FAKULTA APLIKOVANÝCH VĚD ZÁPADOČESKÉ UNIVERZITY

Vylepšení úspěšnosti klasifikace příznaků algoritmem „Nejbližšího souseda“

Vladimír Láznička

20. 11. 2015

Obsah

Zadání

Cílem této práce je využít metodiky pro vylepšení přesnosti (nebo chcete-li výkonu) algoritmu *K Nearest Neighbors* (dále *KNN*) pomocí tzv. *adaptivní míry vzdálenosti* a zjistit, jaký vliv na přesnost bude metodika mít. Tato úprava je vysvětlena a popsána v článku [1] (anglicky) nebo českém výtahu z tohoto článku [2] (ten je jako předchůdce tohoto dokumentu ostatně čtenáři doporučen).

Problém klasifikace příznakových vektorů pomocí algoritmu *KNN* spočívá mimo jiné v možném **prolínání shluků vektorů ve stavovém prostoru**, což často vede k neúspěšné klasifikaci v hraničních případech. Cílem zmíněné úpravy je **omezení vlivu některých vektorů** určených jako trénovací, které se mohou nacházet poblíž nebo mezi vektory s jinou klasifikační třídou a negativně ovlivňovat klasifikační proces. Problematika je podrobněji vysvětlena v článcích [1] a [2], nebude zde tedy dále rozebírána.

Dokument této práce je dále rozdělen na *Návrh řešení*, kde je popsán princip funkce adaptivní míry vzdálenosti pro konkrétní klasifikační program včetně uvedení pseudokódu. Další sekcí je *Řešení implementace*, kde se nachází několik informací o zvolené platformě pro implementaci, stručný popis potřebných funkcí programu a rozepsaný způsob používání výsledného programu (zejména informace týkajících se vstupů a výstupů programu). Následuje sekce *Experimenty a výsledky*, která popisuje způsob měření přesnosti *KNN* algoritmu s a bez použití adaptivní míry vzdálenosti a samozřejmě uvádí naměřené výsledky. Zhodnocení výsledků a zvážení kladů a záporů použité úpravy algoritmu se pak nachází v sekci *Závěr*.

Návrh řešení

Zde je nastíněno řešení pro následnou implementaci *adaptivní míry vzdálenosti* pro algoritmus *KNN*. Je zde popsán pouze princip činnosti této úpravy a nikoliv celého *KNN* algoritmu – jeho popis, vlastnosti a pseudokód lze nalézt v dokumentu [2]. Rovněž zde není rozebírána *adaptivní míra vzdálenosti* z matematického hlediska, neboť toto je rovněž pokryto v dokumentu [2].

Princip činnosti adaptivní míry vzdálenosti

Použití adaptivní míry vzdálenosti spočívá v nastavení jistého **poloměru nebo sféry vlivu jednotlivým vektorům určených jako trénovací** (tedy ta množina vektorů, podle které se klasifikuje třída vektorů z množiny testovacích/neznámých). Tento poloměr se vyjadřuje jako **vzdálenost vektoru k jinému nejbližšímu vektoru s odlišnou klasifikační třídou**. Čím dále ve stavovém prostoru je konkrétní vektor umístěn od vektorů s jinými třídami, tím větší hodnotou tohoto poloměru disponuje. Hodnota poloměru je pak využívána při samotné klasifikaci, kdy se vypočítávají vzdálenosti jednotlivých trénovacích vektorů od klasifikovaného vektoru. **Vypočítaná vzdálenost se poté vydělí hodnotou poloměru** a použije se jako berná vzdálenost při určování nejbližších trénovacích vektorů, podle kterých

se následně určí třída klasifikovanému vektoru. **Trénovací vektory s vyšší hodnotou poloměru vlivu se tedy jeví blíže,** než kdyby nebyl tento poloměr použit a má potenciálně vyšší šanci být označen jako jeden z „nejbližších sousedů“.

Je třeba také poznamenat, že vzdálenostní funkce (Euklidovská, Manhattanská…) použitá k výpočtu poloměru by **měla korespondovat se vzdálenostní funkcí použitou při výpočtu vzdálenosti** mezi trénovacími vektory a vektorem klasifikovaným. V opačném případě může docházet k nekonzistenci výsledků.

