VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

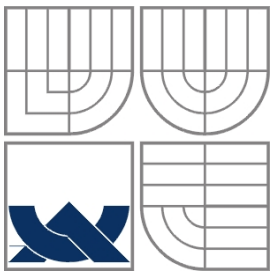
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

KLASIFIKACE PŘÍSPĚVKŮ VE WEBOVÝCH DISKUSÍCH

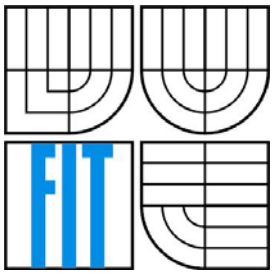
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. Tomáš Margold
AUTHOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

KLASIFIKACE PŘÍSPĚVKŮ VE WEBOVÝCH DISKUSÍCH

CLASSIFICATION OF WEB FORUM ENTRIES

DIPLOMOVÁ PRÁCE

AUTOR PRÁCE
AUTHOR

Bc. Tomáš Margold

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. Radek Burget, Ph.D.

BRNO 2008

Abstrakt

Diplomová práce se zabývá klasifikací textu v prostředí internetu. Jsou v ní popsány dostupné moderní metody pro klasifikaci a následné rozdělení textových příspěvků. Součástí diplomové práce je implementace Bayesova naivního algoritmu a klasifikátoru s využitím neuronových sítí. Vybrané metody jsou zde porovnány vzhledem k jejich chybovosti či jiným klasifikačním vlastnostem.

Klíčová slova

Bayesův naivní algoritmus, neuronové sítě, Backpropagation, diskuzní fórum, klasifikace

Abstract

This thesis is dealing text ranking on the internet background. There are described available methods for classification and splitting of the text reports. The part of this thesis is implementation of Bayes naive algorithm and classifier using neuron nets. Selected methods are compared considering their error rate or other ranking features.

Keywords

Naive Bayes algoritm, Neural network, Backpropagation, forum entries, classifiers

KLASIFIKACE PŘÍSPĚVKŮ VE WEBOVÝCH DISKUSÍCH

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Radka Burgeta, Ph.D.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Tomáš Margold
6.5.2008

Poděkování

Rád bych na tomto místě poděkoval Ing. Radku Burgetovi, Ph.D. za vynikající vedení mé práce i za trpělivost, kterou se mnou měl.
Rád bych také poděkoval mamince Haně Margoldové za jazykovou korekturu práce.

Obsah

1 Úvod

Internet je interaktivní, sociální, komunikačně propojující médium. Je to celosvětově rozšířená počítačová síť, poskytující informační služby téměř ve všech oblastech lidské činnosti. Slouží k poskytování mnoha služeb, jako jsou elektronická pošta, chat, www stránky, sdílení souborů, vyhledávání, katalog a další. Jako prostředek komunikace je levný, rychlý a dostupný a během krátké chvíle dokáže svým způsobem přiblížit člověka jednomu, ale i současně několika lidem kdekoli na světě.

Díky lepší dosažitelnosti a klesající ceně bude ve velice krátké době trvalou součástí českých domácností, tak jako je tomu dnes u televizní obrazovky. Dalším důvodem jeho rozšiřování je skutečnost, že je internet, primárně určený pro práci a získávání informací, v mnoha případech prožíván jako prostředek zábavy.

Toto masivní rozšíření s sebou přináší kromě pozitiv i jistá negativa. Jedním z nich je vliv užívání internetu na lidskou psychiku. V médiích se setkáváme s nejrůznějšími, často protichůdnými zprávami, vyjadřujícími se k této otázce. Je to užitečná komunikace, neškodná zábava, nebo ohlupující zlo? Různí se i názory odborníků v oboru psychologie. Ti se shodují v názoru, že záleží na individuálním přístupu jedince k této problematice. Postupné rozšiřování internetu však přináší neustálé snižování věkové hranice, kdy se děti a mládež dostávají do prvního kontaktu s tímto fenoménem. A právě zástupci této věkové kategorie jsou nejvíce ohroženi.

Komunikační možnosti internetu totiž dávají prostor k navazování nových kontaktů a vztahů. Komunikace po internetu je méně osobní, komunikující nevidí jeden druhého a oba mohou zůstat v naprosté anonymitě. To se ale vývojem internetu a jeho komunikačních prostředků ukázalo jako dvojsečná zbraň. Výhodou anonymity je jistě odstranění všech komunikačních bariér, jako jazykových vad apod., předcházení předsudků a stereotypů. Na druhou stranu tento pocit dokonalé anonymity vede často ke ztrátě kontroly nad svým jednáním. Ta se může projevovat nezdvořilostí, vytahováním se, lhaním, nebo vulgárností. Právě vulgarita je nejznámějším problémem anonymity na internetu. S největší pravděpodobností k ní přispívá pocit uživatele komunikačního kanálu o jeho „neohroženosti“. Tato skutečnost se stává závažným problémem v kombinaci s již zmíněným zvyšujícím se volným přístupem dětí a mládeže do neomezeného prostředí internetu.

Původním cílem mé práce bylo vytvoření aparátu pro detekci vulgárních, či jinak nevhodných příspěvků v těchto komunikačních kanálech. Řešením by bylo vytvoření dostatečně mocného klasifikačního aparátu, který by bylo možné implementovat do funkčního diskusního fóra. Klasifikační model by se nejdříve musel "naučit" rozpoznávat příspěvky dle definovaných pravidel. Následně by pak mohl označit "nevhodný" vzkaz a nabídnout příslušnou akci. Jednou z variant by bylo vymazání vzkazu,

který by byl označen jako nevhodný. Další možností by bylo například upozornění operátora diskuze o tomto příspěvku, zákaz přístupu autora vzkazu k dalšímu vkládání příspěvků atd.

Pochopením základních znalostí o klasifikačních algoritmech se moje cíle pro tuto práci rozšířily. Z definice klasifikačního modelu totiž vyplývá, že algoritmus nemusí dělit příspěvky pouze do dvou tříd "nevhodný příspěvek" a "nezávadný příspěvek". Většina klasifikačních algoritmů umí obecně klasifikovat prvky do *n* kategorií. Využitím tohoto faktu lze vytvořit obecný klasifikátor, použitelný například pro studentské diskusní fórum. Při implementaci je nutné nadefinovat konkrétní kategorie pro klasifikaci. Těmi by mohli být například "studentské aktivity", "zkoušky a hodnocení studentů" či "stravování v menze".

V první části mé práce se zaměřím na teoretické pochopení problému. Ve druhé kapitole uvedu data z historie a vývoje klasifikačních algoritmů. Zároveň nastíním motivační faktory a příklady aplikace klasifikačních modelů v současnosti.

Třetí kapitolu věnuji obecnému modelu klasifikace. Budu v ní definovat matematický model klasifikace a nastíním princip konstrukce klasifikátoru. Uvedu důvody nutnosti přípravy textu před jeho zpracováním a jejich základní metody. Malou, ale důležitou část kapitoly věnuji klasifikaci textu.

Pravděpodobně nejznámějším klasifikátorem je Bayesův naivní algoritmus, který popíšu ve čtvrté kapitole. Definovaný model bayesovské klasifikace demonstruji na experimentu hodu mincí. Ten navíc následně ověřím v jazyce Matlab.

Další kapitolu věnuji teorii neuronových sítí. Popíšu nejjednodušší neuronovou síť – perceptron a na něm přiblížím jejich problematiku. Tato metoda je předmětem implementace.

Praktická část diplomové práce bude věnována implementaci a porovnání zvolených algoritmů. Pokusím se porovnat obě implementované metody na základě několika testů (např. z hlediska chybovosti či generalizace algoritmu).

2 Historie a současnost klasifikace

2.1 Historické mezníky klasifikace

Počátky klasifikace jako vědního oboru, jak jej chápeme v dnešní době, se datují do třicátých let dvacátého století. Za průkopnickou lze označit práci The Genetical Theory of Natural Selection [12] pojednávající o klasifikaci druhového jména rostlin rodu kosatců do tří tříd na základě znalosti hodnot čtyř znaků. Tento postup se dnes nazývá Fisherova diskriminační analýza a použitá data (známá jako Iris data) dodnes patří k uznávanému fondu testovacích dat pro testování nových klasifikačních procedur. Odlišný statistický přístup nabízejí teorie bayesovské statistiky, skrytých markovských řetězců a odhadování hustot.

Další impuls pro rozvoj oboru klasifikace má nematematický základ, který vzešel ze snahy vědců pochopit a precizně popsat funkční děje lidského mozku, konkrétněji práci neuronu jako základního stavebního kamene celé nervové soustavy. Tato myšlenka se pod názvem perceptron dočkala matematického popisu v pracích Rosenblatta shrnutých v monografii Rosenblatt (1962), kde byly položeny teoretické základy modelů, známých jako neuronové sítě (neural networks). Jiný model neuronových sítí zvaných Madaline navrhl nezávisle na Rosenblattovi Widrow a Hoff (1960) [13]. V šedesátých letech nastala na jistou dobu stagnace ve výzkumu i aplikacích neuronových sítí. Zájem teoretiků i koncových uživatelů klasifikačních metod o neuronové sítě oživil tzv. algoritmus zpětného řízení (back-propagation algorithm) v polovině osmdesátých let.

Využití struktur stromového typu přivítali jak specialisté strojového učení, kteří již tyto struktury používali k jiným účelům, tak statistici pro možnost užití CARTu pro klasifikaci i regresi. Uživatelé klasifikačních postupů ze strany lékařů, demografů a jiných oborů oceňovali snadné konstruování procedury a následnou interpretaci jejích výsledků, která byla možná i pro laika. Metoda CART tak otevřela komunikační bránu mezi vědci a veřejností, toužící po možnosti klasifikaci uplatnit na problémy reálného života.

Pohled na klasifikaci diskriminační analýzou R. A. Fishera, [12] jako na hledání co nejlepších lineárních hranic, které od sebe v prostoru oddělují jednotlivé třídy, vedl V. Vapnika [8] k úspěšnému zobecnění této myšlenky o „nelinearitu“. Nové procedury známé jako Support Vector Machine (dále jen SVM) se staly „hitem“ mnoha aplikačních oborů. Přes překvapivě dobré výsledky, snahu zakladatele myšlenky V. Vapnika i práci ostatních na poli teorie SVM, je výzkum této metody stále ve svých počátcích s nutností rozšiřovat teoretický základ a testovat metodu v praxi.

2.2 Aplikace klasifikace v současnosti

O některých příkladech použití klasifikace jsem se zmínil již na začátku této kapitoly. Oblasti použití klasifikačních postupů jsou však daleko širší a postupem doby nabývají na významnosti.

První aplikace se objevily v oboru biologie a zemědělství. Dle různých znaků se klasifikovaly rostliny a zemědělské plodiny. Zajímavé uplatnění našla klasifikace v antropologii. Při vykopávkách mohli vědci podle nalezených předmětů určit, ke kterému typu patří kostry pravěkých lidí, viz Anděl (1985) [8].

Opravdový rozmach však začal až s nástupem počítačových technologií. Ty měly vliv především na zvýšení rychlosti zpracování dat, ale také na zlepšení jejich dostupnosti. Umožňovaly jejich snazší shromažďování, ukládání a přístup k nim. V době, kdy bankovní ústavy začaly zpracovávat data elektronicky a uchovávaly záznamy o transakcích klientů, začaly také využívat těchto dat při rozhodování o poskytnutí úvěrů či vydání kreditních karet. Podobnými metodami se také odhalovaly podezřelé finanční transakce.

Velký význam má klasifikace v medicíně. Používá se pro stanovení diagnózy pacienta, určování rizikových skupin pacientů atd. Využívá se jednak strukturovaných dat uložených v databázi, ale také např. obrazových materiálů z RTG, CT (počítačová tomografie), ultrazvuku apod. Podniky, zabývající se prodejem výrobků (např. maloobchodní prodejní řetězce) nebo poskytováním služeb (např. telekomunikační společnosti), dnes v značné míře shromažďují data a informace o svých zákaznících. Ta potom využívají např. k vytváření nabídek služeb „na míru“ či k lepšímu rozmístění zboží v regálech obchodů tak, aby se různým typům zákazníků „lépe a radostněji“ nakupovalo.

Důležitou oblastí, ve které jsou klasifikační metody významně zastoupeny, je vyhledávání a rozpoznávání informací. Již dnes se s úspěchem používají systémy rozpoznávání řeči (automatický telefonní operátor, ovládání elektronických zařízení hlasem), systémy rozpoznávání psaného textu (digitalizace tištěných materiálů, ovládání elektronických zařízení perem), systémy pro strojový překlad nebo systémy vyhledávání informací v digitálních knihovnách (textových, obrazových i zvukových).

3 Klasifikace

Proces klasifikace je možno definovat jako zařazení daného objektu do jisté skupiny na základě jeho vlastností. Téměř vždy se skládá ze dvou kroků:

1. učení: tvorba klasifikačního modelu, schopného klasifikovat data pomocí trénovacích dat (vzorků dat, u nichž známe výsledek klasifikace, tj. třídu, do které patří)

2. vlastní klasifikace: použití modelu pro klasifikaci nových dat (jejich zařazení do tříd).

3.1 Matematický model klasifikace

Uvažujme situaci, kdy na jednotlivých objektech měříme *p* znaků, které mohou být jak spojitého, tak kategoriálního typu. Výsledky každého měření lze zapsat pomocí p-rozměrného vektoru:

$$\boldsymbol{x} = (x_1, \ldots, x_p)^T \in \mathfrak{X} = \mathfrak{X}_1 \otimes \ldots \otimes \mathfrak{X}_p$$

Složka x_i , $i = 1, \ldots, p$; je prvkem prostoru $\mathfrak{X}_i \subseteq \mathbb{R}^1$, který se nazývá stavový prostor. Mějme *K* tříd, které pro větší přehlednost označme čísly *1, 2,... K* a zaveďme množinu tříd $G = \{1, \ldots, K\}$. Předpokládejme dále, že každý objekt patří právě do jedné z uvažovaných tříd, a tudíž je zcela charakterizován dvojicí *(k; x)* tak, že *k* náleží do *G* a *x* náleží do *X*.

V reálných situacích se často stává, že informace o třídě objektu je neznámá nebo obtížně získatelná. Cílem klasifikace je sestrojit rozhodovací (klasifikační) pravidlo, které systematickou cestou umožní předpovídat (predikovat), do které třídy ten který objekt patří. Matematicky je klasifikační pravidlo definováno jako zobrazení *d* ze stavového prostoru *X* do množiny tříd *G*:

$$\begin{aligned} d : \mathfrak{X} &\to \mathcal{G} \\ \boldsymbol{x} &\mapsto k \end{aligned}$$

3.2 Pravděpodobnostní model klasifikace

Pro zobrazení *d* budu v celé diplomové práci používat kratšího pojmenování klasifikátor. Považuji za důležité, že libovolný klasifikátor si lze ekvivalentně představit jako rozklad prostoru *X* na *K* navzájem disjunktních podmnožin A_k takových, že:

$$\begin{array}{l} \text{i)}\ A_k := \{\boldsymbol{x} \in \mathfrak{X}; d(\boldsymbol{x}) = k\}, \\ \text{ii)}\ \mathfrak{X} = \bigcup_{k=1}^{K} A_k,\ A_k \cap A_l = \emptyset,\ \text{pokud}\ k \neq l. \end{array}$$

Matematický model klasifikace, který byl zaveden výše, nepostačuje přirozenému požadavku porovnávat skutečnou třídu objektu s předpovězenou třídou. Pro libovolný objekt je možné pouze konstatovat shodnost či neshodnost třídy skutečné a předpovězené.

