7. - 9. cvičení

Vrstevnatá neuronová síť

1. Hrátky s vrstevnatou neuronovou sítí a jednoduchými úlohami (tensorflow playground)
2. Vrstevnatá neuronová síť v Matlabu
   - Vyladění parametrů modelu (počet vrstev a neuronů, parametr učení)
   - Porovnání různých algoritmů učení implementovaných v Matlabu z hlediska rychlosti a chyby
   - Regularizace a další vychytávky
3. Ukázky úloh:
   - Příklad 1: regresní úloha - Body Fat
   - Příklad 2: klasifikace - Iris, Wine
   - Příklad 3: časová řada - sin,...
   - Příklad 4: další hříčky: modifikace obrázků

Přehled dalších ukázkových datových sad:

https://www.mathworks.com/help/deeplearning/gs/sample-data-sets-for-shallow-neural-networks.html

Hezká ukázka vrstevnaté neuronové sítě na různých úlohách':

https://playground.tensorflow.org/

- pět úloh (různě složité, nejtěžší je spirála)
- možnost navolit architekturu a další parametry
- moc pěkné vizualizace, včetně průběhu chyby, velikosti a znamének vah a toho, které příznaky jednotlivé neurony reprezentují
- můžeme experimentovat s tím, kolik vrstev a neuronů stačí na kterou úlohu,...
- **nejtěžší úkol:** naučit spirálu

Vrstevnaté neuronové sítě v Matlabu

Vytvoření modelu:

- regresní úloha ... *fitnet*
- klasifikační úloha ... *patternnet*
- predikce časové řady ... *narnet*, *narxnet*
- obecný model ... ... *feedforwardnet*

net = fitnet([5,5],'traingd');

- první parametr: počty neuronů ve skrytých vrstvách
- druhý parametr: algoritmus učení (zde základní algoritmus zpětného šíření)

Naučení modelu:

[net,tr] = train(net,x,t)

Aplikace modelu:

y = net(x);

Příklad 1 -Body Fat

Ukázky v Matlabu:

- bodyfat_data.m - vizualizace dat
- bodyfat_simple.m - ukázka, jak naučit neuronovou síť
- bodyfat_set_parameters.m - ukázka, jak zobrazit a nastavit parametry neuronové sítě
- bodyfat_compare.m - skript, co umožňuje porovnat různá nastavení parametrů
- test_architectures.m - funkce pro testování architektury
- test_param.m - funkce pro testování parametru učení u traingd
- test_param_mc.m - funkce pro testování parametrů u traingdx
- test_param_reg.m - funkce pro testování parametru regularizace

Příklad 1 -Body Fat - začátek

- Začněme s modelem, co bude mít 12 skrytých neuronů a bude se učit pomocí 'traingda'.
- Nastavte u vytvářené sítě maximální počet cyklů (např. na 3000). Nastavte maximální počet zhoršení na validační množině (např. na 10). Nastavte parametr učení net.trainParam.lr (např. na 0,00001).
- Síť naučte se zapnutým grafickým výstupem a prohlédněte si všechny zobrazené grafy. Rozumíte jim?
- Z jakého důvodu se učení zastavilo? Jaká je výsledná chyba na trénovací, validační a testovací množině?
- Naučila se síť dobře?
- Zkuste skript pustit několikrát. Liší se výsledky?

Příklad 1 -Body Fat - parametr učení a architektura

- Zkuste pro metodu ‘traingd’ vyladit parametr učení a optimální počet neuronů ve skryté vrstvě.
- Bylo za domácí úkol na 7. cvičení
- Ukázka výsledků: cv7_results.xlsx

Příklad 1 -Body Fat - shrnutí domácího úkolu

Pozorování na základě experimentů

1. Čím je větší počet neuronů ve skryté vrstvě nebo čím menší parametr učení, tím se síť učí pomaleji
2. Čím je větší počet neuronů ve skryté vrstvě, tím menší je třeba parametr učení (lr)
3. Malý parametr učení $\rightarrow$ modely se učí pomalu
4. Moc velký parametr učení $\rightarrow$ vede k oscilacím v průběhu chyby, k moc velkým vahám a paralýze
5. Větší počet neuronů než je optimální $\rightarrow$ síť se přeučuje (malá chyba na trénovací množině, ale velká na testovací)
6. Moc malý počet neuronů $\rightarrow$ velký rozptyl chyby i času (někdy se síť naučí velmi dobře, jindy naopak vůbec ne)
7. Optimální (nebo moc velký) počet neuronů $\rightarrow$ malý rozptyl chyby,. větší stabilita
8. Ukázka výsledků: cv7_results.xlsx