Další důležitou vlastností algoritmu pro adaptivní míru vzdálenosti je ta, že celý výpočet a přiřazení poloměrů jednotlivým trénovacím vektorům **probíhá před samotnou fází klasifikaci** (tedy při trénování), tudíž prakticky nezatěžuje klasifikaci, co se rychlosti týče.

Pseudokód

```
BEGIN
      FOR EACH instance1 IN trenovaciMnozina
      DO
            polomer := INFINITE;
            FOR EACH instance2 IN trenovaciMnozina
            DO
                  IF(instance1.trida != instance2.trida) THEN
                        vzdalenost := vypoctiVzdalenost(instance1,
                        instance 2);
                        IF (vzdalenost < polomer) THEN polomer :=
                        vzdalenost;
            LOOP

            instance1.polomer := polomer;
      LOOP
END
```

Řešení implementace

Pro vyzkoušení a ověření *adaptivní míry vzdálenosti* pro *KNN* algoritmus bude zapotřebí implementovat program, který má být především schopen **načíst množinu trénovacích vektorů s jejich třídami a množinu testovacích vektorů se správně určenými třídami** (aby bylo možné provést měření úspěšnosti klasifikace). Mezi další vlastnosti programu pak má patřit možnost uživatelsky měnit, **jakou vzdálenostní funkci bude využívat a zda má či nemá použít adaptivní míru vzdálenosti**. To nám zajistí možnost změřit vliv této úpravy v kombinaci s různými vzdálenostními funkcemi. Jako posledním důležitým parametrem pro nastavení je **počet nejbližších sousedních trénovacích vektorů použitých ke klasifikaci** (číslo `K`).

Jako vývojářská platforma bude použit jazyk **Java SE 1.8** především kvůli již implementovaným strukturám, které se budou hodit pro rychlé napsání programu, přičemž vzhledem k poměrně omezenému rozsahu práce není pro pohodlné testování klasifikace

zapotřebí nějakých významných optimalizací ve využívání paměti a výpočetního času. Dalším důvodem je pak možnost spustit výsledný program jako **JAR** archiv na libovolné platformě (Windows, Linux…) bez nutnosti řešit zvlášť kompilační záležitosti. Výsledný program pak samozřejmě bude vyžadovat instalaci **JRE ve verzi 8** nebo novější.

Výsledný program (uživatelská dokumentace)

Výsledný program – `KNN_Classifier` – je distribuován jako **soubor JAR** spustitelný z příkazové řádky operačního systému následujícím příkazem (je zapotřebí mít správně nastavené proměnné prostředí systému s uvedenou cestou k JRE):

```
java -jar KNN_Classifier.jar <soubor1> <soubor2> <K> <vzdFunk>
<pouzAdapt>
```

Všechny vyjmenované parametry **jsou nutné pro běh programu**, pokud se uživatel pokusí spustit program s jiným počtem argumentů než 5, bude vrácena chybová hláška popisující způsob užití. Ukázka použití programu v příkazové řádce je na obrázku [Figure 1].

Parametr `<soubor1>` je cesta k souboru (String, pokud se v řetězci vyskytují mezery, je třeba jej uzavřít do uvozovek), **který uchovává množinu trénovacích vektorů.** Parametr `<soubor2>` zase představuje cestu k souboru s množinou vektorů, **které budou klasifikovány (testovací vektory)**. Oba soubory musí dodržet specifický formát, kdy **na první řádce obsahují počet vektorů (vzorů), které se v souboru vyskytují**, a **na dalších řádkách jednotlivé vektory** v následující podobě:

```
<složka vektoru 1>,<složka vektoru 2>,……;<třída vektoru>
```

Jednotlivé složky vektoru jsou odděleny pomocí **čárek** a za celým vektorem je **třída vektoru oddělená od vektoru středníkem**. Složky vektoru mohou být uvedeny jako **celá nebo desetinná čísla** (jako desetinný oddělovač se používá tečka), třída vektoru pak může být **obecně jakýkoliv řetězec** obsahující jakékoliv znaky s výjimkou oddělovacích znaků řádky s vektorem (čárka a středník). U vektorů pro trénování představuje třída jejich skutečnou třídu, podle které se provádí klasifikace, u vektorů pro testování se jedná o správnou třídu, vůči které se pak porovnává výsledek klasifikace.

