Využitie princípu MDL v strojovom učení

BAKALÁRSKA PRÁCA

Michal Klempa

Univerzita Komenského v Bratislave
Fakulta Matematiky, Fyziky a Informatiky
Katedra Informatiky

9.2.1 Informatika

Školiteľ: Mgr. Ľuboš Steskal

Bratislava, 2009

Čestne prehlasujem, že som túto prácu vypracoval samostatne s použitím citovaných zdrojov.

.....

Abstrakt

KLEMPA, Michal. *Využitie princípu MDL v strojovom učení* [bakalárska práca]. Univerzita Komenského v Bratislave. Fakulta Matematiky, Fyziky a Informatiky; Katedra Informatiky. Školiteľ: Mgr. Ľuboš Steskal. Bratislava, 2009. 30 s.

V práci sa venujeme zhlukovaniu a následne ad-hoc kódovaniu melodických obálok viet slovenského jazyka. Cieľom práce je nájsť základné typy priebehov melodických obálok viet pre potreby syntézy reči. Prostredníctvom ad-hoc kódov (ne)pravdepodobnostného MDL hľadáme globálne minimum kódových dĺžok pre zhlukované melodické obálky viet. Výsledkom práce je návrh na optimálne počty a typy melodických obálok viet získaných zo zdrojových dát.

Kľúčové slová MDL, ad-hoc coding, clustering, pitch contour

Obsah

Zoznam obrázkov

Kapitola 1

Úvod

Aby naše pokusy s MDL neboli len na generovaných umelých a celkom iste aj zbytočných dátach, kontaktovali sme kolegov z Katedry informačných sietí Žilinskej univerzity, ktorí sa venujú vývoju softvéru na syntézu reči v slovenčine. Ide o text-to-speech systém, založený na konkatinatívnej syntéze reči. Systém najprv z krátkych rečových úsekov poskladá slová a vety. Na spojené úseky sa následne aplikujú prozodické vlastnosti: obálky melódie, hlasitosti. V tejto práci sa zaoberáme analýzou melódie viet v slovenskom jazyku.

Pre účely syntézy reči chceme poznať, aké rôzne melódie viet sa v slovenskom jazyku používajú najčastejšie. Kolegovie v Žiline by chceli zistiť aj to, či náhodou neexistuje súvis medzi textovým zápisom vety a melódiou, akou ju človek prečíta. Hlavne sa zameriavajú na také znaky viet, ktoré sa dajú z písaného textu získať pri syntéze: interpunkčné znamienka, počty slov a slabík.

Z nahrávok zvukov boli najprv kolegami zo Žiliny získané priebehy melódií viet ([ČŠ]). Priebeh melodickej obálky vety je funkcia základnej frekvencie hlasu F_0 človeka od času ([PPTL98]). Kolegovia dáta ešte upravili vyhladením plávajúcim váženým aritmetickým priemerom a upravili na jednotný diskrétny definičný obor (8 hodnôt frekvencie na slovo vo vete). S týmito dátami sa pokúšali o zhlukovanie.

Rýchlo však narazili na problém: Koľko zhlukov vlastne existuje v skupine dát? Na tento problém odpovedáme v tejto práci s využitím princípu *MDL* (minimum description length). Najprv dáta zhlukujeme algoritmom *K-means* a následne skúmame pre každý počet zhlukov, ako dobre sa dajú dáta skomprimovať s využitím informácie o stredoch zhlukov (ako dobre stredy zhlukov vyjadrujú regularitu dát). Venujeme sa verzii *Crude MDL* ([Grü07, kap.5]), pri ktorej je kód rozdelený na dve časti: kód modelu a kód dát. Model v našom prípade je počet a stredy zhlukov a pomocou nich kódujeme (komprimujeme) dáta. Očakávame, že pre malý počet zhlukov bude kód pre dáta dlhý a pre model krátky; s pribúdajúcim počtom zhlukov sa bude kód modelu zväčšovať (musíme zakódovať viac stredov) a kód dát zmenšovať (dáta su vo všeobecnosti bližšie k svojim stredom klastrov) až časom kód modelu bude priveľký a kód dát už nebude signifikantne klesať. Počet zhlukov, pre ktorý nastane globálne minimum súčtu dĺžok kódov pre model a dáta, označíme za „skutočný“ počet zhlukov existujúci v dátach.

Naše kódy su navrhnuté ad-hoc (ako vždy v prípade Crude MDL), metódy teda nemusia nutne dať správny alebo aspoň pozorovateľný výsledok. Pri troche šťastia môžeme trafiť dobrý kód,

ktorý využije stredy zhlukov na zakódovanie celej série dát, skomprimuje tak dáta a ukáže nám, aký počet zhlukov je „správny“.

Pre účely aplikácie melódie viet do syntézy reči, budú potom kolegovia zo Žiliny v týchto zhlukoch hľadať spoločné znaky v texte viet. Riešenie problému má aj pragmatické obmedzenie: nechceme zistiť, že v dátach existuje príliš veľa zhlukov. Je rozumné predpokladať, že aj s malým počtom rôznych obálok viet, môže syntetizátor znieť hodnoverne. Navyše, veľa zhlukov by znamenalo veľa práce pri analýze podobností textov a veľa práce pri programovaní rozhodovacieho algoritmu v syntetizátore. Tento problém rieši MDL priamo vo svojej podstate a v Crude forme je len na nastavení parametrov, ako veľmi preferuje jednoduché modely, pred zložitými.

Kapitola 2

Melodická obálka vety

Človek produkuje zvuky za pomoci svojich hlasiviek. Tie sú svalmi napínané na rôznu dĺžku a napätie. Tón vzniká prechodom vzduchu cez hlasivkovú štrbinu, pričom sa hlasivky rozkmitajú na svojej rezonančnej frekvencii (danej dĺžkou a napätím). Tento tón sa ďalej mení, zosilňuje a zoslabuje podľa otvorenosti a zatvorenosti dutín a pier. Vo výsledku interferujú dva posunuté signály: z nosa a z úst. [Ptá92, kap.8]

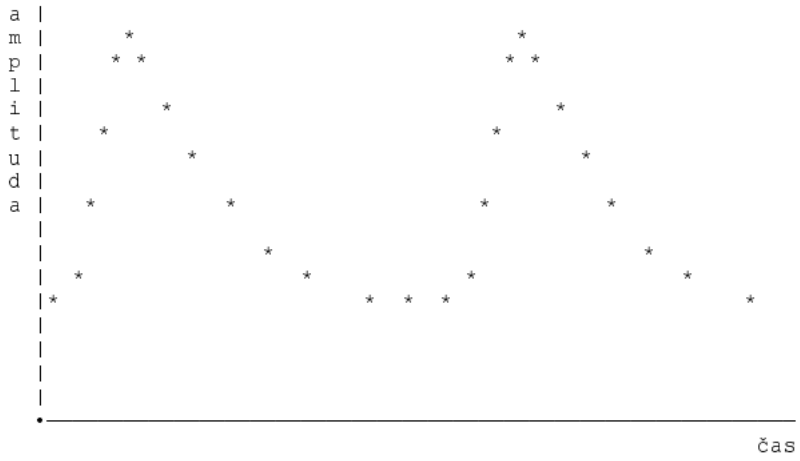
Každý človek disponuje svojim základným hlasivkovým tónom (frekvenciou). Základný tón hlasiviek človek v priebehu reči mení, mimovoľne podľa emočného stavu, ale aj cielene pri prednese. Tvar vlny tohto tónu bol približne určený (obrázok 2.1). Melodická obálka vety je daná zmenou základného tónu v čase ([PPTL98]). Pri vyjadrovaní otázky zvyčajne rečník vetu končí zvyšovaním tónu hlasu, teda zvyšovaním svojej základnej hlasivkovej frekvencie F_0. Pri iných vetách môže byť priebeh frekvencie F_0 iný. Snažíme sa identifikovať priebehy frekvencie F_0 rečníka.

