Cvičení: Model umělého neuronu - pokračování

1. Naši implementaci neuronu rozšíříme o další přenosové funkce (lineární, hyperbolický tangens)
2. Implementujeme další učící algoritmy (LSQ, gradientní metoda)
3. Podíváme se na to, jak si dnes implementované algoritmy poradí s Příklady 1-3 z minulé hodiny
4. Příklad 6: neuron s lineární přenosovou funkcí a úloha lineární regrese
5. Ukážeme si, jak vyladit parametry u gradientní metody.
6. Pokud zbyde čas: Jednovrstvá neuronová síť a klasifikace vzorů do více tříd:
   - Rozšíření učících algoritmů pro model neuronové sítě s jednou vrstvou
   - Příklad 4: Písmena.
   - Příklad 5: Ručně psané číslice.

Vytvářené funkce I:

- Nejprve rozšíříme funkce **response** a **responseM** o volitelnou přenosovou funkci

```
function [y, xi] = response(x, w, activation)
function [y, xi] = responseM(X, w, activation)

purelin, tanh
```

- Učení neuronu metodou **LSQ** (lineární regrese)

```
function w = training_LSQ (X, d)
```

- Učení lineárního neuronu **gradientní metodou**

```
function [w, epoch] = training_gradient_descent (X, d, .
    alpha, max_epochs, alpha_adaptive)
```

- Chyba **SSE** (sum squared error)

```
function [e, y] = SSE_error(X, w, d, activation)
```

Vytvářené funkce II:

- Potom gradientní metodu rozšíříme o další parametry:

```
function [w, epoch] = training_gradient_descent (X, d, .
    activation, activation_derivative, alpha, ...
    max_epochs, alpha_adaptive, min_error)
```

- Derivace přenosových funkcí:

```
function y = dtanh(xi)
function y = dlin(xi)
```

Příklady 1, 2, 3 z minula: učení jednoducých logických funkcí

- Nejprve metody pro kontrolu pustíme na jednoduchých datech z minulého cvičení (1a, 1b, 1c)
  Podíváme se na to, kolik epoch stačilo k naučení u gradientní metody a jak vypadají vektory vah v porovnání s metodami z minulého cvičení
- Potom si na úloze 2b ukážeme, jak je u gradientní metody důležitá normalizace vstupních příznaků
- Případně si ukážeme, jak si poradí metody LSQ a gradientní metoda s odlehlými vzory (úloha 3)

Příklad 6: Úloha lineární regrese

- Budeme učit lineární neuron úlohu lineární regrese metodou LSQ a gradientní metodou.
  - nejprve jednorozměrný případ (6a)
  - potom dvorozměrný (6b)
- Budeme experimentovat s nastavením parametrů u gradientní metody a s množstvím šumu v trénovací množině
- vyzkoušíme si, jak vyladit parametry gradientní metody: parametr učení a maximální počet epoch
- vyzkoušíme si využití testovací a validační množiny dat

Zobrazení dat a regresní křivky/nadroviny ve 2D/3D:

```
plot_regression_2d (X, d, w_g);
plot_regression_3d (X, d, w_g);
```

Příklad pro vás (příklad/domácí úkol na účast))

- Modifikujte data 6b (definujte vlastní unikátní lineární funkci s unikátním šumem):
  - vytvořte si vlastní trénovací i testovací množinu dat: d, d_{test} (X, X_{test} můžete zachovat)
- Snažte se vyladit paremetry gradientní metody (především parametr učení, potom třeba maximální počet epoch nebo jiný parametr řídící ukončení učení)
- Nastavení a průběh experimentu zaznamenejte do zprávy (včetně tabulky nebo tabulek s průměrnou chybou na trénovací a testovací množině pro různé nastavení parametrů). Stručně zhodnoťte výsledek experimentu (jaké nastavení parametrů byste doporučili a proč).
- Srovnejte výsledky (průměrnou chybu na trénovací a testovací množině) s metodou LSQ.
- Skript(y), které spustí Váš experiment a zprávu zazipujte a pošlete mi na email. (včetně všech volaných funkcí/skriptů, které jste

Příklad 4: Písmena

- Využijeme připravenou datovou sadu **pismena.mat**
- Písmena byla segmentována z **pismena.png**
- Prohlédněte si datovou sadu a zobrazte si některá písmenka (využijte předpřipravený skript)
- Naučte perceptron pomocí různých algoritmů (a variant) rozpoznávat jednotlivá písmena.
- Určete chybu klasifikace na trénovací množině (popř. i počet epoch / čas učení)
- Určete chybu klasifikace na testovací množině (s přidaným šumem) (viz předpřipravený skript)
- Podívejte se, se kterými písmenky měl perceptron největší problémy.
- Soutěž o nejlepší učící algoritmus :-)

Příklad 5: Ručně psané číslice

- Využijeme připravenou datovou sadu **OcrData.mat** s ručně psanými číslicemi
- Prohlédněte si datovou sadu a zobrazte si některé číslice (využijte předpřipravený skript)
- Naučte perceptron pomocí různých algoritmů (a variant) rozpoznávat jednotlivé číslice.
- Určete (a porovnejte) chybu klasifikace na trénovací množině (popř. i počet epoch / čas učení)
- Podívejte se, se kterými číslicemi měl perceptron největší problémy