Cvičení: Model umělého neuronu - pokračování

Co jsme dělali minule

1. Minule jsme naší implementaci neuronu rozšířili o další přenosové funkce (lineární, hyperbolický tangens) a implementovali jsme pro něj další učící algoritmy (LSQ, gradientní metoda)
2. Nejprve jsme nové algoritmy otestovali, že fungují, na úlohách 1a, XOR.
3. Pak jsme vyzkoušeli učit úlohu lineární regrese na uměle vygenerovaných datech (příklad 6)
4. Na příkladu 6b jsme si ukázali, jak vyladit (hyper)parametry u gradientní metody.
5. Za domácí úkol jste si měli vyzkoušet vyladit parametry sami na pozměněné úloze 6b

Dnešní hodina

1. Diskuse domácího úkolu. Rozšíření testování modelu. Rozšíření gradientní metody o techniku early stopping
2. Ukázka, že je pro gradientní metodu důležitá normalizace vstupních dat (Příklad 2b) a jak se chová v případě odlehlých vzorů (Příklad 3)
3. Jednovrstvá neuronová síť a klasifikace vzorů do více tříd (rozpoznávání vzorů)
   - Implementace modelu neuronové sítě s jednou vrstvou neuronů. Rozšíření algoritmů učení.
   - Příklad 4: Písmena.
   - Příklad 5: Ručně psané číslice.

Příklad 6b: Úloha lineární regrese

- Lineární model neuronu. Učení metodou LSQ nebo gradientní metodou.
- Experimentovali jsme s nastavením parametrů u gradientní metody a s množstvím šumu v trénovací množině
- Vyzkoušeli jsme si, jak vyladit parametry gradientní metody: parametr učení a maximální počet epoch
- Zbývá: testování a porovnávání modelů a využití validační množiny dat

Skript z minulé hodiny:

`run_example6b_regression.m`

Skript z rozšířeným testem o další kritéria:

`run_example6b_test.m`

Příklad z minule (domácí úkol)

- 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 parametry 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 přidali/změnili oproti repozitáři)

Příklad 6b: Úloha lineární regrese

Porovnáváme modely:

- Každý z porovnávaných modelů naučíme vícekrát (např. 100krát) a zaznamenáme si hodnoty testovacích kriterií.
- U každého kritéria nás zajímá průměr (říká, jak je metoda dobrá) a směrodatná odchylka hodnot (říká, jak je metoda stabilní).
- Jaká kritéria nás zajímají?
  - Chyba na trénovací množině dat (jak dobře se model naučil)
  - Chyba na nezávislé testovací množině dat (jak dobře model zobecňuje)
  - Chyba na zašuměné trénovací množině dat (jak moc je model citlivý)
  - Případně další: počet epoch, čas,...

Popř. se dají použít sofistikovanější porovnávací techniky (např. krosvalidace).

Rozšiřujeme program

- Gradientní metodu rozšíříme o ukončení metodou early stopping (s využitím validační množiny dat)

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

- Pozor! : validační množina musí být vždy odlišná od testovací i trénovací množiny.

Příklad 6b: Úloha lineární regrese

Při učení modelu tedy obvykle používáme tři množiny dat:

- trénovací - na nich učíme model
- validační - na základě nich ukončujeme učení modelu (brání přeučení)
- testovací - slouží k vyhodnocení a porovnání modelu s dalšími modely

Příklady 2, 3 z minula

- Na úloze 2b ukážeme, jak je (nejen) u gradientní metody důležitá normalizace vstupních příznaků.
- Pak si ukážeme, jak si poradí metody LSQ a gradientní metoda s odlehlými vzory (úloha 3).

`run_example2b_gradient.m`
`run_example3_gradient.m`

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)

`run_example4.m`

Příklad 4: Písmena - pokračování

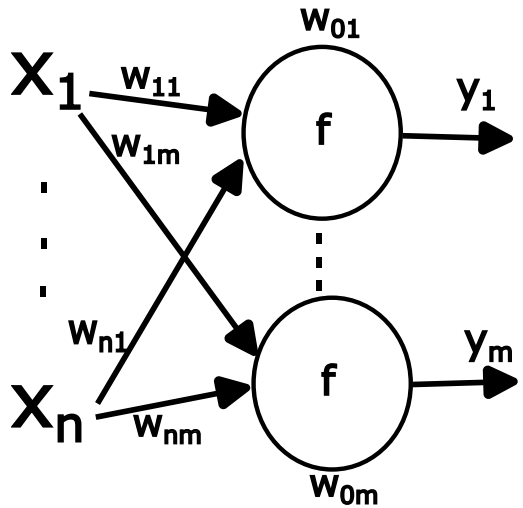
- Naučte perceptron pomocí různých algoritmů (a variant) rozpoznávat jednotlivá písmena.
- Určete chybu klasifikace na trénovací množině (popř. i počet epoch / čas učení)
- Určete chybu klasifikace na testovacích množinách (s přidaným šumem) (viz předpřipravený skript)
- Podívejte se, se kterými písmenky měl perceptron největší problémy.

`run_example4_simple.m`

Užitečné funkce:

`columns_to_classes`
`classes_to_columns`

Jednovrstvá neuronová síť



Rozšiřujeme program:

- Všechny učící algoritmy rozšíříme tak, aby fungovaly i pro jednovrstvou neuronovou síť.
- Díky maticové reprezentaci to půjde poměrně snadno.

```
[e, y] = classification_error_more_classes(X, w, d, ...
 transfer_function)
```

Příklad 4: Písmena - pokračování

- Pak zkusíme naučit jednovrstvou neuronovou síť pomocí různých algoritmů
- U každého algoritmu zkusíme vyladit jeho parametry a nakonec algoritmy navzájem porovnáme.

```
run_example4_finetune
run_example4_compare
```

Užitečné funkce:

```
columns_to_classes
classes_to_columns
```

Příklad 5: Ručně psané číslice

- Využijeme připravenou datovou sadu **OcrData.mat** s ručně psanými číslicemi (jedná se o zjednodušenou a zmenšenou datovou sadu MNIST)
- Prohlédněte si datovou sadu a zobrazte si některé číslice (využijte předpřipravený skript)
- Naučte jednovrstvou neuronovou síť pomocí různých algoritmů (a variant) rozpoznávat jednotlivé číslice.
- Určete chybu klasifikace na testovacích množinách (s přidaným šumem).
- Podívejte se, se kterými číslicemi měly modely největší problémy

`run_example5`

Příklad 5: Ručně psané číslice - úkol za účast

Úkol za účast

- Snažte se vyladit parametry jednotlivých metod a pak je navzájem porovnejte.
- Zhodnoťte výsledek srovnání.
- Najděte matici vah s nejmenší chybou (popř. takovou, která zároveň co nejlépe zobecňuje)

Bude se hodit obdoba skriptu

`run_example4_compare`