Parametr `<K>` pak představuje **počet nejbližších sousedních trénovacích vektorů použitých pro klasifikaci**. Musí být zadán jako **celé číslo menší než je počet trénovacích vektorů.**

Parametr `<vzdFunk>` označuje, **jaká vzdálenostní funkce se má použít při výpočtech vzdálenosti** – *Euklidovská* nebo *Manhattanská*. **Jedná se o celé číslo**, kde `1` znamená použití *Euklidovské* funkce a `2` použití *Manhattanské* funkce. Pokud je zvoleno jakékoliv jiné číslo, bude defaultně použita *Euklidovská* funkce.

Parametr `<pouzAdapt>` značí, **zda má být využito *adaptivní míry vzdálenosti*** pro trénink klasifikátoru a následnou klasifikaci. Opět se jedná o **celočíselný parametr**, kdy `1` značí použití úpravy, zatímco jakékoliv jiné číslo ji zakáže.

Důležité je, aby všechny parametry dodržely předpokládaný typ a soubory byly ve správném formátu, jinak běh programu skončí výpisem chyby a bude třeba jej opět spustit.

Pokud program proběhne správně, vypíše hlášku „`Classification finished.`“ A vytvoří soubor se **stejným jménem jako soubor s testovacími vektory** a příponou `.result`, kde jsou obsažené testovací vektory s klasifikací určenou třídou a na poslední řádce vypsaná procentuální úspěšnost klasifikace (výsledek měření). Řádky s vektory mají stejný formát jako původní soubor, pouze se zde nacházejí dvě třídy – první je přidělená klasifikací a druhá je korektní třída pro kontrolu (z testovacího souboru). Pokud uživatel zvolil použití adaptivní míry vzdálenosti, bude za běhu programu také vypsána hláška „`Using ADM`“.

```
C:\Users\Vlada47\Dokumenty\GitHub\PRO_semestralka>java -jar KNN_Classifier.jar i
ris.trainingset iris.testingset 5 1 1
Using ADM
Classification finished.
Program exiting.
```

Figure 1 - Příklad použití programu

Projekt programu, zdrojový kód s `javadoc` dokumentací (v anglickém jazyce) i testovací data lze najít na stránkách verzovacího systému **Git**, kde je vše volně přístupné [3].

Experimenty a výsledky

Testy úspěšnosti klasifikace probíhaly na celkem **3 množinách vektorů získaných z reálných dat**. Konkrétně se jednalo o datasety *Iris* [4], *Breast Cancer* [5] a *Ionosphere* [6]. Pokud má čtenář zájem dozvědět se o těchto datech více informací, lze nahlédnou přes příslušné odkazy na stránky **UCI Machine Learning Repository**. Všechny sety (jejich vektory a třídy) **byly upraveny, aby vyhovovaly formátu, ve kterém je testovací program dokáže číst**.

Měření probíhalo následovně – z každého datasetu se náhodně vybrala přibližně 1/5 vektorů, které sloužily jako testovací (nechala se u nich klasifikovat třída), zatímco zbytek byl využit pro trénování. Každý dataset byl pak testován pro kombinace nastavení *Euklidovské* a *Manhattanské* vzdálenostní funkce a použití nebo zakázání *adaptivní míry vzdálenosti*. **Pro každý dataset tedy proběhly celkem 4 různá nastavení**, pro něž dále měření proběhlo při těchto hodnotách parametru `K` – **1, 3, 5, 7, 9, 15, 21, 31, 41 a 51**. Takto bylo možné sledovat vliv různých nastavení pro různé počty „nejbližších sousedů“.

Dataset *Iris* se skládá celkem ze **150 vektorů, ze kterých bylo použito 30 jako testovacích**. Vektory zde disponují celkem **4 dimenzemi** a mohou nabývat celkem **3 různých tříd**. Výsledky z jednotlivých měření jsou zachyceny na grafech [Figure 2] a [Figure 3].

