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 RUKOU PSANÉHO TEXTU

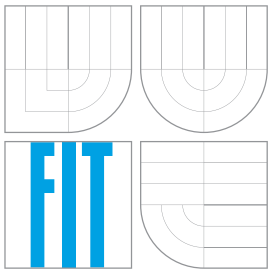
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

DAVID ZOUHAR

BRNO 2010

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 RUKOU PSANÉHO TEXTU

HANDWRITTING RECOGNITION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — DAVID ZOUHAR
AUTHOR

VEDOUCÍ PRÁCE — Ing. JOZEF MLÍCH
SUPERVISOR

BRNO 2010

Abstrakt

Tato bakalářská práce se zabývá rozpoznáváním znaků psaných rukou v reálném čase. Popisuje způsoby získávání informací pro rozpoznávání textu, metody používané při klasifikaci a aplikaci vytvořenou pro získání textu z nakreslených znaků. Dále se také zabývá vyhodnocením vytvořené aplikace. Zaměřuje se na experimenty, které byly prováděny pro zvýšení úspěšnosti rozpoznávání. Díky provedeným experimentům se podařilo dosáhnout úspěšnosti okolo 85%.

Abstract

This bachelor thesis deals with the handwritten character recognition in real time. It describes the ways how to obtain information for the text recognition, methods used in classification and it descibes application made for getting text from drawn characters. It is also engaged in evaluation the created application. It deals with the experiments that were conducted to improve success of recognition. Thanks to the experiments, the success that was achieved was approximately 85%.

Klíčová slova

OCR, on-line rozpoznávání znaků, skryté Markovovy modely.

Keywords

OCR, ol-line character recognition, hidden Markov models.

Citace

David Zouhar: Rozpoznávání rukou psaného textu, bakalářská práce, Brno, FIT VUT v Brně, 2010

Rozpoznávání rukou psaného textu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jozefa Mlícha

.....
David Zouhar
18. května 2010

Poděkování

Tímto bych rád poděkoval vedoucímu mé práce, panu Ing. Jozefu Mlíchovi za vedení mé práce, cenné rady a podnětné návrhy.

Obsah

Kapitola 1

Úvod

Optické rozpoznávání znaků patří mezi disciplíny počítačového vidění. Při rozpoznávání textu počítačem se snažíme převést obecně binární soubor s obrazovými daty na textový soubor. Rozpoznávání textu je možné rozdělit na dvě velké skupiny: Rozpoznávání skenovaného textu (off-line) a rozpoznávání textu psaného rukou (on-line).

Off-line rozpoznáváním se zabývá tzv. OCR (Optical character recognition) software. Existuje nepřeberné množství komerčních i volně šiřitelných programů. Problémy off-line rozpoznávání strojově psaného textu jsou již dnes převážně považovány za vyřešené. Metody pro rozpoznávání strojového textu dokáží rozpoznat text až s 99%ní přesností [10]. Ale i při rozpoznávání strojem psaných textů dnešní rozpoznávací programy naráží na mnoho překážek. Musí se například vypořádat s různými typy písem (fonty). Také je velký problém v rozpoznávání kurzívy. Úspěšnost rozpoznávání takového textu je dokonce nižší než při rozpoznávání rukou psaného textu. Text předávaný rozpoznávači také nemusí být vodorovný, může se zde nacházet mnoho rozmazaných znaků (šumu) a mnoho jiných problémů, které se při rozpoznávání vyskytnou.

On-line rozpoznávání (rukou psaných textů) je více problematické. Lidé nemají, tak jako stroje, *normované písmo*. Každý člověk má svůj styl psaní. Ovšem tento rozdíl je možné částečně potlačit tím, že uživatel rozpoznávač *naučí* svůj styl psaní. K tomu je ovšem třeba, aby uživatel, než začne program používat, napsal rozpoznávači všechna písmena (číslice). Rozpoznávač se na této sadě písmen natrénuje a pak je pro něj jednodušší rozpoznávat písmo jeho uživatele.

Bakalářskou práci lze rozdělit na dva ucelené bloky. Na teoretickou a praktickou část. Ve 2. kapitole se věnuje teorii rozpoznávání a dvěma metodám, které se při rozpoznávání nejčastěji využívají. Kapitola se zabývá neuronovými sítěmi a skrytými Markovovými modely. Projekt je založen na skrytých Markovovýh modelech, proto se na tuto metodu zaměřuje více. Ve 3. kapitole se pak zaměřuje na popis vyvýjeného programu pro rozpoznávání rukou psaných znaků. Kapitola 4 pak shrnuje výsledky programu a vyhodnocuje je.

Kapitola 2

Rozpoznávání znaků

Při rozpoznávání znaků se snažíme docílit toho, aby rozpoznaný znak odpovídal hodnotě, kterou ve skutečnosti představuje. Proces rozpoznávání je velice složitý. I přes mnoho sofistikovaných postupů a modelů se patrně nikdy nedosáhne stoprocentní úspěšnosti. Při rozpoznávání je velice důležité si data vhodně připravit nebo rozdělit do vhodných částí, se kterými se posléze bude efektivněji pracovat. Takto připravená data se pak již mohou poslat rozpoznávači, který se je bude snažit analyzovat a rozpoznat. Rozpoznávačem se myslí model, který je schopen rozeznat odlišnosti v datech a pak je klasifikovat do námi zvolených skupin.

Rozpoznávání textu je možné rozdělit na dva velké celky, na on-line a off-line rozpoznávání. Tyto dva přístupy se od sebe liší hlavně přístupem k datům. Při off-line rozpoznávání se pracuje se statickými daty. Může se tak získat pouze informace o „vzhledu“ textu (písmene) a pomocí ní pak vyhodnotit zadaný text. Při on-line rozpoznávání jsme schopni získat více informací o textu (písmenu), jelikož jsou data získávána od uživatele přímo. Je tak k dispozici celá trajektorie tahu a z ní je možné získat nejen informaci o „vzhledu“, ale i o směru a rychlosti tahu pera.

Kapitola se v první části zabývá předzpracováním [9, 7, 8, 1, 4]. Je to úprava vstupních dat do zpracovatelné podoby. Předzpracování je jiné ještě při on-line a off-line rozpoznávání, jelikož se zpracovávají odlišně získaná data a tím pádem mají různou informační hodnotu. V další části se pak kapitola zabývá dvěma metodami rozpoznávání, a to Neuronovými sítěmi [9, 2] a skrytými Markovovými modely [9, 2, 6, 3, 11].

2.1 Získávání dat pro off-line rozpoznávání

Při získávání off-line dat se většinou zpracovávají obrazová data. Taková data je nutné ještě před použitím upravit a vylepšit tak, aby z nich bylo možné získat co nejvíce relevantních informací. Takovéto úpravě obrazových dat se říká *Předzpracování obrazu*. Předzpracování obrazu (Image pre-processing) představuje operace se vstupním obrazem na nejnižší úrovni abstrakce - vstupem i výstupem jsou obrazová data. Předzpracování obrazu by nemělo zvyšovat informační obsah vstupního obrazu, ale naopak by mělo potlačit informace obrazu, které nejsou významné pro rozpoznávání vzorů. Úkolem předzpracování je vylepšení obrazových dat tak, že potlačuje nežádoucí deformace, nebo vylepšuje některé obrazové rysy důležité pro budoucí zpracování. Pro předzpracování obrazu se používá mnoho metod, přičemž se pro každý problém volí ty, které obraz pro tento daný problém nejlépe upraví. V následujících odstavcích jsou popsány některé z nich.

Transformace jasu pixelu mění hodnoty pouze jednoho pixelu. Do této kategorie patří korekce jasu, která vypočítá jas pixelu na základě jeho původního jasu a jeho polohy v obrázku. Také sem patří převedení barevného obrazu na obraz ve stupních šedi. Pro vypočítání hodnoty pixelu se ale může použít vlastností okolních pixelů.

Geometrická transformace slouží k eliminaci geometrických deformací, které vznikají při snímání obrazu.

Lokální předzpracování (filtrace) zpracovává malé okolí pixelu ze vstupního obrazu k vytvoření hodnoty pixelu ve výstupním obrazu. Tuto kategori lze ještě dále rozdělit na vyhlazování a spádové operátory. Vyhlazování se zaměřuje na potlačení šumu, či malých nepřesností v obraze. Bohužel při použití vyhlazování také dochází k rozmazání všech ostrých hran, které mohou být důležité při rozpoznávání. Spádové operátory jsou založeny na lokálních derivacích obrazu. Derivace je vysoká tam, kde se obraz skokem mění. Tohoto se využívá při detekci hran v obraze.

Renovace obrazu slouží k vylepšení obrazu pomocí znalosti jeho podstaty. Techniky používáné při renovaci lze také rozdělit do dvou kategorií: deterministické a stochastické metody. Pro použití deterministických metod je nutné znát, jak byl obraz poškozen (transformován). Původní obraz se z poškozeného pak získá pomocí aplikace inverzní transformace k transformaci poškození. Stochastické metody se snaží najít nejlepší obnovení za pomoci specifických stochastických kritérií. V některých případech je nutné nejprve odhadnout transformaci poškození.

Po provedené úpravě obrazových dat je nutné tato data pak ještě dále připravit pro zpracování klasifikátorem. Data je nutné rozdělit tak, aby klasifikátor mohl zpracovávat jen jednotlivé části. Takovéto dělení se nazývá segmentace. Segmentace obrazu je jeden z nejdůležitějších kroků vedoucích k rozboru zpracovávaných obrazových dat. Jeho hlavním cílem je rozdělit obraz do částí, které mají silný vztah s oblastmi, či objekty reálného světa (Například text se snažíme rozdělit na části, které odpovídají jednotlivým znakům).

2.2 Získávání dat pro on-line rozpoznávání

Při získávání dat v reálném čase jsme schopni získat obecně více relevantních informací pro rozpoznání textu než při získávání informací ze statického obrazu.

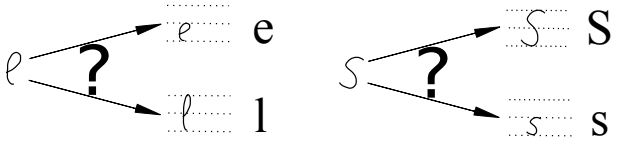
V dřívějších on-line rozpoznávacích systémech se používala slova jako základní jednotky rozpoznávání. Systémy, které byly založeny na rozpoznávání jednotlivých znaků, se začaly rozvíjet později. Modelováním písmen namísto celých slov se stávaly modely jednoduššími. Jejich složitost se nezvyšovala s velikostí slovníku. Navíc mohla být změna ve slovníku jednoduše převedena na změnu gramatických omezení systému, přičemž se model nezměnil.

