KMI/UMIN Umělá inteligence

Deep Learning

4. Optimalizace učení

Jan Konečný

Katedra informatiky, PřF UP v Olomouci

19. října 2026

Co nás dnes čeká

Proč prostý sestup nestačí

Momentum a Nesterov

Kolik dat na jeden krok

Adaptivní učicí krok

AdaGrad, RMSProp, Adam

Co si vybrat

Jeden řádek, který nestačí

Backpropagation dá gradient ztrátové funkce podle všech vah. Prostý gradientní sestup pak aktualizuje parametry proti jeho směru:

$$\boldsymbol{w}_t = \boldsymbol{w}_{t-1} - \eta \nabla E(\boldsymbol{w}_{t-1}).$$

U velkých sítí a zašuměných gradientů je ale takto provedený sestup buď příliš pomalý, nebo se rozkmitá. Dnešní přednáška ukazuje, čím se ten předpis v praxi doplňuje:

- setrvačností,
- volbou počtu vzorů, z nichž se gradient odhaduje,
- škálováním kroku pro každou váhu zvlášť.

Skripta: kap. 4 *Optimalizace učení*

Studijní priorita

Jádro

Mini-batch SGD, momentum, AdamW a vztah mezi optimalizátorem a plánem učicího kroku.

Kontext

Metody Silvy a Almeidy, delta-bar-delta a Rprop se dnes v hlubokém učení nepoužívají. Jsou tu proto, že je na nich dobře vidět, jak se k dnešním adaptivním krokům došlo.

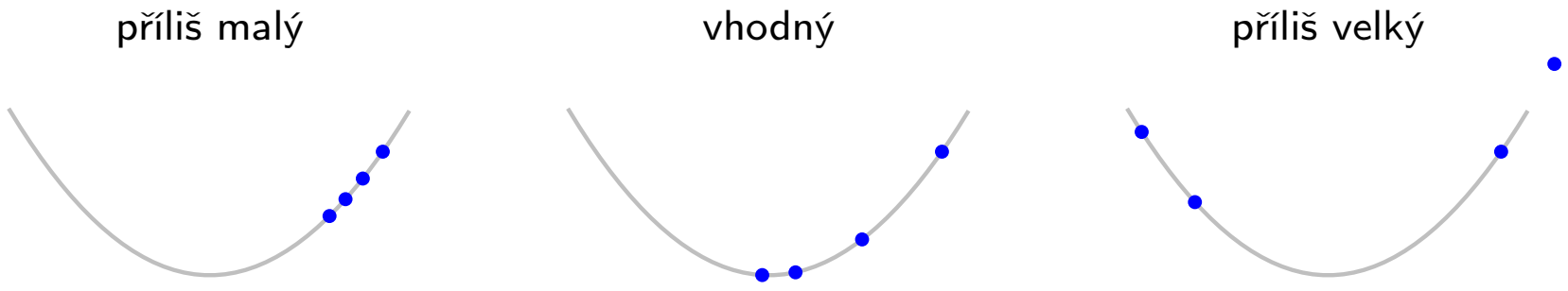
Plány učicího kroku (warmup, kosinový pokles) patří k praktickému trénování a probereme je v šesté přednášce.

Pět potíží

- **Pomalá konvergence.** Učení vyžaduje mnoho epoch; u velkých dat je to neúnosné.
- **Citlivost na η.** Malé η postupuje pomalu, velké přeskakuje minimum nebo diverguje. Vhodná hodnota se navíc v průběhu učení mění.
- **Oscilace v úzkých údolích.** Gradient míří napříč údolím, ne podél něj.
- **Nekonvexita.** Sedlové body, ploché oblasti, mnoho téměř ekvivalentních minim.
- **Plošiny a sedlové body.** Gradient blízký nule zastaví postup; ve vysokých dimenzích jsou takové oblasti časté.

Dlouho se soudilo, že hlavní hrozbou jsou špatná lokální minima. Hlubokou síť však zdržují spíše plochá místa a nepříznivé zakřivení.

