Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Katedra kybernetiky

BAKALÁŘSKÁ PRÁCE

Alternativní tvorba vícesložkových akustických modelů

Plzeň, 2014 Milan Lavička

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Mgr. Josefu V. Psutkovi, Ph.D. za cenné rady, připomínky a spoustu trpělivosti při vypracovávání bakalářské práce.

Prohlášení

Předkládám tímto k posouzení a obhajobě bakalářskou práci zpracovanou na závěr studia na Fakultě aplikovaných věd Západočeské univerzity v Plzni.

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím odborné literatury a pramenů, jejichž úplný seznam je její součástí.

V Plzni dne 22.května 2014 .....

podpis

Abstrakt

Tato práce se zabývá výzkumem a tvorbou akustického modelu za užití alternativních metod. Součástí výzkumu je vliv počtu složek Gaussovské směsi na kvalitě akustického modelu. Dále jsou zde popsány procesy získávání trénovacích dat a výpočet parametrů pro akustický model a ze získaných parametrů sestavení vlastního modelu. Závěr práce porovnává výsledky rozpoznávání modelu HTK na bázi EM algoritmu a modelu sestaveným z parametrů alternativně použité metody k-means.

Klíčová slova: k-means, akustický model, HTK, rozpoznávání řeči

Abstract

This thesis deals with the research and creation of an acoustic model with an alternative method. Part of the research is the influence of the number of components of the Gaussian mixture model on the quality of the sound. It further describes the process of obtaining training data and calculating the parameters for the acoustic model and the obtained parameters to create a model. The conclusion of the work compares the results of the HTK recognition model based on the EM algorithm and model parameters, which is made from the selected alternative methods of k-means.

Keywords: k-means, acoustic model, HTK, speech recognition

Obsah

1. Úvod

Mluvená řeč je nejpřirozenější způsob dorozumívání mezi lidmi. Komunikace mluveným slovem je rychlá a obsahuje mnoho informací jak přímo o sdělení, tak například i o řečníkovi. Může poukazovat na důležitost informací, odrážet náladu či promítat osobní rysy řečníka. Problematika komunikace řečí se strojem se tedy jeví jako velmi perspektivní. Vzhledem k technickému pokroku, a tedy i novým možnostem, zaznamenávají řečové technologie obrovské pokroky.

Strojové porozumění a rozpoznávání řeči spadá do odvětví kybernetiky. Katedra kybernetiky na Západočeské univerzitě v Plzni se specializuje mimo jiné na řečové úlohy, takže jsem se rozhodl na tento směr navázat také.

V této práci je mým úkolem nastudovat stávající optimální tvorbu akustického modelu, pomocí alternativní metody vytvořit vlastní akustický model a oba následně vzájemně porovnat. Jako hlavním nástroj pro tvorbu modelu, trénování a rozpoznávání jsem si zvolil HTK (Hidden Markov Model Toolkit), jež používá při tvorbě modelu popis normálním rozdělením a pro hledání parametrů metodu EM (Expectation-Maximization). Alternativou tohoto popisu mi při tvorbě vlastního modelu bude rovnoměrné a nerovnoměrné binární dělení metodou k-means, zohledňující geometrické vlastnosti dat.

Z trénovacích dat vytvořím akustický model pomocí HTK, budu přidávat složky modelu a testovat, jaký má počet složek vliv na úspěšnost rozpoznávání. Po získání dostačujícího počtu složek natrénuji v HTK model a pomocí fonetických přepisů označím hranice jednotlivých fonémů a následně roztřídím. Roztřízená data rozdělím dynamicky nerovnoměrným a rovnoměrným dělením metodou k-means a spočtu parametry pro všechny složky. Ze získaných parametrů sestavím akustický model a provedu rozpoznávání na testovacích datech. Výsledky rozpoznávání poslouží k ohodnocení kvality modelů a ke vzájemnému porovnání.

2. Rozpoznávání řeči

Snahy o vytvoření automatu schopného rozpoznat plynulou mluvenou řeč trvají už přes šedesát let. Navzdory velkému pokroku však stále bezchybný automat nebyl vytvořen. Hlavními problémy při řešení této úlohy jsou:

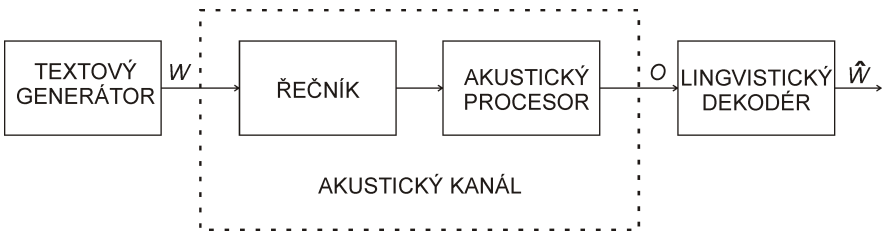
- Složitost úlohy a tedy i výpočetní náročnost, zejména díky prohledávání rozsáhlých slovníků.
- Variabilita tématu promluvy - český jazyk je velmi bohatý. Pokud bychom chtěli popsat všechna slova, dostali bychom obrovský slovník, jehož prohledávání v reálném čase by bylo nemožné.
- Variabilita řečníka - každý člověk mluví jinak a nikdo nezopakuje jedno slovo dvakrát naprosto stejně.
- V reálných úlohách se zpravidla řečník nevyskytuje v naprosto tichém prostředí, tudíž bývají zvukové stopy zašuměny různými nežádoucími elementy.
- Ve spontánní mluvené řeči užívání nespisovných variant slov nebo výskyt neřečových událostí, například váhání, kašlání, zadrhávání, ...

K řešení úlohy rozpoznávání vznikly dva přístupy. Klasifikátory na principu porovnání se vzory (template matching) a klasifikátory s využitím statistických metod. Porovnávací metody pracují zpravidla na úrovni slov a používaly se spíše v minulosti. V tomto přístupu se ukázala jako hlavní překážka rozdílná délka slov a variabilita řečníka. Pro porovnávání se využívalo metod dynamického programování. Hlavní rozdíl mezi slovy byl v rozdílné délce trvání fonémů, jež dokázala vystihnout časově nelineární bortivá funkce (DTW - dynamic time warping). Tento přístup se dá využít spíše při rozpoznávání izolovaných slov nebo při diskrétním diktátu. Oproti tomu statistické metody využívají modelování slov pomocí tzv. skrytých Markovových modelů. Slova mohou být popsána jedním skrytým Markovovým modelem nebo "rozebrána"na subslovní jednotky jako jsou

slabiky, fonémy, trifony apod. Tyto dílčí jednotky jsou pak modelovány samostatně a následně řetězeny. Při rozpoznávání neznámé promluvy je potom vybrána ta posloupnost subslovních jednotek, která nabývá největší aposteriorní pravděpodobnosti. Problém příliš velkých slovníků se řeší postavením malého slovníku obecné řeči, k němuž se dále podle potřeby přidávají moduly se slovy pro konkrétní téma. Tímto způsobem se prohledávané slovníky výrazně zredukují a nároky klesají. Zpravidla se hovoří na nějaké téma, tzn. pokud rozpoznávám promluvu kuchaře o vaření, lze předpokládat, že se zde nebudou vyskytovat slova z oblasti sportu, elektrotechniky apod. Na problematiku variability mluvené řeči a řečníků obecně navazují systémy na řečníku závislé, které jsou uzpůsobeny hlasu jednotlivce nebo malé skupině řečníků. A potom také systémy na řečníku nezávislé, které jsou sice obecnější, ale jejich úspěšnost rozpoznání je o něco nižší.

2.1 Statistický přístup k rozpoznávání řeči

Mějme posloupnost N slov $W = w_1 w_2 ... w_N$. V akustickém kanálu tuto posloupnost převede řečník na řeč a akustický procesor převede zvukový signál na posloupnost vektorů příznaků $O = o_1 o_2 ... o_N$. Soubor vektorů je dále zpracován lingvistickým dekodérem, který se snaží nalézt řešení viz. obr. 1.



Obrázek 2.1: Blokové schéma statistického přístupu rozpoznání řeči

Jedná se o dekódování s maximální aposteriorní pravděpodobností, tedy lingvistický dekodér se snaží maximalizovat podmíněnou pravděpodobnost $P(W|O)$ a najít tak nejpravděpodobnější řešení. Slovy: Hledáme takovou posloupnost W, která nejlépe odpovídá vektoru příznaků O.

Po aplikaci Bayesova pravidla

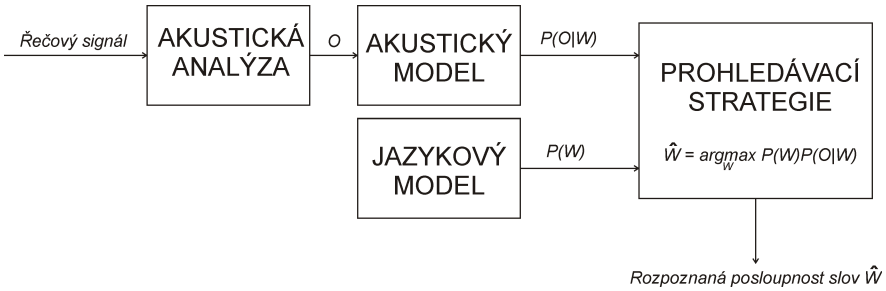
$$\hat{W} = \underset{W}{\operatorname{argmax}}\, P(W|O) = \underset{W}{\operatorname{argmax}}\, \frac{P(W)P(O|W)}{P(O)}$$

kde $P(W)$ je apriorní pravděpodobnost posloupnosti W, $P(O|W)$ označuje pravděpodobnost O při vstupním vektoru W a pravděpodobnosti příznakových vektorů $P(O)$. Hodnota $P(O)$ nezávisí na W a je konstantní, proto tento člen můžeme z rovnice vynechat. Rovnici lze tedy vyjádřit jako sdruženou pravděpodobnost

$$\hat{W} = \underset{W}{\operatorname{argmax}}\, P(W, O) = \underset{W}{\operatorname{argmax}}\, P(W)P(O|W)$$

Úlohu rozpoznávání řeči tedy můžeme dekomponovat na několik částí:

- Určení posloupnosti vektorů O provedením akustické analýzy
- Vytvoření jazykového modelu pro zajištění $P(W)$
- Vytvoření akustického modelu pro ocenění $P(O|W)$
- Hledání řešení prohledávací strategií



Obrázek 2.2: Blokové schéma rozpoznávání řeči po rozdělení na části

Nyní postupně proberu jednotlivé prvky a bloky v úloze statistického rozpoznávání řeči.

