Tomáš Oberhuber

Faculty of Nuclear Sciences and Physical Engineering
Czech Technical University in Prague

Výpočet derivací

- derivace mnoha ztrátových funkcí nebo funkcionálů lze snadno napočítat ručně
- v řadě případů může být ztrátová funkce příliš složitá na ruční derivování
- pak lze použít některou z následujících metod
  - konečné diference – *finite differencing*
  - automatické derivování – *automatic differentiation*
  - symbolické derivování – *symbolic differentiation*
    - například programy Mathematica, Maple, Macsyma nebo Ginac
    - v této přednášce se symbolickým derivováním zabývat nebudeme

Konečné diference

- derivace lze nahrazovat pomocí konečných diferencí
- příkladem je dopředná diference pro aproximaci první derivace

$$\frac{\partial f(\vec{x})}{\partial x_i} \approx \frac{f(\vec{x} + \epsilon \vec{e}_i) - f(\vec{x})}{\epsilon},$$

  pro $f : \mathbb{R}^n \to \mathbb{R}$, kde $\vec{e}_i$ je i-tý bazický vektor
- funkci f je tedy potřeba vyčíslit dvakrát
- otázkou je zde hlavně volba parametru ϵ
  - čím menší ϵ zvolíme, tím přesnější aproximaci dostaneme
  - když ϵ zvolíme příliš malé, můžeme narazit na přesnost počítačové aritmetiky

Připomeneme si definici množiny čísel s pohyblivou desetinnou čárkou s konečnou přesností:

Definition 1

Definujeme **množinu čísel s pohyblivou desetinou čárkou** jako

$$\mathbb{F}(\beta, t, L, U) \equiv \{0\} \cup \left\{ x \in \mathbb{R} \mid x = (-1)^s \beta^e \sum_{i=1}^{t} a_i \beta^{-i} \right\},$$

kde $\beta \in \mathbb{N}, \beta > 1$ určuje základ rozvoje $t \in \mathbb{N}, t > 0$ určuje počet významných cifer, a pro exponent $e \in \mathbb{N}$ platí $L \leq e \leq U$.

Definition 2
Číslo $u = \beta^{-t-1}$ se označuje jako **zaokrouhlovací jednotky** (*unit roundoff*). Někdy se také označuje jako **strojové epsilon** (*machine epsilon*).

Lemma 3
Pro libovolné reálné číslo $x \in [\beta^L, \beta^U]$ *platí*

$$fl(x) = x(1 + \delta) \text{ kde } |\delta| \leq u,$$

kde $fl(\cdot)$ *značí projekci čísla* x *na nejbližší číslo z množiny* $\mathbb{F}$.

Remark 4

Číslo u nám tedy vyjadřuje relativní chybu, které se dopouštíme při zaokrouhlování, tj. při projekci reálných čísel do množiny $\mathbb{F}$. Tato relativní chyba se vztahuje k velikosti čísla x.

- Pro jednoduchou přesnost je $u = 1.19 \cdot 10^{-7}$.
- Pro dvojitou přesnost je $u = 2.22 \cdot 10^{-16}$.

- Z Taylorova rozvoje dále dostáváme

$$f(\vec{x}+\vec{p}) = f(\vec{x}) + \nabla f(\vec{x})\vec{p} + \frac{1}{2}\vec{p}^T\nabla^2 f(\vec{x}+\xi\vec{p})\vec{p},$$

  kde $\xi \in (0,1)$.
- Pokud je $f \in C^2(\mathbb{R}^n)$, pak existuje C takové, že

$$\left\|\nabla^2 f(\vec{x})\right\| < C$$

  pro všechna $\vec{x} \in \mathbb{R}^n$.
- A potom platí

$$\left\|f(\vec{x}+\vec{p}) - f(\vec{x}) - \nabla f(\vec{x})\vec{p}\right\| \leq \frac{C}{2}\|p\|^2. \qquad (1)$$

- Volbou $\vec{p} = \epsilon \vec{e}_i$ dostáváme

$$\frac{\partial f(\vec{x})}{\partial x_i} \approx \frac{f(\vec{x} + \epsilon \vec{e}_i) - f(\vec{x})}{\epsilon} + \delta_\epsilon,$$

kde

$$|\delta_\epsilon| \leq \frac{C}{2} \left\| \epsilon \vec{e}_i \right\| = \frac{C\epsilon}{2},$$

- pro správnou volbu ϵ potřebujeme nyní vzít v úvahu zaokrouhlovací chyby.

- Nyní tedy budeme sledovat zaokrouhlovací chyby vzniklé při výpočtu konečné diference

$$\frac{\partial f(\vec{x})}{\partial x_i} = \frac{f(\vec{x} + \epsilon \vec{e}_i) - f(\vec{x})}{\epsilon} + \delta_\epsilon.$$

- Platí

$$fl(f(\vec{x})) = f(\vec{x})(1 + \delta) \Leftrightarrow fl(f(\vec{x})) - f(\vec{x}) = f(\vec{x})\delta,$$

kde $|\delta| \leq u$.

- A tedy

$$|fl(f(\vec{x})) - f(\vec{x})| \leq L_f|\delta| \leq L_f u$$

- podobně lze ukázat i

$$|fl(f(\vec{x} + \epsilon \vec{e}_i)) - f(\vec{x} + \epsilon \vec{e}_i)| \leq L_f|\delta| \leq L_f u.$$

- Celkem je tedy

$$\left| fl\left(\frac{fl(fl(f(\vec{x}+\epsilon\vec{e}_i))-f(\vec{x}))}{\epsilon}\right) - \frac{f(\vec{x}+\epsilon\vec{e}_i)-f(\vec{x})}{\epsilon}\right| \leq 2\frac{L_f}{\epsilon}U$$