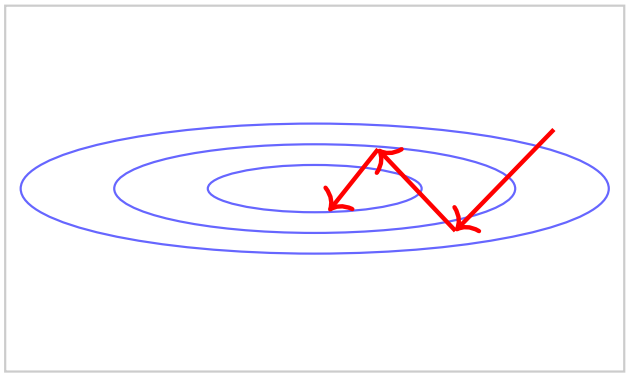
Vliv velikosti kroku



- Příliš malý krok: postup je stabilní, ale trvá neúnosně dlouho.
- Vhodný krok: chyba klesá rychle a monotónně.
- Příliš velký krok: iterace přeskakuje minimum a může divergovat.

Vhodná hodnota navíc není konstantní — na začátku učení bývá jiná než v jeho závěru.

Oscilace v úzkém údolí



trajektorie se odráží ode stěn

Příklad 1 (Kvadratická ztráta)

Pro $E(w_1, w_2) = \frac{1}{2}(100w_1^2 + w_2^2)$ a bod $(1, 1)$ je gradient $(100, 1)$. Krok $\eta = 0{,}02$ dá nový bod

$$(1, 1) - 0{,}02\,(100, 1) = (-1,\ 0{,}98).$$

V prudkém směru jsme přeskočili na druhou stranu údolí, v plochém se téměř nepohnuli.

Menší krok oscilaci omezí, ale ještě více zpomalí postup podél údolí. Potřebné je rozlišit *směr*, ve kterém je zakřivení velké, od směru, kde je malé.

Podmíněnost úlohy

U kvadratické ztráty s Hessiánem H rozhoduje o rychlosti sestupu číslo podmíněnosti

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

poměr největší a nejmenší křivosti. Kruhové vrstevnice mají $\kappa = 1$, úzké protáhlé údolí $\kappa \gg 1$.
Při nejlepší volbě kroku klesá chyba gradientního sestupu faktorem $(\kappa - 1)/(\kappa + 1)$, zatímco s momentem $(\sqrt{\kappa} - 1)/(\sqrt{\kappa} + 1)$.

Příklad 2 (Co to dělá pro $\kappa = 100$)

Prostý sestup zmenší chybu v každém kroku faktorem 0,980, momentum faktorem 0,818. Na desetinásobné zmenšení chyby tedy potřebuje první zhruba 115 kroků, druhý asi jedenáct.

Momentum tedy nahrazuje κ za $\sqrt{\kappa}$. Adaptivní metody útočí na tutéž veličinu jinak: snaží se κ zmenšit přeškálováním souřadnic.

Setrvačnost

Zavedeme vektor rychlosti a v každé iteraci počítáme

$$\boldsymbol{v}_t = \alpha\, \boldsymbol{v}_{t-1} - \eta\, \nabla E(\boldsymbol{w}_{t-1}), \qquad \boldsymbol{w}_t = \boldsymbol{w}_{t-1} + \boldsymbol{v}_t,$$

kde $\alpha \in [0, 1)$, typicky 0,9.

V úzkém údolí se složky gradientu, které mezi iteracemi mění znaménko, částečně vyruší; složky mířící podél údolí se naopak sčítají.

Příklad 3 (Jeden krok)

Pro $v_{t-1} = 0{,}3$, $g_t = 0{,}2$, $\alpha = 0{,}9$ a $\eta = 0{,}1$ vyjde $v_t = 0{,}27 - 0{,}02 = 0{,}25$. Krok pokračuje v dřívějším směru, ale nový gradient jej brzdí.

Demonstrace: momentum na ploché úloze

$E(w) = \frac{1}{2}w^2$, $w_0 = 1$, $\eta = 0{,}1$, $\alpha = 0{,}9$:

t	0	1	2	3	4	5
prostý sestup	1,000	0,900	0,810	0,729	0,656	0,590
s momentem	1,000	0,900	0,720	0,486	0,227	−0,029

První krok je shodný, protože rychlost začíná v nule. Od druhého se příspěvky sčítají a postup se zrychluje: po pěti krocích je prostý sestup na 0,59, zatímco momentum už minimum lehce přestřelilo.