Předzpracování on-line dat je podstatně jednodušší než předzpracování obrazu. Data popisující text jsou uchovávána ve formě bodů (přímek). Text pak charakterizuje soubor x a y souřadnic a stisk pera p v daném čase t [7] (psací pero přiložené na tablet, stisknutí tlačítka myši, aj.). Rukou napsaný text je popsán vzorkem vektorů $\mathbf{s}(t) = (x(t), y(t), p(t))$, který popisuje jednotlivé body trajektorie tahu. Konečnou fází předzpracování bývá normalizace, která převede rukou napsaný text na jednotnou velikost.

Po předzpracování následuje extrakce příznaků, kdy se z dat snažíme získat informace potřebné pro rozpoznávání. Podle [7] můžeme z dat získat až 24 on-line i off-line příznaků. Souhrnně je lze označit F ($F = f_1, f_2, \ldots, f_{24}$).

Extrahované on-line příznaky jsou:

- „stlačení“ pera, které indikuje, kdy se pero dotýkalo podložky (f_1)



Obrázek 2.1: Rozdíly ve velikosti písmen. Obrázek převzat z [7]

- rychlost, se kterou jsou body po sobě kresleny (f_2)
- souřadnice x a y ($f_{3,4}$)
- „směr psaní“ tahů, tj. úhel α kódovaný jako $sin(\alpha)$ a $cos(\alpha)$ ($f_{5,6}$)
- „zakřivení“, tj. rozdíl za sebou jdoucích úhlů $\Delta\alpha = \alpha(t) - \alpha(t-1)$, zakódované opět jako $sin(\alpha)$ a $cos(\alpha)$ ($f_{7,8}$). t reprezentuje označení tahů jdoucích po sobě.
- „vzhled trajektorie ν mezi dvěma body“ $\mathbf{s}(t-\tau)$ a $\mathbf{s}(t)$. Počítá se podle

$$\nu = \frac{\Delta y - \Delta x}{\Delta y + \Delta x}, \quad \begin{array}{l} \Delta x = x(t) - x(t-\tau) \\ \Delta y = y(t) - y(t-\tau) \end{array} \tag{2.1}$$

- „zkosení okolí“, tj. úhel φ mezi tahem $[\mathbf{s}(t-\tau), \mathbf{s}(t))]$, kde $\tau < t$ značí τ-tý vzorkovací čas před $\mathbf{s}(t)$, a linií označující spodní hranu, značený jako $sin(\varphi)$ a $cos(\varphi)$ ($f_{10,11}$)
- „kudrnatost okolí“, tj. délka trajetkorie normalizovaná hodnotou $max(|\Delta x|; |\Delta y|)$ (f_{12})
- průměrná čtvereční vzdálenost všech bodů v trajektorii a čáry $[\mathbf{s}(t-\tau), \mathbf{s}(t))]$ dává příznak (f_{13}).

Off-line příznaky jsou:

- 3×3 „kontextová mapa“ odkazující na 30×30 části obrazu právě napsaného písmene (f_{14-22})
- „horní a dolní dotahy písmene“ (tj. počet pixelů nad/pod aktuálním vzorkovaným bodem) ($f_{23,24}$)

Jak je popsáno v [8], je možné k příznakům přidat také ještě jeden, tzv. „*Příslušnost k vodícím čarám* (line-members)“. Mnoho písmen je rozlišeno ne pomocí jejich tvaru, ale pouze rozdílnou velikosti, jak je vidět na obrázku 2.1.

Napsaný text může být oddělen určitými liniemi v textu (jak ukazuje [1]). *Horní linie* leží na vrcholech vysokých písmen (například „M“ a „t“). *Střední linie* je definována vrcholy malých písmen (například „s“ a „a“). *Hlavní linie* je ta, na které jsou písmena napsaná. A *spodní linie* spojuje spodní body písmen s výběžkem dolů (například „q“ a „y“).

Pokud se podíváme na znaky na obrázku 2.1, není možné rozhodnout mezi „e“ a „l“ nebo „S“ a „s“, bez znalosti kontextu (okolí písmene). Avšak, pokud jsme si definovali linie ohraničující písmena, známe relativní velikost těchto písmen a můžeme tato písmena od sebe odlišit. Tento příznak (f_{25}) pak obsahuje pro každý z N bodů $s(n)$ ležících na trajektorii písmene $\mathbf{S} = [s(1), ..., s(N)]$ následující hodnoty:

$$f_{25} = \begin{cases} 0 & \text{pokud s(t) neleží na žádné vodící linii} \\ 1 & \text{pokud s(t) leží na horní linii} \\ 2 & \text{pokud s(t) leží na střední linii} \\ 3 & \text{pokud s(t) leží na hlavní linii} \\ 4 & \text{pokud s(t) leží na spodní linii} \end{cases} \tag{2.2}$$

2.3 Metody rozpoznávání

Pro rozpoznávání se používá nepřeberné množství metod. Zde se budeme zabývat dvěma, v dnešní době nejvíce používanými metodami: Neuronovými sítěmi a skrytými Markovovými modely.

Neuronové sítě

Pojem *neuronové sítě* se začal používat při pokusech formálně matematicky popsat zpracování informací v biologických systémech. Poté se začal hojně používat pro mnoho rozdílných modelů, z nichž se mnohé vzdálily jejich biologickým obrazům. My se zaměříme na neuronové sítě, jako účinný model pro statistické rozpoznání vzorů.

Model neuronu (perceptronu)

Naprostá většina neuronových sítí je založena na kombinaci elementárních článků (neuronů). Ke každému neuronu (perceptronu) je připojeno množství vstupů, ze kterých je neuron schopen generovat jeden výstup. S každým vstupem neuronu je spojena váha. Výstup neuronu se pak počítá jako funkce těchto váhovaných vstupů. Tato funkce může generovat diskrétní nebo spojité výstupy neuronu. Jednoduchý neuron je ukázán na obrázku 2.2. Vstup neuronu můžeme popsat rovnicí 2.3, kde jsou vstupy označeny v_1, v_2, v_3, ..., a váhy w_1, w_2, w_3, ... Kompletní vstup neuronu (bázová funkce x) je pak

$$x = \sum_{i=1}^{n} v_i w_i - \Theta \tag{2.3}$$

kde Θ je práh spojený s neuronem. S neuronem je také spojena aktivační (přenosová) funkce $f(x)$, která produkuje výstup. Tento výstup může být diskrétní, např.

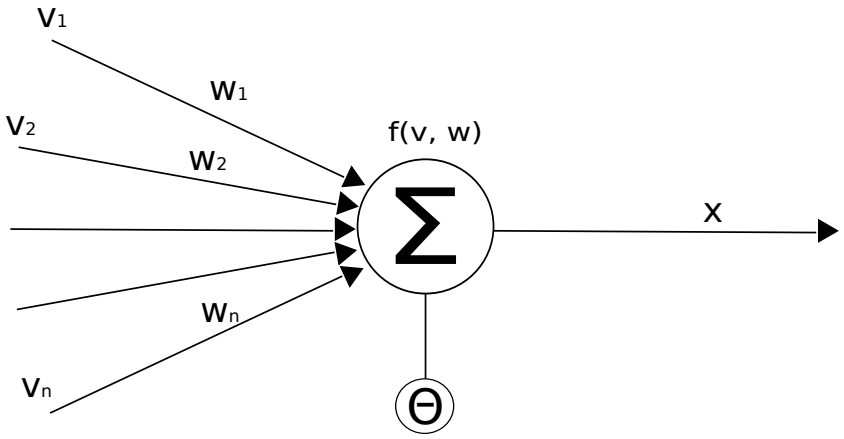
$$f(x) = \begin{cases} 0 & if \quad x \leq 0 \\ 1 & if \quad x > 0 \end{cases} \tag{2.4}$$

nebo spojitý, např.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.5}$$

Neuronová síť

Síť neuronů je tvořena množstvím jednotlivých perceptronů navázaných na sebe (výstup neuronu bude vstupem pro jiný neuron). Takováto síť napodobuje propojení mnoha jednoduchých neuronů v lidském mozku. Podobně jako spojení mezi neurony můžeme do sítě zavést několik vnějších vstupů a z ní vyvést také několik výstupů. To, co leží mezi vstupy



Obrázek 2.2: Model neuronu

a výstupy, je potom specifikováno právě sítí. Může to být obrovské množství složitě spojených neuronů, nebo to také může být jeden neuron, který rozhoduje o výstupech. Naprostá většina používaných sítí je tvořena několika vrstvami perceptronů, které jsou mezi sebou propojeny systémem každý s každým. Podle topologie můžeme sítě rozdělit na přímé (neurony jsou spojeny za sebou) a rekurentní (neurony mohou být spojeny s předcházejícími). Vstupní data do sítě označujeme jako vstupní vektor, výstupní data jako výstupní vektor.

Trénování sítě

Cílem učení sítě je nastavit síť na poskytnutých datech tak, aby pak mohla rozdělovat testovací data do tříd co nejpřesněji. Podle způsobu trénování bychom mohli trénování rozdělit na trénování s učitelem a bez učitele.

Při trénování s učitelem je využita zpětná vazba. Na vstup dáme trénovací vzorek a síť vygeneruje výsledek podle aktuálního nastavení. Tento výsledek porovnáme s požadovaným výsledkem a určíme chybu. Pak přenastavíme váhy, či hodnoty prahů neuronů tak, abychom snížili hodnotu chyby. Toto opakujeme dokud nebude vypočítaná chyba menší než minimální povolená (námi stanoveno).

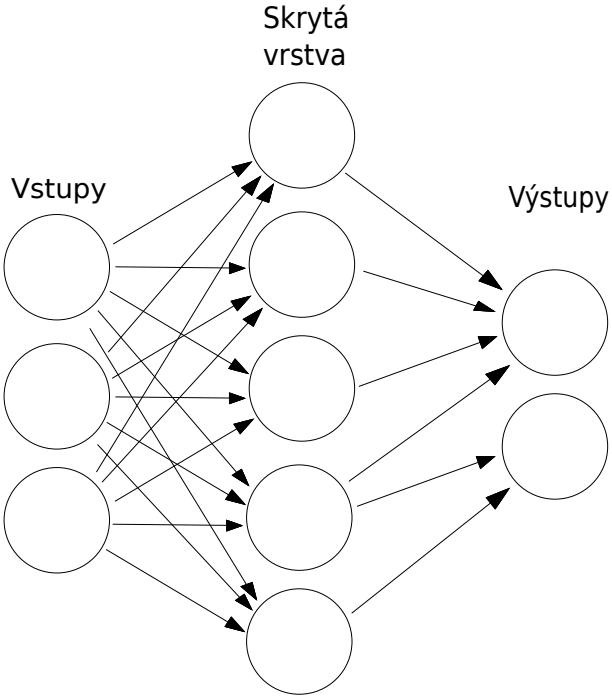
Při trénování bez učitele nesledujeme výstupy. Síti předáváme sady vstupů a síť si je sama snaží rozdelit do tříd.

Neuronová síť se trénuje pomocí učících algoritmů (iterativních). Tyto algoritmy během učení mění váhy jednotlivých spojů.

