1 Back-propagation algoritmus pre vrstvové siete

Budeme uvažovať inkrementálnu metódu, pre $m = 0, 1, 2, \ldots, M$ číslo vrstvy. Označme

$$\begin{array}{rcl} \mu & \ldots & \text{číslo vzoru} \\ M & \ldots & \text{počet vrstiev neurónovej siete} \\ V_i^m & \ldots & \text{výstup } i\text{-tej vrstvy v } m\text{-tej vrstve} \\ V_1^0 & \ldots & i\text{-tý vstup} \\ w_{ij}m & \ldots & \text{váha z } V_j^{m-1} \text{ do } V_i^m \end{array}$$

Back-propagation algoritmus je učenie s učiteľom.

1.1 Popis algoritmu

1. inicializuj váhy na malé náhodné hodnoty
2. vyber vzor $\bar{x}^\mu$ a aplikuj ho na vstupnú vrstvu, takže
$$V_k^P = x_k^\mu \quad k \in \langle 1, N \rangle$$
3. propaguj signál dopredu použitím
$$V_i^m = g(h_i^m) = g(\sum_j w_{ij}^m V_j^{m-1})$$
pre všetky hodnoty i, m. Teda budú vypočítané aj koncové výstupy V_i^M
4. vypočítaj adaptačné hodnoty pre výstupnú vrstvu
$$\delta_i^M = g'(h_i^M)[\xi_i^\mu - V_i^m]$$
Porovnaj skutočný výstup V_i^M s očakávanou hodnotou ξ_i^μ u vzoru μ.
5. vypočítaj adaptačné hodnoty pre predchádzajúce vrstvy propagáciou chýb naspäť
$$\delta_i^{m-1} = g'(h_i^{m-1}) \sum_i w_{ji}^m \delta_j^m$$
6. uprav všetky váhy takto
$$\begin{array}{rcl} \Delta w_{ij}^m & = & \eta \delta_i^m V_j^{m-1} \\ w_{ij}^{m\text{nové}} & = & w_{ij}^{m\text{staré}} + \Delta w_{ij}^m \end{array}$$

Podrobné odvodenie ukážeme pri rekurentných sieťach, ktoré sú zovšeobecnením vrstvových sietí. Metóda back-propagation je veľmi využívaná pri vrstvových sieťach.

2 Univerzálna aproximácia

Neurónové siete sú *univerzálnym aproximátorom* funkcií.

Definícia 2.1
Nech U je trieda funkcií, nech $T \subset U$ a nech ρ je metrika na U. Trieda T sa nazýva *aproximátorom* vzhľadom na (U, ρ), ak T je hustá na U (vzhľadom na topológiu indukovanú ρ).
Ak je trieda funkcií T aproximátorom vzhľadom k triede spojitých reálnych funkcií (alebo $\mathcal{L}_p$ funkcií[1]), budeme ju nazývať *univerzálnym aproximátorom*.

[1] funkcií s konečným počtom bodov nespojitosti

Definícia 2.2
Označíme $\mathcal{C}(X)$ priestor všetkých spojitých funkcií nad kompaktným priestorom X so supremovou normou
$$\|f\| = \sup_{x \in X} |f(x)|$$
Ďalej označíme $\mathcal{L}_p$ priestor funkcií[2], pre ktoré je $|f|^p$ integrovateľná na množine X normou
$$\|f\| = \left[\int_{x \in X} |f(x)|^p \mathrm{d}x \right]^{\frac{1}{p}}$$
Neurónové siete sú univerzálnym aproximátorom vzhľadom na tieto definície, čiže dokážu aproximovať každú spojitú funkciu.
Predpokladajme, že chceme neurónovú sieť „naučiť“ hyperbolu. Keďže funkcia hyperboly je ne-

Obr. 1: Funkcia hyperboly a jej naučená verzia

spojitá v bode 0, tento bod je pre neurónovú sieť problémom. Čím sme bližšie k bodu nespojitosti, tým väčšie nepresnosti vznikajú.

2.1 Prečo neurónové siete aproximujú dobre?

A. N. Kolmogorov sa zaoberal trinástym Hilbertovým problémom – problémom prevodu – a to, či možno napr. funkciu 3 premenných vyjadriť ako funkciu (alebo funkciu funkcií) 1 premennej.

Veta 2.1 (Kolmogorov 1957)
Pre každé $n \geq 2$ existujú spojité reálne funkcie ψ^{pq} definované na $I_1 = \langle 0, 1 \rangle$ také[3], že každá spojitá reálna funkcia n premenných definovaná na n-rozmernom intervale $I^n = \langle 0, 1 \rangle^n$, $f \in \mathcal{C}(I^n)$ môže byť vyjadrená v tvare
$$f(x_1, \ldots, x_n) = \sum_{q=1}^{2n+1} \xi_p (\sum_{p=1}^{n} (\psi^{pq}(x_p)),$$
kde $\xi^p(y)$ sú spojité reálne funkcie jednej premennej.