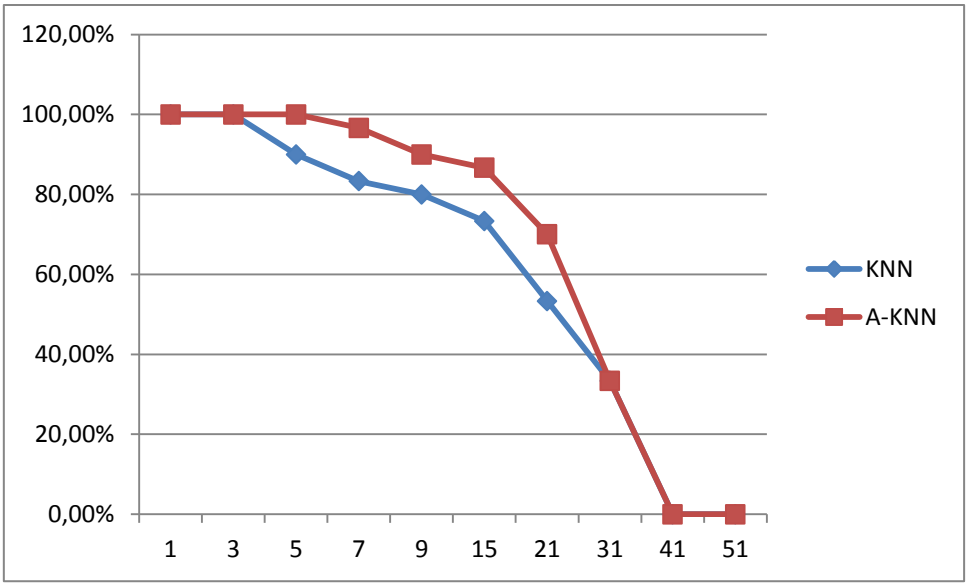

Figure 2 - Porovnání výsledků KNN (modře) a KNN s použitím adaptivní míry vzdálenosti (červeně); set Iris; použitá Euklidovská vzdálenostní funkce

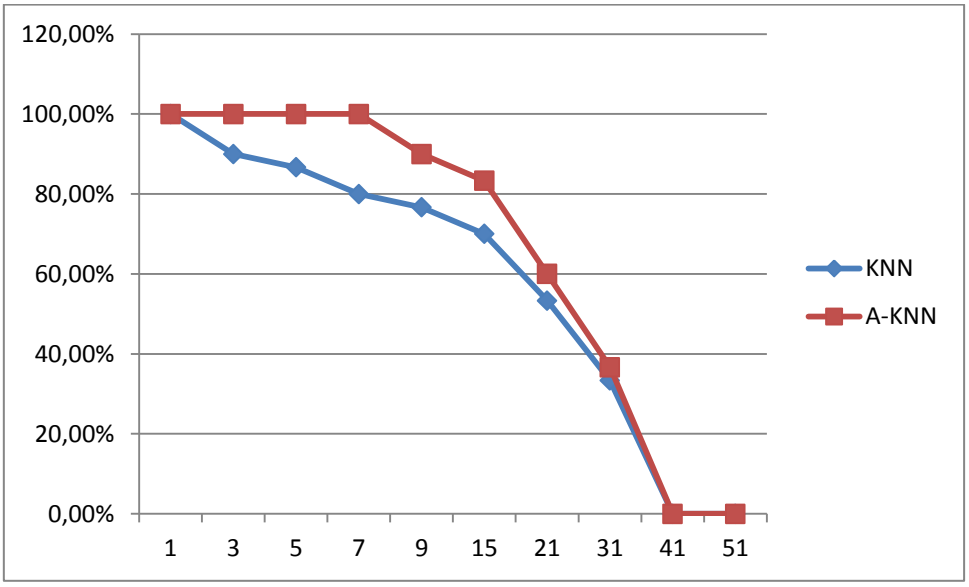

Figure 3 - Porovnání výsledků KNN (modře) a KNN s použitím adaptivní míry vzdálenosti (červeně); set Iris; použitá Manhattanská vzdálenostní funkce

Dataset *Breast Cancer* se skládá celkem z **599 vektorů, ze kterých bylo použito 112 jako testovacích**. Vektory zde disponují celkem **32 dimenzemi** a mohou nabývat celkem **2 různých tříd** (jedná se o binární klasifikační problém). Výsledky z jednotlivých měření jsou zachyceny na grafech [Figure 4] a [Figure 5].