Obojí je vidět na jednom příkladu: setrvačnost v plochém směru urychluje, a zároveň je příčinou přestřelení. Nesterovova varianta právě proto vyhodnocuje gradient až v předpovězeném bodě.

Co momentum přináší a co stojí

Výhody

- rychlejší postup v dlouhých údolích,
- stabilnější trénování i při větším η.

Nevýhody

- dva parametry k ladění (η, α),
- nahromaděná rychlost může minimum přestřelit.

Nesterovova modifikace (NAG)

Gradient se vyhodnotí až v bodě, kam setrvačnost parametry posune:

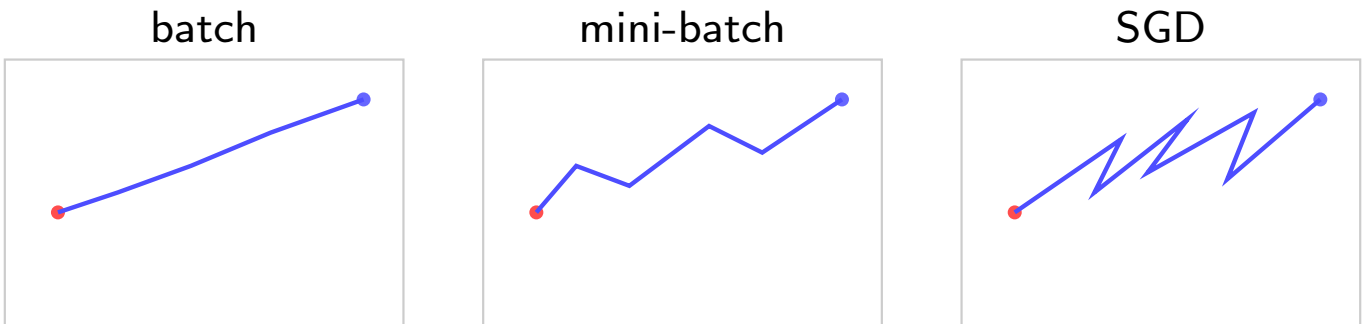
$$\boldsymbol{v}_t = \alpha \boldsymbol{v}_{t-1} - \eta \nabla E(\boldsymbol{w}_{t-1} + \alpha \boldsymbol{v}_{t-1}).$$

Pokud nás momentum žene přes minimum, gradient v předpovězeném bodě už „vidí“ protisvah a začne brzdit dřív.

Y. Nesterov: A Method of Solving a Convex Programming Problem with Convergence Rate $O(1/k^2)$. *Doklady AN SSSR* 269(3) (1983).

Tři strategie

- **Batch:** gradient z celého datasetu. Přesný odhad, drahý krok; u velkých dat se dataset nemusí vejít do paměti.
- **SGD (online):** gradient z jediného náhodného vzoru. Levné kroky, ale velký šum; kolem minima se algoritmus spíše potlouká.
- **Mini-batch:** gradient z malé náhodné dávky (desítky až stovky vzorů). Menší rozptyl než online, dávku lze paralelizovat. Běžný způsob trénování hlubokých sítí.



Šum gradientu není jen na obtíž: může pomoci opustit mělké lokální minimum.

Vzory se nemusí vybírat rovnoměrně

Výběr lze zvýhodnit podle aktuální chyby, aby síť častěji viděla příklady, které dosud nezvládá.

Kde je háček

Vysoká chyba může znamenat i šum nebo chybnou anotaci. Bez omezení by se trénování soustředilo právě na tyto případy a naučilo se je nazpaměť.

Praktické kompromisy:

- omezit poměr, jakým se obtížné vzory zvýhodňují,
- vyvažovat třídy jen v dávce, nikoli v metrikách,
- obtížné vzory oddělit a podívat se na ně ručně.

Jeden krok pro všechny směry je kompromis

Zakřivení ztrátové funkce je v různých směrech různé. Jediné globální η proto vede v plochém směru k pomalému postupu a ve strmém k oscilacím.

Společný princip adaptivních metod

- každá váha dostane *vlastní* učicí krok,
- ten se průběžně upravuje podle *historie gradientů* dané váhy,
- výsledkem je rychlejší a stabilnější konvergence bez ručního ladění kroku pro každý směr.

Ideálním, ale nedostupným řešením by bylo použít druhé derivace.