Túto vetu možno interpretovať v zmysle, že 1 rozmer stačí na všetko. Dôkaz tejto vety je konštruktívny.
Kolmogorovu vetu možno zosilniť v nasledovnom tvare:

[2]Ak položíme $p = 2$, máme priestor $\mathcal{L}_2$ a budeme integrovať funkcie $|f(x)|^2$, čím v podstate získavame euklidovskú mieru.
[3]Tento interval je kompaktný.

Veta 2.2 (Sprecher & Lorentz)

$$f(x_1, \ldots, x_n) = \sum_{q=1}^{2n+1} \varphi(\sum_{p=1}^{n} \lambda^{pq} \psi_q(x_p)),$$

kde λ je reálna konštanta a ψ_q sú monotónne rastúce funkcie nezávislé od f a pre danú funkciu rovnaké.

Dôkaz tejto vety je už menej konštruktívny. Využívajú sa pri ňom funkcie nazývané „diabolské stupne“.

2.2 Využitie Kolmogorovovej vety v neurónových sieťach

Definícia 2.3

- Sigmoidálna funkcia $\sigma : \mathbb{R} \to I$ má tieto vlastnosti

$$\lim_{z \to -\infty} \sigma(z) = 0 \qquad \lim_{z \to \infty} \sigma(z) = 1$$

- Modul spojitosti funkcie f

$$\begin{aligned} \omega_f(\delta) \quad = \quad & \sup\{|f(x_1, \ldots, x_n) - f(y_1, \ldots, y_n)| : \\ & (y_1, \ldots, y_n), (x_1, \ldots, x_n) \in I^n : \\ & |x_i - y_i| < \delta; \forall i = 1, \ldots, n\} \end{aligned}$$

- Schodište typu σ, ozn. $S(\sigma)$ je funkcia $f : \mathbb{R} \to \mathbb{R}$

$$f(x) = \sum_{i=1}^{k} a_i \cdot \sigma(b_i x + c_i),$$

  pričom $a_i, b_i, c_i \in \mathbb{R}$

Veta 2.3
Nech $n \in \mathbb{N}$, $n \geq 2$, nech $\sigma : \mathbb{R} \to I$ je sigmoidálna funkcia, nech $f \in \mathcal{C}(I^n)$ a nech $\varepsilon > 0, \varepsilon \in \mathbb{R}$ Potom existuje $k \in \mathbb{N}$ a funkcie $\varphi_i, \psi_{pi} \in S(\sigma)$ také, že

$$|f(x_1, \ldots, x_n) - \sum_{i=1}^{k} \varphi_i\Big(\sum_{p=1}^{n} \psi_{pi}(x_p)\Big)| < \varepsilon \quad \forall (x_1, \ldots, x_n) \in I^n$$

Táto veta hovorí, že v neurónovej sieti nám stačí jedna skrytá vrstva. Ďalej v tvrdení k označuje počet neurónov. Žiaľ, veta neudáva koľko neurónov je potrebných, dokonca nemáme ani odhad, koľko neurónov máme na túto skrytú vrstvu dať. Ich počet treba určiť experimentálne. Naproti tomu, v Kolmorogovovej vete je počet neurónov presne daný.
Ak však použijeme viac ako jednu vrstvu, už vieme odhadnúť maximálny počet neurónov na jednej vrstve. Navyše, neurónová sieť sa chová stabilnejšie. Teda, napriek tomu, že jedna vrstva by postačovala na všetko, predsa má použitie viacerých vrstiev význam (v praxi sa zväčša používaju 2–3 vrstvy).

2.3 Aproximácia spojitých funkcií perceptrónovou sieťou s 2 skrytými vrstvami. Odhad počtu neurónov

Veta 2.4
Nech $n \in \mathbb{N}, n \geq 2$, nech sigmoidálna funkcia $\sigma : \mathbb{R} \to I$, nech $f \in \mathcal{C}(I^n)$, $\varepsilon > 0, \varepsilon \in \mathbb{R}$. Potom pre každé $m \in \mathbb{N} : m \geq 2n + 1$ a

$$\frac{n}{m-n} + v < \frac{\varepsilon}{\|f\|} \qquad \omega_f\left(\frac{1}{n}\right) < \frac{m-n}{2m-n} \cdot v$$

pre nejaké $v \in \mathbb{R}, v > 0$ môže byť funkcia f aproximovaná s presnosťou ε dvojvrstvovou perceptrónovou sieťou, pričom

- *v 1. vrstve je $n \cdot m \cdot (m + 1)$ jednotiek*
- *v 2. vrstve je $m^2(m + 1)^n$ jednotiek*

s aktivačnou funkciou r.
Navyše, všetky váhy a prahy siete okrem váh spájajúcich 2. vrstvu s výstupnou jednotkou sú pevné pre všetky funkcie g, pre ktoré platí

$$\|g\| \leq \|f\| \qquad \wedge \qquad \omega_g \leq \omega_f$$

Metóda back-propagation funguje aj na rekurentných sieťach.

