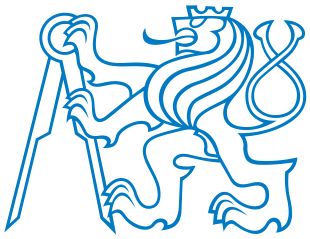
bakalářská práce

Implementace rozpoznávače řeči na bázi TANDEM architektury

Aleš Brich



květen 2014

Doc. Ing. Petr Pollák, CSc.

České vysoké učení technické v Praze
Fakulta elektrotechnická, Katedra kybernetiky

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Aleš B r i c h

Studijní program: Kybernetika a robotika (bakalářský)

Obor: Robotika

Název tématu: Implementace rozpoznávače řeči na bázi TANDEM architektury

Pokyny pro vypracování:

1. Seznamte se s problematikou rozpoznávání řeči na bázi HMM/ANN s užším zaměřením na implementaci rozpoznávače na bázi TANDEM architektury kombinující standardní příznaky na bázi kepstrální analýzy a příznaky na bázi aposteriorních pravděpodobností hlásek odhadnuté pomocí MLP.
2. Realizujte rozpoznávač s malým slovníkem pomocí volně dostupných nástrojů HTK Toolkitu a nalezněte optimální kombinaci příznaků standardních a MLP příznaků.
3. Úspěšnost rozpoznávání vyhodnoťte na experimentech realizovaných na řečové databázi SPEECON.

Seznam odborné literatury:

[1] Lal, P.; King, S.: Cross-Lingual Automatic Speech Recognition Using Tandem Features. In: Audio, Speech, and Language Processing, IEEE Transactions on , vol. 21, no.12, pp.2506-2515, Dec. 2013.
[2] Psutka, J.; Müller, L.; Matoušek, J.; Radová, V.: Mluvíme s počítačem česky.Academia, 2006.
[3] Uhlíř, J. a kol.: Technologie hlasových komunikací. Nakladatelství ČVUT, Praha, 2007.
[4] Huang, X.; Acero, A.; Hon, H.V.: Spoken Language Processing. Prentice Hall, 2001.
[5] Fousek, P.: Extraction of Features for Automatic Recognition of Speech Based on Spectral Dynamics. PhD Thesis, ČVUT FEL, Praha, 2007.
[6] Young, S. et al.: The HTK Book (for HTK version 3.4). Cambridge University Engineering Department, 2009.

Vedoucí bakalářské práce: doc. Ing. Petr Pollák, CSc.

Platnost zadání: do konce letního semestru 2014/2015

L.S.

doc. Dr. Ing. Jan Kybic
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 10. 1. 2014

Poděkování

Rád bych vyjádřil vděk svému vedoucímu Doc. Ing. Petru Pollákovi, CSc. za výborné vedení a cenné rady s danou problematikou. Dále bych rád poděkoval Ing. Petru Mizerovi a Ing. Michalu Borskému za konstruktivní připomínky a technickou podporu.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne _____ _____

podpis

Abstrakt

Tato práce se zabývá implementací jednoduchého rozpoznávače řeči na bázi TANDEM architektury pomocí volně dostupných, široce používaných nástrojů HTK. Cílem je zmapovat, jaké typy příznakových vektorů TANDEM architektury přinášejí zlepšení úspěšnosti klasifikace oproti standartním příznakovým vektorům (například melovským kepstrálním koeficientům). Pro splnění tohoto cíle je nutné nejprve vypočíst mel-spektra akustických signálů. Z nich lze vypočítat jak melovské kepstrální koeficienty, tak TRAPs příznaky. TRAPs příznaky se dále mapují pomocí umělé neuronové sítě na pravděpodobnosti výskytu jednotlivých fonémů. Tyto pravděpodobnosti se po zlogaritmování a KLT spojí z melovskými kepstrálním koeficienty v příznakový vektor TANDEM architektury. Těmito příznaky se následně přetrénovávají skryté Markovovy modely. Nejvyšší úspěšnosti bylo dosaženo při použití příznakového vektoru KLT melovských kepstrálních koeficientů s delta a delta-delta příznaky. Méně úspěšný byl příznakový vektor složený z KLT jednotlivých derivací kepstrálních koeficientů a KLT logaritmovaných aposteriorních pravděpodobností neuronové sítě s průměrnou úspěšností klasifikace 82 %.

Klíčová slova

rozpoznávání řeči; TANDEM architektura; skryté Markovovy modely; HMM; HTK

Abstract

This paper deals with implementation of simple speech recognizer based on TANDEM architecture by using free, widely spread tool HTK. The main goal is to explore which types of TANDEM architecture feature vectors improve the classification accuracy in comparation with standard features (e.g. mel cepstral coeficient). To reach this objective, it is first necessary to calculate the mel-spectrum of acoustic signals. This spectrum can be used to calculate mel kepstral coefficients and TRAPs features. TRAPs features are transformed by artificial neural network to the posterior probabilites of the individual phonemes. These probabilities are modified by using logarithm and KLT transformation. Then these modified probabilites are combined with mel cepstral coefficients into the feature vector of TANDEM architecture. The features are used to re-estimate Hidden Markov Models. The highest accuracy was achieved for feature vector defined by KLT mel cepstral coefficients with delta and delta-delta features. KLT of individual cepstral coefficient derivation and KLT posteriors of artificial neural network feature vector was less successful. The obtained average accuracy of classification for this vector was 82 %.

Keywords

speech recognition; TANDEM features; hidden Markov models; HMM; HTK

Obsah

Seznam obrázků

Použité zkratky

MFCC	Mel-freqvency Cepstral Coeficient (Melovské kepstrální koeficienty)
TRAPs	Temporal Patterns (Časové příznaky)
FFT	Fast Fourier Transform (Rychlá Fourierova transformace)
DFT	Discrete Fourier Transform (Diskrétní Fourierova transformace)
IDFT	Inverse Discrete Fourier Transform (Inverzní diskrétní Fourierova transformace)
DCT	Discrete Cosine Transform (Diskrétní Kosinova transformace)
PCA	Principal Component Analysis (Metoda hlavních komponent)
KLT	Karl-Loev Transform (Karhunenova-Loeveova transformace)
ANN	Artificial Neural Network (Umělá neuronová síť)
MLP	Multilayer Perceptron (Vícevrstvý perceptron)
DTW	Dynamic Time Warping (Dynamické borcení časové osy)
HMM	Hidden Markov Model (Skrytý Markovův model)
HTK	Hidden Markov Model Toolkit
SGE	Sun Grid Engine
ACC	Accuracy
WER	Word Error Rate
WERR	Word Error Rate Reduction

1 Úvod

S mluvenou řečí se setkáváme po celý život. Jedná se o nejpřirozenější způsob přenosu informace mezi lidmi. Je pro nás natolik intuitivní, že dokážeme bez problémů souběžně provádět i jinou činnost. Kromě toho k řeči není potřeba žádných pomůcek, nebo speciálních znalostí.

Různá výpočetní zařízení dnes pracují nepovšimnutě v naší nejtěsnější blízkosti. Problém nastává v okamžiku, kdy se zařízením potřebujeme interagovat. I pro nejbanálnější operace musíme takovému zařízení věnovat nemalou pozornost. U jednoduchých operací se tak přímo nabízí možnost použít hlasového rozhraní.

Rozpoznávání řeči, kterou každý člověk provádí automaticky a prakticky bez námahy, je úloha pro stroje spjatá s řešením velmi náročných úloh. V průběhu času vzniklo mnoho nástrojů jak pro výpočet příznaků, tak i jejich zpracování. Standardně používaným nástrojem je Fourierova transformace a její diskrétní varianta, neboť většina příznaků používaných v dnešních technologiích rozpoznávání řeči vychází z frekvenční charakteristiky akustického signálu. Kromě DFT je možno spektrum odhadnout pomocí lineární prediktivní analýzy (LPC). Další zpracování může vést ke statickým příznakům. Ty je možné doplnit o dynamické příznaky, které reprezentují kontext a tím zvyšují robustnost vektoru parametrů vůči šumu. Příznaky počítající rovnou s kontextem lze případně vypočítat použitím parametrizace uvažující delší časový úsek signálu.

Vypočtené příznaky jsou pak použity jako data pro klasifikaci. Klasifikátory se často spojují do řetězců, kdy výstup z jednoho klasifikátoru slouží, po případné úpravě vhodnou transformací, jako vstupní příznaky dalšího klasifikátoru. Existuje mnoho typů klasifikátorů. Při rozpoznávání řeči se jako velmi vhodné ukázaly klasifikátory na bázi umělých neuronových sítí a skrytých Markovových modelů.

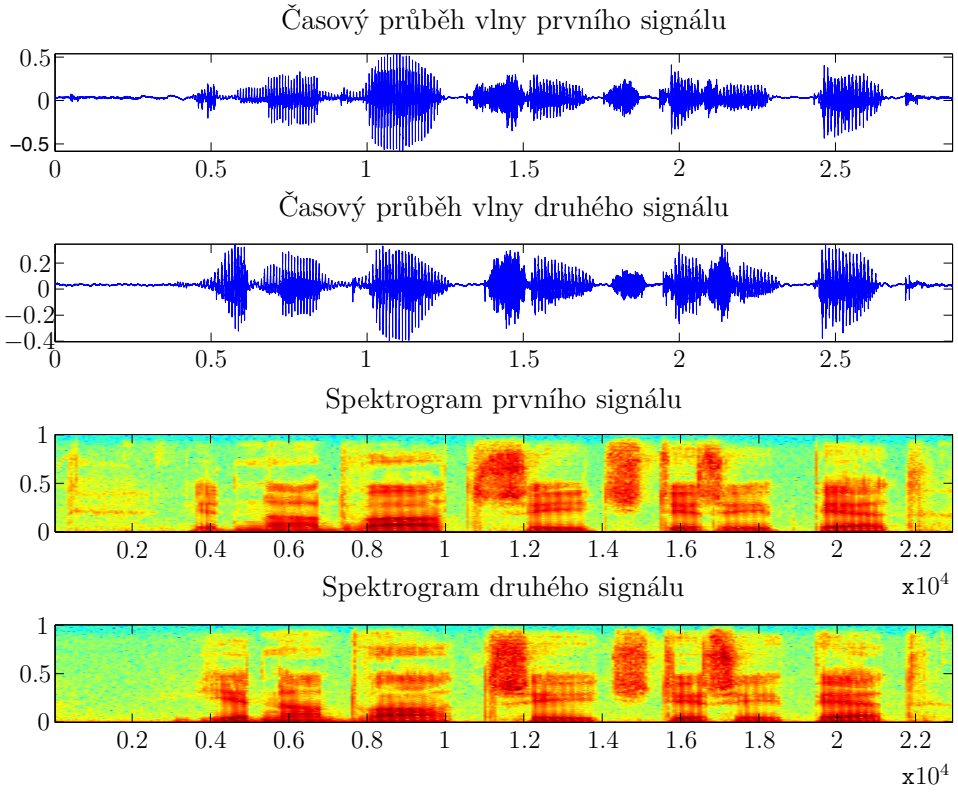
Cílem práce je popsat a implementovat rozpoznávač řeči na principu skrytých Markovových modelů a TANDEM architektury. Příznakové vektory se v případě TANDEM architektury skládají z transformovaných aposteriorních pravděpodobností a akustických příznaků. Implementace bude probíhat hlavně s použitím balíčku nástrojů HTK a skriptovacího jazyka Perl. Výsledkem by měl být rozpoznávač jednoduchých příkazů.

Jedna kapitola bakalářské práce se věnuje výpočtu řečových parametrů pro účely rozpoznávání. Pojednává o metodách výpočtu mel-spektra, MFCC a TRAPs příznaků používaných pro výpočet aposteriorních pravděpodobností. Popisuje metody výpočtu

aposteriorních pravděpodobností pomocí ANN, jejich transformaci pro použití v příznakových vektorech TANDEM architektury a konečné spojení akustických příznaků s transformovanými aposteriorními pravděpodobnostmi do vektoru příznaků TANDEM architektury. Další kapitola se zabývá metodami použití a implementací HMM pro rozpoznávání na úrovni hlásek s použitím příznakových vektorů popsaných v předcházející kapitole. Následuje kapitola popisující vlastní implementaci rozpoznávače nástroji HTK, pfile_utils a jinými. V experimentální části práce je ukázána úspěšnost různých typů příznakových vektorů TANDEM architektury. Celkové zhodnocení je uvedeno v závěrečné kapitole.

2 Parametrizace řečového signálu

Řeč je akustický signál a lze jej reprezentovat mnoha způsoby. V časové oblasti pomocí časového průběhu vlny, ve frekvenční oblasti pak spektrogramem. Problém při rozpoznávání řeči je v tom, že každý signál je originální. Rozdíly v amplitudě i spektru se projevují nejen při různých mluvčích, ale i v případě jediného mluvčího. Obrázek 2.0.1 ukazuje dva signály se stejným obsahem namluvené jedním mluvčím. Byť byl kladen důraz na stejné časování, rozdíly tu jsou, a to i v časové oblasti a jak je vidět, tak i ve složce frekvencí.



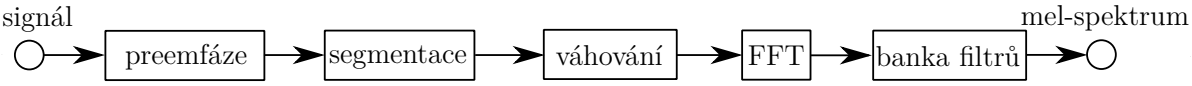
Obrázek 2.0.1 Dvě promluvy slov "jedna dva tři čtyři pět"v časové a frekvenční oblasti.

Hlavním účelem této práce je realizace rozpoznávače řeči. Pro tento úkol je důležité nalézt takovou reprezentaci řeči, která je vhodná k rozpoznávání na úrovni HMM. Tato reprezentace by měla obsahovat co nejvíce užitečné informace, měla by minimalizovat nepodstatné informace a variabilitu příznaků. Takovéto parametry, které se následně používají pro rozpoznávání, se nazývají řečové příznaky (také parametry). Tato úloha se

nazývá extrakce příznaků, front-end zpracování, nebo parametrizace řeči. Jejím úkolem je najít takové parametry řečového signálu, které se pro signály pro jeden foném, resp. hlásku, příliš neliší a zároveň se co nejvíce liší mezi různými fonémy, či hláskami, tj. základními subslovními akustickými elementy každé promluvy. Foném je definován jako nejmenší lingvistická jednotka schopná rozlišovat významové jednotky. Naproti tomu hláska je akustickou realizací fonému. Počty hlásek se v různých jazycích liší od 25 do 61. Čeština obsahuje 40 fonémů. [1]. S tímto počtem hlásek se bude dále pracovat v další části této práce. Existuje mnoho různých typů parametrizací, které jsou optimální pro zpracování za různých podmínek. V práci jsou používány dva typy příznaků vycházející z výkonového mel-spektra. Jedná se o mel-frekvenční kepstrální koeficienty a TRAPs parametrizaci, tedy parametrizaci na bázi časových trajektorií. Výpočet těchto typů příznaků je popsán v dalších kapitolách.

2.1 Výpočet mel-spektra

Před výpočtem samotného mel-spektra se provádí preemfáze, segmentace signálu, váhování okénkem a výpočet Fourierovy transformace. Tyto kroky, zobrazené ve schématu 2.1.1, jsou popsány v následujících částech.



Obrázek 2.1.1 Blokové schéma výpočtu mel-spektra.

2.1.1 Preemfáze

Řečové ústrojí způsobuje u akustického signálu pokles amplitud spektrálních složek ve vyšších frekvencích. Stejně tak lidský sluch je na tyto vyšší frekvence méně citlivý. Útlum signálu je přibližně -20 dB/dek od 100 Hz. Preemfáze zajišťuje zesílení amplitud spektrálních složek řečového signálu s jejich vzrůstající frekvencí tak, aby došlo k vyrovnání energetického spektra v celém pásmu.

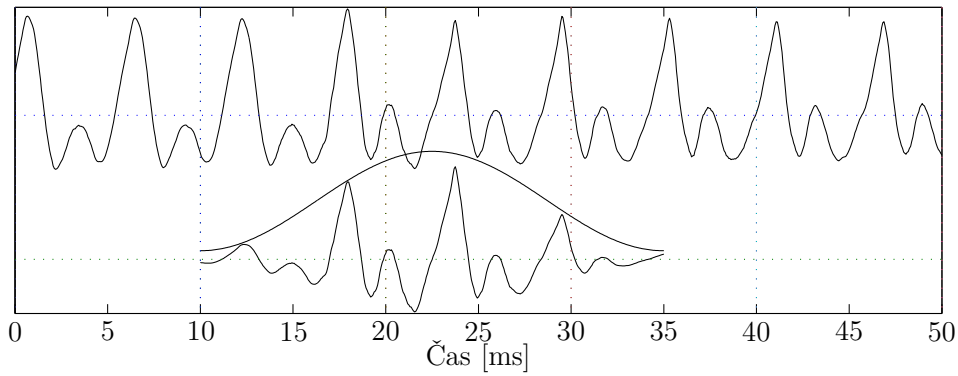
Preemfáze je realizována jako filtr předřazený dalšímu zpracování. Pokud se jedná o analogový filtr, musí mít frekvenční charakteristiku se ziskem +20 dB/dek zhruba od 100 Hz. V případě, že se preemfáze aplikuje na digitalizovaný signál, pracuje podle vztahu 2.1.1,

$$y[n] = x[n] - \alpha x[n-1] \tag{2.1.1}$$

kde $x[n]$ je vstupní vzorek, $y[n]$ je výstupní vzorek v čase n a parametr α se volí z intervalu 0.9 až 1 [1].

2.1.2 Segmentace signálu, váhování oknem a FFT

Pro výpočet mel-spektra je třeba provést segmentaci signálu. Ta vychází z předpokladu, že řeč je v krátkých časových úsecích kvazistacionární. Délka takového mikrosegmentu je typicky 10 ms. Tato doba vychází z dynamických vlastností artikulačních orgánů. Na takový úsek je pak možno uplatnit krátkodobou analýzu signálu. Při té je často žádoucí potlačit vzorky zkraje úseku. Z tohoto důvodu se úsek váhuje Hammingovým oknem. Aby bylo možné u takto váhovaného signálu zachytit periodické vlastnosti znělých úseků signálu, je třeba použít delší časový okamžik. Tato nutnost plyne z frekvenčních vlastností Hammingova okna. Segmentace signálu tedy probíhá po úsecích označovaných jako okno, přičemž toto okno se posouvá o krok, který je často roven délce mikrosegmentu. Obrázek 2.1.2 ukazuje kus akustického signálu, váhovací Hammingovo okno a tímto oknem váhovaný segment. Tento segment má délku 25 ms a krok mezi segmenty je 10 ms.



Obrázek 2.1.2 Ukázka segmentace signálu a váhování Hammingovým oknem. Krok 10 ms, okno 25 ms.

Po segmentaci a váhování je možno převést signál z časové oblasti do frekvenční. K tomu slouží Fourierova transformace. Diskrétní Fourierova transformace je dána vztahem 2.1.2.