Sedlové body, ne špatná minima

V kritickém bodě rozhodují znaménka vlastních čísel Hessiánu. Aby šlo o lokální minimum, musí být *všechna* kladná — v milionu dimenzí je to mimořádně nepravděpodobná shoda.

Typický kritický bod je proto sedlo: v některých směrech se klesá, v jiných stoupá. Gradient je v jeho okolí malý, takže učení zpomalí, ale nezastaví se natrvalo.

Proč to není katastrofa

Ze sedla vede cesta ven; stačí najít směr se zápornou křivostí. Šum stochastického gradientu k tomu obvykle stačí sám, což je jeden z důvodů, proč mini-batch učení funguje lépe, než by se dalo čekat.

Empiricky navíc mívají různá minima hlubokých sítí podobnou hodnotu ztráty. Otázka „našli jsme globální minimum?“ je proto méně zajímavá než otázka, jak rychle a do jak dobře generalizujícího místa se dostaneme.

Y. N. Dauphin a kol.: Identifying and Attacking the Saddle Point Problem in High-Dimensional Non-Convex Optimization. NIPS (2014).

Kontext: Newtonova metoda

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - H(\boldsymbol{w}_k)^{-1} \nabla E(\boldsymbol{w}_k)$$

Ve směrech s velkou křivostí volí menší krok, v plochých větší; v okolí minima konverguje kvadraticky.

Proč se v hlubokém učení nepoužívá

Hessián vyžaduje $\mathcal{O}(n^2)$ paměti a jeho inverze až $\mathcal{O}(n^3)$ času, kde n je počet vah. Pro miliony parametrů je to nepraktické. Navíc mimo okolí minima nemusí být pozitivně definitní.

Princip však motivuje kvazi-newtonovské metody (BFGS, L-BFGS) i adaptivní algoritmy: ty nahrazují úplnou informaci o křivosti levným *diagonálním* škálováním odhadnutým z historie gradientů.

Historická linie: znaménko gradientu

- **Silva a Almeida (1990):** každá váha má vlastní krok η_i; stabilní znaménko gradientu jej násobí faktorem $u > 1$, změna znaménka faktorem $d < 1$.
- **Delta-bar-delta (1988):** totéž, ale rozhoduje se podle *vyhlazeného* gradientu (exponenciální průměr), aby krok nekmital se šumem.
- **Rprop (1993):** velikost gradientu se úplně ignoruje, používá se jen znaménko; při změně znaménka se navíc provede *rollback* poslední aktualizace.

Znaménko součinu dvou po sobě jdoucích gradientů je nejlevnější dostupná informace o zakřivení: nevyžaduje druhé derivace ani inverzi matice.

F. M. Silva, L. B. Almeida: Speeding up Backpropagation (1990). — R. A. Jacobs: Increased Rates of Convergence Through Learning Rate Adaptation. *Neural Networks* 1(4) (1988). — M. Riedmiller, H. Braun: RPROP. ICNN (1993).

Delta-bar-delta podrobněji

Vyhlazený gradient je exponenciální klouzavý průměr

$$\bar{g}_i^{(k)} = (1-\theta)\, g_i^{(k)} + \theta\, \bar{g}_i^{(k-1)}.$$

Učicí krok se pak adaptuje podle souhlasu aktuálního gradientu s tímto dlouhodobým trendem:

- shoda znamének $\Rightarrow$ *aditivní* přírůstek $\gamma_i \leftarrow \gamma_i + u$,
- neshoda $\Rightarrow$ *multiplikativní* zmenšení $\gamma_i \leftarrow d\,\gamma_i$, $0 < d < 1$.

Asymetrie mezi pomalým růstem a rychlým poklesem omezuje rozkmitání. Cenou je další hyperparametr θ, jehož vhodná hodnota silně závisí na úloze.

Má-li chyba v okolí kvadratickou povahu, bývá optimální $\theta = 0$ — a delta-bar-delta pak přechází v algoritmus Silvy a Almeidy.

Rprop: silná i slabá stránka téhož nápadu

Krok Δ_i se adaptuje podle součinu $g_i^{(k)} g_i^{(k-1)}$:

- > 0: $\Delta_i \leftarrow \min(\Delta_i \eta^+, \Delta_{\max})$,
- < 0: $\Delta_i \leftarrow \max(\Delta_i \eta^-, \Delta_{\min})$ a rollback,
- $= 0$: krok zůstává.