- a pro chybu aproximace platí

$$\left|\frac{\partial f(\vec{x})}{\partial x_i} - fl\left(\frac{fl(fl(f(\vec{x}+\epsilon\vec{e}_i))-f(\vec{x}))}{\epsilon}\right)\right| \leq$$
$$\left|\frac{\partial f(\vec{x})}{\partial x_i} - \frac{f(\vec{x}+\epsilon\vec{e}_i)-f(\vec{x})}{\epsilon}\right| + \left|\frac{f(\vec{x}+\epsilon\vec{e}_i)-f(\vec{x})}{\epsilon} - fl\left(fl(fl(\frac{f(\vec{x}+\epsilon\vec{e}_i))-f(\vec{x}))}{\epsilon}\right)\right| \leq$$
$$\delta_\epsilon + 2\frac{uL_f}{\epsilon} \leq 2\frac{uL_f}{\epsilon} + \frac{C\epsilon}{2}.$$

- Dále lze ukázat, že

$$\arg\min_\epsilon\left(\frac{uL_f}{\epsilon} + \frac{C\epsilon}{2}\right) = \sqrt{\frac{4L_f u}{C}}.$$

- Volíme tedy

$$\epsilon \approx u^{\frac{1}{2}}.$$

- Podobně lze dokázat, že pro centrální diferenci tvaru

$$\frac{\partial f(\vec{x})}{\partial x_i} \approx \frac{f(\vec{x} + \epsilon \vec{e}_i) - f(\vec{x} - \epsilon \vec{e}_i)}{\epsilon} + O(\epsilon^2)$$

je vhodné volit

$$\epsilon \approx u^{\frac{1}{3}}.$$

Výpočet Jakobiánů

- Buď $\vec{f} : \mathbb{R}^n \to \mathbb{R}^m$, chceme napočítat Jakobián

$$\mathbb{J}_{\vec{f}}(\vec{x}) = \begin{pmatrix} \nabla f_1(\vec{x}) \\ \vdots \\ \nabla f_m(\vec{x}) \end{pmatrix} \in \mathbb{R}^{m,n}.$$

- Pomocí konečné diference

$$\frac{\vec{f}(\vec{x} + \epsilon\vec{e}_i) - \vec{f}(\vec{x})}{\epsilon} \approx \frac{\partial \vec{f}(\vec{x})}{\partial x_i} = \begin{pmatrix} \frac{\partial f_1(\vec{x})}{\partial x_i} \\ \vdots \\ \frac{\partial f_m(\vec{x})}{\partial x_i} \end{pmatrix}$$

získáme jeden sloupec celého Jakobiánu.

Výpočet Jakobiánů

- Jednou perturbací ve směru $\vec{e}_i$ lze tedy napočítat jeden sloupec.
- Celkem tedy musíme vyčíslit hodnoty

$$\vec{f}(\vec{x}), \vec{f}(\vec{x} + \epsilon\vec{e}_1), \ldots, \vec{f}(\vec{x} + \epsilon\vec{e}_n),$$

  tj. celkem $(n+1)\times$ vyčíslit funkci $\vec{f}$.
- Všechna vyčíslení jsou na sobě nezávislá a lze je případně provést paralelně.

Výpočet Jakobiánů

- V případě některých metod, jako *inexact Newton method*, se nepočítá inverze Jakobiánu, ale Jakobián se vždy aplikuje na určitý vektor $\vec{p}$.
- Pak lze použít přímo vztah

$$\mathbb{J}_{\vec{f}}(\vec{x})\vec{p} \approx \frac{\vec{f}(\vec{x} + \epsilon\vec{p}) - \vec{f}(\vec{x})}{\epsilon}.$$

- Funkci $\vec{f}$ pak vyčíslujeme pouze dvakrát a šetříme paměť, protože Jakobián nemusíme ukládat v paměti explicitně.

Výpočet řídkého Jakobiánů

- V některých úlohách může být Jakobián řídký.
- Například u takto definované funkce

$$\vec{f}(\vec{x}) = \begin{pmatrix} 2(x_2^3 - x_1^2) \\ 3(x_2^3 - x_1^2) + 2(x_3^3 - x_2^2) \\ \vdots \\ 3(x_i^3 - x_{i-1}^2) + 2(x_{i+1}^3 - x_i^2) \\ \vdots \\ 3(x_{n-1}^3 - x_{n-2}^2) + 2(x_n^3 - x_{n-1}^2) \\ 3(x_n^3 - x_{n-1}^2) \end{pmatrix}.$$

Výpočet řídkého Jakobiánů

- Jakobián pak má následující tridiagonální vzor nenulových prvků

$$\begin{pmatrix} * & * & & & & \\ * & * & * & & & \\ & * & * & * & & \\ & & * & * & * & \\ & & & * & * & * \\ & & & & * & * \end{pmatrix}.$$

Výpočet řídkého Jakobiánů

- Zvolme například $n = 6$, pak je

$$\vec{f}(\vec{x}) = \begin{pmatrix} -2x_1^2 & 2x_2^3 & & & & \\ -3x_1^2 & 3x_2^3 - 2x_2^2 & 2x_3^3 & & & \\ & -3x_2^2 & 3x_3^3 - 2x_3^2 & 2x_4^3 & & \\ & & -3x_3^2 & 3x_4^3 - 2x_4^2 & 2x_5^3 & \\ & & & -3x_4^2 & 3x_5^3 - 2x_5^2 & 2x_6^3 \\ & & & & -3x_5^2 & 3x_6^3 - 2x_6^2 \end{pmatrix}.$$

- Pro výpočet $\frac{\partial \vec{f}}{\partial x_1}$ musíme napočítat perturbaci $\vec{f}(\vec{x} + \epsilon \vec{e}_1)$.

