Y33AUI: Aplikace umělé inteligence

Domácí cvičení
Středa 10.11.2010

½ b. 1. Vaše jméno: _____

½ b. 2. Přidělená definice cílových tříd: _____
Např. 1 1 2 1 2 3 4 4 3 4

Otázka:	1	2	3	4	5	6	7	8	9	Celkem
Body:	½	½	0	0	0	1	0	1	½	3½
Srážky:										

Otázka:	10	11	12	13	14	15	16	17	18	19	20	21	Celkem
Body:	2	½	1	0	2	0	2	0	3	2	1	3	16½
Srážky:													

Cílem samostatné práce je porovnat 3 modely pro statistické rozpoznávání (MLP, KNN a SVM) na úloze klasifikace objektů do několika tříd (na 2 datových sadách). Vaším úkolem je vytvořit skript, který na 1 datové sadě porovná

- vícevrstvý perceptron (MLP) s 1, 2, 5, 10, 20, 50, 100, 200 a 500 jednotkami ve skryté vrstvě,
- metodu *k* nejbližších sousedů (kNN) s 1, 2, 5, 10, 20, 50, 100, 200 a 500 sousedy a
- support vector machine (SVM) s RBF jádrem o velikosti 1, 2, 5, 10, 20, 50, 100, 200 a 500

a přehledně vytiskne klasifikační přesnost každého z těchto 27 modelů na trénovací a testovací sadě dat.

Informace o práci:

- Domácí úloha obsahuje 21 otázek a lze za ni získat 20 bodů.
- Na vypracování máte 5 dní, přesný termín odevzdání (den, hodina, minuta) najdete v aplikaci Upload system.
- Odevzdávat budete 2 části:
  - skript, který jste použili k vypracování úlohy, spolu se všemi potřebnými funkcemi do aplikace Upload system,
  - dokument s odpověď mi na otázky uvedenými v tomto zadání (buď elektronickou verzi do Upload systému nebo papírovou verzi přímo cvičícímu).
- Pro některé podúlohy nadefinujeme funkce a ze skriptu je budeme volat. Udržujte funkce i skript jednoduché, budete-li vytvářet vlastní funkce, volte vhodné názvy, z nichž bude zřejmé, co funkce dělají. Nebojte se komentářů.
- U úkolů, kde je uvedeno místo na odpověď, se od vás očekává textová odpověď. Můžete ji zapsat přímo do tohoto dokumentu. Pokud se rozhodnete odpovědi psát do nového dokumentu (třeba pro elektronické odevzdání), vždy uveďte číslo otázky, ke které se odpověď vztahuje. Velikost rámečků naznačuje, jak dlouhá odpověď se od vás očekává.
- U úkolů, kde místo na odpověď není, se většinou očekává, že vytvoříte kus skriptu nebo funkci. Nezapoměňte je proto všechny odevzdat do Upload systému.
- Buďte struční a pište čitelně. Hodně štěstí!

1 Seznamte se s daty

Máte k dispozici soubory `optdigits.train`, `optdigits.test` a `optdigits.names`; prostudujte především poslední z nich a odpovězte na následující otázky:

0 b. 3. Kolika proměnnými je popsán objekt, tj. kolik vstupů budou modely mít?

0 b. 4. Jakých hodnot mohou jednotlivé vstupy nabývat?

0 b. 5. Do kolika kategorií budeme chtít objekty zařazovat?

1 b. 6. Vytvořte funkci `designClasses`, která z původních 10 tříd (číslice 0 až 9) vytvoří pro vaše konkrétní zadání úlohy nové třídy. Vstupem bude vektor délky n, výstupem bude taktéž vektor délky n. Pokud vaše zadání bude např. 1 2 3 4 4 4 5 5 6 7, bude tato funkce pro jednotlivé prvky vektoru realizovat následující mapování:

Stará třída	0	1	2	3	4	5	6	7	8	9
Nová třída	1	2	3	4	4	4	5	5	6	7

Nezapomeňte tuto funkci odevzdat společně se skriptem do Upload systému!