Pro principiálně jiný pohled na klasifikaci objektů lze matematický model rozšířit na model pravděpodobnostní. Rozšíření modelu spočívá v zavedení následujících náhodných veličin a vektorů:

1) Zaveďme náhodnou veličinu *G*, popisující výskyt jednotlivých tříd k náležících do $\mathcal{G}$. Rozdělení *L(G)* veličiny *G* je diskrétní a je charakterizováno pravděpodobnostmi:

$$\pi_k \equiv p_G(k) := \mathsf{P}[G = k],\ k \in \mathcal{G}.$$

Od pravděpodobností π_k požadujeme, aby splňovaly:

$$\begin{array}{l} \text{(a)}\ \forall k \in \mathcal{G}\ \pi_k > 0, \\ \text{(b)}\ \sum_{k=1}^{K} \pi_k = 1. \end{array}$$

2) Na stavovém prostoru *X* zaveďme p-rozměrný náhodný vektor $\boldsymbol{C} = (C1, \ldots, Cp)^T$. Budeme předpokládat, že rozdělení vektoru $\boldsymbol{C}$ je charakterizováno hustotou *pC(x)* : Poznamenejme, že pro každé *i* z intervalu *i = 1 ... p*; může být náhodná veličina *Ci*, popisující chování znaku i jako spojitého nebo kategoriálního typu.

V dalším textu budu potřebovat i sdružené rozdělení vektoru $(G;\boldsymbol{C}^T)^T$, resp. podmíněné rozdělení *G* při daném $\boldsymbol{C}$ a podmíněné rozdělení $\boldsymbol{C}$ při daném *G*. Označme si hustoty jednotlivých náhodných veličin (vektorů) $(G;\boldsymbol{C}^T)^T$, $(G \mid \boldsymbol{C})$ a $(C \mid G)$ po řadě $p_{G,\boldsymbol{X}}(k, x)$, $p_{G|x}(k \mid x)$ a $p_{C|G}(x \mid k)$. Pro podmíněné hustoty $p_{G|C}$ (k | x) budu využívat větu o podmíněné hustotě, viz např. Anděl (1985) [8], s jejíž pomocí se hustoty dají vyjádřit vztahem:

$$p_{G|\boldsymbol{X}}(k|x) = \begin{cases} 0 & \text{pokud}\ p_{\boldsymbol{X}}(x) = 0 \\ \dfrac{p_{G,\boldsymbol{X}}(k,x)}{p_{\boldsymbol{X}}(x)} & \text{jinak} \end{cases}$$

3.3 Princip konstrukce klasifikace

Po zavedení pravděpodobnostního modelu lze pomocí klasifikátoru *d* definovat náhodnou veličinu *d(C)*. Pravděpodobností model přirozeným způsobem nabízí kritérium optimálního klasifikátoru. V pravděpodobnostním modelu je cílem najít takový klasifikátor, aby pravděpodobnost *P [d(C) = G]* byla co nejblíže k jedné.

3.4 Omezení modelu

Model klasifikace byl definován proto, aby umožnil použití matematických a pravděpodobnostních postupů. Klasifikační úloha se nezabývá otázkou, jaké znaky se mají měřit a jak při definici úlohy vybrat stavový prostor X. Ten je dán zadáním klasifikační úlohy. Na základě vyřešení klasifikační úlohy nalezením optimálního klasifikátoru bude pravděpodobně možné z prvotně zadaného stavového prostoru zvolit znaky podstatné pro klasifikaci. Eliminujeme tím méně podstatné znaky, jejichž vynecháním při konstrukci klasifikátoru se kvalita predikce významně nesníží. Potom můžeme doporučit, aby se tento znak nesledoval.

3.5 Klasifikace textu

Jak jsem již naznačil v předchozím odstavci, nemusí být klasické vyhledávání dostatečné. V oblasti dolování z textů byla proto vyvinuta celá řada metod, využívajících algoritmy strojového učení. Nejčastěji se jedná o tzv. klasifikátory, tj. systémy, přiřazující příkladům některou z předem definovaných tříd. Existuje celá řada typů klasifikátorů a metod pro jejich tvorbu. Mohu zmínit např. pravidlové systémy, rozhodovací stromy, neuronové sítě, Bayesovské klasifikátory, či v poslední době velmi populární SVM (Support vector machine).

Klasifikátory jsou vytvářeny z množiny tzv. učících příkladů, kterým je přiřazena některá z předem definovaných tříd. Tato množina je většinou vytvořena expertem v dané oblasti. Jako příklad klasifikační úlohy v textech mohu zmínit úlohu, při které je třeba rozhodnout, jaké konkrétní živelné katastrofy popisují texty. Jako třídy je možné definovat například „povodně“, „vichřice“, „požáry“ a „sucha“. Z množiny příspěvků o živelných katastrofách vybereme část, ve které přiřadíme každému příspěvku jednu třídu.

Metoda využívající vytvoření množiny dat expertem přímo nabízí možnost vzniku veřejných databází. Tyto databáze obsahují předpřipravená data, která již byla někým protříděna. Nejznámější databází takto vytvořených tradičních kolekcí jsou např. tzn. Reuters kolekce [14].

Po vytvoření (nebo použití již připravené) kolekce již můžeme použít některý algoritmus strojového učení, který vytvoří klasifikátor (označovaný také jako model). Podoba modelu je závislá na zvolené metodě učení a reprezentaci dat. V případě pravidlového klasifikátoru a prostých textů je modelem sada pravidel tvaru: „pokud je splněna podmínka (text obsahuje nějakou kombinaci slov) potom dokument patří do dané třídy“. Příkladem může být pravidlo, které říká, že pokud je v textu slovo „oheň“, „lesy“ a „škoda“, potom tento text popisuje živelnou katastrofu ze třídy „požáry“. Stejně jako u vyhledání dokumentů je potřeba hodnotit kvalitu vytvořeného klasifikátoru. K tomu lze využít metriky „přesnost“ a „pokrytí“, častěji se ale setkáváme s alternativní definicí míry přesnosti (označuje se jako accuracy). Ta udává, kolik z celkového počtu dokumentů bylo zařazeno do správné třídy.

3.6 Příprava dat pro klasifikaci

V mnoha případech jsou v podnikových databázích a datových skladech ukryty informace o dosud neznámých zajímavých závislostech mezi sledovanými veličinami a jevy. Před samotným započetím klasifikace je třeba data upravit do takové formy, aby mohli klasifikační algoritmy vrátit požadovaná data. V další části práce uvedu základní možnosti pro úpravu vstupních dat.

3.6.1 Čištění dat

Zejména u velkých a neúplných souborů dat je nutná jejich úprava tak, aby byly vhodné k dalšímu zpracování. Na rozdíl od předzpracování, u něhož je nastavení příslušných parametrů obvykle součástí iterativního procesu, se u čištění dat obvykle jedná o jednorázovou aplikaci určitého algoritmu. Jedná se o operaci, při které se odstraňuje „šum“ v datech, například odstranění hodnot s nesmyslnou hodnotou či upravení dat s hodnotou chybějící (např. s využitím regresní analýzy).

Předpokládejme například databázovou tabulku, do které byla data nasbírána prostřednictvím webového formuláře. Takto zadané informace mohou být často neúplné, nesmyslné či dokonce chybějící. Dalším důvodem pro nekorektně vyplněnou tabulku může být například špatná funkce automatického sběrače dat atd.

Některé z dolovacích algoritmů často vyžadují, aby všechny tyto informace byly vyplněny ještě předtím, než započnou svoji činnost. Dále uvedu některé možnosti, jak se můžeme vypořádat s neúplností dat v tabulkách.

3.6.1.1 Ignorace řádku s chybějící hodnotou

Jde o pravděpodobně nejjednodušší cestu, jak dosáhnout úplnosti dat v tabulce. Řádky, které obsahují nějaká prázdná pole či nesmyslná data, jsou při zpracování vynechány. Tato metoda je vhodná zejména v případě, kdy chybějících hodnot není převážná část zdrojové databázové tabulky. Při čištění dat totiž může snadno dojít k případu, že nám v tabulce určené pro dolování nezůstanou skoro žádná použitelná data. Ignorace řádku s chybějící hodnotou může být navíc zautomatizována.

3.6.1.2 Doplnění prázdných polí jinou hodnotou

Další možností je manuální nahrazení chybějící hodnoty jinou hodnotou na základě znalostí dat v tabulce. Pokud například chybí informace o měření tělesné teploty pacienta, lze tuto informaci doplnit na základě měření provedených v jiné dny. Výhodou této metody je fakt, že tabulka neztratí chybějícími daty poškozené řádky. Nevýhodou by mohlo být zkreslení informační hodnoty při větším počtu takto ovlivněných řádků.

3.6.1.3 Zavedení příznaku pro chybějící hodnoty

Chybějící data v buňkách tabulky lze nahradit i speciálním příznakem, který bude při pozdějším zpracování ignorován. Na využití této možnosti řešení se pak váže problém volby globální konstanty. Tato hodnota by totiž měla být z domény příslušného atributu, což není vždy snadné.

3.6.1.4 Použití průměrné hodnoty

Jednou z metod pro doplnění chybějící informace je vložení průměrné hodnoty daného sloupce. Vkládaná hodnota nemusí být nutně matematickým průměrem. Výhodou této metody je fakt, že nedochází ke ztrátě řádků, ani informací. Největší nevýhodou je náročnější zpracování, popř. zisk průměrné hodnoty. Ta totiž nemusí být pouze číselného charakteru.

3.6.2 Významnostní analýza

Ve vstupním souboru dat mohou být často data, která nejsou pro konkrétní klasifikační úlohu relevantní. Výstupem významnostní analýzy by mělo být odstranění takto nepotřebných dat.

3.6.3 Transformace dat

Transformací dat rozumíme úpravu vstupních dat a jejich následný převod. Typicky jde například o zobecnění dat. Nejčastějším jevem je transformace číselných hodnot na diskrétní. Číselná hodnota, udávající výši výplaty zaměstnance, tak může být převedena na slovní údaj (malý / velký).

Speciálním příkladem transformace je normalizace dat. Operace je často užívána k převodu libovolných hodnot do intervalu <0, 1>.

3.7 Porovnání klasifikačních metod

3.7.1 Přesnost předpovědí

Přesnost klasifikačního modelu je základním a pravděpodobně nejdůležitějším aspektem porovnání. Vyjadřuje míru schopnosti dobře třídit neznámá data (tzn. data, na která model nebyl trénován). Přesnost je nejčastěji definována jako poměr počtu správně klasifikovaných příspěvků *Σki* a počtu všech příspěvků *Σkn*.:

$$\theta = \frac{\Sigma k_i}{\Sigma k_n}$$

3.7.2 Složitost

Složitost určuje výpočetní složitost pro vygenerování a používání klasifikačních pravidel. Nejčastěji zkoumáme dva typy složitostí. Časovou složitost určuje počet kroků algoritmu provedených od počátku do konce výpočtu. Druhým hlediskem složitosti je složitost prostorová. Ta je dána počtem paměťových buněk požadovaných pro daný výpočet.

Složitost můžeme zároveň měřit ve třech různých případech. Každý případ udává zcela jinou hodnotu, ačkoli jde o stejný algoritmus. Analýzu složitosti můžeme rozlišit na

- analýzu složitosti nehoršího případu
- analýzu složitosti nejlepšího případu
- analýzu složitosti průměrného případu

V případě obecné klasifikace je nejpřesnějším hlediskem složitosti analýza průměrného případu. Ta je definována následovně:

Jestliže algoritmus (TS) vede k m různým výpočtům (případům) se složitostí c_1, c_2... c_m, jež nastávají s pravděpodobností p_1, p_2...p_m, pak průměrná složitost algoritmu je dána jako $\sum_{i}^{n} {}_{1} p_i c_i$.

3.7.3 Robustnost

Robustnost je schopnost klasifikačních metod vytvořit správný model, pokud daná data obsahují šum a chybějící hodnoty.

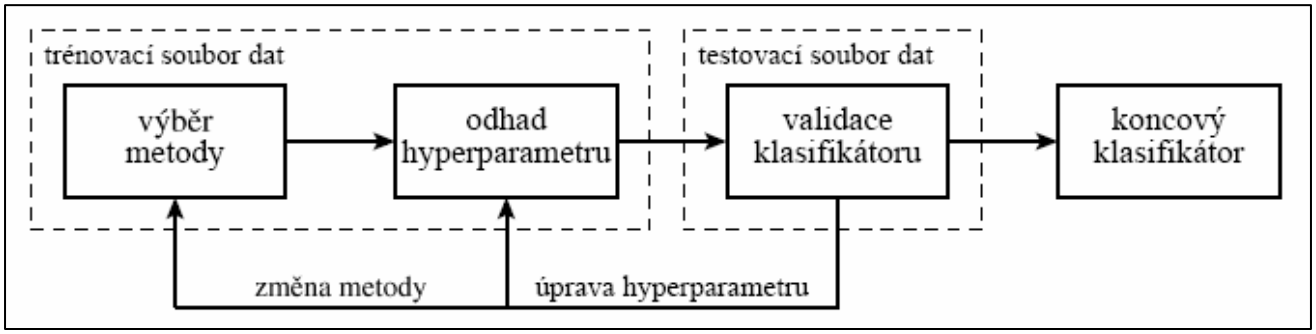
3.7.4 Stabilita

Stabilita klasifikačního modelu určuje míru schopnosti vytvořit správný model pro velké množství dat.

4 Proces učení

Zatím jsem se snažil popsat klasifikační úlohu z teoretického hlediska a ukázal jsem, jak lze při znalosti příslušných rozdělení znaků a tříd sestrojit optimální klasifikátor. V reálných klasifikačních úlohách často informace o rozděleních přirozeně chybí. Stojíme tedy před problémem, jak se k optimálnímu klasifikátoru alespoň přiblížit, resp. jak se při klasifikaci co nejlépe rozhodovat, máme-li omezený zdroj informací. Často chceme ověřit i kvalitu „reálného“ klasifikátoru pomocí odhadu pravděpodobnosti chybné predikce. Jako přirozený požadavek se přitom jeví oddělení „konstrukce“ klasifikátoru z dat od následného „měření“ jeho kvality. Abychom mohli výsledky vícero klasifikátorů na stejném souboru reálných dat porovnávat (i zpětně), potřebujeme metodologicky stanovit jakousi platformu, tj. souhrn přesně stanovených pravidel, podle kterých budeme při vzájemném porovnávání různých klasifikátorů vždy postupovat. Jednou z možných platforem je dále popsaný proces učení.

Proces učení je postup skládající se ze dvou částí: trénování a testování. První část lze chápat jako soubor kroků, které vedou k výběru konkrétního klasifikátoru, splňujícího předem zvolená kritéria. Ve druhé části, která je na první nezávislá, se zkoumá kvalita zkonstruovaného klasifikátoru. Schéma procesu je znázorněno na následujícím obrázku:



Obr. 1: Schéma procesu učení [2]

Obě části probíhají na souborech reálných dat, které jsou složeny z realizací náhodného vektoru C a náhodné veličiny G . Souborem dat o velikosti n se rozumí množina:

$$\{(x_1, g_1), \ldots, (x_n, g_n)\},$$

ve které je i-tý objekt reprezentován dvojicí (x_i, g_i), kde $x^T_i = (x_{i1}, \ldots, x_{ip})$. Data realizací vektoru C se

zanášejí do matice X typu n krát p:

$$\mathsf{X} = \begin{pmatrix} x_1^T \\ \vdots \\ x_n^T \end{pmatrix}$$

Prvek matice x_{ij} je tedy roven j-té složce realizace x_i, tzn. $x_{ij} = x_{ij}$. Hodnoty realizací veličiny G se zanášejí do n-rozměrného vektoru $G^T = (g_1, \ldots, g_n)$. V procesu učení se často nepoužívá jeden, ale dva soubory dat. První se nazývá trénovací a druhý testovací. Trénovací a testovací soubory musí splňovat podmínku, že žádný objekt není zastoupen v obou souborech. Podmínka zaručuje nezávislost testovací části na trénování klasifikátoru. Před užitím koncového klasifikátoru je v praxi obvyklé pro kontrolu porovnat výsledky z testovacího souboru s třetím nezávislým datovým souborem, někdy odborně nazývaným *development set*. Poněkud upravenou cestu dvoufázového učení ukazuje tzv. křížové ověřování (cross-validation), kdy se používá pouze jeden datový soubor. Hlavní odlišností ve vlastnostech křížového ověřování od popsaného procesu učení, je ztráta nezávislosti testovací fáze na fázi trénovací, která je zapříčiněna použitím dostupných pozorování pro obě části.