Výpočet řídkého Jakobiánů

- Perturbace vektorem $\epsilon\vec{e}_1$ ovlivní pouze první dvě složky f_1 a f_2

$$\frac{\partial \vec{f}(\vec{x})}{\partial x_1} \approx \begin{pmatrix} \frac{f_1(\vec{x}+\epsilon\vec{e}_1)-f_1(\vec{x})}{\epsilon} \\ \frac{f_2(\vec{x}+\epsilon\vec{e}_1)-f_2(\vec{x})}{\epsilon} \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

- Perturbace vektorem $\epsilon\vec{e}_4$ zase ovlivní pouze složky f_3, f_4 s f_5

$$\frac{\partial \vec{f}(\vec{x})}{\partial x_4} \approx \begin{pmatrix} 0 \\ 0 \\ \frac{f_3(\vec{x}+\epsilon\vec{e}_4)-f_3(\vec{x})}{\epsilon} \\ \frac{f_4(\vec{x}+\epsilon\vec{e}_4)-f_4(\vec{x})}{\epsilon} \\ \frac{f_5(\vec{x}+\epsilon\vec{e}_4)-f_5(\vec{x})}{\epsilon} \\ 0 \end{pmatrix}.$$

Výpočet řídkého Jakobiánů

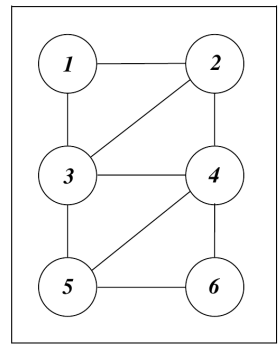
- Pokud tedy napočítáme perturbaci vektorem $\epsilon\vec{p} = \epsilon(\vec{e}_1 + \vec{e}_4)$, dostaneme vektor

$$\nabla\vec{f}\vec{p} = \begin{pmatrix} \frac{f_1(\vec{x}+\epsilon(\vec{e}_1+\vec{e}_4))-f_1(\vec{x})}{\epsilon} \\ \frac{f_2(\vec{x}+\epsilon(\vec{e}_1+\vec{e}_4))-f_2(\vec{x})}{\epsilon} \\ \frac{f_3(\vec{x}+\epsilon(\vec{e}_1+\vec{e}_4))-f_3(\vec{x})}{\epsilon} \\ \frac{f_4(\vec{x}+\epsilon(\vec{e}_1+\vec{e}_4))-f_4(\vec{x})}{\epsilon} \\ \frac{f_5(\vec{x}+\epsilon(\vec{e}_1+\vec{e}_4))-f_5(\vec{x})}{\epsilon} \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{f_1(\vec{x}+\epsilon(\vec{e}_1))-f_1(\vec{x})}{\epsilon} \\ \frac{f_2(\vec{x}+\epsilon(\vec{e}_1))-f_2(\vec{x})}{\epsilon} \\ \frac{f_3(\vec{x}+\epsilon(\vec{e}_4))-f_3(\vec{x})}{\epsilon} \\ \frac{f_4(\vec{x}+\epsilon(\vec{e}_4))-f_4(\vec{x})}{\epsilon} \\ \frac{f_5(\vec{x}+\epsilon(\vec{e}_4))-f_5(\vec{x})}{\epsilon} \\ 0 \end{pmatrix} \approx \begin{pmatrix} \frac{\partial f_1(\vec{x})}{\partial x_1} \\ \frac{\partial f_2(\vec{x})}{\partial x_1} \\ \frac{\partial f_3(\vec{x})}{\partial x_4} \\ \frac{\partial f_4(\vec{x})}{\partial x_4} \\ \frac{\partial f_5(\vec{x})}{\partial x_4} \\ 0 \end{pmatrix}.$$

- První dvě složky tedy odpovídají prvnímu sloupci Jakobiánu a složky 3, 4 a 5 odpovídají čtvrtému sloupci Jakobiánu.

Výpočet řídkého Jakobiánů

- Abychom zjistili, které perturbace lze vzájemně kombinovat, sestrojíme si incidenční graf.
- Pro funkci $\vec{f} : \mathbb{R}^n \to \mathbb{R}^m$ má tento graf n uzlů.
- Mezi uzly i a k vede hrana právě když existuje f_j, které závisí na x_i a x_k.
- V grafu pak napočítáme vrcholové obarvení.
- Každé barvě pak lze přiřadit jeden perturbační vektor.



J. Nocedal, S. J. Wright, Numerical Optimization, Springer, 2006.

Výpočet Hessiánu

- Mějme funkci $f \in C^2(\mathbb{R}^n; \mathbb{R})$ a chceme napočítat Hessián $\nabla^2 f(\vec{x}) \in \mathbb{R}^{n,n}$.
- Lze postupovat stejně jako při výpočtu Jakobiánu funkce $\nabla f(\vec{x})$.
- Hessián je však symetrický a díky numerickým chybám může být symetrie porušena.
- Někdy se proto vyplatí provést průměrování příslušných transponovaných prvků.
- Pokud potřebujeme výsledný Hessián pouze aplikovat na určitý vektor $\vec{p}$, pak si lze opět pomoci konečnou diferencí, tj.

$$\nabla^2 f(\vec{x})\vec{p} \approx \frac{\nabla f(\vec{x} + \epsilon\vec{p}) - \nabla f(\vec{x})}{\epsilon}.$$

Výpočet Hessiánu

- Hessián lze také počítat podle vztahu

$$\frac{\partial^2 f(\vec{x})}{\partial x_i \partial x_j} \approx \frac{f(\vec{x} + \epsilon\vec{e}_i + \epsilon\vec{e}_j) - f(\vec{x} + \epsilon\vec{e}_i) - f(\vec{x} + \epsilon\vec{e}_j) + f(\vec{x})}{\epsilon^2} + O(\epsilon).$$

- Tento postup vyžaduje celkem

$$\frac{n(n-1)}{2} + 2n + 1 = O(n^2)$$

vyčíslení funkce f.

Výpočet řídkého Hessiánu

- V řadě úloh může být Hessián řídký. Pak lze použít podobný postup jako pro výpočet řídkého Jakobiánu aplikovaný na $\nabla f(\vec{x})$.
- Obecně je potřeba sestrojit adjacenční graf o n uzlech.
- Uzly i a j, $i \neq j$ jsou spojené hranou pokud platí

$$\frac{\partial^2 f(\vec{x})}{\partial x_i \partial x_j} \neq 0.$$

- Následně hledáme obarvení, kde každé dva uzly spojené hranou mají různou barvu a každá cesta v grafu délky tři obsahuje alespoň tři barvy.

T. F. Coleman, J. J. Moré, *Estimation of sparse hessian matrices and graph coloring problems*, Mathematical Programming vol. 28, pp. 243–270, 1984.

