VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Demonstrace učení Madaline III
Projekt do SFC

5. prosince 2021 Michal Sova

1 Adaline

Pojem adaline (Adaptive linear neuron) byl poprvé představen roku 1960 Bernardem Widrowem and Tedem Hoffem[1]. Adaline je neuron podobný perceptronu[1], liší se však ve způsobu učení a úpravou vah. Zatímco perceptron upravuje své váhy podle výstupu své aktivační funkce a očekávané hodnoty, adaline upravuje váhy dle svého potenciálu, tedy vstupem své aktivační funkce. Celý proces učení lze popsat následujícím způsobem:

$$\vec{w}(0) = \textit{libovolé} \tag{1}$$

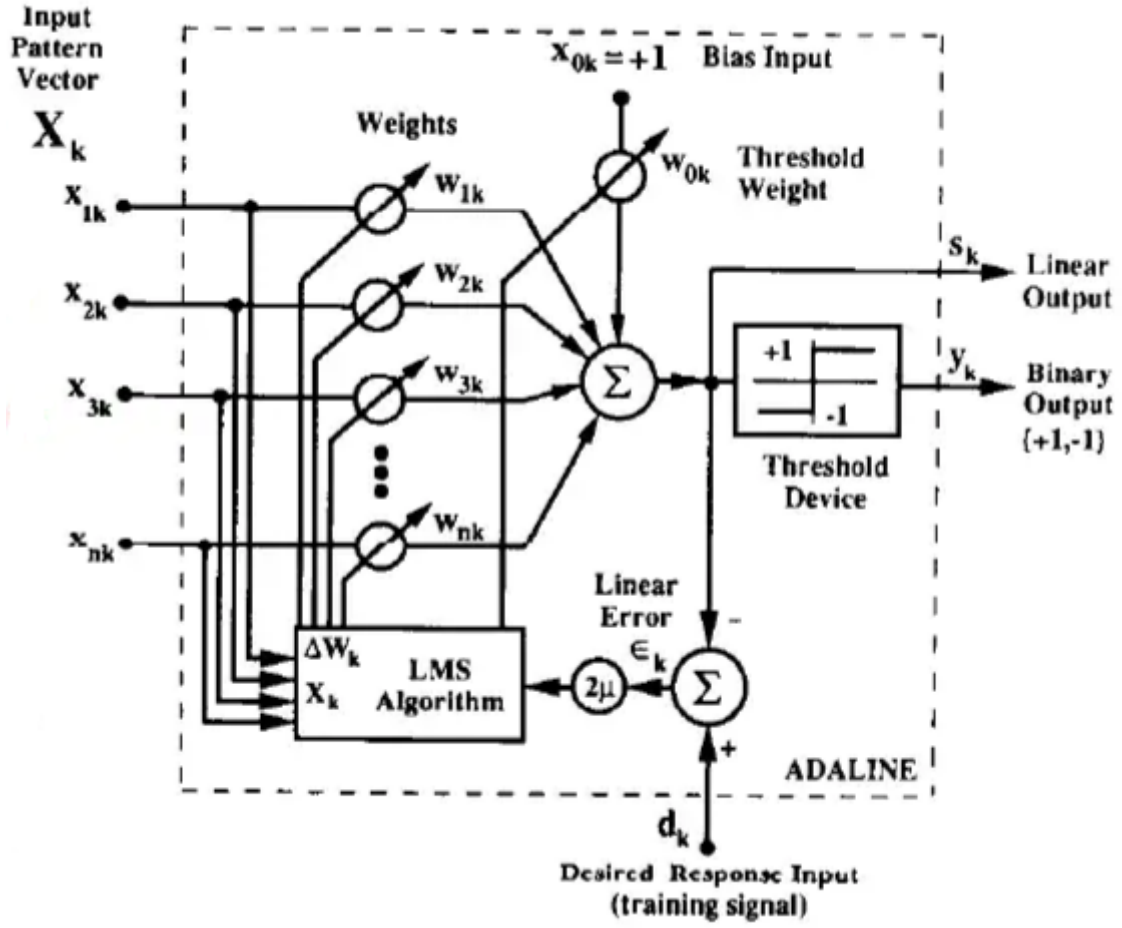
$$\vec{w}(k) = \vec{w}(k-1) + \mu * (d(k) - u(k)) * \vec{x}(k) \tag{2}$$