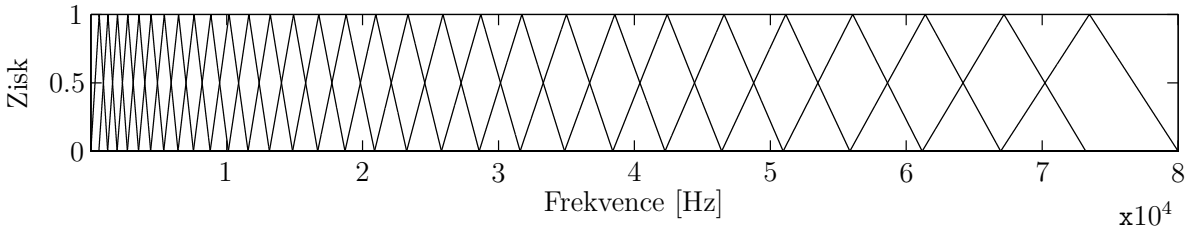
$$S[k] = \sum_{n=0}^{N-1} s[n] \exp\left(-j\frac{2\pi}{N}kn\right) \tag{2.1.2}$$

2.1.3 Banka melovských filtrů

Melovské spektrum se snaží kompenzovat nelineární vnímání frekvencí pomocí banky trojúhelníkových pásmových filtrů s rozložením frekvencí v melovské frekvenční škále. Tato škála je nelineární a je definována pomocí vztahu 2.1.3.

$$f_m = 2595 \log_{10}\left(1 + \frac{f}{700}\right) \tag{2.1.3}$$

Filtry jsou v melovské škále rovnoměrně rozmístěné rovnoramenné trojúhelníky s částečným překryvem. Je možno je rozložit přes celé frekvenční pásmo nebo je možné je omezit tak, aby nezabíraly frekvenční oblasti, kde se nepřenáší užitečná energie signálu. V lineárním měřítku (např. Hz) by filtry už nebyly trojúhelníky, neboť jejich strany by se zdeformovaly transformací do nelineárních křivek. Nicméně většinou se tyto pseudotrojúhelníky aproximují na trojúhelníky. Možné rozložení takových filtrů zobrazuje obrázek 2.1.3. Frekvence od 0 Hz po 8 kHz je pokryta 30 rovnoramennými trojúhelníkovými filtry v melovské nelineární škále, které se částečně překrývají. Filtry jsou transformované a aproximované do lineární škály.



Obrázek 2.1.3 Trojúhelníkové filtry v lineární škále.

Koeficienty FT vypočteny v předcházejících krocích jsou násobeny odpovídajícím ziskem filtru a výsledky jsou pro příslušné filtry sečteny. Výsledkem jsou výkony v jednotlivých frekvenčních pásmech.

2.2 Melovské kepstrální koeficienty

Reálné kepstrum je definováno podle vztahu 2.2.1.

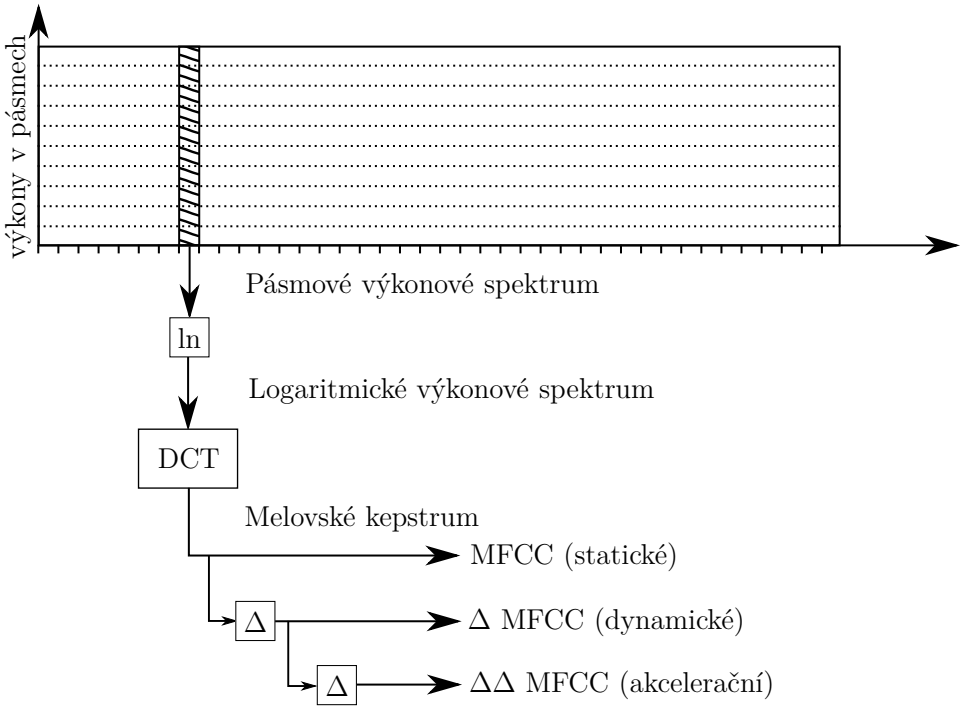
$$C = \mathrm{IDFT}\left\{\ln\left|\mathrm{DFT}(x)\right|\right\} \tag{2.2.1}$$

Melovské kepstrální koeficienty reprezentují krátkodobé výkonové spektrum. Na rozdíl od TRAPs neobsahují informaci o kontextu signálu. Příznakový vektor je vypočítán s použitím hodnot mel-spektra pro jediný segment signálu. Tyto hodnoty jsou zlogaritmovány a následně transformovány pomocí IDFT. Vzhledem k tomu, že výkonové mel-spektrum je reálné a symetrické, IDFT se redukuje na DCT. Výpočet je pak dán vztahem 2.2.2, kde $y_m(i)$ je výkon v i-tém pásmu, M^* je počet pásem banky filtrů a M je počet melovských kepstrálních koeficientů.

$$c_m(j) = \sum_{i=1}^{M^*} \log y_m(i) \cos\left[\frac{\pi j}{M^*}\left(i - \frac{1}{2}\right)\right] \qquad \text{kde } j = 0, 1, ..., M \tag{2.2.2}$$

DCT koncentruje energii signálu na nízkých frekvencích. Proto je možné volit M výrazně menší, než je počet pásem banky filtrů. Obvykle se uvažuje prvních 10 až 13. Díky absenci informace o kontextu není MFCC příliš robustní a snadno dojde ke znehodnocení

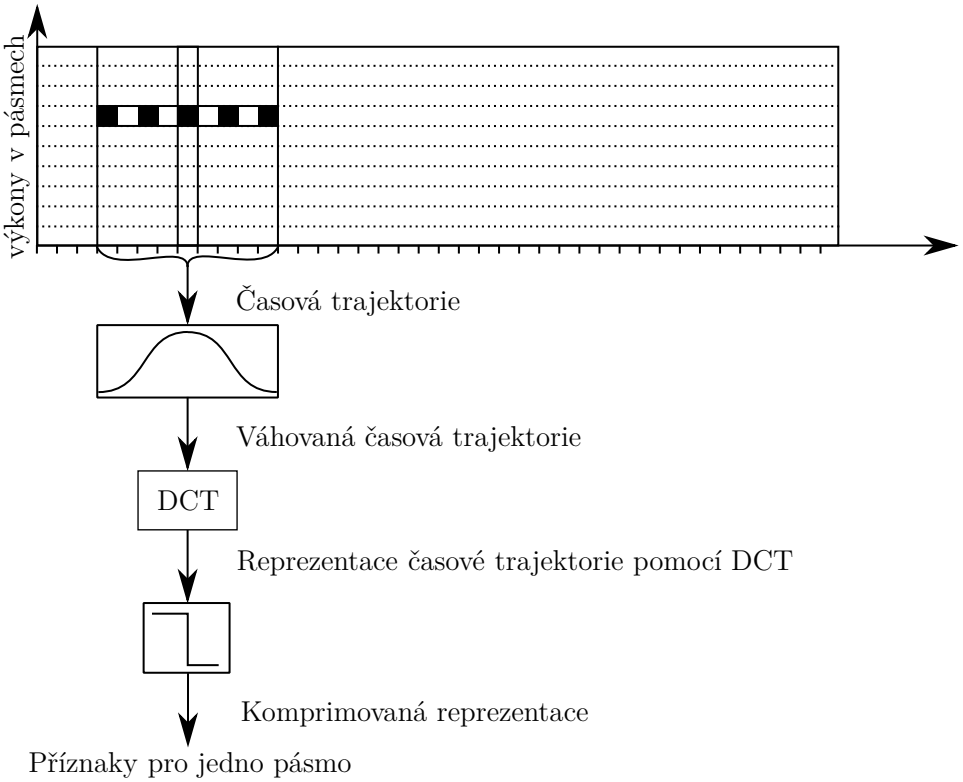
příznaků šumem. Tento efekt lze do jisté míry potlačit přidáním delta a delta-delta koeficientů, které vzniknou derivací příznaků. Délka příznakového vektoru se v tom případě ztrojnásobí. Schematický nákres výpočtu MFCC spolu s delta a delta-delta koeficienty naznačuje obrázek 2.2.1.



Obrázek 2.2.1 Výpočet MFCC z banky melovských filtrů

2.3 TRAP příznaky

TRAPs příznaky stejně jako MFCC vycházejí z mel-spektra. Na místo něj však uvažují i kontext. Příznaky se počítají z výstupu jednoho pásmového filtru. Kontext zde zajišťuje použití několika segmentů, jež tvoří tzv. trajektorii o celkové délce často až jedné vteřiny. Je zřejmé, že takto dlouhá trajektorie obsahuje informace i o okolních fonémech, a ne jen o tom, jež je ve středu trajektorie. Z tohoto důvodu se provádí dekorelace a váhování celé trajektorie Hammingovým oknem. Z hodnot vypočtených DCT je vybráno několik prvních a ty tvoří příznaky pro jedno pásmo výkonového mel-spektra. Výpočtem příznaků pro všechna pásma s daným středovým segmentem a jejich spojením vznikne příznakový vektor. Celý postup schematicky zobrazuje obrázek 2.3.1.



Obrázek 2.3.1 Výpočet TRAPs pro jedno pásmo z banky melovských filtrů

2.4 Pravděpodobnostní příznaky

Z TRAPs příznaků je pro další zpracování potřeba získat pravděpodobnost příslušnosti vektoru parametrů k hláskám (pravděpodobnostní příznaky). Toto je úloha představující nelineární mapování. Je velmi vhodné ji realizovat pomocí umělých neuronových sítí. Při použití funkce *softmax* na výstupu ANN, jsou výstupem přímo pravděpodobnostní příznaky. Za účelem zvýšení úspěšnosti další klasifikace se tyto příznaky po vhodné transformaci mohou zahrnout do nového příznakového vektoru, složeného z těchto transformovaných pravděpodobností a klasických příznaků.

2.4.1 Mapující MLP síť

Umělé neuronové sítě se zakládají na principu fungování biologických neurálních sítí. Snaží se je matematicky popsat a tuto teorii následně aplikovat. Základní prvek neuronových sítí, ať už těch umělých, nebo biologických je neuronová buňka - neuron.

Existuje tak mnoho matematických popisů neuronů, lišících se v používaných matematických funkcích a složitosti modelu. Nejrozšířenější model neuronu je tzv. formální neuron (nebo též McCulloch-Pittův neuron, podle svých autorů) s více vstupy a jediným výstupem. Tento neuron se stejně jako většina modelů skládá z obvodové funkce a akti-

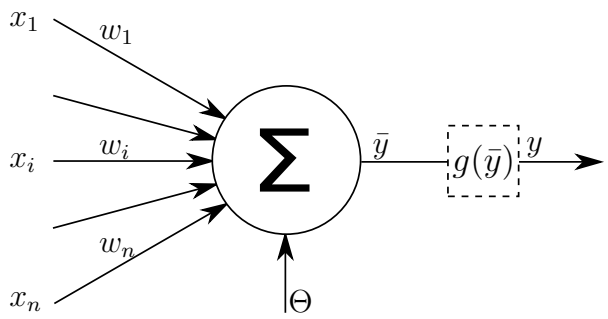
vační funkce. Obvodová funkce určuje jakým způsobem se vstupní parametry kombinují uvnitř neuronu. Aktivační funkce pak určuje transformační vztah mezi vstupy neuronu a jeho výstupem. Jedná se o přenosovou funkci [4]. Obvodová funkce formálního neuronu je dána vztahem 2.4.1, kde x_i jsou vstupní parametry, w_i jsou váhy jednotlivých vstupů a Θ je aktivační práh. Počet vstupních parametrů je n.

$$\bar{y} = \sum_{i=1}^{n} x_i w_i + \Theta \tag{2.4.1}$$

Výsledkem obvodové funkce je aktivační potenciál neuronu na který se následně aplikuje aktivační funkce. Výstup neuronu je tedy dán vztahem 2.4.2.

$$y = g(\bar{y}) \tag{2.4.2}$$

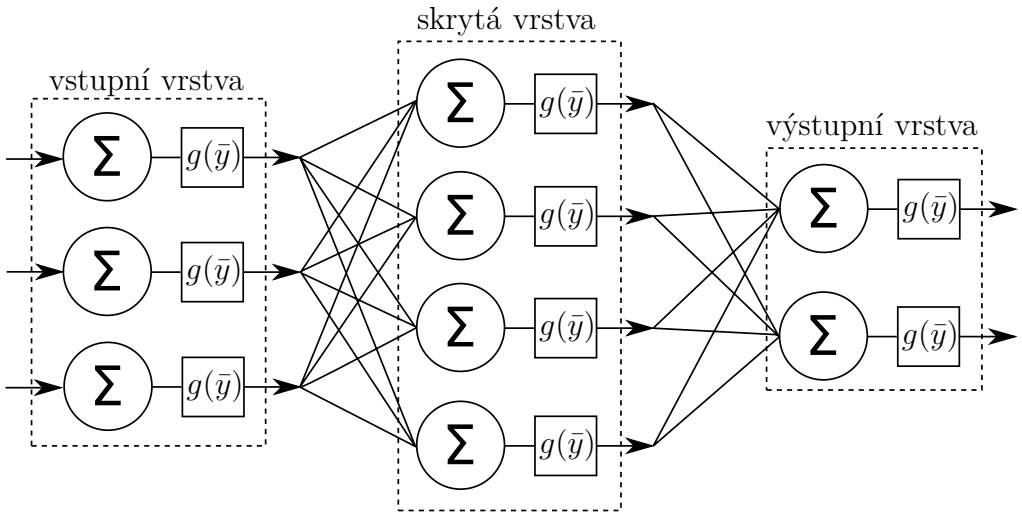
Aktivační funkce může mít mnoho předpisů. Nejčastěji se používá *sigmoida*, *tanh*, nebo *skoková funkce*. Princip formálního neuronu zobrazuje obrázek 2.4.1.



Obrázek 2.4.1 Nákres formálního neuronu

Možné aplikace samotného neuronu jsou velice omezené, a proto se spojují do více či méně složitých sítí. Neuronové sítě mohou mít více vrstev s různými počty a typy neuronů. Základní struktura je však většinou stejná. Vstupní vrstva neuronů obsahuje vždy jeden neuron s jedním vstupem pro každý příznak. Tato vrstva provádí lineární transformaci příznaků. Následuje jedna, nebo několik skrytých vrstev, kde do každého neuronu vstupují výstupy všech neuronů vrstvy předcházející. Jako poslední je výstupní vrstva, která upravuje hodnoty výstupního vektoru do požadované podoby. Například v případě použití ANN k výpočtu aposteriorních pravděpodobností se používá funkce *softmax*, která součet výstupních hodnot normuje na jednotku. Schematický nákres tří-vrstvé umělé neuronové sítě zobrazuje obrázek 2.4.1.

Aby byla neuronová síť schopna správně fungovat, je třeba ji natrénovat a tím nastavit vstupní váhy a práh každého neuronu. Trénování se provádí standardně tak, že se na vstup sítě přiloží trénovací data a výstup sítě se porovnává s očekávaným výstupem. Pomocí různých algoritmů se nastavují hodnoty každého neuronu, dokud se výstup sítě nepřiblíží požadované hodnotě s požadovanou maximální odchylkou. Tomuto způsobu



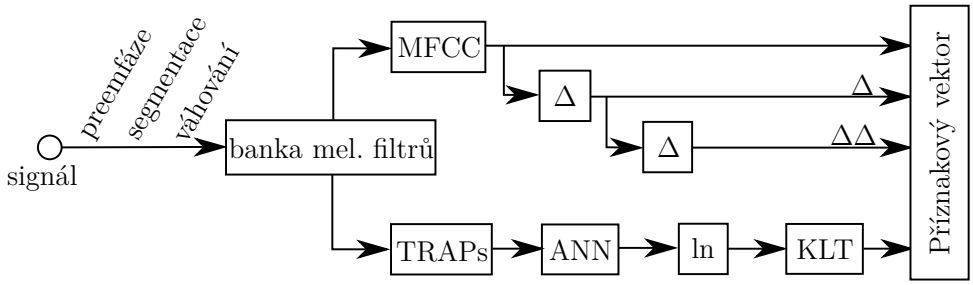
Obrázek 2.4.2 Nákres třívrstvé neuronové sítě

trénování se říká učení s učitelem. Jako algoritmus nastavující váhy a práh se většinou používá algoritmus zpětného šíření chyby.

Výpočet pravděpodobnostních příznaků není součástí této práce. Pro experimenty byla použita data z paralelně řešené bakalářské práce *Jiřího Fialy* [12]. Jeho práce poskytla klíčová data pro tuto práci, neboť experimenty, s konfigurací ANN, za účelem dosažení optimálních příznaků pro použití v TANDEM architektuře a zároveň jejich aplikace, jsou nad rámec jedné bakalářské práce.

2.5 TANDEM příznaky

TANDEM architektura používá speciální typ příznakových vektorů. Jeden ze základních typů vzniká spojením MFCC příznaků a transformovaných pravděpodobnostních příznaků. Transformace pravděpodobnostních příznaků musí zajistit gaussovské rozdělení příznaků a jejich dekorelaci. Gaussovské rozdělení získají po zlogaritmování a dekorelaci zajišťuje KLT. Postup výpočtu příznaků pro TANDEM architekturu zobrazuje nákres 2.5.1.



Obrázek 2.5.1 Postup výpočtu příznaků používaných v základní TANDEM architektuře

2.5.1 Metoda hlavních komponent

Metoda hlavních komponent se někdy nazývá Karhunenova-Loeveova transformace (KLT). Jedná se o transformaci, jež hledá směry největšího rozptylu vstupních dat. To znamená, že je přizpůsobena přímo datům pro něž byla vypočtena, na rozdíl od například DCT. Tato transformace slouží k dekorelaci a kompresi dat. Transformace je dána vztahem:

$$\mathbf{Y} = \mathbf{V} \cdot \mathbf{X}, \tag{2.5.1}$$

kde:

$$\mathbf{Y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}, \quad \mathbf{V} = \begin{bmatrix} v_1[1,n] \\ v_2[1,n] \\ \vdots \\ v_m[1,n] \end{bmatrix} \texttt{a} \quad \mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}. \tag{2.5.2}$$

Vektor $\mathbf{Y}$ je výstupním vektorem s redukovanou dimenzí m. Vektor $\mathbf{X}$ je vstupním vektorem s dimenzí n. Matice $\mathbf{V}$ je transformační matice, v řádcích obsahující vlastní vektory o dimenzi shodné se vstupním vektorem. Těchto vektorů je n. Snížení dimenze se provádí odebráním vlastních vektorů v matici $\mathbf{V}$ na počet požadovaných výstupních dimenzí m.