Typicky $\eta^+ = 1{,}2$, $\eta^- = 0{,}5$.

Váha se posouvá *jen podle znaménka* gradientu; velikost posunu určuje Δ_i. Rprop proto není citlivý na měřítko nenulového gradientu.

Táž vlastnost je i slabinou

Vyjde-li gradient numericky nulový nebo mu šum převrátí znaménko, nemá Rprop z čeho směr rekonstruovat. Ve stochastickém režimu s malými dávkami proto selhává — a právě ten je v hlubokém učení běžný.

AdaGrad

Akumulátor čtverců gradientů a krok nepřímo úměrný jeho odmocnině:

$$s_{t,i} = s_{t-1,i} + g_{t,i}^2, \qquad w_{t+1,i} = w_{t,i} - \eta \frac{g_{t,i}}{\sqrt{s_{t,i} + \varepsilon}}.$$

- Zřídka aktivní souřadnice akumuluje pomalu, takže jí zůstává *velký* krok — výhodné u řídkých dat.
- Jde o levnou diagonální náhradu inverze Hessiánu.

Krok jen klesá

Akumulátor pouze roste, takže efektivní krok $\eta/\sqrt{s_t}$ monotónně klesá. U nestacionárních úloh — a hluboké sítě takové jsou — může učení předčasně ustat.

J. Duchi, E. Hazan, Y. Singer: Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *JMLR* 12 (2011).

AdaGrad na řídkých datech

Příklad 4 (Dvě souřadnice s různou četností)

Souřadnice A dostává v každém kroku gradient velikosti 1, souřadnice B jen v každém desátém kroku, ale velikosti 3. Po sto krocích je

$$s_A = 100, \qquad s_B = 10 \cdot 9 = 90,$$

takže efektivní kroky jsou $\eta/10$ a $\eta/\sqrt{90} \doteq \eta/9{,}5$. Vzácná, ale silná souřadnice si udrží srovnatelný krok, přestože byla aktivní desetkrát méně často.

Odtud plyne obvyklé doporučení: AdaGrad se hodí tam, kde jsou příznaky řídké — zpracování textu, doporučovací systémy, one-hot kódované atributy.

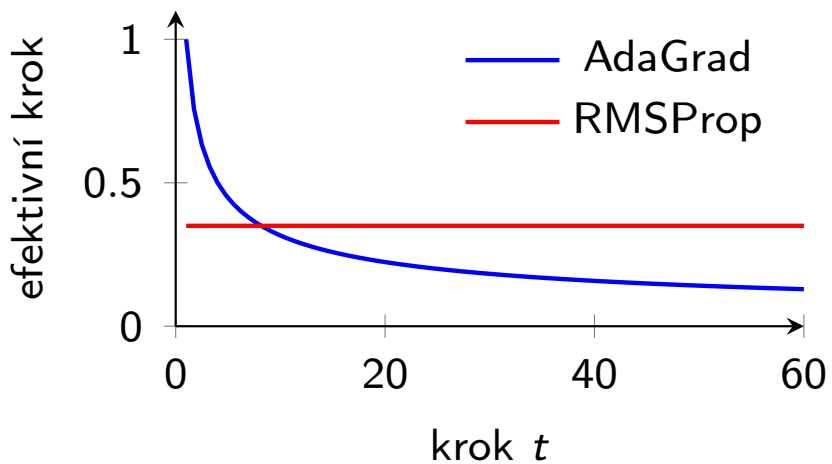
Pro husté hluboké sítě se ale monotónní pokles kroku stává překážkou.

RMSProp

Místo kumulace exponenciální klouzavý průměr:

$$s_t = \rho\, s_{t-1} + (1-\rho)\, g_t^2, \qquad w_{t+1} = w_t - \eta\, \frac{g_t}{\sqrt{s_t} + \varepsilon},$$

typicky $\rho = 0{,}9$.



Při přibližně konstantním gradientu klesá efektivní krok AdaGradu jako $1/\sqrt{t}$, zatímco RMSProp se ustálí. Váha starých gradientů exponenciálně klesá, takže učení nemusí vyhasnout.

