VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ROZPOZNÁVÁNÍ RUČNĚ PSANÝCH ČÍSLIC METODOU K-NEAREST NEIGHBOR

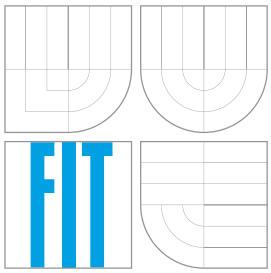
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

VLADIMÍR HORKÝ

BRNO 2009

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ROZPOZNÁVÁNÍ RUČNĚ PSANÝCH ČÍSLIC METODOU K-NEAREST NEIGHBOR

HANDWRITTEN DIGIT RECOGNITION USING K-NEAREST NEIGHBOR

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — VLADIMÍR HORKÝ
AUTHOR

VEDOUCÍ PRÁCE — Ing. OLDŘICH PLCHOT
SUPERVISOR

BRNO 2009

Abstrakt

Tato práce se zabývá problematikou rozpoznávání ručně psaných číslic. Rozebírá problémy při řešení rozpoznávání metodou K-nejbližších sousedů. Část práce popisuje postup při návrhu a implementaci této metody.

Abstract

This paper describes problems of handwritten digit recognition. Discuss about problems with solution of recognition by algorithm K-nearests neighbor. In second part there is described design and implementation of this method.

Klíčová slova

OCR, ICR, Skeletonizace, Ztenčování, PCA, algoritmus K-nejbližších sousedů, K-NN

Keywords

OCR, ICR, Skeletonization, Thinning, PCA, algorithm of K-nearest neighbor, K-NN

Citace

Vladimír Horký: Rozpoznávání ručně psaných číslic
metodou K-nearest neighbor, bakalářská práce, Brno, FIT VUT v Brně, 2009

Rozpoznávání ručně psaných číslic metodou K-nearest neighbor

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Oldřicha Plchota. Další informace mi poskytl Ing. Michal Hradiš. Uvedl jsem všechny zdroje, ze kterých jsem čerpal.

.....
Vladimír Horký
19. května 2009

Poděkování

Děkuji vedoucímu mé práce panu Ing. Oldřichu Plchotovi za vstřícný přístup, za jeho připomínky a návrhy, kterými mě podporoval v práci.

Obsah

Kapitola 1

Úvod

Vzhledem k masivnímu rozvoji informačních technologií dochází i k modernizaci přístupu k psanému a tištěnému textu. To vede společnosti čím dál více k nahrazení starých papírových kartoték za kartotéky elektronické. Jejich nespornou výhodou oproti textu „na papíře"je totiž možnost použití automatizace, schopnost rychlého a efektivního zpracování dokumentu, snadný převod do papírové podoby, možnost sdílet a využívat informace, které jsou v dokumentech obsaženy apod.

V dnešní době si život bez elektronického textu nedokážeme ani představit, ať už jej používáme k posílání emailů, textových zpráv(SMS) či při vývoji softwaru . . .

OCR [15] je zkratkou anglického slova Optical Character Recognition - Optické rozpoznávání znaků. Je metodou/technologií elektronického čtení znaků a jejich převádění do digitální formy, které lze dále zpracovávat počítačem. Tato technologie rozpozná pouze tištěné a strojopisné písmo. OCR je dnes běžně dodáváno s většinou scannerů. Další uplatnění najdeme například při kontrole dopravy, kde se klasifikace používá při detekci a rozpoznávání značek aut.

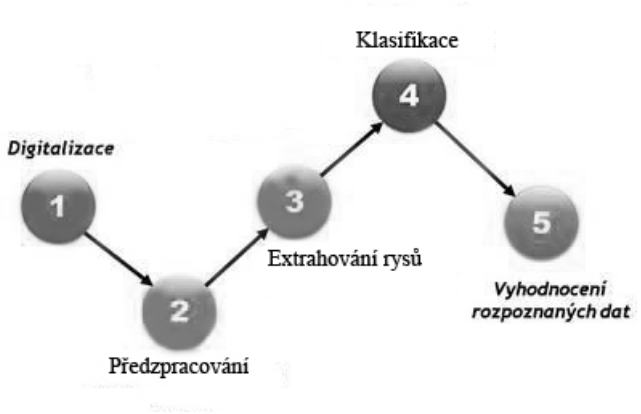
Metoda která rozpozná ručně psané písmo je Inteligentní rozpoznávání znaků (ICR) z anglického slova Inteligent Charakter Recognition. Jde o specifičtější formu OCR, která si poradí i s odlišným fontem a stylem písma.[15]
ICR je používána například na poštách pro třídění zásilek podle poštovního směrovacího čísla nebo jste se s ní mohli setkat v bankovnictví při předávání podpisového vzoru apod.

Tato práce se zaobírá ICR, konkrétně rozponáváním ručně psaných číslic. Je zaměřena zejména na použití algoritmu k-nearest neighbor. V kapitole 2 jsou shrnuty základní poznatky o zpracování obrazu a ztenčovací metodě. Kapitola 3 je věnována používaným rysům pro popis znaků. V kapitole 4 se obecně dozvíte něco o klasifikátorech, podrobněji je zde rozebrán hlavně algoritmus nejbližších sousedů a nahlédneme také k problematice neuronových sítí. Obsahem 5. kapitoly jsou navržená řešení a jejich výsledky. Celou práci uzavírá 6. kapitola, ve které naleznete shrnutí celé práce.

1.1 OCR/ICR

Prvním krokem v rozpoznávání vzorů je „naučit"počítač, jaké jsou třídy vzorů. Učení probíhá formou ukázek příkladů všech tříd, kdy si stroj vytvoří svůj prototyp všech tříd. V průběhu rozpoznávání jsou znaky porovnávány s naučenými daty a podle jejich nejbližší shody jsou přiřazeny do odpovídající třídy.

Typické OCR/ICR systémy se skládají z několika komponent:



Obrázek 1.1: Komponenty ICR

Při rozpoznávání znaků uvažujeme dvě formy dat a to na základě způsobu jejich získání. Data obecně dělíme na online a offline data. On-line data získáváme přímo při psaní textu uživatelem a naopak off-line data získáme až po napsání uživatelem.

1.1.1 On-line data

K zisku takovýchto dat slouží speciální pero a digitalizační tablet. Tablet zachycuje v daném rozlišení pohyby špičky pera po povrchu. Data jsou snímána po celou dobu kontaktu pera s povrchem. Při tomto způsobu získávání dat má velký vliv na kvalitu zejména použitý materiál (tablet, plochý displej - to se projevuje jiným stylem písma apod.)

1.1.2 Off-line data

Jak jsem již zmínil, off-line data se získávají až po jejich napsání. To znamená, že text je nejdříve napsán obyčejným perem na papír a posléze je digitalizován pomocí skeneru. Skenery přiřazují každému pixelu jednu binarní hodnotu (0 = černá , 1 = bílá) pokud se jedná o černobílé provedení nebo hodnotu 0-255, která určuje odstíny šedi nebo intenzitu barevné hloubky každé barevné složky. Jedná-li se např. o barevný model RGB, budou tyto hodnoty tři. Jedna pro intenzitu červené složky, druhá pro intenzitu zelené složky atd.

Dále v textu se budeme zabývat rozpoznáváním spojené pouze s off-line daty.

Kapitola 2

Zpracování obrazu

2.1 Co je to obraz

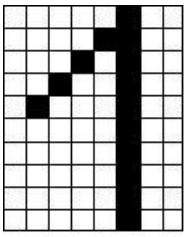
V oblasti zpracování obrazu se vědci zabývají zejména analýzou digitálního obrazu, který můžeme získat např. jako digitální signál ze skeneru nebo z digitální kamery či převodem analogového signálu z kamery.

Tento obraz je poté tvořen elementárními částmi zvané pixely. Pixely jsou uloženy do mřížky zvané pixmapa (bitmapa jde-li o binární data) neboli rastr. Každý pixel nese informaci o své barevné hloubce (viz. [2]):

1. binární data:
   Pixel je uložen na 1 bitu a nabývá hodnot 0 = bílá nebo 1 = černá.

2. grayscale data:
   Pixel je uložen na 8 bitech a nabývá hodnot 0 - 255 (černá - bílá).

3. colour data:
   Pixel je uložen na 24 bitech a nabývá hodnot 0 - 255 pro každou barvu.



Obrázek 2.1: Uložení znaku jedničky do jedno-bitového rastru

2.2 Předzpracování

Obrázek může obsahovat různé nežádoucí efekty jako je např. šum . . . K redukci šumu se používají tzv. **vyhlazovací** filtry. Pro vyhlazování se používají např. binomiální, box, mediánové, Kuwahara filtry, které využívají odlišné algoritmy.

