_p A = \tfrac{1}{2} N \times e$$

Forrás: K. Crane



Deriváltak Számítása

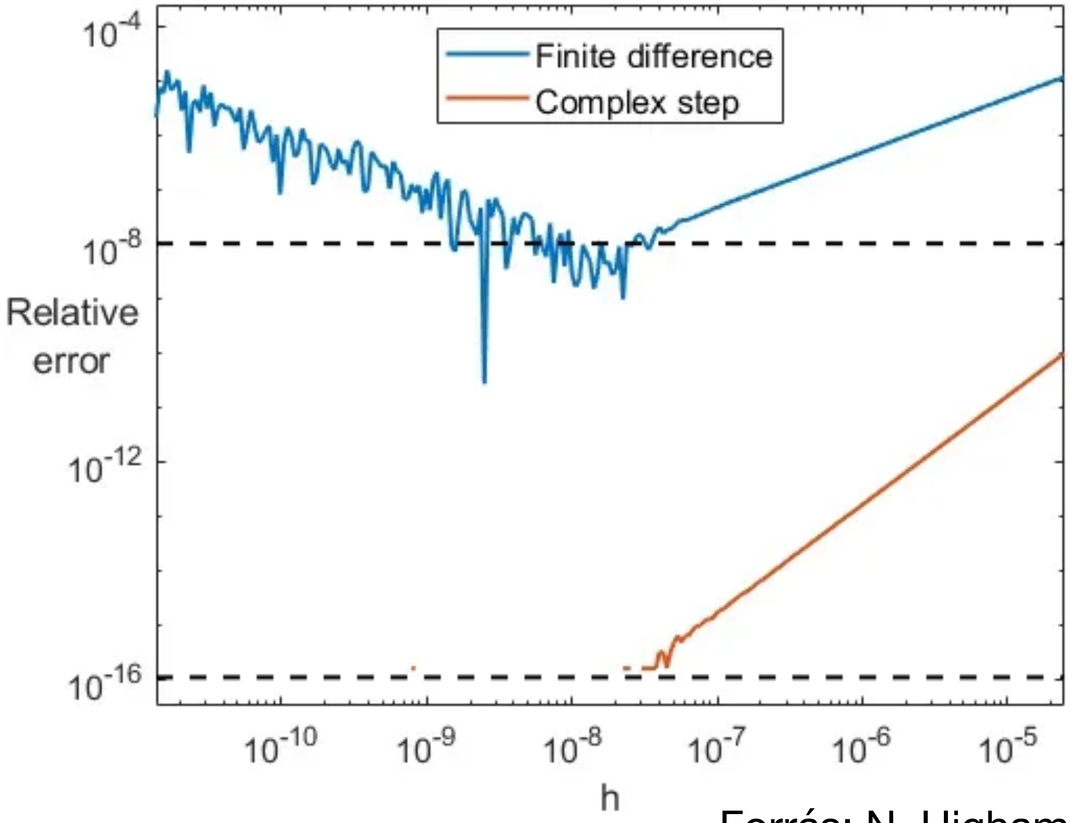
Numerikus Differenciálás — Véges Differencia

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

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

A h lépésköz megválasztása:

- Túl nagy — durva közelítés!
- Túl kicsi — kerekítési hiba kezd dominálni!



Forrás: N. Higham

Sok változó esetén nagyon költséges lenne!

(Bár lehet trükközni, ha a loss kellően “lokális”)

Deriváltak Számítása

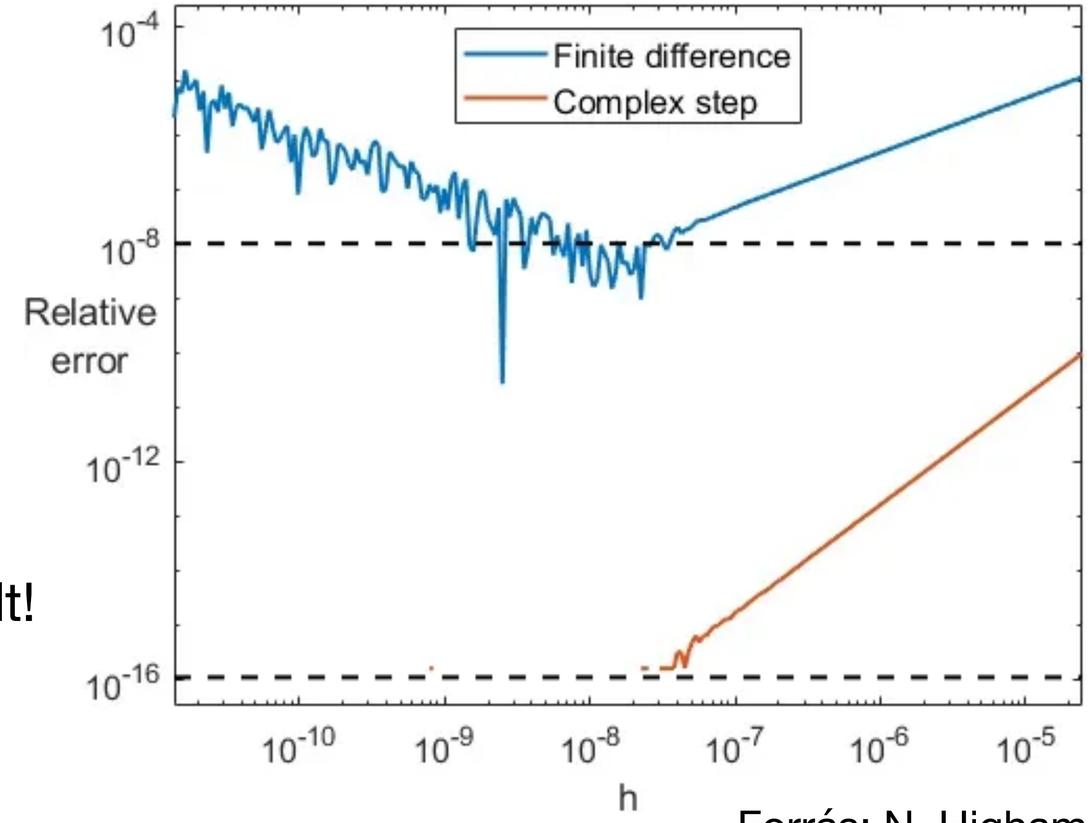
Numerikus Differenciálás — Komplex lépésköz

$$f(x+ih) = f(x) + ihf'(x) + \cdots \qquad i^2 = -1$$

$$f'(x) \approx \mathrm{Im}\left(\frac{f(x+ih)}{h}\right)$$

Nem tartalmaz kivonást
nagyon kicsi lépésközzel nagyon pontos derivált!

$$h = 10^{-200}!!!$$



Forrás: N. Higham

Deriváltak Számítása

Automatikus Differenciálás

- Kompozit függvény:

$$F(x) = h(g(f(x)))$$

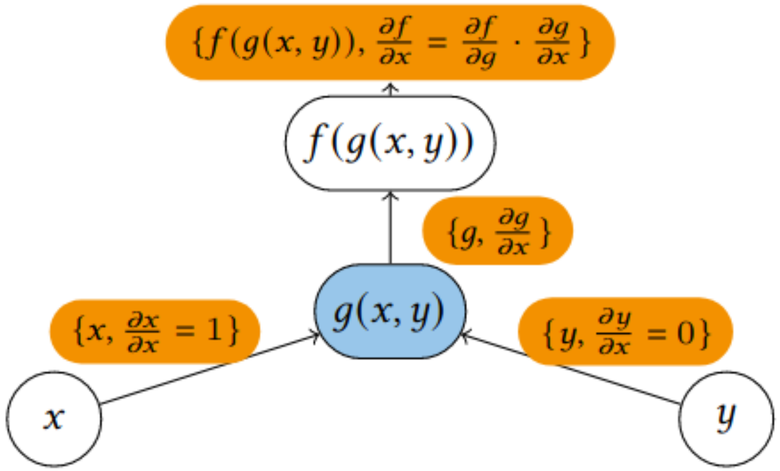
- Deriválás láncszabálya:

$$F'(x) = h'(g(f(x))) \cdot g'(f(x))) \cdot f'(x)$$

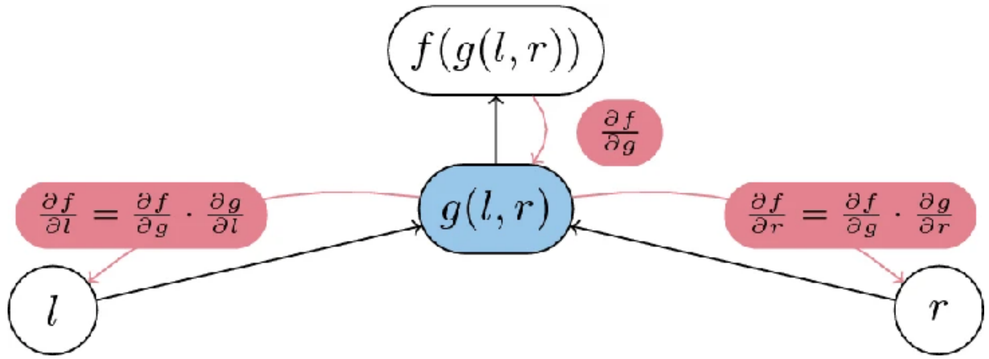
- **Automatikus / Algoritmikus Differenciálás (AutoDiff, AD)** — láncszabály alkalmazása kompozit függvényre (sőt tetszőleges numerikus programkódra)
- Kétféle üzemmód:
  - Forward Mode
  - Reverse Mode (Backpropagation / Adjoint)