Mezi první způsoby učení patří *Hebbův zákon* z roku 1949. Vychází z představy trénování neuronů v mozku, že při učení se posilují vazby mezi neurony, které byly aktivovány. Pokud je k dispozici vstup x, hodnota y udávající, do které třídy vstup patří, a hodnota váhy w daného vstupu, lze vypočítat novou hodnotu váhy. V (umělých) neuronech by se dalo toto pravidlo pro více vstupů formulovat jako

$$\mathbf{w}_{i+1} = \mathbf{w}_i + y_i \ \mathbf{x}_i \tag{2.6}$$

Nová hodnota váhy je tak vypočítána pomocí předešlé váhy a vstupu, o němž víme, do které třídy má být zařazen.



Obrázek 2.3: Model neuronové sítě

Dopředné neuronové sítě

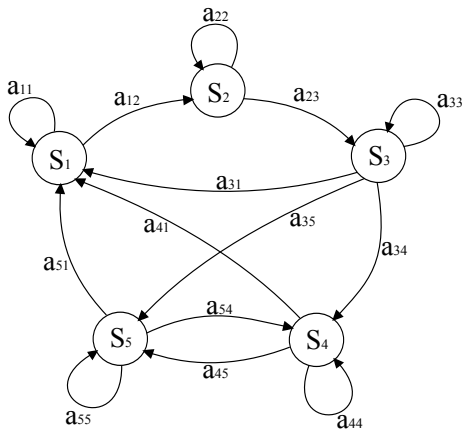
Abychom mohli síť neuronů označit za dopřednou, musí být tato síť přímá a musí obsahovat alespoň jednu skrytou vrstvu perceptronů. Dopředné neuronové sítě se trénují algoritmem zpětného šíření chyby (back propagation), kdy se po porovnání výstupu s očekávaným výstupem upravují váhy nejdříve a nejvíce v poslední vrstvě, méně v předposlední, atd.

Markovovy modely

Markovův model bychom mohli popsat jako diskrétní stavový model. Tento model se může nacházet v různých stavech mezi kterými přechází pomocí diskrétních kroků. Diskrétní krok znamená přímý přechod z jednoho stavu do jiného. Změny stavů v systému se nazývají *přechody* a pravděpodobnosti spojené s různými změnami stavů se nazývají *přechodové pravděpodobnosti*.

Základní teorie Markovových modelů poprvé teoreticky popsal Andrey Markov v roce 1906. Tyto modely tedy nesou jeho jméno - *Markovovy modely*. Velkého rozšíření a používání pro rozpoznávání se ale dočkaly až v sedmdesátých letech minulého století. Dnes je mnoho různých druhů skrytých Markovových modelů. Používají se u většiny klasifikačních úloh, což poukazuje na jejich obecnost a relativní jednoduchost. Popisují pravděpodobnostní distribuci mezi potencionálně nekonečným počtem sekvencí. Například při rozpoznávání řeči se pozoruje sekvence jednotlivých hlásek (fonémů), při rozpoznávání rukou psaného textu se pozoruje sekvence získaných příznaků (pozice, rychlost a směr kreslení). Tuto sekvenci rozpoznává systém založený na Markovových modelech. Jeho výstupem je tedy číslo udávající pravděpodobnost, se kterou vstupní sekvence odpovídá natrénovanému modelu.

Diskrétní Markovův model si můžeme představit jako sadu N odlišných stavů (S_1, S_2, S_3, …, S_N) jak je znázorněno na obr.2.4. V jednotlivých intervalech (nemusí být vázany



Obrázek 2.4: Markovův model s pěti stavy a se znázorněnými přechody. Obrázek převzat z [6]

na čas) systém přejde do jiného stavu (může se vratit do stejného) podle sady pravděpodobností, které jsou spojeny s tímto stavem. Jednotlivé okamžiky, ve kterých se mění stavy označíme jako $t = 1, 2, 3, ...$ a aktuální stav v čase t jako q_t. Úplný pravděpodobnostní popis systému by vyžadoval specifikaci aktuálního stavu stejně jako všech jeho předcházejících stavů. Ale pro případ diskrétního Markovova modelu prvního řádu můžeme celkový pravděpodobnostní popis zkrátit na charakteristiku aktuálního a předchozího stavu.

$$P[q_t = S_i | q_{t-1} = S_j, q_{t-2} = S_k, ...] = P[q_t = S_i | q_{t-1} = S_j] \tag{2.7}$$

Mimoto můžeme požadovat procesy, ve kterých je pravá strana v 2.7 nezávislá na čase. Sadu přechodových pravděpodobností a_ij můžeme napsat ve tvaru

$$a_{ij} = P[q_t = S_i | q_{t-1} = S_j] \quad , \quad 1 \leq i, j \leq N \tag{2.8}$$

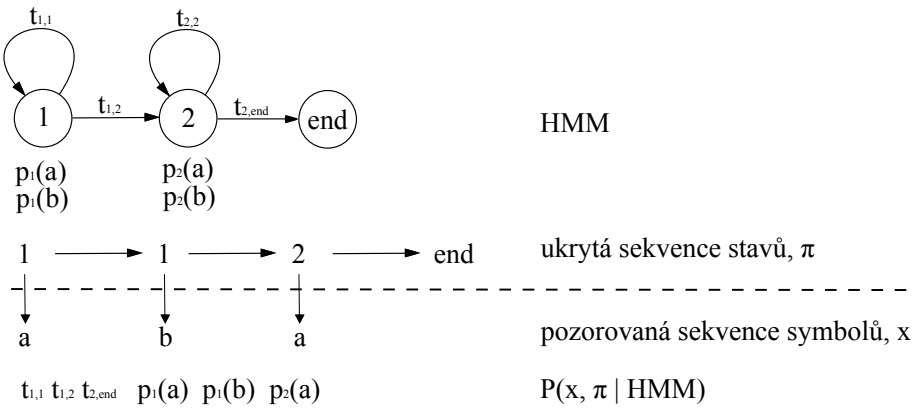
a koeficienty přechodových pravděpodobností musí splňovat:

$$a_{ij} \geq 0 \quad , \quad \sum_{j=1}^{N} a_{ij} = 1 \tag{2.9}$$

Skryté Markovovy modely (Hidden Markovs models - HMM)

Jméno „Skryté Markovovy modely“ vzniklo z faktu, že sekvence stavů je Markovův model prvního řádu, ale pozorován je pouze sled po sobě jdoucích symbolů na výstupu, jak je vidět na obrázku 2.5.

Příklad jednoduchého skrytého Markovova modelu, který modeluje sekvenci dvou znaků (a, b), je vidět na obrázku 2.5. Tento jednoduchý HMM může být vhodný model pro problém, ve kterém uvažujeme sekvenci začínající jedním symbolem (například a) a pak se změní na jiný symbol (například b). HMM se kládá ze dvou stavů spojených přechody. Každý stav vysílá pravděpodobnost, se kterou tvoří (nebo porovnává) nové symboly. Nyní je vhodné uvažovat o HMM jako o modelu vytvářejícím sekvenci symbolů. Začínáme v počátečním stavu. Vybereme si nový stav s určitou přechodovou pravděpodobností (buď zůstaneme ve stavu 1 s přechodovou pravděpodobnosí $t_{1,1}$, nebo přejdeme do stavu 2 s přechodovou pravděpodobností $t_{1,2}$). Pak můžeme generovat výstupy s pravděpodobností specifokavané aktuálním stavem (například si vybereme a s pravděpodobností $p_1(a)$). Opakujeme



Obrázek 2.5: Znázornění přechodů ve skrytém Markovově modelu. Obrázek převzat z [3]

přecházení mezi stavy (vysílání pravděpodobností) dokud nedosáhneme koncového stavu. Na konci tohoto procesu máme skrytou sekvenci stavů, které nepozorujeme a sekvenci symbolů, která nás zajímá.

Tři základní problémy při používání HMM

Je důležité si uvědomit, že při používání skrytých Markovových modelů musíme podle [6] čelit třem základním problémům.

První problém První problém nastává při vyhodnocování. Jak zjistit, že získaná sekvence dat odpovídá natrénovanému modelu? Pokud jsou k dispozici dva různé natrénované modely a sekvence vypozorovaných dat, tato otázka se ptá na to, který z těchto modelů lépe popisuje tuto sekvenci dat. Například pokud jsou k dispozici modely tvarů dvou písmen a sekvence bodů reprezentujících písmeno, které se právě rozpoznává, tak se musí rozhodnout, který z modelů lépe popisuje rozpoznávané písmeno.

Pokud chceme získat praděpodobnost s jakou pozorovaná sekvence $O = o_1, o_2, o_3, \cdots, o_n$ odpovídá modelu λ (t.j. $P(O|\lambda)$), nejjednodušší cesta je vypočítat všechny možné průchody modelem pro pozorovanou sekvenci O délky T (T je počet pozorování).

Uvažujme jednu takovou pevnou sekvenci stavů

$$Q = q_1, q_2, q_3, \cdots, q_n \tag{2.10}$$

kde q_1 je počáteční stav. Pravděpodobnost pozorované sekvence O pro jednu stavovou sekvenci (průchod modelem) ve 2.10 je

$$P(O|Q, \lambda) = \prod_{t=1}^{T} P(O_t|q_t, \lambda) \tag{2.11}$$

nebo

$$P(O|Q, \lambda) = b_{q_1}(O_1) \cdot b_{q_2}(O_2) \cdots b_{q_T}(O_T) \tag{2.12}$$

kde předpokládáme statistickou nezávislost vstupních dat. Vzorce 2.11 a 2.12 vyjadřují celkovou pravděpodobnost, kterou získáme v jednotlivých stavech. Ještě je ale nutné získat pravděpodobnost na přechodech Q, kterými stavová sekvence prošla.

$$P(Q|\lambda) = \pi_{q_1} a_{q_1 q_2} a_{q_2 q_3} \cdots a_{q_{T-1} q_T} \tag{2.13}$$

Spojením pravděpodobností O a Q získáme pravděpodobnost pro jeden z možných průchodů (jednu stavovou sekvenci Q) pozorované sekvence HMM modelem

$$P(O, Q|\lambda) = P(O|Q, \lambda)P(Q, \lambda) \tag{2.14}$$

Celkovou pravděpodobnost O s jakou sekvence pozorování odpovídá modelu získáme jednoduše součtem pravděpodobností pro všechny možné kombinace stavových sekvencí

$$P(O|\lambda) = \sum_{Q} P(O|Q, \lambda)P(Q|\lambda) \tag{2.15}$$

Výklad počítání v předchozích vzorcích je následující. Nejprve (v čase $t = 1$) jsme ve stavu q_1 s pravděpodobností π_{q_1} a generujeme symbol O_1 (v tomto stavu) s pravděpodobností $b_{q_1}(O_1)$. Pak se změní čas na $t + 1$ ($t = 2$) a přejdeme do stavu q_2 ze stavu q_1 s pravděpodobností $a_{q_1 q_2}$ a generujeme symbol O_2 s pravděpodobností $b_{q_2}(O_2)$. Takto pokračujeme, dokud naposledy nepřejdeme (v čase T) ze stavu q_{T-1} do stavu q_T s pravděpodobností $a_{q_{T-1} q_T}$ a negenerujeme symbol O_T s pravděpodobností $b_{q_T}(O_T)$.