4.1 Trénovací část učení

Definujme typ klasifikátoru pomocí rodiny funkcí $\{d_\theta\}_{\theta \in \Theta}$, kde Θ je blíže nespecifikovaná množina hyperparametru. Pro zjednodušení značení budeme pro rodinu funkcí $\{d_\theta\}_{\theta \in \Theta}$ používat symbol D_θ. Pevnou volbou hyperparametru θ zvolíme prvek rodiny. O prvcích $d \in \theta$ budeme mluvit jako o klasifikátorech stejného typu.

Trénovací část učení se skládá:

i) Ze stanovení typu hledaného klasifikátoru d volbou rodiny D_θ.

ii) Z výběru θ_1 hyperparametru θ na základě trénovacího souboru dat.

4.2 Testovací část učení

Na trénování klasifikátoru d navazuje testovací část. V té nejprve sestrojíme predikce tříd objektů testovacího datového souboru na základě jejich znaků, pak pro každý objekt provedeme porovnání jeho skutečné třídy s třídou získanou predikcí. To konkrétně znamená, že pro každé i, $i = 1, \ldots, n$ provedeme

porovnání třídy g_i s třídou $d(x_i)$. Tím získáme odhad rizika klasifikátoru, resp. odhad pravděpodobnosti chybné predikce.

Přestože je tento odhad důležitý zejména pro odhad pravděpodobnosti chybné predikce na testovacím souboru dat (tzv. testovací chyba), využívá se většinou stejného odhadu i pro trénovací soubory dat. Je ovšem dobré si uvědomit, že hodnota trénovací chyby je víceméně orientační, neboť vypovídá o chybě predikce na datech, z nichž byl klasifikátor sestrojen. Hodnoty testovací chyby budou sloužit ke kvalitativnímu porovnání klasifikátorů.

5 Bayesův klasifikátor

5.1 Thomas Bayes

Thomas Bayes byl koncem 20. let 18. století stejně jako jeho otec vysvěcen a se stal knězem. Zabýval se teorií pravděpodobnosti, konkrétně problémem pravděpodobností různých možností. Svoji první práci publikoval v "Philosophical Transactions" Královské společnosti v Londýně v roce 1764.

Své metody založil na dlouho nepřijatelné myšlence, že lze určit pravděpodobnost určitého jevu při určitých podmínkách na základě pozorování, kolikrát za těchto podmínek tento jev již nastal a kolikrát za těchto podmínek nenastal.

V roce 1742 byl Bayes přijat za člena Královské společnosti, přestože do té doby nepublikoval žádnou práci z matematiky a navíc žádná z jeho prací nebyla během jeho života publikována pod jeho jménem. Zemřel 17. dubna 1761 v Tunbridge Wells v Anglii.

5.2 Bayesova věta v klasifikaci

V další části budu pracovat s modelem, ve kterém používám pravděpodobnosti a hustoty, včetně jejich značení, zavedené v kapitole věnované obecným principům klasifikátorů. Zde se omezím na klasifikační úlohy, určené jednoduchou ztrátovou funkcí. Pro tyto klasifikační úlohy je možno z již poznaného odvodit funkční předpis optimálního klasifikátoru:

$$d_{\text{opt}}(\boldsymbol{x}) = k_{\text{opt}} \quad \text{právě když} \quad k_{\text{opt}} = \underset{k \in \mathcal{G}}{\operatorname{argmax}}\ p_{G|\boldsymbol{X}}(k|\boldsymbol{x})$$

Pomocí Bayesovy věty, viz například Anděl (1985) [1], lze podmíněné pravděpodobnosti $pG|X\ (k|x)$ vyjádřit jako:

$$p_{G|\boldsymbol{X}}(k|\boldsymbol{x}) = \begin{cases} 0 & \text{pokud } p_{\boldsymbol{X}}(\boldsymbol{x}) = 0 \\ \dfrac{\pi_k\, p_{\boldsymbol{X}|G}(\boldsymbol{x}|k)}{p_{\boldsymbol{X}}(\boldsymbol{x})} & \text{jinak} \end{cases}$$

V terminologii Bayesovy statistiky se o pravděpodobnostech π_K mluví jako o apriorních pravděpodobnostech, podmíněné pravděpodobnosti $p_{G|X}\ (k|x)$ se nazývají aposteriorní. Pokud je pro vektor x hustota $p_X\ (x)$ nulová, pak jsou rovněž nulové všechny aposteriorní pravděpodobnosti a na volbě třídy z

hlediska optimality nezávisí. V takovém případě můžeme predikovanou třídu zvolit libovolně, například využitím generátoru náhodných čísel. Pokud dosadíme výše uvedený vztah podmíněné pravděpodobnosti $p_{G|X}(k|x)$ do předpisu optimálního klasifikátoru, dostáváme:

$$d_{\text{opt}}(\boldsymbol{x}) = \begin{cases} \text{náhodně zvolená třída } k \in \mathcal{G} & \text{pokud } p_{\boldsymbol{X}}(\boldsymbol{x}) = 0 \\ \underset{k \in \mathcal{G}}{\operatorname{argmax}} \ \dfrac{\pi_k \, p_{\boldsymbol{X}|G}(\boldsymbol{x}|k)}{p_{\boldsymbol{X}}(\boldsymbol{x})} & \text{jinak} \end{cases}$$

Úloha, nalézt optimální klasifikátor tak spočívá v nalezení největší aposteriorní pravděpodobnosti tříd na základě znalosti apriorních pravděpodobností tříd a podmíněných hustot znaků. Protože se zlomek $\pi_K \, p_{G|X}(k|x) / p_X(x)$ ve výše uvedeném dosazení mezi třídami liší pouze hodnotou čitatele, lze úlohu:

$$\max_{k \in \mathcal{G}} \ \frac{\pi_k \, p_{\boldsymbol{X}|G}(\boldsymbol{x}|k)}{p_{\boldsymbol{X}}(\boldsymbol{x})}$$

zjednodušit na úlohu:

$$\max_{k \in \mathcal{G}} \ \pi_k \, p_{\boldsymbol{X}|G}(\boldsymbol{x}|k)$$

Poslední uvedený vztah v sobě zahrnuje i případ $pX(x) = 0$, u kterého ovšem z hlediska optimality na volbě třídy nezávisí.[1]

5.3 Empirické ověření Bayesovy věty

Vzhledem k faktu, že definované vztahy v minulých kapitolách mají svůj základ ve statistice, je možno je ověřit na jednoduchém pokusu. Uvažujme příklad hodu poškozenou mincí, kdy z deseti hodů padl sedmkrát líc. Ukážeme výpočet odhadu parametru pomocí obvyklého a bayesovského učení.

5.3.1 Obvyklé učení

Počet líců je 7, počet hodů 10. Odhad parametru apriorní *hp* π_K je:

$$\pi_{\text{k}} = \frac{7}{10} = 0{,}7$$

5.3.2 Bayesovské učení

5.3.2.1 Bayesovské učení se slabší apriorní informací

Předpokládejme, že poškození mince je zanedbatelné, a tedy obě pravděpodobnosti by měly být stejné a rovny 0,5. Touto skutečností si ale nejsme příliš jisti. Zvolíme proto apriorní statistiky $n_{1;0}$ *a* $n_{0;0}$ rovny 1, které odpovídají „dvěma apriorním datům". Apriorní *hp* potom odpovídá odhadu $1/(1+1) = 0,5$; konečné statistiky jsou $n_{1;10} = 1 + 7 = 8$ a $n_{0;10} = 1 + 3 = 4$ a konečný odhad je:

$$\boxed{\pi_{\mathrm{k}} = \frac{8}{12} = 0,\bar{6}}$$

5.3.2.2 Bayesovské učení se silnější apriorní informací

Uvažujeme stejnou situaci jako v předchozím případě, ale již jsme se z dřívějších pokusů přesvědčili, že tak malé poškození, jaké je u naší mince, nemá téměř žádný vliv. Statistiky proto volíme $n_{0;0} = 100$ a $n_{1;0} = 100$, odpovídající „dvěma stům apriorních dat". Apriorní odhad je stejný: $100/(100+100) = 0,5$; konečné statistiky jsou $n_{1;10} = 100 + 7 = 107$ a $n_{0;10} = 100 + 3 = 103$. Konečný odhad je:

$$\boxed{\pi_{\mathrm{k}} = \frac{107}{210} = 0,51}$$

Vidíme, že obě apriorní informace „přitáhly" výsledek k a priori předpokládané hodnotě 0,5. Větší hodnoty statistik mají proti informaci z dat větší vliv. Je to pochopitelné. První apriorní informace odpovídá jednomu a priori změřenému líci a jednomu rubu. Druhá odpovídá 100 lícům a 100 rubům. Proti nim vždy stojí deset změřených dat a odhad je optimálním kompromisem mezi apriorní informací a informací získanou z dat.

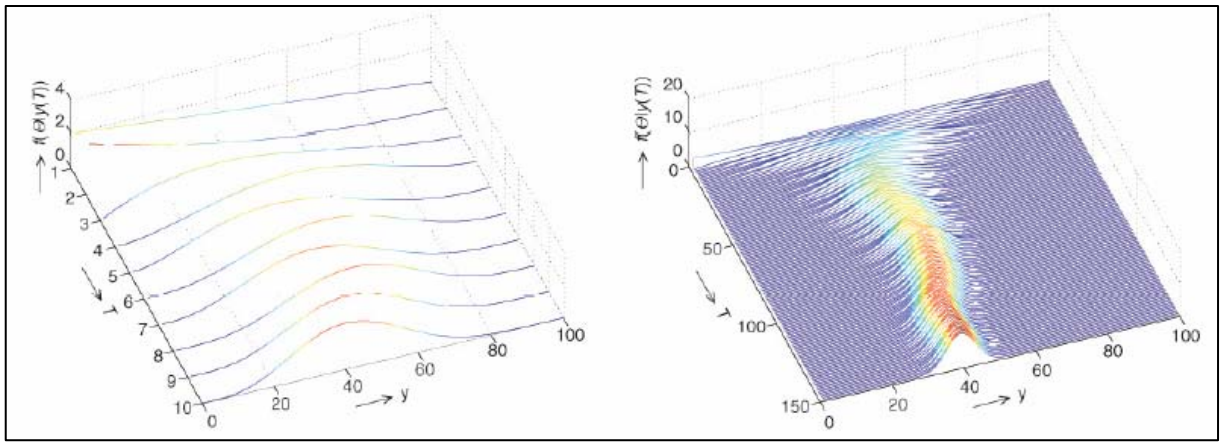
Úplným popisem parametru π_K je ale aposteriorní *hp*. Na dalším příkladu bude demonstrován její vývoj během rekursivního odhadu a její výsledný tvar pro ráznou apriorní informaci.

5.3.2.3 Příklad aposteriorní hustoty pravděpodobnosti při bayesovském učení

Uvažujeme stejný hod mincí jako v předchozím příkladu, ale budeme sledovat celou aposteriorní *hp* neznámého parametru π_K, a to pro různý počet měřených dat a různou apriorní informaci. Data získáme na simulovaném experimentu, kde volíme skutečnou, tj. simulovanou, hodnotu parametru $\pi_K = 0,4$. Pro ilustraci si budeme v jednotlivých případech uvádět trojrozměrný graf, ve kterém budou za sebou řazeny *hp* parametrů tak, jak se měnily s přibývajícím počtem zpracovaných dat. Nejstarší průběhy *hp* jsou v grafu vzadu, směrem dopředu postupují novější.

5.3.2.4 Srovnání krátkého a dlouhého datového vzorku

Se zvětšujícím se počtem zpracovaných dat se zpřesňuje odhad. To se projeví snižováním rozptylu aposteriorní *hp*. Na obr.1a jsou dva experimenty s parametrem $\pi_K = 0,4$. Levý obsahuje deset dat, pravý 150 dat.



Obr. 1a: Odhadování s různou délkou datového vzorku a nulovou apriorní informací

Z levé části obrázku je patrné, že aposteriorní *hp* z malého datového vzorku je dosti neurčitá (má velký rozptyl - šířku). Pravá část obrázku ukazuje, jak se s přibývajícím počtem zpracovaných dat aposteriorní *hp* postupně zpřesňuje (rozptyl klesá). Konečný průběh (nejvíce vpředu) je již dosti přesný.

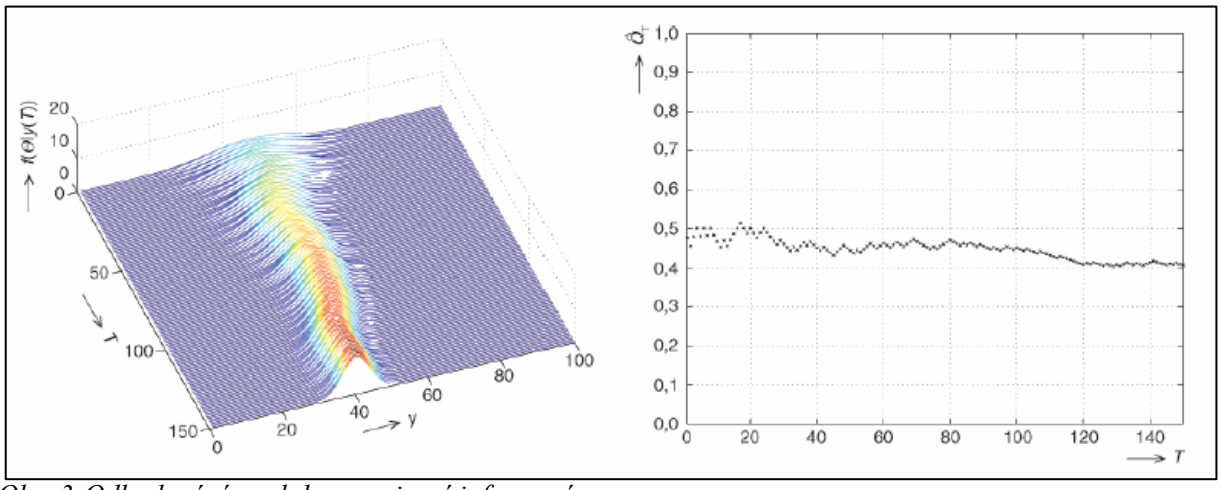
Začátky na obou částech grafu (tj. průběhy úplně vzadu) jsou velmi „nejisté" a „roztěkané". Je to dáno tím, že oba starty jsou bez jakékoliv apriorní informace a snaží se přesně respektovat veškerou informaci, kterou jednotlivá data přinášejí.

```
clc, clear all, clf, rand('seed',125)
% Zadání vstupních údaju
h=.01; % krok diskretizace
T=150; % pocet kroku simulace
Ths=.4;                                % simulovaná pravdepodobnost lícu
n1=20; n0=20;                          % pocátecní hodnoty statistik

Thv=h:h:1;                             % diskretizace hodnot parametru
yt=fix(rand(1,T)+Ths);                 % simulace mince
Thm=[]; Thh=[];

% Cyklus pro cas (zpracování dat)
for t=1:length(yt)
   n1=n1+yt(t); n0=n0+1-yt(t);         % prepocet statistik
   Tht=[];
   for Th=Thv
      fTh=Th^(n1)*(1-Th)^(n0);         % konstrukce aposteriorní hp
      Tht=[Tht fTh];
   end
   Tht=Tht/sum(Tht)/h; Thm=[Tht; Thm]; % normování hp
   The=n1/(n1+n0);     Thh=[Thh The];  % bodové odhady
end

% Kreslení výsledku
figure(1)
subplot(211),waterfall(Thm), grid on, view(-20,75)
subplot(212),plot(Thh,'x','markersize',3),grid on,axis([1,T,0,1])
```

Obr. 2: Zdrojový kód simulace v jazyce MATLAB 5.0

5.3.2.5 Vliv slabé apriorní informace

Tento experiment odpovídá předchozímu se 150 daty, na začátku je ale aplikována apriorní informace. Tváříme se, jako bychom neznali skutečnou hodnotu parametru π_K, a zadáme apriorní informaci odpovídající nepoškozené minci.