T. Tieleman, G. Hinton: Lecture 6.5 — RMSProp. COURSERA (2012), nepublikovaná přednáška.

Adam

Spojuje setrvačnost prvního momentu s adaptací podle druhého:

$$\boldsymbol{m}_t = \beta_1 \boldsymbol{m}_{t-1} + (1-\beta_1)\boldsymbol{g}_t, \qquad \boldsymbol{v}_t = \beta_2 \boldsymbol{v}_{t-1} + (1-\beta_2)\boldsymbol{g}_t^2,$$

$$\hat{\boldsymbol{m}}_t = \frac{\boldsymbol{m}_t}{1-\beta_1^t}, \qquad \hat{\boldsymbol{v}}_t = \frac{\boldsymbol{v}_t}{1-\beta_2^t}, \qquad \boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta \frac{\hat{\boldsymbol{m}}_t}{\sqrt{\hat{\boldsymbol{v}}_t} + \varepsilon}.$$

Doporučené výchozí hodnoty: $\eta = 10^{-3}$, $\beta_1 = 0{,}9$, $\beta_2 = 0{,}999$, $\varepsilon = 10^{-8}$.

D. P. Kingma, J. Ba: Adam: A Method for Stochastic Optimization. ICLR (2015).

Proč Adam koriguje počáteční bias

Momenty se inicializují nulami, takže jsou na začátku systematicky podhodnocené. Dělení faktorem $1 - \beta^t$ to napravuje.

Příklad 5 (První krok)

Pro $g_1 = 2$, $\beta_1 = 0{,}9$ a $\beta_2 = 0{,}999$ vyjde

$$m_1 = 0{,}2, \qquad v_1 = 0{,}004.$$

Po korekci ovšem

$$\hat{m}_1 = \frac{0{,}2}{1 - 0{,}9} = 2, \qquad \hat{v}_1 = \frac{0{,}004}{1 - 0{,}999} = 4.$$

Bez korekce by nuly z inicializace uměle zmenšily oba odhady.

S rostoucím t konverguje faktor k jedničce a korekce sama vymizí.

Adam: tři složky, tři role

- $\boldsymbol{m}_t$ hraje roli *momenta*: vyhlazuje krátkodobý šum, takže směr kroku je stabilnější.
- $\boldsymbol{v}_t$ je *adaptace podle měřítka*: odhaduje typickou velikost gradientu v každé souřadnici.
- *Bias-korekce* napravuje podhodnocení obou odhadů na začátku učení.

Adam tedy kombinuje „směr“ a „měřítko“. S výchozími hodnotami hyperparametrů obvykle funguje bez pracného ladění; proto patří k nejpoužívanějším algoritmům hlubokého učení.

Kdy naopak zklame

Původní algoritmus může na některých konvexních konstrukcích selhat; varianta AMSGrad upravuje odhad druhého momentu tak, aby obnovila konvergenční záruku. A u některých úloh generalizuje SGD s momentem lépe, přestože konverguje pomaleji.

AdamW a oddělený weight decay

U SGD je přičtení L2 penalizace ke ztrátě ekvivalentní násobnému zmenšení vah. U adaptivní metody však gradient penalizace projde souřadnicovým škálováním, takže ekvivalence neplatí.

AdamW proto provádí adaptivní krok a zmenšení vah odděleně:

$$\boldsymbol{w}_{t+1} = (1 - \eta\lambda)\, \boldsymbol{w}_t - \eta\, \frac{\hat{\boldsymbol{m}}_t}{\sqrt{\hat{\boldsymbol{v}}_t} + \varepsilon}.$$

- Biasy a parametry normalizačních vrstev se z decay obvykle vyjímají.
- Táž hodnota λ znamená v SGD a v AdamW jinou sílu regularizace; čísla z různých optimalizátorů proto nelze srovnávat.

I. Loshchilov, F. Hutter: Decoupled Weight Decay Regularization. ICLR (2019).

Srovnání tří moderních metod

	AdaGrad	RMSProp	Adam
adaptace kroku	kumulace g^2 přes celou historii (krok monotónně klesá)	exponenciální průměr g^2	průměr g i g^2 s bias-korekcí
výhody	řídká data, konvexní úlohy	stabilní i po dlouhém trénování	robustní, rychlá konvergence, funguje i s výchozím nastavením
nevýhody	krok stále klesá	nutno volit ρ	více hyperparametrů; někdy horší generalizace než SGD + momentum

Výchozí volba

AdamW obvykle poskytne použitelný první běh bez dlouhého ladění. U velkého, dobře normalizovaného datasetu se vyplatí porovnat jej se SGD s momentem. Rozhoduje validační výsledek při srovnatelném rozpočtu.