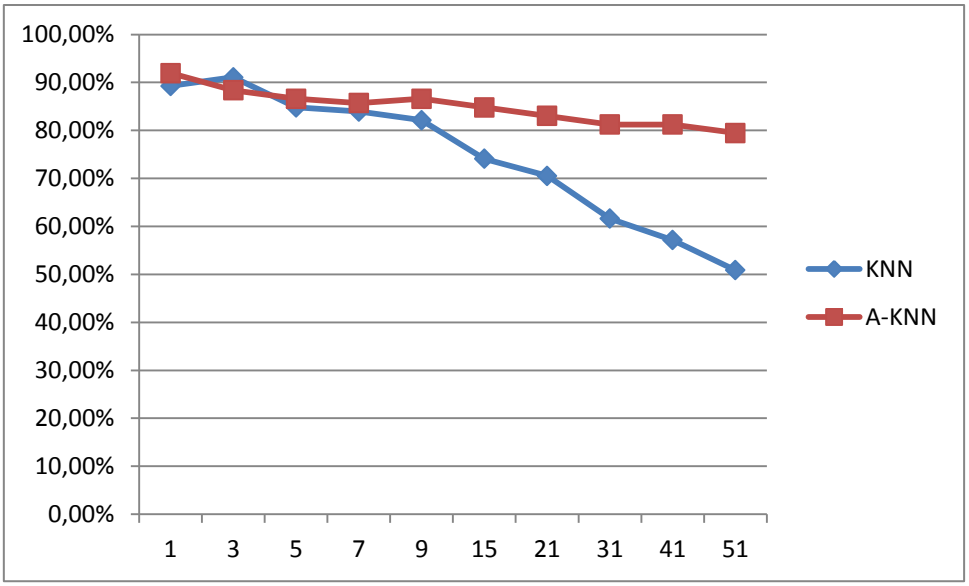

Figure 4 - Porovnání výsledků KNN (modře) a KNN s použitím adaptivní míry vzdálenosti (červeně); set Breast Cancer; použitá Euklidovská vzdálenostní funkce

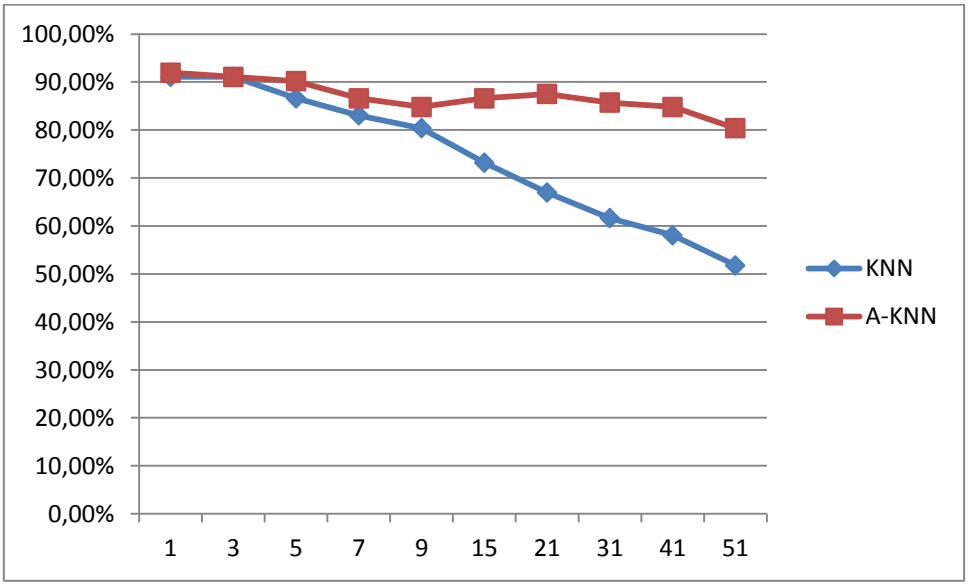

Figure 5 - Porovnání výsledků KNN (modře) a KNN s použitím adaptivní míry vzdálenosti (červeně); set Breast Cancer; použitá Manhattanská vzdálenostní funkce

Dataset *Ionosphere* se skládá celkem z **351 vektorů, ze kterých bylo použito 70 jako testovacích**. Vektory zde disponují celkem **34 dimenzemi** a mohou nabývat celkem **2 různých tříd** (jedná se o binární klasifikační problém). Výsledky z jednotlivých měření jsou zachyceny na grafech [Figure 6] a [Figure 7].

