Z Wikipedie, otevřené encyklopedie

Příklad neuronové sítě: vstup, skrytá vrstva, výstup

Umělá neuronová síť, anglicky **Artificial Neural Network** či jen **Neural Network (ANN, NN)**, je jeden z výpočetních modelů používaných v oblasti umělé inteligence, podoblasti anglicky tzv. **Soft Computing** (spolu s evolučními algoritmy a fuzzy logikou). Vzorem je chování neuronů v mozku, od toho je odvozen i název, v současnosti se však princip neuronových sítí od původního zamýšleného napodobení neuronů liší.[1]

Umělá neuronová síť je struktura určená pro distribuované paralelní zpracování dat. Skládá se z umělých (formálních) neuronů, jejichž volným předobrazem je biologický neuron. Neurony jsou vzájemně propojeny synaptickými vazbami a navzájem si předávají signály a transformují je pomocí aktivačních přenosových funkcí. Neuron má libovolný počet vstupů, ale pouze jeden výstup.

Využití

Neuronové sítě se používají mimo jiné pro rozpoznávání obrazů[2] (např. ve formě pixelů) či akustických (např. rozpoznávání řeči) nebo elektrických (např. EKG, EEG) signálů[3], dále ke klasifikaci, kompresi či segmentaci dat[4], k predikci vývoje časových řad (např. burzovních indexů), k analýze psaného textu[5] či k filtrování spamu. V lékařství slouží k diagnostice onemocnění či k prohlubování znalostí o fungování informačních systémů (nervových soustav) živých organismů. Například Grossbergova síť[6] vznikla původně jako simulace fyziologického modelu rozpoznávání vzorů na sítnici lidského oka.

Historie

První umělé neurony byly vytvořeny Warrenem McCullochem a Walterem Pittsem v roce 1943.[7] Tyto neurony fungovaly tak, že byl jejich výstup 0 nebo 1 v závislosti na tom, jestli vážená suma vstupních signálů překročila prahovou hranici, nebo ne. Jejich teorie byla, že takový neuron v principu

spočte jakoukoli aritmetickou či logickou funkci. Tehdy však nebyla vypracována žádná tréninková metoda.

V padesátých letech přišel Frank Rosenblatt[8] s perceptrony, které doprovázela již i učící pravidla. Perceptrony využíval k rozpoznávání vzorů. Mimo to dokázal, že pokud existují váhy, které zadaný problém řeší, pak k nim učící pravidlo konverguje.

Počáteční nadšení však uvadlo, když se zjistilo, že takovýto perceptron umí řešit pouze lineárně separovatelné úlohy. Frank Rosenblatt se sice snažil model upravit a rozšířit, ale nedařilo se mu to a tak až do osmdesátých let přestal být o perceptrony a neuronové sítě zájem.

V 80. letech došlo k vývoji vícevrstvých perceptronových sítí s asociačními pravidly, schopných aproximovat libovolnou vektorovou funkci, díky čemuž nastala nová vlna zájmu o neuronové sítě,[9] který později částečně upadl kvůli neschopnosti učit sítě o větším počtu vrstev. Až okolo roku 2010 opět došlo k renesanci neuronových síti díky objevu hloubkového učení.

Nejužívanější způsob učení neuronových síti je algoritmus zpětného šíření chyby. Podle různých zdrojů[10][11][12] byly základy algoritmu zpětného šíření chyby v kontextu teorie řízení odvozeny z principů dynamického programování; konkrétně se na tom podíleli Henry J. Kelley v roce 1960[13] a Arthur E. Bryson v roce 1961.[14] Roku 1962 publikoval Stuart Dreyfus jednodušší odvození pomocí řetězového pravidla.[15] Vladimir Vapnik cituje ve své knize o support vector machines článek z roku 1963.[16] V roce 1969 Bryson a Yu-Chi Ho algoritmus popsali jako vícestupňovou optimalizaci dynamických systémů.