Ako dáta na tento účel slúži nahrávka knižky Cukor a soľ rečníčkou. Nahrávka je rozstrihaná na vety, pričom za vetu sa považuje úsek medzi interpunkčnými znamienkami. Teda aj súvetia sú rozdelené na dve vety.

Keďže vlna hlasiviek má svoj typický tvar kopca a následnej dohry, je možné v zvukovej nahrávke hľadať takéto vzory a periódy. Na identifikáciu základnej hlasivkovej frekvencie v časových bodoch nahrávky vety použili kolegovia zo Žiliny program Praat. Na vypočítané priebehy hlasivkovej frekvencie v čase potom aplikovali vyhladzovanie, keďže pre veľký šum sa niektoré hodnoty (dĺžky periód) výrazne líšili od pomerne stáleho priebehu ostatných. Za melódiu vety sa potom považuje diskrétna funkcia základnej hlasivkovej frekvencie F_0 rečníka v čase. ([ČŠ])

Ešte pred samotným zhlukovaním a poskytnutím dát na naše spracovanie, vykonali kolegovia redukciu dimenzie dát, kedy z rádovo miliónov hodnôt funkcie melódie, na ktoré nemáme kapacity na výpočtove spracovanie, získali priemerovaním presne 8 hodnôt na jedno slovo vo vete. Vety sú už delené podľa počtu slov, čo nepovažujeme za najlepší prístup. Na skutočné odhalenie zákonitostí melódií viet, by sme sa radšej mohli pozerať na všetky vety s rôznou dĺžkou, akurát počet vzoriek (dimenzia) by sme si určili dostatočne veľký, aby zachytil priebeh dlhej vety aj krátkej vety. Protiargument našich kolegov zo Žiliny je, že tak by sme možno priveľmi zachytávali aj melodické zmeny vo vnútri slov a nie celkový melodický charakter vety. Obmedzenie tohto triedenia dát je dané.

Obrázok 2.1: Približný tvar vlny hlasivkového tónu, zdroj [Pta92, s.54]



Podľa nášho názoru ešte musíme poznamenať, že aj po úspešnom zhlukovaní dát a následnom využití v text-to-speech systéme je možné, že nedôjde k optimálnemu výsledku. Keďže text je nahovorený len jedným rečníkom, nedá sa určiť, ktoré zmeny melódie sa dajú prisúdiť zákonitostiam slovenského prejavu a ktoré zmeny melódie do prejavu vkladá rečníčka ako vlastné špecifikum. Mohlo by sa stať, že zvolená miera redukcie dimenzie nevystihuje zákonitosti slovenského prejavu, ale zachytáva aj prejav rečníčky. Naivne predpokladáme, že systém bude prejavom podobný rečníčke.

Pri syntéze reči prebehne proces melodizácie prejavu. Na vetu, poskladanú zo základných difém, syntetizátor aplikuje jednu z preddefinovaných melódií. To znamená, že v niektorých časových úsekoch dĺžky periód roztiahne, inde zúži, tak, aby melódia vety zodpovedala preddefinovanej.

Kapitola 3

Teoretická príprava

3.1 Zhlukovanie

3.1.1 Dáta

Dané dáta sú v tvare diskrétnych funkcií frekvencie hlasu v čase. Ibaže čas v tomto prípade nadobúda hodnoty prirodzených čísel $t \in 1, 2, 3, ...$ Pretože pre n slovnú vetu bolo vyčíslených presne $8n$ vzoriek, nie je dôležité ako si hodnoty času označíme. Každá jedna veta v knižke má svoje presné pomenovanie (taxonomické zaradenie) v tvare: *nov_01s0001v0001t04c_b_*. Čo znamená, že ide o novelu číslo 1, na strane 1 knižky, veta v poradí prvá, má štyri slová a začína čiarkou a končí bodkou.

Dáta (priebehy melódií) sú rozdelené podľa počtu slov do zložiek a následne aj čiastočne podľa toho, akým znamienkom končia - do súborov s názvom v tvare *1_o_obalky_f0_vyhladene.csv*. Formát súboru je štandardný - hodnoty oddelené bodkočiarkou: názov vety; $y_1; y_2; \ldots; y_8 n$ (n je počet slov vo vete). Prvé číslo v názve dátového súboru znamená počet slov viet, ktoré sú v ňom uložené, písmeno značí interpunkčné znamienko, ktorým sa veta končí. Keďže chceme skúmať či sa náhodou dáta nerozpadnú do zhlukov podľa znamienok, najprv sme všetky dáta v jednej zložke (rovnaký počet slov) spojili do jedného súboru, kde sme písmeno označujúce znamienko, ktorým veta končí, zamenili za písmeno x. Na týchto dátach budeme robiť naše experimenty.

3.1.2 Zhlukovací algoritmus K-means

Na rozdelenie dát do zhlukov používame algoritmus *K-means*, pre jeho jednoduchosť. Algoritmus K-means má na vstupe jeden parameter (okrem dát na ktorých pracuje): k - počet stredov (zhlukov). Pojmom zhluk označujeme množinu dát prislúchajúcu jednému stredu. Dáta sa snažíme rozdeliť do k zhlukov, na základe metriky, ktorú si na dátach definujeme (v našom prípade štandardná euklidovská). Stredom zhluku je jeho ťažisko, dáta v zhluku sú tie, pre ktoré je daný stred najbližší zo všetkých k stredov.

Algoritmus K-means nájde stredy (ťažiská) zhlukov aj rozdelenie dát (bodov) do zhlukov.

Vstup: Číslo *dim* predstavujúce dimenziu bodov. Číslo $N \in \mathbb{N}$ reprezentujúce počet dát. Postupnosť dát $D = \{P_i = (y_1, \ldots, y_{dim})\}_{i=1}^{N}$. Číslo $k \in \mathbb{N}$, reprezentujúce počet zhlukov.