Kde:

- $\vec{w}(0)$ je počáteční vektor vah,
- $\vec{x}(k)$ je k-tý vstup adaline,
- $\vec{w}(k-1)$ jsou staré váhy adaline,
- $\vec{w}(k)$ jsou nové váhy adaline,
- $d(k)$ je předpokládaný výstup vstupu $\vec{x}(k)$.
- $u(k)$ je potenciál neuronu .

Celý proces lze také vidět na obrázku 1, kde vstupní vektor je X_k a jednotlivé váhy jsou $w_0k, w_1k, ..., w_nk$. Dále d_k je předpokládaný výstup, y_k výstup adaline a s_k potenciál. Na obrázku si lze povšimnout *"LMS Algorithm"*, tento algoritmus je ekvivalentní s rovnicemi 1 a 2.

[1]Perceptron je nejjednodušším modelem dopředné neuronové sítě. Sestává pouze z jednoho neuronu. [https://cs.wikipedia.org/wiki/Perceptron]



Obrázek 1: Učení adaline [5]

2 Madaline III

2.1 Madaline

Madaline je síť z adaline neuronů. Původní madaline I je dvouvrstvá síť, kde adaline ve skryté vrstvě mají nastavovatelné váhy a adaline ve výstupní vrstvě mají pevné váhy, realizující určitou funkci (například OR, XOR)[2]. Princip učení Madaline Rule I a II je podobný. Oba typy madaline mají skokové aktivační funkce.

Při učení Madaline Rule I se pro každý prvek trénovací množiny, kde byl nesprávně vyhodnocen výstup, nejdříve vybere několik adaline, které mají nejnižší absolutní hodnotu potenciálu. Pro tyto adaline se pak postupně zkouší, zda úprava znaménka potenciálu dané adaline (tím se taktéž změní výstup adaline) změnila odezvu sítě. Pokud ano, změní se váhy dané adaline tak, aby změna byla trvalá.

Princip učení Madaline Rule II je podobný jak u Madaline Rule I. Madaline Rule II provádí změnu vah příslušných adaline po vrsvách, narozdíl od Madaline Rule I.

2.2 Madaline Rule III

Madaline Rule III (MRIII) je narozdíl od ostatních modelů madaline síť, jejíž adaline mají sigmoidální aktivační funkce. Princip učení MRIII je rozdílný oproti Madaline Rule I a II. Díky spojité aktivační funkci algoritmus minimalizuje chybu odezvy ve směru záporné derivace této chyby. Algoritmus MRIII je matematicky ekvivalentní s algoritmem zpětné propagace[2], ovšem změna vah u algoritmu MRIII je výpočetně složitější než u zpětné propagace.

V obrázku 2, kde jsou dva výstupy, je chyba sítě definována jako:

$$e_k^2 = (d_{1k} - y_{1k})^2 + (d_{2k} - y_{2k})^2 \tag{3}$$

Kde d_{1k}, d_{2k} je očekávaný výstup k-tého vstupního trénovacího vektoru a y_{1k}, y_{2k} je výstup sítě k-tého vstupního trénovacího vektoru. Pokud je k potenciálu p-té adaline je přidáno malé Δs (*perturbation*), pak nová chyba sítě je:

$$e_{kp}^2 = (d_{1kp} - y_{1kp})^2 + (d_{2kp} - y_{2kp})^2 \tag{4}$$

Pak vzorec pro úpravu vah p-té adaline je následující:

$$\vec{w}_{kp} = \vec{w}_{kp} - 2 * \mu * e_k * \frac{e_{kp} - e_k}{\Delta s} * \vec{x}_{kp} \tag{5}$$