V roce 1970 Seppo Linnainmaa publikoval obecnou metodu automatického derivování (AD) diskrétních sítí vnořených diferencovatelných funkcí.[17][18] Jedná se o moderní variantu metody zpětného šíření chyby, která je efektivní i v řídkých sítích.[19][20]

V roce 1973 Stuart Dreyfus pomocí zpětného šíření chyby upravoval parametry řídicích systémů úměrně jejich chybovým gradientům.[21] Paul Werbos zmínil možnost uplatnění tohoto principu na umělé neuronové sítě roku 1974[22] a v roce 1982 tak učinil způsobem, který se používá nyní.[23] O čtyři roky později David E. Rumelhart, Geoffrey E. Hinton a Ronald J. Williams

experimentálně prokázali, že tato metoda může vést k užitečným interním reprezentacím vstupních dat v hlubších vrstvách neuronových sítí, což je základem hlubokého učení.[24] V roce 1993 byl Eric A. Wan první, kdo vyhrál pomocí backpropagace mezinárodní soutěž v modelování dat.[25]

V současnosti se neuronové sítě převážně užívají v úlohách zpracování přirozeného jazyka v rámci tzv. jazykových modelů jako např. Word2Vec či Transformer a v úlohách počítačového vidění v rámci tzv. konvolučních neuronových sítí.

Model umělého neuronu

Je popsána celá řada modelů neuronů. Od těch velmi jednoduchých používajících skokové přenosové funkce (perceptron) až po velmi složité popisující každý detail chování neuronu živého organismu, jako např. Hindmarshův–Roseův model neuronu[26].

Model umělého neuronu s funkcí S

Jedním z nejpoužívanějších je model popsaný McCullochem a Pittsem[7]:

$$y = f(\sum_{i=1}^{N}(w_i x_i) - \vartheta)$$

kde:

- x_i jsou vstupy neuronu
- w_i jsou synaptické **váhy**
- ϑ je **práh** citlivosti neuronu
- f je **aktivační funkce** neuronu
- y je výstup neuronu

Velikost **vah** w_i vyjadřuje uložení zkušeností do synapsí neuronu. Čím je vyšší hodnota, tím je daný vstup důležitější. V biologickém neuronu **práh** ϑ označuje prahovou hodnotu aktivace neuronu. Tzn. je-li $\sum_{i=1}^{N}(w_i x_i)$ menší než

práh, je neuron pasivní (inhibovaný) a je-li $\sum_{i=1}^{N}(w_i x_i)$ větší než práh, je neuron aktivní (excitovaný).

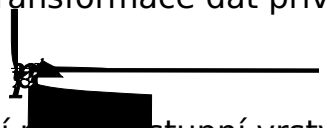
Aktivační funkce neuronu

Aktivační (**přenosová**) funkce neuronu v umělých neuronových sítích definuje výstup neuronu při zadání sady vstupů neuronu. Nelineární aktivační funkce umožňují neuronovým sítím řešit nelineární problémy. Klasická nelineární aktivační funkce je **sigmoida** (logistická funkce) o parametrech strmosti (určující šířku pásma citlivosti neuronu na svůj aktivační potenciál) a prahové hodnoty (určující posunutí počátku funkce) spolu s jejími limitními tvary jako je linearita pro strmost blížící se nule resp. ostrá nelinearita (skoková funkce) pro strmost blížící se nekonečnu:[27]

$f(x) = \frac{1}{(1 + e^{-p(x-\vartheta)})}$, tj. $\lim_{p \to 0} f(x) =$ [redacted] resp. pro

a pro

Aktivační funkce. V levém sloupci sigmoida spolu se svými limitními případy, v pravém sloupci možné transformace dat přiváděných na vstupní resp. výstupní neurony.



Volbou aktivační funkce neuronů vstupní resp. výstupní vrstvy neuronové sítě můžeme určit způsob transformace dat na síť přiváděných:

Sigmoida: – z ad 1) a ad 2) (viz níže) plyne

ad 1) z plyne

ad 2) z $0,05 = (1 + e^{3p\sigma})^{-1}$ plyne $e^{3p\sigma} = \frac{0,95}{0,05}$

- Gaussova křivka: $g(x) = e^{-p(x-\vartheta)^2}$ – z $0,05 = e^{-p6\sigma^2}$ plyne $p = -\frac{\ln 0,05}{6\sigma^2} \cong \frac{1}{2\sigma^2}$
- Mexický klobouk: $h(x) = -\sigma^2 g''(x)$ – uvedené transformaci resp. její nezáporné části odpovídají různá pásma citlivosti.