3 Algoritmus back-propagation na rekurentných sieťach

Označme

$1, 2, 3$	–	vstupné neuróny
$6, 7$	–	výstupné neuróny
vstupné a výstupné neuróny	–	viditeľné neuróny
ostatné neuróny	–	skryté
V_j	–	počiatočné stavy
w_{ij}	–	váhy od i-teho k j-temu neurónu

Back-propagation algoritmus už nemôže pracovať presne tak, ako u vrstvových sieťach – adaptácia nemôže prebiehať po vrstvách zhora nadol, pretože pri adaptácii by sa adaptovali aj neuróny 6,7 (keďže sú prepojené so skrytou vrstvou). Riešením je vytvorenie pomocnej siete propagujúcej chybu, ktorá bude mať rovnakú štruktúru ako pôvodná, len hrany budú mať opačnú orientáciu. Potom označíme

$$\begin{aligned} Y_j &\quad - \quad \text{počiatočné stavy, t.j. chyba na začiatku} \\ \bar{w}_{ij} &\quad - \quad \text{váhy} \end{aligned}$$

Na začiatku zvolíme Y_j, V_j náhodne. Váhy $w_{ij}, \bar{w}_{ij}$ sú rovnaké, rovnaké sú aj počiatočné stavy. V algoritme budeme pracovať s oboma sieťami naraz.

3.1 Výpočet aktivačných hodnôt neurónov

Označme

$$x_i = \begin{cases} \text{vstup} & \\ 0 & \text{ak } i\text{-tý neurón nie je vstupný} \end{cases}$$

Potom

$$\begin{aligned} V_i &= g(h_i) = g(\sum_{j=1}^{n} \omega_{ij} V_j + x_i) \\ \text{chyba siete: } E &= \frac{1}{2} \sum k = 1^n E_k^2 \\ E_k &= \begin{cases} \xi_k - V_k & \text{ak máme výstupný neurón} \\ 0 & \text{chyba nevýstupného neurónu} \end{cases} \\ \Delta w_{pw} &= \nu \frac{\partial E}{\partial w_{pq}} \end{aligned}$$

V poslednej rovnosti je ∂E parciálna derivácia v smere váhy w_{pq}, a ∂w_{pq} je najmenší gradient spádu.[4]

3.2 Popis back-propagation algoritmu na rekurentných sieťach

Algoritmus 3.1 (rekurentný back-propagation algoritmus)

1. inicializuj váhy w_{ij}, $j = 1, \ldots, n$ a štartovacie hodnoty $V_j, Y_j; j = 1, \ldots, n$ na malé náhodné hodnoty.
   Priraď váham do vstupných neurónov hodnotu 1, prahovým hodnotám hodnotu 0.

2. vyber príklad a aplikuj ho na vstup a vypočítaj aktivačné hodnoty všetkých neurónov pomocou vzťahu

$$V_i = g(h_i) = g\left(\sum_{j=1}^{n} V_j w_{ij} + x_i\right)$$

3. pomocou výstupných hodnôt výstupných neurónov vypočítaj chybu originálnej siete

$$E_i = \xi_i - V_i$$

4. vypočítaj hodnoty error-propagation siete pomocou

$$\bar{w}_{ij} = g'(h_j) w_{ji}$$

5. hodnoty E_i sú prezentované na vstupe siete, ktorá propaguje chybu a nové hodnoty pomocou

$$Y_i = \bar{g}(h_i) = \bar{g}\left(\sum_{i=1}^{n} \bar{w}_{ij} Y_j + E_i\right)$$

[4] Chceme najväčší spád, aby redukcia chyby bola čo najväčšia.

6. použitím Y_i vypočítaj δ_i váh originálnej siete, kde

$$\begin{aligned} \Delta w_{ij} &= y\delta_i V_j \\ \delta_i &= g'(h_i)Y_i \end{aligned}$$

7. adaptuj všetky prepojenia.

$$w_{ij}^{\text{nové}} = w_{ij}^{\text{staré}} + \Delta w_{ij}$$

8. prejdi na krok 2 a opakuj algoritmus pre nový príklad

Typeset by LaTeX, Róbert Novotný, 31. októbra 2002