Pokročilé metody učení neuronových sítí

Tomáš Řehořek
tomas.rehorek@fit.cvut.cz

Problém učení neuronové sítě (1)

- Nechť $N = (V, I, O, S, w, f, h)$ je dopředná neuronová síť, kde:
  - V je množina **neuronů**
  - $I \subseteq V$ je množina **vstupních neuronů**
  - $O \subseteq V$ je množina **výstupních neuronů**
  - $S \subseteq V \times V$ je množina **synapsí**
  - $w : S \to \mathbb{R}$ je **váhová funkce**
  - $f : V \to \{ \varphi \mid \varphi : \mathbb{R} \to \mathbb{R}$ je diferencovatelná $\}$ je přiřazení **aktivačních funkcí** neuronům

Problém učení neuronové sítě (2)

- Předpokládáme, že topologie sítě *(V, I, O, S)* a aktivační funkce neuronů *(f)* jsou pevně dány
- Uvažujme množinu **trénovacích příkladů** $T = \{ (\mathbf{x}_1,\mathbf{d}_1), (\mathbf{x}_2,\mathbf{d}_2), \ldots, (\mathbf{x}_k,\mathbf{d}_k) \}$, kde $\mathbf{x}_i \in \mathbb{R}^{|I|}$ a $\mathbf{d}_i \in \mathbb{R}^{|O|}$
- Definujme **chybovou funkci** $E : \mathbb{R}^{|S|} \rightarrow \mathbb{R}$ jako:

$$E(\boldsymbol{w}) = \sum_{i=1}^{|T|} \sum_{j=1}^{|O|} \left(\boldsymbol{d}_{ij} - \lambda_{\boldsymbol{w}}(\boldsymbol{x}_i)_j \right)^2$$

kde $\lambda_{\boldsymbol{w}}$ je funkce realizovaná sítí *N* za předpokladu váhové funkce $\boldsymbol{w}$

Problém učení neuronové sítě (3)

- Váhovou funkci $\boldsymbol{w}$ lze ztotožnit s odpovídajícím váhovým vektorem, tj. $\boldsymbol{w} \in \mathbb{R}^{|S|}$
- Naším úkolem je nalézt váhový vektor $\boldsymbol{w} \in \mathbb{R}^{|S|}$ takový, že $E(\boldsymbol{w})$ je minimální

Učení neuronové sítě: Algoritmy

- **Gradientní:**
  - Back-Propagation
  - QuickProp
  - Quasi-Newton
  - Levenberg-Marquardt
  - Delta-bar-Delta

- **Negradientní:**
  - Např. evoluční techniky (GNARL, SANE, NEAT, HyperNEAT…)

Back-Propagation (1)

- Objeven vícekrát nezávisle na sobě na počátku 70. let (Paul Werbos, 1974)
- Vektor $\boldsymbol{w}$ je náhodně inicializován a následně iterativně zlepšován
- V každé iteraci je určen vektor gradientu $\boldsymbol{g} = (\nabla E)(w)$ a je učiněn krok proti jeho směru (chybovou funkci minimalizujeme)
- V nejjednodušší verzi $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \eta \cdot \boldsymbol{g}$, kde parametr η nazýváme learning rate

Back-Propagation (2)

- Algoritmus pracuje ve dvou fázích:
  - **Dopředná** – jsou spočítány:
    - hodnota $\lambda_{\boldsymbol{w}}(\boldsymbol{x})$,
    - derivace aktivačních funkcí v daných bodech
  - **Zpětná** – od výstupní vrstvy je zpětně šířena chyba
    - pro každou synapsi $\boldsymbol{w}_{ij}$ ve **výstupní vrstvě** snadno zjistíme $\partial E/\partial w_{ij}$ jako $(\partial E/\partial s) * (\partial s/\partial w_{ij})$ kde s je vnitřní potenciál, pomocí rozdílu očekávaného a skutečného výstupu
    - pro synapse v každé **vnitřní vrstvě** pak počítáme gradient ze znalosti $\partial E/\partial s$ ve vrstvě následující

Back-Propagation (3)

- Nejčastější varianta: **Online learning** (okamžitý update vah), výpočty parciálních derivací v „error landscape“ probíhají již v dopředné fázi
- Náchylný k uvíznutí v lokálním minimu (obyčejný gradientní sestup)
- Klíčový parametr: η – learning rate

Optimalizace 1. a 2. řádu

- **Gradientní optimalizace**
  - optimalizace 1. řádu
  - založena na aproximaci „error landscape“ polynomem 1. stupně, tj. nadrovinou
  - bereme v potaz pouze sklon, nikoli „křivost“
- **Optimalizace 2. řádu**
  - aproximace „error landscape“ polynomem 2. stupně, tj. kvadratickou funkcí
  - problém: vyžaduje znalost **Hessovy matice** rozměrů $|S| \times |S|$, výpočetně velmi náročné

QuickProp (1)