Obecně spočívá filtrace v modifikaci obsahu pixelu s ohledem na nejbližší okolí. Úprava se nejčastěji provádí pomocí čtvercové matice, tzv. **filtrační matice**, kdy se s prvky nejbližšího okolí pixelu a prvky filtrační matice provádí určité operace. Nejjednodušší filtrací je násobení prvků filtrační matice s prvky nejbližšího okolí upravovaného pixelu.

Předzpracování obvykle zahrnuje i proces normalizace. Je to proces, který eliminuje rozdíly mezi obrázky(např. velikost, šum, osvětlení . . .).

2.3 Prahování

Pro extrakci příznaků potřebujeme obraz v binární podobě (viz. 2.1). Ten získáme např. prahováním. Prahování je metoda, která převádí intenzitu pixelu na minimalní respektive maximální hodnotu podle zvoleného prahu. V našem případě použijeme tzv. inverzní prahování, kdy hodnotě menší jak práh nastavíme hodnotu maximální a naopak. Algoritmus pro obrázek v odstínech šedi (grayscale) vypadá následovně (alg. 2.3.1):

```
void Prahovani() {
     zvol práh T ∈ <0;255>
     for(přes všechny pixely p v pixmapě) {
        if(intenzita pixelu p > T) { p = 0; }
        else { p = 255; }
     }
}
```

Algoritmus 2.3.1 *Pseudokód metody prahování*

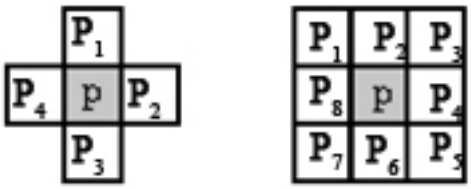
2.4 Skeletonizace - ztenčování

Skeletonizace je metoda, která se aplikuje na binární obraz. Cílem teto metody je nalézt skeleton (topologickou kostru) objektu. Skeletonem rozumíme zjednodušený tvar oblasti při zachování její tvarové charakteristiky, čímž redukujeme množství informací. Pro představu zde uvádím příklad (viz. obr. 2.2). V našem případě je originální obraz redukován na linii o šířce jednoho pixelu.

Obrázek 2.2: Rukou psaná číslice 1 a výsledek po skeletonizaci

Nejdůležitějším krokem u metod skeletonizace je rozhodnout, který pixel v obrazu je redundantní. Při posuzování se využívá tzv. sousedství pixelu. O sousedství jde pouze tehdy, je-li vzdálenost mezi pixely rovna 1. Ve většině algoritmů se využívá 8-okolí, ale můžeme využít i 4-okolí. Pojem sousedství demonstruje obrázek (obr. 2.3). My se nadále budeme zabývat pouze algoritmy počítající s 8-okolím. V tomto případě máme tedy $2^8 = 256$

možných kombinací černých a bílých pixelů v sousedství. Tyto kombinace jsou uvedeny v knize [2]. Z toho plyne, že nejdůležitějším a zároveň nejtěžším krokem je rozhodnout, kterou kombinaci vybrat pro odstranění redundantního pixelu.

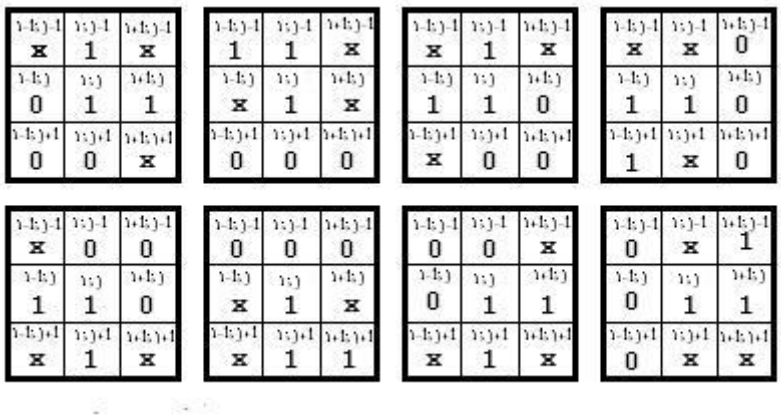


Obrázek 2.3: Ukázka 4-okolí a 8-okolí. $P_1 - P_8$ jsou sousedy pixelu p

2.4.1 Transformace hit or miss

> „Transformace tref či miň (angl. hit or miss) je morfologický operátor[1], který indikuje shodu strukturního elementu a části obrazu." [7]

Strukturní element je vzorem, který se vyhledává. Kromě využití při ztenčování nám poslouží i při vyhledávání rohů, hranic objektů.... Existuje několik posloupností strukturních elementů. V našem případě byly využity strukturní elementy z tzv. Golayovi abecedy (viz obr.2.4). Hodnota 1 v matici požaduje příslušnost k danému objektu, hodnota 0 požaduje příslušnost k pozadí a tam kde je hodnota x, se pixel srovnávání neúčastní. Pokud tedy dojde ke shodě vzoru s obrazem, je možné pixel p odstranit (tzn. změnit hodnotu pixelu na hodnotu pozadí). Operace ztenčování je idempotentní, tzn. že po určitém kroku se přestanou měnit. Pokud po sobě následujících iteracích zůstává obraz nezměněn, je algoritmus ukončen.



Obrázek 2.4: Strukturní elementy Golayovi abecedy

[1] Je to operace, založená na teorii množin. Hit or miss je založena na operátoru *eroze*. Existují ještě další 4 základní operátory: *dilatace, sjednocení, průnik a negace*

2.4.2 Další metody

Kromě metod, které přímo porovnávají sousedství s velkým množstvím vzorů, existují metody rozhodující o redundantnosti pixelu s odlišnými kritérii. Tyto metody mají obrovskou výhodu v tom, že pro rozhodnutí nepotřebují žádný vzor nebo využijí nesrovnale menší počet vzorů. Algoritmy, které jsou uvedeny níže počítají s těmito specifickými čísly:

$A(P) =$ Toto číslo uvádí počet přechodů mezi bílým a černým pixelem. Pixely jsou procházeny v pořadí daném indexem (viz. obr. 2.3).

$B(P) =$ Číslo uvádějící počet černých pixelů v sousedství.

$ORD(\text{výraz}) =$ Je funkce vracející hodnotu 1 pokud je výraz pravdivý, hodnotu 0 v opačném případě.

$$C(P) = ORD(\neg P_2 \wedge (P_3 \vee P_4)) + ORD(\neg P_4 \wedge (P_5 \vee P_6)) + ORD(\neg P_6 \wedge (P_7 \vee P_8)) + ORD(\neg P_8 \wedge (P_1 \vee P_2))$$

$P_1, P_2 \dots P_8 =$ Pixely okolí pixelu p (viz. obr. 2.3)

Nyní se můžeme podívat na samotné algoritmy [2]:

1. **Algorithm of Lü and Wang:**
   Tato metoda se o redundantnosti pixelu rozhoduje ve dvou subiteracích. Pixel p je redundantní pokud splňuje následující podmínky:

   (a) $3 \leq B(P) \leq 6$

   (b) $A(P) = 1$

   (c)
   - První subiterace: $P_2 \wedge P_4 \wedge P_6 = FALSE$ and $P_4 \wedge P_6 \wedge P_8 = FALSE$
   - Druhá subiterace: $P_2 \wedge P_4 \wedge P_8 = FALSE$ and $P_2 \wedge P_6 \wedge P_8 = FALSE$

   Pokud je porušena jediná podmínka, pixel nemůže být vymazán.

2. **Algorithm of Hall and Guo:**
   Narozdíl od metody Algorithm of Lü and Wang používá tato metoda pouze jediný ztenčovací operátor, který obsahuje tyto tři podmínky:

   (a) $C(P) = 1$

   (b) $(P_1 \wedge P_3 \wedge P_5 \wedge P_7) \vee (P_2 \wedge P_4 \wedge P_6 \wedge P_8) = FALSE$

   (c) $B(P) > 1$

   Podmínka a zaručuje nesmazání pixelu pokud leží uprostřed objektu, podmínka b zaručuje smazání hraničních pixelů a c zabranuje zkrácení konců linií v objektu.

Kapitola 3

Extrakce příznaků

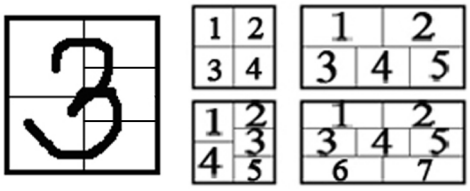
Extrakce příznaků patří k nejproblematičtější části při rozpoznávání vzorů. Cílem je získat základní charakteristiky každého znaku. Musí být diskriminativní, abychom mohli určit do jaké třídy znak zařadit. Většina metod získává popis znaků přímo z rastru obrazu, jiné metody se zaměří na určité rysy znaku a nevšímají si dalších nedůležitých atributů.