Příklad 1 -Body Fat - shrnutí domácího úkolu

Shrnutí experimentů - jak se chovala 'traingd' na datech BodyFat

1. Metoda 'trainbd' potřebuje k naučení poměrně velký počet epoch
2. Je poměrně citlivá na volbu parametru učení (lr).
3. Je méně citlivá na počet neuronů ve skryté vrstvě.
4. Experimentálně jsme dospěli k optimální volbě $lr = 0.0001$ a počet neuronů kolem 5
5. Ukázka výsledků: cv7_results.xlsx

8. cvičení

1. Zhodnocení minulého domácího úkolu
2. Pokračujeme v experimentech s datovou sadou Body Fat
3. Ukázka výsledků: cv7_results.xlsx

Příklad 1 - Body Fat a rychlejší algoritmy učení

- Zkuste i algoritmy učení s momentem a nebo adaptivním parametrem učení (traingdm, traingda, ale především traingdx).
- Nejprve se podívejte na průběh chyby a další grafy. Vidíte nějaké změny oproti 'traingd'?
- Podívejte se, které parametry učení mají tyto metody navíc/jinak oproti traingd - net.trainParam.mc (moment učení) apod.
- Jsou na tyto parametry metody citlivé?
- Zkuste vyladit optimální hodnoty parametru net.trainParam.mc.
- Napomohly tyto metody k rychlejšímu naučení?

Příklad 1 - Body Fat a rychlejší algoritmy učení - shrnutí experimentu

Učení s momentem ('traingdm')

1. Průběh chyby je hladší (nekmitá tolik)
2. Zpočátku o něco rychlejší pokles chyby, ale celkový potřebný počet epoch nesnižuje
3. Optimální volba parametru *mc* pro tuto úlohu je kolem $0.5 - 0.6$

Příklad 1 - Body Fat a rychlejší algoritmy učení - shrnutí experimentu

Učení s adaptivním parametrem učení a momentem ('traingdx')

1. Učení je výrazně rychlejší (dívali jsme se, jak se měnil parametr lr v průběhu učení)
2. Malá citlivost na počáteční volbu parametru učení *lr* i na volbu architektury
3. Pro volbu parametru *mc* kolem $0.5 - 0.6$ nejlepší výsledky
4. Potřebuje jen kolem 80-100 epoch k naučení
5. O něco vyšší chyba než nejlepší u 'traingd', ale stabilní při různé volbě hyperparametrů
6. Ukázka výsledků: cv7_results.xlsx

Příklad 1 -Body Fat a ještě rychlejší algoritmy učení

- Zkuste i další velmi rychlé algoritmy učení (trainlm, trainscg, trainrp).
- Nejprve se podívejte na průběh chyby a další grafy. Vidíte nějaké změny oproti 'traingd' a 'traingdx'?
- Podívejte se, které parametry učení mají tyto metody navíc/jinak oproti traingd
- Jsou na tyto parametry metody citlivé?
- Napomohly tyto metody k lepšímu a rychlejšímu naučení?
- Metody vzájemně porovnejte. Porovnejte je i s metodami traingd, traingdx. Která metoda je podle vás nejlepší?

Příklad 1 - Body Fat a rychlejší algoritmy učení - shrnutí experimentu

Metody druhého řádu a relaxační metoda (trainlm, trainscg, trainrp)

1. Ještě rychlejší učení (10 - 40 epoch v závislosti na metodě, čas u jednotlivých metod srovnatelný)
2. Není třeba ladit další hyperparametry (jako lr a mc)
3. Učí se lépe než 'traingdx'
4. Metoda 'trainlm' byla super-rychlá, ale měla největší problémy s přeučením
5. Metoda 'trainscg' asi byla i nejlepší, jen o málo pomalejší než 'trainlm', ale nepřeučila se

Příklad 1 - Body Fat a rychlejší algoritmy učení - shrnutí experimentu

Shrnutí

1. Pokud nám nezáleží na čase učení a mám čas na dlouhé ladění hyperparametrů, dává nejlepší výsledky vyladěná metoda 'traingd'
2. Pokud nechceme věnovat čas ladění parametrů a chceme, aby se model učil opravdu rychle, volíme 'trainscg' nebo 'trainlm' (konkrétní volba záleží na úloze)
3. Pro hluboké sítě nebo opravdu velká data nejsou 'trainscg' nebo 'trainlm' k dispozici nebo vhodné, pak je nejlepší volba algoritmu s adaptivním parametrem učení

Příklad 1 - Body Fat a regularizace

- Zkuste použít během učení L2-regularizaci:
  - *net.performParam.regularization = 0.5* ... váha chybového členu
  - *trainbr* ... automatická regularizace
- Napomohla regularizace k lepšímu naučení a zobecňování?