Výstup: Množina R pozostávajúca z $N \cdot k$ prvkov. r_{ij}. Prvok r_{ij} vyjadruje, či bod P_i patrí do zhluku j (so stredom $\boldsymbol{\mu}$).

$$r_{ij} = \begin{cases} 1 & P_i \text{ patrí do zhluku } j \\ 0 & \text{inak} \end{cases}$$

Postupnosť stredov zhlukov $S = \{\boldsymbol{\mu}_j\}_{j=1}^{k}$.

Algoritmus: Definujme si celkovú chybovú funkciu J:

$$J = \sum_{n=1}^{N} \sum_{j=1}^{k} r_{ij} ||P_i - \boldsymbol{\mu}_j|| \tag{3.1}$$

kde $\boldsymbol{\mu}_j$ je j-ty stred. Funkcia J vyjadruje celkovú chybu rozdelenia do zhlukov, je súčtom všetkých euklidovských vzdialeností bodov od svojich stredov.

Cieľom je nájsť také priradenie bodov k stredom a také stredové body, aby funkcia J bola čo najmenšia. V inicializačnom kroku algoritmu vyberiemie k navzájom rôznych, náhodných reprezentantov z dát, ktoré označíme za prvotné stredy zhlukov.

Ďalej striedame dva kroky:

1. priradenie bodov k najbližším stredom
2. nájdenie nových stredov

V 1. kroku pre každý bod P_i nájdeme najbližší zo stredov. Nastavíme nové priradenie bodov do zhlukov:

$$r_{ij} = \begin{cases} 1 & \text{ak } j = \arg\min_k ||P_i - \boldsymbol{\mu}_k||^2 \\ 0 & \text{inak} \end{cases}$$

V 2. kroku pre každý zhluk nájdeme nové stredy, pričom stredy vyrátame ako ťažisko bodov zhluku:

$$\boldsymbol{\mu}_j = \frac{\sum_{i=1}^{N} r_{ij} \cdot P_i}{\sum_{i=1}^{N} r_{ij}}$$

Následne kroky striedame, až kým sa pridelenie do zhlukov a polohy stredov nemenia. V každom kroku algoritmu sa hodnota chybovej funkcie J zmenší, takže algoritmus konverguje. Nie je však isté, že skonverguje do globálneho minima, preto ho treba zbiehať vo viacerých iteráciách.

3.2 MDL

3.2.1 Učenie je kompresia

Princíp Minimum description length (minimálna popisná dĺžka, skrátene MDL) sa snaží riešiť problém predikcie (odhadu) budúcich dát na základe už prijatých dát (strojové učenie). Na „dobré“ odhadnutie budúcich dát sa snažíme nájsť v dátach zákonitosti. Napríklad sa pozrime na nasledujúce postupnosti dát:

$$1\ 2\ 4\ 8\ 16\ 32\ 64\ 128\ 256\ 512\ 1024\ 2048\ 4096 \tag{3.2}$$

1 11 21 1112 3112 211213 312213 212223 114213 (3.3)

0 1 0 0 1 0 1 1 1 0 1 1 1 0 0 1 0 0 0 1 0 0 1 0 0 (3.4)

V postupnosti 3.2 ihneď vidíme jej zákonitosť: je generovaná postupným násobením predošlého čísla dvojkou. Preto náš prirodzený odhad nasledujúcich dát je, že budú dodržovať túto zákonitosť.

Postupnosť 3.3 má tiež svoju zákonitosť, hoci nie na prvý pohľad viditeľnú. Riadi sa jednoduchým pravidlom: ďalšie číslo získame popísaním predošlého detským spôsobom „Koľko jednotiek je v predošlom čísle? Jedna jednotka, zapíšeme 11. Koľko dvojok? Trojok?“

Posledná postupnosť je generovaná hodmi mincou, a akékoľvek naše snahy objaviť v nej zákonitosť (ak je dostatočne dlhá) musia zlyhať [Grü07, p. 103].

Ako však odhaliť regularitu postupnosti ak nie sme človek, ale počítač? Podľa princípu MDL, vieme každú zákonitosť postupnosti dát využiť na to, aby sme postupnosť lepšie skomprimovali.

Pozrime sa opäť na postupnosti 3.2 a 3.3. V obidvoch je zákonitosť rekurzívna a teda sa dajú komprimovať jednoducho, zapamätaním si iba poradového čísla prvku. Ak teda máme komprimovať napríklad prvých stotisíc prvkov postupnosti, stačí si pamätať iba číslo 100 000 a nejaký popis zákonitosti (že pre prvky x_i postupnosti platí $x_i = 2^i$).

Ak vieme postupnosť dobre komprimovať, našli sme v nej nejaké zákonitosti, ktoré potom použijeme na odhad nasledujúcich dát.

Univerzálny jazyk, ktorým vieme opísať aj číslo 100 000 aj zákonitosť postupnosti, je ľubovoľný programovací jazyk. Stačí napísať program, ktorý vypíše prvých 100 000 prvkov postupnosti a potom skončí. Takých programov existuje nekonečne veľa a iste je z nich niektorý (alebo niekoľko) najkratší (napríklad v miere počtu znakov kódu). Dĺžku najkratšieho takého programu nazývame *Kolmogorovská zložitosť* postupnosti.

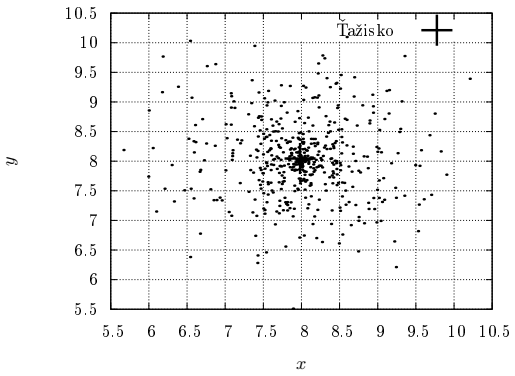
Najlepším odhadom zákonitostí v postupnosti je potom program s najmenšou Kolmogorovskou zložitosťou. Avšak vo všeobecnosti najkratší program, ktorý vypíše našu postupnosť a skončí nevieme algoritmicky nájsť. Preto v praktickom MDL používame omnoho obmedzenejšie formy popisu dát postupnosti. Formou popisu dát máme na mysli *kód*, špeciálne pre naše úvahy myslíme dekódovateľný kód. Je len na nás, čo označíme za dĺžku popisu jedného dáta, podľa toho, aký kód zvolíme. Ak je to výhodné pre výsledky bádania, môžeme za dĺžku popisu považovať napríklad dĺžky vektorov alebo veľkosť alokovanej pamäte v ktorej si dáta pamätáme, prípadne iné extravagantné veličiny.