3.1 Intenzita pixelů

Asi nejjednodušší metodou při extrakci příznaků je použití intenzity jednotlivých pixelů. Platíme za to, ale velkými početními nároky a v případě, že má obrázek velké rozměry je tado metoda nepoužitelná. Určitou inovací těchto rysů by bylo použití metody PCA, o které se dočtete dále v textu. Podstatné pro tuto chvíli je, že touto metodou dokážeme zredukovat množství příznaků a zvýšit tak výpočetní výkon klasifikátoru.

3.2 Zoning

Tato metoda je založena na podobné filosofii jako čtení znaků lidským mozkem. Lidský mozek rozeznává znaky podle určitých částí znaku. Proto je obrázek rozdělen na několik částí, které se mohou navzájem překrývat či nikoli. Rysem znaku je poté brána hustota bodů náležicích znaku v dané oblasti. Touto metodou se zabývaly mnohé studie, které zkoumaly, které oblasti jsou pro charakteristiku znaku klíčové. Vice informaci naleznete v literatuře[12, 14]. Následující obrázek (obr. 3.1) demonstruje nejvíce používaná rozdělení do oblastí.



Obrázek 3.1: Číslo 3 v zóně 5Vertical a další zoning oblasti: Z = 4, Z = 5Horizontal, Z = 5Vertical a poslední Z = 7

3.3 HU momenty

Další používanou technikou při extrakci příznaků je použití momentů. Za rys znaku je brán např. moment tmavých znaků k vybranému počátku souřadnic. Hu momenty jsou sadou absolutně ortogonálních momentových invariantů a to včetně rotace. Tyto momenty mohou být použity pro rozpoznávání nezávisle na rotaci, měřítku či pozici rozpoznávaného objektu. M. K. Hu tyto momenty definoval pomocí normalizovaných centrálních momentů(viz. [8]). Sada Hu momentů vypadá následovně:

$$
\begin{aligned}
I_1 = &\quad \eta_{20} + \eta_{02} \\
I_2 = &\quad (\eta_{20} - \eta_{02})^2 + (2\eta_{11})^2 \\
I_3 = &\quad (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \\
I_4 = &\quad (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \\
I_5 = &\quad (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta)_{21} + \eta_{03})^2] + \\
&\quad +(3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \\
I_6 = &\quad (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \\
I_7 = &\quad (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + \\
&\quad +(\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]
\end{aligned}
$$

3.4 Další techniky

Crossings

Rysem znaku u této metody je počet průsečíků s objektem s předem zvolenými vektory. Je hojně používána v komerčních systémech.

Strukturální analýza

Tato technika nepatří k triviálním metodám, je založena na popisu geometrické a topologické struktuře znaku. Hledají se základní prvky (úsečky, oblouky . . .) znaku. Některé oblasti této techniky jsou ještě součástí výzkumu.

Kapitola 4

Klasifikace

V této kapitole se budeme zabývat z větší části klasifikátorem K-nearest neighbor, ale ukážeme si i příklady dalších klasifikátorů. Dozvíte se jak klasifikátory fungují, jaké mají výhody či nevýhody, jak daný klasifikátor urychlit či zpřesnit.

Existuje mnoho metod klasifikace. Při nasazení do praktických úloh musíme brát v úvahu dostupné výpočetní prostředky, množství trénovacích dat apod. Podle těchto kritérií se pak musíme rozhodnout, kterou metodu využít, protože každá metoda má odlišné výpočetní nároky i různě kvalitní výstup.

4.1 Trénování klasifikátorů

Trénování klasifikátorů můžeme rozdělit na **trénování s učitelem** (supervised training) a na **trénování bez učitele** (unsupervised training). Ve většině případů je použita první možnost. Při trénování s učitelem optimalizujeme objektivní funkci ze známých dat, o kterých dopředu víme, do které třídy patří. Klasifikátor je potom podle dat, které během učení viděl, schopen předpovědět klasifikační třídu pro neznámá data. Tomuto procesu se říká generalizace dat. Pro trénování klasifikátoru je nemožné předat všechna data, která by se mohla na vstupu objevit, proto vzniká chyba. Naším úkolem je natrénovat klasifikátor tak, aby tato chyba byla co nejnižší. Musíme, ale také počítat s tím, že se snižováním chyby roste také komplexnost klasifikátoru. Proto se často volí kompromis mezi rychlostí a chybovostí řešení.

Jak jsem již nastínil při trénování se používají vzorky dat. Existují dva typy vzorků: *trénovací* a *testovací* data. Na první jmenované sadě se optimalizuje klasifikační funkce a na druhé se ověřuje chyba klasifikace. Trénovací sada se v některých případech dělí na vlastní trénovací sadu a sadu *validační*, která se používá pro detekci přetrénování.

Obecně můžeme říci, že trénovací sada obsahuje obrazy jednotlivých objektů a jejich správné zařazení do klasifikačních tříd. Pro jednoduchost zatím uvažujme, že máme naučit klasifikátor na rozpoznání číslice 1, další číslice pro tento případ nehrají roli. Budeme mít tedy dvě klasifikační třídy - jedničky a cokoliv jiného. Třída jedničky bude obsahovat obrazce jedniček napsané různým pisatelem a v druhé bude vše, co se jako jednička nemá klasifikovat. Abychom tedy dosáhli co nejlepších výsledků, musíme pečlivě vybírat trénovací sadu vzorků. Platí totiž pravidlo, že čím lepší trénovací sadu máme k dispozici, tím lepších výsledků při rozpoznávání dosáhneme.

Průběh trénování

Před samotným trénováním jsou ze vstupních dat vyextrahovány příznaky a je vytvořen příznakový vektor. Samotný průběh trénování probíhá ve dvou krocích:

1. Klasifikace
   V tomto kroku se použije příznakový vektor ke klasifikaci a vypočte se chyba klasifikace.

2. Úprava klasifikátoru
   Zde dochází k úpravě klasifikační funkce podle zjištěné chyby v prvním kroku.

Tyto kroky se opakují dokud není splněna podmínka pro zastavení. Jednou z možností je např. porovnávání chyb mezi jednotlivými kroky klasifikace, kdy se pokračuje tak dlouho dokud pokles chyby nepřesáhne určitou hodnotu. Další podmínkou může být počet iterací, které se mají vykonat. Používaným kritériem je i velikost chyby klasifikátoru. Klasifikátor bude trénován dokud se nedostane pod hodnotu maximální chyby.

Jak tedy vyplývá, trénuje se dokud klesá chyba na celé trénovací sadě. Poté je vybrána metoda, která dává nejlepší výsledky a je nasazena na rozpoznávání validační sady. Pokud i nadále klesá chyba klasifikace, můžeme pokračovat v trénování. V opačném případě hrozí přetrénování (při příliš velkém množství trénovacích dat klesá chyba na trénovacích, ale roste chyba na testovacích datech), proto proces ukončíme.

Učení bez učitele: K této trénovací technice se obrátíme tehdy, když klasifikace dat není předem známá (např. dolování v datech - angl. data mining) nebo klasifikace dat člověkem je příliš drahá (např. při zpracování řeči kdy se vyžaduje velké množství trénovacích dat)[6].

Výpočet chyby klasifikátorů

Chyba klasifikátoru se zjišťuje na trénovací sadě, kde je dopředu znám výsledek. Obecně můžeme říci, že výsledná chyba je počet chybně klasifikovaných dat ku datům celé datové sady. Pro výpočet chyby se zjišťují tyto hodnoty: True Positive (TP, správné přijetí), True Negative (TN, správné odmítnutí), False Positive (FP, špatné přijetí) a False Negative (FN, špatné odmítnutí)viz. [4]. Výsledná chyba se poté vypočítá například ze vztahu 4.1, kde P je počet všech pozitivně zařazených dat a N značí chybně zařazená data.

$$chyba = \frac{FP + FN}{P + N} \tag{4.1}$$

4.2 Klasifikace podle nejbližších sousedů

Klasifikace podle nejbližších sousedů spadá mezi neparametrické metody klasifikace. Tyto metody jsou založeny na podstatně nižších předpokladech než metody parametrické. Nepředpokládáme znalost tvaru pravděpodobnostních charakteristik tříd. Tato metoda je založena na hledání Euklidovské vzdálenosti mezi porovnávanými obrázky. Její nesmírnou výhodou je, že nepotřebuje žádný trénovací čas, jak to bývá u ostatních metod. Bohužel tento přístup má i svá negativa a tím jsou vysoká paměťová náročnost a dlouhá vybavovací doba rozpoznávání. Je to způsobeno tím, že při běhu aplikace si musí metoda pamatovat celou trénovací sadu vzorků. Pokud tedy máme velký počet trénovacích vzorků například 10000 s rozlišením 20×20 pixelů, musí být k dispozici 4 Megabyty operační paměti.