Parametry uvedených transformací mají následující význam:

ϑ – střední hodnota dat přiváděných na daný neuron z trénovací množiny

σ – směrodatná odchylka dat přiváděných na daný neuron z trénovací množiny

Kromě uvedených aktivačních funkcí se užívají ještě jejich různé modifikace:

- Identita – linearita modifikovaná posunutím středu symetrie do počátku
- Hyperbolická tangenta – rozšíření oboru hodnot sigmoidy na interval od -1 do +1
- ReLU – složení ostré linearity (vlevo od počátku) s identitou (vpravo od počátku)
- Radiální báze – Gaussova křivka resp. Mexický klobouk

Architektury sítě

> Příklady architektury neuronových sítí: rekurentní autoasociativní paměť RAM, samorganizující síť SOM, vícevrstvý perceptron MLP.

Podle způsobů propojení neuronů existuje více různých architektur neuronových sítí, neurony se řadí do jednotlivých vrstev umístěných nad sebou:[27]

Dopředná neuronová síť:

- Lineární heteroasociativní paměť – dvouvrstvá síť
- Perceptron – pouze jeden neuron
- Vícevrstvý perceptron – libovolný počet skrytých vrstev (minimálně třívrstvá síť)

- Samoorganizující síť – dvouvrstvá síť

Rekurentní neuronová síť:

- Lineární autoasociativní paměť – jednovrstvá síť
- Nelineární autoasociativní paměť (Hopfieldova) – jednovrstvá síť
- Nelineární heteroasociativní paměť (Bidirektní) – dvouvrstvá síť
- Grossbergova či ART dvouvrstvá síť spojitého času

Učení sítě

Cílem učení neuronové sítě je nastavit síť tak, aby dávala co nejpřesnější výsledky. V biologických sítích jsou zkušenosti uloženy v synapsích. V umělých neuronových sítích jsou zkušenosti uloženy v jejich matematickém ekvivalentu – váhách vazeb neuronů. Učení neuronové sítě rozlišujeme na **učení s učitelem (supervised learning)** a **učení bez učitele (unsupervised learning)**. Fáze učení neuronové sítě bývá nazývána **adaptivní**, fáze vybavování po naučení neuronové sítě bývá nazývána **aktivní**. Adaptace vah vázajících neurony závisí na jejich stavech aktivace. V případě učení s učitelem jsou stavy výstupních neuronů sítě aktivovány na ně přiloženým vnějším stimulem (výrokem učitele), v případě učení bez učitele si stavy výstupních neuronů síť nastaví v závislosti na stavech vstupních neuronů sama, a to pomocí aktivní dynamiky sítě, tj. každý krok adaptivní dynamiky je proložen průběhem dynamiky aktivní.[27]

Učení s učitelem

Při učení s učitelem trénovací data sestávají ze vstupních objektů (vektorů jejich parametrů, tj. nezávisle proměnných) a jejich požadovaných výstupních ohodnocení, tj. závisle proměnných (obecně také ve tvaru vektorů), tj. výroků učitele o objektu. Podobně jako v biologických sítích je zde využita zpětná vazba. Neuronové síti je předložen vzor. Na základě aktuálního nastavení sítě je zjištěn aktuální výsledek. Ten porovnáme s vyžadovaným výsledkem a určíme chybu. Poté spočítáme nutnou korekci (dle typu neuronové sítě) a upravíme hodnoty vah tak, abychom snížili hodnotu chyby. Toto opakujeme až do dosažení námi stanovené povolené chyby. Poté je síť adaptována. Naučená funkce sítě pak dokáže odhadovat výstupní ohodnocení každého vstupního objektu, i neobsaženého v trénovacích datech, tj. musí umět

zobecnit (generalizovat) souvislost mezi vstupy a výstupy danou příklady obsaženými v trénovacích datech. U neuronových sítí se učení s učitelem užívá vícevrstvého perceptronu.[28]

Pozn.: Pokud se vstupní objekty shodují s požadovanými výstupními ohodnoceními, mluvíme o tzv. **samoučení (self-supervised learning)**, tj. síť (např. autoenkodér) se pak učí tzv. autoasociativní fuknkci.