Takovýto výpočet je ale velice náročný. Jeho složitost je $2T \cdot N^T$, protože pro každý čas $t = 1, 2, 3, \ldots, T$ se můžeme nacházet v jakémkoliv z N stavů (tj. N^T možných stavových sekvencí) a pro každou sekvenci máme $2T$ výpočtů (podle 2.16). Naštěstí toto řeší *forward-backward* procedura. Skládá se z dopředné (forward) a zpětné (backward) proměnné. Dopředná proměnná $\alpha_t(i)$ je definovaná jako

$$\alpha_t(i) = P(O_1\ O_2 \ldots O_t, q_t = S_i|\lambda) \tag{2.16}$$

což je částečná pravděpodobnost pozorovací sekvence do času t a stavu S_i v čase t, kterou dává model λ. Spočítá se v následujících krocích:

1. Inicializace. Inicializuje dopředné pravděpodobnosti stavu S_i a počáteční pozorování O_1.

2. Dosazení. Do stavu S_j se můžeme dostat z N různých stavů S_i (předcházející stavy). Dopřednou pravděpodobnost v tomto stavu tedy získáme součtem dopředných pravděpodobností všech jeho předcházejících stavů vynásobených přechodovými pravděpodobnostmi (k tomuto stavu). Pak tedy můžeme postupně spočítat dopředné pravděpodobnosti všech stavů systému.

3. Ukončení. Nakonec spočítáme poslení dopřednou pravděpodobnost konečného stavu. Počítá se stejně jako předchozí. Tuto dopřednou pravděpodobnost nazýváme *celkovou dopřednou pravděpodobností.*

Stejně, jako se počítá dopředná ptavděpodobnost, lze spočítat i zpětnou pravděpodobnost $\beta_t(i)$. Pouze se nepočítá od počátečního stavu ke koncovému, ale naopak. *Celková zpětná pravděpodobnost* $\beta_T(i)$ je pak zpětná pravděpodobnost prvního stavu.

Druhý problém Pokud máme sekvenci pozorování $O = O_1, O_2, \ldots, O_T$ a model λ, jak vybrat odpovídající sekvenci stavů, která nejlépe popisuje pozorovanou sekvenci? Potíž je v hledání *optimální* stavové sekvence. Je zde několik možných optimalizačních kritérií. Například jedno možné optimalizační kritérium může být vybrání stavů q_t, které jsou nejpravděpodobnější. Toto kritérium maximalizuje očekávaný počet správně ohodnocených stavů. Pro implementování tohoto kritéria definujeme proměnnou

$$\gamma_t(i) = P(q_t = S_i|O, \lambda) \tag{2.17}$$

která je pravděpodobností „setrvání“ ve stavu S_i v čase t danou pozorovanou sekvencí O a modelu λ. Rovnici 2.17 lze vyjádřit jednoduše pomocí dopředných a zpětných (forward-backward) proměnných

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{P(O|\lambda)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)} \tag{2.18}$$

kde $\alpha_t(i)$ odpovídá za část pozorované sekvence $O_1, O_2, \ldots, O_t$ a stav S_i v čase t, a $\beta_t(i)$ odpovídá za zbývající část pozorované sekvence $O_{t+1}, O_{t+2}, \ldots, O_T$ nabízenou za stavem S_i v čase t. Normalizační jmenovatel $P(O|\lambda)$ zajišťuje, aby výsledek byl v rozmezí $0 \ldots 1$.

Při použití $\gamma_t(i)$ můžeme vypočítat jednotlivé nejpravděpodobnější stavy q_t v čase t jako

$$q_t = \underset{1 \le i \le N}{\operatorname{argmax}}[\gamma_t(i)] \tag{2.19}$$

Ačkoliv 2.3 maximalizuje průchod HMM přes nejlepší stavy (vybíráním nejlepšího stavu pro každý okamžik), mohou nastat nějaké problémy s vyhodnocováním sekvence stavů. Například pokud bude mít HMM mezi některými stavy přechodovou pravděpodobnost rovnou nule ($a_{ij} = 0$ pro nějaká i a j), pak vyhodnocená „optimální“ sekvence stavů nemusí být validní pro dané HMM. To je důsledkem toho, že rozhoduje pouze o nejlepším stavu v každém okamžiku a již nezjišťuje pravděpodobnost, s jakou taková sekvence stavů může nastat.

Jedním z možných řešení může být změna optimalizačního kritéria. Například se může maximalizovat pravděpodobnost pro dvojce stavů (q_t, q_{t+1}), nebo trojce stavů (q_t, q_{t+1}, q_{t+2}), atd. Toto kritérium může být použitelné pro mnoho aplikací. Ale nejrozšířenějším přístupem je hledání „jediné“ nejlepší stavové sekvence (cesty), tj. maximalizování $P(Q|O, \lambda)$, což je stejné jako maximalizování $P(O|Q, \lambda)$. Formální techniky pro hledání této jediné nejlepší stavové sekvence se nazývají *Viterbiho algoritmus*.

Viterbiho algoritmus: K nalezení jediné nejlepší sekvence stavů $Q = \{q_1, q_2, \ldots, q_T\}$, pro danou pozorovanou sekvenci $O = \{O_1, O_2, \ldots, O_T\}$ je třeba definovat velikost

$$\delta_t(i) = \max_{q_1, q_2, \ldots, q_{t-1}} P[q_1, q_2, \ldots, q_t = i, O_1, O_2, \ldots, O_t|\lambda] \tag{2.20}$$

kde $\delta_t(i)$ je nejlepší skóre (nejvyšší pravděpodobnost) po projití části HMM v čase t a odpovídá prvním t pozorováním končících ve stavu S_i. Pouze dosazením získáme

$$\delta_{t+1}(j) = [\max_i \delta_t(i)a_{ij}] \cdot b_j(O_{t+1}) \tag{2.21}$$

K opravdovému získání stavové sekvence je třeba uchovávat dosavadní projité dráhy, ze kterých vybíráme nejlepší 2.21 pro každé t a j. Toto se provádí přes pole $\psi_t(j)$. Kompletní procedura pro hledání nejlepší stavové sekvence může být popsáná následujícími čtyřmi kroky:

1. Inicializace:

$$\begin{aligned} \delta_1(i) &= \pi_i b_i(O_1), \ 1 \le i \le N \\ \psi_1(i) &= 0 \end{aligned} \tag{2.22}$$

2. Rekurze:

$$\delta_t(j) = \max_{1\leq i\leq N}[\delta_{t-1}(i)a_{ij}]b_j(O_t),\ \begin{array}{rcl} 2 & \leq & t \leq T \\ 1 & \leq & j \leq N \end{array} \tag{2.23}$$

$$\psi_t(j) = \operatorname*{argmax}_{1\leq i\leq N}[\delta_{t-1}(i)a_{ij}],\ \begin{array}{rcl} 2 & \leq & t \leq T \\ 1 & \leq & j \leq N \end{array} \tag{2.24}$$

3. Ukončení:

$$P^* = \max_{1\leq i\leq N}[\delta_T(i)] \tag{2.25}$$

$$P_T^* = \operatorname*{argmax}_{1\leq i\leq N}[\delta_T(i)] \tag{2.26}$$

4. Zpětné vyhledání cesty (stavové sekvence):

$$q_t^* = \psi_{t+1}(q_{t+1}^*),\ t = T-1, T-2, \ldots, 1 \tag{2.27}$$

Viterbiho algoritmus je podobný (vyjma zpětného vyhledávání) implementaci počítání dopředné pravděpodobnosti. Největší rozdíl je ve vybírání nejlepšího stavu ze všech předchozích namísto sčítání jejich pravděpodobností.

Třetí problém Tímto třetím a nejtěžším problémem při používání HMM je získat způsob, jakým určit parametry modelu $\lambda = (A, B, \pi)$, aby byla maximalizována pravděpodobnost pozorované sekvence generované modelem $P(O|\lambda)$. Neexistuje žádná známá cesta analytického řešení modelu, která maximalizuje pravděpodobnost pozorované sekvence. Ve skutečnosti, předáním jakékoliv konečné sekvence pozorování jako trénovacích dat, neexistuje žádný optimální způsob jak odhadnout parametry modelu. Nicméně je možné zvolit $\lambda = (A, B, \pi)$ takové, že lokálně maximalizuje $P(O|\lambda)$ použitím iterativní procedury, jako je *Baum-Welchova* metoda (nebo ekvivalentní EM (Expectation-Maximizaton) metoda).

Baum-Welchova restimace: Pro popis procedury pro „znovuodhadnutí“ (iterativního aktualizování a zlepšování) parametrů modelu HMM je třeba nejdříve definovat $\xi_t(i, j)$, pravděpodobnost bytí ve stavu S_i v čase t a stavu S_j v čase $t+1$ dávané modelem a pozorovanou sekvencí

$$\xi_t(i, j) = P(q_t = S_i, q_{t+1} = S_j|O, \lambda) \tag{2.28}$$

Pomocí dopředných a zpětných proměnných můžeme napsat $\xi_t(i, j)$ ve formě

$$\xi_t(i, j) = \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{P(O|\lambda)} = \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)} \tag{2.29}$$

kde je čitatel pouze $P(q_t = S_i, q_{t+1} = S_j, O|\lambda)$ a dělení $P(O|\lambda)$ dává požadovaný rozsah pravděpodobnosti.