Teorie

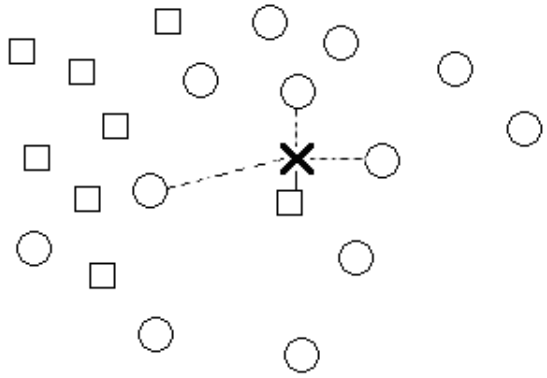
Každý vstupní obrázek je reprezentován sloupcovým vektorem $x \in R$, jehož složky indikují jasové hodnoty pixelů v obrázku. Délka vektoru x je dána rozlišením obrázku, v našem případě se jedná o obrázky v rozlišení 20×20 pixelů. Výsledkem tedy bude vektor s 400 dimenzemi. Výstupem klasifikátoru je klasifikační třída obrázku, kterou budeme značit proměnnou ω. Tato proměnná může nabývat hodnot z konečné množiny $W = \{0, 1, 2, \ldots, 8, 9\}$. Známe tedy naši trénovací množinu $\{(x_1, \omega_1), (x_2, \omega_2) \ldots (x_m, \omega_m)\}$, kde m značí počet sloupcových vektorů (obrázků) x v trénovací množině a jejich správnou klasifikaci ω. Klasifikátor podle nejbližšího souseda zařadí vstupní obrázek x do stejné třídy jako je nejbližší vzor z trénovací množiny podle funkce, která vypočte minimální euklidovskou vzdálenost od hledaného objektu:

$$f(x) \quad = \quad w_i \quad kde\ i = \min_{j \in \{1,\ldots m\}} (x - x_j) \tag{4.2}$$

Nejčastěji se používá metoda nejbližšího souseda (1-NN), ale v některých případech je lepší zvolit metodu s více sousedy.

1-NN

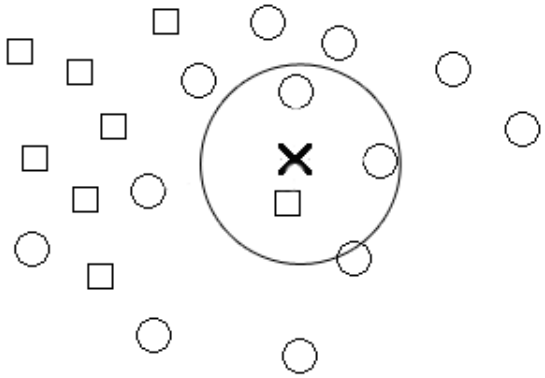
Metoda se chová přesně tak, jak je vysvětlováno výše v 4.2. Tedy nalezne prvek nejblíže hledanému objektu. Proto v některých specifických případech můžeme početnost příznakových vektorů v trénovací množině zredukovat na počet, který odpovídá počtu klasifikačních tříd. To provedeme tak, že jako příznakový vektor každé klasifikační třídy budeme brát průměr příznakových vektorů. Vznikne nám tedy jeden příznakový vektor pro každou třídu tzv. etalon. Klasifikátor poté objekt klasifikuje podle vzdálenosti od těchto etalonů. Při použití tohoto způsobu je klasifikátor rychlý, ale má velkou chybu klasifikace, protože použitá reprezentace tříd (vytváří pouze tzv. předpokládaný statistický model) nepopisuje trénovací data s dostatečnou přesností.



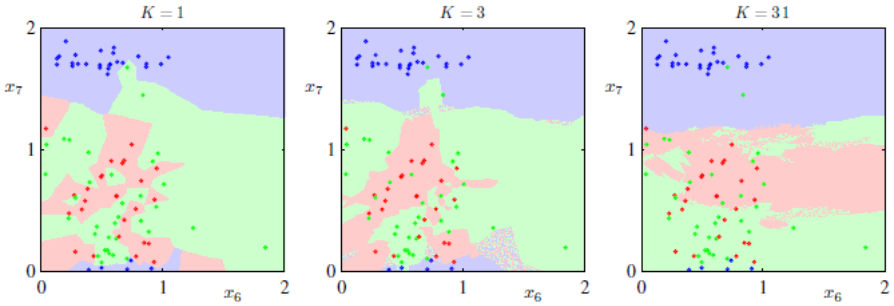
Obrázek 4.1: Klasifikace podle nejbližšího souseda: Který obrazec bude na místě křížku? V našem případě je nejblíže čtverec, proto i neznámý prvek klasifikujeme jako čtverec.

K-NN kde K >1

Použijeme-li klasifikaci pomocí více sousedů, tak rozhodovací algoritmus probíhá následovně. Nalezne se K nejbližších sousedů od neznámého prvku a tento prvek poté klasifikujeme do té třídy, která byla mezi sousedy početnější. Tento postup ilustruje obrázek 4.2. Při použití K-NN kde K >1 je velmi důležitá volba K. Při klasifikaci mezi dvěma třídami se K volí liché, kvůli jednoznačnosti klasifikace. Pokud je tříd více, mohou nastat situace, kdy není možné jednoznačně rozhodnout. Důležitost zvoleného K je demostrováno na obrázku 4.3. Na první pohled by se mohlo zdát jako nejlepší řešení použití K = 1, které separuje data přesně podle generovaných bodů. Podíváme-li se blíže, tak zjistíme, že některé body jsou spíše odchylkou své třídy, které zanášejí do naší klasifikace chyby. Proto lepším řešením je zde volba K = 3. Ve třetím případě naopak vidíme, že volba K = 31 nebude též optimální. Zaměříme-li se na spodní modré body, tak vidíme, že i přesto že body nejsou odchylkou (jsou zde koncentrovány ve větším počtu) tak je klasifikátor ignoruje.



Obrázek 4.2: Klasifikace 3-NN: Který obrazec bude na místě křížku? Kolem neznámého prvku vytvoříme hyperkouli (nezapomeňte, vzdálenost počítáme v Euklidovském prostoru), tak aby do ni padly právě 3 prvky. V našem případě do hyperkoule padly dva kruhy a jeden čtverec, proto prvek klasifikujeme jako kruh.

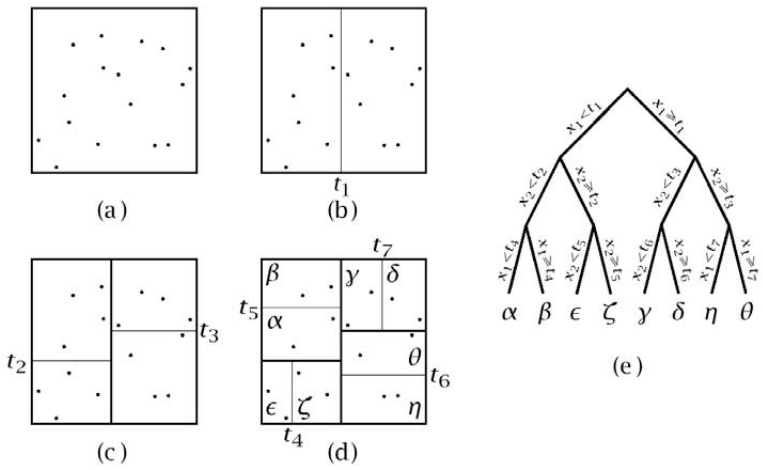


Obrázek 4.3: Ukázka klasifikace s různými K (zdroj [1])

4.2.1 Urychlení klasifikátoru

K-D stromy

K-D stromy jsou modifikací algoritmu K-NN, která využívá rozdělení trénovací množiny na oblasti. Data trénovací množiny jsou setříděny podle jedné souřadnice. Nejdříve se nalezne medián, podle kterého se rozdělí trénovací data na dvě množiny. Tyto množiny opět rozdělíme podle mediánu, tentokrát ale podle druhé souřadnice. Tento postup provádíme do té doby, dokud nedostaneme právě jeden bod. Při půlení podmnožin cyklicky střídáme souřadnice, podle kterých hledáme medián. Klasifikace nového prvku probíhá tak, že nalezneme buňku, do které patří nový objekt. Poté změříme vzdálenost nového prvku od prvku ležícího uvnitř buňky a vzdálenost od jejich hranic. Pokud je klasifikovaný prvek blíže k bodu, nalezli jsme nejbližšího souseda. Pokud ne, musíme prohledat buňky i za hranicemi.