Pro výpočet KLT je důležité aby data byla zarovnána na střed. Za tímto účelem se vypočte střední hodnota

$$\bar{\mathbf{x}} = \frac{1}{k} \sum_{i=1}^{k} \mathbf{x}_i \,, \tag{2.5.3}$$

kde k je počet vstupních vektorů použitých pro výpočet KLT. Ta se následně od dat odečte

$$\tilde{\mathbf{x}} = \mathbf{x}_i - \bar{\mathbf{x}} \,. \tag{2.5.4}$$

Tím jsou data vycentrována. Dalším krokem je výpočet kovarianční matice z trénovací množiny podle vztahu

$$\mathbf{C}_x = \frac{1}{k} \sum_{i=1}^{k} \tilde{\mathbf{x}} \tilde{\mathbf{x}}^\intercal \tag{2.5.5}$$

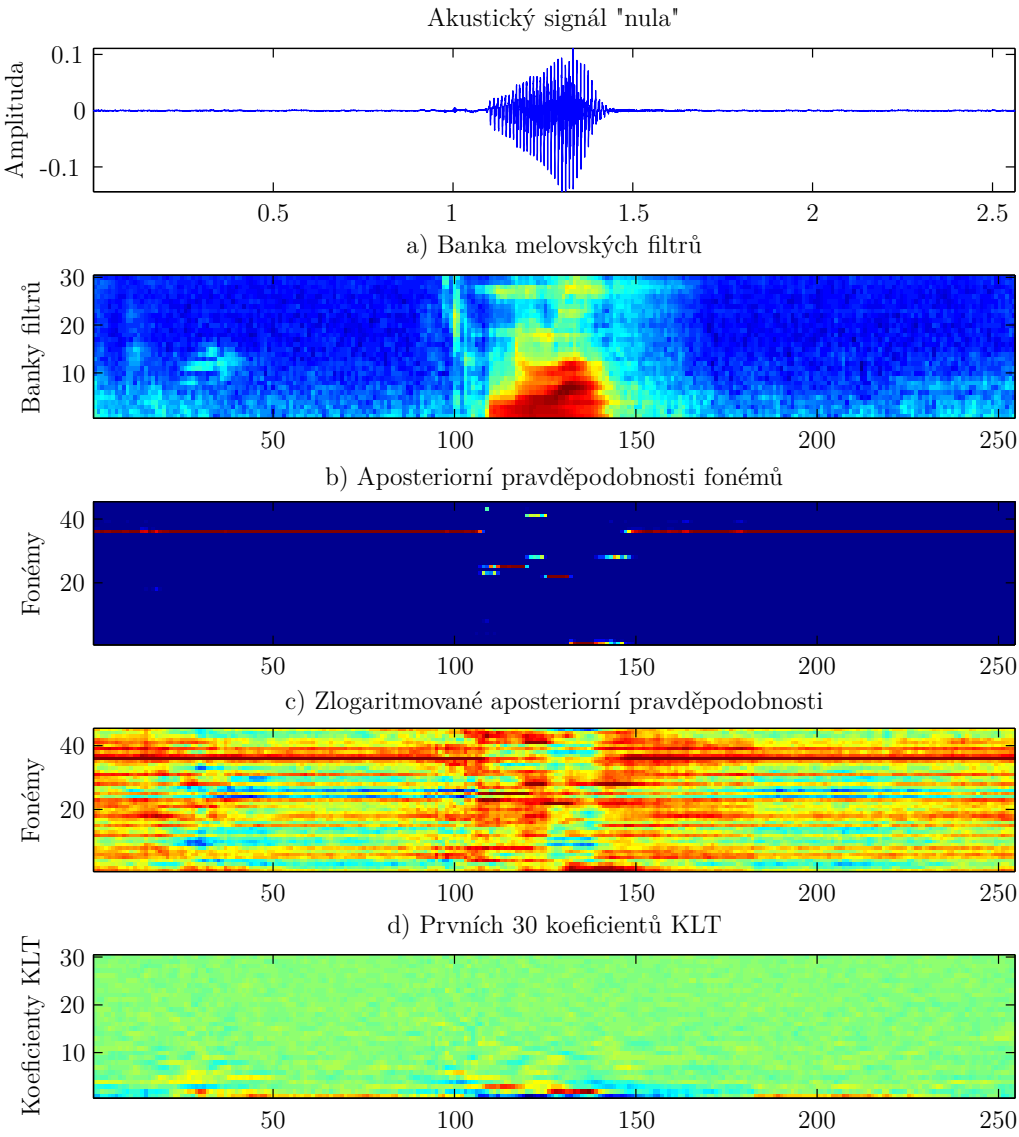
V dalším kroku je třeba vypočítat vlastní čísla a vlastní vektory. Přitom platí vztah

$$\mathbf{C}_y = \mathbf{V} \mathbf{C}_x \mathbf{V}^\intercal \,, \tag{2.5.6}$$

kde $\mathbf{C}_y$ je diagonální matice obsahující vlastní čísla. Seřadí-li se vlastní vektory v matici $\mathbf{V}$ podle velikosti vlastních čísel z matice $\mathbf{C}_y$, budou složky matice $\mathbf{Y}$ seřazené podle rozptylu. V tuto chvíli, pokud neproběhlo zkrácení vlastních vektorů v matici $\mathbf{V}$, má matice $\mathbf{X}$ i $\mathbf{Y}$ stejné rozměry a matice $\mathbf{V}$ a $\mathbf{C}_y$ jsou čtvercové. Výběrem vhodných složek matice $\mathbf{Y}$ nebo úpravou matice $\mathbf{V}$ lze omezit počet dimenzí.

KLT je na rozdíl od diskrétní kosinovy transformace výpočetně náročnější. Ovšem DCT je na datech nezávislá transformace. V případě, že není k dispozici dostatečně reprezentativní vzorek signálů pro výpočet KLT, dochází ke kreslení výsledků pro data výrazněji odlišná od dat použitých pro výpočet KLT. Nicméně pomocí KLT lze dosáhnout lepšího kompresního poměru [2].

Ukázkový příklad aplikace KLT zobrazuje obrázek 2.5.2. Na akustických datech se provede preemfáze, segmentace a váhování Hammingovým oknem. Následně se vypočte mel-spektrum 2.5.2 a). Na tato data se aplikuje TRAP parametrizace a její příznaky se předají ANN. Natrénovaná ANN transformuje TRAP příznaky v aposteriorní pravděpodobnosti. Ty jsou zobrazeny na obrázku 2.5.2 b). Aposteriorní pravděpodobnosti se zlogaritmují, čímž získají gaussovské rozdělení (obrázek 2.5.2 c). Následuje KLT. Data po KLT se zmenšenou dimenzí ukazuje obrázek 2.5.2 d). Je vidět, že nejvíce informace je na nejnižších koeficientech KLT, tedy na vektorech s nejvyšším rozptylem dat. Tato data mají Gaussovské rozložení a jsou dekorelována. Po spojení s MFCC vzniknou příznaky TANDEM architektury, které se používají v rozpoznávači založeném na HMM.



Obrázek 2.5.2 Postup výpočtu KLT koeficientů

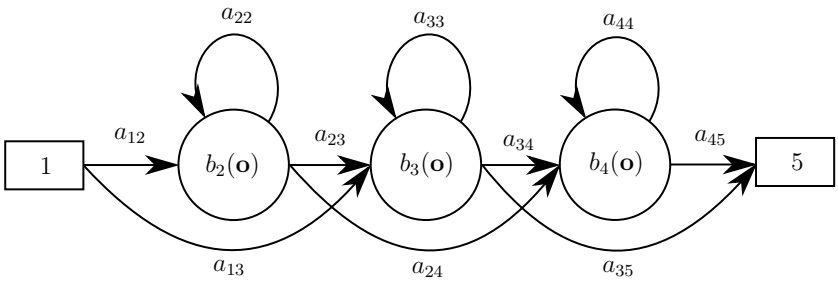
3 Rozpoznávání na bázi HMM

Klasifikace má za úkol z vhodně zvolených příznaků řečového signálu extrahovat obsah. Existuje mnoho metod klasifikace. Velmi používané jsou zmiňované umělé neuronové sítě a v poslední době rozpoznávače založené na skrytých Markovových modelech (HMM). HMM používá i příznaky založené na TANDEM architektuře. Tou se zabývá i tato práce.

3.1 Skryté Markovovy modely (HMM)

Tato technika je dnes nejrozšířenějším způsobem konstrukce rozpoznávačů. Je založená na statistických modelech rozpoznávaných jednotek. Přitom vychází z myšlenky, že se příznakové vektory, příslušející jedné rozpoznávané jednotce, shlukují (jejich vzájemná vzdálenost je malá). Tento princip využívají i rozpoznávače založené na metodě dynamického borcení časové osy (DTW), kdy se neznámý signál porovnává se známými referenčními vzory v prostoru vzdáleností (může se jednat například o eukleidovskou vzdálenost mezi vektory parametrů). Vzhledem k tomu, že DTW porovnává signál se všemi referencemi, ani při použití optimalizované metody DTW není možné realizovat příliš rozsáhlé rozpoznávače, nemluvě o nepřekonatelných problémech při rozpoznávání spojité řeči. Naproti tomu HMM nahrazuje jednotlivé vzory statistickým modelem, vzniklým pomocí zprůměrování velkého množství promluv.

Základem modelu je konečný automat, jehož stavy jsou propojeny do přímé posloupnosti (levopravý model). Pětistavový levopravý model s možnými přeskoky obstav a třemi emitujícími stavy ukazuje obrázek 3.1.1.



Obrázek 3.1.1 Schéma levopravého pětistavového modelu

Lze tvrdit, že Markovův model se skládá z matice pravděpodobností přechodů $\mathbf{A}$ a funkcí $b_i(\mathbf{o})$, které náleží každému emitujícímu stavu i.

3.1.1 Matice pravděpodobností přechodů

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{11} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \tag{3.1.1}$$

Matice přechodů $\mathbf{A}$ reprezentuje pravděpodobnosti přechodu mezi jednotlivými stavy. Její obecný tvar ukazuje rovnice 3.1.1. Jelikož se jedná o matici pravděpodobností, součet čísel na jednotlivých řádcích musí být roven jednotce nebo nule. Pokud se jedná o levopravý HMM, všechny prvky pod hlavní diagonálou jsou nulové. Příklad matice pětistavového levopravého modelu se třemi emitujícími stavy ukazuje rovnice 3.1.2. Model definovaný touto maticí nemá přeskoky, neboť prvky, jež jsou v obrázku 3.1.1 označeny jako a_{13}, a_{24} a a_{35}, jsou nulové.

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0.6 & 0.4 & 0 & 0 \\ 0 & 0 & 0.6 & 0.4 & 0 \\ 0 & 0 & 0 & 0.7 & 0.3 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.1.2}$$

3.1.2 Pravděpodobnostní funkce $b_i(\mathbf{o})$

Funkce $b_i(\mathbf{o})$ jsou pravděpodobnostní funkce, které vyjadřují hustotu pravděpodobnosti jednotlivých klasifikovaných tříd ve stavovém prostoru $\mathbf{O}$, obsahujícím všechna možná pozorování $\mathbf{o}$. Vektory pozorování představují příznakové vektory. Tyto pravděpodobnostní funkce jsou nejčastěji reprezentovány pomocí gaussovské funkce. Jejich výhodou je, že se dají reprezentovat pouze středními hodnotami a rozptyly, které lze získat z trénovacích dat. Obecný předpis n-rozměrné gaussovské funkce s obecnou kovarianční maticí ukazuje rovnice 3.1.3.

$$b_i(\mathbf{o}) = \frac{1}{\sqrt{(2\pi)^n |\mathbf{C}_i|}} \cdot \exp\left(-\frac{1}{2}\sum_{k=1}^{n}(\mathbf{o}-\mu_i)^\intercal \mathbf{C}_i^{-1}(\mathbf{o}-\mu_i)\right) \tag{3.1.3}$$

Pokud jsou data dekorelována, kovarianční matice je pak diagonální a vztah lze zjednodušit tak, jak ukazuje rovnice 3.1.4.

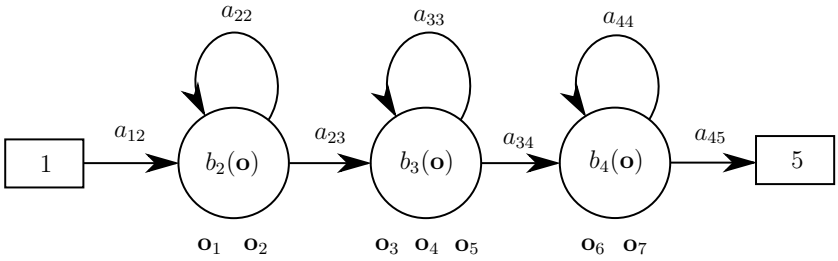
$$b_i(\mathbf{o}) = \frac{1}{\sqrt{(2\pi)^n \prod_{k=1}^{n} \sigma_{ik}^2}} \cdot \exp\left(-\frac{1}{2}\sum_{k=1}^{n}\frac{(o_k-\mu_{ik})^2}{\sigma_{ik}^2}\right) \tag{3.1.4}$$

Směsi (Mixtures)

Aproximace rozložení trénovacích dat pomocí jednoduchých gaussovských funkcí nemusí dostatečně pokrýt variabilitu příznaků. Z tohoto důvodu se často přidávají směsi (anglicky mixtures). V tu chvíli se funkce b stává lineární kombinací několika gaussovských funkcí, výsledný vztah zobrazuje rovnice 3.1.5.

$$b_i(\mathbf{o}) = \sum_{m=1}^{M} c_m \cdot \frac{1}{\sqrt{(2\pi)^n \prod_{k=1}^{n} \sigma_{mk}^2}} \cdot \exp\left(-\frac{1}{2}\sum_{k=1}^{n} \frac{(o_k - \mu_{mk})^2}{\sigma_{mk}^2}\right) \tag{3.1.5}$$

3.1.3 Výpočet pravděpodobnosti průchodu HMM



Obrázek 3.1.2 Příklad funkce HMM s příslušností příznakových vektorů

Vysvětlení funkce HMM bude nejlepší provést na příkladě. Základem je pětistavový model Φ se třemi emitujícími stavy tak, jak je zobrazen na obrázku 3.1.2. Matice $\mathbf{A}$ pak může vypadat tak, jak je naznačeno ve vztahu 3.1.2. Množina příznakových vektorů $\mathbf{O}$ se skládá ze sedmi pozorování $\mathbf{o}_1, \mathbf{o}_2, \ldots \mathbf{o}_7$. Nejpravděpodobnější emise S těchto vektorů pro tento model může být následující:

- $\mathbf{o}_1$, $\mathbf{o}_2$ - stav 2
- $\mathbf{o}_3$, $\mathbf{o}_4$, $\mathbf{o}_5$ - stav 3
- $\mathbf{o}_6$, $\mathbf{o}_7$ - stav 4

Celková pravděpodobnost emise této posloupnosti je vypočtena podle vztahu 3.1.6.

$$P(\mathbf{O}, S|\Phi) = a_{12}b_2(\mathbf{o}_1)a_{22}b_2(\mathbf{o}_2)a_{23}b_3(\mathbf{o}_3)a_{33}b_3(\mathbf{o}_4)a_{33}b_3(\mathbf{o}_5)a_{34}b_4(\mathbf{o}_6)a_{44}b_4(\mathbf{o}_7) \tag{3.1.6}$$

Ve skutečnosti je známá pouze pozorovaná posloupnost $\mathbf{O}$. Nejpravděpodobnější emise $S = \{s(0), s(1), \ldots, s(T{+}1)\}$ je skrytá. Podmíněná pravděpodobnost $P(\mathbf{O}|\Phi)$ proto musí být vypočtena sumací přes všechny možné posloupnosti stavů S. V obecném případě platí vztah 3.1.7.

$$\begin{aligned} P(\mathbf{O}|\Phi) &= \sum_S P(\mathbf{O}, S|\Phi) = \\ &= \sum_S a_{s(0)s(1)} b_{s(1)}(\mathbf{o}_1) a_{s(1)s(2)} b_{s(2)}(\mathbf{o}_2) \cdots_{s(T)} (\mathbf{o}_T) a_{s(T)s(T+1)} = \\ &= \sum_S a_{s(0)s(1)} \prod_{t=1}^{T} b_{s(t)}(\mathbf{o}_t) a_{s(t)s(t+1)} \end{aligned} \tag{3.1.7}$$

Je uvažován $s(0)$ jako vstupní neemitující stav modelu a $s(T+1)$ jako výstupní neemitující stav modelu [1].

Přímý výpočet

Přímý výpočet $P(\mathbf{O}|\Phi)$ podle vztahu 3.1.7 je výpočetně velmi náročný a často neproveditelný. Je totiž potřeba provést $2TN^T$ výpočtů násobení, kde N je počet stavů modelu a T je počet příznakových vektorů $\mathbf{O}$. Je jasné, že tento přístup se pro větší počet příznakových vektorů rychle dostane nad výpočetní možnosti jakýchkoliv počítačů. Mnohem efektivnějším způsobem výpočtu pravděpodobnosti $P(\mathbf{O}, \Phi)$ je tzv. dopředný a zpětný iterativní algoritmus (anglicky forward-backward procedure).

Dopředný a zpětný iterativní algoritmus

Algoritmus forward-backward vyžaduje na rozdíl od přímého výpočtu přibližně pouze N^2T výpočtů násobení.

Při **výpočtu dopředu**, je definována sdružená pravděpodobnost $\alpha_j(t)$. To je pravděpodobnost pozorování prvních t příznakových vektorů $\{\mathbf{o}_1\mathbf{o}_2\ldots\mathbf{o}_t\}$ a případu, že v čase t se model nachází ve stavu s_j za podmínky daného modelu Φ. Platí vztah 3.1.8.

$$\alpha_j(t) = P(\mathbf{o}_1\mathbf{o}_2\ldots\mathbf{o}_t, s(t) = s_j|\Phi) \tag{3.1.8}$$

Hodnoty $\alpha_j(t)$ lze spočítat rekurzivně. Za podmínky, že použijeme model, jež má první a poslední stav neemitující, platí následující postup:

- Inicializace

$$\begin{aligned} \alpha_1 &= 1 \\ \alpha_j(1) &= a_{1j}b_j(\mathbf{o}_1) \qquad \text{pro } 1 < j < N \end{aligned} \tag{3.1.9}$$

- Rekurze pro t = 2, 3, …, T

$$\alpha_j(t) = \left[\sum_{i=2}^{N-1} \alpha_i(t-1)a_{ij}\right] b_j(\mathbf{o}_t) \qquad \text{pro } 1 < j < N \tag{3.1.10}$$

  Nákres principu výpočtu jednoho koeficientu α ukazuje obrázek 3.1.3 zvýrazněný červenou barvou. Tečka ukazuje použité funkce $b_i(\mathbf{o})$.

- Výsledná pravděpodobnost

$$P(\mathbf{O}|\Phi) = \sum_{i=2}^{N-1} \alpha_i(t-1)a_{iN} \tag{3.1.11}$$

Při **výpočtu odzadu**, je definována pravděpodobnost $\beta_j(t)$. To je pravděpodobnost pozorování posledních $T-t$ příznakových vektorů $\{\mathbf{o}_{t+1}\,\mathbf{o}_{t+2}\,\ldots\,\mathbf{o}_T\}$ za podmínky daného modelu Φ, jež je v čase t ve stavu s_j. Platí vztah 3.1.12.

$$\beta_j(t) = P(\mathbf{o}_{t+1}\mathbf{o}_{t+2}\ldots\mathbf{o}_T|s(t) = s_j, \Phi) \tag{3.1.12}$$

Hodnoty $\beta_j(t)$ lze spočítat rekurzivně. Za podmínek shodných s předcházejícími, platí následující postup:

- Inicializace

$$\beta_j(T) = a_{jN} \qquad \text{pro } 1 < j < N \tag{3.1.13}$$

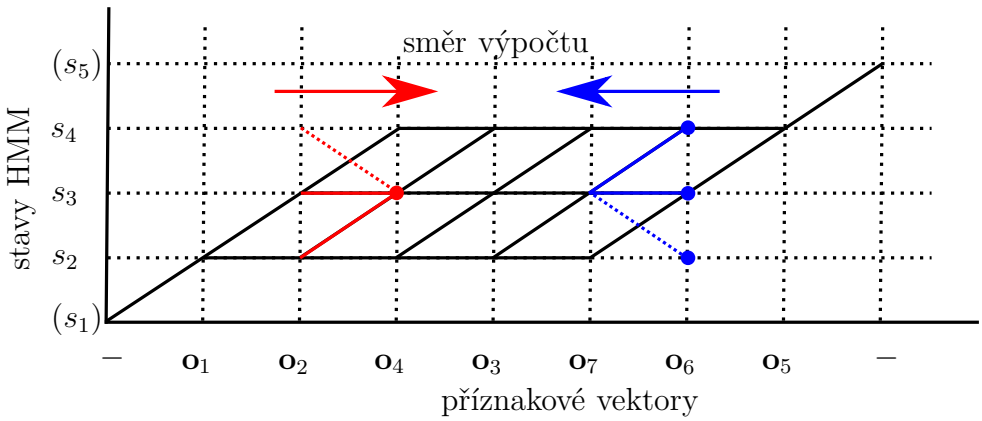
- Rekurze pro t = T-1, ..., 1

$$\beta_j(t) = \sum_{i=2}^{N-2} a_{ji} b_i(\mathbf{o}_t + 1)\beta_i(t+1) \qquad \text{pro } 1 < j < N \tag{3.1.14}$$

  Nákres principu výpočtu jednoho koeficientu β ukazuje obrázek 3.1.3 zvýrazněný modrou barvou. Tečka ukazuje použité funkce $b_i(\mathbf{o})$.