Dříve byla definována $\gamma_t(i)$ jako pravděpodobnost bytí ve stavu S_i v čase t dávaná pozorovanou sekvencí a modelem. Proto lze nyní spojit $\gamma_t(i)$ s $\xi_t(i, j)$ sečtením přes j:

$$\gamma_t(i) = \sum_{j=1}^{N}\xi_t(i, j) \tag{2.30}$$

Pokud je $\gamma_t(i)$ sečtená přes indexy času t, dostáváme tak číslo, které může být interpretováno jako počet „časů“ (časových okamžiků), ve kterých bude stav S_i navštíven, nebo ekvivalentně počet přechodů provedených ze stavu S_i. Podobně součet $\xi_t(i,j)$ přes t (od $t=1$ do $t=T-1$) může být interpretován jako očekávaný počet přechodů ze stavu S_i do stavu S_j. To je

$$\sum_{t=1}^{T-1} \gamma_t(j) = \text{ očekávaný počet přechodů z } S_i \tag{2.31}$$

$$\sum_{t=1}^{T-1} \xi_t(i,j) = \text{ očekávaný počet přechodů z } S_i \text{ do } S_j \tag{2.32}$$

Použitím předchozích rovnic (a návrhu počítání výskytu událostí) lze získat metodu pro reestimaci (přehodnocení) parametrů HMM. Sada dostupných reestimačních rovnic pro π, A a B jsou

$$\bar{\pi}_i = \text{ očekávaný počet opakování ve stavu } S_i \text{ v čase } (t=1) = \gamma_1(i) \tag{2.33}$$

$$\bar{a}_i j = \frac{\text{očekávaný počet přechodů ze stavu } S_i \text{ do stavu } S_j}{\text{očekávaný počet přechodů ze stavu } S_i} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(j)} \tag{2.34}$$

$$\begin{aligned} \bar{b}_j(k) &= \frac{\text{očekávaný počet opakování ve stavu } j \text{ a pozorovaného symbolu } v_k}{\text{očekávaný počet opakování ve stavu } j} \\ &= \frac{\sum_{t=1, O_t=v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \end{aligned} \tag{2.35}$$

Definováním tohoto modelu jako $\lambda = (A, B, \pi)$ a používáním tak, že se počítá pravá strana 2.33 - 2.35 a definováním reestimačního modelu jako $\bar{\lambda} = \bar{A}, \bar{B}, \bar{\pi}$, jak je určeno v levých stranách 2.33 - 2.35 bylo zjištěno [6]:

1. Počáteční model λ definuje rozhodující bod pravděpodobnostní funkce v případě, že $\bar{\lambda} = \lambda$
2. Model $\bar{\lambda}$ je pravděpodobnější než model λ v případě, že $P(O|\bar{\lambda}) > P(O|\lambda)$, to znamená, že byl nalezen nový model $\bar{\lambda}$, ze kterého je pozorovaná sekvence generována s větší pravděpodobností.

Na základě předchozí procedury je iterativně použito $\bar{\lambda}$ na místě λ a přepočítává se reestimace. Pak lze vylepšit pravděpodobnost sekvence O sledovanou v modelu. Pravděpodobnost je vylepšována, dokud není dosažen nějaký rozhodující bod. Konečný výsledek této reestimační procedury se nazývá maximální pravděpodobnost odhadu HMM.

Reestimační rovnice 2.33 - 2.35 mohou být získány přímo maximalizací Baumovi pomocné funkce

$$Q(\lambda, \bar{\lambda}) = \sum_{Q} P(Q|O,\lambda) \log[P(O,Q|\bar{\lambda})] \tag{2.36}$$

přes $\bar{\lambda}$. Bylo prokázáno [6], že maximalizace $Q(\lambda, \bar{\lambda})$ vede ke zvýšení pravděpodobnosti

$$\max_{\bar{\lambda}}[Q(\lambda, \bar{\lambda})] \Rightarrow P(O|\bar{\lambda}) \geq P(O|\lambda) \tag{2.37}$$

Nakonec konverguje pravděpodobnostní funkce k rozhodujícím bodům.

Kapitola 3

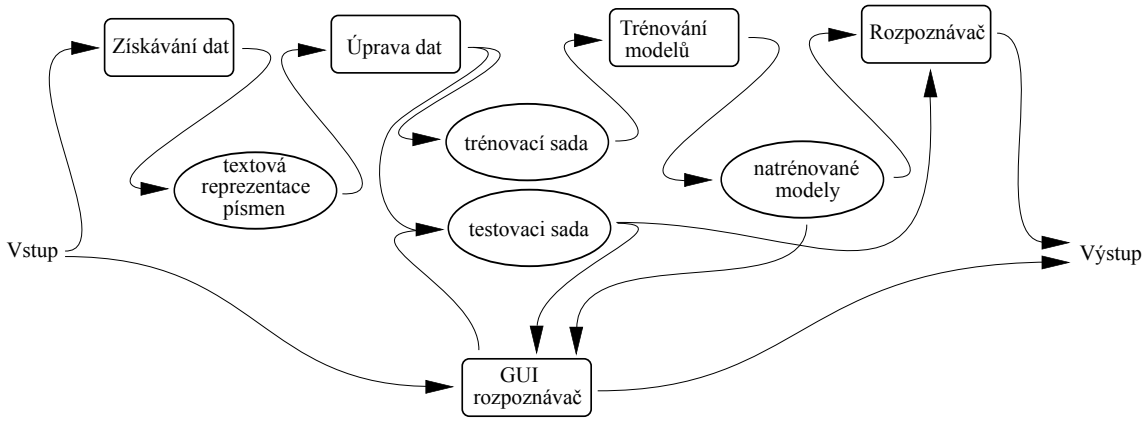
Program pro rozpoznávání rukou psaného textu

Tato kapitola popisuje implementaci zadaného úkolu. Popisuje postupy, které byly využity. Zabývá se rozborem jednotlivých částí rozpoznávání. Jako sada ke klasifikaci byly zvoleny velké znaky anglické abecedy (A - Z). Pro klasifikaci těchto znaků byly využity skryté Markovovy modely.

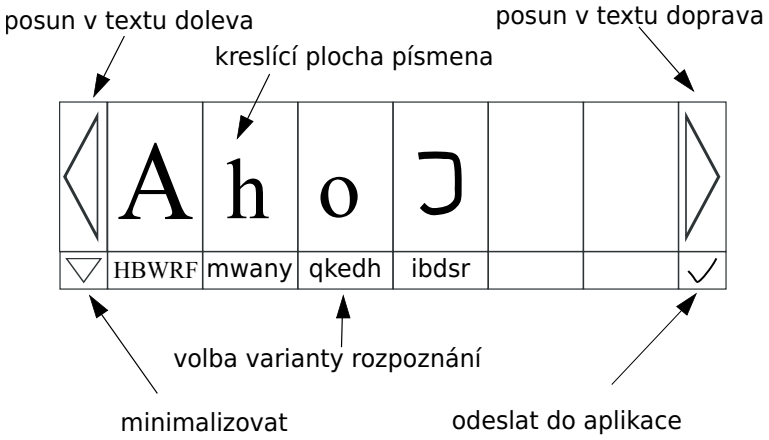
Při implementaci jednotlivých částí rozpoznávače byly použity programovací jazyky C a C++. Pro vytvoření grafického uživatelského rozhraní (GUI) pak byl využit toolkit GTK+, který je sadou knihoven používajících programovací jazyk C.

3.1 Návrh programu pro rozpoznávání

Pro rozpoznávání bylo nutné navrhnout celý systém, který bude schopen získat data. Na získaných datech se pak bude schopen natrénovat (naučit) a poté bude schopen vyhodnotit, jak se trénování podařilo. Podstatnou částí tohoto systému pak měla být aplikace, která by byla schopna interaktivně zpracovávat a vyhodnocovat předávaná data. Tato aplikace by pak mohla sloužit jako rozpoznávač textu pro dotykové mobilní telefony (např. pro psaní krátkých textových zpráv). Návrh celého systému je znázorněn na obr. 3.1. Propojení systému se během vývoje nezměnilo, proto tento obrázek popisuje i konečnou verzi systému.



Obrázek 3.1: Schéma vypracovaného rozpoznávacího systému



Obrázek 3.2: Návrh vytvořeného GUI rozhraní

Uživatelské rozhraní Při tvorbě aplikace, která by měla být spuštěna v mobilním telefonu je nutné brát v potaz to, že není dostupná klasická klávesnice a je důležité dbát na odlišné (specifické) ovládání. Především jsou takové přístroje většinou ovládány elektronickým perem, díky kterému napsaný text věrně odpovídá klasickému psaní na papír. S tímto perem se ale ovládá program poněkud odlišně než na počítači. Bylo proto nutné vytvořit jednoduchou a intuitivní aplikaci, která by měla jednoduché ovládání. Návrh takovéto aplikace je vidět na obr. 3.2.

Hlavní částí programu jsou „boxy“, kam je možné psát písmena (text), která mají být rozpoznána. Pro jednoduchost jsem se rozhodl, že by se rozpoznávání znaků mělo provádět na základě časového limitu, který uběhne od konce psaní. Rozpoznaný znak by pak měl být vepsán namísto nakresleného znaku, jak je znázorněno na obrázku. Změna písmen na velká, nebo malá by se měla provádět na základě klepnutí na rozpoznané písmeno. Smazání písmene (boxu) by mělo být možné pomocí tahu přes toto pole (vodorovné přeškrtnutí pole). Pod takto rozpoznaným písmenem by pak měly být zobrazeny i další varianty rozpoznaných písmen, kterými by pak bylo možné změnit špatně rozpoznaný znak. Rozpoznaný text by pak mohl být přenesen do aplikace, která s textem dále pracuje.

3.2 Vytvořený systém

Na obr. 3.1 je názorně zobrazeno propojení jednotlivých částí rozpoznávače mezi sebou. Jako vstup se myslí uživatelem napsaná sada znaků, a to buď pomocí programu pro získávání datové sady, nebo přímo do GUI rozpoznávače, jako data písmene, které se má rozpoznat. Nejdříve vstupní data od uživatele dostává program pro získávání dat. Ten tato data pouze převede do textové podoby, se kterou bude jednoduché dále pracovat. Vytvoří tedy textový soubor, který je nutné předat dále programu pro úpravu dat. Ten data upraví do vhodného formátu pro trénování. Výstupem tohoto programu je adresářová struktura s textovými soubory s daty, které se pak předají dále ke trénování a s daty pro testování. Trénovací sada je připravena tak, aby mohla být jednoduše předána „trénovadlu“ [5].

Výstupem tohoto trénovadla je pak sada souborů obsahující natrénované modely, které je nutné předat již rozpoznávači spolu s testovacími daty. Rozpoznávač pak testovací data otestuje na natrénovaných modelech a jako výstup vypíše úspěšnosti, s jakými byly znaky rozpoznány. Program s grafickým rozhraním si testovací data tvoří sám na základě znaků

nakreslených uživatelem. Je mu ale nutné předat natrénované modely, na kterých příchozí data testuje. Výstupem tohoto programu je již souvislý text napsaný pomocí rozpoznaných znaků.

Získávání datové sady Pro zpracování dat písma je nutné získat vlatní data od uživatele a převést je do podoby, se kterou se bude s daty lépe pracovat.

Jako datová sada písmen byla zvolena velká písmena anglické abecedy (A - Z). Všechna písmena z takto zvolené datové sady leží mezí horní a hlavní linii textu (viz. obr 2.1) a zabírají tak 1. a 2. sektor ležící mezi nimi. Narozdíl od toho, malá písmena zasahují různě do všech sektorů (např. „a“ leží ve 2. sektoru, „f“ v 1. - 3. sektoru). Proto bylo možné tato data normalizovat a převést je tak do jednotné velikosti.