2.2 Akustická analýza

Abychom mohli mluvenou řeč rozpoznávat, je třeba nejprve převést zvukový signál na posloupnost vektorů. Takovému procesu říkáme parametrizace řečového signálu.

Existuje mnoho způsobů zpracování řeči, odvíjejících se hlavně od typu úlohy a s ní spjatých požadavků. Nejčastějšími úlohami zpracování akustického signálu jsou efektivní kódování pro přenos signálu, rozpoznání mluvené řeči, počítačová

syntéza řeči nebo například indentifikace a verifikace řečníka. Řečový signál s sebou nese velké množství informací. Při parametrizaci zpravidla vybíráme ty aspekty, které jsou pro danou úlohu potřebné a ostatní upostraníme, a proto je třeba metodu zpracování signálu zvolit vhodně.

U většiny parametrizačních metod se uvažuje pomalá změna vlastností akustického signálu v čase, takže se signál zpracovává pomocí metod krátkodobé analýzy. Při tomto procesu je signál rozdělen na mikrosegmenty o délce zpravidla 30 ms, které jsou dále jednotlivě převáděny na vektory parametrů.

2.3 Akustický model

Jak již bylo zmíněno v předchozí části, akustický model by měl co nejpřesněji odhadnout podmíněnou pravděpodobnost $P(O|W)$ pro jakékoliv posloupnosti slov a příznaků O. Akustické modely by měly být flexibilní z důvodů rozdílnosti trénovacích a testovacích dat, přesné kvůli odlišení foneticky podobných slov a účinné pro nasazení při rozpoznávání v reálném čase. Postupem času se jako nejlepší ukázalo využití skrytých Markovových modelů (HMM = Hidden Markov Model).

Modelování skrytými Markovovými modely pohlíží na řečový signál diskrétně. Předpokládá se, že v krátkých časových úsecích je hlasové ústrojí ve stacionární poloze. Při dostatečně jemném rozdělení na mikrosegmenty tedy mohu tyto dílčí části popsat konstantami a pro model celé promluvy tyto subjednotky zřetězit.

2.3.1 Skryté Markovovy modely

Skrytý Markovův model by se dal popsat jako konečný pravděpodobnostní automat, jež popisuje stochastický proces v diskrétních okamžicích. Pravděpodobnosti přechodu a_{ij} mezi jednotlivými stavy, například ze stavu s_i v čase t do stavu s_j v čase $t+1$, lze vyjádřit podmíněnou pravděpodobností

$$a_{ij} = P(s(t+1) = s_j | s(t) = s_i)$$

Platí také, že suma všech pravděpodobností přechodu z určitého stavu je rovna jedné, tedy

$$\sum_{j=1}^{N} a_{ij} = 1$$

Rozdělení pravděpodobnosti o_t ve stavu s_j a čase t popisují funkce rozdělení výstupní pravděpodobnosti $b_j(o_t)$. Funkce $b_j(o_t)$ má rovněž funkci pravděpodobnosti pro diskrétní pozorování, resp. hustoty pravděpodobnosti pro spojité pozorování. Platí

$$b_j(o_t) = P(o_t|s(t) = s_j)$$

a pro všechny emitující stavy rovněž platí, že suma všech výstupních pravděpodobností je rovna jedné, tedy

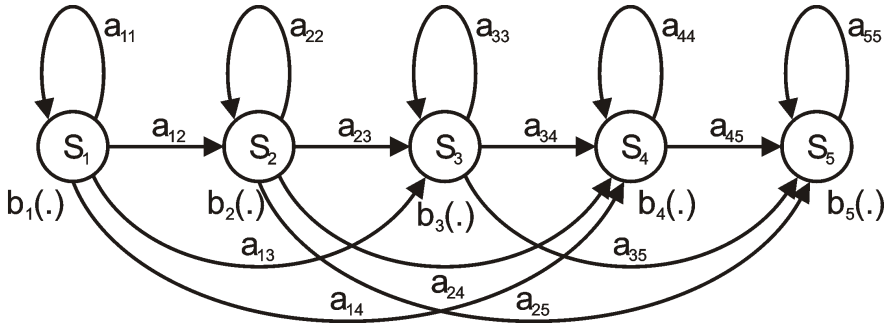
$$\sum_{j=1}^{N} b_j(o) = 1 \qquad \text{resp.} \int_o b_j(o)do$$

Nejčastěji využívanými pravděpodobnostními rozděleními jsou:

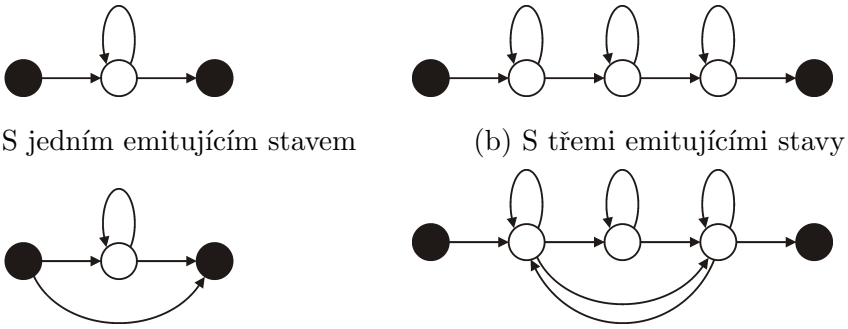
- Spojité rozdělení se směsí normálních hustotních funkcí
- Diskrétní rozdělení s vektorovou kvantizací podle kódové knihy
- Spojité rozdělení se svázanou směsí normálních hustotních funkcí

2.3.2 Různé typy skrytých Markovových modelů

Vzhledem k plynutí času kupředu se používají hlavně tzv. levo-pravé Markovovy modely. Proces začíná v prvním stavu a dále zůstává ve stejném stavu díky smyčce nebo přechází do dalšího stavu s vyšším indexem. Konec nastane příchodem posledního vzorku.



Obrázek 2.3: Příklad pětistavového skrytého Markovova modelu



(a) S jedním emitujícím stavem (b) S třemi emitujícími stavy

(c) Krátká pauza mezi slovy (sp) (d) Dlouhá pauza mezi slovy (sil)

Obrázek 2.4: Příklady modelů fonémů

V reálných úlohách rozpoznávání řeči se zpravidla trénují modely fonémů. Podle typu úlohy a řečové jednotky se volí vhodný tvar skrytého Markovového modelu.

V praxi se nejčastěji používá pětistavových modelů pro zajištění robustnosti odhadu a užití trifonů, tj. trojice po sobě jdoucích fonémů. První a poslední z pětice stavů slouží k řetězení, takže jednotlivé prvky trifonu se rozloží do zbylých třech vnitřních stavů.

K reprezentaci pravděpodobností přechodů mezi stavy slouží tzv. matice přechodu. Jedná se o čtvercovou matici, pro n stavů o velikosti $n \times n$.

$$P_{trans} = \begin{vmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{vmatrix}$$

Musí také platit, že suma jakéhokoliv řádku je rovna jedné. Pro případ pětistavového skrytého Markovova modelu bude mít ve většině případů matice tvar

$$P_{trans} = \begin{vmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & a_{22} & a_{23} & 0 & 0 \\ 0 & 0 & a_{33} & a_{34} & 0 \\ 0 & 0 & 0 & a_{44} & a_{45} \\ 0 & 0 & 0 & 0 & 0 \end{vmatrix}$$

kde na hlavní diagonále (a_{ij}, kde $j = i$) jsou pravděpodobnosti smyčky (loop), tj.

pravděpodobnost přechodu do stejného stavu. A nad hlavní diagonálou (a_{ij}, kde $j = i + 1$) jsou pravděpodobnosti přechodu do následujícího stavu.

2.3.3 Trénování parametrů skrytého Markovova modelu

Na rozdíl od volby topologie skrytého Markovova modelu, který se provádí spíše expertním způsobem, se parametry hledají trénováním nebo-li statistickou indukcí. Pro nalezení vhodného popisu trénovacích dat se většinou provádí pomocí metody maximální věrohodnosti neboli Maximum Likelihood.

Kritérium maximální věrohodnosti uvažuje pravděpodobnostní model $P(x|\lambda)$ s neznámými parametry λ. Na základě trénovacích dat $x_1, x_2, ..., x_N$ se pak snaží nalézt hodnoty příslušných parametrů. Tato funkce má tvar

$$F(x_1, x_2, ..., x_N|\lambda) = \prod_{n=1}^{N} P(x_N|\lambda)$$

přičemž hledáme maximum přes parametry λ tedy

$$\hat{\lambda} = \underset{\lambda}{\operatorname{argmax}} \prod_{n=1}^{N} P(x_N|\lambda)$$

V reálných úlohách se díky násobením velmi malých čísel často dostaneme k číslům tak malým, že je počítač považuje za nulu, proto se častěji používá logaritmus funkce

$$\hat{\lambda} = \underset{\lambda}{\operatorname{argmax}} \log \prod_{n=1}^{N} P(x_N|\lambda) = \underset{\lambda}{\operatorname{argmax}} \sum_{n=1}^{N} \log P(x_N|\lambda)$$

Snažíme se nalézt optimální hodnoty všech parametrů, jimiž jsou pravděpodobnost přechodu a_{ij} a hustotní funkce $b_j(.)$. Zpravidla se využívá normálního rozdělení, proto parametry hustotní funkce jsou střední hodnota μ_{jm}, kovarianční matice C_{jm} a váha jednotlivých složek c_{jm}.

$$\hat{\lambda} \equiv \{a_{ij}, c_{jm}, \mu_{jm}, C_{jm}\}, \text{ kde } \quad 1 \leq i, j \leq N, 1 \leq m \leq M$$

Nalezení optimálních parametrů a maximalizace věrohodnostní funkce nelze explicitně řešit. K řešení se tedy využívá iterativní numerické metody, EM algoritmu. Tento algoritmus využívá skryté proměnné y^e, která obsahuje informaci o stavech i indexech hustotní směsi.

2.3.4 EM algoritmus

Expectation-Maximization neboli EM algoritmus slouží k modelování rozdělení obrazů váženou směsí normálních hustotních funkcí. Jeho úkolem je nalézt parametry příslušných funkcí tak, aby v součtu co nejlépe popisovaly množinu obrazů. Model směsi ω_r je vyjádřen jako vážená směs hustotních funkcí ve tvaru

$$p(x|\omega_r) = \sum_{m=1}^{M} c_{rm} p(x|\theta_{rm}), \qquad r = 1, ..., R$$

kde θ_{rm} jsou parametry hustotní funkce m-té složky směsi ω_r a c_{rm} jsou váhy jednotlivých složek směsi. Každá směs je reprezentována střední hodnotou μ, kovarinační maticí C a apriorní pravděpodobností c.