> Rozdíl mezi učením s učitelem (učitel označí, zda je objekt čtverec či trojúhelník) a bez učitele (síť podobné objekty nashlukuje, aniž by věděla jak se značí)

Učení bez učitele

Na rozdíl od učení s učitelem při učení bez učitele nejsou v trénovacích datech vstupní objekty provázané s jejich ohodnocením, tj. schází výrok učitele a učení bez učitele tedy vykazuje samoorganizaci sítě, která zachycuje vstupní vzory jako hustotu pravděpodobnosti.[29][30] Během učení bez učitele si stavy výstupních neuronů síť nastaví sama v závislosti na stavech vstupních neuronů, nevyhodnocujeme tedy správnost výsledku, síť si vstupní vzory sama třídí přizpůsobením své konfigurace. Vzory třídí do skupin a reaguje dle jejich typického zástupce.[31] U neuronových sítí se učení bez učitele užívá u kompetičních sítí, které si výrok učitele nahrazují tzv. laterální inhibicí.[27]

Pozn.: Kombinaci učení s učitelem a učení bez učitele označujeme jako **semi-supervised learning**.

> Míra změn vah po jednotlivých vrstvách, daná tlumením zpětného šíření chyby, tj. první tři vrstvy se neučí prakticky vůbec, smysluplně se učí pouze poslední tři.

Hluboké učení

Hluboké učení je učení neuronových sítí s velkou hloubkou.[32] Hloubkou sítě se myslí počet vrstev neuronů, které jsou propojeny tak, že výstup jedné vrstvy je vstupem vrstvy následující. U hlubokého učení se hloubka sítě nachází často v řádech desítek a více vrstev. Pro odhad parametrů sítě (trénování) se obvykle používá algoritmus zpětného šíření chyby. Trénování

probíhá ve dvou fázích, tj. nejprve předučení sítě dopředným směrem např. pomocí autoenkodérů (samoučení) a poté doučení sítě zpětným směrem (učení s učitelem), eliminuje se tak tlumení zpětného šíření chyby. Metodologie hlubokého učení se prosadila kolem roku 2010 jako základní možnost pro řešení složitých problémů strojového učení jako je klasifikace obrazů, mluvené či psané řeči nebo překlady z jednoho přirozeného jazyka do jiného.

> Model synaptické vazby ohodnocené vahou w mezi vstupním a výstupním neuronem s úrovní aktivace x a y.

Hebbovské učení

Hebbův[33] princip učení je způsob určování vah vzájemných orientovaných vazeb mezi umělými neurony jako ukládání úrovně aktivace vstupního neuronu vážené úrovní aktivace výstupního neuronu do váhy vazby mezi vstupním a výstupním neuronem. Váha vazby mezi oběma neurony se pak zvyšuje, pokud se oba neurony aktivují synchronně, a snižuje, pokud se aktivují asynchronně. Neurony, které jsou buď oba pozitivně aktivní současně nebo oba negativně aktivní současně, mají silné pozitivní váhy vzájemné vazby, zatímco ty, které jsou současně opačně aktivní, mají silné negativní váhy vzájemné vazby. Hebbovské učení se užívá u asociativních pamětí, jako je např. Hopfieldova či bidirektní síť. Pokud se do váhy vazby mezi vstupním a výstupním neuronem ukládá pouze úroveň aktivace vstupního resp. výstupního neuronu, jedná se o tzv. instar resp. outstar učení.

Přeučení

Přeučení je stav, kdy je systém příliš přizpůsoben množině trénovacích dat, ale nemá schopnost generalizace a selhává na validační množině dat, tj. vykazuje sice minimální chybu na tréninkové množině, ale na validační množině vykazuje zcela chybné výsledky. To se může stát např. při malém rozsahu trénovací množiny nebo pokud je síť příliš mohutná (např. příliš mnoho neuronů skrytých vrstev resp. počet skrytých vrstev v síti). Řešením je zvětšení počtu prvků trénovací množiny, snížení mohutnosti sítě nebo snížení počtu parametrů hodnoceného objektu, které v důsledku snižuje složitost učené funkce sítě. Krajním řešením je předčasné ukončení učení (průběžné

sledování chyby na validační množině a konec učení ve chvíli, kdy se chyba na této množině dostane do svého minima).