3.2.2 Hypotéza

Pozrime sa na nasledujúci problém: Na vstupe máme dáta vo forme usporiadaných dvojíc $P_i = (x_i, y_i)$. Chceme odhadnúť hodnoty nasledujúcich dát. Máme k dispozícii N doteraz prijatých dát (na obrázku 3.1). V obrázku 3.1 sme dáta interpretovali ako body roviny. Vidíme, že rozloženie bodov v rovine nie je náhodné. Body sa zhlukujú v okolí svojho ťažiska s nejakou priemernou vzdialenosťou. Ak chceme odhadnúť pozíciu nasledujúceho bodu, je rozumné predpokladať, že bude takisto blízko ťažiska už prijatých bodov a že jeho vzdialenosť od ťažiska nebude príliš rozdielna od priemernej.

Ak máme navrhnúť algoritmus, ktorý bude prijímať dáta z rovnakého zdroja, a má odhadovať hodnoty ďalších dát, použijeme na to predpoklad: dáta (body) „padajú“ do kruhového okolia

Obrázok 3.1: N doteraz prijatých dát



ťažiska T a udržujú si istú, „malú“ vzdialenosť od ťažiska.

Považujme dĺžku vektora $P_i - \overline{0}$ za dĺžku popisu dáta (bodu), pri našej snahe skomprimovať dáta by sme sa logicky snažili zmenšiť hodnoty x-ových a y-ových súradníc bodov. Keďže body sa zhlukujú okolo ťažiska, využijeme túto zákonitosť na kompresiu. Miesto zapisovania sekvencie dát kódom zo zadania (dvojice súradníc), rozdelíme kód na dve časti:

- Kód hypotézy
- Kód dát

Pojem *hypotéza* označuje zákonitosť dát. Slovo hypotéza je namieste, pretože nepoznáme skutočný zdroj dát (skutočný generátor). Vyslovili sme iba hypotézu, že dáta sa vyskytujú v okolí ich ťažiska.

Skúmajme dáta hlbšie. Na základe histogramu vzdialeností bodov od ťažiska na obrázku 3.3 vyslovíme ďalšiu hypotézu, že histogram vzdialeností bodov od stredu zodpovedá normálnemu rozdeleniu pravdepodobnosti. Zaveďme v rovine polárnu sústavu súradníc s počiatkom T a nulovým smerom $P_1 - T$. Na základe histogramu uhlov bodov od nulového smeru na obrázku 3.2 môžeme vysloviť aj hypotézu, že rozloženie uhlov bodov zodpovedá rovnomernému rozdeleniu pravdepodobnosti.

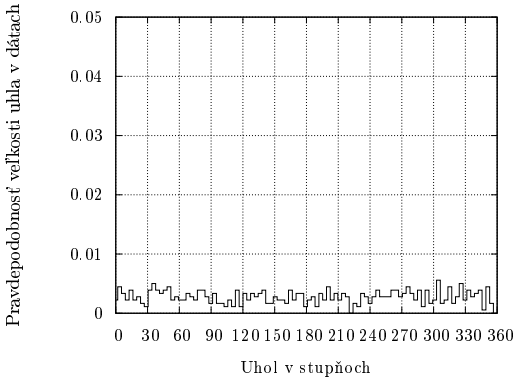
Pre kompresiu použijeme spojenú hypotézu o dátach: Dáta sa správajú tak, akoby boli generované s nejakým pevným stredom, v polárnej sústave súradníc s rovnomerne náhodne vybraným uhlom a podľa normálneho rozdelenia náhodne vybranou vzdialenosťou od stredu.

Keďže nepoznáme generátor dát, iba sledujeme a analyzujeme, ako sa dáta správajú, nemôžeme vedieť, či dáta sú naozaj generované podľa našej hypotézy; a keby aj, nemôžeme vedieť, aký stred sa pri generovaní používa, aký parameter variancie normálneho rozdlenia sa používa. Môžeme však tieto parametre, pre účely kompresie, získať z doteraz prijatých dát (ťažisko aj varianciu vieme z dáť poľahky zrátať).

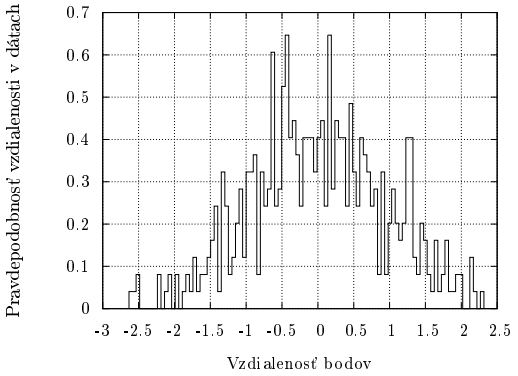
Naša spojená hypotéza má teraz dva parametre: stred, variancia. Kód pre hypotézu bude kódovať tieto dva parametre.

Kód pre dáta využije zákonitosti: stačí kódovať uhol a vzdialenosť od stredu v polárnych súradniciach. V časti 3.2.6 uvedieme ako využiť na kompresiu pravdepodobnostné rozdelenie.

Obrázok 3.2: Histogram uhlov dát



Obrázok 3.3: Histogram vzdialeností dát od ťažiska



Poznámka: Pre úplnosť by sme mali kódovať aj znenie hypotézy. Pre účely MDL však toto nie je potrebné. Akýkoľvek kód tejto informácii pridelíme, jeho dĺžka nebude závisieť od počtu dát a teda je pre účely odhadnutia zákonitosti v dátach irelevantná. Môžeme predpokladať, že enkodér aj dekodér sú špecializované na našu hypotézu.

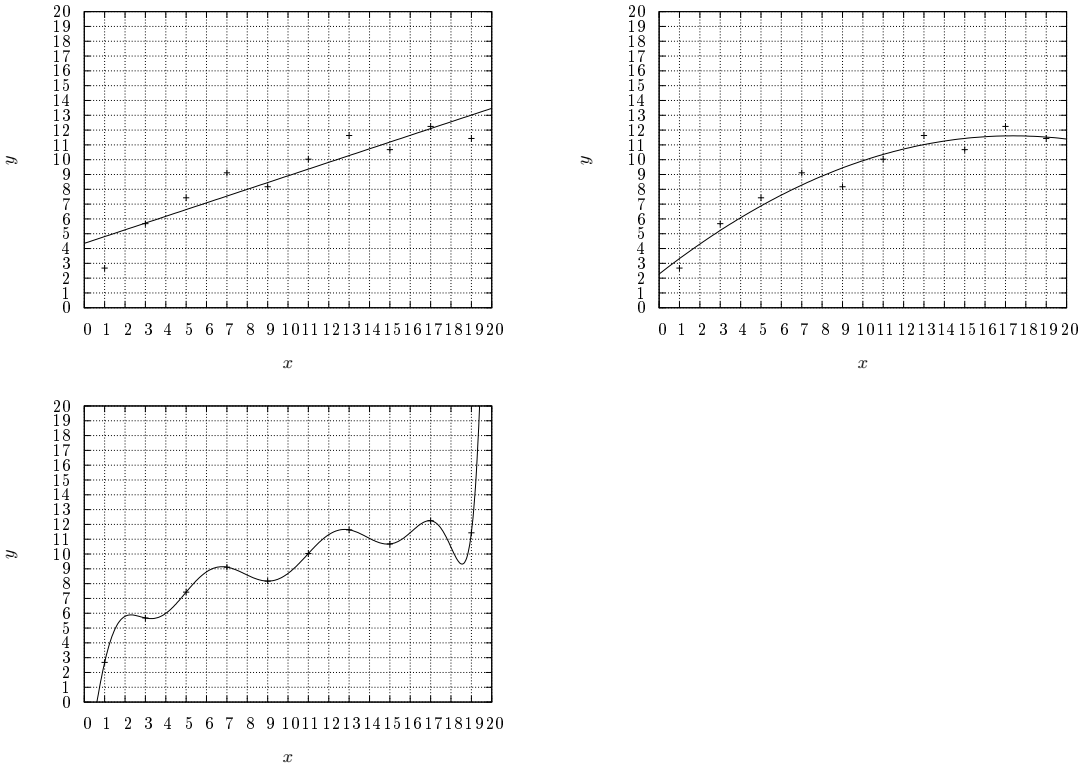
Vyvstáva však problém s kódom pre parametre hypotézy. Ak miera dĺžky kódu je dĺžka vektora, ako kódovať varianciu? Jedno z riešení je zvoliť univerzálny jazyk pre hypotézu aj dáta - bitový kód. Aj tak však musíme kód hypotézy ladiť ad-hoc tak, aby sme dosiahli dobré výsledky. Viac sa tomuto problému venujeme pri jednotlivých metódach, ktoré sme na reálne dáta použili.