Deriváltak Számítása

Automatikus Differenciálás — Forward vs. Reverse



Forward AD



Reverse AD

Deriváltak Számítása

Automatikus Differenciálás — Forward Mód — Duális számok

$$\mathcal{F}(x) = f(x) + f'(x)\epsilon$$

$$\epsilon^2 = 0$$

$$\mathcal{F}(x) \pm \mathcal{G}(x) = (f(x) \pm g(x)) + (f'(x) \pm g'(x))\epsilon$$

$$\mathcal{F}(x)\mathcal{G}(x) = f(x)g(x) + (f'(x)g(x) + f(x)g'(x))\epsilon$$

$$\frac{\mathcal{F}(x)}{\mathcal{G}(x)} = \frac{f(x)}{g(x)} + \frac{f'(x)g(x) - f(x)g'(x)}{g^2(x)}\epsilon$$

Nagyon egyszerű implementálni
~~de csak első deriváltakra működik!~~

Periodica Polytechnica Electrical Engineering and Computer Science, 65(1), pp. 1–10, 2021

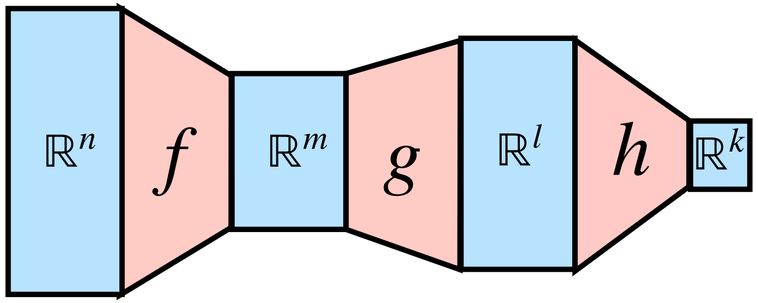
Higher Order Automatic Differentiation with Dual Numbers

László Szirmay-Kalos[1*]

https://pp.bme.hu/eecs/article/view/16341

Deriváltak Számítása

Automatikus Differenciálás — Forward Mód

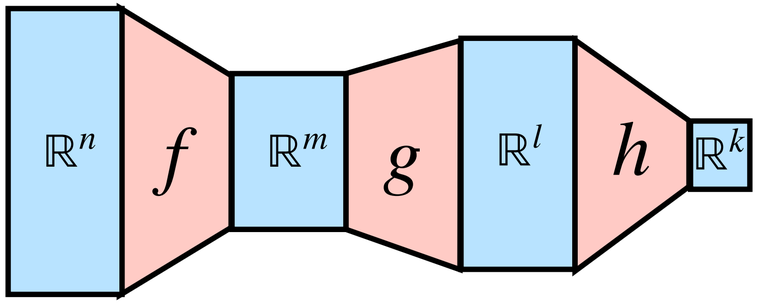


$$F'(x) = h'(g(f(x))) \cdot g'(f(x)) \cdot f'(x)$$

$$\underset{k \times n}{F'} = \underset{k \times l}{h'} \cdot \Big(\underset{l \times m}{g'} \cdot \underset{m \times n}{f'} \Big)$$

Deriváltak Számítása

Automatikus Differenciálás — Forward Mód



$$F'(x) = h'(g(f(x))) \cdot g'(f(x)) \cdot f'(x)$$

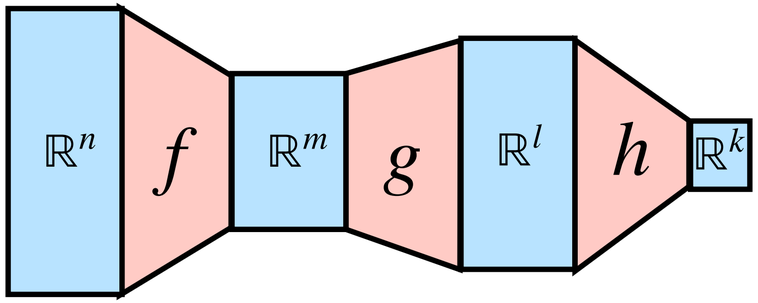
$$\underset{k \times n}{F'} = \underset{k \times l}{h'} \cdot \left(\underset{l \times m}{g'} \cdot \underset{m \times n}{f'} \right)$$

$$x$$

$$\frac{\partial x}{\partial x} = 1$$

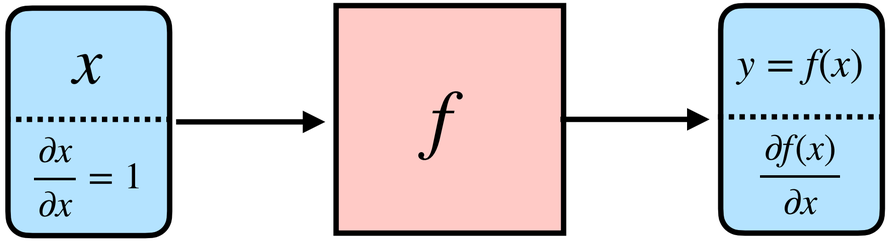
Deriváltak Számítása

Automatikus Differenciálás — Forward Mód



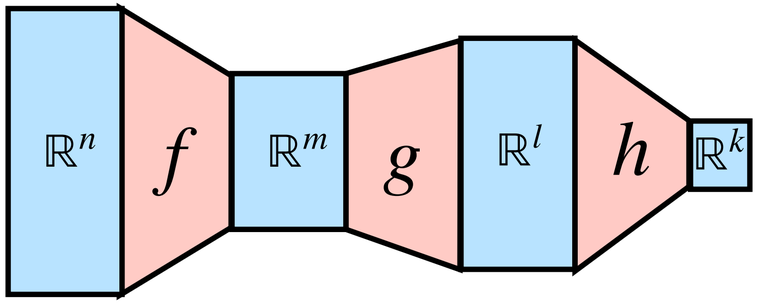
$$F'(x) = h'(g(f(x))) \cdot g'(f(x)) \cdot f'(x)$$

$$\underbrace{F'}_{k \times n} = \underbrace{h'}_{k \times l} \cdot \Big(\underbrace{g'}_{l \times m} \cdot \underbrace{f'}_{m \times n} \Big)$$



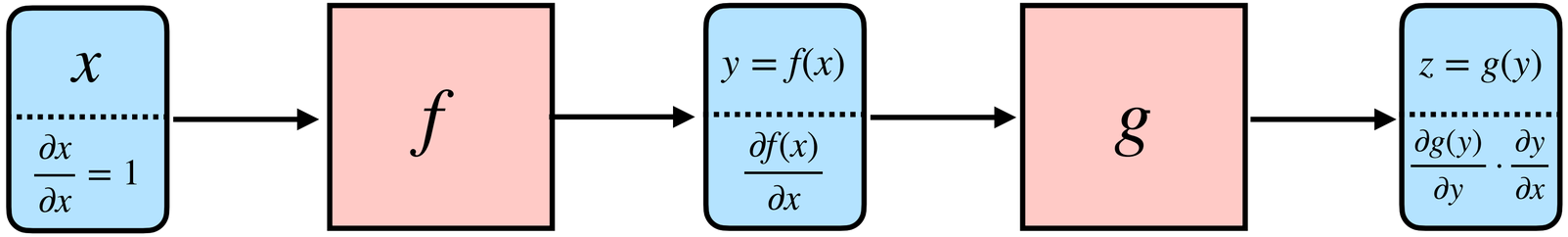
Deriváltak Számítása

Automatikus Differenciálás — Forward Mód



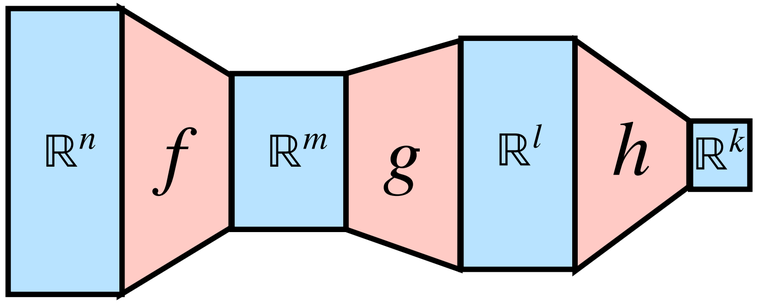
$$F'(x) = h'(g(f(x))) \cdot g'(f(x)) \cdot f'(x)$$

$$\underbrace{F'}_{k \times n} = \underbrace{h'}_{k \times l} \cdot \Big(\underbrace{g'}_{l \times m} \cdot \underbrace{f'}_{m \times n} \Big)$$



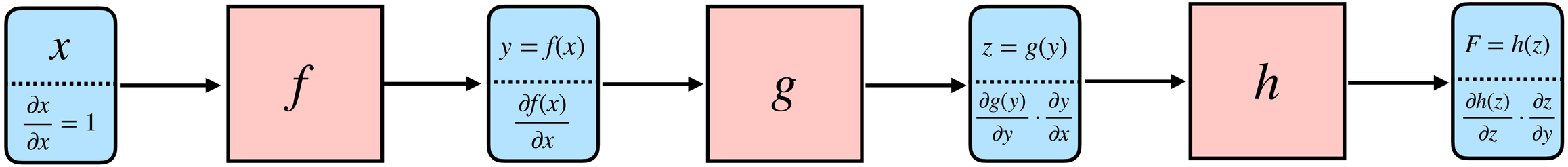
Deriváltak Számítása

Automatikus Differenciálás — Forward Mód



$$F'(x) = h'(g(f(x))) \cdot g'(f(x)) \cdot f'(x)$$

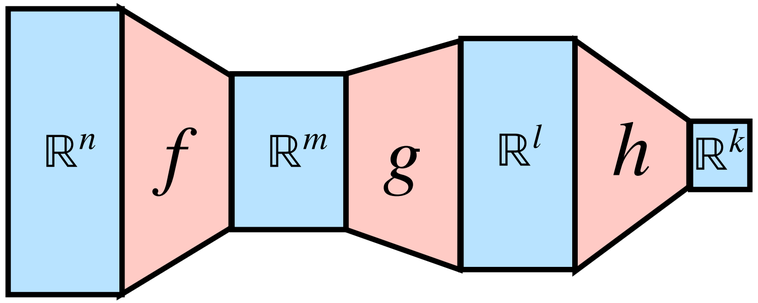
$$\underbrace{F'}_{k \times n} = \underbrace{h'}_{k \times l} \cdot \Big(\underbrace{g'}_{l \times m} \cdot \underbrace{f'}_{m \times n} \Big)$$