Po zvolení počátečních podmínek θ_0 vypočteme očekávání (expectation) přes všechny hodnoty y a všechny pozorování x_n

$$Q(\theta, \bar{\theta}) = \sum_{n=1}^{N} \sum_{y} p(y|x_n, \theta) \ln p(x_n, y|\bar{\theta})$$

Ze všech kombinací hodnot parametrů $\bar{\theta}$ vybereme takovou množinu θ^*, pro kterou funkce Q dosahuje maxima (maximization)

$$\theta^* = \underset{\bar{\theta}}{\operatorname{argmax}}\, Q(\theta, \bar{\theta})$$

Ke zjištění pravděpodobností příslušnosti k jednotlivým složkám použijeme vztah pro normální rozložení, vynásobený navíc ještě apriorní pravděpodobností složky c

$$p(x_i, y|\bar{\theta}) = \frac{c_r}{\sqrt{(2\pi)^n |C_r|}} e^{-\frac{1}{2}(x_i - \mu_r)^T C_r^{-1} (x_i - \mu_r)}$$

kde n je dimenze obrazu, i je index obrazu a r je index směsi. Při hledání nových parametrů využijeme vztahů

$$\bar{c_i} = \frac{1}{N} \sum_{n=1}^{N} p(i|x_n, c_i, \mu_i, C_i)$$

$$\bar{\mu_i} = \frac{\sum_{n=1}^{N} p(i|x_n, c_i, \mu_i, C_i).x_n}{\sum_{n=1}^{N} p(i|x_n, c_i, \mu_i, C_i)}$$

$$\bar{C_i} = \frac{\sum_{n=1}^{N} p(i|x_n, c_i, \mu_i, C_i)(x_n - \bar{\mu}_i)(x_n - \bar{\mu}_i)^T}{\sum_{n=1}^{N} p(i|x_n, c_i, \mu_i, C_i)}$$

2.3.5 Parametry akustického modelu

Apriorní pravděpodobnost Nabývá hodnoty v intervalu $< 0; 1 >$. Tato hodnota vyjadřuje míru očekávatelnosti příslušného shluku. Mějme shluk o N prvcích. Tento shluk dále rozdělíme do M subshluků. Každý subshluk bude mít n_m prvků, kde m určuje index subshluku. Apriorní pravděpodobnost a_m tedy vypočteme jako poměr počtu prvků v m-tém subshluku ku celkovému počtu všech prvků

$$a_m = \frac{n_m}{N}$$

Zároveň platí, že součet všech pravděpodobností pro jednotlivé subshluky je roven jedné

$$\sum_{m=1}^{M} a_m = 1$$

Střední hodnota Jak už napovídá název, tento parametr reprezentuje střed shluku. Dimenze střední hodnoty je stejná jako dimenze dat a značí se μ. Vypočítá se jako součet všech prvků příslušného subshluku dělený počtem prvků subshluku

$$\mu_m = \frac{1}{n_m} \sum_{i=1}^{n_m} x_{im}$$

kde m je index subshluku a i je index prvku.

Kovarianční matice Tento parametr nabývá tvaru čtvercové matice dimenze vstupních dat. Kovarianční matice slouží k popisu tvaru shluku a vypočítá se jako střední hodnota druhé mocniny rozdílu prvku a střední hodnoty, tedy

$$C_m = E\{(x - \mu_m)(x - \mu_m)^T\} = \frac{1}{n_m} \sum_{i=1}^{n_m} (x_i - \mu_{mi})(x_i - \mu_{mi})^T$$

K výpočtu potřebuji střední hodnotu subshluku μ_m a všechny jeho prvky.

G-konstanta V akustickém modelu je rovněž používán další parametr, zvaný G-konstanta. Tento parametr nemá popisný význam jako předchozí parametry, ale pomáhá urychlit dílčí výpočty při užívání modelu. Uvažujeme normální rozdělení, tedy pro složku ω_m bude pravděpodobnost

$$P(x|\omega_m) = \frac{1}{(2\pi)^{\frac{n}{2}}\sqrt{\det C_m}} e^{-\frac{1}{2}(x-\mu_m)C_m^{-1}(x-\mu_m)^T} =$$

$$= (2\pi)^{-\frac{n}{2}} (\det C_m)^{-\frac{1}{2}} e^{-\frac{1}{2}(x-\mu_m)C_m^{-1}(x-\mu_m)^T}$$

po zlogaritmování dostaneme

$$\ln P(x|\omega_m) = -\frac{1}{2}\ln(2\pi)^n - \frac{1}{2}\ln(\det C_m) - \frac{1}{2}(x-\mu_m)C_m^{-1}(x-\mu_m)^T \ln e =$$

$$= -\frac{1}{2}\{\ln[(2\pi)^n \det C_m] + (x-\mu_m)C_m^{-1}(x-\mu_m)^T\}$$

Zlomek $-\frac{1}{2}$ je konstanta stejná pro všechny hodnoty, takže tu můžeme vypustit a $\ln[(2\pi)^n \det C_m]$ označíme jako G-konstantu.

$$\ln P(x|\omega_m) = G_{const} + (x-\mu_m)C_m^{-1}(x-\mu_m)^T$$

Výpočet determinantu matice je složitá operace a jelikož se tato část vzorce nemění, lze si ji napočítat jen jednou dopředu. Tím se výpočetní náročnost sníží a výsledky získáme rychleji.

2.4 Jazykový model

Jazykové modely se používají při rozpoznávání jazyka nebo v úlohách rozpoznání řeči poskytují informaci o jazyku. Pro rozpoznávání se využívá stochastických n-gramových modelů, jejichž úkolem je stanovit pro každou posloupnost slov W apriorní pravděpodobnost $P(W)$.

Při tvorbě jazykového modelu vezmeme korpus dat, ze kterého vypočítáme statistiky slov. Popis celého textu nebo vět by byl náročný, proto se využívá n-gramů, tj. posloupnosti n slov. Nejčastěji používané n-gramy

- zerogramy - jedno slovo $n = 0$, všechny slova mají stejnou pravděpodobnost
- unigramy - jedno slovo $n = 1$

- bigramy - dvě slova $n = 2$, první slovo reprezentuje historii a druhé aktuální slovo
- trigramy - tři slova $n = 3$, první slovo představuje předchůdce, druhé aktuální slovo a třetí následníka

Pravděpodobnosti jsou počítány na základě četností z korpusu dat. Vztah pro výpočet pravděpodobnosti i-tého n-gramu bude vypadat takto

$$P(W_i) = \frac{n_i}{N}$$

kde n_i je relativní četnost i-tého n-gramu a N je počet všech n-gramů. Častým problémem při tvorbě modelů bývá nedostatek trénovacích dat. V tomto případě to znamená, že jsme neviděli všechna slova nebo slovní spojení. Tyto neviděné n-gramy by měly nulovou pravděpodobnost a tím bychom je z jazyka vyloučili. Proto se využívá tzv. vyhlazování jazykového modelu tj. proces, ve kterém se odečte část pozorovaných n-gramů a přičte se n-gramům nepozorovaným.

2.5 Dekodér

Tento blok v procesu rozpoznávání řeči využívá informace ze všech výše popsaných částí. Posloupnost slov$P(W)$, tedy řečový signál, byl převeden parametrizací na posloupnost vektorů pozorování $P(O)$. Jazykový model nám poskytuje pravděpodobnosti výskytu slov nebo slovních spojení $P(W)$ a díky akustickému modelu máme i podmíněnou pravděpodobnost vektoru pozorování O za podmínky posloupnosti slov W. Z těchto dat se dekodér snaží odhadnout vyřčenou posloupnost slov $\hat{W}$ takovou, aby součin $P(O|W)$ a $P(W)$ byl maximální.

$$\hat{W} = \underset{W}{argmax}\, P(W)P(O|W)$$

Toto kritérium se nazývá kritérium maximální aposteriorní pravděpodobnosti nebo-li Maximum A posteriori Probability (zkr. MAP). V některých případech nehledáme pouze jedno nejlepší řešení, ale N nejpravděpodobnějších odhadů posloupností $\hat{W}$.

3. Metody dělení dat

Existuje mnoho metod, zohledňující různé vlastnosti. V této práci jsem zvolil metodu k-means, jež dělí na základě geometrické vzdálenosti. Metodami dělení a shlukováním obecně se zabývá tzv. shluková analýza. Nyní stručně shrnu, co pojem shlukovací analýza zahrnuje a popíši metody, jež budu dále využívat.

3.1 Shluková analýza

Termín shlukové analýzy poprvé použil Tryon roku 1939. Tento nástroj spadá pod analýzu dat, což je proces zjišťující užitečné informace, které mohou pomoci při rozhodování či stanovení závěrů. Jak už název napovídá, shluková analýza je zaměřena na hledání shluků. Shluk neboli cluster by se dal popsat jako seskupení objektů, jež jsou si navzájem podobné, mají společné vlastnosti, rysy a jsou zároveň různé, odlišitelné od ostatních. Metody shlukové analýzy mají široké využití. Můžeme se s nimi setkat například v ekonomii pří třídění zákazníků, ve statistice při zobecňování dat, kdy se kvůli zjednodušení nalezne střed shluku a použije se jako zástupce celé skupiny. Dále pro analýzu sociálních sítí nebo v biologii pro dočasné popsání společenství rostlin. Nebo také při dělení digitálního obrazu pro detekci hran či tzv. data miningu, což znamená získávání a třídění dat například pro průzkumy trhů, třídění dokumentů, atp. Hlavní předpoklad pro úspěšné nalezení shluků je ten, že data trénovací množiny opravdu tvoří shluky. Do stejného shluku patří takové prvky, které jsou si blízké neboli podobné, například geometricky. Oproti tomu nepodobné prvky, leží daleko od sebe, náleží různým shlukům. Často se ztotožňují pojmy shluk (cluster), který je výsledkem shlukování (clustering) a třída (class) jež je spojen s klasifikací (clasification). Pojem klasifikace znamená rozřazování objektů do již známých tříd, kdežto při shlukování známe maximálně jen jejich počet.

Shlukovací metody lze dělit do dvou skupin – **hierarchické** a **nehierarchické**.