Trénovací data

Data určená pro učení neuronové sítě se skládají ze vstupních vektorů dat a v případě učení s učitelem z odpovídajících vstupních vektorů dat. Pro správné naučení neuronové sítě je obvykle potřeba dostatečně reprezentativní množství trénovacích dat. Neuronová síť aproximuje funkci, která ze vstupního vektoru vypočítá výstupní vektor a podle rozdílu od správného výstupního vektoru upraví své vnitřní parametry. Tento proces se opakuje, dokud není systém dostatečně naučen. Při učení se používají trénovací data, validační data či testovací data. Tato data mají být disjunktní. Pomocí trénovacích dat probíhá vlastní učení, podle chování na validačních datech se kontroluje přeučování a na testovacích datech se určuje úspěšnost chování naučené sítě na nových, neznámých datech. Trénovací data se tedy dle způsobu užití dělí do tří skupin:

- **Trénovací množina** je sada dat, ve které učící algoritmus nachází určitý vztah mezi vstupy a výstupy, tj. provádí jejich regresní analýzu.
- **Validační množina** je sada dat, která se používají pro případnou úpravu parametrů učení ve snaze vyhnout se jeho přeučení. Systém je správně naučený tehdy, jestliže se shodnou úspěšností vyhodnocuje trénovací množinu i validační množinu. Pokud má vyhodnocení trénovací množiny výrazně vyšší úspěšnost, je systém přeučený.
- **Testovací množina** je sada dat, která se používají pro ověření kvality naučeného systému.

V případě učení bez učitele se trénovací data skládají pouze ze vstupních vektorů dat obsažených v trénovací množině.

Algoritmus zpětného šíření chyby

> Gradientní sestup - vpravo nahoře naznačeno uváznutí v lokálním minimu

Algoritmus zpětného šíření chyby (**Error backpropagation algorithm**) je typická metoda učení umělých neuronových sítí. Používá se u vícevrstvých

sítí při učení s učitelem, tedy pokud je na množině příkladů použitých k učení vždy znám požadovaný výsledek. Zpětné šíření chyby je založeno na metodě gradientního sestupu.

Kvalita naučení umělé neuronové sítě je popsána chybovou funkcí, nejčastěji kvadratickou chybou[27]:


$$E = \frac{1}{2} \sum_{j=1}^{p} \sum_{i=1}^{n} (y_{ij} - o_{ij})^2$$

, pak

– chyba

– počet vzorů předložených síti

– počet neuronů výstupní vrstvy

– požadovaný výstup daného neuronu a daného vzorku

– vypočítaný výstup daného neuronu a daného vzorku

– vektor přírůstků vah gradientního kroku

– velikost gradientního kroku

Zpětné šíření chyby

Jeden sestupný gradientní krok pak může vypadat následovně:[27]

– potenciál i-tého neuronu

– skutečný stav i-tého neuronu

– požadovaný stav i-tého neuronu

– adaptační funkce i-tého neuronu

– strmost aktivační funkce i-tého neuronu

– práh i-tého neuronu

– váha vazby i-tého neuronu s j-tým neuronem

– velikost gradientního sestupného kroku

– míra setrvačnosti gradientního sestupu

V_L – populace neuronů L-té vrstvy
M – počet vrstev sítě
Δ – předcházející přírůstek příslušné proměnné

Cílem učení je minimalizovat chybovou funkci závislou na vstupních vahách neuronů, přičemž gradientní sestup obecně najde pouze lokální minimum, proto se do gradientního sestupu zavádí jistá jeho setrvačnost, spočívající v míře respektování směru jeho minulého sestupného kroku, tj. k aktuálnímu gradientu se připočte minulý gradient a aktuální sestupný krok se provede ve směru jejich součtu. Tato deformace gradientního sestupu pak umožní vyklouznutí z mělkého lokálního minima. Učení neuronové sítě spočívá ve změně vah vstupů neuronů. Algoritmus zpětného šíření chyby v každém kroku postupuje v následujících třech fázích:

- Aplikují se učící vzory a pro každý vzor se postupně směrem vpřed napočítají výstupy (vstupní signál se sítí šíří směrem dopředu).
- Napočítané výstupy se porovnají s požadovanými výstupy, tj. spočte se chyba, jak byla popsána výše.
- Na základě spočtené chyby se počítají hodnoty adaptačních funkcí ve směru od poslední vrstvy k první vrstvě (pro výpočet hodnoty adaptačních funkcí podřazené vrstvy již musí být vypočteny hodnoty adaptačních funkcí nadřazené vrstvy), tj. spočítá se gradient chybové funkce, na základě kterého se provede sestupný gradientní krok, tj. upraví se vstupní váhy neuronů tak, že klesne hodnota chyby. Výpočet tedy postupuje zpětně od výstupní vrstvy až po vstupní vrstvu (odtud zpětné šíření chyby), váhy se mění podle jejich vlivu na chybu.