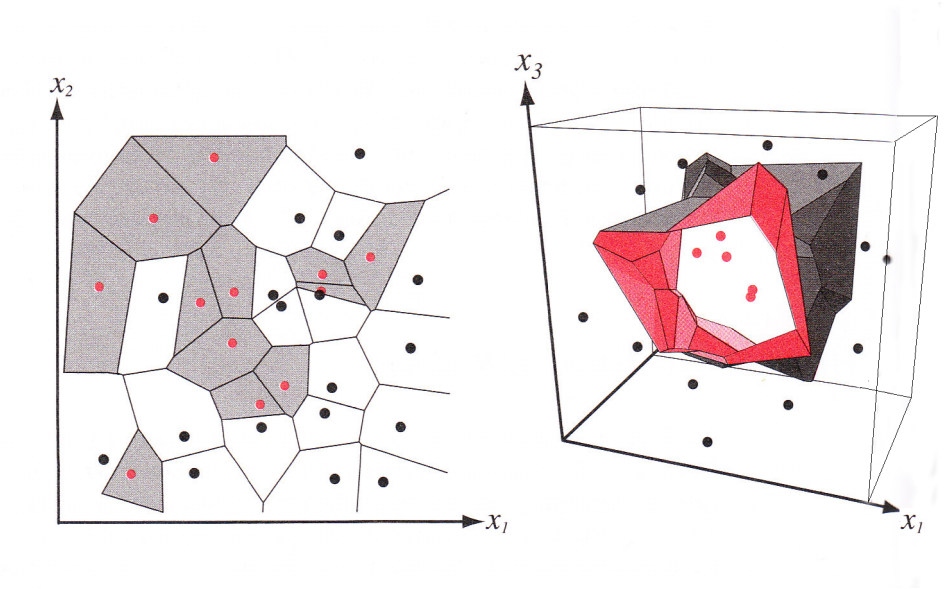
Obrázek 4.4: K-D strom(zdroj [5])

Vyloučení vzorků pomocí Voronoiova diagramu

Tato metoda je založena na odstranění přebytečných prvků, které neovlivňují klasifikaci. Nejprve nalezneme hrany Voronoiova diagramu. Mezi dvěma prvky množiny existuje hrana Voronoiova digramu, pokud existuje bod, který má od obou těchto prvků stejnou vzdálenost a od ostatních bodů leží dál. Mezi takovými dvěma prvky vedeme stěnu tak, že je kolmá na spojnici těchto prvků a tato spojnice protíná stěnu v jejím středu. Spojením takovýchto stěn získáváme Voronoiův diagram, někdy pro svůj tvar nazýván Voronoiova mozaika.

Nalezli jsme graf a nyní můžeme vyloučit ty prvky, jejichž Voronoiovi buňky sousedí pouze s buňky prvku ze stejné třídy. Tímto krokem se nám nezmění rozhodovací nadplocha, ale urychlíme klasifikaci.

Pro dimenze výrazně větší jak 1 je tato metoda nepoužitelná, protože skoro všechny prvky leží kolem rozhodovací nadplochy a při složitosti tvorby Voroniova diagramu $O(n \log n)$ bychom platili velkou cenu za vyloučení malého počtu prvků.



Obrázek 4.5: Ukázka Voroniova diagramu u dvou a tří dimenzionálních prostorů(zdroj [3])

4.2.2 Chyba klasifikátoru a její snížení

Platí, že chyba metody nejbližších sousedů klesá s jejich počtem.

$$P^{NN}(\varepsilon) \leq P^{k-NN}(\varepsilon) \leq P^{3-NN}(\varepsilon) \leq P^{1-NN}(\varepsilon) \leq 2P(\varepsilon) \tag{4.3}$$

$$P^{k-NN}(\varepsilon) < P(\varepsilon) + \frac{P^{1-NN}}{\sqrt{k\pi}} \tag{4.4}$$

Editace trénovací množiny [11]

Cílem této metody je vyloučit z trénovací množiny ty prvky, které vnášejí chybu do klasifikace. Tyto prvky jsou většinou odchylky od typických zástupců nebo prvky vzniklé chybou měření. Algoritmus pro editaci množiny vypadá následovně:

1. náhodně rozdělíme trénovací množinu S_n na dvě poloviny S_{n1} a S_{n2}.
2. klasifikujeme vzorky z S_{n1} metodou nejbližších sousedů. S_{n2} používáme jako trénovací množinu.
3. z S_{n1} vyloučíme vzorky, které nebyly správně klasifikovány v předchozím kroku.
4. vzniklou množinu S_{n1} pak použijeme ke klasifikaci metodou nejbližších sousedů

Klady a zápory

Klady:

- jednoduchá implementace, nepotřebujeme znát tvar pravděpodobnostních charakteristik

- chyba tohoto klasifikátoru je srovnatelná s chybou složitějších metod, proto se často používá jako referenční metoda.

Zápory:

- vysoká paměťová náročnost
- pomalé rozhodování
- klasifikace závisí na velikosti porovnávaných objektů

4.3 PCA

PCA je zkratkou anglického názvu Principal Component Analysis, v překladu tedy Analýza hlavních komponent. Je jednou z nejstarších a nejvíce používaných metod vícerozměrné analýzy. Poprvé byla zavedena Pearsonem již v roce 1901 jako popisná statistická metoda, sloužící především k redukci vícerozměrných dat. H. Hotelling zobecnil v roce 1933 postup aplikací komponentní analýzy na náhodné vektory a navrhl použití analýzy hlavních komponent pro rozbor kovarianční struktury proměnných [10]. Proto se v praxi můžete setkat i s názvy jako Hotellingova transformace nebo Karhunen-Loeveho transformace. Cílem analýzy hlavních komponent je především zjednodušení popisu skupiny vzájemně lineárně závislých (korelovaných) znaků. Techniku lze popsat jako metodu lineární transformace původních znaků na nové, lineárně nezávislé proměnné, nazvané hlavní komponenty. Tyto komponenty shrnují informaci o původních za cenu minimální ztráty informace. Hlavní komponenty jsou setříděny sestupně podle míry variability, neboli rozptylu. Může být chápána jako transformace z původního do nového souřadnicového systému, jehož osy jsou tvořeny hlavními komponentami. Osy procházejí směry maximálního rozptylu, protože podmínka nezávislosti komponent vede ke kolmosti os [16].

PCA je technikou, která je široce používána v aplikacích, které potřebují např. redukovat dimenze, kompresi dat s co nejmeší ztrátovostí, extrakci příznaků apod. Je využívaná, protože je jednoduchou, neparametrickou metodou pro extrakci důležitých informací z neuspořádaných dat. Je citlivá na změnu měřítka, proto se musí nejdříve provést normalizace původních proměnných.

V této práci byla PCA použita pro redukci dimenzí trénovacích dat metody k-nearest neighbor. Vstupem je diskrétní náhodná proměnná, což je barva na dané pozici objektu, která je napříč objektu různá.

Postup pro redukci dimenzí[13]:

1. Máme I obrázků, každý o rozměrech $N \times N$ pixelů. Každý obrázek uložíme do jednoho vektoru o délce N^2, kde hodnoty vektoru jsou intenzity jednotlivých pixelů. Z těchto vektorů, poté vytvoříme matici O:

$$\mathbf{obrazMatice} = \begin{pmatrix} obrazVektor1 \\ obrazVektor2 \\ \vdots \\ obrazVektorI \end{pmatrix}$$

2. Nyní vypočteme průměrnou hodnotu pro každou dimenzi. Průměrnou hodnotu dimenze X budeme značit $\overline{X}$. Od každé hodnoty odečteme průměrnou hodnotu a uložíme do matice P.

3. Vypočteme kovariační matici. Abychom ji mohli vypočítat, musíme nejprve zjistit, jak spočítat kovarianci:

$$cov(X,Y) \quad = \quad \frac{\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})}{n-1} \tag{4.5}$$

   Nyní se tedy podíváme na samotnou kovariační matici C:

$$\mathbf{C} = \begin{pmatrix} cov(X,X) & cov(X,Y) & cov(X,Z) & \dots & cov(X,N) \\ cov(Y,X) & cov(Y,Y) & cov(Y,Z) & \dots & cov(Y,N) \\ cov(Z,X) & cov(Z,Y) & cov(Z,Z) & \dots & cov(Z,N) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ cov(N,X) & cov(N,Y) & cov(N,Z) & \dots & cov(N,N) \end{pmatrix}$$

   Kovarianční matice je čtvercová o rozměrech NxN a je symetrická podle hlavní diagonály, protože platí:

$$cov(X,Y) \quad = \quad cov(Y,X) \tag{4.6}$$

4. Z kovarianční matice vypočteme tzv. vlastní čísla (angl. eigenvalues) a vlastní vektory (angl. eigenvectros.), tyto hodnoty jsou navzájem propojené. Vlastní čísla uvádějí míru variability jednotlivých vlastních vektorů.