3.2.3 Model

Hypotéze s konkrétne dosadenými parametrami hovoríme *bodová hypotéza*. Pojmom *model* označujeme množinu *bodových hypotéz*, spravidla viazanú spoločnou charakteristikou jej členov - napríklad rovnaký počet parametrov.

Napríklad v prípade dát reprezentovaných polynómami, môžeme za hypotézu označiť „polynóm n-tého stupňa“. *Parametre hypotézy* sú koeficienty polynómu. Jednotlivé modely sú: polynómy 1.

Obrázok 3.4: Overfitting: funkcia $3 + 2\log_2(x)$ + šum, regresia metódou najmenších štvorcov: lineárna, polynóm 2-stupňa, polynóm 10-stupňa



stupňa, 2. stupňa, 3. stupňa... A všetky polynómy (bodové hypotézy) tvoria *triedu modelov*.

V príklade z predošlej časti by sme za hypotézu mohli prirodzene považovať nami spojenú hypotézu a model z hry úplne vynechať. Rozšírme však tento príklad o možnosť, že dáta sa zhlukujú v k zhlukoch, teda okolo k rôznych stredov. Pojmom hypotéza označíme „dáta sa zhlukujú okolo k stredov“. Bodová hypotéza je množina konkrétnych k bodov roviny (stredov). Modelom je číslo k, teda počet stredov, ktorými možno dáta dobre reprezentovať.

3.2.4 Overfitting

Dostávame sa k druhému problému, ktorý sa MDL snaží riešiť. *Overfitting* je názov pre situáciu, kedy dáta našou hypotézou vyjadrujeme až tak presne, že nezachytávame zákonitosť dát, ale aj šum v dátach. Príklad s polynómami je na obrázku 3.4. V prípade, že zvolíme príliš presnú, komplexnú hypotézu (model), môžeme dosiahnuť síce nulovú chybu v reprezentácii dát – a teda kompresiu na úrovni dĺžky kódu hypotézy, bez akéhokoľvek kódu pre dáta – ale naša hypotéza zrejme zlyhá v predpovedaní ďalších dát.

Problém overfittingu sa MDL snaží riešiť svojou podstatou: pri malej komplexnosti modelu (krátky kód pre hypotézu) je pravdepodobné, že dáta nevystihuje tak dobre; s rastúcou komplexnosťou modelu rastie dĺžka kódu pre hypotézu, ale klesá dĺžka kódu pre dáta, pretože komplexnejší model (napr. polynóm vyššieho stupňa) dáta lepšie vystihne; v nejakom momente rastúca komplexnosť modelu zabezpečí opätovný nárast súhrnnej dĺžky kódu. Tak vznikne globálne minimum

súhrnnej dĺžky kódu, a je len na dizajnérovi kódov, aký *trade-off* medzi komplexnosťou modelu a kompresiou dát potrebuje.

3.2.5 Štandardné kódy

Pre ďalšiu prácu potrebujeme tri štandardné kódy: *uniformný*, *Shannonov*, *kód pre celé čísla*.

Začnime uniformným kódom. Ak nejaké dáta alebo iné kódované veličiny nadobúdajú každú z N hodnôt s pravdepodobnosťou $\frac{1}{N}$, najvýhodnejšie je zakódovať ich pomocou uniformného (rovnomerného) kódu. Každá hodnota dostane rovnako dlhý kód, pretoža každá hodnota má rovnakú pravdepodobnosť výskytu v dátach. Optimalizujeme tak najkratší worst-case, teda aby vždy v tom najhoršom prípade sme mali čo najkratší kód. Máme N rôznych hodnôt, takže kód pre každú hodnotu bude mať dĺžku $\lceil \log_2(N) \rceil$. Čo je mimochodom len a len inštancia Shannonovho kódu.

Shannonov kód nad pravdepodobnostným rozdelením prideľuje kódové slová dĺžky $\lceil -\log_2 P(x) \rceil$, pričom tieto dĺžky sú nanajvýš 1 bit od optimálneho kódu. Konštrukciu kódu aj dôkaz jeho „competetitive optimality“ nájde čitateľ v [CT06, kap.5]. Nám bude stačiť používať dĺžku kódu, a vedieť pri tom, že je najlepší pre potenciálne nekonečnú postupnosť dát.

Celé číslo n kódujeme tak, že na jeden bit uložíme znamienko. Následne pripočítame 1 a číslo $n+1$ zakódujeme (aby sme zachytili aj číslo 0). Zakódovať ho môžeme napríklad tak, že nájdeme jeho vyjadrenie v binárnej sústave. Dĺžka tohto kódu je $L = \lceil \log_2(n) \rceil$. Aby sme dekóderu ešte pred narazením na tento kód povedali, koľko ďalších bitov musí prečítať, aby dostal číslo, zapíšeme pred tento kód ešte sekvenciu L jednotiek a jednu nulu. Kód pre číslo 27 bude: $27 = 11011_2$ to je 5 bitov, takže prefix je 111110, dokopy je kód:

$$111110|11011$$

s dĺžkou:

$$\lceil \log_2 \lceil \log_2(n) \rceil \rceil + 1 + \lceil \log_2(n) \rceil$$

Tento kód môžme rekurzívne asymptoticky zlepšiť. Zakódujme informáciu, že treba prečítať 5 bitov ako číslo v binárnej sústave a až tomuto číslu dáme prefix: 1110|101|11011. Čo je o jeden bit viac ako v predošlom, avšak pre rastúce n je tento kód asymptoticky lepší, jeho dĺžka je

$$\lceil \log_2 \lceil \log_2 \lceil \log_2(n) \rceil \rceil \rceil + 1 + \lceil \log_2 \lceil \log_2(n) \rceil \rceil + \lceil \log_2(n) \rceil$$

Túto fintu budeme opakovať až pokým nedosiahneme dĺžku prefixu 1. Dĺžku takého kódu označíme L^* a získavame ju algoritmicky postupným logaritmovaním.