Odkazy

Reference

1. ↑ HOŘEJŠ, J.; KUFUDAKI, O. *Počítače a mozek (neuropočítače)*. [s.l.]: SOFSEM, 1988. 38 s.
2. ↑ VENKATESAN, Ragav; LI, Baoxin. *Convolutional Neural Networks in Visual Computing: A Concise Guide*. [s.l.]: CRC Press, 2017-10-23. Dostupné online. ISBN 978-1-351-65032-8. (anglicky)

3. ↑ ZHA, Xiong; PENG, Hua; QIN, Xin; LI, Guang. *A Deep Learning Framework for Signal Detection and Modulation Classification*. [s.l.]: MDPI, 2019-9-19. Dostupné online. (anglicky)
4. ↑ KOHONEN, Teuvo. *The self-organizing map*. [s.l.]: Proceedings of the IEEE 78 (9), 1990. 17 s. (anglicky)
5. ↑ A Beginner's Guide to Word2Vec and Neural Word Embeddings. *Pathmind* [online]. [cit. 2022-10-29]. Dostupné online. (anglicky)
6. ↑ GROSSBERG, Stephen. *Recurrent neural networks*. [s.l.]: Scholarpedia, 2013. Dostupné online. (anglicky)
7. ↑ ***a*** ***b*** MCCULLOCH, W. S. T.; PITTS, W. *A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics 5*. [s.l.]: [s.n.], 1943. 19 s. (anglicky)
8. ↑ ROSENBLATT, F. *The Perceptron - a perceiving and recognizing automaton, Report 85–460–1, Cornell Aeronautical Laboratory*. [s.l.]: [s.n.], 1957. (anglicky)
9. ↑ HAGAN, Martin T. *Neural network design*. druhé. vyd. [s.l.]: [s.n.], 2014. 800 s. Dostupné online. (anglicky)
10. ↑ Stuart Dreyfus: *Artificial Neural Networks, Back Propagation and the Kelley-Bryson Gradient Procedure.* In: *J. Guidance, Control and Dynamics.* 1990.
11. ↑ Jürgen Schmidhuber: *Deep learning in neural networks: An overview.* In: *Neural Networks.* 61, 2015, S. 85–117. ArXiv
12. ↑ Jürgen Schmidhuber: *Deep Learning.* In: *Scholarpedia.* 10(11), 2015, S. 328–332. Section on Backpropagation
13. ↑ Henry J. Kelley: *Gradient theory of optimal flight paths.* In: *Ars Journal.* 30(10), 1960, S. 947–954. (online)
14. ↑ Arthur E. Bryson: *A gradient method for optimizing multi-stage allocation processes.* In: *Proceedings of the Harvard Univ. Symposium on digital computers and their applications.* April 1961.
15. ↑ Stuart Dreyfus: *The numerical solution of variational problems.* In: *Journal of Mathematical Analysis and Applications.* 5(1), 1962, S. 30–45. (online)
16. ↑ A. E. Bryson, W. F. Denham, S. E. Dreyfus: *Optimal programming problems with inequality constraints. I: Necessary conditions for extremal solutions.* In: *AIAA J.* 1, 11, 1963, S. 2544–2550.