5. V této části se rozhodujeme, které dimenze jsou méně významné. Ty poznáme tak, že mají malou hodnotu vlastního čísla. Platí totiž pravidlo, že má-li nějaký původní znak malý či dokonce nulový rozptyl, není schopen přispívat k rozlišení mezi objekty.

   Vytvoříme si matici FV jehož hodnoty jsou hodnoty vlastních vektorů sestupně seřazené podle hodnot vlastních čísel. Délka vektoru je rovna počtu dimenzí, které chceme zachovat.

6. Získáme finální data:

$$FinalData \quad = \quad radekMaticeFV \times radekMaticeP \tag{4.7}$$

Jak jsme si z uvedeného postupu mohli všimnout, model PCA odpovídá aproximaci zdrojové matice dat O, který použijeme namísto původní zdrojové matice. Tato aproximace má řadu výhod v interpretaci dat. PCA má za úkol transformovat data do nového systému os a nalézt a vypustit šum, čímž snížíme rozměrnost úlohy. Problémem zůstává, kolik hlavních komponent je nutné použít. Existuje horní mez počtu hlavních komponent, které mohou být odvozeny ze zdrojové matice dat O. Největší počet hlavních komponent se buď rovná číslu $m-1$, nebo n v závislosti na tom, které z těchto dvou čísel je menší. Je-li matice O složena například z $n = 800$ obrázků a $m = 400$ pixelů v obrázku, bude maximální počet hlavních komponent 399. Počet efektivních hlavních komponent se rovná hodnosti zdrojové matice O [10].

4.4 Neuronové sítě

Nejvíce používaná rozpoznávací metoda. Cílem této podkapitoly je obecný průřez problematiky modelování těchto sítí. Samotná problematika je natolik složitá, že se ji lidé věnují v samostatných publikacích.

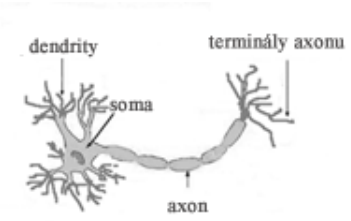
Původním cílem výzkumu neuronových sítí byla snaha pochopit a modelovat činnost lidského mozku, snaha pochopit jakým způsobem myslíme. Jako zdroj informací zde slouží neurofyziologie, která nám umožnila vytvořit zjednodušené matematické modely. Navržené neuronové sítě jsou poté rozvíjeny bez ohledu na to, zda modelují lidský mozek. Přesto bývá užitečné se k této analogii vracet, ať už pro nové inspirace nebo pro popis vlastního modelu [9].

Lidský mozek

Mozková kůra je tvořena asi 13 miliardami zakladními stavebními, výpočetními prvky tzv. neurony. Každý z neuronů může být spojen s dalšími 5000 neurony. Neuron (obr. 4.6) je specializovaná buňka určená k přenosu a uchovávání informací.

Vstupem neuronu jsou dendrity a výstupem je axon. Informace se poté šíří tzv. synapsí, které můžeme považovat za rozhraní mezi dendrity a terminály axonu. Synapse můžeme rozdělit na tzv. excitační, které šíří rozruch, a inhibiční, které naopak rozruch tlumí.

Aktivace neuronu nastane tehdy, pokud překročí hodnota vstupních budících signálů signály tlumící o určitou hodnotu.

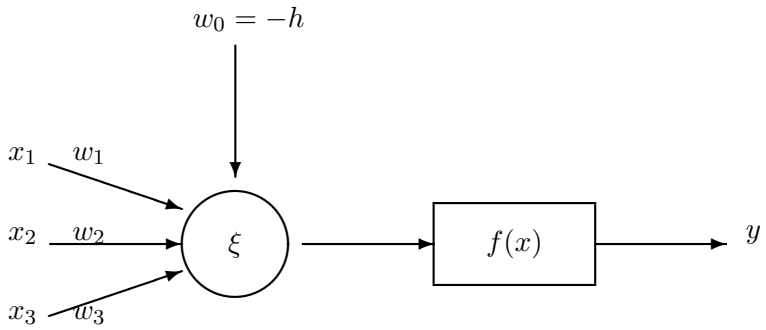


Obrázek 4.6: Neuron

Umělé neuronové sítě

Jak jsem již psal výše, umělé neuronové sítě čerpají inspirace z biologických, proto i zde je základní jednotkou neuron. Tento neuron může mít n vstupů (modelují dendrity), ale pouze jeden výstup (modeluje axon). Jak si můžete všimnout na obrázku 4.7, tak umělý neuron získáme přeformulováním zjednodušené funkce neurofyziologického neuronu (obr. 4.6) do matematické řeči. Zde je schematicky zobrazen neuron se třemi vstupy x_1, x_2 a x_3.
Vstupy jsou ohodnoceny odpovídajícími obecně reálnými synaptickými váhami w_1, w_2 a w_3, které určují jejich propustnost. Ve shodě s neurologickou motivací mohou být synaptické váhy záporné, čímž se vyjadřuje jejich inhibiční charakter. Zvážená suma vstupních hodnot představuje vnitřní potenciál neuronu ξ. Hodnota vnitřního potenciálu ξ po dosažení tzv. prahové hodnoty h indikuje výstup neuronu y. [9]

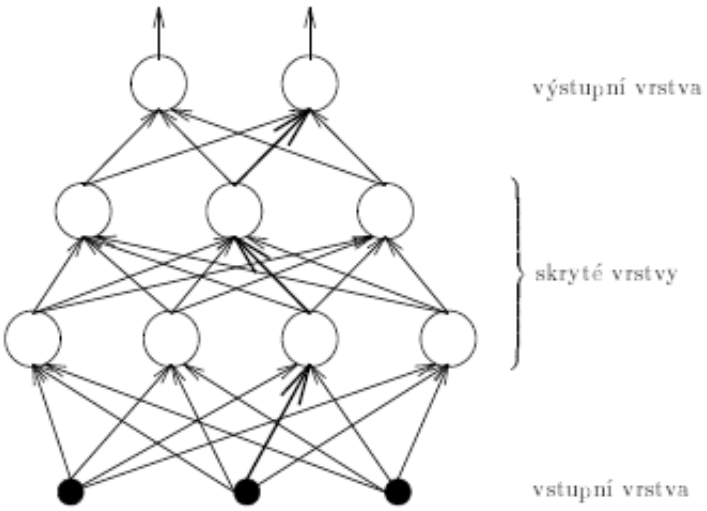
Neuronová síť (obr. 4.8) se skládá z těchto neuronů, které jsou navzájem propojeny tak, že výstupem neuronu je vstup několika dalších neuronů. Počet neuronů a jejich vzájemných propojení určuje tzv. topologie sítě [9].



Obrázek 4.7: Model umělého neuronu se třemi vstupy.

Z hlediska využití neuronů rozlišujeme v síti:

- vstupní neurony: jsou to tzv. pasivni prvky sítě, slouží ke vstupu signálu a k následnému rozdělení další vrstvě neuronů.
- pracovní neurony: někdy nazývané jako skryté, mezilehlé neurony. Tyto neurony transformuji vstupy a výsledek předávají další vrstvě.
- výstupní neurony: výstupem těchto neuronů je odezva na signály vstupní.



Obrázek 4.8: Propojení neuronů v síti (zdroj [9])

Modely neuronových sítí

Historicky prvním úspěšným modelem neuronové sítě byla síť perceptronů. Skládá se pouze z jedné vrstvy výstupních neuronů. Nejznámějším a nejpoužívanějším modelem neuronové sítě je vícevrstvá neuronová síť s učícím algoritmem zpětného šíření chyby (backpropagation).

Kapitola 5

Implementace a experimenty

Ukázkové programy jsou implementovány v jazyce C++ a byly navrženy tak, aby jejich zdrojový kód byl přenositelný. Vyvíjeny a odzkoušeny byly na platformě Windows XP Professional.

Při vývoji byla použita externí knihovna OpenCV, která byla vytvořena firmou Intel. Jde o volně dostupnou knihovnu, která je šířena pod BSD licencí. Lze s ní snadno konat veškerou práci s obrázky i s některými rozpoznávacími metodami. Všechny programy byly vytvořeny jako konzolové aplikace.

Rozpoznávací program

Program se ovládá pomocí těchto parametrů:

Knn.exe [-r -t] -file obrazek [-k -v] [K V] [-pca]

- *-r* indikuje rozpoznávací funkci
- *-t* indikuje trénovací fuknci
- *-file* obrázek k rozpoznávání či trénování
- *-k* parametr při rozpoznávání, počet nejbližších sousedů
- *-v* parametr při trénování, jde o výslednou třídu
- *-pca* použít rozpoznávání s metodou PCA

Více informací naleznete v souboru README, který je uložen u každého programu.