- Scott Elliott Fahlman, 1988
- Autor rozšiřuje BP o pseudo-2.řádovou optimalizaci
- Optimalizační proces stejný jako u BP, pro každou synapsi si však pamatujeme navíc:
  - $\partial E/\partial w_{ij}$ z předchozí epochy (t –1)
  - rozdíl předchozí a současné váhy
    $\Delta w_{ij} = w_{ij}(t) - w_{ij}(t-1)$

QuickProp (2)

- Dva velmi silné předpoklady:
  - Křivka chybové funkce pro každou váhu každé synapse je **konvexní parabola**
  - Synapse **nejsou v interakci**
    - Parametry dvourozměrné paraboly jsou **nezávislé** na všech ostatních vahách
- Předpoklady v praxi téměř jistě neplatí, autor je však využívá k empiricky efektivní optimalizaci

QuickProp (3)

- V každém kroku t známe:
  - předchozí směrnici $\partial E/\partial w_{ij}\,(t-1)$
  - deltu $\Delta w_{ij} = w_{ij}(t) - w_{ij}(t-1)$
  - aktuální směrnici $\partial E/\partial w_{ij}\,(t)$
- To umožňuje jednoznačné určení paraboly
  - v novém kroku se přesuneme přímo do jejího minima za použití vztahu

$$\Delta w_{ij} = \frac{\Delta w_{ij}(t-1)}{S(t-1)-S(t)} S(t), \quad \text{kde } S(t) = \frac{\partial E}{\partial w_{ij}}(t)$$

QuickProp (4)

- Autor hovoří o **batch update** po epochách
- Odolnější proti uvíznutí v lokálním optimu než BP
- Základní parametry:
  - ***r*** … inicializační parametr vah
  - ***ε*** ... velikost kroku
  - ***μ*** … omezení velikosti kroku shora

QuickProp (5)

- ***r*** … inicializační parametr vah
  - na počátku jsou váhy vzorkovány z uniformního náhodného rozdělení U(-***r***,***r***)
  - vhodná velikost závisí na typu problému a charakteristikách aktivačních funkcí
  - přirozená volba: ***r*** = 1, doporučuje i autor

QuickProp (6)

- ε … velikost kroku
  - kroky determinovány předchozím krokem a aktuálním sklonem → nutnost učinit 1. krok!
  - velikost 1. kroku je ε, možno volit např. $\varepsilon = 1$
  - vyjde-li v nějakém kroku příliš malé Δw_{ij}, velikost váhy na mnoho epoch „zamrzne“
    - autor na základě svých experimentů navrhuje: je-li znaménko předchozí směrnice shodné se znaménkem směrnice aktuální, připočtěme k Δw_{ij} vypočtené kvadratickou formulí ještě ε-krát aktuální směrnici

QuickProp (7)

- μ … horní omezení velikosti kroku
  - po každém kroku může u paraboly nastat jedna ze situací:
    - znaménko směrnice zůstává stejné, mění se ale její velikost (stejné rameno paraboly)
    - znaménko směrnice se změní (přeskok na opačné rameno paraboly)
    - **směrnice se téměř nezmění, rozdíl je blízký nule**

QuickProp (8)

- μ … horní omezení velikosti kroku
  - pokud se velikost směrnice mnoho nezmění, vyjde nové Δw_{ij} velmi vysoké
  - proto: velikost nového skoku může být maximálně μ krát předchozí krok
  - autor doporučuje $\mu = 1.75$

Newtonova optimalizační metoda (1)

- Lokální optimalizace náhodně zvoleného počátečního řešení $\mathbf{x} \in \mathbb{R}^n$
- V každé iteraci aproximujeme okolí **x** *n*-rozměrným Taylorovým polynomem 2. stupně, tj. kvadratickou funkcí
- Cílem je nalézt bod s nulovým gradientem
  - předpokládáme, že se jedná o lokální optimum
  - hledáme v Taylorově polynomu, což má přesné analytické řešení

Newtonova optimalizační metoda (2)

- Taylorův polynom získáme jako:

$$T(\boldsymbol{x}_t) = E(\boldsymbol{x}_t) + \nabla E(\boldsymbol{x}_t)^T (\boldsymbol{x} - \boldsymbol{x}_t) + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}_t)^T \mathrm{H}E(\boldsymbol{x})(\boldsymbol{x} - \boldsymbol{x}_t)$$

$\boldsymbol{x}$ ... proměnná

$\boldsymbol{x}_t$ … řešení v čase t

$T(\boldsymbol{x}_t)$ … Taylorův polynom v čase t

$E(\boldsymbol{x}_t)$ … hodnota chybové funkce v čase t

$\nabla E(\boldsymbol{x}_t)$ … gradient chybové funkce v čase t

$\mathrm{H}E(\boldsymbol{x})$ … Hessova matice v bodě x

Newtonova optimalizační metoda (3)

- Bod s nulovým gradientem určíme analyticky v Taylorově polynomu:

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \left(\mathrm{H}E\left(\boldsymbol{x}_t\right)\right)^{-1} \nabla E\left(\boldsymbol{x}_t\right)$$

- Pozn.: v 1-D případě platí:

$$x_{t+1} = x_t - \frac{E'\left(x_t\right)}{E''\left(x_t\right)}$$

Quasi-Newton (1)