- Výsledná pravděpodobnost

$$P(\mathbf{O}|\Phi) = \sum_{i=2}^{N-1} a_{1i} b_i(\mathbf{o}_1)\beta_i(1) \tag{3.1.15}$$



Obrázek 3.1.3 Ilustrace jedné iterace forward-backward algoritmu

Výsledná pravděpodobnost může být také vypočtena s použitím koeficientů $\alpha_i(t)$ a $\beta_i(t)$ podle vztahu 3.1.16.

$$P(\mathbf{O}|\Phi) = \sum_{i=2}^{N-1} \alpha_i(t)\beta_i(t) \qquad \text{pro } 1 \leq t \leq T \tag{3.1.16}$$

Je třeba zmínit, že výpočet forward-backward algoritmu vede obvykle k numerickému podtečení. Problém lze odstranit například použitím logaritmů pravděpodobností.

Viterbiův algoritmus

Při použití dopředného a zpětného iterativního algoritmu jsou ve výsledném vztahu zahrnuty pravděpodobnosti všech možných cest délky T pro pozorovanou posloupnost příznakových vektorů $\mathbf{O}$. Výpočet lze ještě zrychlit pokud se spokojíme s aproximovanou podmíněnou pravděpodobností $P_S(\mathbf{O}|\Phi)$, jako nejpravděpodobnější posloupností stavů. Tato pravděpodobnost jde napsat podle vztahu 3.1.17.

$$P_S(\mathbf{O}|\Phi) = \max_S P(\mathbf{O}, S|\Phi) = \max_S \left[a_{s(0)s(1)} \prod_{t=1}^{T} b_{s(t)}(\mathbf{o}_t) a_{s(t)s(t+1)} \right] \tag{3.1.17}$$

Pro odvození algoritmu definujeme pro částečnou posloupnost stavů $\{\mathbf{o}_1 \dots \mathbf{o}_t\}$ proměnnou $\varphi_j(t)$, jako pravděpodobnost maximálně pravděpodobné posloupnosti stavů $s(1)$, $s(2)$, ... $s(t) = s_j$ podle vztahu 3.1.18.

$$\varphi_j(t) = \max_{s(1),\dots,s(t-1)} P(\mathbf{o}_1 \dots \mathbf{o}_t), s(1), s(2), \dots, s(t) = s_j|\Phi) \tag{3.1.18}$$

Pro určení maximálně pravděpodobné posloupnosti stavů je potřeba si při dopředném běhu algoritmu zapamatovat v každém časovém skoku t, z kterého stavu byla v předchozím iteračním kroku vybrána maximální hodnota. Z tohoto důvodu zavedeme do algoritmu další proměnnou $\psi_j(t)$, již využijeme při zpětném trasování k nalezení hledané posloupnosti stavů.

Viterbiův algoritmus so do značné míry podobá dopřednému průběhu předešlého algoritmu. Sumace výpočtu je ovšem nahrazena hledáním maxima. Iterativní část dopředného algoritmu se tedy změní tak, jak ukazuje vztah 3.1.19.

$$\begin{gathered} \alpha_j(t) = b_j(\mathbf{o}_t) \sum_{i=2}^{N-1} \alpha_i(t-1) a_{ij} \\ \Downarrow \\ \alpha_j(t) = b_j(\mathbf{o}_t) \max_{2<i<N-1} \alpha_i(t-1) a_{ij} \end{gathered} \tag{3.1.19}$$

Celkový algoritmus pak vypadá následovně:

- Inicializace

$$\varphi_j(1) = a_{1j} b_j(\mathbf{o}_1) \qquad \text{pro } j2 < j < N-1 \tag{3.1.20}$$

$$\psi_j(1) = 0 \tag{3.1.21}$$

- Rekurze pro t = 2, 3, . . . , T a j = 2, . . . , N-1

$$\varphi_j(t) = b_j(\mathbf{o}_t) \max_{i=2,\dots,N-1} [\varphi_i(t-1) a_{ij}] \tag{3.1.22}$$

$$\psi_j(t) = \arg \max_{i=2,\dots,N-1} [\varphi_i(t-1) a_{ij}] \tag{3.1.23}$$

- Výsledná pravděpodobnost $P_S(\mathbf{O}|\Phi)$ a index i_T^* maximálně pravděpodobného stavu v čase $t = T$ se rovná:

$$P_S(\mathbf{O}|\Phi) = \max_{i=2,\ldots,N-1} [\varphi_i(T)a_{iN}] \tag{3.1.24}$$

$$i_T^* = \arg \max_{i=2,\ldots,N-1} [\varphi_i(T)a_{iN}] \tag{3.1.25}$$

Optimální posloupnost stavů lze určit zpětným trasováním funkce $\psi_j(t)$. Pro $t = T - 1, \ldots, 1$ lze indexy hledaných stavů určit ze vztahu 3.1.26. Přičemž je stanoveno, že proces začíná ve stavu $s(0) = s_1$ a končí ve stavu $s(T + 1) = s_N$.

$$i_t^* = \psi_{i_{t+1}^*}(t + 1) \tag{3.1.26}$$

Výpočet Viterbiova algoritmu podle předcházejících stavů může také vést k podtečení, proto se při výpočtu obvykle používají logaritmy přechodových a výstupních pravděpodobností [1].

3.1.4 Určení parametrů HMM

Aby HMM dokázal správně fungovat, je třeba ho natrénovat. Při trénování se nastavují parametry přechodové matice $\mathbf{A}$ a funkce $b(\mathbf{o})$. V případě, že jsou v trénovací množině známy příslušnosti ke stavům jednotlivých modelů, je možné použít přímou metodu.

Přímá metoda

Pokud jsou známé příslušnosti každého segmentu trénovacích dat ke stavům použitého HMM, je možné vypočíst parametry gaussovských funkcí $b(\mathbf{o})$ podle vztahů 3.1.27 a 3.1.28 a pravděpodobnosti přechodů podle vztahů 3.1.29 a 3.1.30, kde N_i je počet výskytů ve stavu i. Přitom uvažujeme levopravý model bez přeskoků.

$$\mu_i = \frac{1}{T} \sum_{t=1}^{T} o_i(t) \tag{3.1.27}$$

$$\sigma_i^2 = \frac{1}{T} \sum_{t=1}^{T} (o_k(t) - \mu_k)^2 \tag{3.1.28}$$

$$a_{i,i} = \frac{N_i - 1}{N_i} \tag{3.1.29}$$

$$a_{i,i+1} = 1 - a_{i,i} \tag{3.1.30}$$

Baum-Welchova reestimace

Předcházející metoda potřebuje znát příslušnost stavů v celé databázi, to je však často příliš náročný požadavek. Z toho důvodu je vhodnější použít například Baumův-Welchův reestimační algoritmus, což je speciální případ EM algoritmu (Expectation-Maximization), který pracuje na principu metody maximální věrohodnosti (anglicky Maximum Likelihood - ML).

Kritérium maximální věrohodnosti předpokládá, že je dán pravděpodobnostní model $P(x|\Phi)$ s neznámými parametry Φ. Úkolem je tyto parametry ocenit pomocí trénovacích dat x_1, x_2, ..., x_N. Přitom využívá princip daný Fisherovou funkcí věrohodnosti. Ta je definována vztahem 3.1.31.

$$F(x_1, x_2, \ldots, x_N|\Phi) = \prod_{n=1}^{N} P(x_n|\Phi) \tag{3.1.31}$$

Hledá se maximum této funkce přes neznámé parametry Φ tak, jak ukazuje vztah 3.1.32. Ten se většinou logaritmuje na vztah 3.1.33.

$$\hat{\Phi} = \arg\max_{\Phi} \prod_{n=1}^{N} P(x_n|\Phi) \tag{3.1.32}$$

$$\hat{\Phi} = \arg\max_{\Phi} \sum_{n=1}^{N} \log(P(x_n|\Phi)) \tag{3.1.33}$$

Pro HMM lze říct, že pomocí této metody jsou hledány optimální hodnoty prvků matice přechodů $\mathbf{A}$, váhy $\mathbf{c}$ jednotlivých směsí a parametry funkcí $b(\mathbf{o})$, které jsou v případě uvažování gaussovských funkcí reprezentovány středními hodnotami $\boldsymbol{\mu}$ a, v obecnějším případě, kovarianční maticí $\mathbf{C}$. Můžeme psát $\Phi = \{\mathbf{A}, \mathbf{c}, \boldsymbol{\mu}, \mathbf{C}\}$. Podrobnější popis EM algoritmu je možno nalézt v [1, kapitola 5.3.3, str. 211]. Baumův-Welchův reestimační algoritmus pak probíhá podle následujících kroků:

1. Zvolíme počáteční odhady parametrů $\bar{a}_{1j}$ $(1 < j < N)$, $\bar{a}_{ij}$ $(1 < i, j < N)$, $\bar{a}_{iN}$ $(1 < i < N)$, $\bar{c}_{jm}$, $\bar{\boldsymbol{\mu}}_{jm}$ a $\bar{\mathbf{C}}_{jm}$ $(1 < j < N, 1 \leq m \leq M)$ kde M je počet mixtures.
2. Provedeme přiřazení $a_{1j} = \bar{a}_{1j}$, $a_{ij} = \bar{a}_{ij}$, $a_{iN} = \bar{a}_{iN}$, $c_{jm} = \bar{c}_{jm}$, $\boldsymbol{\mu}_{jm} = \bar{\boldsymbol{\mu}}_{jm}$a $\mathbf{C}_{jm} = \bar{\mathbf{C}}_{jm}$.
3. Pro každý případ e, kdy e jsou data z trénovací množiny E, se spočítají forward-backward algoritmem pro $t = 1, \ldots, T_e$ hodnoty $P(\mathbf{O}^e|\Phi)$ a $\gamma_j^e(t)$:

$$P(\mathbf{O}^e|\Phi) = \sum_{j=2}^{N-1} \alpha_j^e(t)\beta_j^e(t) \tag{3.1.34}$$

$$\gamma_j^e(t) = \frac{\alpha_j^e(t)\beta_j^e(t}{P(\mathbf{O}^e|\Phi)} \tag{3.1.35}$$

a pro $j = 2, \ldots, N-1$ a $m = 1, \ldots, M$:

$$\gamma_{jm}^e(t) = \begin{cases} \dfrac{1}{P(\mathbf{O}^e|\Phi)} a_{1j} c_{jm} b_{jm}(\mathbf{o}_t^e) \beta_j^e(t) & \text{pro } t = 1 \\ \dfrac{1}{P(\mathbf{O}^e|\Phi)} \sum\limits_{i=2}^{N-1} \alpha_i^e(t-1) a_{1j} c_{jm} b_{jm}(\mathbf{o}_t^e) \beta_j^e(t) & \text{pro } t \geq 2 \end{cases} \tag{3.1.36}$$

4. Určíme nové hodnoty parametrů.

$$\bar{a}_{1j} = \frac{1}{E} \sum_{e=1}^{E} \frac{1}{P(\mathbf{O}^e|\Phi)} \alpha_j^e(1) \beta_j^e(1) \qquad \text{pro } 1 < j < N \tag{3.1.37}$$

$$\bar{a}_{ij} = \frac{\sum\limits_{e=1}^{E} \dfrac{1}{P(\mathbf{O}^e|\Phi)} \sum\limits_{t=1}^{T_e-1} \alpha_i^e(t) a_{ij} b_j(\mathbf{o}_{t+1}^e) \beta_j^e(t+1)}{\sum\limits_{e=1}^{E} \dfrac{1}{P(\mathbf{O}^e|\Phi)} \sum\limits_{t=1}^{T_e} \alpha_i^e(t) \beta_i^e(t)} \qquad \text{pro } 1 < i, j < N \tag{3.1.38}$$

$$\bar{a}_{iN} = \frac{\sum\limits_{e=1}^{E} \dfrac{1}{P(\mathbf{O}^e|\Phi)} \alpha_i^e(T_e) \beta_j^e(T_e)}{\sum\limits_{e=1}^{E} \dfrac{1}{P(\mathbf{O}^e|\Phi)} \sum\limits_{t=1}^{T_e} \alpha_i^e(t) \beta_i^e(t)} \qquad \text{pro } 1 < i < N \tag{3.1.39}$$

$$\bar{c}_{jm} = \frac{\sum\limits_{e=1}^{E} \sum\limits_{t=1}^{T_e} \gamma_{jm}^e(t)}{\sum\limits_{e=1}^{E} \sum\limits_{t=1}^{T_e} \gamma_j^e(t)} \tag{3.1.40}$$

$$\bar{\boldsymbol{\mu}}_{jm} = \frac{\sum\limits_{e=1}^{E} \sum\limits_{t=1}^{T_e} \gamma_{jm}^e(t) \mathbf{o}_t^e}{\sum\limits_{e=1}^{E} \sum\limits_{t=1}^{T_e} \gamma_j^e(t)} \tag{3.1.41}$$

$$\bar{\mathbf{C}}_{jm} = \frac{\sum\limits_{e=1}^{E} \sum\limits_{t=1}^{T_e} \gamma_{jm}^e(t) (\mathbf{o}_t^e - \bar{\boldsymbol{\mu}}_{jm})(\mathbf{o}_t^e - \bar{\boldsymbol{\mu}}_{jm})^\intercal}{\sum\limits_{e=1}^{E} \sum\limits_{t=1}^{T_e} \gamma_{jm}^e(t)} \tag{3.1.42}$$

5. Pokračujeme bodem 2, dokud nedojde ke konvergenci. Přičemž test se provádí před přiřazením.

Počáteční odhady parametrů lze nastavit různě. Jednou z možností je tzv. plochý start, kdy všechny parametry startují ze stejných parametrů, těmi jsou typicky globální střední hodnoty a globální kovarianční matice.

Algoritmy pro práci s HMM jsou již implementovány v balíčku nástrojů HTK.

4 Implementace

Postupy popsané v předchozích kapitolách nebylo nutné implementovat. Většina potřebných nástrojů je dostupná volně a je standardně používána i významnými výzkumnými týmy. Existuje několik balíčků na implementaci HMM. V této práci je použita implementace pomocí balíčku HTK [10]. Používané nástroje jsou popsány v následující části.

4.1 Nástroje pro parametrizaci

Pro výpočet příznaků byly použity nástroje HTK a Quicknet [11].

HCopy

HCopy je nástroj z balíčku HTK a slouží k výpočtu příznaků. Zatímco vstupní soubor může být různého typu, výstupní je vždy typu htk, což je standardní typ pro práci s příznaky HTK balíčku. HCopy dokáže vypočítat mnoho typů příznaků, včetně MFCC s delta a delta-delta koeficienty. Vstupem může být jediný soubor nebo seznam souborů uložený ve formátu scp, což je textový soubor obsahující cesty k souborům, s nimiž chceme pracovat. Scp soubor pro HCopy obsahuje na řádku cestu ke zdrojovému souboru a cestu k cílovému souboru. Je výhodné použít pro nastavení HCopy konfigurační soubor, který obsahuje požadovaná nastavení. Příklad použití HCopy pro výpočet příznaků ukazuje následující příkaz:

```
HCopy -C mfcc_0_d_a_16k_2510_30.be.cfg -S train.speecon.CS0.htk.scp
```

Tento příkaz bude počítat, podle nastavení uvedených v konfiguračním souboru (přepínač `-C`), se soubory definovanými v scp souboru (přepínač `-S`). Podobu konfiguračního souboru je možno nalézt v dokumentaci k HTK (HTKbook).

pfile_klt

Tento nástroj realizuje KLT transformaci. Je součástí balíčku pfile_utils. Což je pomocný balíček softwaru pro trénování a klasifikaci pomocí vícevrstvých dopředných neuronových sítí MLP. Pracuje se soubory ve formátu pfile, což je standardní typ souboru Quicknetu. Vlastností souborů pfile je, že obsahují celou dávku signálů v jednom souboru. Lze pak s nimi pracovat bez nutnosti scp souborů. Bohužel nástroje HTK s tímto

formátem nedokáží pracovat. Výstupem pfile_klt může být pouze transformační matice, nebo rovnou transformovaná data, přičemž je možno vybrat kolik a jaké příznaky bude výstup obsahovat.

Přepínače:

-i *soubor*	Definice vstupního souboru.
-o *soubor*	Definice výstupního souboru.
-os *soubor*	Výstup transformační matice.
-is *soubor*	Vstup transformační matice. Při použití tohoto přepínače pfile_klt nepočítá transformační matici, ale použije dodanou.
-a	Určuje v jakém formátu má pracovat s transformační maticí. (Při čtení i zápisu.)
-fr *rozsah*	Rozsah příznaků výstupního souboru. Pokud požadujeme prvních 10 příznaků, použijeme -fr 0:9.
-n	Při použití **nenormuje** výsledky transformace.

pfile_select

Tento nástroj je rovněž součástí balíčku nástrojů pfile_utils a slouží k práci s jedním souborem, z něhož vybírá rozsah příznaků. Výsledek uloží do výstupního souboru.

Přepínače:

-i *soubor*	Definice vstupního souboru.
-o *soubor*	Definice výstupního souboru.
-fr *rozsah*	Rozsah příznaků výstupního souboru. Použití stejné jako u pfile_klt.

pfile_merge

Pfile_merge, nástroj z balíčku pfile_utils, spojuje dva soubory s příznaky do jednoho. Samozřejmě je možné vybrat rozsah příznaků u obou souborů.

Přepínače:

-i1 *soubor*	Definice prvního vstupního souboru.
-i2 *soubor*	Definice druhého vstupního souboru.
-o *soubor*	Definice výstupního souboru.
-fr1 *rozsah*	Rozsah vybíraných příznaků z prvního souboru. Použití stejné jako u pfile_klt.
-fr2 *rozsah*	Rozsah vybíraných příznaků z druhého souboru. Použití stejné jako u pfile_klt.

feacat

Feacat je nástroj, umožňující konverzi mezi rozmanitými typy souborů. Umožňuje spojovat i rozdělovat soubory dané seznamem do/z jednoho souboru. Také umožňuje spočítat některé základní transformace, jako například logaritmus.

Přepínače:

-i *soubor*	Definice vstupního souboru. (V případě jednosouborového vstupu.)
-ip *formát*	Určení formátu vstupního souboru. Základní nastavení je pfile.
-op *formát*	Určení formátu výstupního souboru. Základní nastavení je pfile.
-o *soubor*	Definice výstupního souboru (V případě jednosouborového výstupu.)
-l *soubor*	Definice seznamu vstupních souborů v případě, že vstupem je více souborů. Seznamem je soubor typu scp.
-ol *soubor*	Definice seznamu výstupních souborů v případě, že výstupem je více souborů. Seznamem je soubor typu scp.
-transform *trans*	Požadovaná transformace.

4.2 Nástroje HTK pro trénování a rozpoznávání

HTK je rozsáhlý volně dostupný nástroj pro práci s HMM. Obsahuje mnoho nástrojů, ke kterým je k dispozici rozsáhlý manuál HTKbook, ke stažení na [10]. V této práci jsem použil jen několik z mnoha nástrojů.