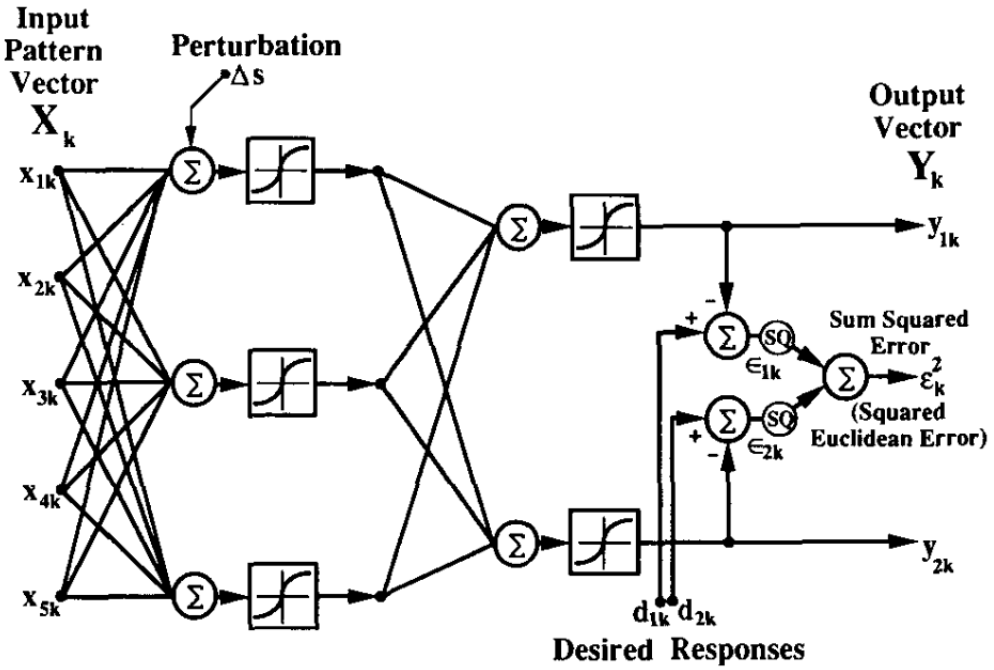
Kde:

- $\vec{w}_{kp}$ jsou váhy p-té adaline
- $\vec{x}_{kp}$ je vstupní vektor p-té adaline
- μ je koeficient učení

Celý algoritmus učení pak vypadá následovně:

1. Pro každý prvek k trénovací množiny:
2. Vypočti výstup a chybu sítě e_k
3. Pro každou vrstvu v síti:
4. Pro každou adaline p v dané vrstvě:
5. Přidej k potenciálu adaline p malé Δs
6. Spočítej chybu sítě e_{kp}
7. Uprav váhy $\vec{w}_{kp}$ podle rovnice 5
8. Pokud chyba sítě není dostatečná a nebyl překročen maximální počet opakování vrať se na 1, jinak skonči

[2]Zpětná propagace viz.[2]



Obrázek 2: Princip učení Madaline Rule III [3]

3 Implementace

MRIII je implementována v jazyce C++, za použití standardních knihoven jazyka C++17. Celá neuronová síť je vytvořená jako třída Madaline. Třída je tvořena vrsvami neuronů, které jsou vytvořené jako třída Adaline. Program nejprve zpracuje argumenty, poté zkonstruuje neuronovou síť.

V případě, že byly dány správné argumenty, může se daná síť natrénovat na trénovacích datech ze souboru. Natrénovanou síť a její váhy lze uložit.

3.1 Třída Adaline

Třída Adaline si nejdříve inicializuje svůj index (pozici v dané vrstvě), váhy na náhodnou hodnotu, μ a Δs. Nejdůležitější funkcí je funkce *forward*, která ze vstupu neuronu $\vec{x}$ vypočte výstup y. Adaline nejdříve ve funkci forward vypočte potenciál u neuronu a poté pomocí aktivační funkce vypočte a uloží si výstup. Aktivační funkce adaline je následující:

$$y = \frac{1}{1 + e^{-u}} \tag{6}$$

Kde y je výstup neuronu a u je potenciál neuronu a výstup y je v rozmezí $< 0; 1 >$.