- **Hierarchické** Tyto metody používáme většinou, neznáme-li přesný počet cílových tříd. Hierarchické shlukování se snaží v každém kroku dosáhnout lokálně nejlepšího řešení a k předchozím krokům se nelze vrátit, což je jeho hlavní nevýhoda. V zásadě se dělí na dva oddíly:
  - **Divizní hierarchické metody** – divizní přístup spočívá v postupném dělení. Na začátku jsou všechna data v jediném shluku a na základě stanoveného kritéria jsou dále dělena, dokud shluky neobsahují pouze jeden prvek. Tím se vytváří hierarchický systém. Avšak kvůli exponenciální časové složitosti je tento přístup použitelný jen na malých datech.
  - **Aglomerativní hierarchické metody** – vycházíme ze stavu, kde všechny prvky jsou samostatný shluk. Pomocí stanoveného kritéria tyto shluky slučujeme, dokud nevznikne jeden velký shluk, obsahující všechny prvky. Shlukování může být předčasně ukončeno, pokud jsou vzdálenosti mezi shluky příliš velké nebo pokud je dosažen požadovaný počet shluků. Problémem tohoto přístupu je, že pokud dojde k nevhodnému sloučení, které se projeví až později, nelze kroky vrátit.
- **Nehierarchické** Pomocí předem zvoleného kritéria rozloží danou množinu na podmnožiny, nevytváří hierarchickou strukturu. Vzniklé podmnožiny se dále jen upravují ve snaze nalézt optimální řešení. Tímto postupem se však většinou nenajde extrém globální, nýbrž jen lokální. Míru kvality rozkladu určíme kritériem, které je nutno zvolit s ohledem na požadované cíle a strukturu dat. **Možná kritéria určení kvality:**
  - Podobnost objektů ve shluku
  - Míra separace shluků
  - Rovnoměrnost rozložení objektů uvnitř shluku
  - Rovnoměrnost rozložení do shluků

  Největší skupinou nehierarchických metod jsou optimalizační metody. Jejich cílem je nalézt optimální rozklad určité množiny. Optimalizace probíhá

přesouváním jednotlivých bodů a snahou minimalizovat nebo maximalizovat dané kritérium. Užívají se z pravidla tehdy, známe-li počet tříd, do kterých chceme klasifikovat. Nyní si přiblížíme metody použité pro dělení parametrizovaných dat k jejich lepšímu popisu. Hlavní roli hrála metoda k-means. Jedná se o metodu optimalizačního nehierarchického shlukování, dělící data do předem stanoveného počtu shluků se snahou o minimalizaci stanoveného kritéria.

3.2 Metoda k-means

K-means neboli k-průměrová metoda je jednoduchá metoda pro dělení dat do předem stanoveného počtu shluků na základě jejich geometrických vlastností. Tuto metodu vytvořil v roce 1967 James MacQueen, proto se nazývá též MacQueenův shlukovací algoritmus. Postupem času vznikly různé modifikace této metody a stala se také vzorem pro mnoho dalších algoritmů. Abychom mohli algoritmus spustit, potřebujeme vstupní data, pevně daný počet tříd, do kterých chceme dělit, kritérium kvality dělení a podmínku pro zastavení. Na výstupu je pro každou třídu jeden centroid, tj. střed shluku, který reprezentuje celý shluk.

T – množina obrazů $\qquad$ J – hodnota kritéria

μ – střední hodnota $\quad$ s_k – počet obrazů x ve shluku T_k

Dělíme-li data o n prvcích do k tříd, zvolíme si na začátku algoritmu náhodně k středů. Dále bereme ze vstupních dat prvek po prvku a na základě zvoleného kritéria přiřazujeme prvky jednotlivým třídám. Definujme si kritérium minima kvadrátu odchylky:

$$J = \sum_{r=1}^{R} J_r = \sum_{r=1}^{R} \sum_{x \in T_r} d^2(x, \mu_r)$$

Jednotlivé shluky jsou reprezentovány centroidy, tj. pomyslnými středy. V každé iteraci se vypočtou vzdálenosti všech bodů od centroidů jednotlivých tříd a vzájemně se porovnají. Bod je vždy přiřazen shluku reprezentovaným nejbližším centroidem.

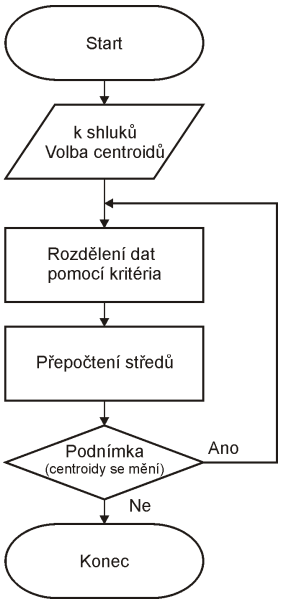
$$\kappa_{nr} = \begin{cases} 1, & \text{jestliže} \quad r = \arg\min\limits_{j} d^2(x_n, \mu_j) \\ 0, & \text{jinak} \end{cases}$$

kde κ představuje soubor binárních proměnných $\kappa \in \{0; 1\}$ jež přiřazuje bod právě jednomu shluku. Na konci každé iterace jsou přepočítány nové středy shluků pomocí vztahu

$$\mu_r = \frac{\sum_{n=1}^{N} \kappa_{nr} x_n}{\sum_{n=1}^{N} \kappa_{nr}}$$

Algoritmus se opakuje, dokud není splněna podmínka pro ukončení nebo pokud se středy jednotlivých shluků již nemění. Činnost k-means by se dala shrnout do čtyř kroků:

- Zvolení k středů (pro každý shluk jeden)
- Přiřazení vstupních dat pomocí kritéria příslušným shlukům
- Přepočítání středů shluků
- Opakování kroků 2. a 3., dokud se středy neustálí



Obrázek 3.1: Vývojový diagram metody k-means

Dosažení globálního minima ukazatele jakosti procesu není zaručeno. Výsledek mohou ovlivnit geometrické vlastnosti dat, počet shluků, do kterých dělíme nebo, a to hlavně, volba počátečních podmínek. Na vhodných datech však tato metoda dosahuje uspokojivých výsledků. Hlavními výhodami jsou jednoduchost a konvergence řešení v konečném počtu kroků.

3.2.1 Kritéria používaná ve shlukovacích algoritmech:

- Kritérium eukleidovské (minimum kvadrátu odchylky)

$$J = \sum_{i=1}^{R} J_i = \sum_{i=1}^{R} \sum_{x \in T_i} d^2(x, m_i)$$

kde μ_i je střední hodnota vektorů shluku T_i.

- Kritérium determinantu (založené na matici rozptylu S_i)

$$S_i = \sum_{x \in T_i} (x - \mu_i)^T (x - \mu_i)$$

Snažíme se tedy minimalizovat

$$J = \sum_{i=1}^{R} J_i = \sum_{i=1}^{R} \det S_i = \sum_{i=1}^{R} \det[\sum_{x \in T_i} (x - \mu_i)^T (x - \mu_i)]$$

Podobně cílené kritérium s euklidovskou vzdáleností, ale obecně může nabývat jiných hodnot.

- Invariantní kritérium Je založené na kombinaci dvou matic rozptylu
  - Matice rozptylu uvnitř shluků

$$S_w = \sum_{r=1}^{R} \sum_{r} = \sum_{r=1}^{R} \sum_{x \in T_i} (x - \mu_i)^T (x - \mu_i)$$

  - Matice rozptylu mezi shluky

$$S_b = \sum_{r=1}^{R} S_r (\mu_r - \mu)^T (\mu_r - \mu)$$

Kde S_r je počet obrazů ve shluku r a kde μ je celková střední hodnota

$$\mu = \frac{1}{s_1 + s_2 + ... + s_R} \sum_{r=1}^{R} S_r \mu_r$$

Maximalizujeme kritérium

$$J = \sum_{i=1}^{d} \frac{1}{++\lambda_i}$$

Kde λ_i jsou vlastní čísla matice $(S_w)^{-1} S_b$. Tohle kritérium je invariantní vůči většině lineárních transformací.

3.2.2 Modifikace metody k-means

K-medoids V této metodě se liší reprezentace středu shluku. Namísto centroidů, imaginární body s průměrnými hodnotami prvků shluku, jsou shluky popsány medoidy. Medoid je reálný prvek shluku, jehož vzdálenost od všech ostatních je minimální, tzn. nejvíce centrálně umístěný prvek shluku.

Optimalizované k-means Na rozdíl od klasického k-means, kde se přepočítávají středy shluků až na konci každé iterace, optimalizovaný k-means přepočítává středy shluků průběžně při rozřazování dat. Tato modifikace vede většinou k lepším výsledků a není třeba tolik iterací, ale výpočtově je náročnější.
Kroky algoritmu:

- Urči k středů shluků
- Dokud není splněná podmínka, opakuj
  - Přiřaď bod nejbližšímu středu
  - Přepočítej střed

Sférický k-means s opakovaným půlením Na začátku jsou všechny prvky v jedné skupině, která se rozdělí klasickým k-means na dvě. Podle zvoleného kritéria se vybere ze dvou vzniklých jedna, například ta s větší chybou. Vybraná skupina se dále dělí pomocí k-means na dvě. Nyní tedy máme tři skupiny. Tento postup se opakuje, dokud nedosáhneme předem zvoleného počtu shluků.
Kroky algoritmu:

- Vytvoření jednoho shluku
- Opakuj, dokud nedosáhneš požadovaného počtu shluků
  - Rozděl metodou k-means na dva
  - Vyber jeden shluk podle předem zvoleného kritéria

Metoda Fuzzy k-means Fuzzy k-means řeší problémy s osamělými prvky, které jsou těžko zařaditelné. Tato metoda přiřadí každému prvku pravděpodobnost, se kterou náleží do shluků. Čím je bod dál od středu, tím

má menší stupeň příslušnosti shluku a tím je i dobře popsáno i rozložení shluků. Dalším velkým rozdílem je to, že jeden bod může patřit do více shluků zároveň. Součet koeficientů příslušnosti jednotlivých bodů musí být roven jedné. Pokud mají všechny body příslušnost 1 (a k ostatním tedy 0), jedná se o pevné shlukování.