HCompV

Tento nástroj počítá globální střední hodnoty a kovariance. Je použit zejména pro inicializaci parametrů HMM při tzv. plochém startu. Je dobré nastavit *floor* jako minimální hodnotu počítaných hodnot. Toto nastavení zabrání podtečení, nastavuje se přepínačem `-f`. Kovariance jsou počítány vždy, ovšem počítání středních hodnot musí být vynuceno přepínačem `-m`.

HERest

HERest slouží k přepočítání parametrů HMM, pomocí Baum-Welchovy reestimace. Podporuje paralelizaci při které spočítá na částech trénovací množiny tzv. *akumulátory*, ze kterých následně spočítá nové parametry HMM. Výpočet *akumulátoru* může vypadat například takto:

```
HERest -p 3 -I speecon.phones_sil.mlf -t 250.0 150.0 1000.0
-S train.TANDEM.part3.scp -H hmmdefs -H macros -M models/hmm monophones_sil
```

Přepínač `-p` značí, že se jedná o třetí část paralelního výpočtu. K výpočtu jsou použity soubory uvedené u přepínače `-S`. Práh prořezávání nastavuje přepínač `-t`. Mlf soubor použitý u přepínače `-I` obsahuje fonetický přepis trénovacích signálů. Tento přepis smí obsahovat pouze fonémy uvedené v souboru se seznamem fonémů `monophones_sil`. Hmmdefs a macros obsahují definici HMM použitého při přetrénování a přepínač `-M` nastavuje výstupní složku pro přepočítané parametry (v tomto případě pouze *akumulátory*). Následující příkaz pak všechny *akumulátory* spojí do nové definice HMM, které uloží do složky definované přepínačem `-M`

```
HERest -p 0 -H hmmdefs -H macros -M models/hmm monophones_sil
models/hmm/*.acc
```

HHEd

Tento nástroj slouží k manipulaci s HMM modely. Umožňuje je klonovat, spojovat stavy i měnit strukturu. Požadované změny je třeba definovat v konfiguračním souboru hed. Tyto soubory obsahují transformační pokyny pro nástroj HHEd. Jedná se o soubory se specifickou syntaxí, která je popsaná v HTKbook.

HParse

HParse slouží k překladu gramatiky z formátu *extended Backus-Naur Form* (EBNF) do formátu používaného HTK. Výsledný soubor se používá jako definice gramatiky pro nástroj HVite.

HResult

HResult je nástroj pro vyhodnocení úspěšnosti testu HMM. Pracuje se soubory typu mlf rozpoznaných signálů, jež porovnává s databázovým přepisem. Výstup pak může vypadat například takto:

```
====================== HTK Results Analysis =======================
  Date: Fri May  9 04:09:00 2014
  Ref : /scratch1/brich/TANDEM/source/test.speecon.CS0.mlf
  Rec : /scratch1/brich/TANDEM/Tm.MFCC_0_D/result/hmm91.mlf
------------------------ Overall Results --------------------------
SENT: %Correct=74.50 [H=149, S=51, N=200]
WORD: %Corr=90.77, Acc=85.05 [H=413, D=7, S=35, I=26, N=455]
===================================================================
```

HVite

Tento nástroj představuje univerzální Viterbiho rozpoznávač. Slouží k testování HMM. K tomu je třeba dodat soubor s gramatikou, jež vytvoříme pomocí nástroje HParse, dále seznam fonémů a soubor s fonetickým přepisem slov v gramatice. Výsledkem je label file obsahující rozpoznané promluvy podle dodané gramatiky.

4.3 Paralelní zpracování úloh

Vzhledem k náročnosti výpočtů a velkému objemu dat byl použit výpočetní cluster, který tvoří několik počítačů s operačním systémem UNIX. Používaný výpočetní cluster tvoří základní server označený *amagi*, jež spravuje jednotlivé výpočetní jednotky označované jako *magi*`xxx`, kde `xxx` je číselné označení jednotky (například *magi201*). Na diskovém poli připojeném k *amagi* jsou složky obsahující databáze signálů (složka /data), pracovní adresáře jednotlivých uživatelů (složka /scratch) a domovské adresáře uživatelů, které se zálohují (složka /home). Paralelizaci zajišťuje *SGE* toolkit [9].

Tento balíček programů, vyvíjený společností Oracle, je na výpočetním clusteru použitý k paralelizaci úloh. Tvoří ho několik nástrojů, z nichž uvedu jen ty, které jsou důležité pro organizaci fronty, další je možné nalézt v manuálu.

qsub

Tento příkaz vkládá úlohy do fronty ke zpracování. Úloha může být jak skript, tak i binární soubor. V tom případě je nutné použít příslušný přepínač. Používané přepínače:

-b *y/n*	Při volbě *y* může být úloha skript i binární soubor. Při volbě *n* pouze skript.
-o *soubor*	Nastavení standardního výstupu úlohy do souboru.
-e *soubor*	Nastavení chybového výstupu úlohy do souboru.
-q *list_stroju*	Nastavení, na jakých výpočetních jednotkách může být úloha spuštěna.
-hold_jid *list_ID*	Nastavuje závislosti. Úloha není spuštěna, dokud všechny úlohy ze seznamu list_ID nejsou dopočítány.

Listy v SGE jsou definovány jako hodnoty oddělené čárkou. Za přepínači následuje volání skriptu, případně jeho parametry. Příklad použití příkazu qsub je:

```
qsub -b y -o ~/log_std -e ~/log_err -hold_jid 1,2,3 ls ~
```

Tento příkaz vloží do fronty příkaz `ls ~`, jelikož to není skript, je nutno použít přepínač `-b`. Výstup tohoto příkazu se uloží do souboru `~/log_std` nebo v případě, že

příkaz skončí chybou, do souboru `~/log_err`. Díky přepínači `-hold_jid` se tato úloha nevykoná, dokud nejsou vykonány úlohy 1,2 a 3. Přepínač `-q` není použit, a tudíž se úloha pustí na jakékoliv z přístupných výpočetních jednotek. Další parametry lze nalézt v manuálu.

qstat

Slouží k vypsání úloh uživatele ve frontě. Při použití přepínače `-u`, následovaného jménem uživatele, vypíše úlohy požadovaného uživatele.

qdel

Tento příkaz odstraňuje úlohy z fronty.
Použití:

qdel *ID_list*	Smaže z fronty úlohy definované *ID_list*em.
qdel -u *uživatel*	Smaže z fronty všechny úlohy *uživatel*e.

4.4 Příklad implementace

Skripty přiložené k této práci předpokládají následující datovou strukturou:

```
|-- data                 # Surová parametrizovaná data
|-- lists                # Soubory scp
|-- mlf                  # Soubory mlf
|-- scripts              # Složka obsahující hlavní skripty
|   |-- addon            # Složka obsahující pomocné skripty
|-- source               # Složka obsahující dodatečné soubory
|-- T.MFCC_0:25_klt_0:29 # Pracovní složka specifické architektury
```

Vzhledem k velkému množství různých typů příznakových vektorů použitelných v TANDEM architektuře, je výhodné napočítat data v co nejobecnějším tvaru a posléze pomocí nástrojů pfile_utils vybrat ta, která chceme použít. Složka data obsahuje právě tato surová data. To znamená bez redukované dimenze. Všechna data jsou uložena odděleně pro testovací a trénovací množinu v souborech typu pfile.

Ve složce lists jsou soubory scp obsahující reference na databázi akustických signálů, databázi aposteriorních pravděpodobností a odvozené seznamy. Seznamy jsou vždy pro testovací a trénovací množinu.

Složka mlf obsahuje label files pro trénovací a testovací množinu.

Ve složce scripts jsou uloženy všechny používané skripty. Ve složce addon jsou pak uložené pomocné skripty, které jsou volány zevnitř hlavních skriptů.

Složka source obsahuje konfigurační soubory, transformační soubory, definice gramatiky, slovníky, atd.

Další složky, jež jsou zde zastoupeny složkou T.MFCC_0:25_klt_0:29, jsou již složky obsahující jednotlivé realizace TANDEM architektury a HMM. Tyto složky mají vnitřní strukturu následující:

```
|-- data                     # Složka s htk soubory parametrů
|-- result                   # Složka obsahující soubory s~výsledky
|-- prubezne_uspesnosti.res # Soubor s vypsanými úspěšnostmi
|-- test.speecon.htk.scp     # Scp soubor s testovacími daty
|-- train.speecon.htk.scp    # Scp soubor s trénovacími daty
|-- test.TANDEM.MFCC_0:25.post_0:29.pfile  # Pfile soubor pro test
|-- train.TANDEM.MFCC_0:25.post_0:29.pfile # Pfile soubor pro trénink
```

Složka data obsahuje htk soubory jednotlivých parametrizovaných signálů. Struktura této složky je uložena v souborech test.speecon.htk.scp a train.speecon.htk.scp. Tyto htk soubory vzniknou rozložením souborů test.TANDEM.MFCC_0:25.post_0:29.pfile a train.TANDEM.MFCC_0:25.post_0:29.pfile na jednotlivé signály. Tyto pfile soubory obsahují signály parametrizované podle požadované architektury a vznikají z obecných parametrizovaných dat pomocí nástrojů pfile_utils.

Složka result obsahuje dva typy souborů. Mlf soubor vzniklý při testu HMM. Tento mlf obsahuje *rozpoznaný* přepis testovacích signálů. A soubor typu res, jež vzniká při porovnání testovacího mlf s databázovým a obsahuje informace o úspěšnostech. K této složce se váže soubor prubezne_uspesnosti.res. Jedná se o soubor typu csv, obsahující pouze číslo modelu a úspěšnost. Slouží k rychlé kontrole průběhu úspěšností a tvorbě grafů.

Samotný postup práce se skládá z několik bodů:

- **Generování scp souborů** pomocí skriptu `make_lists.sh`. Tento skript upravuje scp soubor s odkazy na akustické signály v databázi na scp soubor s odkazy na soubory s příznakovými vektory. V tomto skriptu je třeba správně nastavit následující proměnné:

  `path` Nastavuje jaká cesta se vloží před záznam.

  `lists` Nastavuje cestu ke složce s scp. V této složce jsou očekávány zdrojové scp a jsou do ní ukládány nové scp.

- **Výpočet příznaků MFCC** je realizován skriptem `count_fea.pl` ve dvou fázích. Nejprve se pomocí HCopy vypočítají htk soubory s příznaky a ty se pak pomocí nástroje feacat spojí do jednoho pfile. To vše pro testovací i trénovací data. Příkaz HCopy pro výpočet testovacích příznaků je následující:

```
HCopy -C $root/source/mfcc_0_d_a_16k_2510_30.be.cfg
-S $lists/test.speecon.CS0.htk.scp
```

Příkaz pro vytvoření pfile souboru je:

```
feacat -l $lists/test.speecon.htk.scp -ip htk -op pfile
-o $data/test.speecon.pfile
```

Pro trénovací množinu je postup obdobný.
Z příkazů výše je vidět, že i v tomto skriptu je třeba nastavit některé proměnné:

`root`	Nastavuje domovskou složku výpočtu.
`lists`	Nastavuje složku s scp soubory.
`data`	Nastavuje složku pro uložení napočítaných dat.

- Jako další se počítají **klt příznaky MFCC**. K tomu slouží skript `PCA_MFCC.pl`. Tento skript obsahuje proměnné `frF` a `frS`, které vybírají rozsah příznaků, z nichž je počítána klt. Příklady nastavení jsou:

`frF=0, frS=12`	Vypočte klt příznaky statického MFCC.
`frF=0, frS=25`	Vypočte klt příznaky statického a delta MFCC.
`frF=0, frS=38`	Vypočte klt příznaky statického, delta a delta-delta MFCC.
`frF=13, frS=25`	Vypočte klt příznaky pouze delta MFCC.
`frF=26, frS=38`	Vypočte klt příznaky pouze delta-delta MFCC.

  Je nutné nastavit proměnné `root`, `lists` a `data` na hodnoty shodné s předchozím bodem. Postup výpočtu je téměř shodný s následujícím bodem.
- Následuje výpočet **klt příznaků aposteriorních pravděpodobností** pomocí skriptu `PCA.pl`. Výpočet probíhá ve třech fázích. Nejprve se pomocí příkazu

  ```
  feacat -l $lists/train_post.scp -ip htk -op pfile -transform safelog
  -o $klt/train_post_log.pfile
  ```

  vypočítají logaritmy aposteriorních pravděpodobností, čímž data získají gaussovské rozložení. Tento krok je v předcházejícím skriptu vynechán, neboť MFCC již mají toto rozložení. V druhém kroku se vypočte z trénovacích logaritmovaných dat transformační matice PCA transformace:

  ```
  pfile_klt -i $klt/train_post_log.pfile -os $klt/train_MV -a -n
  ```

  Nakonec se vypočte samotná transformace pomocí příkazu:

  ```
  pfile_klt -i $klt/train_post_log.pfile -o $klt/train_klt.pfile
  -is $klt/train_MV -a -n
  ```

V tomto skriptu je třeba nastavit proměnné `root`, `lists` a `data`.

Práce s dalšími skripty je o něco složitější, neboť je třeba nastavovat je přímo pro požadované parametry příznakového vektoru TANDEM architektury.

- **TANDEM.pl**
  Tento skript je používán pro výpočet příznakových vektorů TANDEM architektury. V základní podobě umožňuje spojení MFCC příznaků s klt příznaky aposteriorních pravděpodobností. Výpočet se provádí ve třech krocích:

  – V prvním kroku se vytvoří složka se jménem označujícím realizovanou TANDEM architekturu a v této složce se vytvoří scp soubory odkazující na budoucí příznakové soubory.

  – V dalším kroku se v této složce vytvoří pfile soubory (jeden s testovacími a druhý s trénovacími daty). Tuto část je potřeba upravit v případě, že je požadována jiná architektura TANDEM, než je tento skript schopen vygenerovat v současné podobě. Použitý příkaz je:

  ```
  pfile_merge -i1 $data/test.speecon.pfile -i2 $klt/test_klt.pfile
  -fr1 $frMF:$frMS -fr2 $frF:$frS
  -o $outdir/test.TANDEM.MFCC_$frMF:$frMS.post_$frF:$frS.pfile
  ```

  Proměnné `frMF` a `frMS` určují rozsah příznaků vybraných z prvního souboru (přepínač `-i1`) a `frF` a `frS` určují rozsah příznaků z druhého souboru (přepínač `-i2`).

  – V posledním kroku dojde k rozdělení souborů pfile, vygenerovaných v předcházejícím kroku, podle seznamu v scp souborech, vytvořených v prvním kroku.

- **train.TANDEM.mix.monot.pl**
  Tento skript slouží k samotnému trénování HMM. Je třeba v něm správně nastavit hodnoty proměnných:

`npar` Počet příznaků v použitém příznakévém vektoru.

`param` Název pracovní složky. Jedná se o složku obsahující scp soubory, pfile soubory a složku s htk soubory vygenerovanými skriptem TANDEM.pl.

`sign` Počet dat použitých pro trénování.

`parts` Tato proměnná určuje, kolik bude počítáno *akumulátorů* a tedy míru paralelizace.

`domov` Tato proměnná je ekvivalentní proměnné `root` z předcházejících skriptů.

`source` Proměnná obsahuje cestu ke složce source.

`scr` Toto je cesta ke složce obsahující ostatní skripty. Tato proměnná je důležitá hlavně proto, že v průběhu tohoto skriptu se volají pomocné skripty z této složky.

`mlf` Proměnná obsahuje cestu ke složce mlf.

Schéma skriptu je zobrazeno na obrázku 4.4.1. Popis jednotlivých kroků je následující:

1. V tomto bloku dojde k vytvoření adresáře hmm0 jako výchozího adresáře pro další kroky. Také se překopíruje, a případně ořízne na počet záznamů určených proměnnou `sign`, scp soubor s odkazy na trénovací data.

2. Scp soubor vytvořený v minulém bloku se zde rozdělí na `parts` částí. Jedná se o přípravu na paralelní trénování.

3. V této části se volá skript `create_proto.sh`. Ten má za úkol vygenerovat soubor *proto*, který slouží jako prototyp při generování HMM.

4. Tento blok počítá hodnoty pro tzv. *plochý start*. Pracuje s jednou částí výše zmíněného scp souboru. Hodnoty pro *plochý start* není nutné počítat přes celou trénovací množinu, stačí dostatečně reprezentativní vzorek.

5. Prototypový soubor s hodnotami *plochého startu* se nyní naklonuje pro každý foném, čímž vznikne první kompletní definice HMM. Tu je nyní nutné přetrénovat.

6. Tento blok počítá *akumulátory*. Využívá se přitom paralelního zpracování. Každý *akumulátor* pracuje s jednou částí scp souboru, která byla vytvořena v bloku 4.4.

7. Po spočtení všech akumulátorů jsou spojeny a aktualizují se parametry HMM.

8. Skript `recognizer.pl` slouží k získání úspěšnosti právě aktualizovaného modelu. Implementuje HMM rozpoznávač, který lze použít i samostatně, a tudíž bude podrobněji popsán v další části. Po tomto bloku dochází k návratu a novému přetrénování právě vypočteného HMM. Tato smyčka proběhne 10x, čímž dojde k postupnému zlepšení parametrů HMM, nebo naopak k přetré-

nování a poklesu úspěšnosti.

9. V tomto bloku dojde k vyhodnocení úspěšností právě proběhlé smyčky a vybrání modelu s nejvyšší úspěšností, jako výchozího k dalšímu zpracování.

10. - 12. V těchto částech dochází k přidání modelu krátké pauzy do HMM a opětovnému desetinásobnému přetrénování. Pro model krátké pauzy se nepoužívá pětistavový levopravý model, ale pouze třístavový.

13. - 16. V této části dochází k zarovnání trénovací množiny. Pro úlohy rozpoznávání řeči jsou pro trénování potřeba databáze signálů se známými hranicemi hlásek. To je samostatná netriviální úloha, která se řeší dvěma způsoby. Buď se tyto hranice hledají ručně, nebo se provádí automatická segmentace. Obě tyto metody mají samozřejmě určitou chybovost. Zarovnání je proces, který odstraňuje tato chybná data. Postup je takový, že se provede test trénovacích dat na částečně natrénovaném HMM a vyřadí se data, jejichž úspěšnost klasifikace je o mnoho horší než je průměr na celé množině. Taková data jsou s nejvyšší pravděpodobností chybná, nebo natolik odlišná, že jsou pro trénování nepoužitelná. Veškerá další práce s HMM se již provádí na zarovnaných datech. Nutno podotknout, že z 55000 signálů trénovací množiny u žádného typu příznaků nedošlo k odstranění více než několik desítek signálů.

17. Tento blok přidává mixtures. To se provádí pomocí nástroje HHEd s jednoduchým nastavením konfiguračního souboru:

```
MU 2 {*.state[2-4].mix}
```

Číslice po MU určuje nový počet mixtures v modelu a následuje specifikace, na jaké stavy budou tyto mixtures aplikovány.

18. Následuje opět deset přetrénování. Dále je možné opakovat přidání mixtures.

- **recognizer.pl**

  Tento skript má, na rozdíl od předcházejících popisovaných, tři vstupní parametry:

`model`	Číslo testovaného modelu.
`param`	Název pracovní složky.
`sil`	Boolean hodnotu určující, zda se jedná o model s přidanou krátkou pauzou (SP), nebo ne.

  Volání tohoto skriptu pak vypadá následovně:

  ```
  ./recognizer.pl 15 T.MFCC_0:25_klt_0:29 0
  ```

  V tomto případě se bude testovat model 15 v pracovní složce `T.MFCC_0:25_klt_0:29` a jedná se o model s přidanou krátkou pauzou.