Deriváltak Számítása

Automatikus Differenciálás — Reverse Mód

a.k.a:
“Backpropagation”
“Adjoint Method”



$$F'(x) = h'(g(f(x))) \cdot g'(f(x)) \cdot f'(x)$$

$$\underset{k \times n}{F'} = \left(\underset{k \times l}{h'} \cdot \underset{l \times m}{g'} \right) \cdot \underset{m \times n}{f'}$$

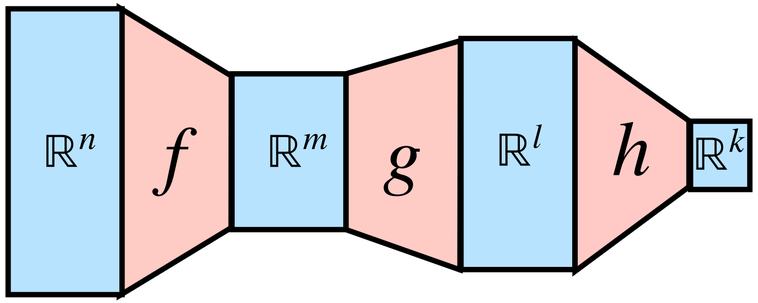
Sokkal hatékonyabb, ha $l < n$!

Viszont: el kell tárolni
a köztes számítási lépéseket!

Deriváltak Számítása

Automatikus Differenciálás — Reverse Mód

a.k.a:
“Backpropagation”
“Adjoint Method”

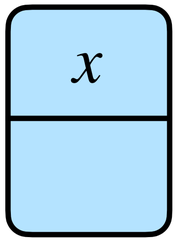


$$F'(x) = h'(g(f(x))) \cdot g'(f(x)) \cdot f'(x)$$

$$\underset{k \times n}{F'} = \left(\underset{k \times l}{h'} \cdot \underset{l \times m}{g'} \right) \cdot \underset{m \times n}{f'}$$

Sokkal hatékonyabb, ha $l < n$!

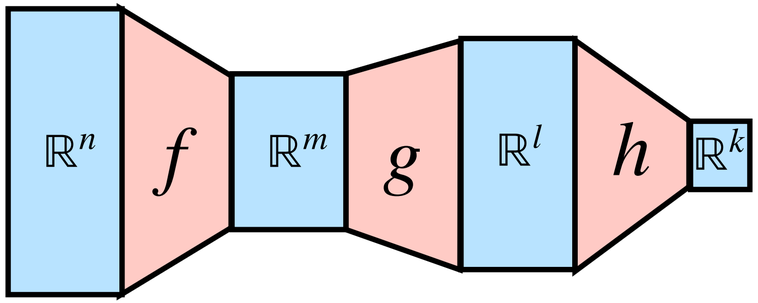
Viszont: el kell tárolni
a köztes számítási lépéseket!



Deriváltak Számítása

Automatikus Differenciálás — Reverse Mód

a.k.a:
“Backpropagation”
“Adjoint Method”

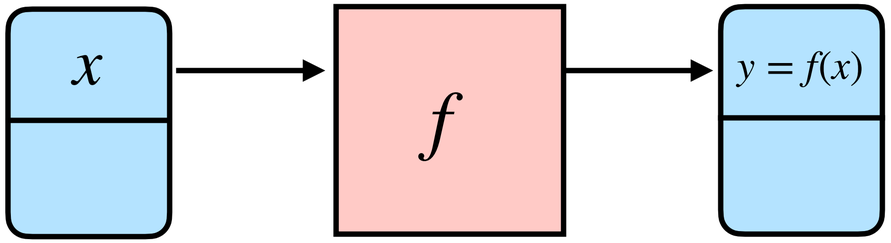


$$F'(x) = h'(g(f(x))) \cdot g'(f(x)) \cdot f'(x)$$

$$\underset{k \times n}{F'} = \left(\underset{k \times l}{h'} \cdot \underset{l \times m}{g'} \right) \cdot \underset{m \times n}{f'}$$

Sokkal hatékonyabb, ha $l < n$!

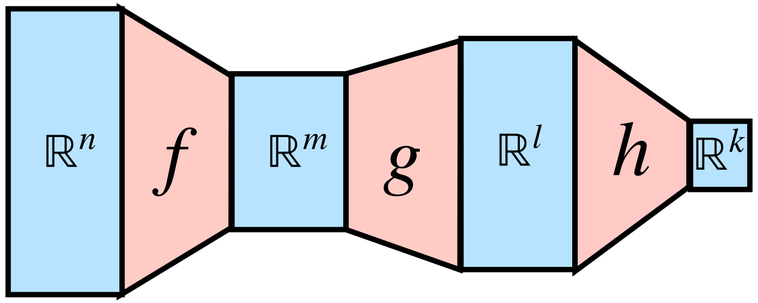
Viszont: el kell tárolni
a köztes számítási lépéseket!



Deriváltak Számítása

Automatikus Differenciálás — Reverse Mód

a.k.a:
“Backpropagation”
“Adjoint Method”

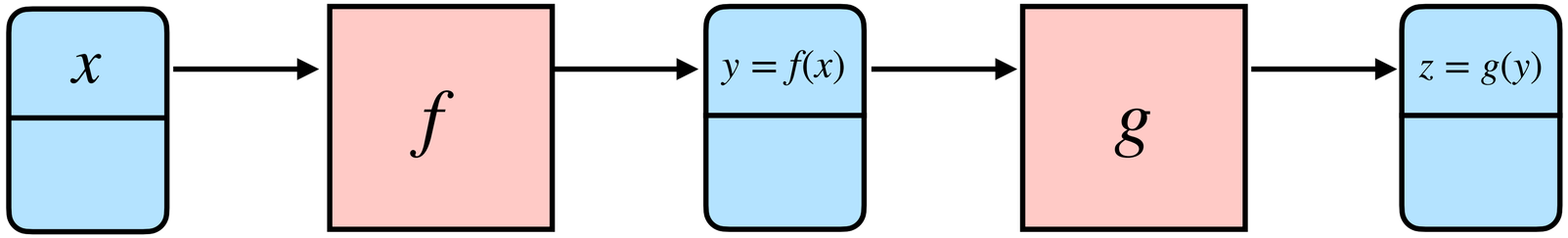


$$F'(x) = h'(g(f(x))) \cdot g'(f(x)) \cdot f'(x)$$

$$\underset{k \times n}{F'} = \left(\underset{k \times l}{h'} \cdot \underset{l \times m}{g'} \right) \cdot \underset{m \times n}{f'}$$

Sokkal hatékonyabb, ha $l < n$!

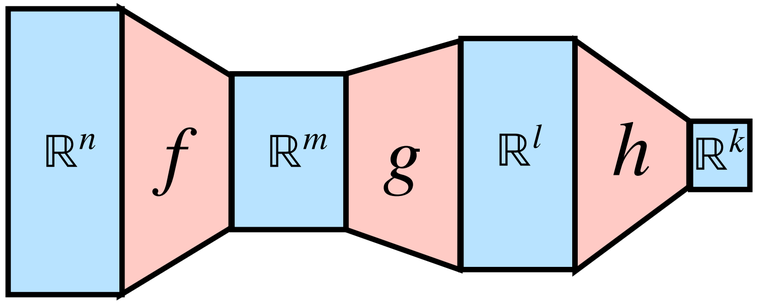
Viszont: el kell tárolni
a köztes számítási lépéseket!



Deriváltak Számítása

Automatikus Differenciálás — Reverse Mód

a.k.a:
“Backpropagation”
“Adjoint Method”

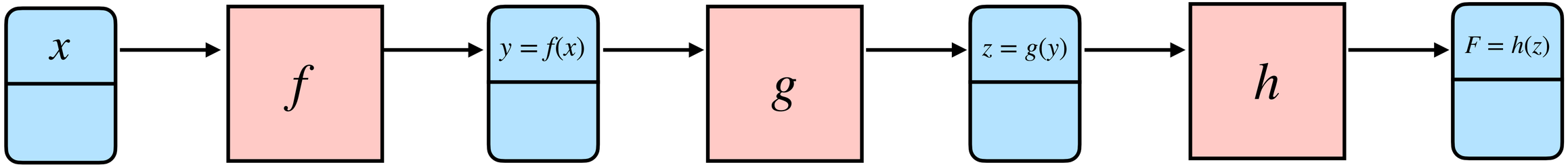


$$F'(x) = h'(g(f(x))) \cdot g'(f(x)) \cdot f'(x)$$

$$\underset{k \times n}{F'} = \left(\underset{k \times l}{h'} \cdot \underset{l \times m}{g'} \right) \cdot \underset{m \times n}{f'}$$

Sokkal hatékonyabb, ha $l < n$!