Potřebná data pro rozpoznávání byla vytvořena pomocí *touch screenu*. Část dat se použila pro trénovávní modelů a zbývající část se pak použila pro vyhodnocení úspěšnosti trénování a jako vstupní data do *gui programu*. Byla vytvořena sada vzorů pro každé písmeno. Tato prvotní sada obsahovala okolo 240 obrázků každého písmene. Při tvoření této sady ale vzniklo mnoho „nepovedených“ znaků. Bylo proto nutné všechna trénovací i testovací data zkontrolovat vizuálně a odstranit nevhodné tvary, aby natrénovaný model co nejvíce odpovídal skutečnému znaku. Po těchto provedených úpravách se počet vzorů pro jedno písmeno zredukoval na počet mezi 177 a 236. Aby bylo trénování i testování srovnatelné mezi písmeny, zvolil jsem pro každé písmeno stejný počet vzorů pro trénování i pro testování. Prvních 100 bylo použito pro natrénování modelů a na zbývajících 75 se natrénované modely testovaly.

Program po přeložení a spuštění získává od uživatele reprezentaci jednotlivých znaků abecedy. Nakreslené znaky pak převádí do textové podoby jako seznam za sebou jdoucích tahů uživatele. Tato data pak uloží do souboru, který se dále zpracovává.

Při získávání dat vyvstává problém, jaká data uchovávat, aby bylo možné získat o nakresleném vzoru co nejvíce informací. Zvolený přístup je jednoduchý a lze z něj pak dle pořeby získat všechny příznaky, které lze v klasifikaci uplatnit.

Trénování Před začátkem samotného trénování je důležité si data připravit. Proto byl vytvořen program, který převezme textovou reprezentaci dat a upraví je. Úprava dat je nezbytná pro správné natrénování modelu. Proto tento program nejprve data normalizuje, tzn. napsaný znak roztáhne do jednotné velikosti. Všechny nakreslené znaky tak mají stejnou velikost.

Program tedy po provedené normalizaci převede reprezentaci písmena na jednotlivé body. Při trénování modelu se ještě objevil problém s nedostatkem bodů u některých písmen (například I). Při nedostatku vstupních dat pro trénování skrytých Markovových modelů nelze modely spolehlivě natrénovat. Proto jsou data při jejich nedostatku uměle přidána tak, aby odpovídala nakreslenému písmenu.

Po provedených úpravách pak program z dat extrahuje příznaky (byly použity dva až osm příznaků, viz. kapitola 4.

Pro trénování skrytých Markovových modelů jsem použil [5], který pracuje s toolkitem HTK. Zde je možné nastavit počet stavů skrytých Markovových modelů a počet přechodů mezi stavy. Výsledkem trénování jsou soubory, které reprezentují jednotlivá písmena a obsahují konfiguraci odpovídajícího HMM modelu.

Konzolový rozpoznávač Tento program byl vytvořen pro statistické vyhodnocení úspěšnosti rozpoznávání pomocí natrénovaných modelů. Jako vstup vyžaduje adresář s testova-

cími daty a textové soubory s natrénovanými modely. Po spuštění vytvoří klasifikátor, naplní jej natrénovanými modely a pak na něm otestuje všechna testovací data. Na výstup pak vypíše pravděpodobnostní matici úspěšnosti rozpoznávání (*confusion matrix*) jednotlivých znaků a celkovou úspěšnost rozpoznávání.

GUI rozpoznávač Rozpoznávač s grafickým uživatelským rozhraním byl vytvořen pro demonstraci možného využití vytvořeného systému. Po spuštění očekává nakreslení znaku uživatelem. Po dokončení kreslení pak tento znak vyhodnotí a napíše písmeno, které rozpoznal. Ještě ale vypíše 4 další nejpravděpodobnější písmena, která mohou odpovídat nakreslenému znaku. Napsaný text je pak možné odeslat jiné aplikaci, která je tento text schopna dále zpracovat. Tento program by mohl být náhradou klávesnice. Je ovšem nutná přítomnost zařízení (touchscreenu), které je schopno přesně převést tah ruky do aplikace.

Program využívá pro rozpoznávání modely natrénované pomocí dvou příznaků (souřadnic tahu x a y). Byly provedeny experimenty i s více příznaky, ale úspěšnost nijak výrazně nepřesáhla modely s dvěma příznaky.

Aby mohl být program plnohodnotně využíván, je třeba ještě implementovat vložení libovolného znaku, symbolu či diakritiky do textu.

Kapitola 4

Vyhodnocení

Úspěšnost rozpoznávání ovlivňuje mnoho faktorů. Tato kapitola se zabývá některými, které ovlivňovaly úspěšnot rozpoznávání ve vyvýjeném programu. V první části se zabývá datovou sadou používanou pro trénování (jejími úpravami, velikostí). V další části se pak věnuje různým nastavením při trénování modelů. Nakonec jsou diskutována možná vylepšení, které by mohly vést k lepším výsledkům při rozpoznávání.

4.1 Datová sada

Při vytváření datové sady byl brán ohled na to, že písmena může každý člověk psát jinak. Písmena proto byla kreslena různými způsoby tak, aby vytvořená sada představovala co nejlépe obecný tvar jednotlivých písmen. Jelikož jsou ale některá písmena napsána jinak (například obráceným směrem tahu - pozpátku), mohou se modely natrénovat špatně a úspěšnost tím klesá. Pokud by ale datová sada byla vytvořena uživatelem na základě jeho stylu psaní, úspěšnost rozpoznávání by se pro daného uživatele zvýšila. Pokud by ale rozpoznávač pak používal někdo s jiným stylem psaní, úspěšnost klasifikátoru by byla horší. Pro zvýšení úspěšnosti bylo nutné nakreslenou sadu vizuálně projít a ručně smazat ta písmena, která neodpovídala tvarem své hodnotě, či byla jinak znehodnocena (například přeškrtnuté písmeno, čárka navíc, aj.).

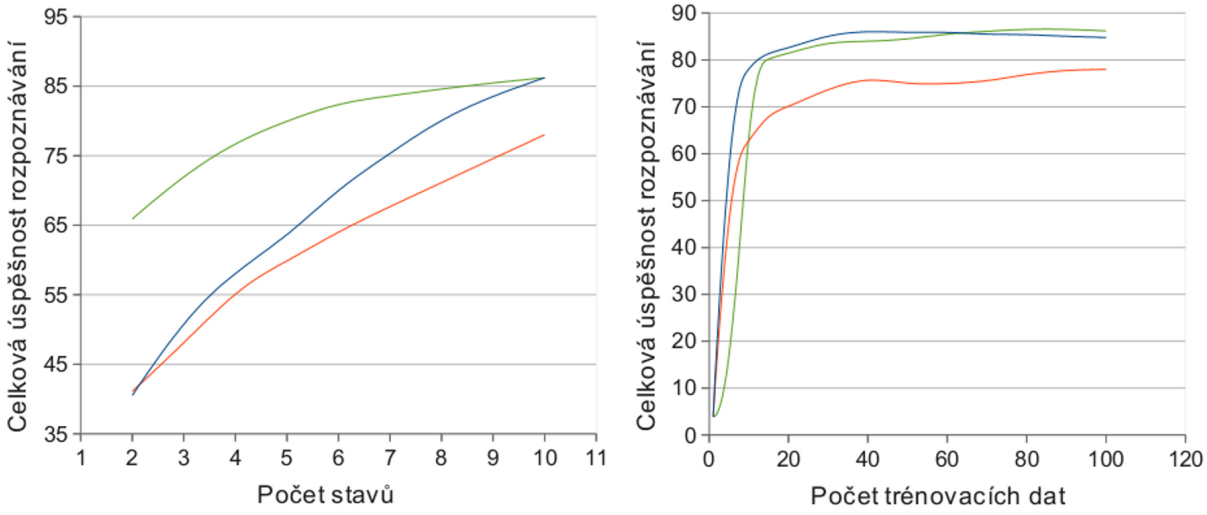
Úspěšnost výrazně ovlivňuje velikost trénovací sady. Při dostatečném počtu trénovacích dat, model celkem přesně reprezentuje písmeno, které má představovat. Ovšem je třeba si dát pozor na „přetrénování“ modelu. Při velkém počtu trénovacích dat model konverguje k malým odchylkám ve vstupních datech a tak ztrácí obecnost popisu jedotlivých písmen. V grafu vpravo na obrázku 4.1 je zobrazena úspěšnost rozpoznávání pro různé velikosti trénovacích dat a pro různé počty příznaků (podrobněji viz. níže).

4.2 Nastavení modelů

Nastavení parametrů modelů je jednou z klíčových věcí pro úspěšné rozpoznávání. V [5] bylo možné nastavit počet stavů a počet přechodů mezi jednotlivými stavy. Počet přechodů byl zvolen na dva. Při větším počtu přechodů byla úspěšnost modelů velice špatná.

Při trénování modelů jsem použil tři sady příznaků:

1. Trénování se dvěma příznaky (souřadnice bodů tahu x a y). V grafech zobrazeno modře.



Obrázek 4.1: Grafy závislosti počtu stavů modelů a počtu trénovacích dat na ceklové úspěšnosti

2. Trénování se třemi příznaky (souřadnice bodů tahu x a y, úhel směru tahu α). V grafech zobrazeno červeně.

3. Trénování s osmi příznaky (souřadnice bodů tahu x a y, úhel směru tahu α, rychlost psaní ve směru os x a y dx, dy, zrychlení při psaní ve směru os x a y d^2x, d^2y, vzdálenost bodů jednoho tahu $d = \sqrt{(x_t - x_{t-1})^2 + (y_t - y_{t-1})^2}$). V grafech zobrazeno zeleně.

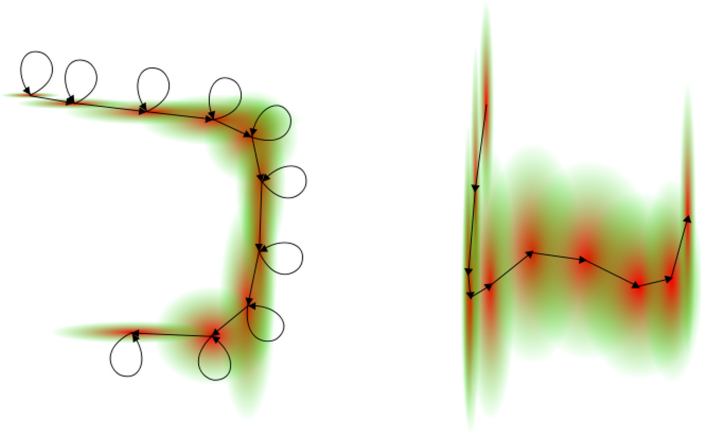
Na obrázku 4.1 jsou znázorněny úspěšnosti rozpoznávání při nastavení počtů stavů a počtů trénovacích dat. Tři křivky představují modely natrénované různými příznaky.