  Uvnitř skriptu je pak nutné nastavit proměnné:

`sign`	Počet dat použitých pro test.
`test_source`	Jedná se o název scp souboru s odkazy na testovací data.
`domov`	Tato proměnná je ekvivalentní proměnné `root` z předcházejících skriptů.
`gram`	Definuje polohu souboru s gramatikou.
`dict`	Definuje soubor slovníku s fonetickým přepisem slov v gramatice.
`master_mlf`	Určuje mlf soubor s databázovým přepisem obsahu testovací množiny.

Vlastní výpočet je rozdělen do tří fází. V první se pomocí příkazu `HParse` vygeneruje zápis gramatiky ve formátu používaném HTK. Ve druhé dochází k samotné klasifikaci pomocí příkazu

```
HVite -A -H $hmm$model/hmmdefs -H $hmm$model/macros
-S $hmm$model/$test_file -l \\'*\\' -i $result/hmm$model.mlf
-w $wdnet -p 0.0 -s 5.0 $dict $monophones
```

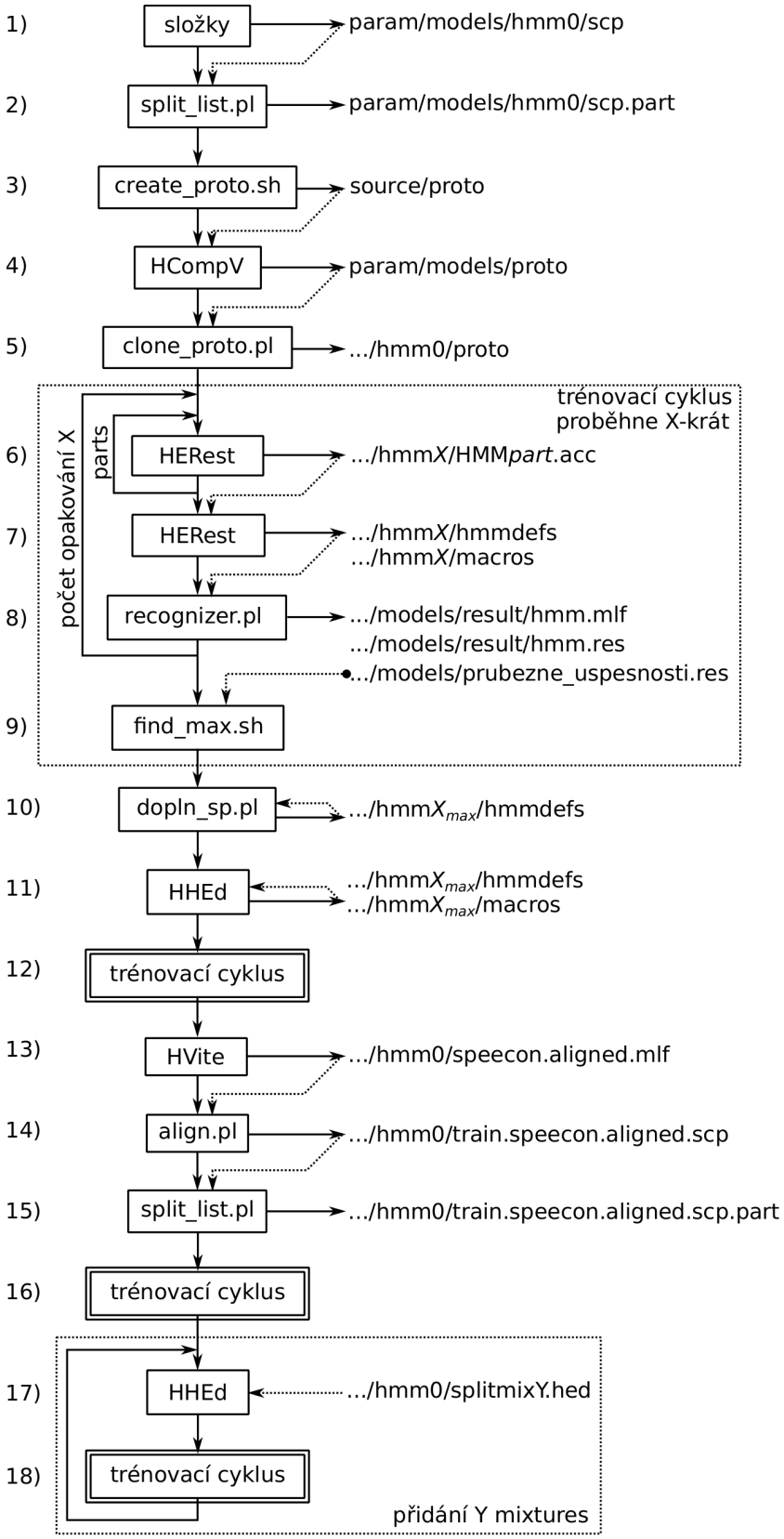
Proměnné `$hmm$model` určují složku s definicí HMM, který chceme testovat. `Test_file` je scp soubor s testovacími daty, parametr `-i` nastavuje kam se uloží rozpoznaný label file. Pomocí `-w` definujeme gramatiku, `dict` je slovník fonetických přepisů slov v gramatice a `monophones` je pak seznam fonémů.

Ve třetí fázi probíhá vyhodnocení výsledků. Pomocí nástroje `HResult` se porovná databázový mlf s rozpoznaným mlf.

Pro paralelní výpočet je třeba vložit jednotlivé úkony do fronty SGE. K tomu slouží příkaz `qsub`. Je vhodné nastavit výstup standardních a chybových hlášení do jednoho souboru. SGE umožňuje také tvorbu závislostí mezi úlohami, čehož lze výhodně použít pro automatické spravování výpočtů. Příkazy SGE se umisťují před příkazy, které chceme do SGE fronty přidat. Důležitý požadavek pro takové úlohy je, aby všechny cesty byly definovány absolutně. Příkaz pak může vypadat takto:

```
qsub -b y -o $log -e $log -hold_jid $IDD $scr/recognizer.pl $M $param 1
```

Proměnná `log` odkazuje na soubor pro standardní i chybová hlášení. `IDD` může obsahovat seznam závislostí (jednotlivá ID oddělená čárkou) a `scr` obsahuje absolutní cestu ke skriptu `recognizer.pl`.



Obrázek 4.4.1 Funkce skriptu train.TANDEM.mix.monot.pl

5 Experimenty

V této části byly realizovány experimenty, kde byla testována úspěšnost rozpoznávání na úloze rozpoznávání číslovek. Rozpoznávač byl realizován na bázi HMM s příznakovými vektory TANDEM architektury. Experimenty byly prováděny s ohledem na různá nastavení.

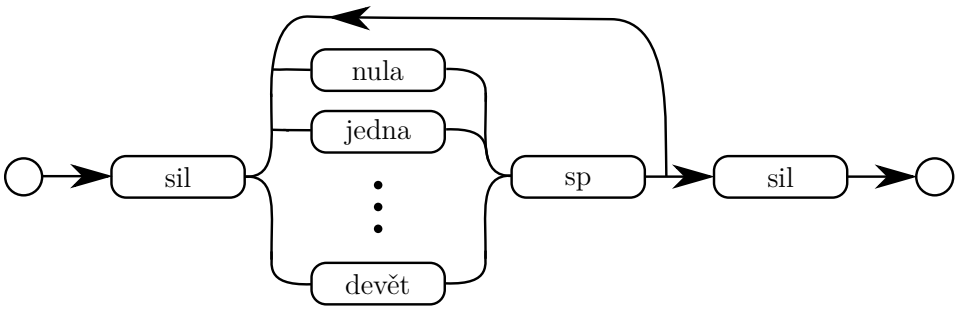
5.1 Popis rozpoznávače

5.1.1 Data pro rozpoznávání

Data, použitá pro příznaky použité k trénování i testování rozpoznávače, jsou z české části databáze SPEECON [13]. Tato databáze obsahuje promluvy pro různé mluvčí, různá prostředí a různé druhy záznamových zařízení. V této práci byla použita data z prostředí OFFICE s dospělými mluvčími nahrávanými zařízením s nízkým šumem. Trénink HMM probíhal na rozličných promluvách. Test pak pouze na datech, obsahující slova použitá v gramatice.

5.1.2 Gramatika

Použitou gramatikou pro rozpoznávač HMM jsou číslovky nula až devět s možnou pauzou mezi jednotlivými slovy. Jedná se tedy o rozpoznávač oddělených slov. Grafické znázornění gramatiky ukazuje obrázek 5.1.1. Na této gramatice byl realizován rozpoznávač s různými typy příznakových vektorů.



Obrázek 5.1.1 Grafické znázornění použité gramatiky.

5.1.3 Směsi

Při trénování HMM podle schématu 4.4.1, byly v krocích 17 - 18 několikrát přidávány směsi. Ve standardním případě bylo přidáváno po dvou, tedy postupně na 2, 4, 6, 8, 10 směsí. U příznakových vektorů, které dosáhly vysoké úspěšnosti, popřípadě zde byl potenciál dalšího růstu, bylo použito druhé řady směsí, 2, 4, 6, 12, 24. První řada směsí je v dalších částech označena mix10, druhá pak mix24.

5.2 Interpretace výsledků

5.2.1 Kritérium úspěšnosti

Jako hodnotící kritérium úspěšnosti HMM modelu na úrovni slov je použita standardní úspěšnost rozpoznávání (ACC), jež je definována vztahem 5.2.1.

$$ACC = \frac{N - S - D - I}{N} \cdot 100[\%] \tag{5.2.1}$$

N je celkový počet slov v testovací množině a S, D a I jsou počty nahrazených, smazaných a vložených slov. Nahrazené znamená, že rozpoznávač detekoval slovo, ale rozpoznal ho špatně. Smazaná jsou slova, která rozpoznávač vůbec nedetekuje a vložená slova jsou případy, kdy rozpoznávač detekuje slovo tam, kde není.

Alternativním hodnotícím kritériem je Word Error Rate (WER). Tato hodnota udává chybovost a je definována podle vztahu 5.2.2.

$$WER = 100 - ACC[\%] \tag{5.2.2}$$

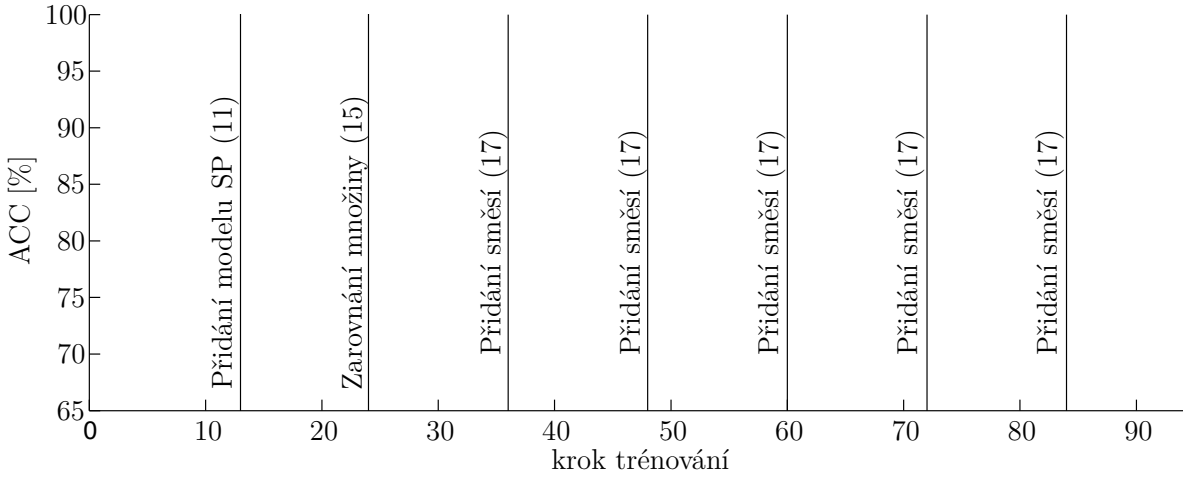
Relativní změnu chybovosti udává Word Error Rate Reduction (WERR), daný vztahem 5.2.3.

$$WERR = \frac{WER_{ref} - WER_{new}}{WER_{ref}} \cdot 100[\%] \tag{5.2.3}$$

WER_{ref} je chybovost referenčního modelu a WER_{new} je chybovost testovaného modelu.

5.2.2 Popis struktury grafů

K plnému porozumění výsledkům je třeba vysvětlit strukturu používaných grafů. Ta je zobrazena na obrázku 5.2.1. Jednotlivé svislé linky určují změny modelu v rámci schématu 4.4.1. Jak už bylo zmíněno, po každé změně modelu dochází k deseti přetrénování a pro další změnu se vychází z modelu s nejvyšší úspěšností v poslední trénovací množině. Z důvodu přehlednosti se v grafu nezobrazují hodnoty, následující po maximální hodnotě aktuálního cyklu.



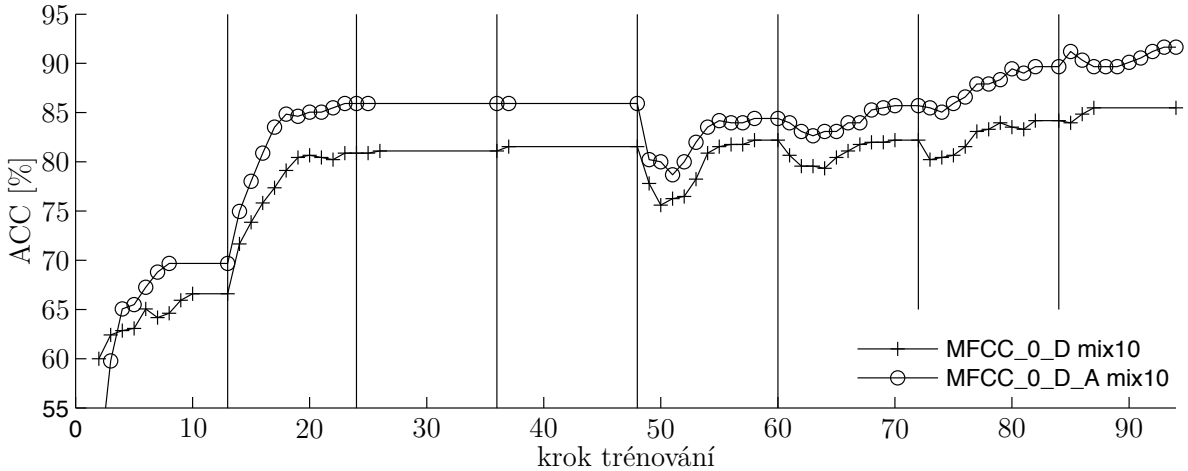
Obrázek 5.2.1 Popis struktury grafu s odkazy na schéma 4.4.1.

5.3 Dosažené výsledky

V rámci experimentů proběhlo několik řad testů, které používaly různé typy příznakových vektorů TANDEM architektury.

5.3.1 Výsledky pro základní mel-kepstrální koeficienty

Nejjednodušší příznakové vektory byly tvořeny pouze melovskými kepstrálními koeficienty s postupně přidanými delta a delta-delta parametry. Tyto typy příznaků byly použity k získání reference k ostatním typům příznakových vektorů.



Obrázek 5.3.1 Úspěšnosti melovských kepstrálních koeficientů.

Jak je vidět v grafu 5.3.1, nejvyšší úspěšnosti dosáhl příznakový vektor MFCC_0_D_A, to ukazuje na příznaky s velkou užitečnou informací a dobrou dekorelací. Naopak MFCC

pouze se statickými příznaky dosáhl úspěšnosti pouze 58.24 %, což je způsobeno absencí kontextu. Hodnoty dosažených ACC pro MFCC příznaky zobrazuje tabulka 5.3.1.

typ příznakového vektoru	ACC [%]
MFCC_0_D_A mix10	91.65
MFCC_0_D mix10	85.49
MFCC_0 mix10	58.24

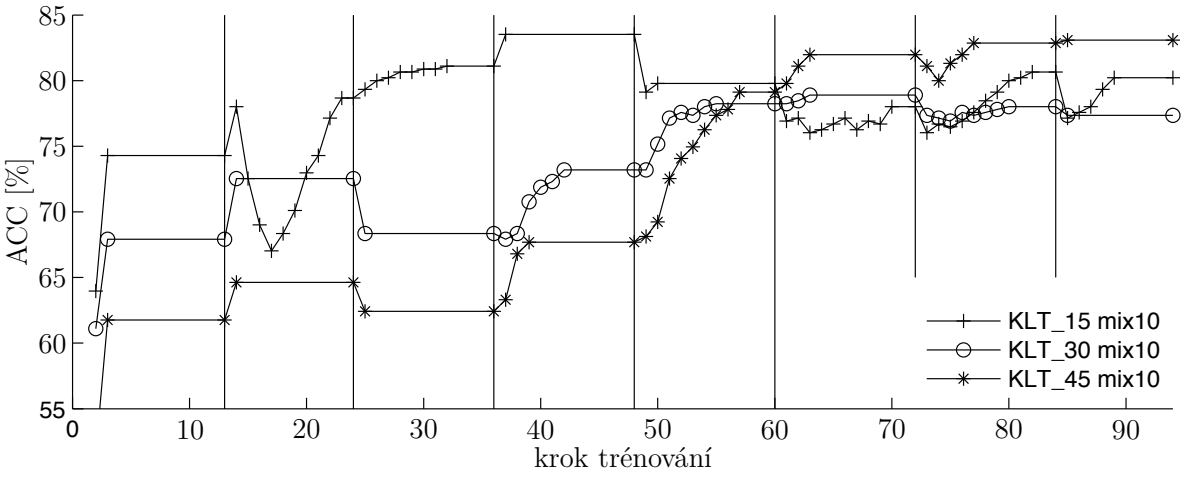
Tabulka 5.3.1 Tabulka úspěšností melovských kepstrálních koeficientů.

Podle očekávání nejlepší úspěšnosti dosáhl příznakový vektor složený ze statických, delta a delta-delta příznaků. Tento vektor obsahuje nejvíce informace. Naopak příznakový vektor pouze se statickými příznaky obsahuje příliš málo informace, než aby dosáhl vyšší úspěšnosti. Absence kontextu dělá tento typ příznakového vektoru pro další zpracování nepoužitelným.

5.3.2 Výsledky pro KLT aposteriorní pravděpodobnosti

Jako další byly testovány příznakové vektory KLT logaritmovaných aposteriorních pravděpodobností s různým omezením dimenze.

V grafu 5.3.2 jsou vidět úspěšnosti pro logaritmované aposteriorní pravděpodobnosti po KLT. Zajímavé je, že zatímco u MFCC_0_D a MFCC_0_D_A došlo k prudkému nárůstu úspěšností po přidání modelu SP, u příznaků KLT úspěšnost roste významněji až po přidání směsí, ovšem tento nárůst se brzy zastaví. Je třeba poznamenat, že při použití KLT příznaků dojde mnohem snadněji k přetrénování HMM. Častý je také jev, kdy po úpravě modelu úspěšnost poklesne, aby se po několika přetrénování zlepšila. Hodnoty dosažených ACC pro KLT příznaky s různě omezenou dimenzí zobrazuje tabulka 5.3.2.