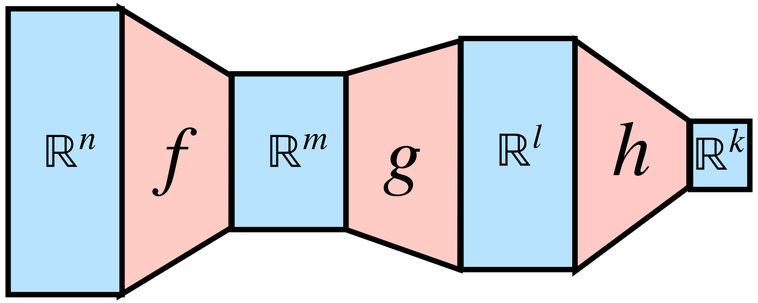
Viszont: el kell tárolni
a köztes számítási lépéseket!



Deriváltak Számítása

Automatikus Differenciálás — Reverse Mód

a.k.a:
“Backpropagation”
“Adjoint Method”

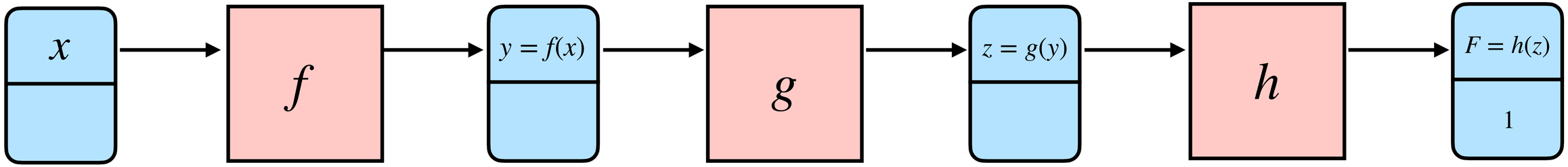


$$F'(x) = h'(g(f(x))) \cdot g'(f(x)) \cdot f'(x)$$

$$\underset{k\times n}{F'} = \left(\underset{k\times l}{h'} \cdot \underset{l\times m}{g'} \right) \cdot \underset{m\times n}{f'}$$

Sokkal hatékonyabb, ha $l < n$!

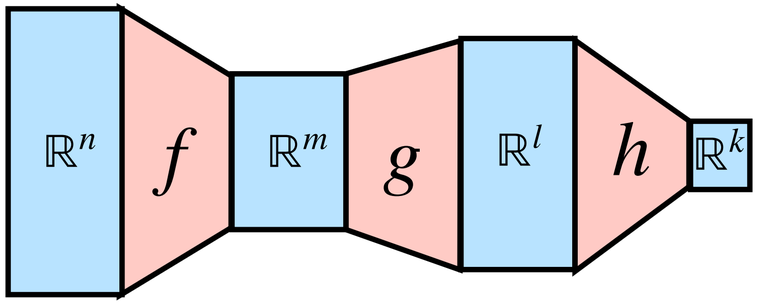
Viszont: el kell tárolni
a köztes számítási lépéseket!



Deriváltak Számítása

Automatikus Differenciálás — Reverse Mód

a.k.a:
“Backpropagation”
“Adjoint Method”

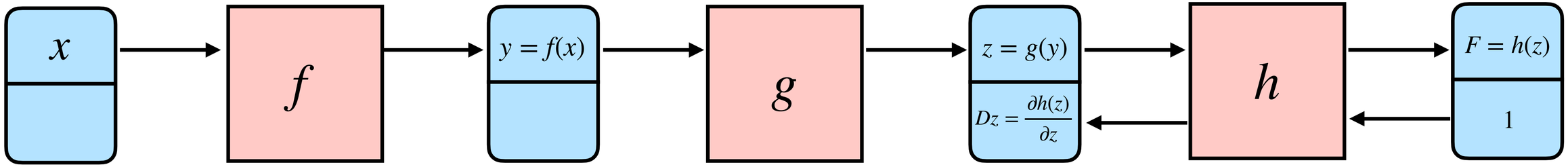


$$F'(x) = h'(g(f(x))) \cdot g'(f(x)) \cdot f'(x)$$

$$\underset{k \times n}{F'} = \left(\underset{k \times l}{h'} \cdot \underset{l \times m}{g'} \right) \cdot \underset{m \times n}{f'}$$

Sokkal hatékonyabb, ha $l < n$!

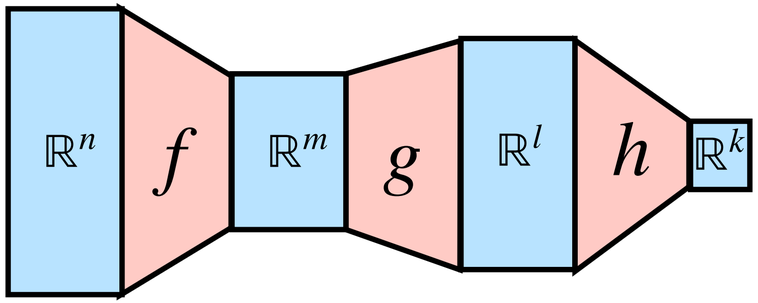
Viszont: el kell tárolni
a köztes számítási lépéseket!



Deriváltak Számítása

Automatikus Differenciálás — Reverse Mód

a.k.a:
“Backpropagation”
“Adjoint Method”

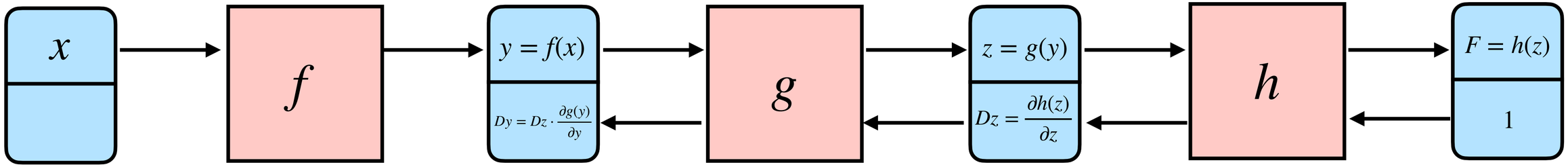


$$F'(x) = h'(g(f(x))) \cdot g'(f(x)) \cdot f'(x)$$

$$\underset{k \times n}{F'} = \left(\underset{k \times l}{h'} \cdot \underset{l \times m}{g'} \right) \cdot \underset{m \times n}{f'}$$

Sokkal hatékonyabb, ha $l < n$!

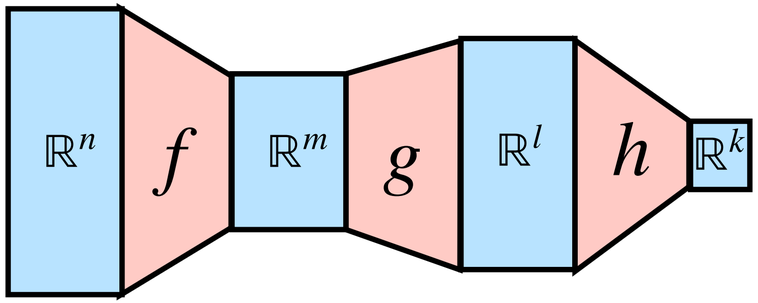
Viszont: el kell tárolni
a köztes számítási lépéseket!



Deriváltak Számítása

Automatikus Differenciálás — Reverse Mód

a.k.a:
“Backpropagation”
“Adjoint Method”

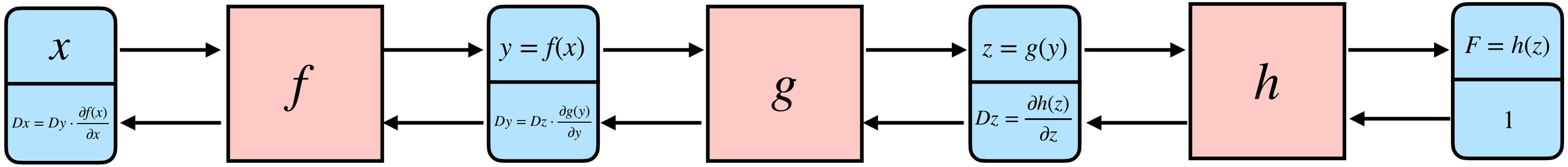


$$F'(x) = h'(g(f(x))) \cdot g'(f(x)) \cdot f'(x)$$

$$\underset{k \times n}{F'} = \left(\underset{k \times l}{h'} \cdot \underset{l \times m}{g'} \right) \cdot \underset{m \times n}{f'}$$

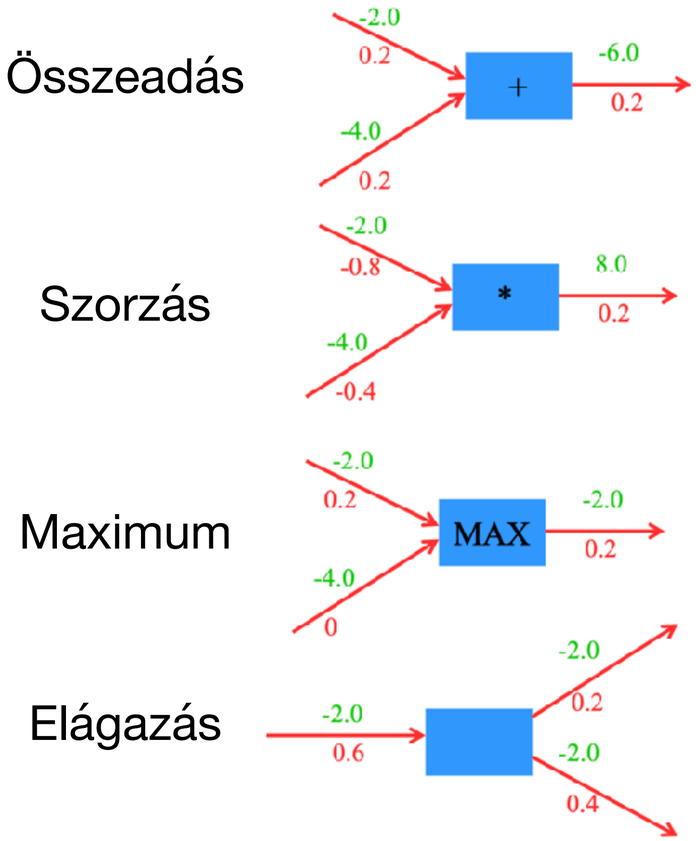
Sokkal hatékonyabb, ha $l < n$!