Další důležitou funkcí je funkce *update_weights*, která upraví váhy podle rovnice 5.

3.2 Třída Madaline

Pro běh sítě madaline je důležitá její topologie. Tu lze načíst ze souboru, buď neinicializovanou, nebo již s nastavenými vahami [3].

[3] viz. kapitola 4

První vrstva je použita jako vstup. Neurony první vrstvy nemají váhy (jenom bias, který se ale nikde nepoužívá) a na jejich výstup je ukládán vstup. Další vrstvy už jsou implementovány dle obrázku 2, tzn. vstup každé adaline ve vrstvě l je namapován na výstup vrsvy $l - 1$.

Trénovvání sítě probíhá ve funkci *train* a funkci *update_network*. Funkce *train* postupně nad každým prvekem trénovací množiny volá funkci *update_network*. Tato funkce pak dle algoritmu v kapitole 2.2 (kroky 2 až 7).

4 Návod na použití

4.1 Překlad

Součástí je Makefile, za pužití příkazu 'make' se projekt přeloží.

4.2 Spuštění

Příkazy:

- ./main [-h] -g TOPOLOGY [-r TRAIN] [-t TEST] [-s SAVE] [-m MI] [-e EPS] [-p THRESHOLD] [-i I] [-d LEVEL]
- ./main [-h] -l LOAD [-r TRAIN] [-t TEST] [-s SAVE] [-m MI] [-e EPS] [-p THRESHOLD] [-i I] [-d LEVEL]

První příkaz načte topologii ze zvoleného souboru TOPOLOGY. Tento soubor obsahuje jediný řádek s počty neuronů v každé vrstvě. Pokud je v souboru například:
4 4 2
značí, že jsou na vstupu 4 hodnoty, 1. vrstva (vstupní) obsahuje 4 adaline a 2. vrstva (výstupní) obsahuje 2 adaline.

Druhý příkaz načte již inicializovanou síť (s již inicializovanými vahami jednotlivých neuronů) ze souboru LOAD, kde každý řádek je jedna vrstva sítě a první řádek je vstupní vektor.

4.3 Další důležité argumenty

- -h, –help: zobazí nápovědu a ukončí se
- -r, –train TRAIN: načte trénovací data, trénovací data představují jeden řádek = jeden vzorek, jednotlivé hodnoty jsou odděleny mezerou, řádek obsahuje jak vstup tak vzorový výstup, program na této trénovací množině natrénuje neuronovou síť
- -t, –test TEST: stejné jak u trénovacích dat; program netrénuje, jen testuje
- -s, –save SAVE: uložení sítě (možné načíst pomocí argumentu -l LOAD)
- -p, –threshold THRESHOLD: minimální chyba sítě při učení (pokud je průměrná chyba sítě menší než tento práh, trénování se ukončí)
- -d, –debug LEVEL: úroveň výpisu informací

4.4 Ukázkový příklad

Ukázkový příklad (učení xor) se spustí pomocí Makefilu příkazem 'make run'. Neuronová síť se na tomto příkladu naučí na trénovacích datech, poté se otestuje na testovacích datech, kde se vypíší výsledky tohoto testování a následně se natrénovaná síť uloží do textového souboru 'w.test', ze kterého může být znova načtena. Program se po uložení vah ukončí.

Reference

[1] Kriesel, D.: A Brief Introduction to Neural Networks, 2005. Dostupné z: http://www.dkriesel.com/_media/science/neuronalenetze-en-zeta2-2col-dkrieselcom.pdf

[2] Přednášky do předmětu Soft Computing. Dostupné z: https://www.fit.vutbr.cz/study/courses/SFC/private/index.html.cz

[3] http://ziyang.eecs.umich.edu/~dickrp/iesr/lectures/widrow90sep-present.pdf

[4] https://www.slideshare.net/infobuzz/adaline-madaline

[5] https://image.slidesharecdn.com/lecture11-090302042359-phpapp01/95/lecture11-neural-networks-8-728.jpg?cb=1235978279