17. ↑ Seppo Linnainmaa: *The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors.* Master's Thesis (in Finnish), Univ. Helsinki, 1970, S. 6–7.
18. ↑ Seppo Linnainmaa: *Taylor expansion of the accumulated rounding error.*In: *BIT Numerical Mathematics.* 16(2), 1976, S. 146–160.
19. ↑ Andreas Griewank: *Who Invented the Reverse Mode of Differentiation?* Optimization Stories. In: *Documenta Matematica.* Extra Volume ISMP, 2012, S. 389–400.
20. ↑ Andreas Griewank, Andrea Walther: *Principles and Techniques of Algorithmic Differentiation.* 2. Auflage. SIAM, 2008.
21. ↑ Stuart Dreyfus: *The computational solution of optimal control problems with time lag.* In: *IEEE Transactions on Automatic Control.* 18(4), 1973, S. 383–385.
22. ↑ Paul Werbos: *Beyond regression: New tools for prediction and analysis in the behavioral sciences.* PhD thesis. Harvard University 1974.
23. ↑ Paul Werbos: *Applications of advances in nonlinear sensitivity analysis.* In: *System modeling and optimization.* Springer, Berlin/ Heidelberg 1982, S. 762–770. (online)
24. ↑ David E. Rumelhart, Geoffrey E. Hinton, Ronald J. Williams: *Learning representations by back-propagating errors.* In: *Nature.* Band 323, 1986, S. 533–536.
25. ↑ Eric A. Wan: *Time series prediction by using a connectionist network with internal delay lines.* In: *Santa Fe Institute Studies in the Sciences of Complexity-Proceedings.* Vol. 15, Addison-Wesley Publishing Co., 1993, S. 195–195.
26. ↑ HINDMARSH, J. L.; ROSE, R. M. *A model of neuronal bursting using three coupled first order differential equations. Biological Sciences 221 (1222)*. London: The Royal Society, 1984. 16 s. (anglicky)
27. ↑ ***a*** ***b*** ***c*** ***d*** ***e*** ***f*** KŘIVAN, Miloš. *Umělé neuronové sítě*. první. vyd. Praha: Oeconomica, 2021. 76 s. Dostupné online. ISBN 978-80-245-2420-7.
28. ↑ GENTLEMAN, R.; CAREY, V. J. *Supervised Machine Learning. Bioconductor Case Studies*. New York: Springer, 2008. 16 s. Dostupné online. ISBN 978-0-387-77239-4. (anglicky)
29. ↑ HINTON, Geoffrey; SEJNOWSKI, Terrence. *Unsupervised Learning: Foundations of Neural Computation*. [s.l.]: MIT Press, 1999. ISBN 978-0262581684.

30. ↑ GENTLEMAN, R.; CAREY, V. J. *Unsupervised Machine Learning. Bioconductor Case Studies*. New York: Springer, 2008. 21 s. Dostupné online. ISBN 978-0-387-77239-4. (anglicky)
31. ↑ GARLÍK, B.; KŘIVAN, M. *Identification of type daily diagrams of electric consumption based on cluster analysis of multi-dimensional data by neural network*. [s.l.]: Neural Network World 3/13, 2013. Dostupné online. S. 271-283. (EN)
32. ↑ GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. *Deep Learning*. [s.l.]: MIT Press, 2016. 767 s. Dostupné online. (anglicky)
33. ↑ HEBB, D.O. *The Organization of Behavior*. New York: Wiley & Sons, 1949. Dostupné online. (anglicky)

Související články

- Modulární neuronová síť
- Elmanova Jordanova neuronová síť
- Polynomiální neuronová síť

Externí odkazy

- Obrázky, zvuky či videa k tématu umělá neuronová síť na Wikimedia Commons
- Umělá neuronová síť v České terminologické databázi knihovnictví a informační vědy (TDKIV)
- Předpovídání pomocí neuronové sítě – včetně Java appletu pro experimenty s předpovídáním funkce

Portály: Informační věda a knihovnictví

Citováno z „https://cs.wikipedia.org/w/index.php?title=Umělá_neuronová_síť&oldid=25469332“
Kategorie:

- Umělé neuronové sítě

Skryté kategorie:

- Údržba:Články s dočasně použitou šablonou
- Monitoring:Články s odkazem na TDKIV
- Monitoring:Články s identifikátorem NKC

- Monitoring:Články s identifikátorem TDKIV
- Monitoring:Články s identifikátorem GND
- Monitoring:Články s identifikátorem LCCN
- Monitoring:Články s identifikátorem LNB
- Monitoring:Články s identifikátorem NDL
- Monitoring:Články s identifikátorem NLI
- Portál Informační věda a knihovnictví/Zapojené články

Hledání

Speciální:Hledání

Hledat

Umělá neuronová síť

69 jazyků

Přidat téma