5.1 Trénovací data

Abych mohl ověřit teorii v praxi, bylo nutné vytvořit vhodnou databázi ručně psaných číslic. K tomuto účelu bylo vytvořeno několik variant formulářů. Pro ilustraci je jeden z nich uveden v příloze. Tyto formuláře byli posléze vyplněny studenty na základní škole v Mohelně a na gymnáziu Jana Blahoslava v Ivančicích. Díky tomu, že připadl jeden formulář na osobu, dostal jsem dostatek reálných vzorků písem. Formuláře jsem naskenoval a vytvořil program pro dolování jednotlivých číslic a znaků. Můj scanner ukládá data v rozlišení 2480×3507 pixelů. Byl proto vytvořen program, který na vstupu očekává formulář v tomto rozlišení.

Program zjistí verzi formuláře a podle této verze, vydoluje znaky a uloží pod příslušným názvem. Je založen na nalezení čtvercových obrysů (kontur), které si seřadí podle řádků od levého horního rohu po pravý spodní roh formuláře. Výstupem tohoto formuláře jsou soubory ve formátu *.jpg v rozlišení 82×82 pixelů.

Po přečtení literatury a po dohodě s vedoucím jsem dospěl k názoru, že vzorek v rozlišení 82×82 pixelů je až přespříliš, proto jsem se rozhodl, že ke klasifikaci použiji obrázky v obvyklém rozlišení, tedy 20×20 pixelů. Jelikož si myslím, že databázi ve větším rozlišení by mohl někdo později ve svém výzkumu využít, rozhodl jsem se neměnit program na dolování dat z formuláře, ale vytvořil jsem další konzolovou aplikaci. Tato aplikace čeká tedy na vstupu obrázek znaku 82×82 pixelů. Protože klasifikační metoda pomocí nejbližších sousedů je závislá na měřítku a není invariantní vůči posunutí, slouží tento program i k předzpracování nebo-li generalizaci klasifikačních vzorků. Na základě zjištěných rozměrů ručně psaného znaku, je oříznuto přebytečné bílé okolí znaku. Poté je znak vycentrován a zmenšen na požadované rozlišení. Při zmenšování je použita interpolace počítající s oblastí pixelu.

5.2 Klasifikační program

Celý program lze rozdělit na několik podprogramů, které řeší jednotlivé podproblémy:

- Načtení obrazu a prahování
- Extrakce příznaků
- Samotná klasifikace

5.2.1 Načtení obrazu a prahování

Obrázek je načten jako 8 bit mapa, tedy jako obrázek v odstínech šedi. Pro naše účely, ale potřebujeme obrázek v binarizovaném stavu, proto provedeme prahování (viz. teorie 2.3). Experimentální metodou byla nalezena nejvhodnější hodnota prahu, která je v programu neměnná a je nastavena na 245.

5.2.2 Extrakce příznaků a výsledky klasifiace

První návrh

Pro popis znaků jsem nejprve zvolil Hu momenty. Těchto sedm momentů má zaručit invariantnost vůči měřítku a rotaci. Způsob výpočtu jsme si uvedli v sekci 3.3. Pomocí těchto momentů byl vytvořen příznakový vektor, který byl použit ke klasifikaci pomocí algoritmu nejbližších sousedů. Dataset jsem si rozdělil přibližně na polovinu. První částí byla trénovací data ve které bylo 250 vzorků od každé číslice. Druhou část jsem zvolil jako testovací a ta obsahovala kolem 300 vzorků od každé číslice. Proběhlo několik testů, kdy jsem nejdříve zvolil $K = 1$ a posléze je zvyšoval s jednotkovým krokem až do hodnoty 31, pokaždé s žalostným výsledkem. Úspěšnost klasifikátoru se pohybovala v rozmezí 5 - 10 %.

Po prozkoumání extrahovaných rysů vyšlo najevo, že hu momenty jsou silně závislé na tvaru znaku a pro ručně psané znaky při klasifikaci metodou nejbižších sousedů je tento popis nedostačující. Naopak při nasazení do OCR systémů, kde se porovnávají tištěné znaky, si myslím, že by bylo vhodné tento popis nasadit, neboť na popis znaků stačí pouze vektor o sedmi dimenzích.

Druhý návrh

Po předchozím zklamání jsem zvolil pro popis znaků jas jednotlivých pixelů, jak je uvedeno na str.12. Výsledky jsou uvedeny v následujících tabulkách.

Číslice		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Počet		202	216	266	255	168	212	182	224	188	219
Ch[%]	**K ↓**										
	1	4,95	4,63	4,51	11,76	8,93	15,57	7,14	15,63	14,36	17,35
	5	3,95	3,24	6,02	11,76	8,93	16,51	6,59	12,50	13,50	17,81

Tabulka 5.1: Tabulka ukazuje chybovost u jednotlivých znaků v závislosti na zvoleném K. Trénovací sada obsahovala 250 vzorů od každého znaku. Počet klasifikovaných znaků je v kolonce *Počet*.

Hned při prvních pokusech bylo zřejmé, že pro větší vypovídací hodnotu experimentů je třeba zvětšit datovou sadu. Při dalších experimentech bylo v trénovací sadě 300 vzorků od každé číslice.

Číslice		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Počet		529	534	608	594	431	500	459	537	453	533
Ch[%]	**K ↓**										
	1	5,10	6,00	6,25	13,64	10,44	17,4	6,10	16,39	12,58	20,64
	5	3,21	4,68	5,26	10,61	9,51	16,20	5,88	16,76	15,67	20,26
	27	5,29	4,68	8,06	11,95	10,21	22,00	7,19	21,42	16,34	21,76

Tabulka 5.2: Tabulka ukazuje chybovost u jednotlivých znaků v závislosti na zvoleném K. Počet klasifikovaných znaků je v kolonce *Počet*.

Pro zajímavost zde ještě uvedu tabulku, ve které naleznete celkovou chybovost na zvoleném K.

K	**Chyba [%]**
1	11,45
3	11,51
5	10,72
7	11,34
11	11,61
27	12,84

Tabulka 5.3: Tabulka ukazuje chybovost klasifikátoru v závislosti na zvoleném K.

Nejhorších výsledků bylo dosaženo při volbě K = 27 naopak nejlepších výsledků při volbě **K = 5**. Výsledek ukazuje následující konfuzní matice. Z této matice mimo jiné vyčteme, za co byl znak nejvíce zaměňován. Podíváme-li se na poslední řádek, vidíme, že klasifikátor nejčastěji zaměnil číslici 9 za čislici 3 a číslici 1.

Číslice	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	512	2	2	0	1	0	5	0	7	0
1	0	509	10	1	4	1	2	1	4	2
2	6	2	576	2	4	2	2	3	9	2
3	5	8	5	531	3	9	7	2	7	17
4	0	23	1	0	390	0	3	3	0	11
5	5	5	2	18	7	419	29	1	7	7
6	6	2	2	0	3	9	432	0	5	0
7	0	28	6	1	29	1	2	447	10	13
8	6	7	5	14	6	5	11	4	382	13
9	5	30	4	34	13	1	2	8	11	425

Tabulka 5.4: Konfuzní matice pro K = 5. Ukazuje, kolikrát byl znak klasifikován za jiný. Na y-ové ose je předkládaný znak a na x-ové ose je jeho klasifikace.

Třetí návrh

Pro zmenšení rozhodovacího času klasifikace jsem na příznaky použil metodu PCA viz. sekce 4.3, kdy jsem zredukoval množství dimenzí. Předmětem experimentu bylo i zjištění nejmenšího počtu dimenzí, kdy se ještě chyba klasifikace nezvyšuje. Použití PCA je výhodné tehdy, pokud trénovací sada čítá velké množství prvků. Na použité sadě se rychlost PCA nedokázala projevit, ale při použití větší trénovací sady nebo větších obrázků má své opodstatnění. Uvádím zde výsledky při redukci dimenzí na 200.

Číslice	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Počet	529	534	608	594	431	500	459	537	453	533
Ch[%]	3,02	4,49	5,59	9,60	9,28	16,80	5,88	15,64	16,11	21,39

Tabulka 5.5: Tabulka ukazuje chybovost při použití PCA a K = 5. Celková chyba klasifikátoru byla **10,68 %** s použitím polovičního počtu příznaků.

Čtvrtý a poslední návrh

Posledním experimentem bylo aplikování metody skeletonizace a její projevení na chybě klasifikátoru, ať již s metodou PCA nebo bez. Podle dostupných informací jsem předpokládal zpřesnění klasifikace. Avšak ze zjištěných výsledků vyplynulo, že skeletonizace negativně ovlivnila klasifikaci, proto jsem se rozhodl ji v programu zakomentovat. Na vině je nejspíše použitá euklidovská metrika a pro tuto metodu nevhodně zvolené příznaky. U jiných metod klasifikace či odlišně zvolených příznaků může být volba skeletonizace prospěšná.