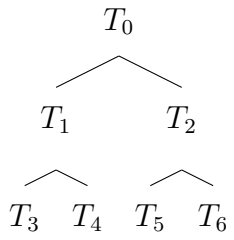
Kroky algoritmu:

- Volba počtu shluků
- Náhodné přiřazení koeficientů příslušnosti všem bodům
- Dokud není změna koeficientů menší než stanovený práh, opakuj
  - Výpočet středů shluků
  - Přepočtení pravděpodobností příslušnosti všech bodů ke všem shlukům

3.3 Rovnoměrné a nerovnoměrné binární dělení

Rovnoměrné binární dělení

Metoda rovnoměrného binárnoho dělení je jednoduchá divizní metoda. V principu vždy vybere jeden shluk a rozdělí jej na dva podshluky, proto binární. K výběru nepoužívá žádného kritéria, ale bere shluky tak, jak jdou za sebou, proto rovnoměrné. Díky tomuto typu výběru je vhodné volit počet cílových shluků rovný některému z mocniny čísla dva.

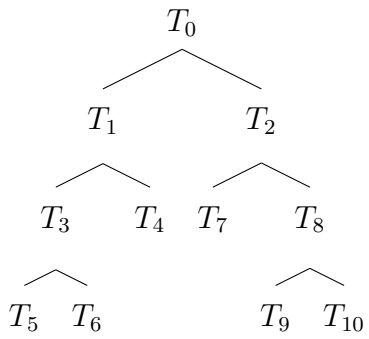


Obrázek 3.2: Strom dělení metodou rovnoměrného binárního dělení

Výsledkem jsou vždy jen koncové stavy, tedy v tomto případě čtyři shluky T_3, T_4, T_5, T_6. Zastavovací podmínkou může být například přesný počet tříd nebo pokud se celková chyba dostane pod předem stanovenou hodnotu.

Nerovnoměrné binární dělení

Tato metoda pracuje podobně jako předchozí metoda rovnoměrného dělení. Rozdíl spočívá ve výběru shluku k dělení. Na rozdíl od předchozí metody vybere nerovnoměrné binární dělení shluk s největší chybou pomocí zvolené kriteriální funkce, nejčastěji euklidovské, a tu pak dále dělí.



Obrázek 3.3: Strom dělení metodou nerovnoměrného binárního dělení

Výslednými shluky jsou $T_4, T_5, T_6, T_7, T_9, T_{10}$. Hodnoty $T_i, i = 0, 1, 2, ...$ reprezentující shluk bývají často hodnoty kriteriální funkce, tedy chyby shluku. Opět můžeme dělit do přesného počtu tříd, ale častěji se tato metoda omezuje velikostí chyby.

4. Trénování akustického modelu pomocí HTK

S rozšiřováním řešení úloh pomocí skrytých Markovových modelů vznikaly i různé nástroje na jejich bázi. Jedním z nich je tzv. HTK nebo-li Hidden Markov Model Toolkit, vyvinutý na Cambridge University Engineering Department (CU-ED). Tento software je hlavně využíván v úlohách rozpoznávání řeči, ale dá se použít i v mnoha jiných rozpoznávacích úlohách využívajících skryté Markovovy modely.

4.1 Příprava k trénování

Pro to, abychom mohli s HTK pracovat, potřebujeme si připravit několik věcí

- nahrané trénovací promluvy ve formátu wav
- seznam trénovacích promluv pro parametrizaci *param.scp*
- seznam testovacích parametrizovaných promluv *test.scp*
- seznam trénovacích parametrizovaných promluv *train.scp*
- soubor s přepisem všech promluv na úrovni slov *words.mlf*
- slovník výslovností *dict*
- seznamy symbolů fonetické abecedy používaných při transkripci *monophones0* a *monophones1*

Dále musíme vytvořit přepis na úrovni fonémů, což uděláme pomocí slovní transkripce, slovníku výslovností a programu *HLEd* z balíku HTK. Tento krok provedeme pro monophones0 i monophones1. Rozdíl mezi těmito dvěma seznamy symbolů je ten, že monophones1 obsahuje navíc symbol krátké mezislovní pauzy *sp*.

Před započetím tvorby modelu si ještě zparametrizujeme data. Mohli bychom provádět tento krok přímo za běhu programu, nicméně výhodnější je připravit si

data předem. Parametrizaci provádíme dalším programem balíku HTK a to *HCopy*. Jedním z parametrů *HCopy* je i soubor obsahující parametry parametrizace.

4.2 Tvorba monofonních modelů

Všechny fony české fonetické abecedy jsou reprezentovány jedním pětistavovým skrytým Markovovým modelem. První a poslední stavy jsou neemitující, tedy nevytváří žádné vektory parametrů a slouží pouze k řetězení. Parametry modelů jsou uloženy ve speciálním formátu

```
~o <VecSize> 39 <MFCC_0_D_A>
~h "proto"
```

Tyto údaje v hlavičce nám říkají, jak velké jsou vektory parametrů a jejich typ. Dále následuje název modelu zpravidla pojmenovávaném *proto* tzn. prototyp. Struktura popisu jednotlivých modelů vypadá pak takto

```
<BeginHMM>
<NumStates> 5
<State> 2
<Mean> 39
0.0 0.0 0.0 ...
<Variance> 39
1.0 1.0 1.0 ...
<State> 3
...
<TransP> 5
0.0 1.0 0.0 0.0 0.0
0.0 0.6 0.4 0.0 0.0
0.0 0.0 0.6 0.4 0.0
0.0 0.0 0.0 0.7 0.3
0.0 0.0 0.0 0.0 0.0
<EndHMM>
```

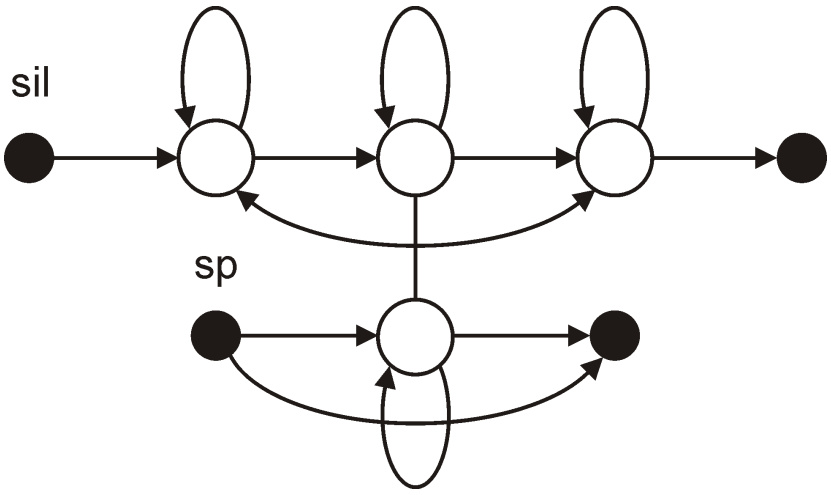
<*BeginHMM*> označuje začátek popisu nějakého monofonu. Následuje <*NumStates*> s číslem, označujícím počet stavů. V další části už popisujeme jednotlivé stavy parametry používaných v normálním rozdělení, tedy střední hodnotou <*Mean*> a diagonálou kovarianční matice <*Variance*>. Popis modelu zakončuje matice přechodu <*TransP*>, obsahující informaci o pravděpodobnostech přechodu mezi jednotlivými stavy.

Pro výpočet střední hodnoty a kovarianční matice využijeme program *HCompV*, které nastaví všechna Gaussova rozdělení v modelu *proto* na spočtené hodnoty.

Dále je třeba vytvořit *Master Macro File* (MMF), jež obsahuje definice HMM jednotlivých fonémů. Tento krok provedeme spuštěním *MakeMMF* ve stejné složce jako máme model *proto*. Tímto krokem jsou připravy a inicializace hotovy a můžeme přejít k samotnému trénování.

K trénování neboli reestimaci použijeme další program z balíčku HTK, tentokrát *HERest*, který pracuje na základě Baum-Welchově algoritmu.

Dalším krokem v tvorbě modelu bude úprava modelů pauz. Doposud jsme používali pouze model dlouhé pauzy *_sil_*, avšak pro zkvalitnění je třeba uvažovat i model krátké pauzy *_sp_*. Úprava spočívá ve spojení těchto dvou modelů přidáním vazby mezi prostředními stavy modelů.



Obrázek 4.1: Spojení modelů krátké a dlouhé pauzy

Takto upravený model pauzy bude mnohem lépe modelovat jak krátké, tak dlouhé pauzy nebo ticho, ale dokáže i absorbovat šumy, které mohou pauzy obsahovat.

Posledním krokem trénování modelu se nazývá přerovnání trénovacích dat. K tomuto úkonu použijeme ze sady HTK program *HVite*, pracujícím na základě Viterbiho algoritmu. Přerovnání by se dalo shrnout jako výběr takové fonetické transkripce ze slovníku, která nejlépe sední na monofonovou trankripci vytvořenou *HVite* na úrovni slov. Výstupem tohoto procesu je soubor *aligned.mlf* obsahující novou monofónní transkripci. Některá slova se zarovnat nepodařilo kvůli špatným přepisům nebo vadným datům, takže příslušné nahrávky se v seznamu již neobjeví. Pro nalezení a následné odstranění ze seznamu *train.scp* nám poslouží program *CreateAligned.exe*. Nyní modely znovu reestimujeme pomocí *HERest* a můžeme přejít k rozpoznávání.

4.3 Rozpoznávání

Vytvořili jsme monofonový akustický model a nyní jej otestujeme na "neznámých"promluvách. Pokud jsme tak neudělali dříve, data zparametrizujeme a předložíme seznam těchto nahrávek společně s vytvořeným modelem programu *HVite*. V předchozích krocích jsme *HVite* používali k přerovnávání trénovacích dat, ale změnou parametrů jej můžeme použít i na tuto úlohu. Výsledky rozpoznání *HVite* ještě upravíme aplikací programu *HResults*, který vypočítá úspěšnost rozpoznání neznámých promluv. Ke zjištění úspěšnosti potřebuji k testovacím neznámým datům i přepisy, takže úplně neznámá nejsou.

```
------------------------ Sentence Scores -------------------------
====================== HTK Results Analysis =====================
  Date: Sun Aug 18 10:00:57 2013
  Ref : test\_words.mlf
  Rec : vysledek_all_31.txt
------------------------- File Results --------------------------
00010008.rec:   84.62( 84.62)  [H=  11, D=  0, S=  2, I=  0, N= 13]
00010026.rec:  100.00(100.00)  [H=  14, D=  0, S=  0, I=  0, N= 14]
...
01060036.rec:   66.67( 60.00)  [H=  10, D=  3, S=  2, I=  1, N= 15]
```

```
---------------------- Overall Results -------------------------
SENT: %Correct=8.25 [H=8, S=89, N=97]
WORD: %Corr=81.04, Acc=79.57 [H=1103, D=84, S=174, I=20, N=1361]
================================================================
```

Z této ukázky výsledků lze vyčíst, že jsme úspěšně rozpoznali 81.04% slov z 97 testovacích promluv. Co přesně rozpoznávač rozpoznal nalezneme ve výstupním souboru *HVite* a můžeme se tak o úspěšnosti přesvědčit sami.