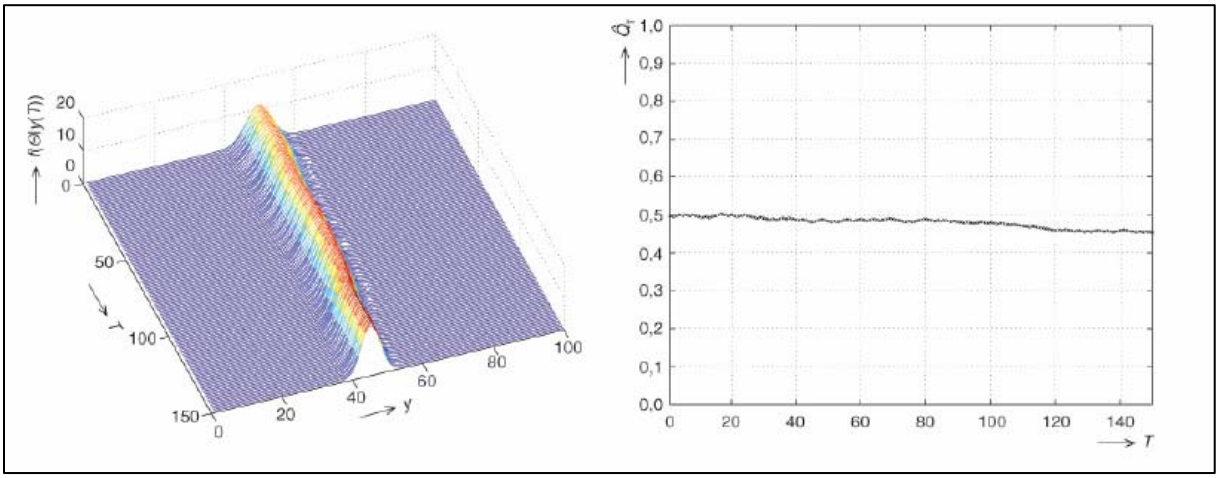
Obr. 3:Odhadování se slabou apriorní informací

Nejdříve použijeme „slabou informaci“ s hodnotami apriorních statistik $n_{1;0} = n_{0;0} = 10$. Levý obrázek ukazuje vývoj aposteriorní hp, pravý vývoj bodového odhadu.

Z grafů tohoto případu je patrné, že počáteční vývoj aposteriorní hp se hezky „uklidnil". Navíc hodnota koeficientu byla určena správně, protože na konci experimentu byla ne zcela správná apriorní informace „přebita" měřenými daty a hodnota parametru π_K se odhadla správně.

5.3.2.6 Vliv silné apriorní informace

Opět nám půjde o stejný experiment, jako je předchozí, ale s apriorními statistikami $n_{1;0} = n_{0;0} = 100$. Výsledek ukážeme na obr.3 a obr.4. Z grafu na obr.4 je vidět, že použitá apriorní informace byla příliš silná - nedovolila, aby se prosadila správná hodnota parametru, která je 0,4.



Obr. 4: Odhadování se silnou apriorní informací

5.4 Bayesův klasifikátor

Nechť jsou dány všechny apriorní pravděpodobnosti π_K a všechny podmíněné hustoty $p_{G|X}(k|x)$ $k = 1...K$. Pak klasifikátor definovaný jako

$$d_{\text{opt}}(\boldsymbol{x}) := \underset{k \in \mathcal{G}}{\operatorname{argmax}} \ \pi_k \, p_{\boldsymbol{X}|G}(\boldsymbol{x}|k)$$

minimalizuje riziko

$$\mathsf{R}^*(d), \text{ tj. } \mathsf{R}^*(d_{\text{opt}}) \leq \mathsf{R}^*(d), \forall d \in \mathcal{D}$$

Pro optimální klasifikátor definovaný tímto vztahem se užívá termínu Bayesův klasifikátor. Pro optimální riziko R^* (d_{opt}) se analogicky používá výrazu Bayesovo riziko. Připomínám, že Bayesovo riziko je v ideálním případě znalosti všech potřebných pravděpodobností a hustot dolní mezí pravděpodobnosti chybné predikce všech klasifikátorů.

5.5 Konstrukce Bayesova klasifikátoru

Následující kroky popisují konstrukci Bayesova klasifikátoru v praxi:

i) Stanovení odhadů $p'_{X|G}(x|k)$ podmíněných hustot $p_{X|G}(x|k)$

ii) Stanovení odhadů π'_K apriorních pravděpodobností π_K.

iii) Nalezení maximálního součinu $\pi'_K \,.\, p'_{X|G}(x|k)$.

Zásadní část konstrukce Bayesova klasifikátoru spočívá v odhadování hustot $p_{X|G}(x|k)$. Pokud nemáme žádnou představu o rozdělení, vyplývajícím z povahy dat, nezbývá než přikročit k neparametrickým (často jádrovým) odhadům hustot. Já se však budu zabývat parametrickými metodami, které jsou založeny na předpokladu, že hustoty $p_{X|G}(x|k)$ pocházejí z parametrických rodin $p\theta_k(x)$ s parametry $\theta_k \in \Theta_k$. Odhad hustot pak spočívá ve stanovení odhadů θ'_k.

Rodina hustot parametrických modelů je pro klasifikační úlohu buď známa dopředu (použití dřívějších znalostí odjinud než z dostupného souboru dat), nebo je její volba součástí konstrukce klasifikátoru. V praxi se často volí rozdělení *ad hoc*. Nejpoužívanější *ad hoc* volbou jsou modely s předpokladem p-rozměrného normálního rozdělení $\mathcal{L}(\boldsymbol{X}|G=k) \sim \mathsf{N}_p\{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}$, kde $\boldsymbol{\mu}_k = \mathsf{E}_{\boldsymbol{X}|G}[\boldsymbol{X}|G=k]$ a $\boldsymbol{\Sigma}_k = \mathsf{Var}_{\boldsymbol{X}|G}[\boldsymbol{X}|G=k]$ jsou střední hodnota a varianční matice v rámci k-té třídy. Parametrem θ_k pro normální model je tedy dvojice (μ_k, Σ_k).[4]

Odhady π'_K jsou podobně jako v případě odhadů hustot buď dopředu známy, nebo se konstruují z trénovacího souboru dat. Ze všech možností, jak odhadovat π_K, zmiňme dva v praxi nejpoužívanější typy odhadů. Prvním jsou odhady $\pi'_K = n_k/n$, kde n je celková velikost souboru a n_k je počet objektů k-té třídy v souboru. Pravděpodobnosti π_K jsou jimi odhadnuty na základě poměrného zastoupení tříd v trénovací množině. Druhým typem jsou odhady $\pi_K = 1/K$, které z datového souboru nevyužívají žádnou informaci. O rozdělení tříd se tak předpokládá, že je rovnoměrné.

5.6 Výhody a využití Bayesovského přístupu

Výhodou Bayesovské klasifikace je její poměrně velké využití. Bayesův vzorec se často používá v populačních etiologických studiích a v některých matematických modelech diagnostického, terapeutického či prognostického lékařského rozhodování. Ve světě informačních technologií se Bayesovské filtry používají v programech proti spamu, protože se dokáží postupně přizpůsobovat situaci a využívat nové informace.

K jejich rozšíření přispěla jistě i jejich snadná implementace a rychlost, se kterou klasifikují testované vzorky. Pokud má Bayesovský filtr dostatečné množství „naučených" dat, vykazuje velmi přesné výsledky.

Velkou výhodou je i fakt, že umožňuje učení klasifikátoru i úplným laikům.

5.7 Nevýhody Bayesovského přístupu

Nevýhodou Bayesovského přístupu je nesnadná integrace přes více nejistých parametrů, která často vyžaduje náročnou metodu Markovova řetězce a metody Monte Carlo.

Možnou nevýhodou by mohlo také být i nesprávné pojetí pravděpodobnosti, jako měřítka (ne)jistoty jednotlivých experimentů, místo pravděpodobnosti, jako frekvence opakovaných experimentů.

Za určitých okolností by mohl být nevýhodou i předpoklad, že dané atributy musí být na sobě nezávislé. V praxi bývají na sobě atributy často závislé. Takto specifikované úlohy již nelze řešit využitím klasického Bayesovského přístupu. Řešením těchto úloh je použití tzv. Bayesovských sítí.

Nejznámějším problémem Bayesovského přístupu je závislost na počtu atributů v třídě. Pokud pro nějaké k platí: $pG|X\ (k|x) = 0$, potom $pX\ (x) = 0$. V praxi to znamená, že malý, resp. nulový počet prvků v kategorii, může uměle zvyšovat pravděpodobnost příslušnosti do této kategorie. Řešením tohoto problému je tzv. Laplaceova korekce, tedy přidání jednoho prvku do všech množin (kategorií).

5.8 Bayesovské filtry

Bayesovské principy našly uplatnění ve velkém množství různých aplikací a metod. V případě klasifikace textu jsou nejčastěji užívanou metodou v detekci nevyžádané pošty.

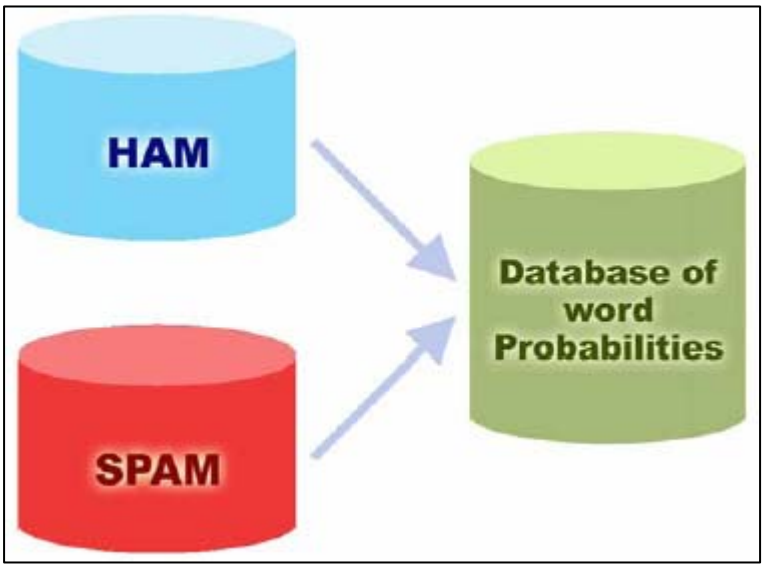
5.8.1 Využití ve filtrování elektronické pošty

Nejrozšířenější využití mají Bayesovské principy v oblasti filtrování nevyžádané pošty, tzv. spamu. Protože je princip této činnosti v podstatě totožný s řešením mého problému, popíšu v následujících odstavcích princip této činnosti.

První aplikace bayesovské logiky na problémy spamu byly odstartovány textem Paula Grahama Plán pro spam [11]. Princip popisovaného filtru je velmi jednoduchý.

Každému slovu či znaku je přidělena hodnota pravděpodobnosti, ta je založena na výpočtech, které berou v úvahu četnost výskytu slova v nevyžádané poště (spamu) vzhledem k legitimní zprávě (hamu). Děje se tak analýzou uživateli odeslané pošty a analýzou známého spamu: všechna slova z obou množin jsou analyzována a je vygenerována pravděpodobnost indikující spam.

Důležitým faktorem v práci s Bayesovským filtrem je jeho učení na konkrétních případech. Například finanční ústav používá slovo hypotéka poměrně často, ale s obecnými antispamovými pravidly by se pošta vyhodnotila jako spam.



Obr. 5: Třídy filtrovaných množin

Bayesův filtr bere naopak v úvahu obsah odesílaných emailů (a rozpozná, že se slovo „hypotéka“ ve validní poště objevuje poměrně často). Může tak mít velice nízké procento tzv. false-positiv (nesprávného vyhodnocení).

Bayesův filtr se spoléhá na soubor s daty o spamu. Tento soubor musí obsahovat široký vzorek známého spamu a musí být průběžně aktualizován. Bayesovu filtru zajistí schopnost reakce na nejnovější triky spammerů, a tím i vysokou schopnost reakce na nevyžádanou poštu.

5.8.2 Průběh filtrování

Ve chvíli, kdy je doručen nový email, je rozdělen na slova. Poté jsou vybrána ta, která jsou pro detekci spamu nejdůležitější. Z těchto slov vypočítá Bayesův filtr pravděpodobnost, zda se jedná spam. Zpráva je označena za spam, jestliže je pravděpodobnost vyšší než prahová hodnota.

Bayesův přístup je při posuzování spamu velmi efektivní – v květnu 2003 informoval článek BBC o faktu, že Bayesova technologie může dosáhnout přesnosti 99,7% při minimálním množství falešných pozitiv.

5.8.3 Výhody Bayesova filtrování

1. Bayesova metoda bere v úvahu celou zprávu – rozpoznává slova, která identifikují spam, ale i slova, které označují validní zprávu. Například: ne každý email, který obsahuje slova „zdarma“ a „hotovost“, musí být spam. Výhodou Bayesova filtru je, že bere ohledy na nejdůležitější slova a spočítá pravděpodobnost toho, zda se jedná o spam. Bayesova metoda sice shledá slova „zdarma“ a „hotovost“ důležitými, ale také rozpozná přítomnost jména odesilatele v kontaktech a tím pádem vyhodnotí zprávu jako běžnou poštu. Jinými slovy: Bayesovo filtrování je daleko inteligentnější přístup, jelikož oproti kontrole na klíčová slova, zkoumá veškeré aspekty zprávy.

2. Bayesův filtr se neustále zdokonaluje – učením z nového spamu i validní odesílané pošty se Bayesův filtr vyvíjí a přizpůsobuje novým praktikám spammerů. Pokud spammeři začnou používat „z-d-a-r-m-a“ namísto slova „zdarma“, budou úspěšně obcházet kontrolu na klíčová slova do doby, než bude spojení „z-d-a-r-m-a“ doplněno do databáze klíčových slov. Bayesův filtr takové praktiky ihned odhalí a pokud nalezne spojení „z-d-a-r-m-a“, je to ten nejlepší ukazatel pro identifikaci spamu.

3. Bayesova technika je citlivá k uživateli – aby spam dosáhl svého adresáta, musejí spammeři zasílat takové emaily, které nemohou filtry příjemce odhalit. Jelikož Bayesovo filtrování bere v úvahu profil konkrétních emailů, rozpoznává spam daleko snadněji tento profil by spammeři museli k obelstění filtru znát.

4. Bayesova metoda je multilingvální – Bayesův antispamový filtr může být – díky adaptabilitě – použit pro jakýkoliv jazyk. Mnoho seznamů klíčových slov existuje pouze v anglickém jazyce, a proto jsou pro jinou řečí mluvící oblasti zcela nepoužitelné. Bayesův filtr bere v potaz i jazykové odchylky či různý

význam slov v různých oblastech. Tato schopnost zvyšuje šance na zachycení většího počtu nevyžádané pošty.

5. Oklamání Bayesova filtru je obtížnější, než oklamání filtrování klíčových slov – pokročilý spammer, který chce Bayesův filtr obejít, může buď použít méně slov, ukazujících na spam (např. Viagra, Cash, atd.), nebo použít více slov identifikujících validní email (platné jméno z kontaktů, apod.). Provedení druhého způsobu můžeme vyloučit, jelikož spammer nemůže znát poštovní profily jednotlivých příjemců. Použití neutrálních slov, jako např. „public“ nepřikládá závěrečná analýza žádnou váhu a proto nemůže spammerům pomoci.

6 Neuronové sítě

6.1 Stručná historie neuronových sítí

Za počátek vzniku oboru neuronových sítí je považována práce Warrena McCullocha a Waltera Pittse [17] z roku 1943, kteří vytvořili velmi jednoduchý matematický model neuronu, jakožto základní buňky nervové soustavy. Číselné hodnoty parametru v tomto modelu byly převážně bipolární, tj. z množiny {-1,0,1}. Autoři ukázali, že nejjednodušší typy neuronových sítí mohou v principu počítat libovolnou aritmetickou nebo logickou funkci.

V roce 1957 Frank Rosenblatt vynalezl tzv. perceptron, který je zobecněním McCullochova a Pittsova modelu neuronu pro reálný číselný obor parametrů. Pro tento model navrhl učící algoritmus, o kterém matematicky dokázal, že pro daná tréninková data nalezne po konečném počtu kroků odpovídající váhový vektor parametrů (pokud existuje) nezávisle na jeho počátečním nastavení.