- **SGD + momentum** — málo stavových proměnných, ale vyžaduje pečlivěji nastavit učicí rychlost a její plán.
- **Adam / AdamW** — adaptuje krok po souřadnicích; AdamW navíc odděluje weight decay od gradientu ztráty.
- **RMSProp** — exponenciální průměr druhých mocnin bez momenta prvního řádu.
- **L-BFGS** — kvazi-newtonovská metoda s omezenou pamětí, vhodná spíše pro menší hladké úlohy a velké nebo úplné dávky.

Jak vybírat

U malého hladkého modelu má smysl zkusit i L-BFGS. U velkého datasetu porovnejte AdamW se SGD s momentem při stejném výpočetním rozpočtu; rozdíly bývají menší, než se čeká, a často je přebije volba plánu učicího kroku.

Co konvergenční záruky říkají a co ne

předpoklad	výsledek	platí pro sítě?
konvexní, hladká	$O(1/T)$ ke globálnímu minimu	ne
silně konvexní	lineární konvergence	ne
nekonvexní, hladká	$O(1/\sqrt{T})$ k malému gradientu	částečně

Pro hluboké sítě platí nanejvýš třetí řádek, a i ten mluví jen o tom, že gradient bude malý — tedy že se dostaneme do okolí *nějakého* kritického bodu. O hodnotě ztráty ani o generalizaci neříká nic.

Teorie tedy neurčuje volbu optimalizátoru; slouží k pochopení, proč jednotlivé mechanismy fungují.

Praktické rozhodnutí zůstává empirické: stejný rozpočet, několik seedů a validační metrika.

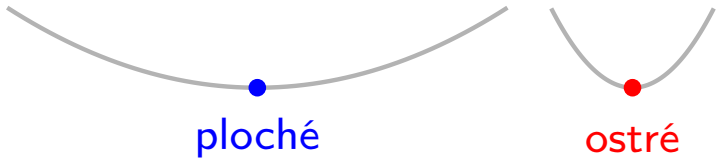
Co adaptivní metody stojí

- **Paměť.** Adam si ke každé váze pamatuje ještě dvě čísla, takže stav optimalizátoru je třikrát větší než samotný model. U sítě s miliardou parametrů jde o podstatnou položku rozpočtu.
- **Hyperparametry.** Přibývá β_1, β_2, ε a u AdamW ještě λ.
- **Srovnatelnost.** Férové porovnání používá stejný rozpočet a samostatně naladěný plán učicího kroku pro každou metodu.

Žádný optimalizátor neodstraňuje nutnost hlídat učicí rychlost, numerickou stabilitu a validační výsledek.

Ploché a ostré minimum

Dvě minima se stejnou trénovací ztrátou se mohou lišit okolím. V plochém minimu malá změna vah ztrátu téměř nezmění, v ostrém ano.



Posun mezi trénovací a testovací ztrátou se v plochém minimu projeví méně, což je obvyklé vysvětlení, proč malé dávky často generalizují lépe než velmi velké.

Opatrně s tímto vysvětlením

„Ostrost“ lze změnit pouhým přeškálováním vah, aniž by se funkce sítě jakkoli změnila. Sama o sobě tedy generalizaci neurčuje; jde o užitečnou představu, ne o měřitelné kritérium.

N. S. Keskar a kol.: On Large-Batch Training for Deep Learning. ICLR (2017). — L. Dinh a kol.: Sharp Minima Can Generalize for Deep Nets. ICML (2017).

Diagnostika běhu

Ztráta diverguje

Krok je příliš velký, chybí warmup, nebo jsou špatně škálované vstupy. Zkuste η o řád menší a sledujte normu gradientu.

Ztráta stojí

Krok je příliš malý, gradient mizí, nebo se model dostal na plošinu. Pomůže LR range test a kontrola, zda se vůbec mění váhy.

Nejprve data, potom optimalizátor

Než začnete měnit metodu, ověřte, že model dokáže přeučit jedinou malou dávku. Pokud ne, chyba je v datech, ztrátě nebo grafu — žádný optimalizátor ji nespraví.