Jakmile bylo zjištěno optimální zarovnání, lze vypočítat počet substitučních chyb, což značí písmeno S, chyb vypouštěním označených D a chyb vložením I. Na základě těchto údajů jsou pak vypočítány pravděpodobnosti správného odhadu

$$\text{Percent Correct} = \frac{N - D - S}{N} \times 100\%$$

kde N je celkový počet testovacích promluv. Tento výpočet ale nebere v potaz chyby vkládání. Tento aspekt zohledňuje procentuální přesnost

$$\text{Percent Accuracy} = \frac{N - D - S - I}{N} \times 100\%$$

což je pro popis rozpoznávání mnohem popisnější.

4.4 Přidávání složek v HTK

Použil jsem zparametrizovaná řečová data a pomocí HTK toolkitu jimi trénoval řečový model. HTK uvažuje normální rozdělení dat, tedy popisuje data pomocí Gaussovy křivky. Tento popis dosahuje jednoho z nejlepších výsledků, nicméně popis fonému jen jednou Gaussovkou je příliš obecný a nepřesný. Z toho důvodu budu postupně zvyšovat počet složek směsi a budu sledovat, jaký to bude mít vliv na kvalitu rozpoznání.

Pro urychlení získávání výsledků jsem si připravil skripty *fronta.bat* a *sada.bat*. Skript *fronta.bat* volá druhý skript a jako parametry přikládá, jakému modelu se má přidat složka a jaký vznikne.

fronta.bat

```
call sada.bat 00 01
call sada.bat 01 02
call sada.bat 02 03
.
.
.
```

Volaný skript *sada.bat* vytváří nové složky pro nové modely, přidá složku a několikrát přerovná.

sada.bat

```
md hmm_%2
md hmm_%2_01
.
.
.

HHEd -T 1 -A -C cf.mfc -H hmm_%1_04\models -M hmm_%2
add_next.hed monophones1.sdx > hmm_%2\log

Herest -T 1 -C cf.mfc -I aligned.mlf -t 250.0 150.0 1000.0
-S aligned.scp -s hmm_%2_01\stats -H hmm_%2\models
-M hmm_%2_01 monophones1.sdx > hmm_%2_01\log
.
.
.
```

5. Experimenty

V předchozích kapitolách jsem popsal, jakou funkci plní akustický model a jak vypadá jeho vnitřní struktura. Nyní přejdu k praktické části, kde si akustický model vytvořím z trénovacích dat a otestuji na testovacích datech. Nejdříve budu vytvářet akustický model s využití HTK, kde budu testovat i vliv počtu složek směsi na kvalitu modelu. Poté si připravím data jednotlivých fonémů, rozdělím metodou k-means a sestavím vlastní akustický model, který rovněž otestuji na sadě testovacích dat. Nakonec výsledky rozpoznávání akustických modelů porovnám.

5.1 Trénovací a testovací data

Abychom mohli provádět trénování akustického modelu, potřebujeme sadu trénovacích řečových dat. Kvalitu budeme následně testovat rozpoznáváním neznámé sady testovacích dat. Avšak kvůli věrohodnosti výsledků by tyto dvě sady měly obsahovat odlišná data.

Vzal jsem sadu trénovacích nahrávek formátu wav, obsahujících 2149 promluv. Jako testovací sada mi posloužila menší skupina 97 nahrávek. Obě tyto sady jsem zparametrizoval. Použil jsem melovské frekvenční kepstrální koeficienty (MFCC). Počet kepstrálních koeficientů 11, šířka okénka 32 ms s posunem 10 ms.

5.2 Testování počtu složek směsi v HTK

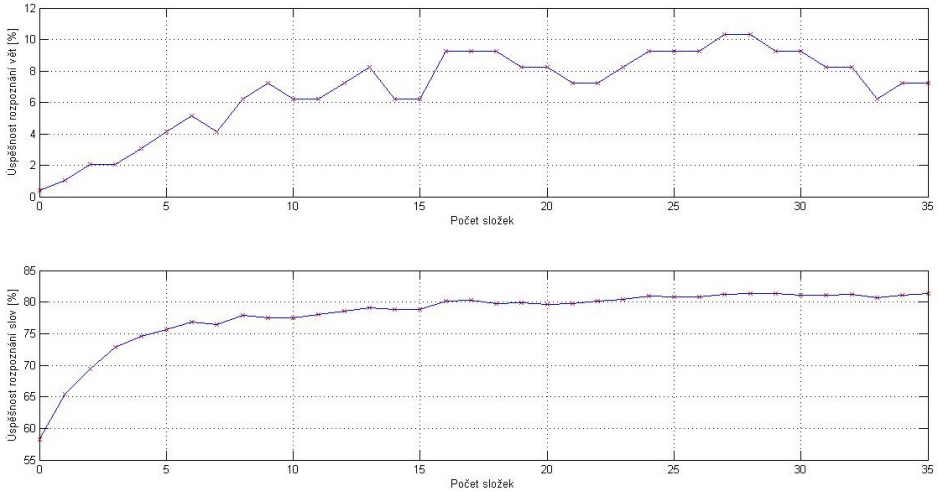
Po parametrizaci dat jsem pomocí HTK začal tvořit akustický model. Nejprve jsem začal na ve stavu, kdy všechny směsi měly jen jednu složku. Provedl jsem rozpoznávání pomocí *HVite* a s *Hresults* si výsledky upravil, ale úspěšnost byla velmi nízká.

```
HVite -C cf.mfc -H hmm_%2_04\models -S test\_test.scp
-i vysledek_all_%2.txt -l * -p -60.0 -w test\_wdnet.sdx
test\_dict.sdx monophones1.sdx
Hresults -f -I test\_words.mlf monophones1.sdx
vysledek_all_%2.txt > vysledek_%2.txt
```

Proto jsem začal navyšovat počet složek ve směsích vždy o jedna, model přerovnal a otestoval na testovacích datech. Tento proces navyšování jsem prováděl až do počtu 35 složek. Výsledky testování modelů jsem znázornil graficky a vypsal do tabulky

Tabulka 5.1: Úspěšnost rozpoznávání slov v závislosti na počtu složek

Mixes	corr[%]	acc[%]	Mixes	corr[%]	acc[%]
1	59.74	56.06	19	79.72	77.96
2	65.39	62.67	20	79.87	78.18
3	69.43	66.72	21	79.65	78.03
4	72.89	70.17	22	79.72	78.1
5	74.58	72.01	23	80.09	78.4
6	75.61	73.48	24	80.38	78.91
7	76.78	74.5	25	80.9	79.57
8	76.41	74.28	26	80.82	79.5
9	77.88	75.61	27	80.82	79.43
10	77.44	75.24	28	81.19	79.79
11	77.52	75.31	29	81.34	79.87
12	78.03	75.83	30	81.41	80.09
13	78.62	76.41	31	81.04	79.57
14	79.06	77.22	32	81.04	79.57
15	78.84	77.0	33	81.19	79.87
16	78.84	77.15	34	80.68	79.28
17	80.09	78.47	35	81.12	79.72
18	80.24	78.55	36	81.41	80.16



Obrázek 5.1: Závislost úspěšnosti rozpoznávání na počtu složek směsi

Z grafu je patrné, že přidáváním složek se úspěšnost rozpoznávání zvýší. Nejprve byl nárůst markantní, ale s přibývajícími složkami se nárůst úspěšnosti zmenšoval. Zvolím si pro popis 32 složek. Z provedeného pokusu vím, že tento počet složek popisuje dostatečným způsobem a dále budu používat binární rovnoměrné dělení, takže je 32 vhodné i z tohoto důvodu.

5.3 Příprava dat pro dělení metodou k-means

Předchozím pokusem jsem zjistil počet složek k dostatečnému popisu fonémů. Pomocí HTK jsem vytvořil akustický model o příslušném počtu složek. Nyní sestavím další model, ale k jeho tvorbě tentokrát použiji metodu k-means.

Abych mohl vlastní model vytvořit, budu potřebovat trénovací data pro všechny stavy všech fonémů. Dalším krokem bude tedy získání trénovacích dat.

Pomocí HTK si na základě vytvořeného modelu označím trénovací data. K tomu využiji příkaz *HVite*.

```
HVite -T 1 -l * -y lab -b _SIL_ -o N -C _cf.mfc -a -f
-H hmm/models -i aligned.txt -t 250.0 -I words.mlf -S test.scp
dict.txt monophones> hvite.log
```

Výsledek jsme uložili do souboru *hvite.log*. Příklad výstupu

```
"*/00010001.lab"
          0     100000  _sil_[2]  61.320652 _SIL_
     100000     300000  _sil_[3]  63.304131
     300000     800000  _sil_[4]  59.098217
        ...        ...  ...       ...
    5300000    5600000  a[2]      49.337833 akcie
    5600000    5800000  a[3]      43.396046
    5800000    6000000  a[4]      55.299007
    6000000    6200000  k[2]      60.133831
    6200000    6500000  k[3]      59.676769
        ...        ...  ...       ...
```

V uvozovkách název souboru promluvy a dále informace, která část dat nejlépe odpovídá jakým stavům. Když tedy máme data označená, je třeba je roztřídit. V jazyce python jsem napsal skript, který bude číst odkud kam trvá jaký foném a bude rozkopírovávat příslušná data do textových souborů. Každý stav každého fonému má svůj vlastní soubor. Máme-li 32 fonémů, dostaneme tedy 96 souborů s roztřízenými daty.

5.4 Spočtení akustického modelu metodou k-means

A nyní k praktickému užití výše uvedených metod. V předchozích krocích jsem zjistil, jak závisí kvalita popisu na počtu složek směsi a rozhodl se zvolit popisovat 32 složkami. Dále jsem vytvořil model pomocí HTK, označil si v trénovacích datech jaký vektor připadá jakému stavu fonému a rozdělil tato data do souborů podle stavů fonémů.