Viszont: el kell tárolni
a köztes számítási lépéseket!



Deriváltak Számítása

Automatikus Differenciálás — Reverse Mód — Számítási Gráfok



Deriváltak tetszőleges számítási gráfon terjedhetnek!

Forrás: Szemenyei M.

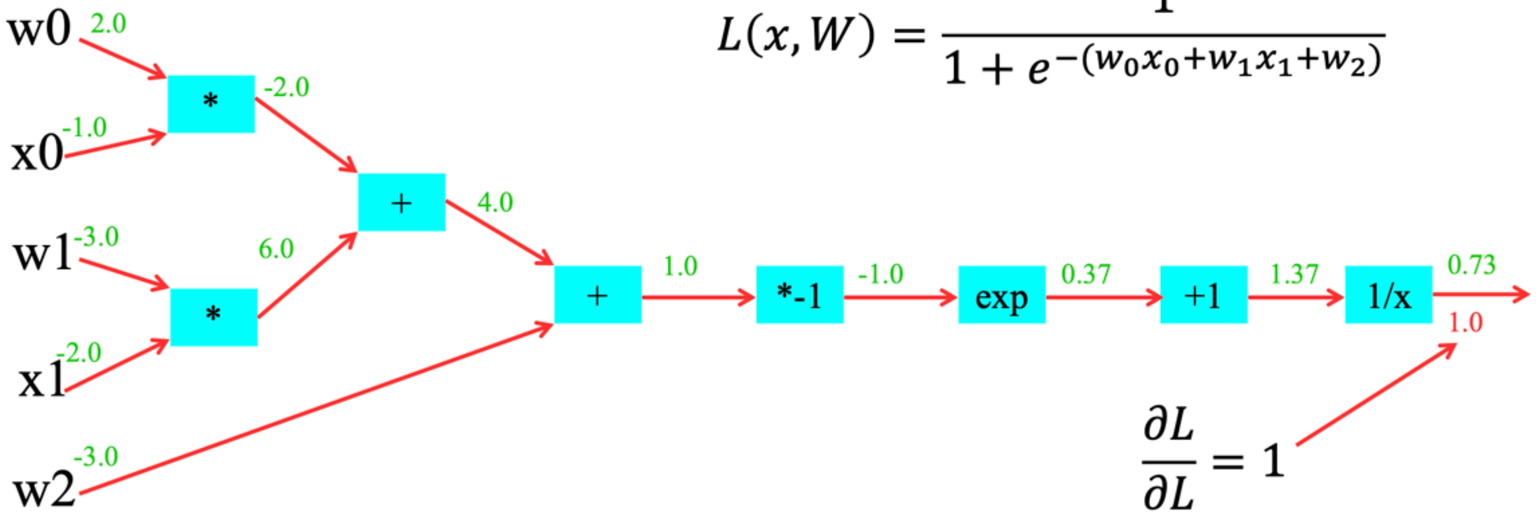
Deriváltak Számítása

Automatikus Differenciálás — Reverse Mód — Példa

Forrás: Szemenyei M.

Deriváltak Számítása

Automatikus Differenciálás — Reverse Mód — Példa



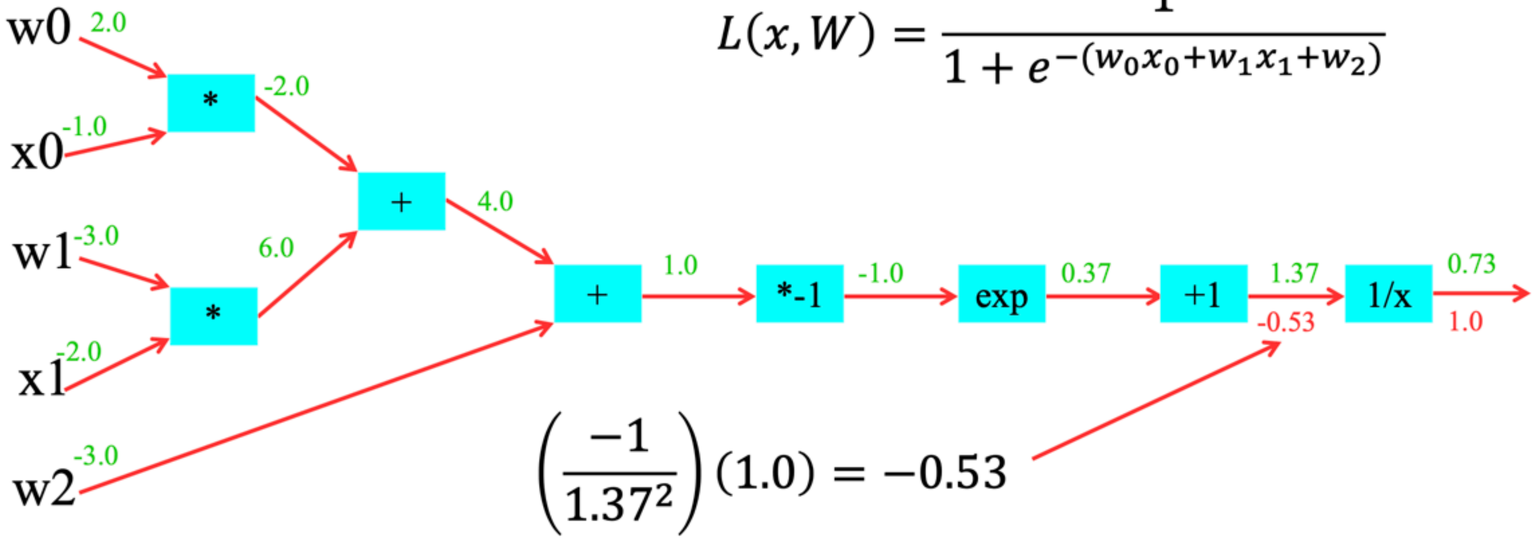
$$f(x) = e^x \rightarrow \frac{\partial f}{\partial x} = e^x \qquad f(x) = a + x \rightarrow \frac{\partial f}{\partial x} = 1$$

$$f(x) = ax \rightarrow \frac{\partial f}{\partial x} = a \qquad f(x) = \frac{1}{x} \rightarrow \frac{\partial f}{\partial x} = -\frac{1}{x^2}$$

Forrás: Szemenyei M.

Deriváltak Számítása

Automatikus Differenciálás — Reverse Mód — Példa



$$L(x,W) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$

$$\left(\frac{-1}{1.37^2}\right)(1.0) = -0.53$$

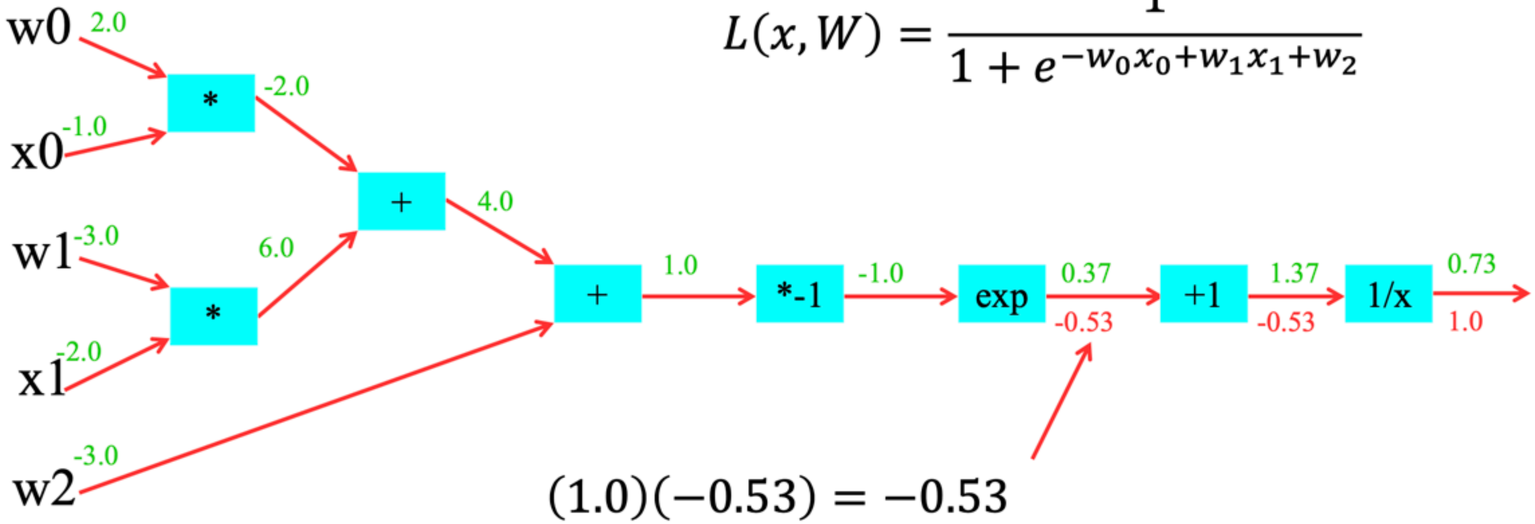
$$f(x) = e^x \rightarrow \frac{\partial f}{\partial x} = e^x \qquad f(x) = a + x \rightarrow \frac{\partial f}{\partial x} = 1$$

$$f(x) = ax \rightarrow \frac{\partial f}{\partial x} = a \qquad f(x) = \frac{1}{x} \rightarrow \frac{\partial f}{\partial x} = -\frac{1}{x^2}$$

Forrás: Szemenyei M.