2 NETLAB

Stáhněte si a rozbalte toolbox Netlab. Zopakujte si z minulých cvičení, co dělají funkce `mlp`, `netopt` a `mlpfwd`. Nastudujte si také funkci `confmat` (viz `help confmat`; dejte pozor, aby to byla skutečně funkce `confmat` z NETLAB toolboxu, a věnujte pozornost tomu, v jakém tvaru očekává své parametry, pokud se jedná o klasifikaci do více tříd).

0 b. 7. Co ukrývá hodnota `RATE(1)` po zavolání `[C, RATE] = CONFMAT(Y, T)`?

3 Načtení 1. datové sady

Vezměme soubor `optdigits.train` s trénovacími daty. Vytvořte matici `trin`, která bude obsahovat vstupní proměnné z trénovacích dat, a matici `trout`, která bude obsahovat výstupní proměnné trénovacích dat. Transformujte `trout` prostřednictvím vaší funkce `designClasses`.

Obdobně načtěte testovací data ze souboru `optdigits.test`. Vytvořte matice `tstin` a `tstout`; matici `tstout` tranformujte pomocí `designClasses`.

4 MLP

Budeme používat více než dvě výstupní kategorie a v takovém případě je vhodné použít síť s více než jedním výstupem. Každý výstup sítě pak bude představovat diskriminační funkci jedné třídy. Musíme proto umět přetransformovat výstupní proměnnou `trout` na několik binárních výstupních proměnných a následně zase z výstupů sítě zjistit predikovanou třídu.

1 b. 8. Vytvořte funkce `class2indicator` a `indicator2class` s následujícími hlavičkami:

```
function ind = class2indicator(cls, Ncls)
function cls = indicator2class(ind)
```

Specifikace:

- `cls` je matice $[N \times 1]$ obsahující identifikátor třídy pro N objektů.
- `Ncls` je číslo udávající počet tříd.
- `ind` je matice $[N \times Ncls]$ obsahující na každém řádku pouze 1 jedničku, jinak samé nuly. Na i. řádku bude jednička vždy na pozici `cls(i)`. V případě funkce `indicator2class`, která bude sloužit k převodu výstupu sítě na indentifikátory tříd, mohou být v matici `ind` reálná čísla, ovšem největší z nich by mělo indikovat predikovanou třídu.

Nezapomeňte funkce odevzdat do Upload systému.

1/2 b. 9. Na základě vektoru `trout` vytvořte pomocí funkce `class2indicator` matici `trouti`. Jakým způsobem příkaz zavoláte? Jaké budou rozměry výsledné matice `trouti`?

2 b. 10. Vytvořte funkci s následující hlavičkou:

```
function [trAcc, tstAcc] = trainAndTestMLP(trin, trout, tstin, tstout, nhidden)
```

které předáte trénovací a testovací data a počet neuronů ve skryté vrstvě a která na základě těchto informací síť naučí na trénovacích datech a vypočítá přesnost sítě na trénovacích a testovacích datech. Vyzkoušejte např. pro 10 neuronů ve skryté vrstvě.

- Jako výstupní nelin. funkci použijte `softmax`.
- Jako optimalizační algoritmus použijte `scg`.
- Počet iteraci nastavte na cca 200.
- K výpočtu přesnosti použijte funkci `confmat`.

1/2 b. 11. Stručně popište, jak vypadá tzv. matice záměn (confusion matrix). Pokud má nějakou význačnou strukturu, pokuste se ji několika slovy vysvětlit.

1 b. 12. Na základě matice záměn porovnejte úspěšnost sítě pro testovací data s údaji získanými pro trénovací data. Stručně vysvětlete.

0 b. 13. Obalte volání funkce `trainAndTestMLP` ve vašem skriptu cyklem s dalšími potřebnými příkazy a zjistěte trénovací a testovací přesnost sítě pro 1, 2, 5, 10, 20, 50, 100, 200 a 500 neuronů ve skryté vrstvě. Doplňte následující tabulku:

	Počet neuronů ve skryté vrstvě								
	1	2	5	10	20	50	100	200	500
Klas. přesnost, trén. data [%]									
Klas. přesnost, test. data [%]									

5 KNN

K modelování pomocí nejbližších sousedů využijeme funkce `knnrule()` a `knnclass()` ze STPR toolboxu. Stáhněte si jej a nainstalujte. Zopakujte si, v jakém formátu požaduje data.