V dalším kroku na data v jednotlivých souborech aplikuji dělící metody z předchozí kapitoly a vypočítám parametry, jež danou množinu reprezentují. Aplikuji dvě kombinace:

1. k-means + rovnoměrné dělení
2. k-means + nerovnoměrné dělení

V obou případech dělím metodou k-means daný shluk nebo podshluk na dvě části. V prvním případě však k výběru používám rovnoměrné dělení až do cílového počtu 32 a v druhém případě postupuji obdobně, jen vždy dělím shluk s největší kriteriální funkcí, tj. největší chybou. Jako příklad uvedu strom dělení stavu jednoho z fonémů.

<table>
<tr><td colspan="32">Inf</td></tr>
<tr><td colspan="16">865</td><td colspan="16">996</td></tr>
<tr><td colspan="8">225</td><td colspan="8">409</td><td colspan="8">479</td><td colspan="8">315</td></tr>
<tr><td colspan="4">70</td><td colspan="4">103</td><td colspan="4">134</td><td colspan="4">206</td><td colspan="4">187</td><td colspan="4">225</td><td colspan="4">163</td><td colspan="4">112</td></tr>
<tr><td colspan="2">32</td><td colspan="2">28</td><td colspan="2">35</td><td colspan="2">43</td><td colspan="2">65</td><td colspan="2">51</td><td colspan="2">64</td><td colspan="2">114</td><td colspan="2">91</td><td colspan="2">72</td><td colspan="2">93</td><td colspan="2">104</td><td colspan="2">93</td><td colspan="2">39</td><td colspan="2">39</td><td colspan="2">57</td></tr>
<tr><td>8</td><td>18</td><td>17</td><td>7</td><td>23</td><td>6</td><td>27</td><td>7</td><td>25</td><td>33</td><td>15</td><td>27</td><td>33</td><td>22</td><td>39</td><td>59</td><td>34</td><td>45</td><td>28</td><td>37</td><td>47</td><td>33</td><td>45</td><td>44</td><td>37</td><td>42</td><td>5</td><td>27</td><td>15</td><td>18</td><td>15</td><td>34</td></tr>
</table>

Obrázek 5.2: Rovnoměrné dělení metodou k-means

<table>
<tr><td colspan="64">Inf</td></tr>
<tr><td colspan="32">865</td><td colspan="32">996</td></tr>
<tr><td colspan="16">225</td><td colspan="16">409</td><td colspan="16">479</td><td colspan="16">315</td></tr>
<tr><td colspan="8">70</td><td colspan="8">103</td><td colspan="8">134</td><td colspan="8">206</td><td colspan="8">187</td><td colspan="8">225</td><td colspan="8">163</td><td colspan="8">112</td></tr>
<tr><td colspan="4">32</td><td colspan="4">28</td><td colspan="4">35</td><td colspan="4">43</td><td colspan="4">65</td><td colspan="4">51</td><td colspan="4">64</td><td colspan="4">114</td><td colspan="4">91</td><td colspan="4">72</td><td colspan="4">93</td><td colspan="4">104</td><td colspan="4">93</td><td colspan="4">39</td><td colspan="4">39</td><td colspan="4">57</td></tr>
<tr><td colspan="2">0</td><td colspan="2">0</td><td colspan="2">0</td><td colspan="2">0</td><td colspan="2">0</td><td colspan="2">0</td><td colspan="2">27</td><td colspan="2">7</td><td colspan="2">25</td><td colspan="2">33</td><td colspan="2">15</td><td colspan="2">27</td><td colspan="2">33</td><td colspan="2">22</td><td colspan="2">39</td><td colspan="2">59</td><td colspan="2">34</td><td colspan="2">45</td><td colspan="2">28</td><td colspan="2">37</td><td colspan="2">47</td><td colspan="2">33</td><td colspan="2">45</td><td colspan="2">44</td><td colspan="2">37</td><td colspan="2">42</td><td colspan="2">0</td><td colspan="2">0</td><td colspan="2">0</td><td colspan="2">0</td><td colspan="2">15</td><td colspan="2">34</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>17</td><td>31</td><td>0</td><td>0</td><td>21</td><td>21</td><td>0</td><td>0</td><td>0</td><td>0</td><td>25</td><td>18</td><td>0</td><td>0</td><td>20</td><td>18</td><td>22</td><td>17</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
</table>

Obrázek 5.3: Nerovnoměrné dělení metodou k-means

Na konci každého rozdělení dat stavu fonému vezmu výsledné shluky jeden po druhém a pro každý vypočítám parametry, ze kterých budu později sestavovat model. Potřebnými parametry jsou apriorní pravděpodobnost, střední hodnota, kovarianční matice a tzv. G-konstanta.

5.4.1 Problémy při výpočtu parametrů

Při vypočítávání parametrů jsem narazil na dva problémy. Příčinou obou problémů byl v zásadě nedostatek trénovacích dat.

Po roztřídění dat do souborů podle příslušnosti ke stavům fonémů jsem dostal různě velké sady vektorů. Některé fonémy jsou v mluvené řeči velmi frekventované a jiné se naopak vyskytují zřídka. Tento jev se odrazil i v mých datech, takže soubory často se vyskytujících fonémů obsahovaly velké mnoství dat a poskytovaly tak výborná data pro získání parametrů. Oproti tomu data méně častých fonémů neposkytovala příliš bohatou informaci, což vedlo i k několika problémům.

První problém nastal, když se po rozdělení vyskytly takové shluky, které měly pouze jeden prvek. Tento prvek byl totožný se střední hodnotou a kovarianční matice by tedy byla nulová. Řešení nabídlo HTK, které takový problém řeší nahrazením kovarianční matice nějakou výchozí optimalizovanou kovarianční maticí.

V určitém případě se ale stalo, že dat k rozdělení nebylo ani 32, což vedlo k druhému problému. Například 17 vektorů do 32 shluků rozdělit nelze. Dělící algoritmus tedy dokonvergoval k 17 jednoprvkovým shlukům při nerovnoměrném dělení a k ještě méně shlukům při dělení rovnoměrném. Kovarianční matici jsem pro shluky vyřešil maticí výchozí jako v předchozím problému a nedostatečný počet složek jsem nechal být. Model nepotřebuje nezbytně popis stejným počtem složek u všech stavů všech fonémů, takže si stačilo poznamenat, kolik složek mám a podle toho upravit příslušné parametry při tvorbě modelu.

Pokud nemám dostatek dat z důvodu malého výskytu fonému a model tedy není příliš přesný, můžu předpokládat, že ani v testovacích datech se tento foném často vyskytovat nebude. Výskyt tohoto fonému a případné následné špatné určení by nám nemělo úspěšnost rozpoznávání příliš snížit. Nicméně k vypočtení kvalitního popisu a sestavení co nejlepšího modelu, potřebuji dostatečné množství trénovacích dat pro všechny fonémy.

5.4.2 Sestavení akustického modelu z vypočtených dat

Při sestavování vlastního modelu jsem jako vzor použil model vytvořený HTK. Hlavičku s parametry jsem nechal stejnou, jen obsah jsem měnil. Každý foném je uvozen *h"značka fonému"*. Celý popis fonému je mezi značkami *<BEGINHMM>* a *<ENDHMM>*. V těle popisu je také na začátku uveden počet stavů modelu tagem *<NUMSTATES>*, což je zpravidla 5 a na konci matice přechodu. Mezi těmito prvky jsou jednotlivé popisy stavů. Jako první je uvedeno číslo popisovaného stavu *<STATE>*, tedy v našem případě 2 - 4. Následuje počet složek směsi *<NUMMIXES>*, což pro většinu fonémů je 32 a dál jsou jednotlivé složky vypisovány. Prvním parametrem je apriorní pravděpodobnost spolu s číslem směsi *<MIXTURE> 1-32 apriorní pravděpodobnost*. Následuje střední hodnota *<MEAN> dimenze vektoru* a na novém řádku hodnota. Dalším parametrem je variance. Formát má obdobný jako střední hodnota, tedy *<VARIANCE> dimenze vektoru* a na nový řádek hodnoty, jen z matice se vypisuje pouze diagonála. Jako poslední zaznamenáme g-konstantu v prostém formátu *<GCONST> hodnota*. Obdobným postupem popíšeme všech 32 složek všech třech stavů. Takto postupujeme pro všechny fonémy. Ukázka kódu modelu

```
~h "m"
<BEGINHMM>
<NUMSTATES> 5
<STATE> 2
<NUMMIXES> 32
<MIXTURE> 1 3.341376e-02
<MEAN> 36
 3.575482e+00 -4.373349e-01 ... 3.512352e-04
<VARIANCE> 36
 1.313378e-02  1.022574e-02 ... 1.305377e-04
<GCONST> -1.603830e+02
<MIXTURE> 2 4.109088e-02
<MEAN> 36
 3.554643e+00 -4.931567e-01 ... 4.995180e-03
<VARIANCE> 36
 1.143814e-02  1.070835e-02 ... 1.185954e-04
<GCONST> -1.594945e+02
...
...
<MIXTURE> 32 2.062589e-02
<MEAN> 36
 5.373437e+00 -6.011907e-01 ... -1.748535e-02
<VARIANCE> 36
 2.090508e-02  1.864629e-02 ... 1.209827e-04
<GCONST> -1.586797e+02
<TRANSP> 5
 0.000000e+00 1.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
 0.000000e+00 5.943106e-01 4.056894e-01 0.000000e+00 0.000000e+00
 0.000000e+00 0.000000e+00 6.591032e-01 3.408968e-01 0.000000e+00
 0.000000e+00 0.000000e+00 0.000000e+00 6.371614e-01 3.628386e-01
 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
<ENDHMM>
```

Tímto způsobem jsem sestavil celý model pro rovnoměrné i nerovnoměrné dělení metodou k-means. Modely byly sestaveny, takže jsem pomocí HTK otestoval jejich kvalitu.

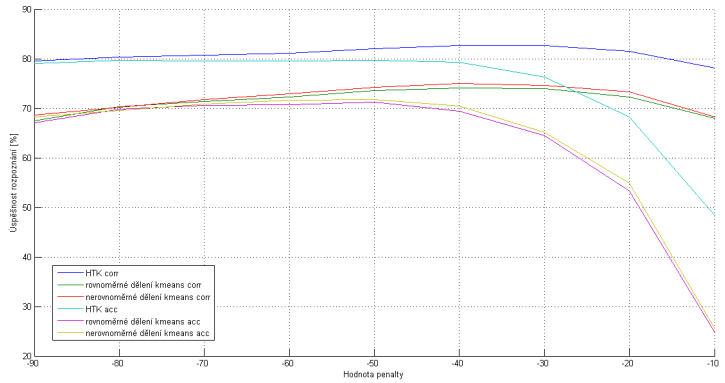
```
HVite -C cf.mfc -H tvorba_modelu/model_kmeans_nerovnomerny
-S test/_test.scp -i all_kmeans_ner.txt -l * -p -60.0
-w test/_wdnet.sdx test/_dict.sdx monophones1.sdx
Hresults -f -I test\_words.mlf monophones1.sdx all_kmeans_ner.txt
> vysledek_kmens_ner.txt
```

A obdobně pro model z dat získaných rovnoměrným dělením.