Jak je vidět v grafu vlevo na obrázku 4.1, nejlepší výsledky dávaly modely při deseti stavech. Více stavů nebylo možné v [5] nastavit. Také je možné z grafu vyčíst, že přidáním informace udávající směr tahu se úspěšnost snížila. Proti tomu ale pomohlo přidání dalších příznaků. Úspěšnost takového modelu dosahovala zajímavých hodnot již při malém počtu stavů. Ale při deseti stavech byla srovnatená s modelem pouze se dvěma příznaky.

V grafu napravo na obr. 4.1 je zobrazena úspěšnost rozpoznávání v závislosti na počtu dat použitých pro trénování. I zde je vidět zřetelně nižší úspěšnost modelů, které byly natrénovány pomocí tří příznaků. Je také možné vidět podobný trend úspěšnosti modelů se dvěma příznaky a osmi příznaky. Tyto dvě sady modelů dosahovaly podobných úspěšností. Modely natrénované na dvou příznacích dosahovaly nejlepší úspěšnosti při 40 trénovacích datech (86.2%). Modely natrénované na osmi příznacích měli nejlepší úspěšnost při 80 trénovacích datech (86.6%).

Na obrázku 4.2 je ukázka modelů písmen „J“ a „N“. Tyto modely byly natrénovány pomocí dvou příznaků (souřadnic). Jak lze z obrázku zjistit, model písmena „J“ je velice podobný jeho tvaru (celkem přesně kopíruje tvar písmena). Model písmena „N“ již tak věrohodně tvaru písmena neodpovídá.

Tabulka 4.1 přestavuje část tabulky úspěšnoti rozpoznávání (*confusion matrix*). Ukazuje úspěšnost rozpoznávání jednotlivých písmen při trénování modelů pomocí dvou příznaků (souřadnic) (celá je zobrazena v příloze B spolu s tabulkou úspěšnosti při trénovávní osmi příznaky). V tabulce 4.1 je vidět, že rozpoznávač má velké problémy při rozpoznávání písmena T. V naprosté většině případů, kdy je T rozpoznáno špatně je rozpoznáno jako P.



Obrázek 4.2: Výslené modely písmen „J“ (vlevo) a „N“ (vpravo). Obrázky reprezentují modely natrénované pomocí dou příznaků (souřadnice x a y)

Tento problém nastává v případě, že jsou modely nepřesně natrénované a jsou si podobné, jak je znázorněno na obr. 4.4. Na obr 4.3 je možné vidět i další případy chybného rozpoznání. Řešením by v tomto případě mohlo být zvýšení počtu stavů HMM.

Při testování osmi příznaků měl rozpoznávač podobné problémy s rozpoznáním písmena „B“ od „R“ (tabulka úspěšnosti rozpoznávání v příloze B). Pro toto písmeno byla úspěšnost ještě horší než pro „T“ natrénované na dvou příznacích, proto jsem pro rozpoznávač s grafickým uživatelským rozhraním zvolil modely natrénované na dvou příznacích.

4.3 Možná vylepšení

Pro zvýšení úspěšnosti rozpoznávání by bylo možné trénovat modely ještě s dalšími získanými příznaky. I když se úspěšnost po přidání jednoho příznaku zhoršila, bylo vidět, že po přidání dalších příznaků se již úspěšnost vylepšila a je možné, že přidáním dalších by úspěšnost ještě vzrostla.

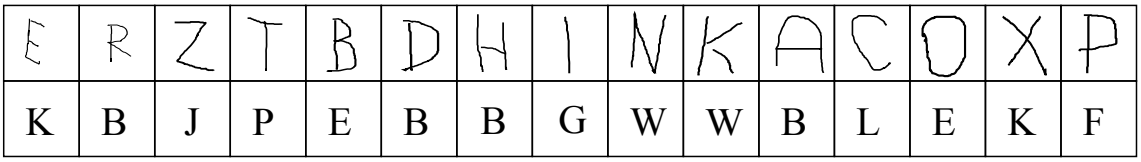
Úspěšnost by také zvýšilo přidání stavů do trénovaných modelů (tzn. používat trénovací prostředek, kde lze nastavit větší počet stavů). Toto nastavení by ale mohlo také vést ke zhoršení již dobře natrénovaných modelů. Je třeba také poznamenat, že počet stavů nelze stále zvyšovat. Již pro deset stavů bylo nutné pro některá písmena doplňovat počet bodů, aby bylo možné modely natrénovat (dat musí být nejméně tolik, kolik je stavů modelu). Doplňováním bodů pro modely s velkým počtem stavů by mohlo napsané znaky „znehodnotit“. Také je třeba brát v potaz, že počítání pravděpodobnosti písmena pro daný model je výpočetně náročné. Proto by byly modely s velkým počtem stavů pro on-line rozpoznávání nevhodné.

Zvýšení úspěšnosti by se mohlo dosáhnout také přidáním modelů pro každou napsanou variantu jednoho písmene, tzn. pokud je v trénovací sadě více variant pro jedno písmeno (např. tahy písmena jsou v jiném pořadí - v datové sadě pro tento rozpoznávač to tak bylo), měl by být jeden model pro každou takovou variantu.

Také je možné rozšířit klasifikátor o slovník. Klasifikátor by pak každou souvislou sekvenci rozpoznaných písmen mohl testovat na nejpravděpodobnějších možných slovech a ta nabízet uživateli.

Rozpoznávané znaky \ Rozpoznané znaky	O	P	Q	R	S	T	U	V	W
O	98.7	0	0	0	0	0	0	1.33	0
P	0	98.7	0	0	0	0	0	0	0
Q	0	0	85.3	1.33	0	0	0	0	6.67
R	0	0	0	80	0	1.33	0	0	5.33
S	0	0	0	0	96	0	0	0	0
T	6.67	41.3	0	0	0	42.7	0	0	0
U	2.67	0	0	0	0	0	72	25.3	0
V	0	0	0	0	0	0	5.33	93.3	0
W	0	0	0	0	0	0	0	0	98.7

Tabulka 4.1: Confusion matrix

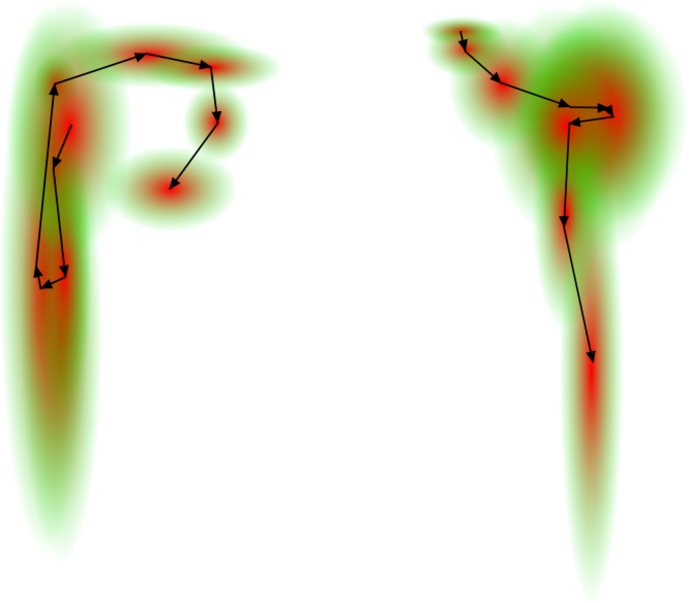


Obrázek 4.3: Ukázka špatně rozpoznaných písmen

Úspěšnost je také možné zvýšit tzv. *on-line trénováním*, kdy se špatně rozpoznaným písmenem přetrénuje stávající model.

Pro větší „uživatelské pohodlí“ by se do sady písmen mohla zařadit i malá písmena, psací písmena, písmena s diakritikou. Je možné také rozpoznávač rozšířit o možnost rozpoznávání souvislého textu. Tato možnost ale s sebou nese další velká úskalí v podobě hledání hranic jednoho znaku, rozhodování o velikosti písmene, aj. Do rozpoznávače by také bylo vhodné umístit:

- Možnost vložení znaků, které se příliš často nepoužívají (diakritika, čárka, tečka, a další).
- Možnost vložení znaků, které nelze rozpoznat (možnost psaní pomocí softwarové klávesnice).
- Doplnění tlačítek, jako je enter, escape, control, alt, a podobné.



Obrázek 4.4: Model písmena P (vlevo) a model písmena T (vpravo) (trénované pomocí dvou příznaků). Tyto dva modely jsou si velice podobné, proto dochází k časté záměně při rozpoznávání.

Kapitola 5

Závěr

Cílem této práce bylo seznámit se blíže s rozpoznáváním znaků, zvláště pak s on-line rozpoznáváním a vytvořit systém, který bude schopen rozpoznávat rukou napsaný text. Vytvořený rozpoznávač je schopen klasifikovat velká písmena anglické abecedy s úspěšností okolo 85%.

Teoretický popis problematiky rozpoznávání je popsán v kapitole 2. Je zde popsána problematika získávání informací pro rozpoznávání z obrazových dat. Dále jsou zde popsány dvě metody používané při rozpoznávání: neuronové sítě a Markovovy modely, z nichž skryté Markovovy modely byly použity pro klasifikaci ve vytvořeném programu.

V kapitole 3 je popsán vytvořený systém a jeho návrh. Systém se skládá z několika programů. Referenční aplikací je program s jednoduchým uživatelským rozhraním, který rozpoznává psané znaky a je schopen odeslat text do jiné aplikace (obrázek programu v příloze A na obr. A.1). Všechny programy jsou k dispozici na přiloženém CD.

Vyhodnocení úspěšnosti a experimety s nastavením modelu popisuje kapitola 4. Úspěšnost rozpoznávání je zobrazena v matici úspěšnosti rozpoznávání (confusion matrix), která je zobrazena v příloze B. V kapitole 4 jsou rovněž navrhnuta vylepšení, která by úspěšnost rozpoznávání zvýšila.

Literatura

[1] Bengio, Y.; Le Cun, Y.: Word normalization for on-line handwritten word recognition. In *International Conference on Pattern Recognition*, Citeseer, 1994, s. 409–409.

[2] Bishop, C. M.: *Pattern Recognition and Machine Learning*. Springer Science + Business Media, LLC, 2006, iSBN 0-387-31073-8.

[3] Eddy, S.: Profile hidden Markov models. *Bioinformatics*, ročník 14, č. 9, 1998: str. 755.

[4] Hu, J.; Brown, M.; Turin, W.: HMM based on-line handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, ročník 18, č. 10, 1996: s. 1039–1045.

[5] pcjuranek: The web page serves as the gate to simple training tool for HMM based on HTK. http://pcjuranek.fit.vutbr.cz/HTK/, 2010, [Online; accessed 22-May-2010].

[6] Rabiner, L.: A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, ročník 77, č. 2, 1989: s. 257–286.

[7] Schenk, J.; Kaiser, M.; Rigoll, G.: Selecting Features in On-Line Handwritten Whiteboard Note Recognition: SFS or SFFS? *Document Analysis and Recognition, International Conference on*, ročník 0, 2009: s. 1251–1254, doi:http://doi.ieeecomputersociety.org/10.1109/ICDAR.2009.130.