3.2.6 Definície MDL

Crude two-part MDL

Pojem Crude two-part MDL označuje verziu MDL pracujúcu s kódom zloženým z dvoch kódov: kód pre hypotézu, kód pre dáta. Pričom oba kódy volí dizajnér ad-hoc podľa problému, ktorý rieši. Informatívna definícia Crude two-part MDL adaptovaná podľa [Grü07, p.14]:

Definícia 1 (Crude two-part MDL) *Nech $\mathcal{H}_1, \mathcal{H}_2, \ldots$ sú kandidujúce modely (napr. $\mathcal{H}_\gamma$ je množina polynómov stupňa γ). Bodovú hypotézu (t.j. hypotéza s konkrétne dosadenými parametrami) $H \in \mathcal{H}_1 \cup \mathcal{H}_2 \cup \ldots$, ktorá najlepšie vystihuje dáta, nájdeme minimalizovaním súčtu $L(H) + L(D|H)$, kde*

- *$L(H)$ je dĺžka bitového kódu pre hypotézu*
- *$L(D|H)$ je dĺžka bitového kódu pre dáta s využitím hypotézy H*

Najlepší model vystihujúci dáta D je najmenší (najmenej komplexný) z modelov obsahujúcich najlepšiu bodovú hypotézu H.

Hoci Grünwald obmedzuje svoje postupy na bitový kód, my sa venujeme aj intuitívnym kódom a porovnávame výsledky s bitovými kódmi.

Simplistic two-part MDL

Pre účely zavedenia *simplistic MDL* obmedzíme kódy, ktoré môžeme použiť na zakódovanie dát na prefixové. Vyhneme sa tak riešeniu problémov so znakom ukončovania kódového slova a naše kódy môžu byť slobodne jednorozmerné (prefixový bitový kód). Hypotézy obmedzíme na pravdepodobnostné rozdelenia, tým získame optimálny kód pre dáta na základne pravdepodobnostného rozdelenia s dĺžkou ([CT06, s.127–130])

$$-\log_2 P(D|H)$$

Pričom neceločíselné dĺžky kódov nám nebudú prekážať, pretože v celkovom súčte dĺžok bude rozdiel oproti celočíselným maximálne 1 bit [Grü07, s.97]

Definícia adaptovaná z [Grü07, p.139]:

Definícia 2 (Simplistic two-part MDL) *Nech $\mathcal{M} = \cup_{k\geq 1}\mathcal{M}^{(k)}$ je množina pravdepodobnostných parametrických rozdelení určených k parametrami z k priestorov parametrov $\Theta^{(1)}, \Theta^{(2)}, \ldots$. Ak pre všetky k je množina $\Theta^{(k)}$ kompaktom, diskretizujeme ju uniformnou mriežkou s dĺžkou dielika $\frac{1}{2^d}$ na množinu stredov dielikov $\Theta_d^{(k)}$. Potom kód $L(H)$ pre bodovú hypotézu má dĺžku:*

$$L(H) = kd + L^*(k) + L^*(d)$$

kde kd je počet bitov pre zakódovanie každého parametra na d bitov. Celé čísla k, d kódujeme štandardným kódom pre prirodzené čísla.

Ak aspoň jeden, napr. i-ty, z priestorov parametrov je otvorený alebo neohraničený, diskretizujeme ho uniformnou mriežkov na jeho podintervale $[a_i, b_i]; a_i \in \mathbb{Z}, b_i \in \mathbb{Z}$ s dĺžkou dielika $\frac{b_i - a_i}{2^d}$ na množinu stredov dielikov $\Theta_d^{(k)}$. Nech priestory $\Theta^{(1)}, \ldots, \Theta^{(j)}$, kde $j \leq k$ nie sú kompakty, potom kód $L(H)$ pre bodovú hypotézu má dĺžku:

$$L(H) = \sum_{i=1}^{j} L^*(|a_i|) + L^*(|b_i|) + 2j + kd + L^*(k) + L^*(d)$$

kde pre každý parameter z nekompaktu kódujeme interval možných hodnôt parametrov ako dve celé čísla a_i, b_i pre ktoré pridáme $2j$ bitov ako informáciu o ich znamienkach. Ostatné priestory $\Theta^{(j+1)}, \ldots, \Theta^{(k)}$ diskretizujeme uniformnou mriežkou.

Bodová hypotéza najlepšie vystihujúca dáta je daná parametrom k, a parametrami

$$(\theta^1, \theta^2, \ldots, \theta^k) \in (\Theta_d^{(1)}, \Theta_d^{(2)}, \ldots, \Theta_d^{(k)})$$

ktoré nájdeme minimalizáciou súhrnnej dĺžky kódu:

$$L(H) + L(D|H)$$

Minimalizujeme vzhľadom na parameter presnosti d, resp. aj a_i, b_i (v prípade nekompaktných priestorov parametrov).

Použime definíciu na príklade. Dáta D vyzerajú byť generované (alebo dobre aproximovateľné) pravdepodobnostným rozdelením P, ktoré má funkcionálny tvar s dvoma ($k = 2$) parametrami: α, β. Parameter α je vždy z intervalu $[0, 4]$ (priestor $\Theta^{(1)} = [0, 4]$), parameter β je ľubovoľné reálne číslo (priestor $\Theta^{(2)} = \mathbb{R}$). Priestor $\Theta^{(1)}$ je kompaktom, takže na jeho diskretizáciou použijeme uniformnú mriežku s dĺžkou dielika $\frac{4-0}{2^d} = \frac{4}{2^d}$. Hodnotu parametra α zaokrúhlime na najbližší stred dielika mriežky a zakódujeme pozíciu dielika. Počet dielikov na intervale je:

$$\frac{4-0}{\frac{4}{2^d}} = 2^d \tag{3.5}$$

Takže na zakódovanie pozície dielika potrebujeme d bitov, pretože existuje práve 2^d rôznych pozícií (ktoré zoradíme a pridelíme im kódy v poradí).

Analogicky diskretizujeme aj priestor $\Theta^{(2)}$. Aplikačne nájdeme hranice $[a, b]$, tak, aby celkovú dĺžku kódu minimalizovali, pričom a, b sú celé čísla a dĺžka kódu na ich zakódovanie je

$$L^*(a) + L^*(b) + 2$$

Kapitola 4

Metódy použité na dáta

Popíšeme všetky postupy, ktoré sme s cieľom získať výsledok na dátach odskúšali. V poradí sú to: dve „naivné“ crude formy MDL a následne 4 pravdepodobnostné simplistic MDL metódy kombinované s crude formou.