Na základě tohoto výzkumu Rosenblatt spolu s Charlesem Wightmanem a dalšími sestrojili během let 1957 a 1958 první úspěšný neuropočítač, který nesl jméno „Mark I Perceptron“. Protože původním odborným zájmem Rosenblatta bylo rozpoznávání obrazců, „Mark I Perceptron“ byl navržen pro rozpoznávání znaků. Znak byl promítán na světelnou tabuli, ze které byl snímán polem 20x20 fotovodičů. Intenzita 400 obrazových bodů byla vstupem do neuronové sítě perceptronů, jejímž úkolem bylo klasifikovat, o jaký znak se jedná.

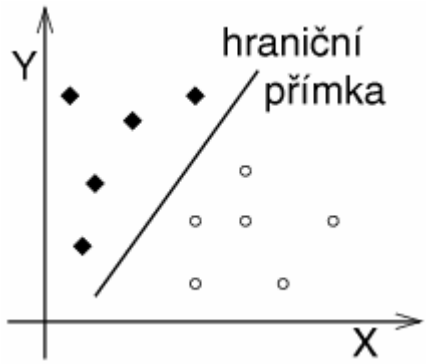
Na přelomu 50. a 60. let došlo k úspěšnému rozvoji neurovýpočtů v oblasti návrhu nových modelů neuronových sítí a jejich implementací. Výsledky z uvedeného období shrnul Nilse Nilssona v knize „Principles of Artificial Intelligence“ z roku 1965 [18].

V roce 1987 se v San Diegu konala první větší konference, specializovaná na neuronové sítě (IEEE International Conference on Neural Networks), na které byla založena mezinárodní společnost pro výzkum neuronových sítí INNS (International Neural Network Society). O rok později INNS začala vydávat svůj časopis „Neural Networks“. V následujících letech vznikly další specializované časopisy: Neural Computing (1989), IEEE Transactions on Neural Networks (1990) a mnoho jiných[1]. Od roku 1987 mnoho renomovaných univerzit založilo nové výzkumné ústavy, zabývající se neuronovými sítěmi a vyhlásilo výukové programy zaměřené na neuro-výpočty. Tento trend pokračuje dodnes.

[1] v Praze vychází od roku 1991 mezinárodní časopis Neural Network World

6.2 Perceptron

Perceptron je typ neuronu, který rozděluje vstupní prostor nadrovinou na dva poloprostory. Obecně lze říci, že se jedná o nejjednodušší neuronovou síť, hledající v rovině rovnici hraniční přímky.



Obr. 6: Vyhledání hraniční přímky neuronovou sítí

Jako inspirace pro umělé neuronové sítě samozřejmě sloužil biologický vzor. Perceptron pak představuje silné zjednodušení biologického neuronu a tvoří základní stavební element umělé neuronové sítě.

Vstup perceptronu reprezentuje n prvkový vektor X = (x1, x2...xn), výstup je skalární veličina y. Míra přenosu synaptických spojení je modelována tzv. váhami a ofsetem (bias), kdy ofset zpravidla tvoří n+1 prvek váhové matice W(w1,w2..wn). Na sečtený signál x je aplikována nelineární funkce f, výsledek je pak výstupem perceptronu. Chování modelu neuronu je popsáno vztahem

$$y = f\left(\sum_{i=1}^{n} w_{ij} O_i + \theta_j\right)$$

Kde O_i je výsledek jedné z výstupních neuronových funkcí - sigmoida:

$$O_j = \frac{1}{1+e^{-I_j}}$$

Tvar výstupní funkce má důležitý vliv na výsledné chování modelu neuronu. Nejčastěji používané funkce jsou sigmoida, hyperbolický tangens, identita, saturace, signum a Gausova funkce [6]. Funkce identita, saturace a signum se používají převážně při klasifikaci, zatímco zbývající pro identifikaci a simulování systémů. Analytické vyjádření těchto funkcí je:

Sigmoida:

$$y = \frac{1}{1 + e^{-x}}$$

Hyperbolický tangens (tanh):

$$y = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

Gausova funkce:

$$y = e^{-x^2}$$

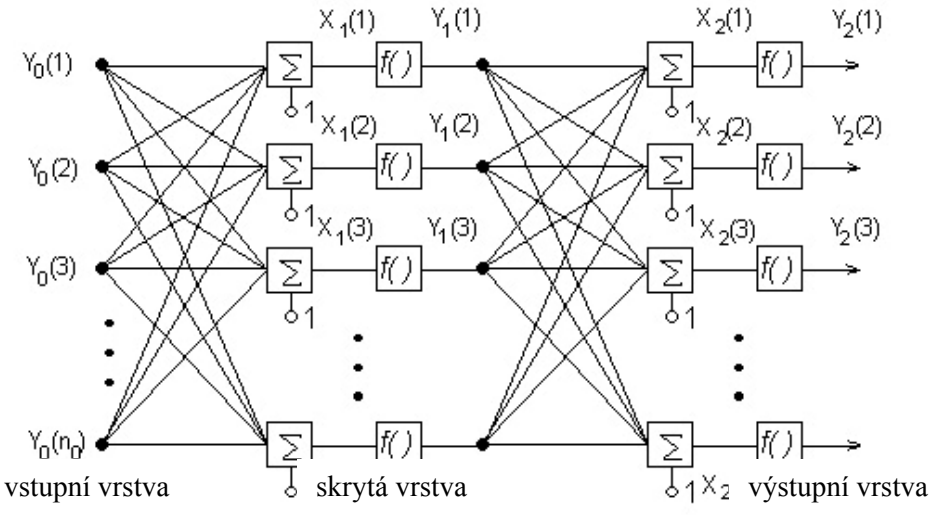
Nejběžněji požívaná výstupní funkce u umělých neuronových sítí je sigmoida, v teorii řízení se však více aplikuje hyperbolický tangens z důvodu jeho symetrie podle osy *x* [6].

Nyní, když jsem vysvětlil pojem umělé neuronové sítě, budu v následujícím textu umělé neuronové sítě nazývat prostě neuronovými sítěmi.

6.3 Vícevrstevná perceptronová síť

Pro identifikaci a řízení je nejčastěji používaný typ neuronové sítě vrstevnatá perceptronová síť (Multi Layer Perceptron - MLP), v kombinaci s učícím algoritmem Backpropagation (BP), či v poslední době novějším a rychlejším algoritmem Marquardt-Levenberg (ML).

Základní struktura MLP je zobrazena na obr.7. Jedná se o vrstevnatou dopřednou síť, kde směr toku dat je zleva doprava. (Y_0 je vstupní vektor, Y_2 je výstupní vektor).



Obr. 7:Vrstevnatá perceptronová síť s jednou vnitřní vrstvou[7]

Síť na obr.7 obsahuje celkem tři vrstvy. Z tohoto důvodu nazýváme tyto sítě jako vícevrstvé perceptronové sítě. První vrstvou je vrstva vstupní, obsahující n_0 uzlů. Další, skrytá vrstva je připojena na výstup vstupní vrstvy a často obsahuje nelineární výstupní funkci. Skrytých vrstev může být v neuronových sítích definován libovolný počet. Výstupní vrstva s n_2 neurony obsahuje velmi často lineární výstupní funkci pro identifikaci.

6.3.1 Parametry vícevrstvých perceptronových sítí

Správná funkce neuronové sítě je závislá na následujících parametrech:

- *počet vrstev* - síť se dvěmi vnitřními vrstvami je schopna modelovat většinu systémů, síť s jednou vnitřní vrstvou je brána jako univerzální aproximátor
- *počet neuronů v každé vrstvě* - počet neuronů ve výstupní vrstvě je dán počtem požadovaných výstupů sítě, takže pouze počet neuronů ve vnitřních vrstvách sítě je zadáván uživatelem. Správný počet záleží na tom, jak složitý problém chceme řešit. Příliš málo neuronů má za následek, že síť není schopna se naučit chování daného systému, příliš mnoho neuronů (tzv. overtraining) zvyšuje riziko, že síť bude mít spoustu stupňů volnosti a učící algoritmus nemusí najít nejlepší řešení nebo síť se může naučit i šum.
- *výstupní funkce neuronů* - nejběžněji používaná výstupní funkce je sigmoida, v teorii řízení se často potkáme s funkcí hyperbolický tangens.

6.4 Učení neuronové sítě

Neuronová síť musí být nastavena tak, aby při předložení daných vstupních dat byly na výstupu sítě požadované výstupní hodnoty. Konfigurace sítě se provádí změnou vah jednotlivých vstupů jednotlivých neuronů. Existuje několik způsobů jak tyto váhy co nejlépe nastavit. Jedna cesta je nastavit tyto váhy přímo použitím znalosti a priori. Druhá cesta je učení (adaptace) neuronové sítě postupným předkládáním vstupních a požadovaných výstupních hodnot a změnou vah podle jistého adaptačního pravidla.

V závislosti na typu učícího algoritmu jsou neuronové sítě rozdělovány podle kritéria, které pak slouží pro optimální nastavení vah sítě. Sítě jsou děleny na sítě, které k adaptaci potřebují učitele a sítě bez učitele. Nejpoužívanější jsou sítě, které potřebují učitele a obsahují zpětnou vazbu. Tyto neuronové sítě jsou i součástí praktické části mé diplomové práce. Zde je zpětná vazba obecně hlediskem, podle něhož se optimalizuje síť. Zatímco sítě, které vyžadují učitele patří do skupiny nejpoužívanějších, sítě bez učitele jsou zpětnovazební sítě, které se blíží skutečným biologickým soustavám. Ty jsou matematicky, i

po stránce aplikací, výrazně komplikovanější. Hledisko pro modifikaci vah si tedy u sítí, které nevyžadují učitele, stanovuje sama síť a na základě něho se učí.

Jak jsem již bylo zmínil, budu se dále zabývat pouze zpětnovazební neuronovou sítí s využitím učitele (NS Backpropagation). Algoritmus zpětného šíření chyby využívá ve fázi učení znalosti očekávané hodnoty na výstupu. Skutečná hodnota, kterou neuronová síť vrátí, se porovná s očekávanou hodnotou a vypočte se chyba výstupní vrstvy. Tato chyba je pro výstupní vrstvu dána vztahem:

$$Err_j = O_j(1 - O_j)(T_j - O_j),$$

kde T_j je očekávaná hodnota. Vypočtená chyba se nadále distribuuje rekurzivním způsobem do skrytých vrstev. Ve skrytých vrstvách neuronové sítě platí pro výpočet chyby vztah:

$$Err_j = O_j(1 - O_j)\sum_k Err_k w_{jk} .$$

Po vypočtení chyb neuronu můžeme upravit jednotlivé váhy w a biasy θ:

$$\Delta w_{ij} = (l) Err_j O_i \qquad w_{ij} = w_{ij} + \Delta w_{ij}$$

$$\Delta \theta_j = (l) Err_j \qquad \theta_j = \theta_j + \Delta \theta_j$$

Ve výpočtu a úpravě vah a biasů hraje velkou roli i parametr l. Jde o tzv. koeficient učení, jehož hodnota se pohybuje v intervalu <0,1>. Doporučená hodnota je dána vztahem $l = 1 / t$, kde t je aktuální počet iteračních kroků [3].

7 Support Vector Machine

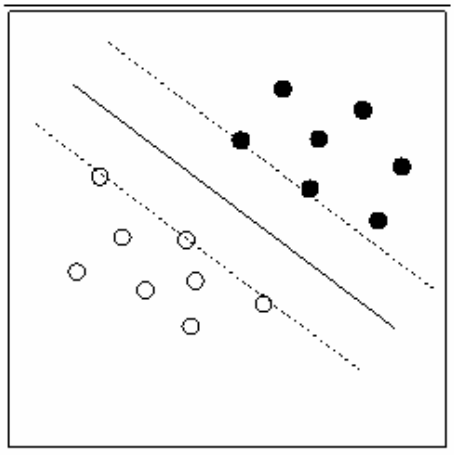
SVM (Support Vector Machine) je nová technika klasifikace dat do dvou tříd. SVM byl prvně publikován v roce 1992 V. N. Vapnikem [8]. Jeho oblíbenost a rozšířenost spočívá především v tom, že efektivita správně nastaveného SVM je srovnatelná se složitými neuronovými sítěmi.

Předpokládejme, že chceme rozdělit nějaké soubory dat do dvou tříd (základní verze klasifikuje jen do dvou tříd). Každý soubor dat je reprezentován hodnotami (slovy) a vytváří bod v prostoru. SVM klasifikátor se snaží nalézt nadrovinu, která by tyto body oddělila a rozdělila tak data na pozitivní a negativní případy. Nejjednodušším případem je lineárně separovatelná úloha. To znamená, že body obou tříd je možno oddělit lineární funkcí. Jako elementární případ si můžeme představit čísla v komplexní rovině, která chceme rozdělit do dvou tříd - záporná a kladná. Je asi zřejmé, že hledanou rozdělující funkcí bude přímka procházející počátkem souřadného systému kolmo na reálnou osu *x*. Dále je vhodné, aby zvolená nadrovina měla co největší vzdálenost od všech bodů (margin). Tím je zajištěno, že rozdělení do tříd bude co nejspolehlivější.Předpokládejme trénovací data ve tvaru: *{(x1,c1),(x2,c2)...(xn,cn)}* kde *x* jsou jednotlivé body a *c* je buď -1 nebo 1, podle toho, do které třídy daný bod patří. Jelikož se snažíme získat co největší vzdálenost od bodů obou tříd, hledáme dvě rovnoběžné nadroviny ve tvarech :

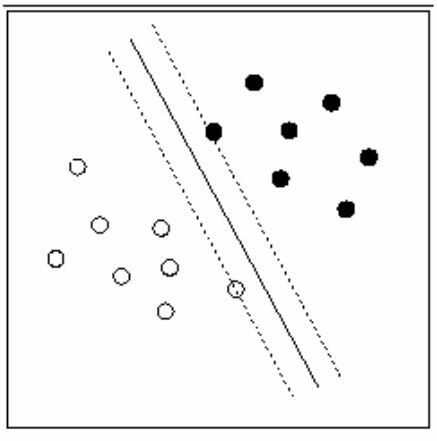
$$w\,x - b = -1$$

$$w\,x - b = 1$$

Cílem je, aby tyto nadroviny měly co největší vzdálenost od optimální nadroviny a aby mezi nimi neležely žádné body (viz obr. 8). Vzdálenost mezi oběma nadrovinami je $2/|w|$. Snahou je tedy minimalizovat $|w|$.



Obr. 8: Optimální poloha nadroviny [7]



Obr. 9: Špatná poloha nadroviny [7]

8 Implementace a měření

V praktické části diplomové práce se zaměřím na popis implementace vytvořených klasifikačních knihoven a na testování knihoven v reálných podmínkách. Knihovny pracují s online webovými diskuzemi, a proto je k jejich vytvoření třeba použít některého ze skriptovacích jazyků pro generování webových stránek.

Obě klasifikační třídy jsou vytvořeny ve skriptovacím jazyce PHP, který je v současné době nejpoužívanější platformou tohoto typu. Výhodou jazyka je velká podpora na serverech (hosting), velmi dobrá dokumentace dostupná z velké části i v českém jazyce, nebo například velká uživatelská základna. Nevýhodou, se kterou jsem se potýkal zejména při testování neuronových sítí, jsou omezující serverová zabezpečení, paměťová omezení, či nedostatek volných systémových prostředků.

Obě metody si vzhledem k svému charakteru vyžádali přítomnost databázového systému. Vhodný typ databáze jsem vybíral na základě požadavků, s ní spjatých. Jako vhodného kandidáta jsem vybral databázový stroj MySQL. Práce s touto databází je rychlá a existuje zde velká podpora. Pro účely testování jsem nepředpokládal značné množiny záznamů. Pokud by se nasbíraná data měla pohybovat v milionech databázových řádků, pravděpodobně by bylo vhodné doporučit jinou databázi. Jako vhodnou bych doporučoval například Oracle.