Předpokládejme, že máme funkci $\vec{f} : \mathbb{R}^n \to \mathbb{R}^m$

$$\vec{f}(\vec{x}) = \vec{f}_1(\vec{f}_2(\vec{f}_3(\ldots \vec{f}_n(\vec{x})))) = (\vec{f}_1 \circ \vec{f}_2 \circ \ldots \vec{f}_n)(\vec{x}).$$

Pro gradient pak platí

$$\frac{\partial \vec{f}}{\partial \vec{x}}\Big|_{\vec{x}^0} = \frac{\partial \vec{f}_1}{\partial \vec{y}_1}\Big|_{y_1^0} \times \frac{\partial \vec{f}_2}{\partial \vec{y}_2}\Big|_{y_2^0} \times \ldots \frac{\partial \vec{f}_n}{\partial \vec{y}_n}\Big|_{y_n^0},$$

kde $\vec{y}_i^0$ značí vstup funkce $\vec{f}_i$ a platí

$$\begin{aligned} \vec{y}_n^0 &= \vec{x}^0, \\ \vec{y}_i^0 &= (\vec{f}_{i+1} \circ \ldots \circ \vec{f}_n)(\vec{x}^0), \\ \vec{y}_0^0 &= \vec{f}(\vec{x}^0). \end{aligned}$$

Je-li $\vec{f}_i : \mathbb{R}^{m_i} \to \mathbb{R}^{n_i}$, potom

$$\left.\frac{\partial \vec{f}_i}{\partial \vec{y}_i}\right|_{\vec{y}_i^0} = \mathbb{J}_i \in \mathbb{R}^{n_i, m_i}.$$

Výpočet gradientu pak spočívá v:

- výpočet Jakobiánů $\mathbb{J}_1, \ldots, \mathbb{J}_n$,
- výpočet součinu $\left.\frac{\partial \vec{f}}{\partial \vec{x}}\right|_{\vec{x}^0} = \mathbb{J}_1 \times \ldots \mathbb{J}_n$.

V závislosti na rozměrech Jakobiánů jsou pak možné dva způsoby pronásobení:

- dopředný - postupujme od $\mathbb{J}_n$ k $\mathbb{J}_1$

$$\left.\frac{\partial \vec{f}_i}{\partial \vec{y}_i}\right|_{\vec{y}_i^0} = \mathbb{J}_\mathbb{1} \times (\mathbb{J}_2 \times (\mathbb{J}_3 \times \ldots (\mathbb{J}_{n-1} \times \mathbb{J}_n)))$$

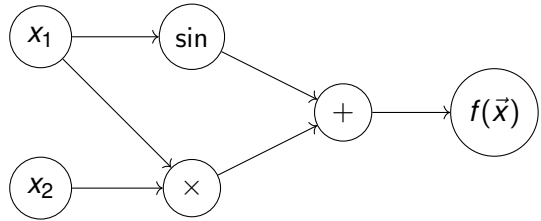
- zpětný - postupujme od $\mathbb{J}_1$ k $\mathbb{J}_n$

$$\left.\frac{\partial \vec{f}_i}{\partial \vec{y}_i}\right|_{\vec{y}_i^0} = ((((\mathbb{J}_1 \times \mathbb{J}_2) \times \mathbb{J}_3) \times \ldots) \times \mathbb{J}_{n-1}) \times \mathbb{J}_n$$

Automatické derivování

Mějme funkci

$$f(\vec{x}) = x_1 x_2 + \sin(x_1).$$



Tu lze popsat grafem vpravo a vyčíslit takto:

$$\vec{f}_1 : \mathbb{R}^2 \to \mathbb{R}^1 \quad \vec{f}_1(\vec{y}_1) = y_{1,1} + y_{1,2} \qquad \vec{y}_0 = (x_1 x_2 + \sin(x_1)),$$
$$\vec{f}_2 : \mathbb{R}^2 \to \mathbb{R}^2 \quad \vec{f}_2(\vec{y}_2) = (\sin(y_{2,1}), y_{2,1} y_{2,2})^T \quad \vec{y}_1 = (\sin(x_1), x_1 x_2)^T$$

a celkem

$$f(\vec{x}) = \vec{f}_1(\vec{f}_2(\vec{x}))$$

$$\begin{array}{lll} \vec{f}_1 : \mathbb{R}^2 \to \mathbb{R}^1 & \vec{f}_1(\vec{y}_1) = y_{1,1} + y_{1,2} & \vec{y}_0 = (x_1 x_2 + \sin(x_1)), \\ \vec{f}_2 : \mathbb{R}^2 \to \mathbb{R}^2 & \vec{f}_2(\vec{y}_2) = (\sin(y_{2,1}), y_{2,1} y_{2,2})^T & \vec{y}_1^0 = (\sin(x_1), x_1 x_2)^T \end{array}$$

$$\frac{\partial \vec{f}}{\partial \vec{x}} = \frac{\partial \vec{f}_1}{\partial \vec{y}_1}\Big|_{\vec{y}_1^0} \times \frac{\partial \vec{f}_2}{\partial \vec{y}_2}\Big|_{\vec{y}_2^0 \equiv \vec{x}} = \mathbb{J}_1 \times \mathbb{J}_2$$

$$\frac{\partial \vec{f}_1}{\partial \vec{y}_1}\Big|_{\vec{y}_1^0} = \begin{pmatrix} 1 & 1 \end{pmatrix} \quad \frac{\partial \vec{f}_2}{\partial \vec{y}_2}\Big|_{\vec{y}_2^0} = \begin{pmatrix} \cos(y_{2,1}) & 0 \\ y_{2,2} & y_{2,1} \end{pmatrix} = \begin{pmatrix} \cos(x_1) & 0 \\ x_2 & x_1 \end{pmatrix}$$

$$\frac{\partial \vec{f}}{\partial \vec{x}} = \mathbb{J}_1 \times \mathbb{J}_2 = \begin{pmatrix} 1 & 1 \end{pmatrix} \times \begin{pmatrix} \cos(x_1) & 0 \\ x_2 & x_1 \end{pmatrix} = \begin{pmatrix} \cos(x_1) + x_2 & x_1 \end{pmatrix}$$

Dopředný mód

V dopředném módu napočítáváme postupně $\mathbb{J}_n, \ldots \mathbb{J}_1$ a spolu s nimi i $\vec{y}_n, \ldots, \vec{y}_1$.