→ bylo za domácí úkol na 8. cvičení

Příklad 1 - Body Fat a regularizace - shrnutí experimentu

Učení s L2-regularizací (net.performParam.regularization)

1. U většiny metod regularizace nevedla k výrazně lepším výsledkům
2. Jen u metody trainlm, která měla problémy s přeučením, pomohla

Automatická regularizace (trainbr)

1. Podobné výsledky jako vytuněná traingd, učí se podobně pomalu (sice méně epoch, ale každá epocha trvá dlouho)

9. cvičení

1. Zhodnocení minulého domácího úkolu
2. Shrnutí výsledků nad datovou sadou Body Fat: cv7_results.xlsx
3. Shrnutí učení regresních úloh pomocí vrstevnatých neuronových sítí

nftool - ukázky dalších regresních úloh
https://www.mathworks.com/help/deeplearning/gs/sample-data-sets-for-shallow-neural-networks.html
https://www.mathworks.com/help/stats/sample-data-sets.html
https://www.kaggle.com/
https://archive.ics.uci.edu/

Vrstevnaté neuronové sítě a klasifikace

Příklad 2 - klasifikace (různé datové sady: iris, wine, cancer,...)

- Klasifikační úloha
- ukázky v Matlabu:
  - iris_data.m - vizualizace dat
  - iris_simple.m - ukázka, jak naučit neuronovou síť pro klasifikaci
  - iris_compare.m - skript, co umožňuje porovnat různá nastavení parametrů
  - test_architecture.m_clas - funkce pro testování architektury
- více příkladů: nprtool

Příklad 2 - klasifikace

- Nejprve si zobrazte různé informace o datech
- Pak naučte neuronovou síť se zapnutým grafickým výstupem a prohlédněte si všechny zobrazené grafy. Rozumíte jim?
- Z jakého důvodu se učení zastavilo? Jaká je výsledná chyba na trénovací, validační a testovací množině?
- Naučila se síť dobře? Na rozdíl od regresní úlohy se můžeme podívat i na chybu klasifikace.
- Vyzkoušejte si různé metody učení a podívejte se, jak se pro ně liší průběh chyby a další grafy.
- Úkol: zkuste nalézt pro úlohy iris, wine a cancer optimální parametry neuronové sítě (počet vrstev a neuronů, učící algoritmus). Pomůže regularizace?

Vrstevnaté neuronové sítě a predikce časové řady

Příklad 3 - sin

- Úloha predikce časové řady (sin apod.)
- ukázky v Matlabu:
  - sin_simple.m - ukázka, jak naučit vrstevnatou neuronovou síť pro predikci časové řady
  - sin_narnet.m - ukázka, jak naučit rekurentní neuronovou síť pro predikci časové řady
  - podívejte se, jak se modely naučily na čistých datech a jak na zašuměných datech a zda jim pomůže regularizace
  - vyzkoušejte si nastavení validační a testovací množiny po souvislých časových blocích
  - více příkladů: ntstool

Vrstevnaté neuronové sítě a regularizační hrátky

Příklad 4 - obrázek

- Ilustrace toho, jak může učení ovlivnit ,,regularizace" (zde - rozšíření trénovací množiny o další vzory)
- ukázky v Matlabu:
  - bitmap_simple.m - učení identity
  - bitmap_red.m - preference sytě červené barvy
  - bitmap_red_yellow_violet.m - preference tří sytých barev
- Zkuste si pohrát s trénovací množinou i s počtem neuronů a podívejte se na výsledky.

Vrstevnaté neuronové sítě a regularizační hrátky

Příklad 4 - obrázek

- **Dobrovolná (bonusová) domácí úloha':**
  - Najděte si nějakou vlastní fotografii (obrázek) a upravte některý ze skriptů tak, aby výrazně pozměnil její barevnost (ale jinak, než v původním skriptu).
  - Pozor: pro velký obrázek se síť bude učit dost dlouho
  - Zvolte si svoje vlastní trénovací vzory ,,navíc", popř. vlastní trénovací obrázek
  - Je možné, že si s modelem bude třeba pohrát, aby dával výsledky, které chceme (pro jiný obrázek zřejmě budete muset nastavit jinak i počet neuronů v síti, počet epoch apod.).
  - Pošlete mi pozměněný skript, původní obrázky a několik výsledných obrázků.