8.1 Implementace Bayesovy metody

Jako součást řešení implementace jsem vytvořil jednu objektovou knihovnu, do které jsem soustředil veškeré funkční prvky. V rámci jednoho objektu je tak možné Bayesův klasifikátor nechat učit i testovat přicházející vstupy.

Třída obsahuje čtyři atributy. Atribut $*knowledgeBase* je typu *public* a obsahuje trojrozměrné pole slov načtených z databáze. Toto pole obsahuje i jejich výskyty, procentuální hodnotu pravděpodobnosti aj.

Atribut $poleKategorii obsahuje jednorozměrné pole naplňované již z konstruktoru objektu. Jako klíče pole slouží primární klíče databázové tabulky obsahující kategorie.

Třetím privátním atributem je $newText, jenž obsahuje pole vkládaného textu. Toto pole obsahuje informaci o kategorii oklasifikovaného textu.

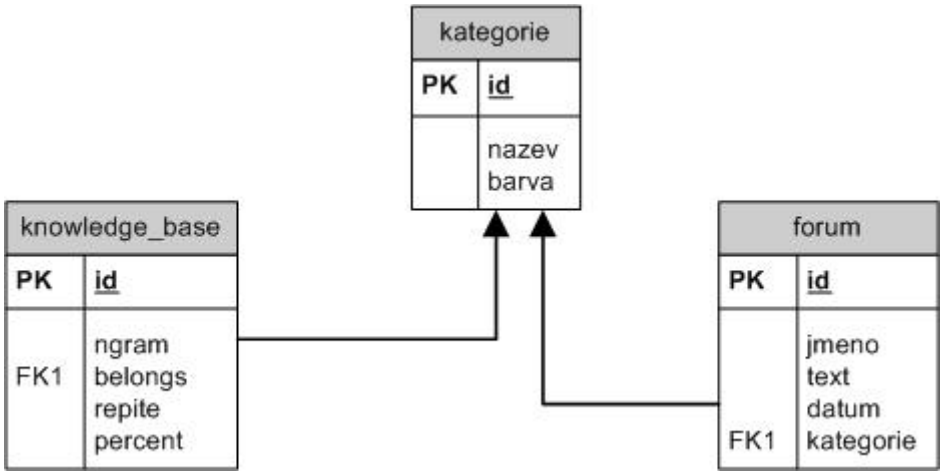
Jako úložiště součtů výskytů slov v rámci jednotlivých kategorií slouží atribut $pocetVyskytuKategorii.

Objekt obsahuje šestnáct metod. V rámci dobrého pochopení třídy popíšu nejdůležitější z nich. Konstruktor objektu *bayes*() načte při vytvoření třídy kategorie a pole $knowledgeBase obsahující výskyty slov. Veřejná metoda *PrepoctiKB*() použije atribut $*newText* a po rozbije příspěvek na slova. Tato slova potom připočítá do hlavního slovníku. Metodu *OdectiKB*() lze použít pro pravý opak v případě, že dojde k chybnému zařazení příspěvku. Takto vypočtené pole je vloženo do metody *VypoctiBayesianFiltering*(), která vypočítá bayesovskou pravděpodobnost a vloží čísla zpět do pole, se kterým pracuje. Takto upravené pole potom stačí jen uložit zpět do databáze. Metodou pro test, zda vkládaný příspěvek patří do zadané kategorie je metoda *isItSpam_v2*(), která má návratovou hodnotu typu float.

Třída požaduje databázovou tabulku knowledge_base, která obsahuje pět sloupců a její struktura by se dala popsat následujícím SQL dotazem:

```
CREATE TABLE `knowledge_base` (
  `id` int(11) NOT NULL auto_increment,
  `ngram` varchar(100) NOT NULL default '',
  `belongs` int(11) NOT NULL default '',
  `repite` int(11) NOT NULL default '0',
  `percent` float NOT NULL default '0',
  PRIMARY KEY  (`id`),
  KEY `repite` (`repite`)
) TYPE=MyISAM;
```

Ačkoli stále je potřebná tabulka jako úložiště diskusních příspěvků a úložiště kategorií, pro samotné potřeby klasifikátoru již není třeba dalších tabulek.



Obr.10: Schéma databázové struktury pro Bayesovskou klasifikaci

8.2 Implementace metody neuronových sítí

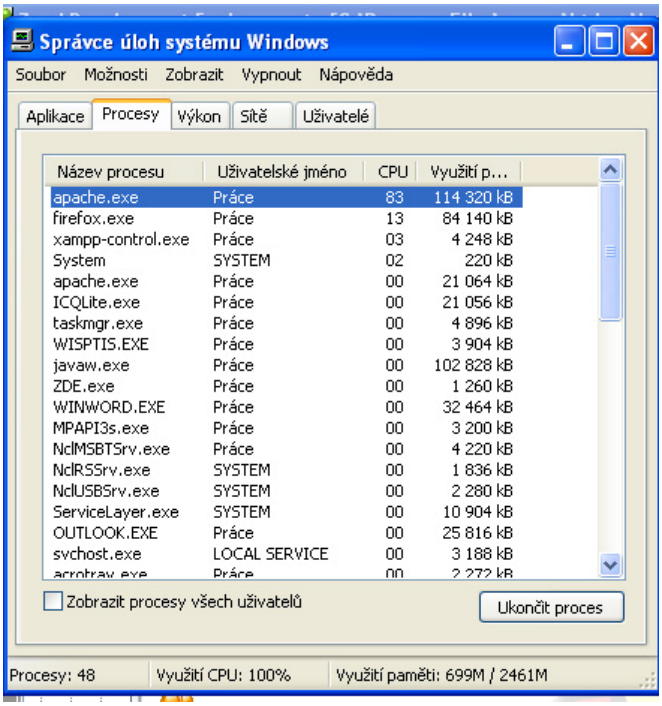
Po studiu klasifikačních metod jsem se na doporučení Ing. Romana Lukáše, Ph.D. rozhodl implementovat klasifikátor s využitím neuronových sítí. Tuto problematiku přehledně vysvětlují skripta do předmětu Základy získávání znalostí Klasifikace a predikce [9].

Algoritmus je postaven především pro klasifikaci a následnou predikci číselných hodnot. Bylo proto nutné najít způsob, kterým by bylo možno převést textové hodnoty na číselné vstupy. Rozhodl jsem se pro poměrně náročné řešení, které však mělo podle výpočtů a předpokladů dosahovat nejlepších výsledků.

Jako vstup do neuronové sítě použiji množinu všech slov, kde každé slovo bude mít svůj vlastní vstup. Při učení a později i samotné predikci vložím jednoduše hodnotu „1“ ke slovu, jenž se ve vkládaném příspěvku vyskytuje. Analogicky vložím opačnou hodnotu „0“ do výsledku, jenž se v příspěvku neobjevuje. Tato metoda mi tak dá vzniknout poměrně velké neuronové síti, která bude mít na vstupu unikátní posloupnost nul a jedniček. Posloupnost na vstupu bude různá, pokud se slova v příspěvku jakýmkoli způsobem liší. Pokud ale dojde pouze ke záměně pořadí slov v příspěvku, zůstane posloupnost stejná. Toto řešení je obecně pokládáno za vyhovující.

Důležitou součástí neuronových sítí je správné nastavení skrytých vrstev. Zatímco nastavení počátečních hodnot vah není nijak důležité, nastavení počtu skrytých vrstev se ukázalo jako zcela klíčová záležitost. Nastavením malého počtu skrytých vrstev dojde k nepřesnosti výpočtu, resp. síť se je nebude schopna naučit. Nastavením velkého množství skrytých vrstev může dojít k šumu. Navíc takové nastavení může být problematické vzhledem k výpočetní a paměťové náročnosti.

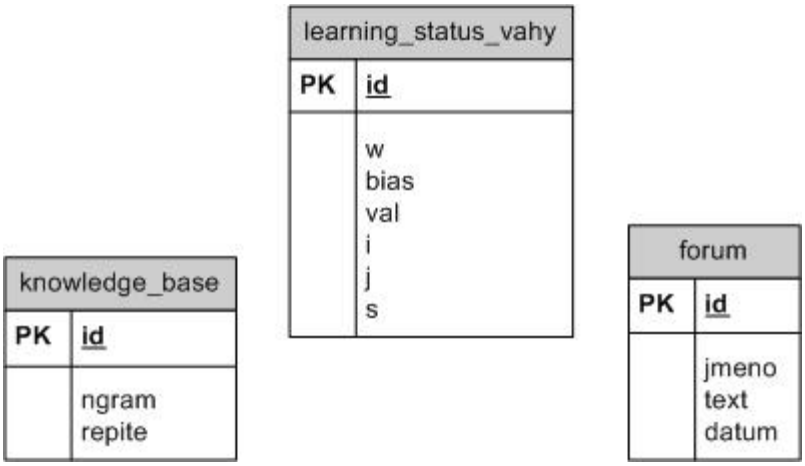
Při implementaci jsem se dostal do rozporu s bezpečnostním nastavením hostingu. Při větším počtu vstupů, skrytých vrstev a počtu iteračních kroků jsem se dostal za hranici povolené časové kvóty „*execution_time*“ a později i do problémů s množstvím přidělené paměti. Bylo tak nutné pracovat v umělých podmínkách, kde nebude záležet na nastavení těchto kvót. Rozhodl jsem se pro simulaci serveru Apache na osobním počítači. Po instalaci jsem změnil nastavení problematických kvót. Na obrázku 11 lze vidět vytížení prostředků ve fázi učení.



Obr.11: Využití systémových prostředků při učení neuronových sítí

Poslední důležitou částí neuronové sítě je výstupní vrstva. Tato vrstva při učení obsahovala binární kód reprezentující diskusní kategorii. Prakticky jsem použil binární vyjádření databázového ID z tabulky kategorií. Zbývající případné pozice byly doplněny nulami.

Při práci s konečnou množinou slov bylo třeba slova ošetřit tak, aby zbytečně nevznikaly neuronové sítě obrovských rozměrů. Experimentální metodou bylo zjištěno, že výsledek neuronových sítí se nezmění, pokud odebereme množinu slov s nižšími výskyty. R. Feldman a J. Sanger V knize The Text Mining Handbook [15] uvádějí, že výsledek neuronové sítě bude stejný při dosazení pouhých deseti procent nejčastěji používaných výrazů. Neuronovou síť tak nemusíme stavět nikterak velkou a stačí nám pouhých deset procent předních slov.



Obr.12: databázové schéma pro neuronové sítě

Na obrázcích 11 a 12 lze vidět rozdíl v databázové struktuře. Tabulka *knowledge_base* má u neuronových sítí pouze atributy detekující počet výskytů. Z těchto hodnot lze sestavit vstupy do neuronové sítě. Ve zvláštní tabulce (tabulka learning_status_vahy) zaznamenáváme výsledky výpočtů.

Fáze implementace dala vzniknout jedné třídě, která pracuje s neuronovými sítěmi. Tato třída dokáže neuronovou síť postavit, naučit a potom podle takové sítě predikovat. Třída *neuron*() má devět atributů. Popíši některé z nich:

- $pocetIteracnichKroku – určuje konečný počet iteračních kroků vedoucí k naučení neuronové sítě
- $w – trojdimenzionální vektor vah
- $bias – dvourozměrné pole, uchovávající hodnoty biasů
- $hodnoty – hodnoty jednotlivých neuronů *($pole[$s][$i] = (float)*)
- $err – chyba jednotlivých neuronů *($pole[$s][$i] = (float))*
- $s – celočíselný počet skrytých vrstev
- $vstup – jednorozměrný vektor, popisující vstupní hodnoty $x_1,x_2...x_n$
- $ocekavanyVystup – vektor očekávaných výstupních hodnot, nutný k naučení neuronové sítě
- $debugMode – hodnota, povolující výpis ladících informací

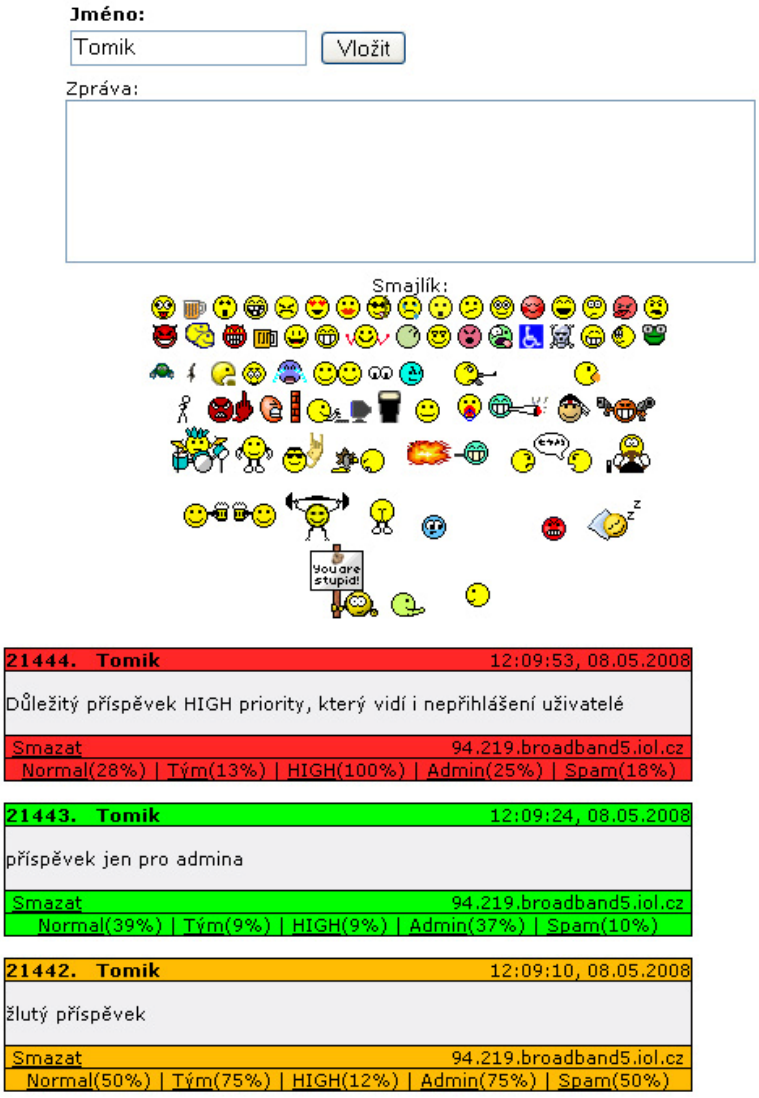
Konstruktor objektu přijímá dva argumenty a připraví objekt pro další práci. Jeho vstupními parametry jsou hodnoty vstupu a počet skrytých vrstev. Konstruktor zároveň nastavuje počáteční hodnoty váhového vektoru. Tyto hodnoty jsou libovolné. Metoda si tyto hodnoty sama přegeneruje.

Dále popíši základní metody pro práci s knihovnou. Metoda *NastavPocatecniHodnoty*() je volána konstruktorem a nastavuje trojrozměrné pole reprezentující rozsah skrytých vrstev a konkrétní polohu v rámci vrstvy. Náhodná hodnota je volána z intervalu <-1, 1> .

Zpracování neuronové sítě je dávkově rozděleno po vrstvách. Pro zpracování jedné vrstvy lze použít metodu *prepoctiSloupec*(), přijímající dva parametry: *vstup* a *pozice.* Tento parametr udává o kolikátou vrstvu se jedná. Parametr *vstup* není jen hodnotou vstupního vektoru. Jde o výstup předchozí vrstvy. Metoda přepočítá hodnoty neuronů na základě váhových koeficientů a biasů. Zmíněnou metodu volá metoda *predniChod*(), která má na starosti správné volání již popsané metody. Druhou výpočetní metodou je *zpetnyChod*(). Tato bezparametrická metoda zabezpečuje kompletní zpětný chod (Backpropagation). Cyklicky postupuje po vrstvách a upravuje váhové koeficienty na základě vypočtené chyby a hodnoty funkce. Pomocná metoda *sigmoida*($val) slouží k výpočtu hodnoty funkce neuronu sigmoida. Metoda *NeuronLerning*() spouští učící proces. Ostatní metody slouží především k práci s databází.