1: **procedure** FORWARDMODE($\vec{x}$)
2: $\quad \vec{y}_n^0 = \vec{x}, \frac{\partial \vec{y}_n}{\partial \vec{x}} = \mathbb{I}$
3: $\quad$ **for** $i = n \ldots, 1$ **do**
4: $\quad\quad \vec{y}_{i-1}^0 := \vec{f}_i(\vec{y}_i^0)$
5: $\quad\quad \mathbb{J}_i := \left.\frac{\partial \vec{f}_i}{\partial \vec{y}_i}\right|_{\vec{y}_i^0}$
6: $\quad\quad$ push forward the gradient $\left.\frac{\partial \vec{y}_{i-1}}{\partial \vec{x}}\right|_{\vec{x}} = \mathbb{J}_i \times \frac{\partial \vec{y}_i}{\partial \vec{x}}$
7: $\quad$ **end for**
8: $\quad$ return $(\vec{y}_0, \left.\frac{\partial \vec{y}_0}{\partial \vec{x}}\right|_{\vec{x}})$
9: **end procedure**

Všimneme si:

- Výstupem algoritmu je

$$\vec{y}_0 \equiv \vec{f}(\vec{x}) \quad \text{a} \quad \frac{\partial \vec{y}_0}{\partial \vec{x}}\Big|_{\vec{x}} \equiv \frac{\partial \vec{f}}{\partial \vec{x}}\Big|_{\vec{x}},$$

  tedy jak funkční hodnota, tak i gradient.
- Obojí jsme napočítali současně pouze jedním průchodem.

Při reálné implementaci se ale postupuje trochu jinak a pomocí tzv. **duálních čísel**.

Duální čísla

Jde o čísla, která v sobě udržují jak funkční hodnotu, tak i derivaci:

$$x = v + \dot{v}\epsilon$$

- Jde o koncept podobný komplexním číslům, ale místo imaginární jednotky i používáme ϵ.
- Je možné toto také chápat jako Taylorův rozvoj funkce x v bodě v s přesností prvního řádu.
- Při výpočtech potom vyšší mocniny ϵ zahazujeme, tj. $\epsilon^2 = 0$ místo $i^2 = -1$.

Platí následující vztahy:

$$\begin{aligned}
(v + \dot{v}\epsilon) + (u + \dot{u}\epsilon) &= (v + u) + (\dot{v} + \dot{u})\epsilon, \\
(v + \dot{v}\epsilon)(u + \dot{u}\epsilon) &= vu + (u\dot{v} + \dot{u}v)\epsilon + \dot{v}\dot{u}\epsilon^2 = vu + (u\dot{v} + \dot{u}v)\epsilon, \\
\frac{v + \dot{v}\epsilon}{u + \dot{u}\epsilon} &= \frac{v + \dot{v}\epsilon}{u + \dot{u}\epsilon}\frac{u - \dot{u}\epsilon}{u - \dot{u}\epsilon} = \frac{vu + (u\dot{v} + \dot{u}v)\epsilon}{u^2 - \dot{u}^2\epsilon^2} = \frac{v}{u} - \frac{(u\dot{v} + \dot{u}v)}{u^2}\epsilon
\end{aligned}$$

Např. pro derivaci předchozích výrazů podle v dosadíme

$$(v, \dot{v}) = (v, 1) \text{ a } (u, \dot{u}) = (u, 0) :$$

$$\begin{aligned}
(v + \dot{v}\epsilon) + (u + \dot{u}\epsilon) &= (v + u) + (\dot{v} + \dot{u})\epsilon = (v + u) + 1\epsilon, \\
(v + \dot{v}\epsilon)(u + \dot{u}\epsilon) &= vu + (u\dot{v} + \dot{u}v)\epsilon = vu + u\epsilon, \\
\frac{v + \dot{v}\epsilon}{u + \dot{u}\epsilon} &= \frac{v}{u} - \frac{(u\dot{v} + \dot{u}v)}{u^2}\epsilon = \frac{v}{u} - \frac{1}{u}\epsilon.
\end{aligned}$$

Podobně pro derivaci předchozích výrazů podle u dosadíme

$$(v, \dot{v}) = (v, 0) \text{ a } (u, \dot{u}) = (u, 1) :$$

$$\begin{aligned}
(v + \dot{v}\epsilon) + (u + \dot{u}\epsilon) &= (v + u) + (\dot{v} + \dot{u})\epsilon = (v + u) + 1\epsilon, \\
(v + \dot{v}\epsilon)(u + \dot{u}\epsilon) &= vu + (u\dot{v} + \dot{u}v)\epsilon = vu + v\epsilon, \\
\frac{v + \dot{v}\epsilon}{u + \dot{u}\epsilon} &= \frac{v}{u} - \frac{(u\dot{v} + \dot{u}v)}{u^2}\epsilon = \frac{v}{u} - \frac{v}{u^2}\epsilon.
\end{aligned}$$

Pro každou složku gradientu tedy musíme provést jeden průchod.

Pro obecnou funkci $f : \mathbb{R} \to \mathbb{R}$ platí:

$$f(v + \dot{v}\epsilon) = \sum_{i=0}^{\infty} \frac{f^{(i)}(v)}{i!} \dot{v}^i \epsilon^i = f(v) + f'(v)\dot{v}\epsilon$$

a stačí tedy vyčíslit funkci f pro (duální) hodnotu $(v + 1\epsilon)$ a koeficient u ϵ ve výsledku nám udává její derivaci.

Duální čísla

- Výhoda duálních čísel je v tom, že je lze snadno implementovat pouze s pomocí přetížení operátorů.
- Nevýhoda je potřeba extra výpočtu pro každou složku gradientu.
- Stále je ale možnost implementovat dopředný mód pomocí Jakobiánů.

- Dopředný mód může být implementovaný jako:
  - součást překladače,
  - formou precompileru,
  - pomocí přetížení operátorů v C++ – autodiff.