[8] Schenk, J.; Lenz, J.; Rigoll, G.: Line-Members–a Novel Feature in On-Line Whiteboard Note Recognition. In *Proceedings of the International Conference on Frontiers in Handwriting Recognition*, 2008, s. 469–474.

[9] Sonka, M.; Hlavac, V.; Boyle, R.: *Image Processing, Analysis, and Machine Vision*. Brooks/Cole Publishing Company, 1999, iSBN 0-534-95393-X.

[10] Wikipedia: Optical character reconition — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/wiki/Optical_character_recognition, 2010, [Online; accessed 16-March-2010].

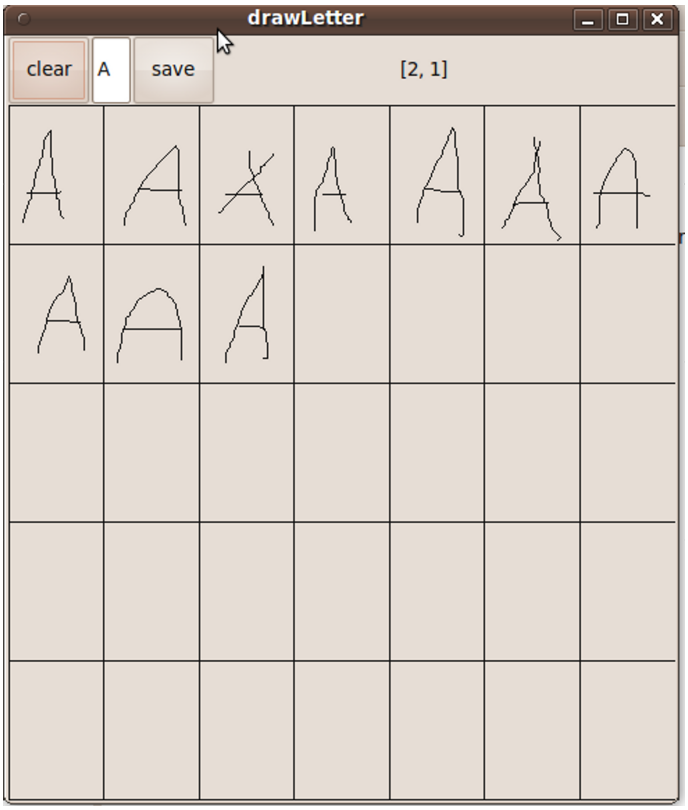
[11] Young, S.; Evermann, G.; Kershaw, D.; aj.: The HTK book. 2000.

Dodatek A

Výsledné aplikace

Na obrázccích A.1 a A.2 jsou zobrazeny screenshoty výsledného rozpoznávače a programu pro získání datové sady.

Obrázek A.1: Rozpoznávač



Obrázek A.2: Program pro získávání datové sady od uživatele

Dodatek B

Confusion matrix

Na obrázku B.1 je znázorněna úspěšnost rozpoznávání jednotlivých znaků pomocí modelů natrénovaných pomocí dvou příznaků. Na obrázku B.1 je znázorněna úspěšnost rozpoznávání jednotlivých znaků pomocí modelů natrénovaných pomocí osmi příznaků.

Rozpoznané znaky

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	96	0	0	1.33	0	0	0	0	0	0	0	0	1.33	0	0	0	0	1.33	0	0	0	0	0	0	0	0
B	0	73.3	0	9.33	0	0	0	0	0	0	0	0	0	0	0	0	0	10.7	2.67	1.33	0	0	0	1.33	1.33	0
C	0	0	96	0	0	0	2.67	0	0	0	0	1.33	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D	0	17.3	0	70.7	0	0	0	0	0	1.33	0	0	4	0	0	2.67	0	1.33	0	0	0	0	0	0	2.67	0
E	0	0	0	0	94.7	0	0	0	0	0	4	0	0	0	1.33	0	0	0	0	0	0	0	0	0	0	0
F	1.33	0	0	2.67	0	80	0	0	0	0	0	0	5.33	0	0	5.33	0	5.33	0	0	0	0	0	0	0	0
G	0	0	0	0	0	0	98.7	0	0	0	0	0	0	0	1.33	0	0	0	0	0	0	0	0	0	0	0
H	0	0	0	0	0	0	0	70.7	0	0	0	0	1.33	13.3	1.33	1.33	9.33	1.33	0	0	0	0	1.33	0	0	0
I	0	0	0	0	0	0	0	0	98.7	0	0	0	0	0	0	0	0	0	1.33	0	0	0	0	0	0	0
J	0	0	0	0	0	0	0	0	0	98.7	0	0	0	0	0	0	0	0	0	0	0	0	0	1.33	0	0
K	0	0	0	0	0	0	6.67	0	0	0	82.7	0	0	5.33	1.33	0	0	0	0	0	0	2.67	0	0	0	1.33
L	0	0	0	0	0	0	0	0	0	0	0	84	0	2.67	0	0	0	0	0	0	1.33	8	2.67	0	1.33	0
M	4	0	0	0	1.33	0	0	4	0	0	0	0	88	0	0	2.67	0	0	0	0	0	0	0	0	0	0
N	0	0	0	0	0	0	0	0	0	0	0	0	8	69.3	0	0	0	0	0	0	20	0	2.67	0	0	0
O	0	0	0	0	0	0	0	0	0	0	0	0	0	0	98.7	0	0	0	0	0	0	1.33	0	0	0	0
P	0	0	0	0	0	0	0	0	0	0	0	0	1.33	0	0	98.7	0	0	0	0	0	0	0	0	0	0
Q	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	85.3	1.33	0	0	0	0	6.67	0	2.67	0
R	0	1.33	0	5.33	0	2.67	0	0	0	0	0	0	0	1.33	0	0	0	80	0	1.33	0	0	5.33	0	2.67	0
S	0	0	2.67	1.33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	96	0	0	0	0	0	0	0
T	0	0	0	0	2.67	0	0	0	0	0	0	0	0	0	6.67	41.3	0	0	0	42.7	0	0	0	6.67	0	0
U	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2.67	0	0	0	0	0	72	25.3	0	0	0	0
V	0	0	0	0	0	0	0	0	0	0	0	0	0	1.33	0	0	0	0	0	0	5.33	93.3	0	0	0	0
W	0	0	0	0	0	0	0	0	0	0	0	0	0	1.33	0	0	0	0	0	0	0	0	98.7	0	0	0
X	0	0	1.33	2.67	0	0	0	0	0	1.33	0	0	0	0	0	0	0	1.33	1.33	0	0	0	0	92	0	0
Y	0	0	0	2.67	0	0	0	2.67	0	0	0	0	0	0	0	0	0	0	1.33	1.33	0	0	2.67	0	89.3	0
Z	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5.33	0	0	0	0	2.67	92

Rozpoznávané znaky

Obrázek B.1: Confusion matrix pro dva příznaky (souřadnice bodů tahu x a y).

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	94,7	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1,33	0	0	0
B	0	37,3	0	0	0	0	0	0	0	2,67	0	0	0	0	0	1,33	1,33	52	0	4	0	0	0	0	1,33	0
C	0	0	94,7	0	0	1,33	0	0	1,33	0	0	1,33	0	0	0	0	0	0	0	0	0	0	0	0	1,33	0
D	0	0	0	89,3	0	0	0	0	0	2,67	0	0	0	0	0	4	0	1,33	0	1,33	0	0	0	0	1,33	0
E	0	0	0	0	97,3	0	0	0	0	0	0	0	0	0	0	0	1,33	1,33	0	0	0	0	0	0	0	0
F	1,33	0	0	0	0	94,7	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0
G	1,33	0	0	0	1,33	0	96	0	0	0	1,33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
H	0	0	0	0	1,33	0	0	82,7	0	0	0	0	1,33	4	0	0	2,67	1,33	0	2,67	0	0	0	0	4	0
I	0	0	10,7	0	0	0	0	0	84	0	0	1,33	0	0	0	0	0	0	0	0	0	0	0	2,67	1,33	0
J	0	0	0	0	0	1,33	1,33	1,33	0	93,3	0	0	0	0	0	0	0	0	0	1,33	0	0	0	0	0	1,33
K	0	0	0	0	0	0	0	0	0	0	88	1,33	0	2,67	0	0	2,67	2,67	0	1,33	0	0	0	0	0	1,33
L	0	0	0	0	0	0	0	1,33	0	0	0	97,3	0	0	0	0	0	0	0	1,33	0	0	0	0	0	0
M	0	0	0	1,33	0	0	0	0	0	0	0	0	92	2,67	1,33	0	0	0	0	1,33	0	0	0	0	1,33	0
N	0	0	0	0	0	0	0	5,33	0	0	0	0	0	80	0	0	0	0	0	0	2,67	2,67	6,67	1,33	1,33	0
O	0	0	0	0	0	0	0	0	0	0	1,33	0	0	0	88	0	1,33	0	0	0	8	0	0	0	1,33	0
P	0	2,67	0	29,3	0	0	0	0	0	1,33	0	0	0	0	0	62,7	0	2,67	0	1,33	0	0	0	0	0	0
Q	0	0	0	0	0	0	0	2,67	0	0	0	0	0	0	0	0	76	0	0	1,33	0	0	5,33	1,33	12	1,33
R	0	0	0	1,33	0	0	0	0	0	0	0	0	0	0	0	0	0	98,7	0	0	0	0	0	0	0	0
S	0	0	0	0	0	1,33	0	0	0	0	0	0	0	0	0	0	0	0	97,3	1,33	0	0	0	0	0	0
T	0	0	0	1,33	1,33	0	0	0	0	1,33	0	0	0	0	0	2,67	0	2,67	0	81,3	0	0	0	0	2,67	6,67
U	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	2,67	76	16	1,33	0	0	0
V	0	0	0	0	0	0	0	0	0	0	1,33	0	0	0	0	0	0	0	0	0	1,33	93,3	2,67	0	1,33	0
W	0	0	0	0	0	0	0	1,33	0	0	0	0	0	4	0	0	1,33	0	0	0	0	5,33	88	0	0	0
X	0	0	0	0	0	0	0	0	5,33	0	4	0	0	0	0	0	0	0	0	0	0	0	0	88	2,67	0
Y	0	0	0	0	0	1,33	0	0	0	0	0	0	0	0	0	2,67	0	0	0	0	1,33	2,67	1,33	1,33	88	1,33
Z	0	0	0	0	0	0	0	0	0	2,67	0	0	0	0	0	0	1,33	1,33	0	1,33	0	0	0	0	0	93,3

Obrázek B.2: Confusion matrix pro osm příznaků (souřadnice bodů tahu x a y, úhel směru tahu, rychlost psaní ve směru os x a y, zrychlení při psaní ve směru os x a y, vzdálenost bodů jednoho tahu).