8.3 Testování vlastností a experimenty

Následující kapitola je věnována testům obou metod. Je velmi obtížné srovnávat obě metody stejnými testy. Bayesovská klasifikace je zcela jiná, než klasifikace pomocí neuronových sítí. V následujících kapitolách se pokusím srovnat především přesnost obou metod. Zajímavým srovnáním by bylo i srovnání složitosti zpracování.



Obr.13: Diskusní fórum[16] s implementovaným klasifikátorem textu

Ačkoli je zcela zřejmé, že by v tomto testu byl nesrovnatelně lepší Bayesův klasifikační algoritmus, rozhodl jsem se tento test nedělat. Test totiž může být zkreslen řadou faktorů, které nelze objektivně vytěsnit (implementační nároky, aktuální volné prostředky serveru aj.).

Klasifikátor byl implementován jako rozšíření mé bakalářské práce [16]. Součástí této práce bylo diskusní fórum hráčů florbalového týmu. Pro výběr tohoto fóra jsem se rozhodl po konzultaci s vedoucím

práce. Informační systém, který byl v rámci praktické části bakalářské práce vytvořen, užívá velké množství hráčů. V diskusním fóru jsem tak měl k dispozici velké množství dat vhodných ke klasifikaci. Diskusní fórum je přístupné na adrese http://www.mac-eagles.com/?s=gb. Pro aktivaci všech funkcí systém vyžaduje autorizaci. Bez přihlášení umožňuje pouze základní funkce a zobrazení některých příspěvků.

8.3.1 Testování vlastností Bayesova klasifikátoru

Z oficiálních zdrojů [5] vyplývá, že bayesova klasifikační metoda je schopna samostatné klasifikační práce až po naučení na dostatečně širokém vzorku dat. V případě antispamového filtru jde podle Paula Grahama [11] o dobu více než tří měsíců při každodenním použití. Po implementaci bayesovských klasifikátorů jsem bohužel neměl k dispozici takový časový potenciál. Bylo proto nutné zvýšit frekvenci vzkazů.

Za testovanou dobu klasifikátor uložil necelých 800 příspěvků, na kterých se zároveň i učil.

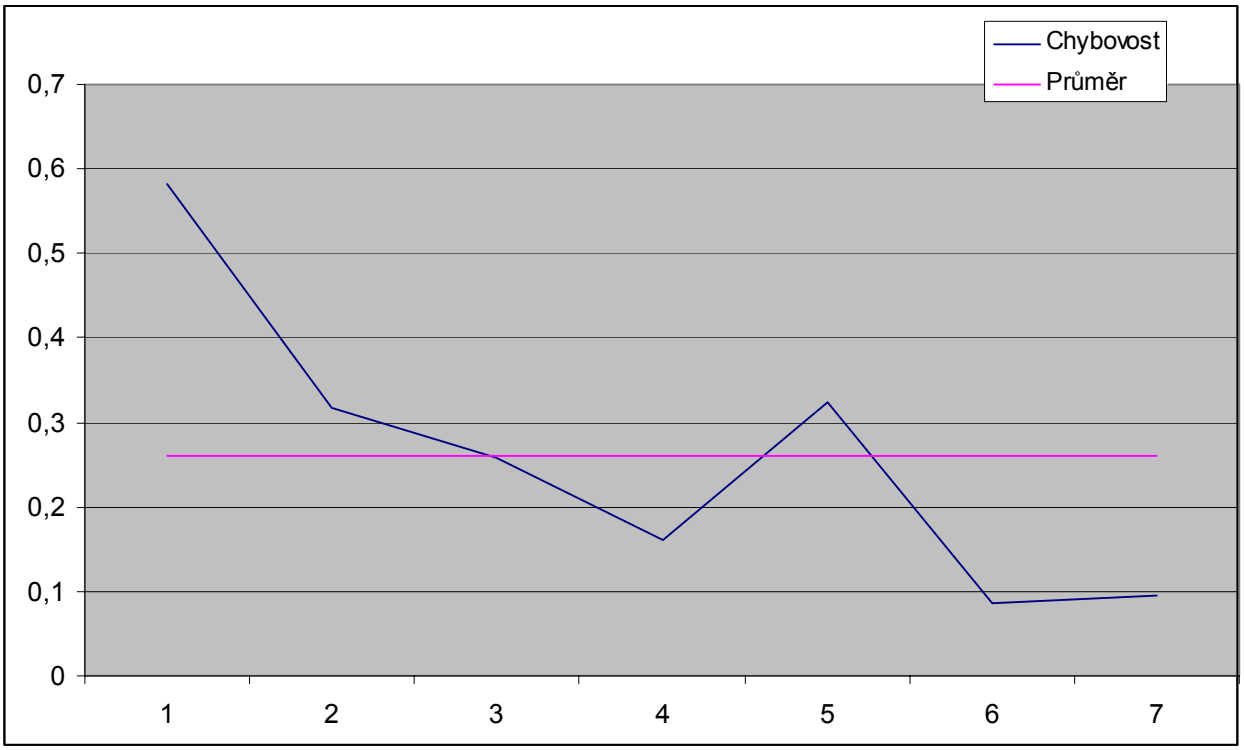
V současné době obsahuje slovník klasifikátoru cca 9500 výrazů, tj. slov v různých tvarech. Tomuto počtu pomohl nejspíš i fakt, že jsem při nasazení klasifikátoru zaregistroval stránku u „spamovacího“ robota. Ten každodenně vkládal umělé příspěvky do knihy vzkazů.

Vkládané příspěvky byly rozděleny do několika kategorií, které v podstatě vznikly již během předchozího života návštěvní knihy. Nově přidané diskusní příspěvky mohly nabývat těchto typů:

- Normální vzkazy – tento typ příspěvku je určen pro všechny vzkazy, které může vidět i neregistrovaný uživatel. Vnitřní pravidla florbalového týmu, kterému je diskusní fórum věnováno říká, že veřejné příspěvky nesmí obsahovat eticky nevhodná slovní spojení.
- Týmové vzkazy – typ, určený převážně pro mezitýmovou interní komunikaci. Vzkazy nejsou zobrazeny nepřihlášeným uživatelům. Z toho vyplývá, že není nutné striktně dodržovat pravidla určených pro typ *normální vzkaz*.
- Důležitý vzkaz – vzkaz tohoto typu je zobrazen varovnou červenou barvou, zobrazuje se i nepřihlášeným členům týmu a často se týká důležitých změn a oznámení. V minulosti byl tento typ vzkazu spojen i s emailovým notifikátorem.
- Vzkaz pro Admina – typ zprávy adresovaný administrátorovi. Často se jedná o administrativní informaci, či přání ke změně atributů stránek. Vzkaz je viditelný pouze pro administrátora webu.
- Spam – Nově přidaná kategorie, sdružující vzkazy ze zaregistrované spamovací služby.

8.3.1.1 Test průběhu a detekce chyb

Po dokončení fáze učení Bayesova klasifikátoru jsem vložil pomyslný milník, od kterého do současné chvíle přibylo necelých 800 vzkazů. Tyto příspěvky jsem rozdělil dle času vložení na sedm stejných celků. U každého celku jsem se snažil získat chybovost v rámci tohoto celku, bez ohledu na celistvý průměr. Z grafu na obr.14 vyplývá, že počáteční chybovost vzkazů byla neúměrně vyšší, než chybovost na konci testovaného období. Výsledek testu mohl být ovlivněn tím, že jsem testování fóra prováděl v době, kdy přispívající členové florbalového týmu ukončovali hráčskou sezonu. V tomto období se málo různí témata příspěvků a vzkazy jsou si navzájem velmi podobné. Na ose x se v grafu nachází vyjádření chyb, pohybující se v intervalu <0,1>. Na ose y je patrné rozložení jednotlivých skupin příspěvků.



Obr.14: vyjádření chyby v závislosti na jednotlivých skupinách příspěvků

Tento jev popisuje velmi dobře i následující tabulka (1). Tabulka obsahuje čísla v intervalu <0,1>, udávající velikost chyby. Průměrná hodnota chyby pro všechny skupiny je 0,260227.

Skupina	Velikost chyby
1	0,58132
2	0,317821
3	0,257601
4	0,15972
5	0,32469
6	0,08621
7	0,09423

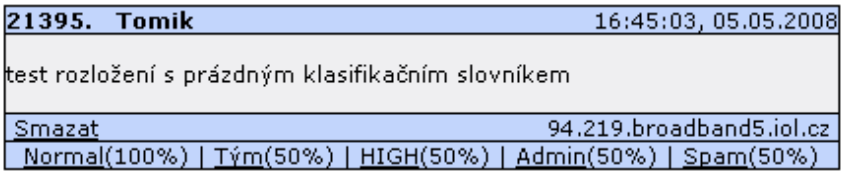
Tab.1: chybovost jednotlivých skupin příspěvků

Z tabulky 1 je dobře patrné, že se třetí skupina příspěvků blíží k celkovému aritmetickému průměru. Později se průběh chybovosti ustaluje. V poslední fázi je správnost klasifikace zatížena chybou menší než 10%. Toto číslo je obecně považováno jako velmi dobrý výsledek. [11]

8.3.1.2 Test generalizace

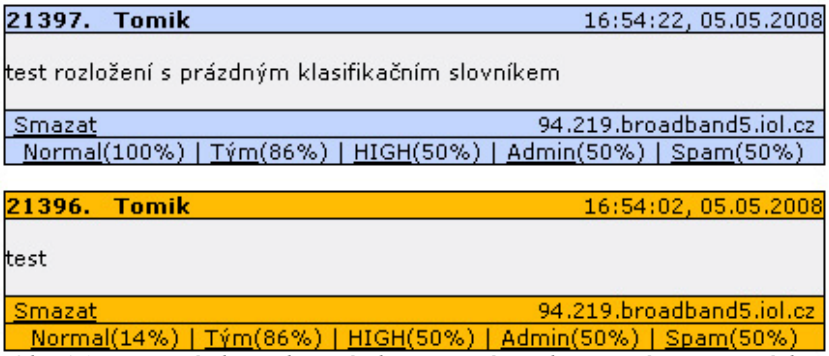
Tento typ testu je na rozdíl od neuronových sítí poměrně velmi těžko proveditelný. Důvodem je úzká spojitost Bayesovy metody s procentuální hodnotou naučených dat. Měření i teoretické základy ukazují, že pokud bude výskyt slov z příspěvku v databázovém slovníku chybět, bude navrácena pro všechny kategorie stejná hodnota. Pokud totiž v testované kategorii slovo chybí, vrací se neutrální hodnota 50%. Celý vzkaz s neznámými slovy tak dostane příznak kategorie, která má vyšší prioritu. V mém případě je tato priorita dána pořadím kategorií v databázi.

Obr.15 ukazuje diskusní příspěvek s neznámými použitými slovy. Kategorie „normální“ je hodnocena úplnou příslušností, protože po vložení byla slova naučena. Použitá slova tak dostala příslušnost ke vzkazu typu „normální“.



Obr.15: Přiřazení příspěvku v nenaučeném slovníku

Přesně podle očekávání je příslušnost k jednotlivým kategoriím rovna padesáti procentům. Protože jde o nenaučený vzkaz. Zkusme nyní vložit slovo „test“ do skupiny příspěvků pro tým a poté vložit stejný vzkaz znovu (obr 16).



Obr.16: naučení slova do jiné skupiny příspěvku změní poměr příslušnosti

Na horním příspěvku na obr.16 vidíme navýšení pravděpodobnosti u druhé skupiny (oranžová barva). Příspěvek #21397 byl správně rozpoznán do skupiny normálních příspěvků. Pravděpodobnostní rozložení již ale není stejné jako na obr.15.

8.3.2 Testování vlastností neuronových sítí

Z již popsaných důvodů nebylo vhodné neuronovou síť testovat na datech, která byla získána v praxi. Neuronová síť by byla příliš vytížena a testování by bylo časově velmi náročné. Rozhodl jsem se tedy dělat testy s menší neuronovou sítí. Jako optimální byla vybrána konfigurace x-3-1, kde *x* je velikost vstupního vektoru. Ta byla určena v rozsahu 5,10,15. Počet opakování neuronové sítě (iterací) byl ustanoven na počtu 2000. Posledním důležitým parametrem je počet vzorků, na kterých jsem zkoušel neuronovou síť učit. Tento počet byl stanoven na 127.

8.3.2.1 Test průběhu a detekce chyb

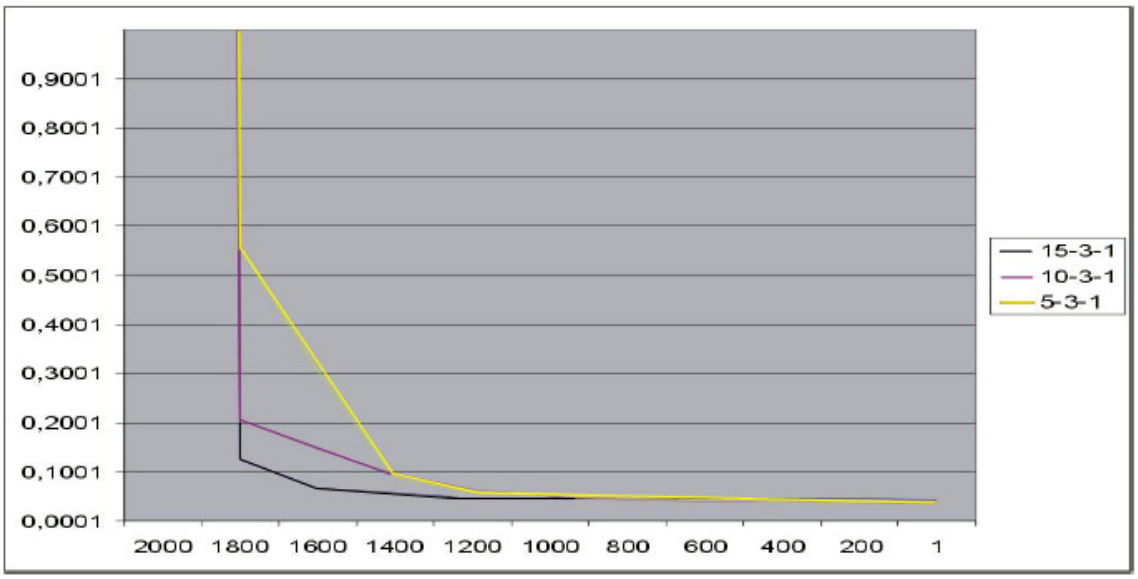
Z grafu (obr. 17) je zřejmé, že chyba je největší u sítě s nejmenším vstupním vektorem a nejmenší u sítě s největším vstupním vektorem. O tom se lze přesvědčit i z tab.2. Z této tabulky je také patrné, že průměrná chyba sítě klesá s každou další iterací.