Zpětný mód

- Při zpětném módu nejprve počítáme Jakobián

$$\mathbb{J}_0 = \frac{\partial \vec{y}}{\partial \vec{y}_0} = \mathbb{I}.$$

- Dále napočítáváme

$$\mathbb{J}_1 = \frac{\partial \vec{f}_1}{\partial \vec{y}_1}\Big|_{\vec{y}_1^0}.$$

- K tomu ale potřebujeme znát hodnotu $\vec{y}_1^0$ a pro její výpočet i $\vec{y}_i^0$ pro $i = 2, \ldots n$.

Proto musíme nejprve provést tzv. **dopředný chod** nebo také **aktivaci**.

Zpětný mód

1: **procedure** FORWARDMODE($\vec{x}$)
2: $\mathbb{J}_0 := \frac{\partial \vec{f}(\vec{x})}{\partial \vec{y}_0} = \frac{\partial \vec{y}_0}{\partial \vec{y}_0} = \mathbb{I}$
3: **for** $i = n \ldots, 1$ **do** *Forward pass*
4: $\vec{y}^0_{i-1} := \vec{f}_i(\vec{y}^0_i)$
5: **end for**
6: **for** $i = 1 \ldots, n$ **do** *Backward pass*
7: $\mathbb{J}_i := \frac{\partial \vec{f}_i}{\partial \vec{y}_i}\Big|_{\vec{y}^0_i}$
8: pull back the gradient $\frac{\partial \vec{f}(\vec{x})}{\partial \vec{y}_i}\Big|_{\vec{y}^0_i} = \frac{\partial \vec{y}_0}{\partial \vec{y}_i}\Big|_{\vec{y}^0_i} = \frac{\partial \vec{y}_0}{\partial \vec{y}_{i-1}} \times \mathbb{J}_i$
9: **end for**
10: return $(\vec{y}_0, \frac{\partial \vec{y}_0}{\partial \vec{x}}\Big|_{\vec{x}})$
11: **end procedure**

Zpětný mód

- Výpočet nyní probíhá ve dvou průchodech.
- Navíc si musíme ukládat hodnoty $\vec{y}_i^0$, což může být paměťově náročné.

V praxi se zpětný mód implementuje dvěma možnými postupy:

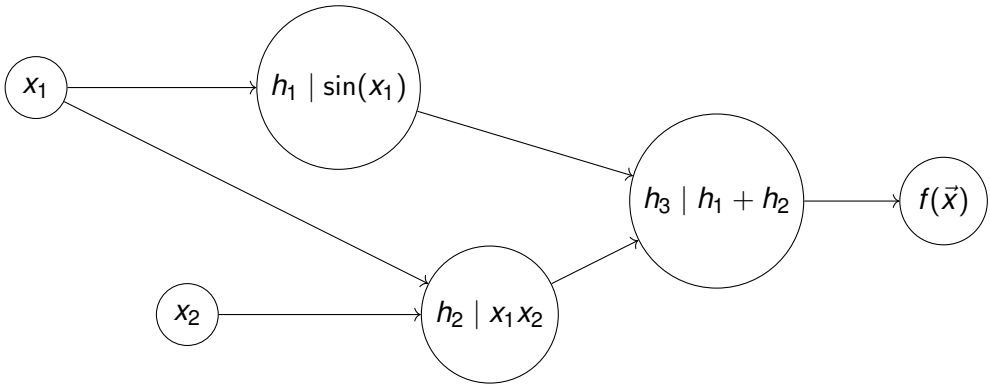
- grafový přístup,
- tracking base.

Zpětný mód - grafový přístup

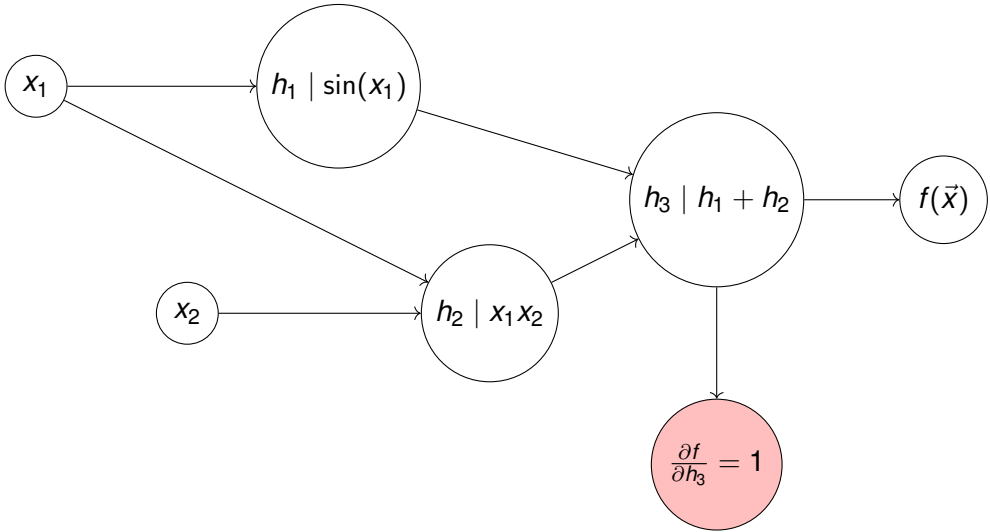
- V tomto případě předpokládáme znalost kompletního výpočetního grafu.
  - To je případ třeba neuronových sítí.
- Na základě tohoto grafu vytvoříme rozšířený graf, podle kterého se bude počítat gradient.

Vrátíme se k našemu příkladu.