Deriváltak Számítása

Automatikus Differenciálás — Reverse Mód — Példa



$$L(x, W) = \frac{1}{1 + e^{-w_0 x_0 + w_1 x_1 + w_2}}$$

$$(1.0)(-0.53) = -0.53$$

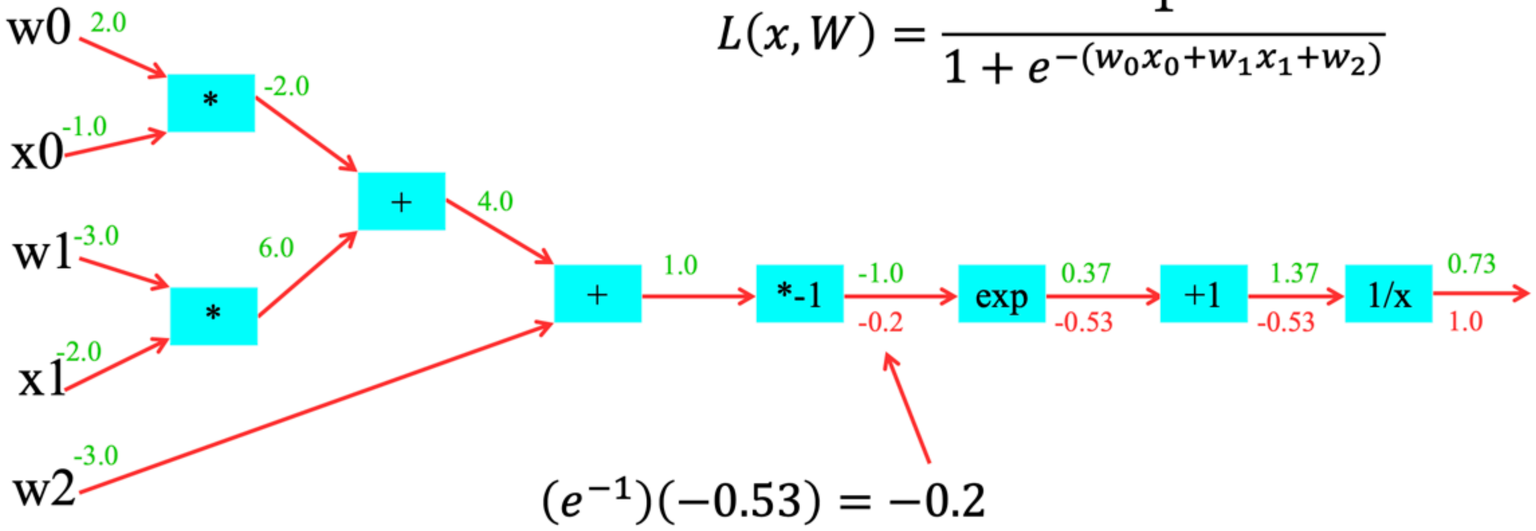
$$f(x) = e^x \rightarrow \frac{\partial f}{\partial x} = e^x \qquad f(x) = a + x \rightarrow \frac{\partial f}{\partial x} = 1$$

$$f(x) = ax \rightarrow \frac{\partial f}{\partial x} = a \qquad f(x) = \frac{1}{x} \rightarrow \frac{\partial f}{\partial x} = -\frac{1}{x^2}$$

Forrás: Szemenyei M.

Deriváltak Számítása

Automatikus Differenciálás — Reverse Mód — Példa



$$L(x, W) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$

$$(e^{-1})(-0.53) = -0.2$$

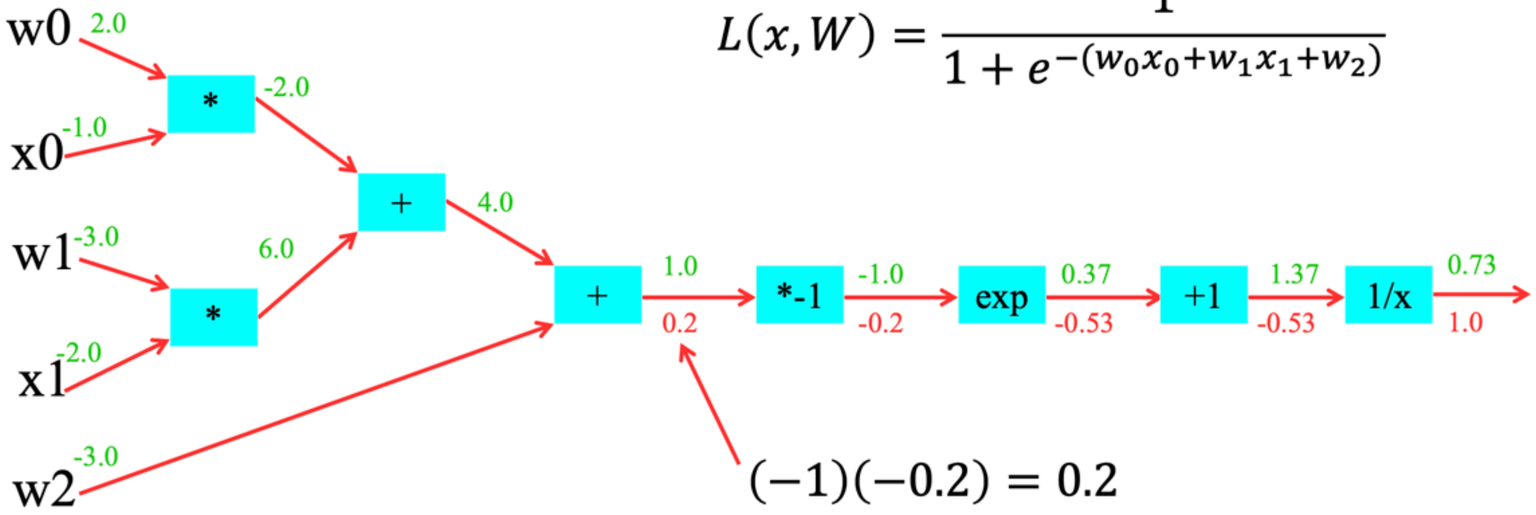
$$f(x) = e^x \rightarrow \frac{\partial f}{\partial x} = e^x \qquad f(x) = a + x \rightarrow \frac{\partial f}{\partial x} = 1$$

$$f(x) = ax \rightarrow \frac{\partial f}{\partial x} = a \qquad f(x) = \frac{1}{x} \rightarrow \frac{\partial f}{\partial x} = -\frac{1}{x^2}$$

Forrás: Szemenyei M.

Deriváltak Számítása

Automatikus Differenciálás — Reverse Mód — Példa



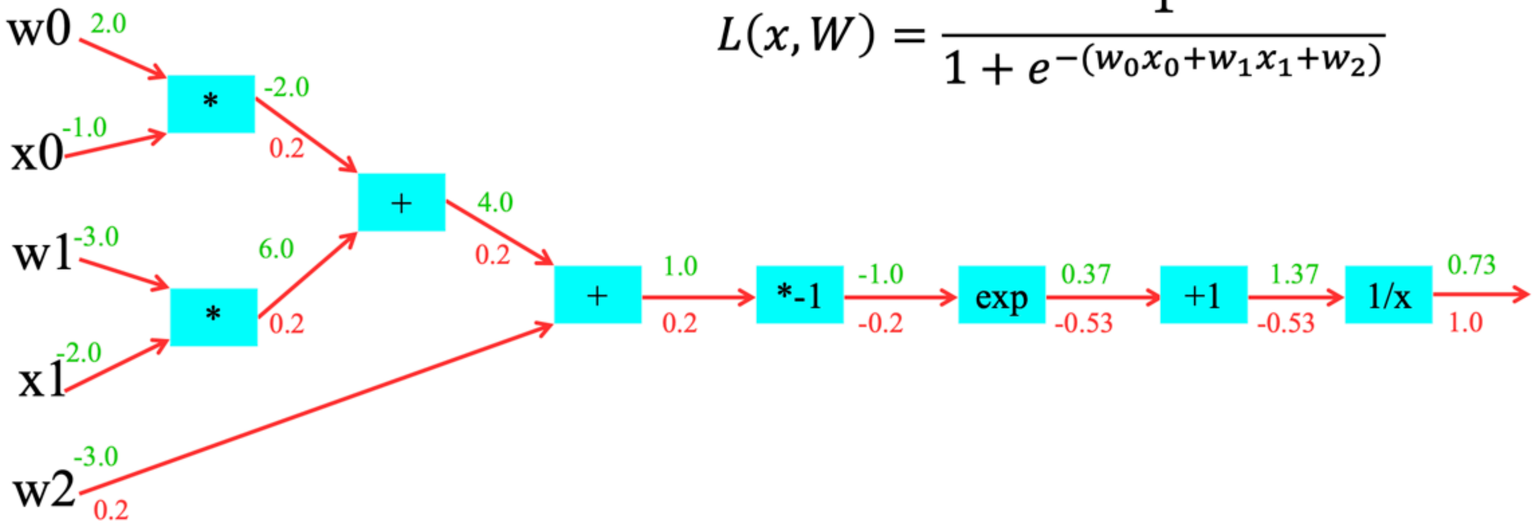
$$f(x) = e^x \rightarrow \frac{\partial f}{\partial x} = e^x \qquad f(x) = a + x \rightarrow \frac{\partial f}{\partial x} = 1$$

$$f(x) = ax \rightarrow \frac{\partial f}{\partial x} = a \qquad f(x) = \frac{1}{x} \rightarrow \frac{\partial f}{\partial x} = -\frac{1}{x^2}$$

Forrás: Szemenyei M.