Vrstevnaté neuronové sítě v Matlabu

- následuje stručný tahák pro použití Matlabovské knihovny pro vrstevnaté neuronové sítě

Vrstevnaté neuronové sítě v Matlabu

Vytvoření modelu:

- regresní úloha ... *fitnet*
- klasifikační úloha ... *patternnet*
- predikce časové řady ... *narxnet*, *narnet*
- obecný model ... ... *feedforwardnet*

net = fitnet([5,5],'traingd');

- první parametr: počty neuronů ve skrytých vrstvách
- druhý parametr: algoritmus učení (zde základní algoritmus zpětného šíření)

Naučení modelu:

[net,tr] = train(net,x,t)

Aplikace modelu:

y = net(x);

Matlab a vrstevnaté neuronové sítě

Algoritmus zpětného šíření a jeho modifikace I.

- *traingd* ... algoritmus zpětného šíření (BP)
- *traingdm* ... BP s momentem
- *traingda* ... BP s adaptivním parametrem učení
- *traingdx* ... BP s adaptivním parametrem učení a momentem

Pseudonewtonovské metody

- *trainlm* ... Levenberg-Marquardtův algoritmus
- *trainoss* ... Quickprop
- *trainbfg* ... BFG

Matlab a různé metody učení vrstevnaté neuronové sítě

Algoritmus zpětného šíření a jeho modifikace I.

- *traingd* ... algoritmus zpětného šíření (BP)
- *traingdm* ... BP s momentem
- *traingda* ... BP s adaptivním parametrem učení
- *traingdx* ... BP s adaptivním parametrem učení a momentem

Pseudonewtonovské metody

- *trainlm* ... Levenberg-Marquardtův algoritmus
- *trainoss* ... Quickprop
- *trainbfg* ... BFG

Matlab a různé metody učení vrstevnaté neuronové sítě

Algoritmus zpětného šíření a jeho modifikace I. Metody konjugovaných gradientů

- *trainscg* ... Moeller ...Metoda škálovaných konjugovaných gradientů
- *traincgf* ... Fletcher-Reeves
- *traincgp* ... Polak-Ribiere
- *traincgb* ... Powell-Beale

Relaxační metoda

- *trainrp* ... Resilent method

Matlab - implementované techniky pro zlepšení zobecňování

Early stopping a rozdělení dat mezi trénovací, validační a testovací množinu

- lze nastavit *net.divideFcn* a jí příslušné *net.divideParam*
- *dividerand* ... náhodně (*trainRatio*, *valRatio*, *testRatio*)
- *divideint* ... bez přeuspořádání (*trainRatio*, *valRatio*, *testRatio*)
- *divideind* ... indexy zadá uživatel (*trainInd*, *valInd*, *testInd*)

Regularizační techniky - Weight decay

- *net.performParam.regularization* = *0.5* ... váha chybového členu
- *trainbr* ... automatická regularizace

Vrstevnaté neuronové sítě v Matlabu

- Váhy a prahy
  - net.IW{1,1} ... váhy hran mezi vstupní a první skrytou vrstvou... L1 X n
    net.IW{1,1}(j,i) ... váha ze vstupního neuronu i do skrytého neuronu j
  - net.LW{L,K} ... váhy hran ze skryté vrstvy K do vrstvy L
    net.LW{L,K}(j,i) ... váha z neuronu i ve vrstvě K do neuronu j ve vrstvě L
  - net.b ... prahy neuronů
    net.b{L,1} ... prahy neuronů ve vrstvě L (počínaje první skrytou)
    net.b{L,1}(j,1) ... práh j-tého neuronu ve vrstvě L

Vrstevnaté neuronové sítě v Matlabu

Učení se ukončí v jednom z následujících případů

- Byl dosažen maximální počet cyklů (epoch). net.trainParam.epochs (implicitně 1000)
- Byla dosažena chyba na trénovací množině menší než net.trainParam.goal (implicitně 0)
- Gradient klesl pod net.trainParam.min_grad
- Čas učení překročil net.trainParam.time (implicitně Inf)
- Chyba na validační množině vzrostla v net.trainParam.max_fail po sobě jdoucích cyklech (implicitně 6)

Vrstevnaté neuronové sítě v Matlabu

Automaticky přednastavená transformace dat

- Pro vstupní vzory:
  net.inputs{1}.processFcns =
  {'removeconstantrows','mapminmax'}
  vynechají se konstantní řádky, pak se provede minmax normalizace
- Pro výstupní vzory:
  net.outputs{2}.processFcns =
  {'removeconstantrows','mapminmax'}
  vynechají se konstantní řádky, provede se minmax normalizace
- Mohu nastavit i jinak