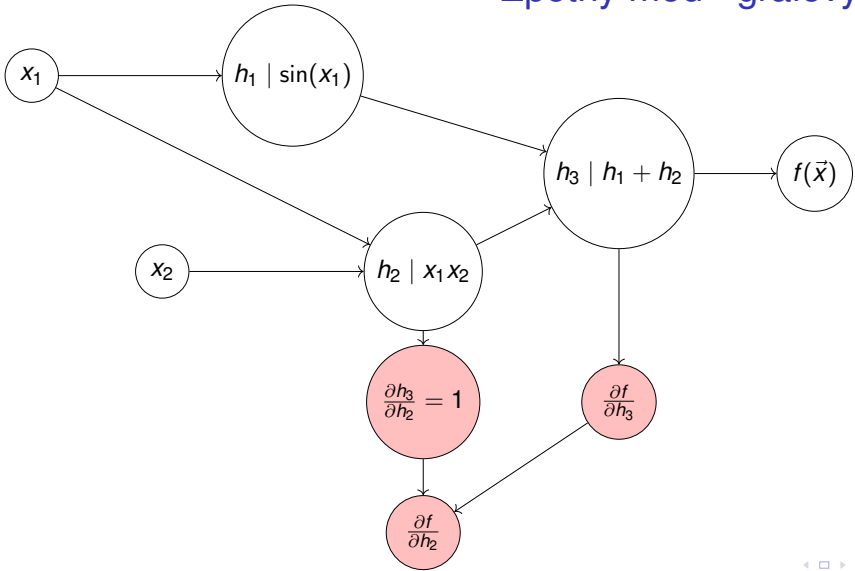
Zpětný mód - grafový přístup



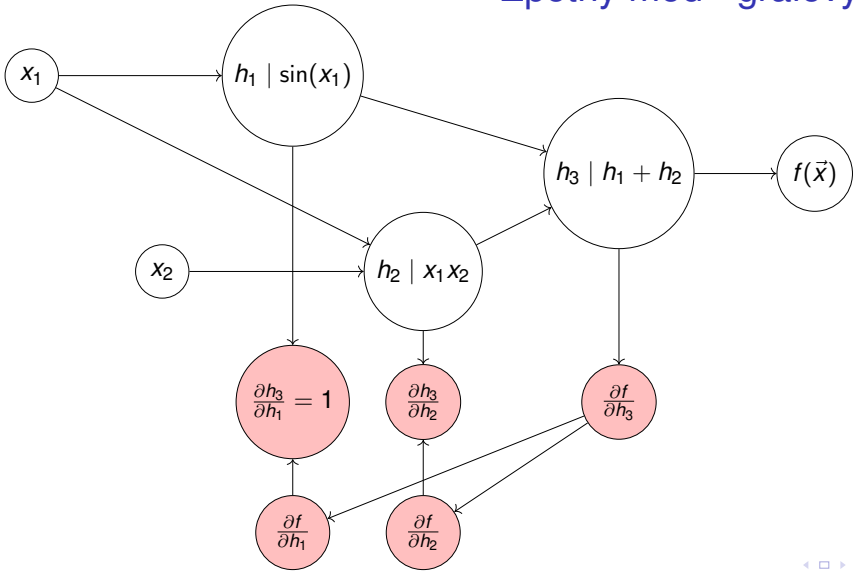
Zpětný mód - grafový přístup



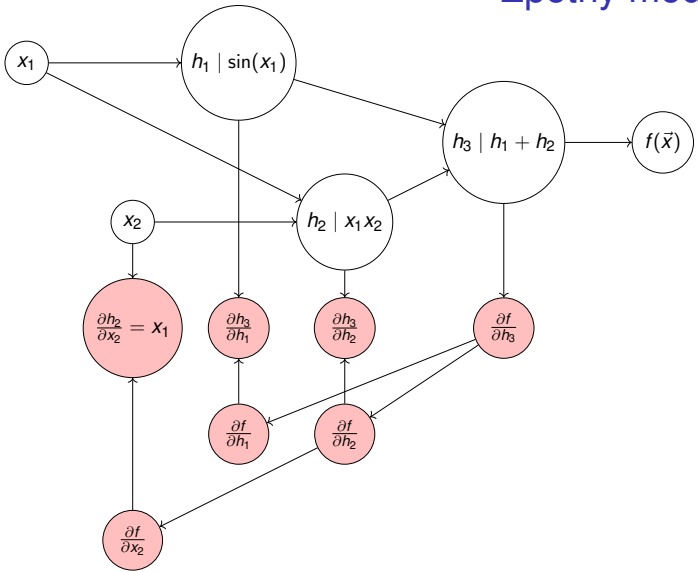
Zpětný mód - grafový přístup



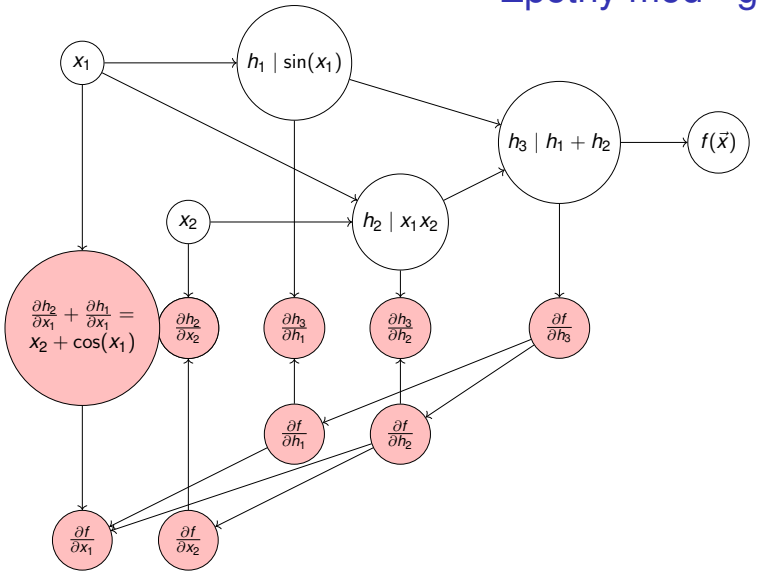
Zpětný mód - grafový přístup



Zpětný mód - grafový přístup



Zpětný mód - grafový přístup



Zpětný mód - grafový přístup

- Výsledkem zpětného chodu je kompletní gradient.
- Zpětný chod je proto **výhodnější pro počítání gradientů, které mají hodně složek**.
- Uložení výpočetního grafu v paměti může být paměťově náročné.
- Pomocné proměnné uložené v rozšířené části grafu se označují jako *adjungované (adjoint)*.
- Tento postup používají knihovny jako *Theano* a *TensorFlow*.
  - Operace jako `tf.mul(a,b)` nebo `tf.add(a,b)` neprovádí výpočet, ale vytváří výpočetní graf.
- Explicitní znalost umožňuje provést optimalizace v preprocesingu.
- Tvorbu rozšířeného grafu pro výpočet gradientu lze snadno zopakovat a získat tak i vyšší derivace.