Obrázek 5.3.2 Úspěšnosti KLT příznaků.

typ příznakového vektoru	ACC [%]
KLT_15 mix10	83.52
KLT_45 mix10	83.08
KLT_30 mix10	78.90

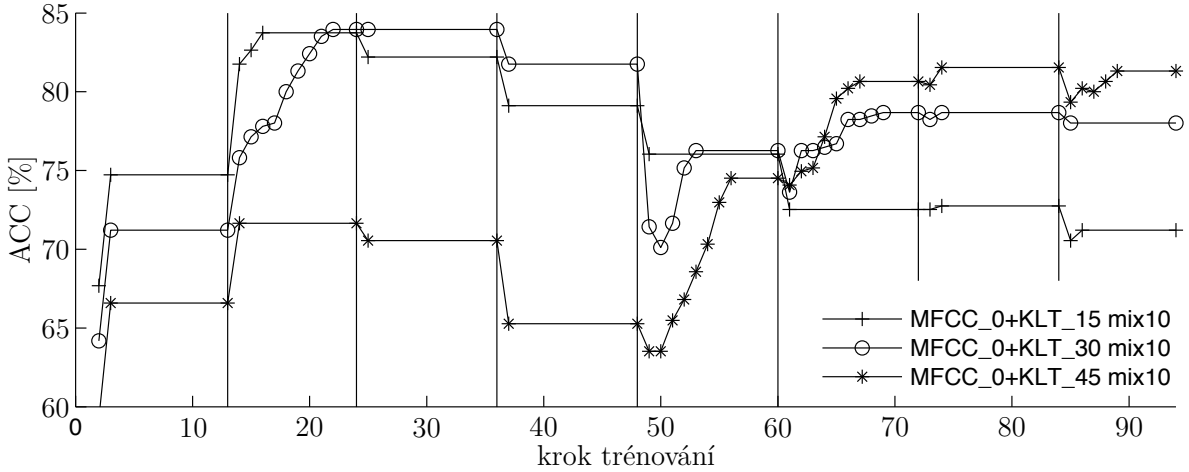
Tabulka 5.3.2 Tabulka úspěšností KLT aposteriorních pravděpodobností.

Metoda hlavních komponent koncentruje důležité informace na nízkých dimenzích, z toho důvodu vektor KLT_15 obsahuje nejvíce užitečné informace a dosahuje tedy nejvyšších úspěšností. Nicméně v grafu je vidět, že po přidání více směsí začne jeho ACC klesat. Pro další použití je tedy výhodnější použít jiné KLT vektory, které samozřejmě obsahují více nadbytečné informace, ale také přidávají druhotné informace, které lze získat právě přidáním směsí.

5.3.3 Výsledky pro příznakové vektory TANDEM architektury

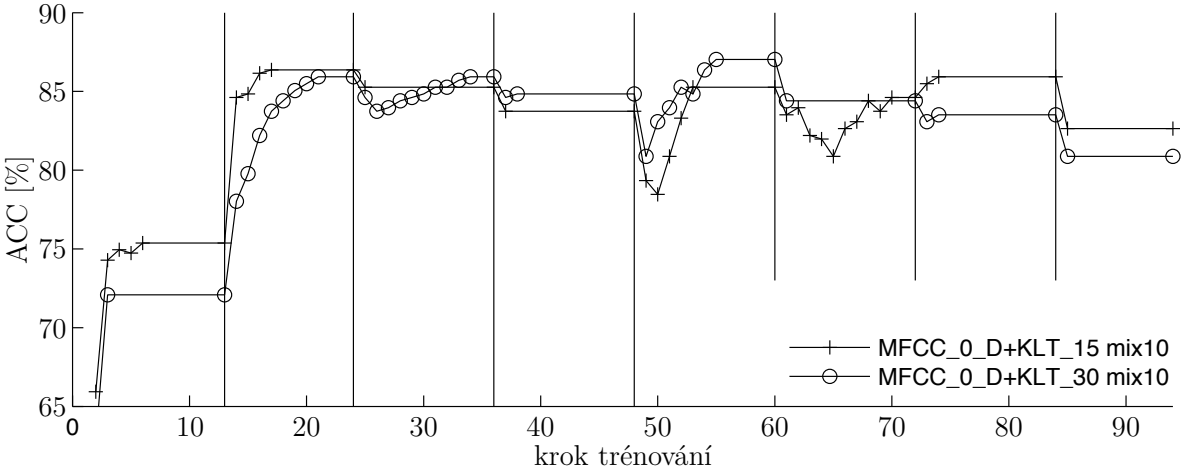
Je mnoho způsobů kombinace kepstrálních a KLT koeficientů. Jedním z nich je například kombinace statických kepstrálních příznaků a KLT příznaků. Z tabulky 5.3.1 je patrné, že statická kepstra nenesou příliš informace a jak ukazuje graf 5.3.3, jejich spojení s KLT nemá na úspěšnosti velký vliv. Většinu informace zde nesou KLT.

Po přidání delta koeficientů je v příznakovém vektoru víc informace dodané od kepstrálních příznaků. To se projeví tak, že úspěšnost je vyšší už před přidáním směsí. Výsledky zobrazuje graf 5.3.4. Není výhodné kompenzovat nedostatek informace v kepstrálních koeficientech použitím více příznaků z KLT. V případě použití vektoru KLT_45 k MFCC_0_D, je maximální úspěšnost 72 %.



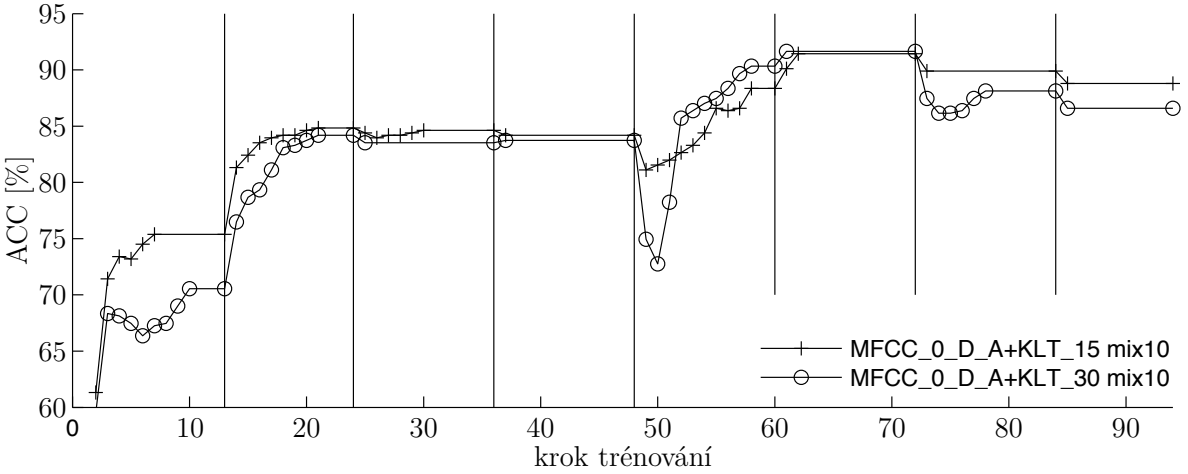
Obrázek 5.3.3 Úspěšnosti MFCC_0 + KLT příznaků.

Příznakový vektor kepstrálních koeficientů s delta i delta-delta koeficienty již obsahuje dostatek informace, aby se kepstrální koeficienty výrazněji projevily i po přidání směsí,



Obrázek 5.3.4 Úspěšnosti MFCC_0_D + KLT příznaků.

jak je vidět v grafu 5.3.5. Použití delšího vektoru KLT příznaků už je silně kontraproduktivní. Maximální dosažená úspěšnost v tom případě činila pouhých 70 %. Tabulka 5.3.3 obsahuje maximální úspěšnosti zmíněných typů příznakových vektorů.



Obrázek 5.3.5 Úspěšnosti MFCC_0_D_A + KLT příznaků.

Na prvních pozicích tabulky 5.3.3 se umístily příznakové vektory s MFCC_0_D_A, což značí velkou informační váhu těchto příznaků. Naopak příznakové vektory s KLT_45 dopadly nejhůře, což potvrzuje princip KLT.

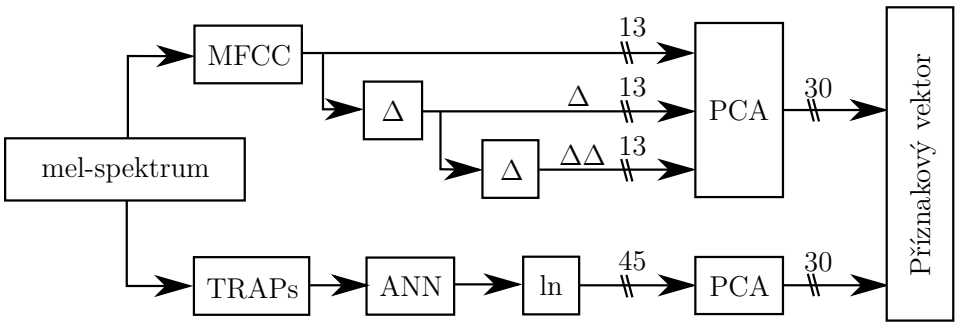
typ příznakového vektoru	ACC [%]
MFCC_0_D_A + KLT_30 mix10	91.65
MFCC_0_D_A + KLT_15 mix10	91.43
MFCC_0_D + KLT_30 mix10	87.03
MFCC_0_D + KLT_15 mix10	86.37
MFCC_0 + KLT_30 mix10	83.96
MFCC_0 + KLT_15 mix10	83.74
MFCC_0 + KLT_45 mix10	81.54
MFCC_0_D + KLT_45 mix10	72.97
MFCC_0_D_A + KLT_45 mix10	70.77

Tabulka 5.3.3 Tabulka úspěšností MFCC + KLT příznaků

5.3.4 Vliv KLT v různých stupních TANDEM architektury

Další typy příznakových vektorů TANDEM architektury využívají transformovaných kepstrálních koeficientů, čímž se snaží zlepšit jejich informační hodnotu.

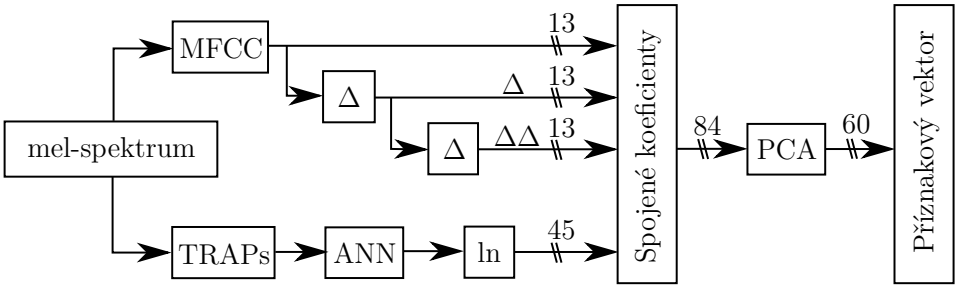
1. **KLT na MFCC_0_D_A + KLT aposteriorních pravděpodobností**
   Jednou z možností je transformace statických, dynamických i akceleračních kepstrálních koeficientů dohromady. Diferenční a akcelerační data jsou částečně typově odlišná od statických kepstrálních koeficientů, proto zde vzniká určitý prostor pro dekorelaci a redukci dimenze. Pak se spojí s KLT aposteriorními pravděpodobnostmi tak, jak ukazuje obrázek 5.3.6. Dimenze obou KLT příznaků jsou zvolené na prvních 30 komponent. Výsledkem je příznakový vektor KLT(MFCC_0_D_A)_30 + KLT_30.



Obrázek 5.3.6 Znázornění výpočtu příznaků KLT(MFCC_0_D_A) + KLT.

2. **KLT na MFCC_0_D_A s aposteriorními pravděpodobnostmi**
   Další možností je spojit kepstrální koeficienty s logaritmy aposteriorních pravděpodobností a provést KLT na celém takto vzniklém vektoru příznaků. Postup naznačuje obrázek 5.3.7. Vzhledem k faktu, že aposteriorní pravděpodobnosti jsou typově naprosto odlišné od kepstrálních příznaků, redukce dimenze zde může být

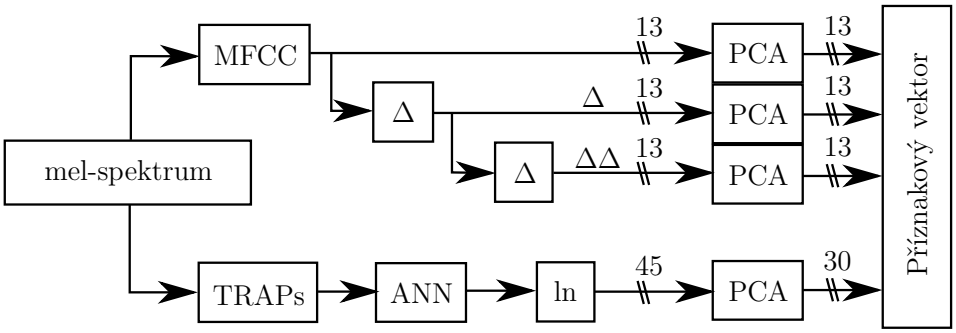
rozsáhlejší. V práci je použito 60 hlavních komponent. Vektor příznaků pak je KLT(MFCC_0_D_A + POST)_60.



Obrázek 5.3.7 Znázornění výpočtu příznaků KLT(MFCC_0_D_A + POST).

3. **KLT postupně na MFCC_0_D_A a KLT aposteriorních pravděpodobností**

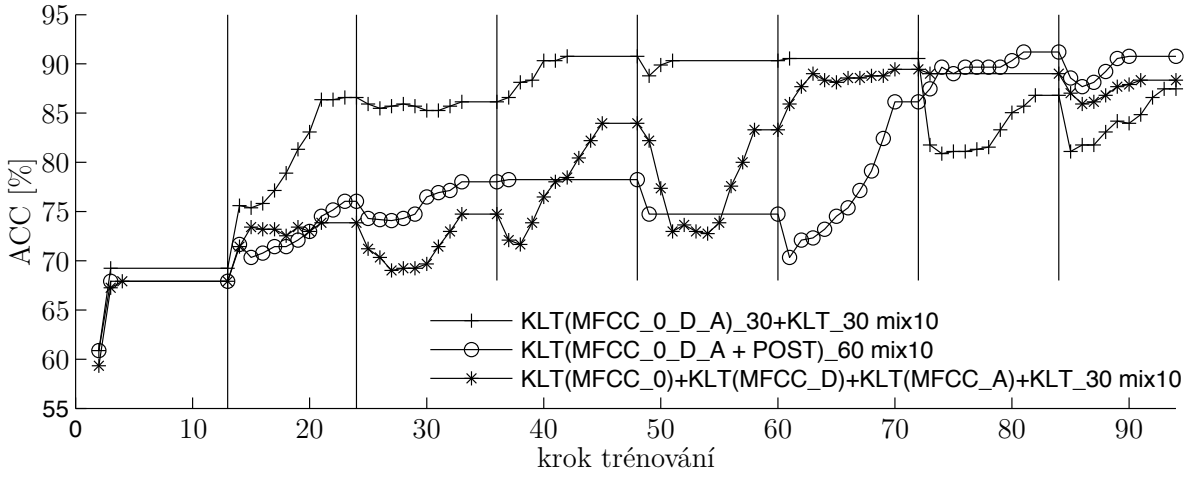
   Další z mnoha možností, je postupná aplikace KLT na statická kepstra, dynamická kepstra a akcelerační kepstra. Tyto tři se nakonec spojí s KLT logaritmem aposteriorních pravděpodobností. Vzhledem ke stejnému typu dat v každé transformaci je třeba zvážit, zda a jak redukovat výsledné dimenze. V této práci je redukována pouze dimenze KLT aposteriorních pravděpodobností na 30. Vzniká tak příznakový vektor
   KLT(MFCC_0) + KLT(MFCC_D) + KLT(MFCC_A) + KLT_30, podle schématu 5.3.8.



Obrázek 5.3.8 KLT(MFCC_0) + KLT(MFCC_D) + KLT(MFCC_A) + KLT_30 .

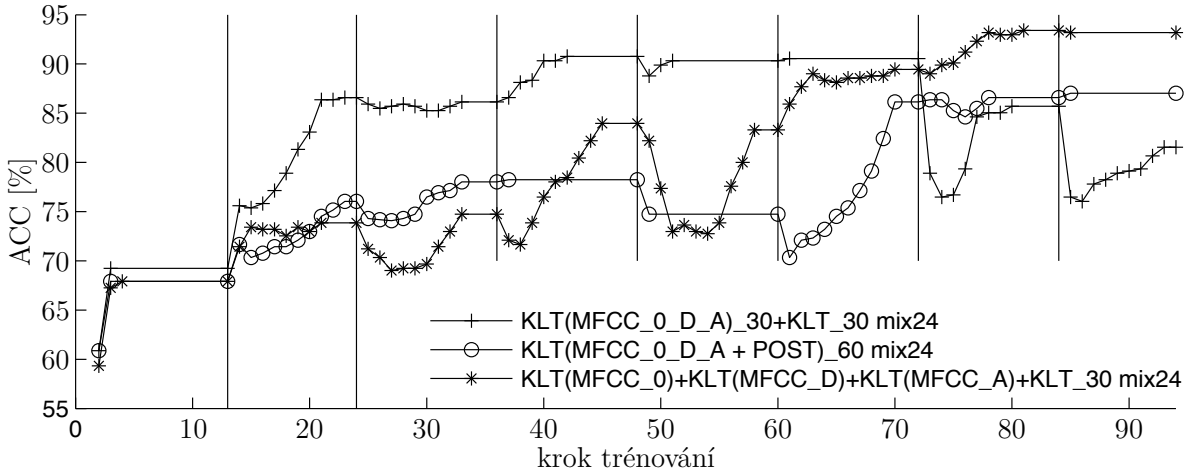
Průběžné úspěšnosti těchto tří typů příznakových vektorů zobrazuje graf 5.3.9. Jako nejperspektivnější se jeví příznakový vektor 3, který sice nedosáhl nejvyšší úspěšnosti, ovšem během každého cyklu přetrénování, až do přidání na 8 směsí, dosáhl zlepšení oproti minulému cyklu.

U těchto příznakových vektorů typu 2 a 3 je vidět, že při přidávání směsí dochází ke zlepšení až do 8 směsí. V tu chvíli se nabízí místo 8 směsí přidat více, aby se variabilita příznaků rozšířila i na méně významné členy. Graf 5.3.10 zobrazuje úspěšnosti druhého



Obrázek 5.3.9 Úspěšnosti hybridních příznakových vektorů.

typu řady směsí.



Obrázek 5.3.10 Úspěšnosti hybridních příznakových vektorů s vyšší řadou směsí.

Nejvyšší úspěšnosti těchto typů vektorů zobrazuje tabulka 5.3.4. Jak je vidět, přidání směsí může, ale nemusí přinést zlepšení. V případě 1. typu příznakového vektoru došlo ke zlepšení. WERR tohoto typu příznakového vektoru vůči příznakovému vektoru MFCC_0_D_A je 21.08 %. Příznakový vektor typu 2 se naopak zhoršil. U vektoru typu 2 nedošlo ke změně ACC, neboť maxima bylo dosaženo již před posledníma dvěma cykly. Naopak při zvýšení počtu směsí došlo k ještě většímu poklesu úspěšnosti.