2 b. 14. Vytvořte funkci s následující hlavičkou:

```
function [trAcc, tstAcc] = trainAndTestKNN(trin, trout, tstin, tstout, neighbors)
```

které předáte trénovací a testovací data a počet nejbližších sousedů, které má metoda použít. Funkce na základě těchto informací vytvoří model z trénovacích dat a vypočítá přesnost sítě na trénovacích a testovacích datech. Vyzkoušejte např. pro 10 sousedů.

- K výpočtu přesnosti použijte funkci `confmat` z NETLAB toolboxu. Dejte pozor, abyste funkci předali data ve správném formátu.

0 b. 15. Obalte volání funkce `trainAndTestKNN` ve vašem skriptu cyklem s dalšími potřebnými příkazy a zjistěte trénovací a testovací přesnost modelu pro 1, 2, 5, 10, 20, 50, 100, 200 a 500 sousedů. Doplňte následující tabulku:

	Počet sousedů								
	1	2	5	10	20	50	100	200	500
Klas. přesnost, trén. data [%]									
Klas. přesnost, test. data [%]									

6 SVM

K modelování pomocí support vector machine využijeme funkce `bsvm2()` a `svmclass()` ze STPR toolboxu.

2 b. 16. Vytvořte funkci s následující hlavičkou:

```
function [trAcc, tstAcc] = trainAndTestSVM(trin, trout, tstin, tstout, rbfsize)
```

které předáte trénovací a testovací data a velikost RBF jádra. Funkce na základě těchto informací vytvoří model z trénovacích dat a vypočítá přesnost sítě na trénovacích a testovacích datech. Vyzkoušejte např. pro velikost jádra 10.

- Použijte SVM s RBF jádrem. (`options.ker`)
- Nastavte správně parametr jádra. (`options.arg`)
- Nastavte konstantu C (váhu, s jakou vám záleží na správné klasifikaci vzhledem k maximalizaci marginu) na 100. (`options.C`)
- Nastavte maximální počet iteraci např. na 20000. (`options.tmax`)
- K výpočtu přesnosti použijte funkci `confmat` z NETLAB toolboxu. Dejte pozor, abyste funkci předali data ve správném formátu.

0 b. 17. Obalte volání funkce `trainAndTestSVM` ve vašem skriptu cyklem s dalšími potřebnými příkazy a zjistěte trénovací a testovací přesnost modelu pro velikost jádra 1, 2, 5, 10, 20, 50, 100, 200 a 500. Doplňte následující tabulku:

	Velikost jádra								
	1	2	5	10	20	50	100	200	500
Klas. přesnost, trén. data [%]									
Klas. přesnost, test. data [%]									

7 Diskuze výsledků

3 b. 18. Okomentujte výsledky uvedené v tabulkách pro MLP, kNN a SVM.

2 b. 19. Jak byste vybrali nejlepší model a proč?

8 Jiná data

Nyní byste měli mít hotov skript, který spočte trénovací a testovací přesnost tří různých typů modelů s různým nastavením (celkem 27 modelů) na jistých datech. Skript nyní zkusíme aplikovat na jinou datovou sadu. Trénovací data jsou v souboru `optdigits2.train`, testovací jsou v souboru `optdigits2.test`.

1 b. 20. Spusťe skript na nových datech a vyplňte následující tabulky:

MLP:

	Počet neuronů ve skryté vrstvě								
	1	2	5	10	20	50	100	200	500
Klas. přesnost, trén. data [%]									
Klas. přesnost, test. data [%]									

KNN:

	Počet sousedů								
	1	2	5	10	20	50	100	200	500
Klas. přesnost, trén. data [%]									
Klas. přesnost, test. data [%]									

SVM:

	Velikost jádra								
	1	2	5	10	20	50	100	200	500
Klas. přesnost, trén. data [%]									
Klas. přesnost, test. data [%]									

3 b. 21. Co můžete říct o výsledcích modelů na 2. datové sadě v porovnání s výsledky dosaženými na 1. sadě? Pokud se nějak liší, zkuste vysvětlit proč. Jistou nápovědu vám dají matice záměn pro trénovací a testovací data.

Konec domácího cvičení.