	5-3-1	10-3-1	15-3-1
maximální chyba	0,001034	0,000992	0,000605
průměrná chyba	0,00026418	0,000214892	0,000126352

Tab.2: Průměrná a maximální chyba pro jednotlivé vstupní vektory

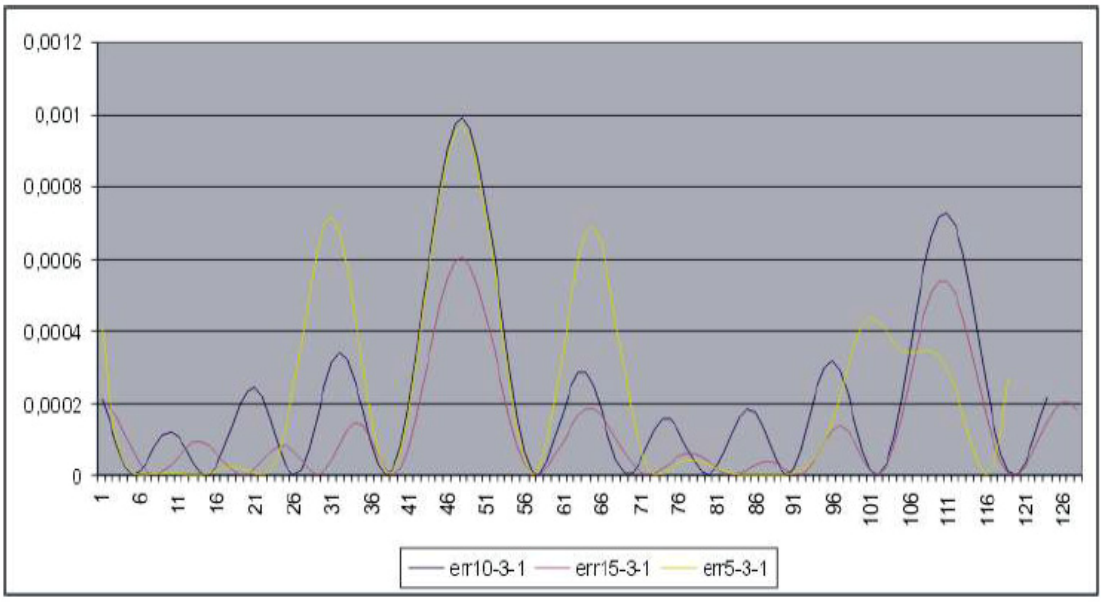
Při učení předkládáme síti vzor po vzoru a sledujeme její odpověď. V případě špatné odpovědi se váhy upraví. Takto se předloží všechny vzory v několika iteracích (epochách). Učení je charakterizováno chybou při učení, které se snaží chybu minimalizovat. Průběh chyby při učení je znázorněn na grafu (obr.15). Z tohoto grafu je zřejmé, že chyba exponenciálně klesá, až ke svému minimu. Z tohoto grafu je také patrný vliv vstupních parametrů na síť. Síť s větším počtem vstupů se rychleji učí, dokonce s menší chybou (viz dále).

Osu *x* na grafu v obr.17 tvoří jednotlivé iterace. S každou iterací se upraví chyba každého vstupu. Na ose *y* je potom vyvedena odchylka od očekávané hodnoty tzn. velikost chyby.



Obr.17: Typický průběh chyby při učení, popis vyjadřuje konfiguraci sítě

Je zcela zřejmé, že chyba bude zpočátku záviset i na počátečním nastavení neuronové sítě. Velikost chyb jednotlivých vzorků ukazuje obr.18.



Obr.18: Chyba výstupu v jednotlivých chybových vektorech

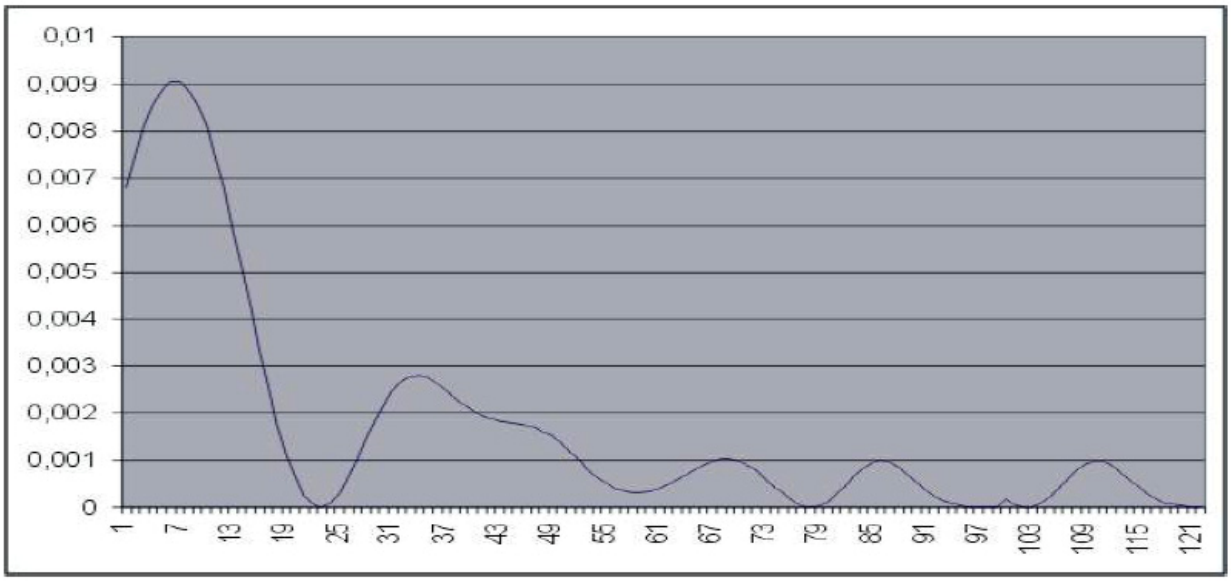
8.3.2.2 Test generalizace

Tento test je nutný při neúplných trénovacích množinách. Ukazuje jak je síť schopná používat znalosti z již naučených vzorů pro chybějící nebo poškozené vzory. Byla použita síť 15-3-1, tedy chyby můžeme srovnat s tab.2. Tab.3 ukazuje maximální a průměrnou chybu při generalizaci. Při srovnání je zřejmé, že chyba je výrazně větší, přibližně o řád.

	15-3-1
maximální chyba	0,009059
průměrná chyba	0,001760282

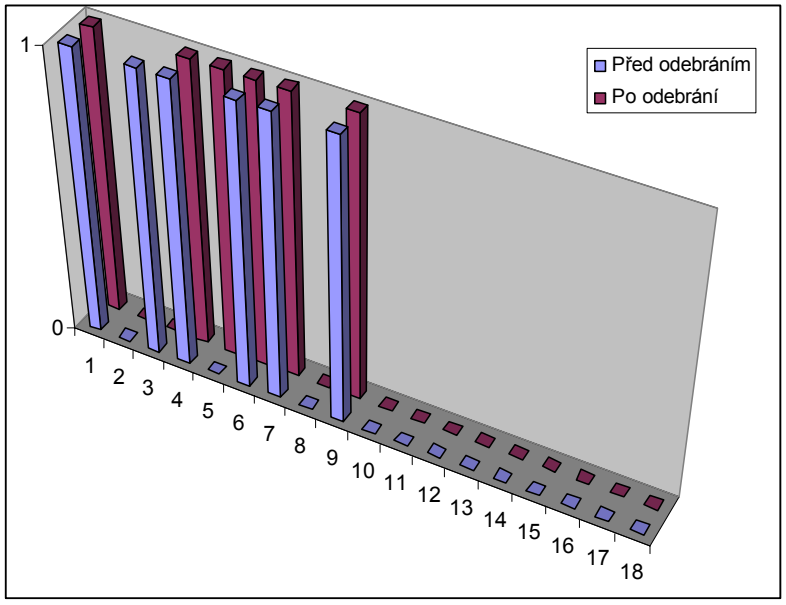
Tabulka 3: statistické údaje o chybě při generalizaci

Test jsem provedl na již naučené síti po odebrání 40 libovolných vzorků. Tento počet vzorků tvoří necelou třetinu. Průběh detekce chyb je znázorněn na obr.19.



Obr.19: Chyba po odebrání 40 vzorků

Po odebrání jedné třetiny vzorků bylo zřejmé, že se tato změna projeví i do výsledného vektoru. Nechal jsem proto rozpoznat znovu několik vzorků. Ani jeden ze znovu testovaných vzorků se neshodoval s původním testovaným vektorem. Důvodem je neschopnost zvolené metody rozpoznat výstup, který neočekává. V minulé kapitole jsem popsal, že výstupem neuronové sítě je binární zápis databázového ID. Pokud ale z důvodu odebrání naučených vzorků změníme výstupní nastavení bitů, je možné, že již nenalezneme hledaný binární záznam. Obr.20 ukazuje jednu z konfigurací výstupních bitů před odebráním a po odebrání.



Obr.20: Binární konfigurace výstupního vektoru

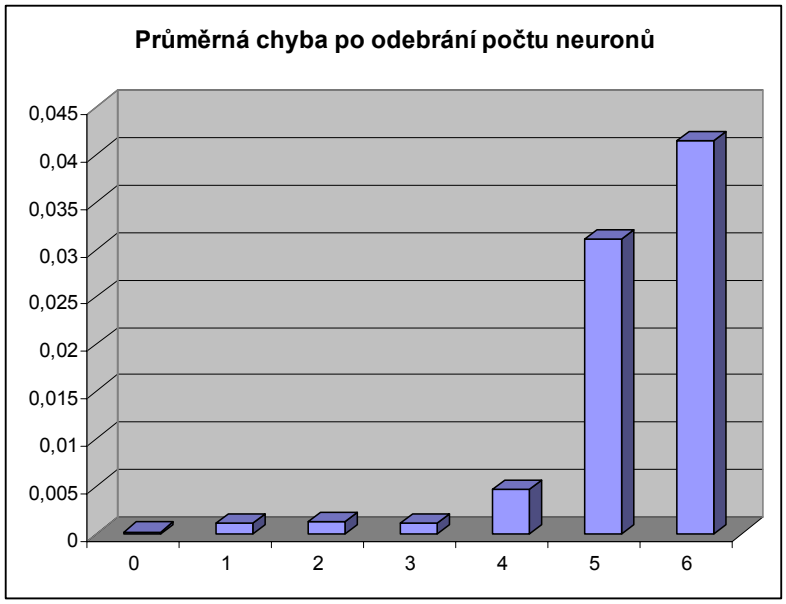
Vidíme, že bity na třetím a pátém místě si neodpovídají. Je proto zřejmé, že v tomto případě neuronová síť zařadí příspěvek do špatné kategorie. Jinak nastavená konfigurace či odebrání menšího počtu naučených vzorků by mohlo mít za následek správné zařazení. Hodnoty od pozice devět jsem proto doplnil nulami tak, jak jsem popisoval v minulé kapitole.

8.3.2.3 Test robustnosti

Test robustnosti spočívá v naučení sítě a postupném odebírání neuronů. Z tabulky a následného grafu na obr.21 je zřejmé, že nejmenší chybu má plně naučená síť konfigurace 15-10-1. Tato chyba se dále zvětšuje do odebrání 3 neuronů. Po odebrání 4 neuronů se chyba rapidně zvětší a po odebrání 6 neuronů si již síť naučené vzory nevybavuje správně. Výsledky ukazuje tab.4. Je patrné, že průměrná a maximální chyba sítě roste s odebraným počtem neuronů.

Počet odebraný neuronů	Průměrná chyba po odebrání	Maximální chyba
0	0,000213314	0,000996
1	0,001147468	0,00606
2	0,001305744	0,006673
3	0,001076314	0,005053
4	0,004717872	0,038005
5	0,03113969	0,194842
6	0,041421864	0,195753

Tab.4: Tabulka chyb sítě při ubírání neuronů



Obr.21: Znázornění průměrné chyby v závislosti na počtu odebraných neuronů

9 Závěr

Při značném množství dat, které je v moderní době nutné zpracovat, je absolutně nemožné vystačit s jednoduchými metodikami zpracování textu. Je proto důležité, mít osvojené postupy, kterými lze ruční klasifikaci dat urychlit. Zde ale často narážíme na mnohá úskalí, která se s automatickým zpracováním dat pojí. Specifickým druhem klasifikace textu je klasifikace příspěvků diskusního fóra. Pro nutnou rychlost zpracování takových dat není možné použít některých, jinak povolených postupů.

Cílem první části diplomové práce bylo teoretické zvládnutí problému. Zavedl jsem matematické modely, kterými jsem definoval všechny důležité pojmy v oblasti klasifikace a zhodnotil metody pro přípravu dat před klasifikací. V kapitole věnované Bayesovské klasifikaci jsem již definované obecné modely pro klasifikátory použil na přiblížení Bayesovské problematiky. Správnost modelu Bayesovské pravděpodobnosti jsem pak mohl ověřit na pokusu hodu mincí. Ten jsem porovnával s výsledky výpočtů v prostředí Matlab.

Jako druhou metodu pro klasifikaci diskusních příspěvků jsem si zvolil metodu s využitím neuronových sítí. Pokusil jsem se zavést matematický model nejjednodušší neuronové sítě – perceptronu. Na jeho příkladu jsem potom definoval principy, které pro neuronové sítě platí.

Třetí metodou, která nebyla součástí implementace a praktické části, byla Support vector machine. Při studiu klasifikátorů textu jsem se podrobně seznámil i s touto novou metodou. Velmi mě zaujala její efektivita a jednoduchost klasifikace. Z kapacitních důvodů však nedošlo k její implementaci. Nepodařilo se mi studiem objevit dostatek implementačních řešení, proto považuji implementaci této metody za vhodnou možnost, jak svou práci rozšířit případným další výzkumem.

V praktické části diplomové práce jsem popsal implementaci obou metod. Obě klasifikační metody jsou implementovány jako samostatné objektové celky, schopné učení i klasifikace do tříd. Veškerý zdrojový kód, který je v těchto kapitolách popisován je součástí přiloženého CD. Nejdůležitější metody obou objektů jsou vloženy a popsány v příloze.

Závěrečná kapitola se věnuje srovnání a testům jednotlivých metod. Srovnání obou metod je velmi obtížné, neboť její výsledky nevykazují srovnatelné hodnoty. Jako lepší metoda pro diskusní fórum se však ukázala Bayesova klasifikační metoda. Ačkoli jsem zaznamenal problémy ve fázi učení, vykazuje výrazně rychlejší a přesnější výsledky, než zpětné neuronové sítě.

Diplomová práce pro mne byla přínosem. Seznámil jsem se s velkým množstvím klasifikačních metod a s řešením problémů při jejich implementaci.

Literatura

[1] Anděl, J.: Matematická statistika. Praha, SNTL/ALFA 1996.

[2] Lubínek, R.: Bayesovská klasifikace v praxi. Computerworld, 16, 2005, s.43.

[3] Han J., Kamber M.: Data Mining: Concepts and Techniques. Second Edition. Elsevier Inc., 2006, 770 p., ISBN 1-55860-901-3.

[4] Rychlý, M.: Klasifikace a predikce. [online], [cit. 2008-05-03]. Dostupné na URL: <http://www.fit.vutbr.cz/~rychly/classification-and-prediction/classification-and-prediction.pdf>

[5] Wikipedie: Paul Graham. [online], [cit. 2008-05-03]. Dostupné na URL: <http://en.wikipedia.org/wiki/Paul_Graham>

[6] Drábek, O., Seidl, P., Taufer, I., Umělé neuronové sítě. [online], [cit. 2008-05-07]. Dostupné na URL: <http://www.chemagazin.cz/Texty/CHXVI_1_cl3.pdf>

[7] Gunn, R.: Support Vector Machine for Classification and Regression. University of Southampton, 1998.

[8] Boser, B. E., Guyon I. M., Vapnik V. N.: A training algorithm for optimal margin classifiers. In D. Haussler. Pittsburgh, PA, ACM Press 1992. s.144-152.

[9] Lukáš, R.: Klasifikace a predikce [skriptum předmětu ZZN], Fakulta informačních technologií VUT, Božetěchova 2, 612 66 Brno 2007.

[10] Krátký, D.: Proč je Bayesovo filtrování nejefektivnější antispamovou metodou. Praha, White Paper 2006.

[11] Graham, P.: A plan for Spam. [online], [cit. 2008-03-14]. Dostupné na URL: <http://www.paulgraham.com/spam.html>

[12] Fisher, R.: The Genetical Theory of Natural Selection. East Finchley 1936.

[13] Widrow, S. D.: Adaptive Signal Processing. New Jersey 1985.

[14] Reuters: Test Collections. [online], [cit. 2008-05-10]. Dostupné na URL: <http://www.daviddlewis.com/resources/testcollections/reuters21578>

[15] Feldman, R., Sanger, J.: The Text Mining Handbook. Cambridge University Press 2007.

[16] Margold, T.: Informační systém sportovního klubu [bakalářská práce], Fakulta informačních technologií VUT, Božetěchova 2, 612 66 Brno 2006.

[17] McCulloch, W., Pitts W.: A Logical Calculus of Ideas Immanent in Nervous Activity. NewYork 1943, str.115-133.

[18] Nils, N.: Principles of Artificial Intelligence. Boston, Tioga Publishing Company, 1965.

Seznam příloh

Diplomová práce neobsahuje žádné přílohy.