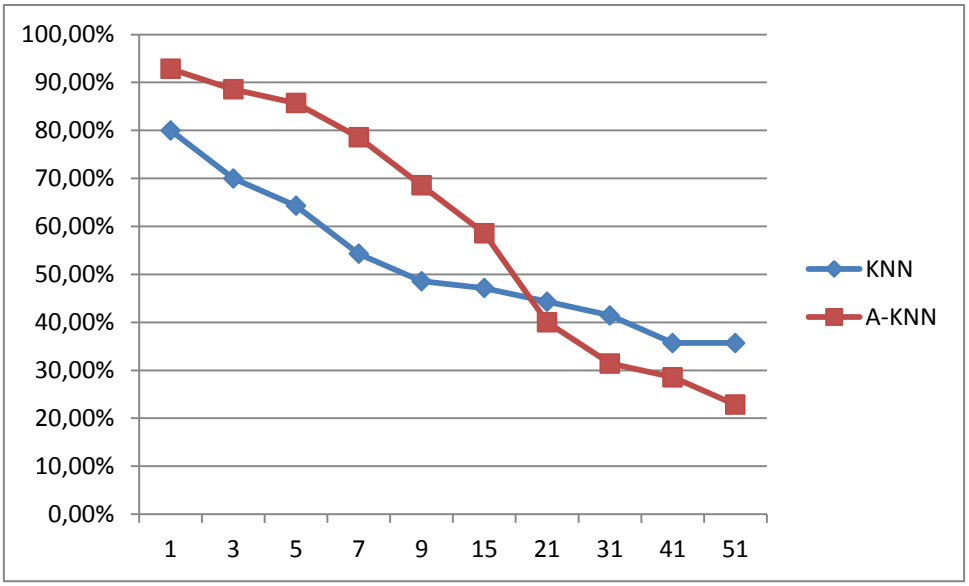

Figure 6 - Porovnání výsledků KNN (modře) a KNN s použitím adaptivní míry vzdálenosti (červeně); set Ionosphere; použitá Euklidovská vzdálenostní funkce

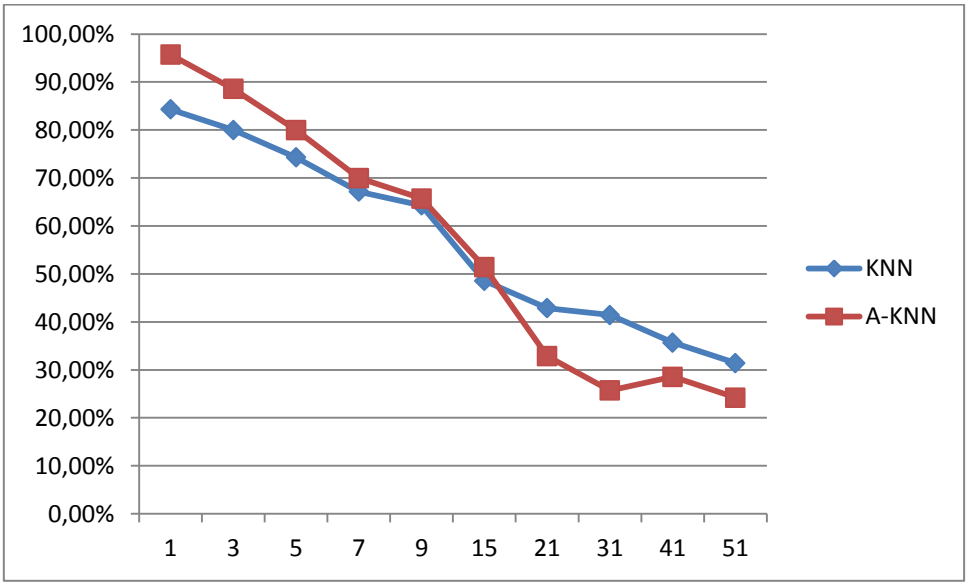

Figure 7 - Porovnání výsledků KNN (modře) a KNN s použitím adaptivní míry vzdálenosti (červeně); set Ionosphere; použitá Manhattanská vzdálenostní funkce