Deriváltak Számítása

Automatikus Differenciálás — Reverse Mód — Példa



$$L(x, W) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$

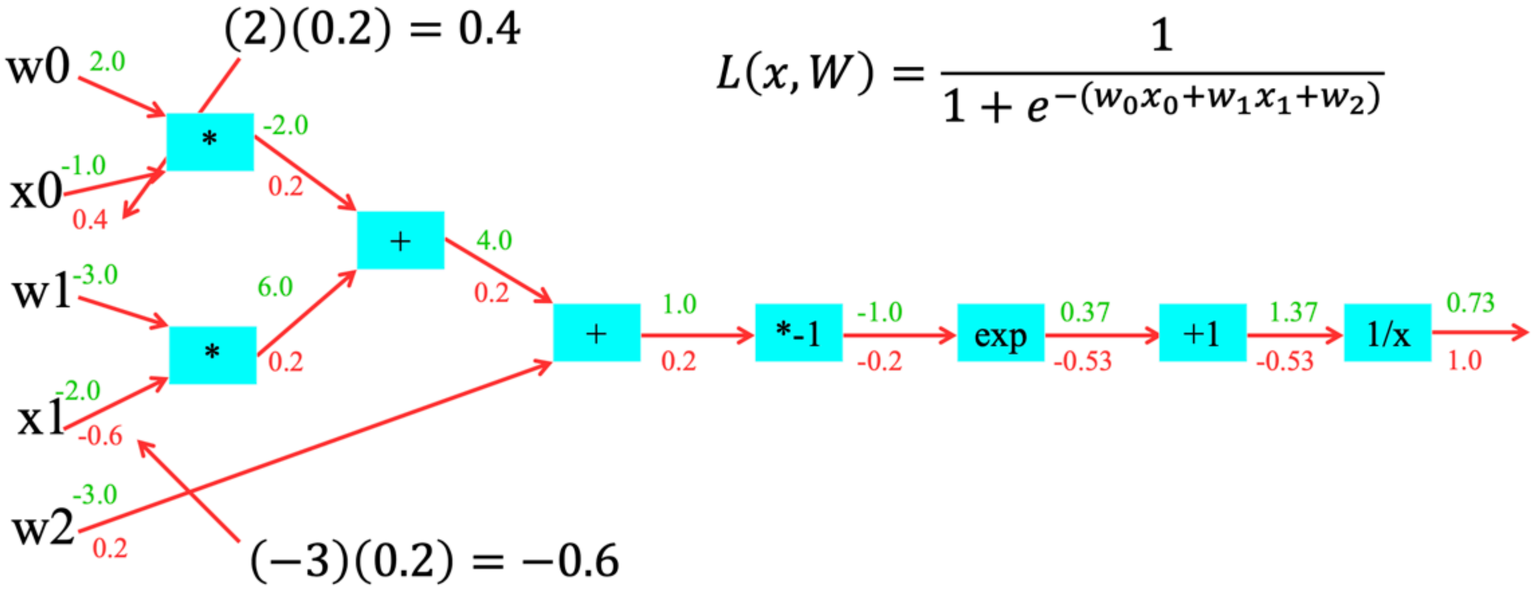
$$f(x) = e^x \rightarrow \frac{\partial f}{\partial x} = e^x \qquad f(x) = a + x \rightarrow \frac{\partial f}{\partial x} = 1$$

$$f(x) = ax \rightarrow \frac{\partial f}{\partial x} = a \qquad f(x) = \frac{1}{x} \rightarrow \frac{\partial f}{\partial x} = -\frac{1}{x^2}$$

Forrás: Szemenyei M.

Deriváltak Számítása

Automatikus Differenciálás — Reverse Mód — Példa



$$L(x, W) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$

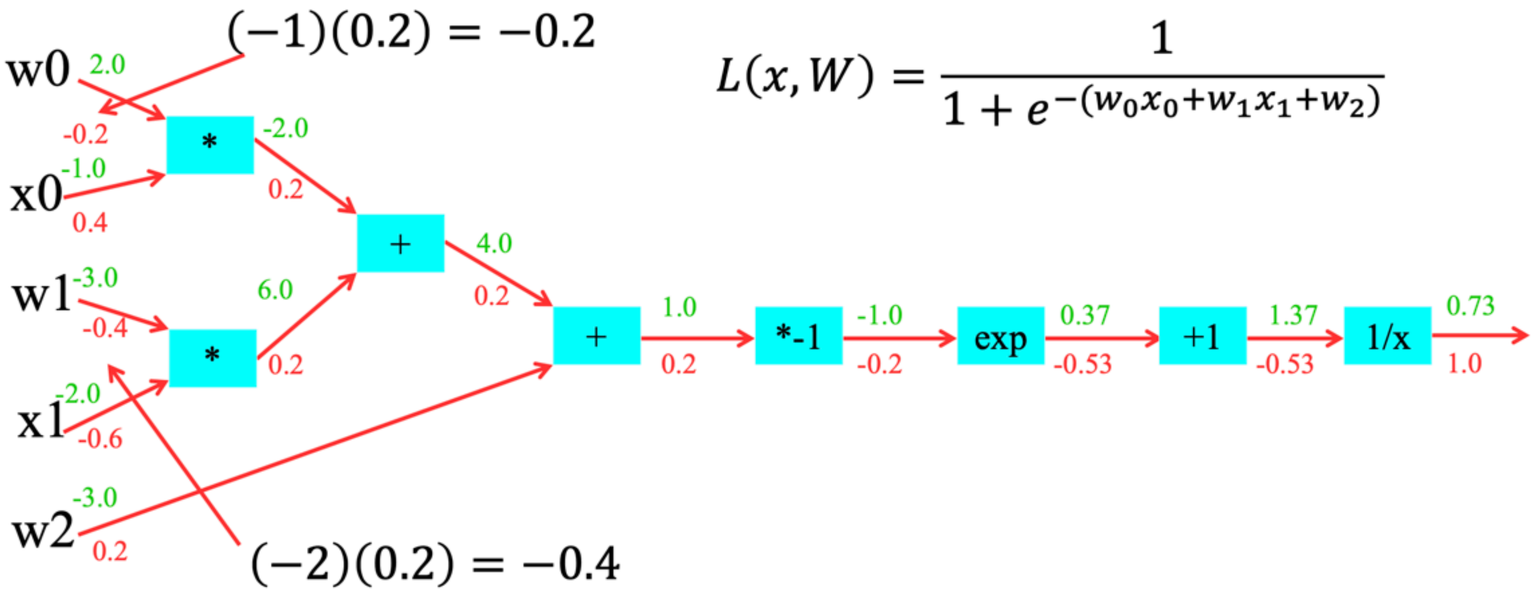
$$f(x) = e^x \rightarrow \frac{\partial f}{\partial x} = e^x \qquad f(x) = a + x \rightarrow \frac{\partial f}{\partial x} = 1$$

$$f(x) = ax \rightarrow \frac{\partial f}{\partial x} = a \qquad f(x) = \frac{1}{x} \rightarrow \frac{\partial f}{\partial x} = -\frac{1}{x^2}$$

Forrás: Szemenyei M.

Deriváltak Számítása

Automatikus Differenciálás — Reverse Mód — Példa



$$L(x, W) = \frac{1}{1 + e^{-(w_0 x_0 + w_1 x_1 + w_2)}}$$

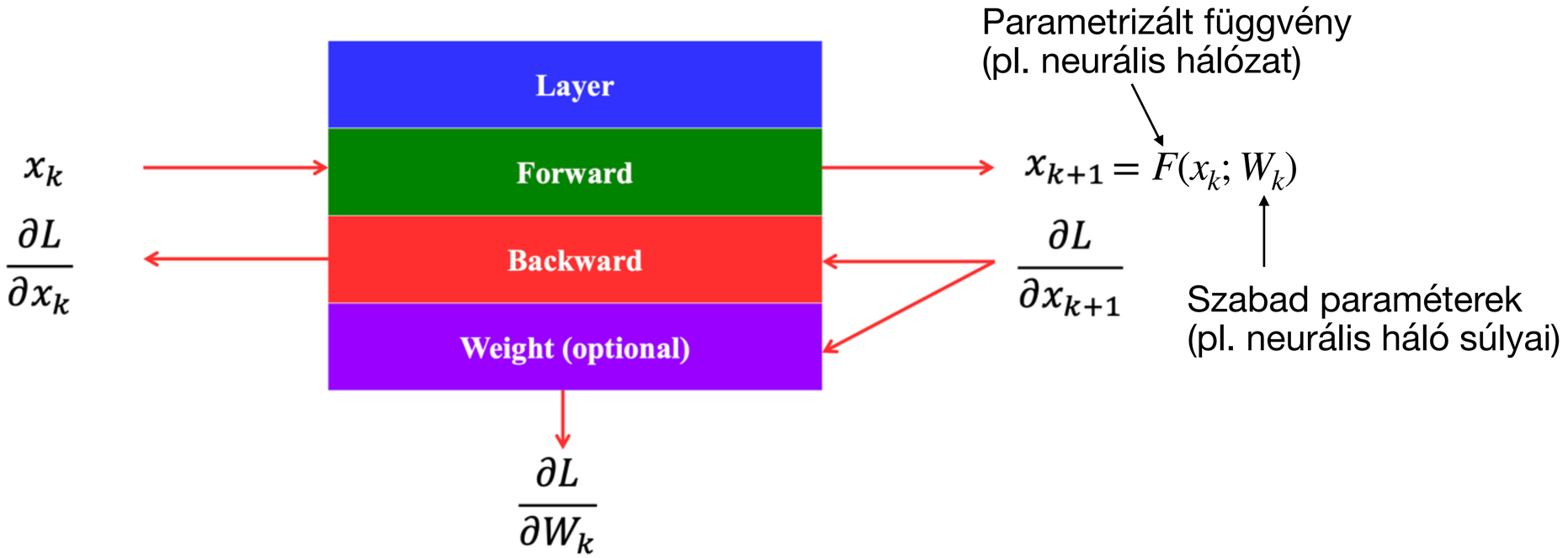
$$f(x) = e^x \rightarrow \frac{\partial f}{\partial x} = e^x \qquad f(x) = a + x \rightarrow \frac{\partial f}{\partial x} = 1$$

$$f(x) = ax \rightarrow \frac{\partial f}{\partial x} = a \qquad f(x) = \frac{1}{x} \rightarrow \frac{\partial f}{\partial x} = -\frac{1}{x^2}$$

Forrás: Szemenyei M.