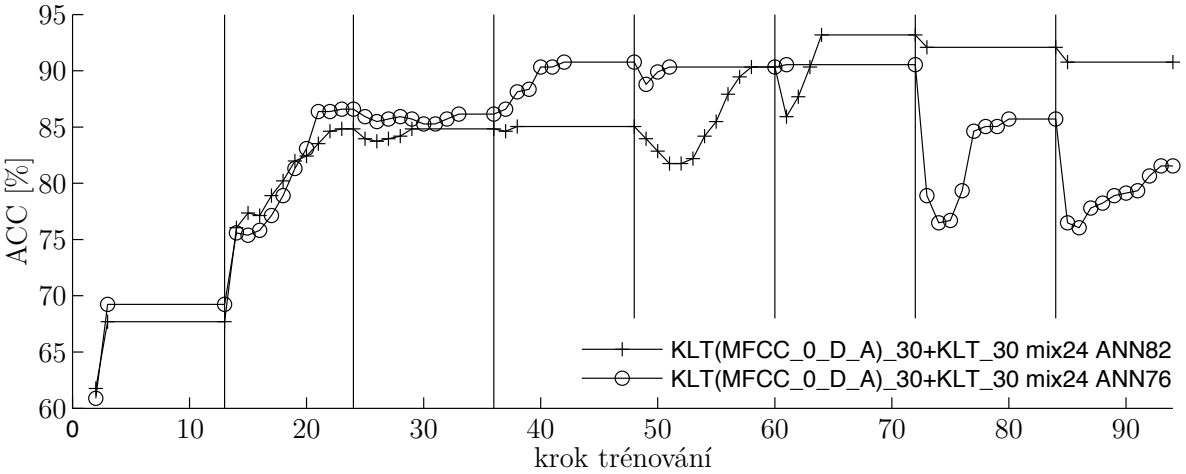
typ příznakového vektoru	ACC [%]
KLT(MFCC_0)+KLT(MFCC_D)+KLT(MFCC_A)+KLT_30 mix24	93.41
KLT(MFCC_0_D_A + POST)_60 mix10	91.21
KLT(MFCC_0_D_A)_30+KLT_30 mix10	90.77
KLT(MFCC_0_D_A)_30+KLT_30 mix24	90.77
KLT(MFCC_0)+KLT(MFCC_D)+KLT(MFCC_A)+KLT_30 mix10	89.45
KLT(MFCC_0_D_A + POST)_60 mix24	87.03

Tabulka 5.3.4 Tabulka úspěšností hybridních typů příznakových vektorů

5.3.5 Výsledky pro lepší aposteriorní pravděpodobnosti

Jak již bylo řečeno, aposteriorní pravděpodobnosti byly vypočteny v rámci jiné práce, která byla řešena paralelně s touto. Použité aposteriorní pravděpodobnosti tedy v průběhu řešení zastaraly, neboť bylo dosaženo vyšší úspěšnosti mapování příznaků pomocí ANN. Zatímco do této doby používané aposteriorní pravděpodobnosti byly vypočteny sítí s úspěšností klasifikace 76 %, nové příznaky vypočítala síť s úspěšností 82 %. V následujících grafech jsou průběhy s různými úspěšnostmi v ANN rozlišeny jako ANN76, pro staré a ANN82 pro nové aposteriorní pravděpodobnosti.

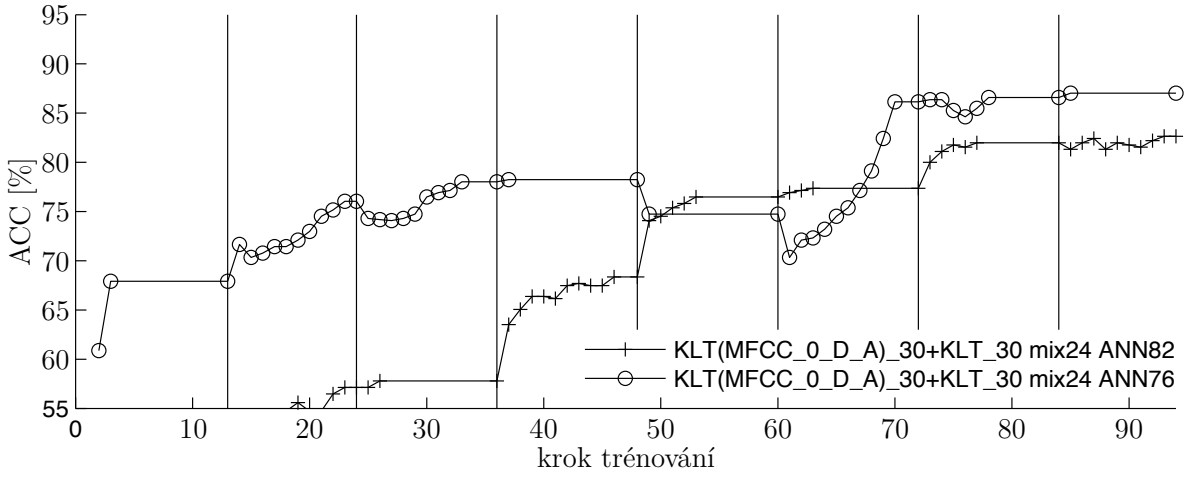
Graf 5.3.11 ukazuje, že lepší aposteriorní pravděpodobnosti zabránily markantnímu poklesu úspěšnosti při přidání směsí na 12 a 24.



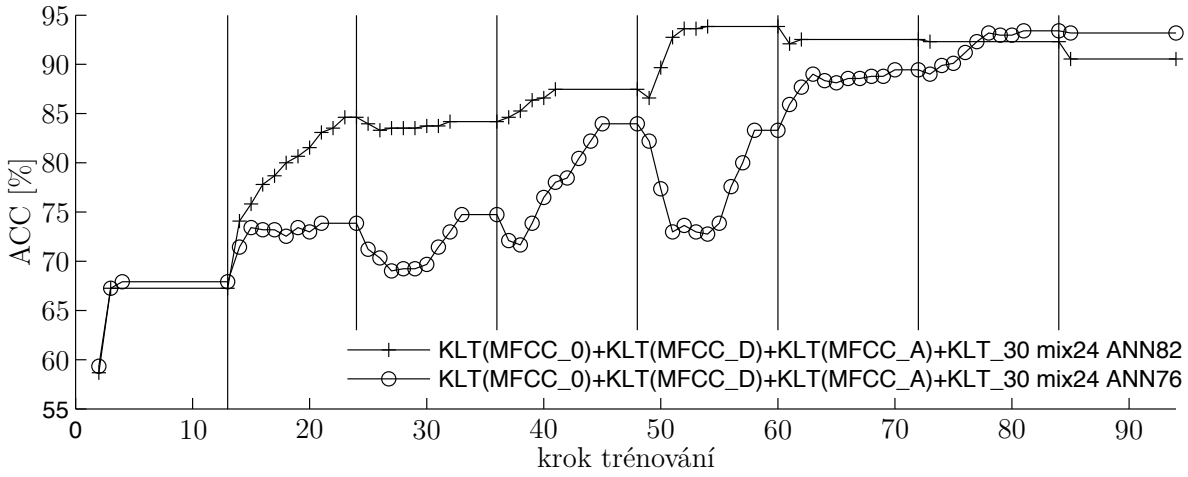
Obrázek 5.3.11 Porovnání úspěšností vektoru 1 pro různé úspěšnosti mapování ANN.

Pro příznakový vektor typu 2 sice dochází k postupnému nárůstu úspěšností, což zobrazuje graf 5.3.12, ale celková úspěšnost je nižší.

V grafu 5.3.13 je vidět, že pro příznakový vektor typu 3 dojde k navýšení maximální ACC.



Obrázek 5.3.12 Porovnání úspěšností vektoru 2 pro různé úspěšnosti mapování ANN.



Obrázek 5.3.13 Porovnání úspěšností vektoru 3 pro různé úspěšnosti mapování ANN.

Hlavní přínos aposteriorních pravděpodobností vypočtených s lepší úspěšností je ve *vyhlazení* průběhů ACC. V grafech už nedochází k velkým propadům úspěšností. Tabulka 5.3.5 ukazuje maximální ACC pro zmíněné typy příznakových vektorů.

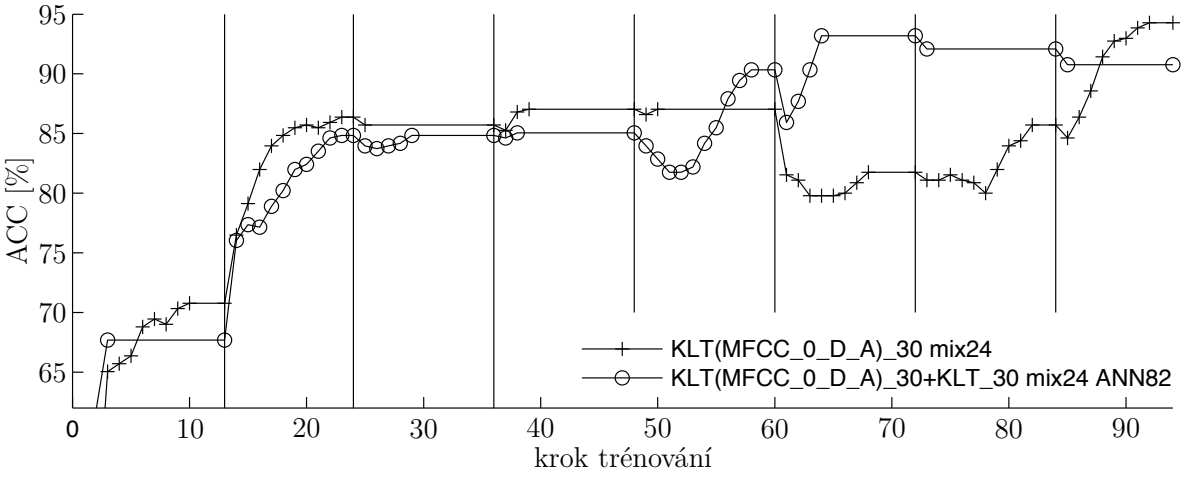
Jak je vidět, kromě *vyhlazení* průběhu ACC dojde ke zlepšení maximálních úspěšností. Nejlépe vychází příznakový vektor typu 3. Při porovnání výsledků s ANN76 je WERR tohoto typu příznaků 6.68 %. Nejhůře pak vektor typu 2 se vzájemným WERR -33.85 %, ovšem vzhledem k rostoucí tendenci je možné, že pro více směsí by se úspěšnost tohoto typu příznakového vektoru zlepšila.

typ příznakového vektoru	ACC [%]
KLT(MFCC_0)+KLT(MFCC_D)+KLT(MFCC_A) +KLT_30 mix24 ANN82	93.85
KLT(MFCC_0)+KLT(MFCC_D)+KLT(MFCC_A) +KLT_30 mix24 ANN76	93.41
KLT(MFCC_0_D_A)_30+KLT_30 mix24 ANN82	93.19
KLT(MFCC_0_D_A)_30+KLT_30 mix24 ANN76	90.77
KLT(MFCC_0_D_A + POST)_60 mix24 ANN76	87.03
KLT(MFCC_0_D_A + POST)_60 mix24 ANN82	82.64

Tabulka 5.3.5 Tabulka úspěšností pro příznaky vycházející z ANN s vyšší ACC.

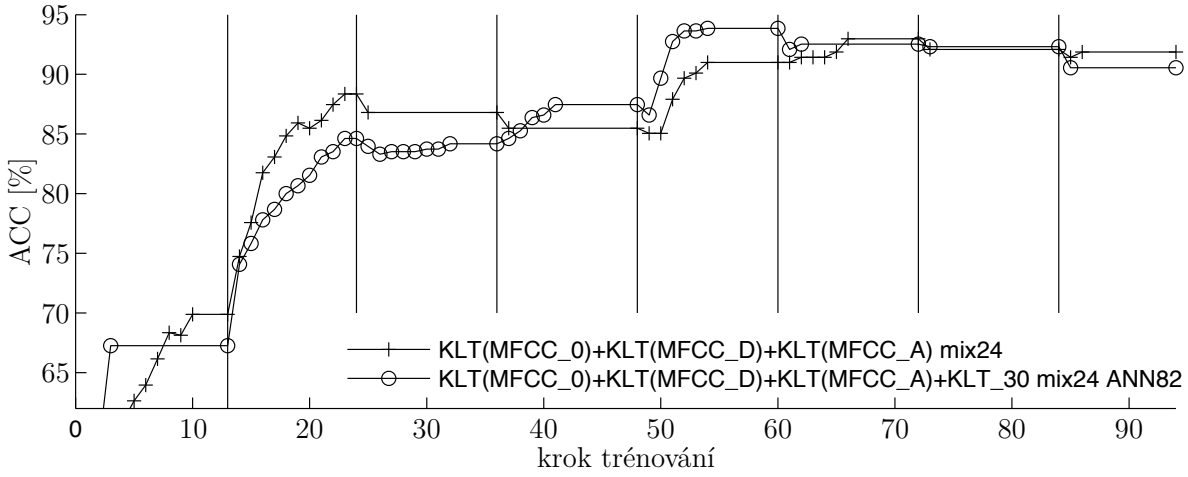
5.3.6 Úspěšnost KLT kepster jako samostatných příznaků

Při použití KLT na kepstrální koeficienty se nabízí otázka, jaký vliv má KLT aposteriorních pravděpodobností na celkovou úspěšnost. Průběh úspěšností pro příznakový vektor typu 1 zobrazuje graf 5.3.14. Úspěšnost bez KLT aposteriorních pravděpodobností je ve finále vyšší a je pravděpodobné, že s přidáním dalších směsí by ještě rostla.



Obrázek 5.3.14 Úspěšnosti vektoru typu 1 s/bez KLT aposteriorních pravděpodobností.

Vliv absence KLT aposteriorních pravděpodobností na příznakový vektor typu 3, je zobrazen v grafu 5.3.15. Pro příznakový vektor typu 3 je lepší přidat ke KLT kepstrálním koeficientům i KTL aposteriorní pravděpodobnosti. Maximální ACC pro tuto část zobrazuje tabulka 5.3.6.



Obrázek 5.3.15 Úspěšnosti vektoru typu 3 s/bez KLT aposteriorních pravděpodobností.

typ příznakového vektoru	ACC [%]
KLT(MFCC_0_D_A)_30 mix24	94.29
KLT(MFCC_0)+KLT(MFCC_D)+KLT(MFCC_A) +KLT_30 mix24 ANN82	93.85
KLT(MFCC_0_D_A)_30+KLT_30 mix24 ANN82	93.19
KLT(MFCC_0)+KLT(MFCC_D)+KLT(MFCC_A) mix24	92.97

Tabulka 5.3.6 Tabulka úspěšností s/bez KLT aposteriorních pravděpodobností.

Ve vektoru typu 3 hrají KLT aposteriorní pravděpodobnosti důležitou roli. WERR tohoto typu příznakového vektoru s KLT aposteriorních pravděpodobností v porovnání s vektorem bez KLT koeficientů je 12.52 %. Naproti tomu ve vektoru typu 1, lze dosáhnout vysoké úspěšnosti i bez KLT aposteriorních pravděpodobností. WERR tohoto vektoru s KLT vůči vektoru bez KLT je -19.26 %.

6 Závěr

Cílem práce bylo implementovat rozpoznávač řeči na bázi TANDEM architektury. Byla vytvořena základní verze rozpoznávače s TANDEM architekturou s použitím základního postupu trénování akustických modelů na bázi HMM pro monofóny. Rozpoznávače spojité řeči jsou velmi náročné na implementaci a znalosti potřebné pro jejich realizaci dalece přesahují rozsah bakalářské práce. Z tohoto důvodu je v rámci této práce realizován rozpoznávač izolovaných slov, konkrétně rozpoznávač číslovek s možným opakováním. Na této úloze se dají demonstrovat základní principy, které se používají i v úloze rozpoznávání spojité řeči a lze očekávat jejich daleko větší přínos.

Rozpoznávač popisovaný v této práci je založen na principu skrytých Markovových modelů a je implementován pomocí balíčku nástrojů HTK. Jedná se o velmi rozsáhlý balíček nástrojů pro práci s HMM, jež umožňuje realizaci parametrizace, trénování, rozpoznávání, vyhodnocování výsledků, modifikace modelů a mnoho dalšího. Pro vytváření příznakových vektorů TANDEM architektury se s balíčkem HTK nevystačí, proto byl použit balíček nástrojů pfile_utils, který umožňuje pokročilejší práci s příznakovými vektory, a nástroj feacat, sloužící ke konverzi formátů souborů s příznakovými vektory. Nástroj HTK pro trénování HMM umožňuje paralelizaci výpočtů, což může vést ke zrychlení výpočtů při práci s velkými databázemi signálů.

TANDEM architektura je specifikována používáním příznakových vektorů, spojených z více základních typů příznaků. V této práci byly použity melovské kepstrální koeficienty v kombinaci s aposteriorními pravděpodobnostmi výskytu fonémů v jednotlivých segmentech signálu. MFCC příznaky byly vypočteny pomocí nástroje HCopy. Aposteriorní pravděpodobnosti byly získány z TRAPs příznaků nelineárním mapováním pomocí ANN. Jedná se o zcela samostatnou úlohu fonémového rozpoznávání řeči. S používáním umělých neuronových sítí jsem se seznámil v rámci individuálního projektu. Ve finálním zpracování bakalářské práce se tímto problémem detailněji zabýval *Jiří Fiala*. Aposteriorní pravděpodobnosti použité v této práci byly získány z výstupu MLP sítí implementovaných v jeho práci. Tyto pravděpodobnosti musí mít před použitím v HMM gaussovské rozložení, kterého lze dosáhnout zlogaritmováním, a musí být dekorelovány, což zajišťuje KLT, které bylo aplikováno i na MFCC.

V práci bylo testováno několik typů příznakových vektorů TANDEM architektury. Nejlepší úspěšnosti 94.29 % dosáhl příznakový vektor kepstrálních příznaků s delta a

delta-delta koeficienty po KLT při použití HMM s postupným přidáváním směsí na 2, 4, 6, 12 a 24. Další v pořadí byl příznakový vektor, jež se skládá postupně z KLT statických, dynamických a akceleračních kepstrálních koeficientů doplněných o KLT aposteriorních pravděpodobností. Tento typ příznakového vektoru dosáhl úspěšnosti 93.85 %.

Pro dosažení co nejlepších výsledků je vedle správné volby typů použitých příznaků potřeba optimálně zvolit dimenzi KLT příznaků. Vzhledem k počtu možných kombinací různých příznaků nebylo možné v této práci provést experimenty pro všechny možné kombinace. Další navazující práce v této oblasti, bude zaměřena především na optimalizace trénovacích kroků, použití trifónů místo monofónů, hledání optimálního příznakového vektoru a zejména pak na použití v úloze rozpoznávání spojité řeči s velkým slovníkem.

Literatura

[1] Josef Psutka et al. *Mluvíme s počítačem česky*. Academia, 2006.

[2] Jana Koucká. "Rozpoznávání řeči s příznaky na bázi TRAP trajektorií". MA thesis. ČVUT v Praze, 2013.

[3] Jan Uhlíř et al. *Technologie hlasových komunikací*. Praha: Nakladatelství ČVUT, 2007.

[4] Jana Tučková. *Vybrané aplikace umělých neuronových sítí při zpracování signálů*. Nakladatelství ČVUT, 2009.

[5] Partha Lal and Simon King. "Cross-Lingual Automatic Speech Recognition Using Tandem Features". In: *Audio, Speech, and Language Processing, IEEE Transactions on* (2013), pp. 2506–2515.

[6] Daniel Ellis, Rita Singh, and Sunil Sivadas. "Tandem Acoustic Modeling In Large-Vocabulary Recognition". In: *in Proc. ICASSP-2001*. 2001, pp. 517–520.

[7] Qifeng Zhu et al. "Tandem Connectionist Feature Extraction for Conversational Speech Recognition". In: *Proceedings of the First International Conference on Machine Learning for Multimodal Interaction*. Martigny, Switzerland, 2005.

[8] Daniel P. W. Ellis and Manuel J. Reyes Gomez. *Investigations into Tandem Acoustic Modeling for the Aurora Task*. 2001.

[9] Oracle. *Sun Grid Engine*. 2000. URL: `http://www.oracle.com/us/products/tools/oracle-grid-engine-075549.html`.

[10] Steve Young. *Hidden Marcov Model Toolkit*. 1989. URL: `http://htk.eng.cam.ac.uk/`.

[11] *Quicknet*. URL: `http://www1.icsi.berkeley.edu/Speech/icsi-speech-tools.html`.

[12] Jiří Fiala. *Implementace fonémového rozpoznávače s nástoji TNetu Toolkitu*. BA thesis. 2014.

[13] *SPEECON*. URL: `http://speech.fit.vutbr.cz/cs/projects/speecon`.

[14] Xian Tang. "Hybrid Hidden Markov Model and Artificial Neural Network for Automatic Speech Recognition". In: *Circuits, Communications and Systems, 2009. PACCS '09. Pacific-Asia Conference on*. 2009, pp. 682–685.

[15] Amlan Kundu and Aruna Bayya. "Speech recognition using hybrid hidden markov model and NN classifier". In: *International Journal of Speech Technology* (1998), pp. 227–240.