5.5 Porovnání modelů

Nyní mám tedy trojici výsledků rozpoznávání nad totožnými testovacími daty. První z modelu vytvořeným HTK, který používá EM algoritmu. Druhý sestavený mnou na základě nerovnoměrného dělení dat metodou k-means. A třetí model, také ručně vytvořený, na základě dat rovnoměrného dělení metodou k-means.

Při rozpoznávání pomocý *HERest* z balíku HTK je jedním z parametrů i penalizace $-p$, kterou můžeme měnit váhu vkládání slov. Bohužel při zjišťování optímální hodnoty musíme zkoušet a sledovat, jak se výsledky vyvíjí. Zkusil jsem tedy několik hodnot penalizace a výsledky zapsal do tabulky



Obrázek 5.4: Závislost hodnoty penalizace na úspěšnosti rozpoznávání

Tabulka 5.2: Hodnoty úspěšnosti rozpoznávání v závislosti na penalizaci

	HTK		Rovnoměrný k-means		Nerovnoměrný k-means	
Penalizace	Corr [%]	Acc [%]	Corr [%]	Acc [%]	Corr[%]	Acc [%]
-10	78.1	48.42	68.04	24.69	68.19	25.5
-20	81.48	68.26	72.3	53.27	73.33	54.89
-30	82.66	76.34	73.92	64.44	74.65	65.17
-40	82.66	79.35	74.14	69.36	75.02	70.46
-50	82.0	79.72	73.55	71.2	74.21	71.71
-60	81.04	79.57	72.3	70.76	72.96	71.57
-70	80.68	79.57	71.34	70.54	71.79	70.9
-80	80.38	79.65	70.32	69.8	70.17	69.51
-90	79.57	79.06	67.45	67.08	68.63	68.26

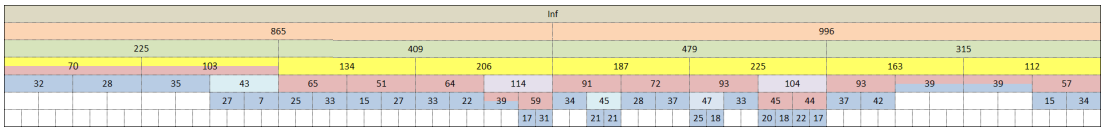
Na základě tohoto pokusu jsem zjistil, že ve většině případů dosahují modely nejlepší hodnoty rozpoznání při penalizaci -50.

Tabulka 5.3: Výsledky testování kvality modelů

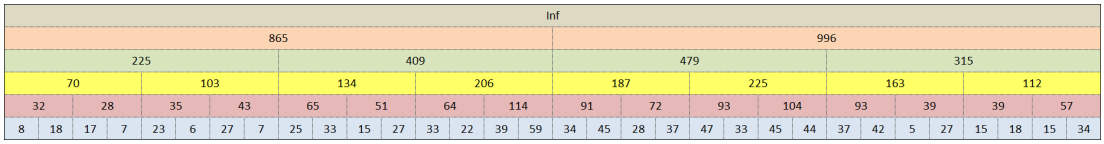
Úspěšnost rozpoznání	word corr [%]	word acc [%]
HTK - EM algoritmus	82.00	79.72
K-means - nerovnoměrné	74.21	71.71
K-means - rovnoměrné	73.55	71.20

Z tabulky výsledků testování modelů je patrné, že pravděpodobnostní popis EM algoritmem popisuje data lépe než geometrický k-means. Oproti tomu rovnoměrné a nerovnoměrné binární dělení poskytuje výsledky téměř totožné.

Ačkoliv by se dalo předpokládat, že nerovnoměrné dělení bude mít z principu výrazně lepší výsledky než dělení rovnoměrné, hodnoty se příliš neliší. Hlavním důvodem toho výsledku by měl být charakter dat. Jsou-li data rozložena v prostoru rovnoměrně, budou oba postupy dělit ve většině případů stejné shluky. Dalším důvodem pro podobnost výsledků může být malý počet dělení. Dělíme do 32 shluků, tj $2^5 \Rightarrow 5$ dělení. S rostoucím počtem dělení roste i počet složek a tím klesá pravděpodobnost, že si nerovnoměrné dělení vybere stejnou složku jako rovnoměrné dělení. Pro názornost použiji záznam chyb jako v kapitole *Testování počtu složek*.



Obrázek 5.5: Příklad nerovnoměrného dělení



Obrázek 5.6: Příklad rovnoměrného dělení

Tabulka 5.4: Celkové chyby po 1. - 5. dělení

	Rovnoměrné	Nerovnoměrné
1. iterace	1861	1861
2. iterace	1427	1427
3. iterace	1201	1201
4. iterace	1020	1023
5. iterace	873	869

Z tabulky a vizualizace dělení je zřejmé, že první tři iterace byly totožné a zbylé dvě téměř shodné. Může se stát, že při dělení třídy s větší chybou dostanu v součtu chyb následníků menší hodnotu než při rozdělení shluku s menší chybou. Díky tomuto jevu ve 4. iteraci dosáhlo rovnoměrné dělení lepšího výsledku než nerovnoměrné dělení. Výsledky jsou si velmi blízké, takže i vznikly podobné modely a následný test rozpoznávání dopadl rovněž téměř totožně.

Rozdíly ve výsledcích jsou zanedbatelné, avšak nerovnoměrné dělení má díky zjišťování aktuální největší chyby větší výpočetní náročnost. Proto pokud bych měl mezi metodami rozhodnout, přiklonil bych se k rovnoměrnému dělení.

Pokud však srovnám rovnoměrné dělení metodou k-means s EM-algoritmem, výsledky se razantně liší. EM algoritmus dosáhl na slovech zhruba o 8-9% větší úspěšnosti. Nevýhodou EM algoritmu je jeho velká výpočetní náročnost, protože hledání parametrů gaussových křivek není jednoduchá záležitost. Pokud bych měl vybrat mezi EM-algoritmem a rovnoměrným dělením k-means, záleželo by hlavně na definici úlohy, požadavcích a prostředcích, jaké jsou pro řešený problém dány.

6. Závěr

Cílem této práce bylo nastudovat problematiku rozpoznávání řeči a hlavně tvorbu akustického modelu. Seznámit se s prostředím HTK toolkitu, otestovat závislost počtu složek směsi na úspěšnosti rozpoznávání a na základě tohoto pozorování zvolit vhodný počet složek a následně sestavit alternativní cestou vlastní akustický model.

Postupným přidáváním složek jsem zjistil, že z počátku se úspěšnost razantně zvyšuje, ale s přibývajícími složkami se nárůst úspěšnosti menší. Na základě tohoto zjištění jsem usoudil, že při vytváření vlastního akustického modelu mi 32 složek poskytne dostatečně přesný popis.

Zparametrizovaným datům jsem pomocí HTK přiřadil foném, který reprezentují, a stav tohoto fonému. Podle tohoto označení jsem vektory řečových dat roztřídil do souborů. Každý soubor tedy obsahoval vektory trénovacích dat odpovídající jednomu konkrétnímu stavu fonému.

Když jsem si data připravil, přešel jsem k získávání parametrů. Nad každým souborem vektorů jsem provedl rovnoměrné a nerovnoměrné dělení metodou k-means do cílového počtu 32 složek. Pro každou složku jsem vypočítal příslušné parametry potřebné k sestavení modelu. Parametry jsou apriorní pravděpodobnost, střední hodnota, kovarianční matice a g-konstanta, která sice složku přímo nepopisuje, ale slouží hlavně k urychlení výpočtů při užívání modelu.

Při popisování jsem narazil na problémy vzniklé nedostatkem trénovacích dat, díky čemuž některé popisy neměly požadovaných 32 složek, ale méně. Tento problém byl způsoben počtem vektorů menším než 32. Na tento nedostatek navazuje i jev, kdy po dělení vznikla pouze jednoprvková množina vektorů. Z takové složky nelze vypočítat kovarianční matici, proto byla v těchto případech použita výchozí kovarianční matice používaná HTK.

Po získání parametrů jsem na základě modelu vytvořeným HTK sestavil vlastní akustický model. Sestavování proběhlo po vzoru již sestaveného modelu, tedy četl jsem řádek po řádku, formát jsem zachovával a původní hodnoty jsem nahrazoval vlastními. Takto jsem sestavil modely jak pro nerovnoměrné, tak pro rovnoměrné rozdělení.

V závěru jsem všechny tři modely porovnal. Modely vytvořené na základě rovnoměrného a nerovnoměrného dělení metodou k-means měly téměř totožné výsledky. Slova byla rozpoznána s úspěšností 73.55% a 74.21%. To zapříčinilo hlavně rozložení dat v prostoru, povaha metod a dělení pouze do 32 složek, což znamená pouze pět iterací. Oproti tomu model vytvořený HTK byl na úrovni slov úspěšný na 82.00%.

Tato práce byla pro mě velmi přínosná, jelikož mi ucelila znalosti problematiky rozpoznání řeči, hlavně oblasti tvorby akustických modelů. Prakticky jsem si vyzkoušel vytvořit vlastní model a aplikovat jej na testovací data. Narazil jsem na řadu problémů a byl nucen nalézt nějaké schůdné řešení. Jedním z těchto problémů byl nedostatek dat, na který jistě v budoucnu ještě narazím.

7. Literatura

[1] J. Psutka, L. Muller, J. Matoušek, V. Radová *Mluvíme s počítačem česky*. Praha: Academia, 2006. ISBN 80-200-1309-1

[2] HTK, *Hidden Markov Model Toolkit* [online]
Dostupné z <http://htk.eng.cam.ac.uk>

[3] J. V. Psutka, *Trénování akustických modelů pomocí HTK*.
Plzeň, 2013. Skripta. KKY ZČU v Plzni.

[4] J. Psutka, *Strojové učení, řešení úloh a rozpoznávání*.
Plzeň, 2013. Skripta. KKY ZČU v Plzni.

8. Seznam tabulek

9. Seznam obrázků