Differenciális Programozás

Általános Interfész



Ilyen “rétegekből” *bármilyen* számítási procedúra felépíthető és differenciálható!
(Pl. minden neurális hálózat is)

Forrás: Szemenyei M.

Differenciális Programozás

Implementációk — C/C++

Forward Mode (using dual)

First-Order Derivatives

```
dual x, y, z;
dual u = f(x, y, z);
double ux = derivative(f, wrt(x), at(x, y, z));
double uy = derivative(f, wrt(y), at(x, y, z));
double uz = derivative(f, wrt(z), at(x, y, z));
```

Reverse Mode (using var)

First-Order Derivatives

```
var x, y, z;
var u = f(x, y, z);
auto [ux, uy, uz] = derivatives(u, wrt(x, y, z));
```

https://autodiff.github.io/

Általános célokra

https://github.com/patr-schm/TinyAD

Geometriai problémákra

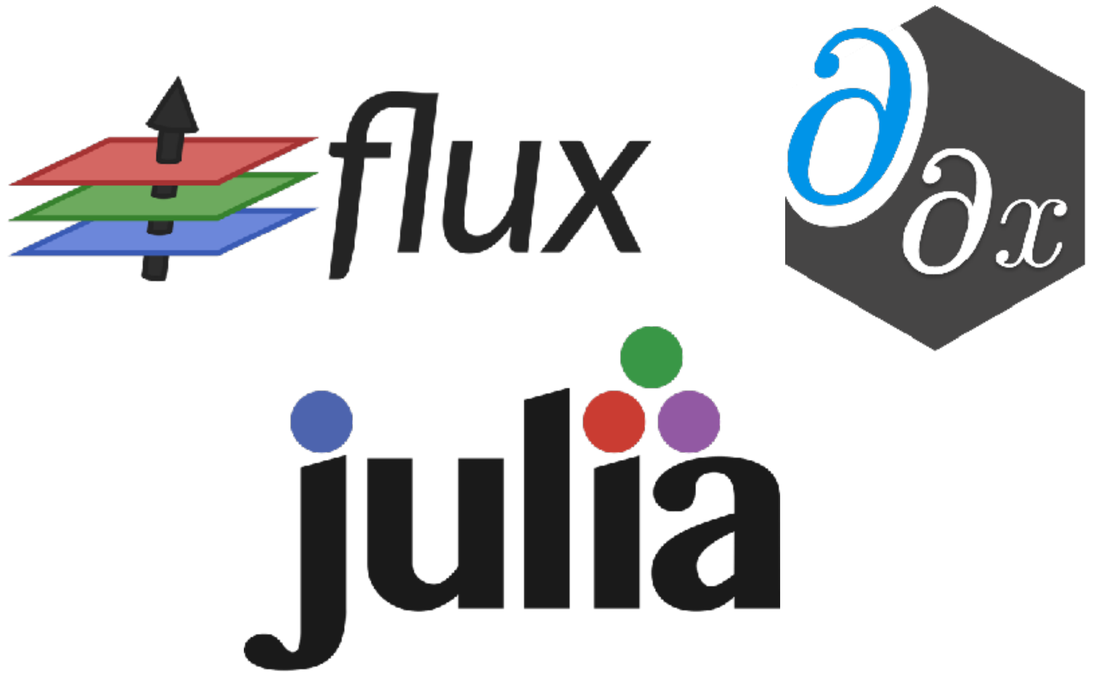
Differenciális Programozás

Implementációk — Python

Minden neurális / deep learning könyvtár központi eleme az auto-differenciálás!

Differenciális Programozás

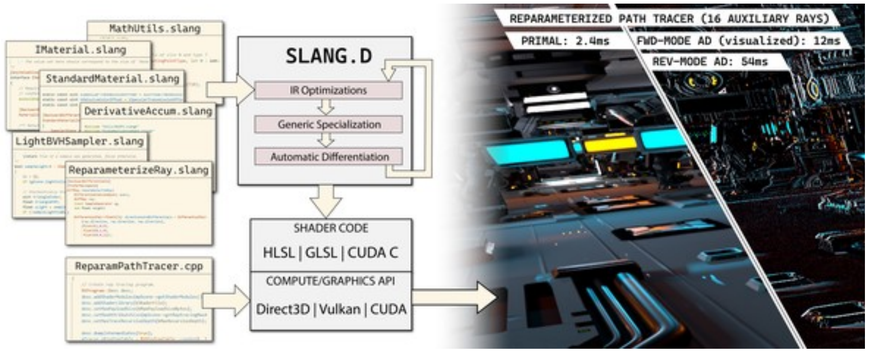
Implementációk — Julia



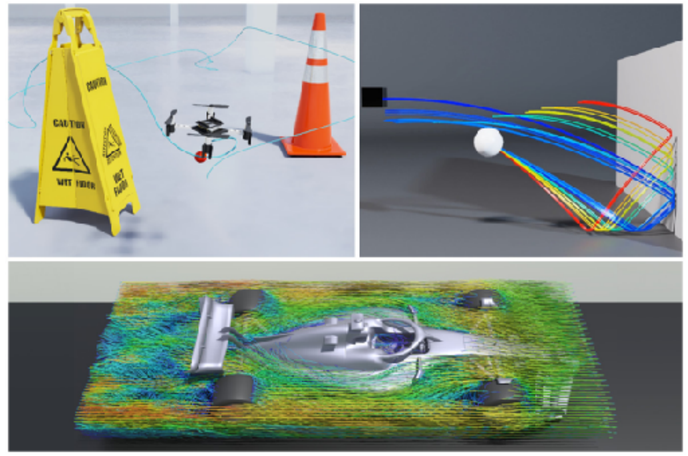
Léteznek modern, auto-differenciálásra tervezett programnyelvek is (pl. Julia)

Differenciális Programozás

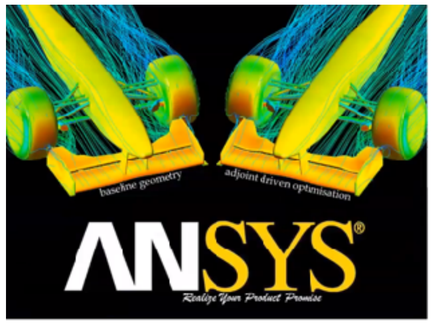
Implementációk — Egyéb Példák



Differenciálható Shadernyelv



NVIDIA Warp



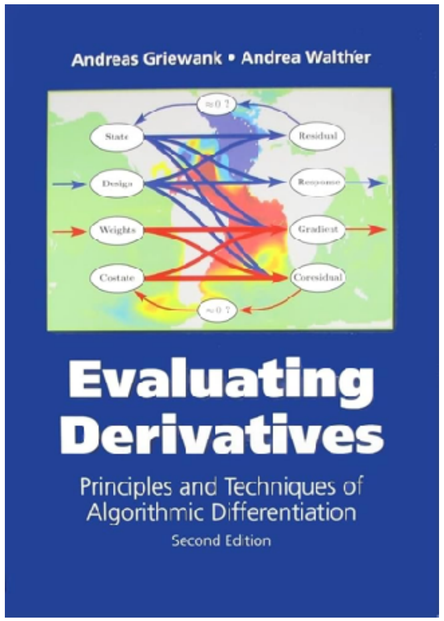
OpenFOAM

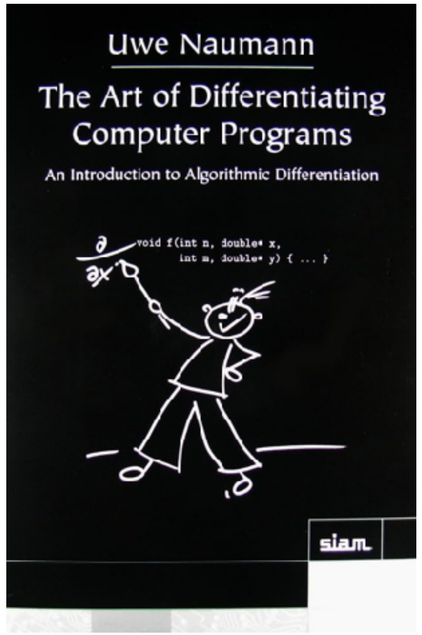
FENICS PROJECT

“Adjoint” Szimuláció

Differenciális Programozás

Könyvajánló





Klasszikus tankönyvek

The Elements of Differentiable Programming

Mathieu Blondel
Google DeepMind
mblondel@google.com

Vincent Roulet
Google DeepMind
vroulet@google.com

https://arxiv.org/abs/2403.14606

Egy modern jegyzet

Következő előadás:

Neurális Hálózatok

- Perceptron Hálók
- Konvolúciós Neurális Hálók
- Háló Architektúrák