Shrnutí

Ze zvolených řešení dosahoval nejlepších výsledku druhý a třetí návrh. Metodou PCA bylo možné zredukovat počet dimenzí na polovinu, aniž by se do klasifikace vnášela chyba. Editací množiny bylo dosaženo lepších výsledků, ty se ale projevovali jen v desetinách procent. Při použití skeletonizace jsem očekával, že se klasifikace ještě zpřesní, avšak bylo tomu právě naopak.

Kapitola 6

Závěr

Cílem této práce bylo seznámit se s používanými technikami při rozpoznávání ručně psaných číslic a na jejich základě vytvořit program pro klasifikaci. Tato práce byla zaměřena na klasifikaci algoritmem k-nejbližších sousedů, hledání vhodných popisných příznaků znaku pro tuto metodu a průzkum modifikací daného algoritmu. Při experimentech byla použita metoda skeletonizace a PCA.

Byly navrženy formuláře a program pro jejich zpracování za účelem tvorby své vlastní databáze obsahující izolované číslice a znaky abecedy. Poté byl navržen a zrealizován program na rozpoznávání číslic metodou k-nejbližších sousedů. Při realizaci programu bylo největším problémem zvolení vhodných příznaků pro rozpoznávání, neboť mnoho publikací se této problematice nevěnuje. Velké množství času bylo také věnováno na zkoumání získaných výsledků a hledání optimálního řešení i výběru ukázkových řešení uvedené v kapitole 5.

Chyba klasifikace je kolem 10%. Nedosáhl jsem sice takových výsledků, jaké jsou uvedeny u databáze MNIST[1], protože je použita daleko menší trénovací sada. Pro tuto použitou datovou sadu se však výsledky jeví jako dobré.

Výsledky experimentů dávají najevo, že pro rozpoznávání ručně psaných číslic není metoda k-nearest neighbor optimální, proto bych se v budoucích fázích projektu zaměřil na silnější metody v rozpoznávání obrazu. Zajímavou variantou se jeví neuronové sítě, kde lze vyzkoušet různé topologie neuronových sítí a porovnat dosažené výsledky klasifikace.

[1] http://yann.lecun.com/exdb/mnist/

Literatura

[1] Bishop, C. M.: *Pattern recognition and machine learning*. Springer, 2006, ISBN 0-387-31073-8.

[2] Bräunl, T.: *Parallel image processing*. Springer, 2001, ISBN 3-540-67400-4.

[3] Duda, R. O.; Hart, P. E.; Stork, D. G.: *Pattern Classification*. A Wiley-Interscience Publication, 2001, ISBN 0-471-05669-3.

[4] Fawcett, T.: An introduction to ROC analysis. [online] [cit. 3.5.2009].
URL http://www.csee.usf.edu/~candamo/site/papers/ROCintro.pdf

[5] Gutierrez-Osuna, R.: Introduction to Pattern Analysis. [online] 2002 [cit. 3.5.2009].
URL http://www.lsi.upc.edu/~belanche/docencia/mineria/Apunts/MD3.pdf

[6] Hlaváč, V.: Učení bez učitele. [online] [cit. 3.5.2009].
URL http://cmp.felk.cvut.cz/~hlavac/Public/TeachingLectures/UceniBezUcitele.pdf

[7] Hlaváč, V.; Sedláček, M.: Zpracování signálu a obrazu. 1999, 77 s.

[8] Hu, M. K.: Visual Pattern Recognition by Moment Invariants. *Information Theory,*, ročník 8, 1962: s. 179–187, ISSN 0018-9448.

[9] Šíma, J.; Neruda, R.: *Teoretické otázky neuronových sítí*. Matfyzpress, 1996, ISBN 80-85863-18-9.

[10] Meloun, M.; Militný, J.; Hill, M.: *Počítačová analýza vícerozměrných dat v příkladech*. Academia, 2005, ISBN 80-200-1335-0.

[11] Pivoňková, A.; Šmerák, P.: Zápis z přednášky - Klasifikace podle KNN. [online] 2002 [cit. 3.5.2009].
URL http://cmp.felk.cvut.cz/cmp/courses/recognition/zapis_prednasky/zapis_03/nearest--neigh03.doc

[12] Sabourin, R.; Bortolozzi, F.; Aires, S.; aj.: Perceptual Zoning for Handwritten Character Recognition.
URL http://www.livia.etsmtl.ca/publications/2005/perceptual.pdf

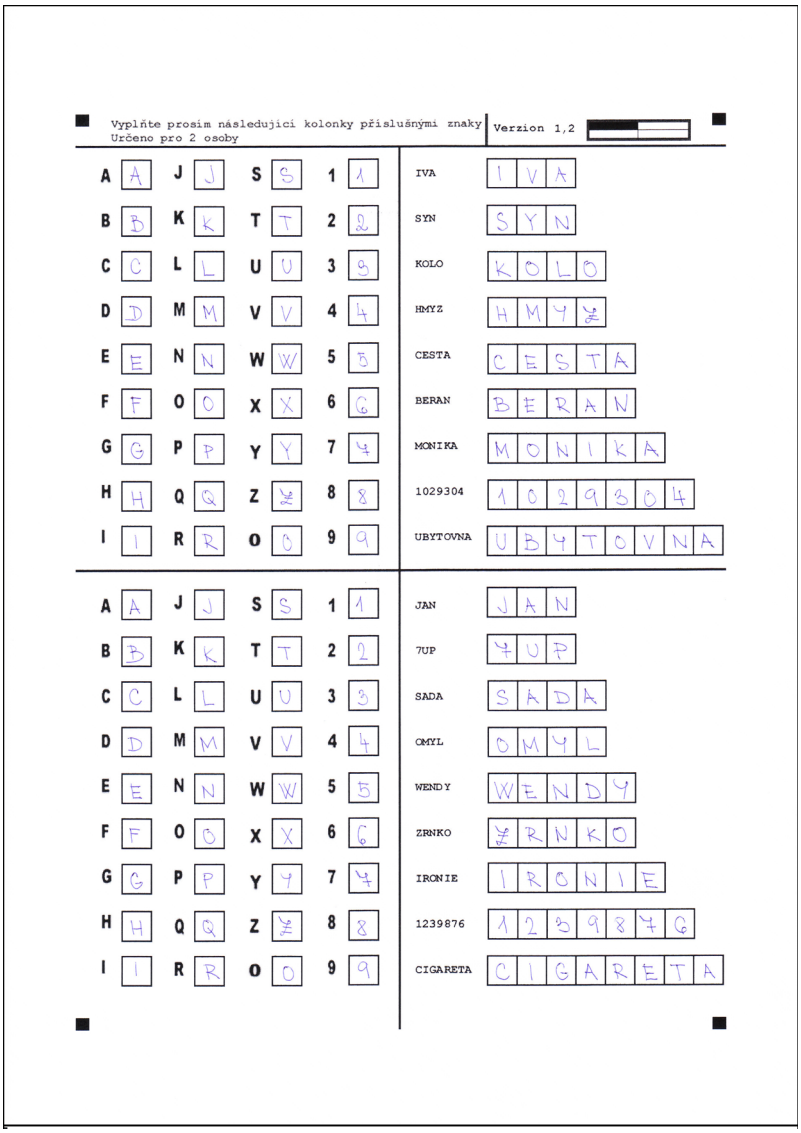
[13] Smith, L. I.: A tutorial on Principal Components Analysis. [online] 2002 [cit. 3.5.2009].
URL http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf

[14] Verma, B.; Blumenstein, M.: *Pattern Recognition technologies and applications*. IGI Global, 2008, ISBN 978-1-59904-807-9.

[15] Wikipedie: Neuronová síť. [online], [rev. 7.2.2009], [cit. 2.5.2009].
URL http://cs.wikipedia.org/wiki/Neuronová_síť

[16] WWW stránky: Analýza hlavních komponent v programu SAS. [online] [cit. 3.5.2009].
URL http://info.lu2.name/soubory/Principal_components_analysis_481.pdf

Příloha



Obrázek 1: Vzorový formulář, zmenšeno na 50%

Obsah CD

Na přiloženém CD je uložena elektronická podoba tohoto dokumentu. Kromě zdrojových souborů jsou zde přiloženy i naskenované vyplněné formuláře, vyřezané znaky z těchto formulářu roztříděné do odpovídajících složek. Také obsahuje vycentrované znaky v rozlišení 20×20 pixelů. Součástí jsou i dávkové soubory demonstrující funkčnost programu.