Jak lze vysledovat z grafů, použití *adaptivní míry vzdálenosti* přináší pro různé datasety v závislosti na nastavení parametrů klasifikace různé výsledky. U datasetu *Iris* je přínos úpravy v závislosti na použité vzdálenostní funkci zhruba **do 10%** do hodnoty `K` **21 až 31**, pro vyšší hodnoty nebyl přínos prakticky žádný (přesnost klasifikace u těchto hodnot klesla až k 0%). U datasetu *Breast Cancer* byla situace odlišná – přínos se s rostoucí hodnotou `K` rovněž zvětšoval (byť přesnost klasifikace algoritmu jako takového pozvolna klesala). Rozdíl zde dosáhl **takřka 30%** pro nejvyšší hodnotu `K`. U datasetu *Ionosphere* se částečně opakovala situace jako u datasetu *Iris* – při nižších hodnotách `K` (zhruba **do 15**) podával algoritmus s použitím *adaptivní míry vzdálenosti* lepší výsledky, poté se nicméně situace otočila. Můžeme se zde rovněž všimnout většího přínosu při použití *Euklidovské* vzdálenostní funkce, je to způsobeno tím, že *Manhattanská* funkce podávala pro *KNN* algoritmus lepší

výsledky. Rozdíl ve výsledcích mezi upraveným a neupraveným *KNN* algoritmem byl při použití *Euklidovské* funkce **až zhruba 25%**.

Závěr

Z výsledků měření je patrné, že adaptivní míra vzdálenosti může potenciálně **přinést i výrazně lepší přesnost klasifikace v závislosti na jejích parametrech a samozřejmě také konkrétních datech** (výběr vektorů pro trénování i testování). U dvou ze tří datasetů byly zisky v přesnosti **lepší pro menší hodnoty** parametru `K`, což v podstatě odpovídá výsledkům naměřeným v článku [1]. U datasetu *Breast Cancer* však docházelo k lepší přesnosti **zejména pro vyšší hodnoty** parametru `K`, přičemž rozdíl byl místy poměrně značný. Celkově vzato – použití *adaptivní míry vzdálenosti* je určitě přínosná alternativa k běžnému *KNN* algoritmu a může pomoci vyřešit situaci, kdy se část trénovacích vektorů různých tříd prolíná ve stavovém prostoru. Výhodou úpravy je **prakticky nulové zvýšení výpočetní náročnosti při samotné klasifikaci**, veškeré nastavení potřebných hodnot proběhne při trénování klasifikátoru. Roli zde mohou hrát vyšší nároky na paměť, neboť každý trénovací vzor musí mít uloženou hodnotu poloměru svého vlivu, ale navýšení velikosti datových struktur nebude nijak výrazné (zvlášť pro vektory s velkým počtem dimenzí). Jako možné vylepšení klasifikace by šla zvážit kombinace s dalšími metodami ovlivňování vlivu vzdálenosti od testovaného vektoru nebo metodami pro ohodnocování důležitosti jednotlivých dimenzí vektorů.

Bibliografie a odkazy

1. Jigang Wang, Predrag Neskovic and Leon N. Cooper / ELSEVIER Pattern Recognition Letters, Volume 28, Issue 2, 15 January 2007, Pages 207–213 (http://www.sciencedirect.com/science/article/pii/S0167865506001917)
2. Vladimír Láznička, Vylepšení algoritmu „Nejbližšího souseda“ pomocí jednoduché adaptivní míry vzdálenosti, říjen 2015
3. Projekt této práce pro vývojové prostředí Eclipse – zdrojový kód, tabulky s přesnými výsledky a dodatečná dokumentace (https://github.com/Vlada47/PRO_semestralka)
4. UCI Machine Learning Repository, dataset Iris (https://archive.ics.uci.edu/ml/datasets/Iris)
5. UCI Machine Learning Repository, dataset Breast Cancer (http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29)
6. UCI Machine Learning Repository, dataset Ionosphere (https://archive.ics.uci.edu/ml/datasets/Ionosphere)