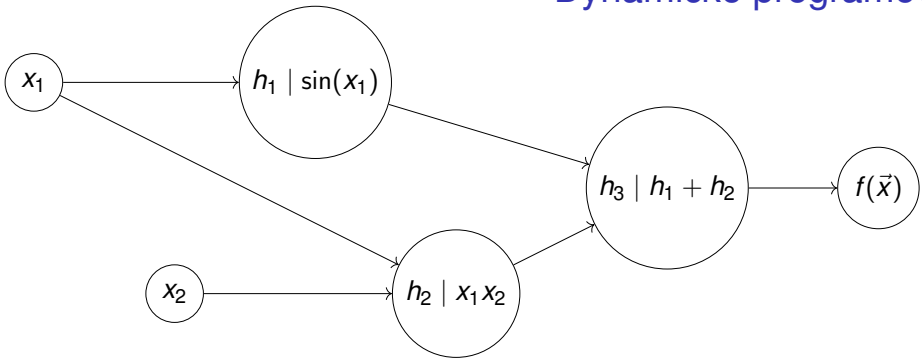
Zpětný mód - tracking based

- Pokud neznáme výpočetní graf předem, můžeme ho vytvořit dynamicky za běhu programu.
- Takto vytvořený graf se často nazývá jako páska - *tape*.
- Tento přístup využívá např. *PyTorch*.
- Implementuje se snadno pomocí přetížení operátorů.
- Implementace v C/C++ např. ADOL-C.

Zpětný mód

- Můžeme si všimnout, že výpočet se hodně podobá algoritmu zpětné propagace (*backpropagation*)

Dynamické programování



Vidíme, že

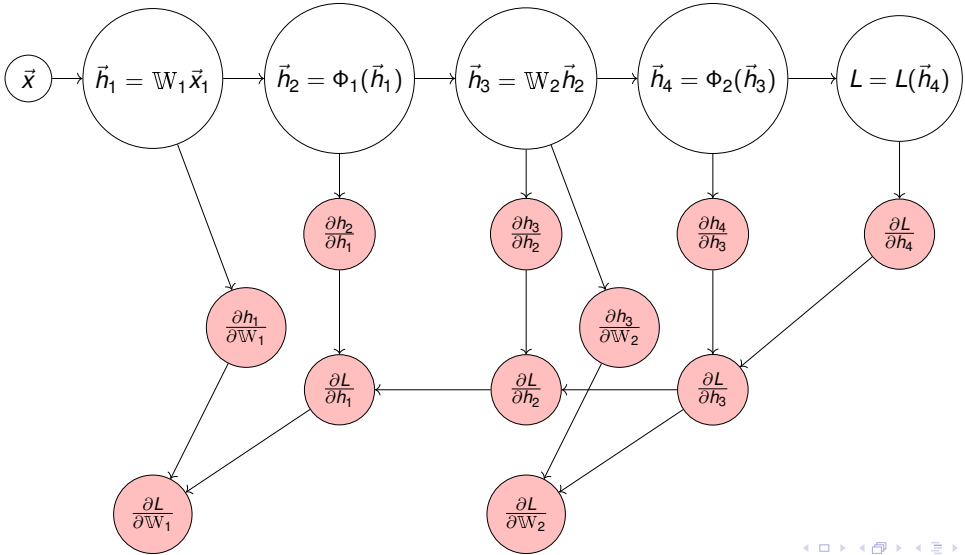
$$\frac{\partial f}{\partial x_1} = \frac{\partial f}{\partial h_3}\frac{\partial h_3}{\partial h_1}\frac{\partial h_1}{\partial x_1} + \frac{\partial f}{\partial h_3}\frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial x_1},$$

kde každý člen odpovídá jedné cestě v grafu z uzlu x_1 do uzlu $f(\vec{x})$.

Dynamické programování

- Zejména u neuronových sítí takových cest přibývá exponenciálně s rostoucí hloubkou sítě.
  - I díky tomu jsou neuronové sítě tak efektivní výpočetní model.
- Metody AD lze chápat i jako aplikaci **dynamického programování** na úlohu výpočtu gradientu.
- Např. u zpětného chodu napočítávám postupně derivace vůči skrytým proměnným a na jejich základě rozšiřujeme znalost gradientu vůči dalším skrytým proměnným, až dojdeme je vstupním proměnným.

Backpropagation pro NN



Kde platí např.

$$\left(\frac{\partial \vec{h}_1}{\partial \mathbb{W}_1}\right)_{ijk} = \left(\frac{\partial \mathbb{W}_1 \vec{x}_1}{\partial \mathbb{W}_1}\right)_{ijk} = \frac{\partial \sum_{l=1}^{n} w_{kl} x_l}{\partial w_{ij}} = \delta_{ik} x_j,$$

$$\left(\frac{\partial L}{\partial \mathbb{W}_1}\right)_{ij} = \sum_{k=1}^{n} \frac{\partial L}{\partial h_{1,k}} \frac{\partial h_{1,k}}{\partial w_{ij}} = \sum_{k=1}^{n} \frac{\partial L}{\partial h_{1,k}} \delta_{ik} x_j = \frac{\partial L}{\partial h_{1,i}} x_j.$$

Derivování ve funkcionálním programování

- Výpočetní grafy, které jsme viděli v předchozí části, ze dají vytvářet i z kódu, který vypočítává nějakou matematickou funkci.
- Ve výsledku tak můžeme derivovat i kód nebo funkce ve smyslu funkcionálního programování.

Zdroje

Další zdroje:

1. Scientific programming in Julia, T. Pevný, V. Šmídl, M. Zorek, N. Heim.
2. A. G. Baydin, B. A. Pearlmutter, A. A. Radul, J. M. Siskind, *Automatic differentiation in machine learning: a survey*, Journal of Marchine Learning Research, vol. 18, pp. 1–43, 2018.
3. www.autodiff.org
4. juliadiff.org