- Optimalizační metoda založená na klasické Newtonově metodě
- Předpokládá lokální aproximovatelnost chybové funkce kvadratickou funkcí, hledá stacionární bod této funkce
- **Vyhýbá se náročnému výpočtu Hessovy matice druhých derivací**
- Hessova matice je průběžně aktualizována na základě po sobě jdoucích gradientů

Quasi-Newton (2)

- Základní myšlenka:
  - Učiňme sekvenci lineárně nezávislých kroků
    $(\Delta \boldsymbol{x}_1, \Delta \boldsymbol{x}_2, \ldots, \Delta \boldsymbol{x}_n)$
  - Zaznamenejme přírůstky gradientu
    $(\Delta \boldsymbol{g}_1, \Delta \boldsymbol{g}_2, \ldots, \Delta \boldsymbol{g}_n)$
  - Hessovu matici pak spočítáme jako:

    $(\Delta \boldsymbol{g}_1, \Delta \boldsymbol{g}_2, \ldots, \Delta \boldsymbol{g}_n) * (\Delta \boldsymbol{x}_1, \Delta \boldsymbol{x}_2, \ldots, \Delta \boldsymbol{x}_n)^{-1}$
  - Potřebnou inverzi spočítáme jako:

    $(\Delta \boldsymbol{x}_1, \Delta \boldsymbol{x}_2, \ldots, \Delta \boldsymbol{x}_n) * (\Delta \boldsymbol{g}_1, \Delta \boldsymbol{g}_2, \ldots, \Delta \boldsymbol{g}_n)^{-1}$

Quasi-Newton (3)

- Algoritmus začíná s libovolnou pozitivně-definitní maticí **S**(0), která je v každém kroku aktualizována pomocí $\Delta \boldsymbol{x}_t$ a $\Delta \boldsymbol{g}_t$
- Existuje celá řada variant
  - Davidon-Fletcher-Powell (DFP)
    - historicky první Quasi-Newton metoda
  - Broyeden-Fletcher-Goldfarb-Shanno (BFGS)
    - považován za nejlepší známou QN metodu
- Typicky se používá batch update vah

Levenberg-Marquardt (1)

- Kompromis mezi:
  - Quasi-Newtonovskou optimalizací
    - konverguje rychle k lokálnímu optimu,
    - ovšem může i divergovat
  - Gradientní optimalizací
    - při správné volbě η zajištěna konvergence,
    - ovšem konverguje pomalu

Delta-Bar-Delta (1)

- Robert A. Jacobs, 1988
- Heuristická úprava BP
- Automaticky upravuje learning rate η pro každou synapsi (dimenzi v $\mathbb{R}^{|S|}$) **odděleně**
- Používá se zásadně pro batch learning, nefunguje pro online learning

Delta-Bar-Delta (2)

- Pro každou synapsi si pamatujeme směrnice z předchozích kroků (ty jsou průměrovány)
- Pokud má aktuální směrnice stejné znaménko, η se zvýší
- Pokud má aktuální směrnice opačné znaménko, η se sníží
- Urychluje konvergenci: Dynamická adaptace η pro každou dimenzi zvlášť umožňuje:
  - zrychlovat blížení se k lokálnímu optimu, je-li příliš vzdáleno
  - zpomalovat blížení se lokálnímu optimu, je-li míjeno

Delta-Bar-Delta (3)

- Na základě znalosti gradientu $g_{ij}(t)$ určíme:

$$\bar{g}_{ij}(\mathrm{t}) = (1-\beta) g_{ij}(t) + \beta \bar{g}_{ij}(t-1), \qquad 0 < \beta < 1$$

- Learning rate $\eta_{ij}(t+1)$ pak vypočteme jako:

$$\eta_{ij}(\mathrm{t}+1) = \begin{cases} \eta_{ij}(t) + \kappa, & \bar{g}_{ij}(t-1) g_{ij}(t) > 0 \\ (1-\gamma)\eta_{ij}(t), & \bar{g}_{ij}(t-1) g_{ij}(t) < 0 \\ \eta_{ij}(t), & \text{jinak} \end{cases}$$

- Problém: parametry β, κ, a γ jsou voleny uživatelem na empirickém základě

Použité zdroje

[1] Jan Drchal: Přednáška ke kurzu A4M33BIA na FEL ČVUT, 2010.

[2] Scott Elliott Fahlman: An Empirical Study of Learning Speed in Back-Propagation Networks. Technical report, 1988.

[3] Simon Haykin: Neural Networks and Learning Machines. Third Edition. Prentice Hall. 2009.

[4] Aristidis Likas, Andreas Stafylopatis: Training the random neural network using quasi-Newton methods. European Journal of Operational Research, 2000.

[5] Genevieve B. Orr: Delta-Bar-Delta. Dostupné online na http://www.willamette.edu/~gorr/classes/cs449/Momentum/deltabardelta.html

[6] Raul Rojas: Neural Networks, Springer-Verlag, Berlin, 1996.

[7] Wikipedia: Newton's method in optimization.

Děkuji za pozornost!

Tomáš Řehořek
tomas.rehorek@fit.cvut.cz