V prvom rade si musíme odpovedať na otázku: čo je zaujímavé? alebo: čo chceme o dátach vedieť? Základná otázka, ktorú si položili už kolegovia zo Žilinskej univerzity znie: *Koľko rôznych typov melódií sa nachádza v týchto dátach?* Preložené do reči dát je to otázka, aký je „správny“ počet zhlukov v týchto dátach. Preto prvé, čo sme z uvažovania o dátach vylúčili, bolo poradie dát v dátových súboroch. Okrem toho, že nevieme, či je poradie zachované s poradím viet v čítanej knihe, nemáme v dátach nijak zachytený kontext ostatných viet. Vieme iba interpunkčné znamienka, ktorými vety začínajú a končia, nepoznáme väzby medzi vetami (napríklad: tieto dve vety tvoria súvetie, po tejto vete končí odstavec, a základné: táto veta má túto ako predchodcu a túto ako nasledovníka), nepoznáme ani náladu, pocit alebo iné zafarbenie viet závisiace od kontextu v ktorom sa nachádzali. Pre takéto dáta nemá zmysel uvažovať o ich poradí v dátových súboroch, máme jednoducho množinu funkcií definovaných na diskrétnych definičných oboroch.

Čo má zmysel brať do úvahy, je počet stredov zhlukov v dátach, a to, ako vyšší počet využiť na lepšiu kompresiu dát. Formalizujme naše dáta a základné crude MDL, ktoré na ne použijeme.

Dáta Dáta tvoria postupnosť $D = \{\boldsymbol{y}_i\}_{i=1}^{N}$ kde $\boldsymbol{y}_i = (y_{i1}, y_{i2}, \ldots, y_{idim}); y_{i*} \in \mathbb{R}$. Pričom dimenzia dim je daná dátami ako osemnásobok počtu slov vety. Po zhlukovaní algoritmom K-means, s parametrom k - počet stredov, máme aj informáciu o stredoch: $S = \{\boldsymbol{s}_i\}_{i=1}^{k}; \boldsymbol{s}_i = (s_0, \ldots, s_{dim}); s_i \in \mathbb{R}$ a informáciu o príslušnosti jednotlivých dát ku zhlukom: $R = \{r_{ij}\}_{i=1,j=1}^{N,k}$, kde r_{ij}, vyjadruje, či dáto y_i patrí do zhluku s číslom j (a stredom $\boldsymbol{s}_j$).

4.1 Crude MDL

Naše základné MDL pozostáva z využitia zákonitosti dát, že sa zhlukujú pri k stredoch. Nepoznáme ešte správne k, ale pre rôzne k sme algoritmom K-means našli stredy zhlukov S a informáciu o príslušnosti dát k zhlukom. Hypotéza, ktorú máme o dátach je, že pomocou stredov vieme dáta komprimovať tak, že miesto dát si stačí pamätať iba rozdiel dát od stredu. Formálnejšie:

Nech D sú príchodzie dáta, nech k je počet stredov zhlukov, nech S je množina stredov, nech R je množina vyjadrujúca príslušnosť dát ku zhlukom. Hypotéza H je, že v dátach existuje k stredov. Samotné polohy stredov a číslo k sú parametrami hypotézy. Vzhľadom na číslo k vieme bodové hypotézy rozdeliť do modelov s rovnakým parametrom k. Náš cieľ je vybrať najlepší model, teda množinu bodových hypotéz s konkrétnym k. Na to, musíme podľa definície 1 nájsť pre každý model najlepšiu bodovú hypotézu. Na nájdenie bodovej hypotézy, teda na nájdenie stredov S, pre pevne zvolené k používame opäť MDL (K-means v podstate implementuje myšlienku MDL). Dáta kódované za pomoci hypotézy H majú tvar rozdielu od stredov, t.j.:

$$D|H = \{\boldsymbol{dy}_i\}_{i=1}^{N}; \boldsymbol{dy}_i = \boldsymbol{y}_i - \sum_{j=1}^{k} r_{ij}\boldsymbol{s}_j \tag{4.1}$$

Zvoľme pevne k a hľadajme iba množinu stredov S. Popis hypotézy H zvolíme ako nulový, vieme aké pevné k skúmame. Dĺžka popisu nájdených stredov nie je zaujímavá, pretože hľadáme optimálne stredy na popis dát (nie trade-off stredy). Za dĺžku popisu dát kódovaných pomocou stredov vezmeme euklidovskú dĺžku vektora:

$$L(D|H) = \sum_{i=1}^{N} ||\boldsymbol{dy}_i|| \tag{4.2}$$

Za dĺžku popisu stredov vezmeme nulovú dĺžku. Potom minimalizujeme súhrnnú dĺžku:

$$L(H) + L(D|H) = 0 + \sum_{i=1}^{N} ||\boldsymbol{dy}_i||$$

vzhľadom na parametre - stredy S s pevne zvoleným k. Presne túto úlohu pre nás rieši algoritmus K-means opísaný v 3.1.2.

Takto sme našli a uložili stredy zhlukov. Nastáva čas hľadania správneho modelu. Hypotéza pozostáva z čísla k a množiny S. Najlepšiu bodovú hypotézu pre každé k máme nájdenú, musíme vybrať správne k. Na to potrebujeme určiť nejaké kódovanie dát a hypotézy. Nasledujú rôzne kódovania hypotézy a rôzne spresnenia hypotézy v poradí v akom sme nimi skúšali dáta zdolať.

4.1.1 Metóda filozof

Predstavme si filozofa, ktorý žije v priestore s dimenziou *dim*. Takýto filozof môže vektory tohto priestoru baliť do krabíc a nakladať na ľoď. Priestor, ktorý vektormi zaberie, bude súčet dĺžok týchto vektorov. Kód pre dáta bude kód 4.1. Kód pre stredy navrhnime analogický, najprv nájdeme ťažisko samotných stredov:

$$\boldsymbol{T}_s = \frac{1}{k}\sum_{i=1}^{k} s_i$$

a potom stredy kódujeme analogicky k 4.1:

$$\{\boldsymbol{s}_i - \boldsymbol{T}_s\}_{i=1}^{k} \tag{4.3}$$

Samotné ťažisko stredov kódujeme ako vektor od počiatku. Číslo k kódujeme ako vektor $(k, 0, 0, 0, \ldots)$. Dĺžka kódu pre hypotézu je:

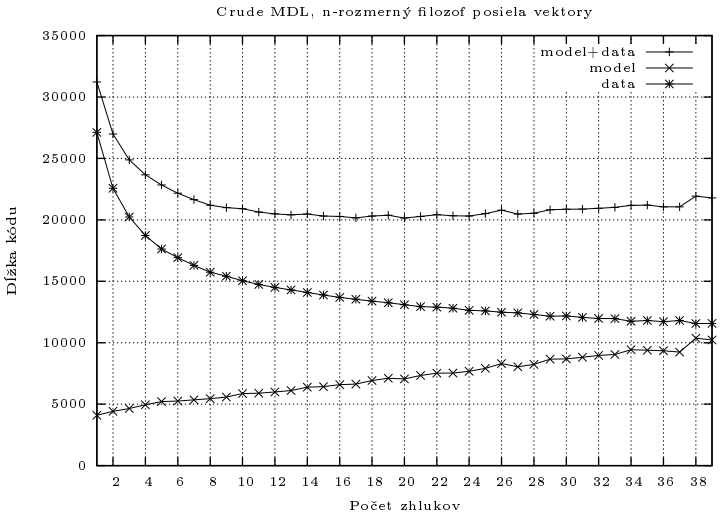
$$L(H) = ||\boldsymbol{T}_s|| + \sum_{i=1}^{k} ||\boldsymbol{s}_i - \boldsymbol{T}_s|| + k \tag{4.4}$$

V súčte je dĺžka kódu

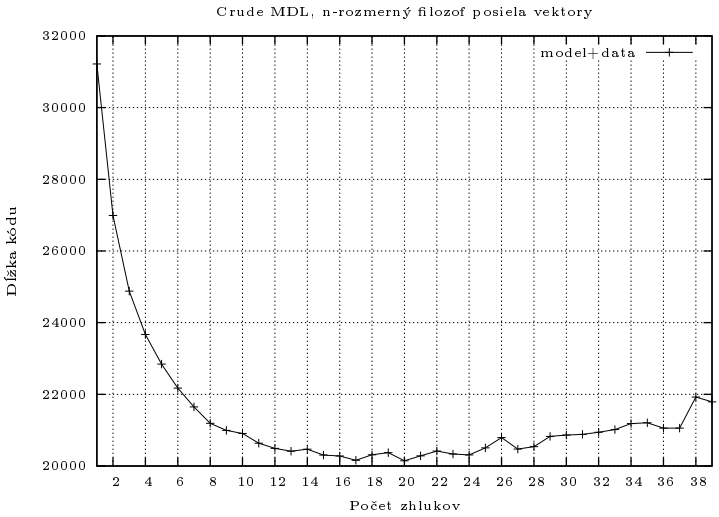
$$||\boldsymbol{T}_s|| + \sum_{i=1}^{k} ||\boldsymbol{s}_i - \boldsymbol{T}_s|| + k + \sum_{i=1}^{N} ||\boldsymbol{dy}_i|| \tag{4.5}$$

S týmto kódom sme dosiahli výsledky ako na obrázkoch 4.1 a 4.2. Ako vidieť, darislo sa mu celkom dobre.

Obrázok 4.1: Výsledok MDL metódy: filozof, na 3-slovných vetách



Obrázok 4.2: Výsledok MDL metódy: filozof, na 3-slovných vetách, súhrnná dĺžka kódu



4.1.2 Metóda jednorozmerný filozof

Analogicky k viacrozmernému filozofovi, ak dáta posiela jednorozmerný filozof, znamená to, že vie vlastne poslať číslo aj so znamienkom a stojí ho to dĺžku absolútnej hodnoty z čísla. K dĺžke hypotézy musíme pridať aj dimenziu dim, aby dekódujúci filozof vedel, po koľko číslach má prúd strihať. Dĺžka kódu bude:

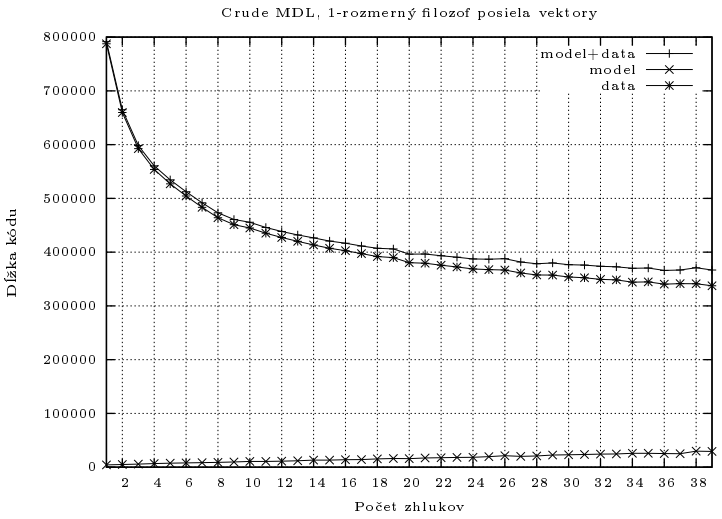
$$|\boldsymbol{T}_s| + \sum_{i=1}^{k} |\boldsymbol{s}_i| + k + dim + \sum_{i=1}^{N} |\boldsymbol{dy}_i| \tag{4.6}$$

kde absolútnu hodnotu vektora počítame ako:

$$|\boldsymbol{y}| = \sum_{j=1}^{dim} |y_j|$$

Tento spôsob kódovania už nefungoval tak dobre, ako je vidno na obrázkoch 4.3 a 4.4

Obrázok 4.3: Výsledok MDL metódy: 1 rozmerný filozof, na 3-slovných vetách



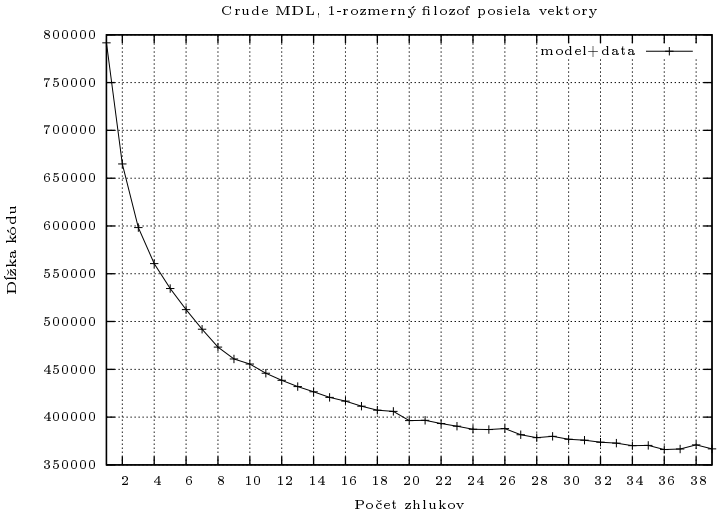
4.2 Simplistic MDL

Keďže prvé dve metódy nás neuspokojili, pustili sme sa do ďalšieho experimentovania s dátami. Z histogramu na obrázku 4.5 sme usúdili, že je možné dáta reprezentovať normálnym rozdelením pravdepodobnosti. Náše základné crude MDL: stredy + dáta rozšírime tak, že na dáta aplikujeme ešte jedno simplistic MDL, a ad-hoc niečo podobné použijeme aj na kódovanie hypotézy crude časti. Spravíme akési zloženie (vnorenie) crude MDL „stredy + dáta“ a simplistic MDL „dáta pochádzajú z normálneho rozdelenia“. Dáta budeme kódovať podľa definície 2.

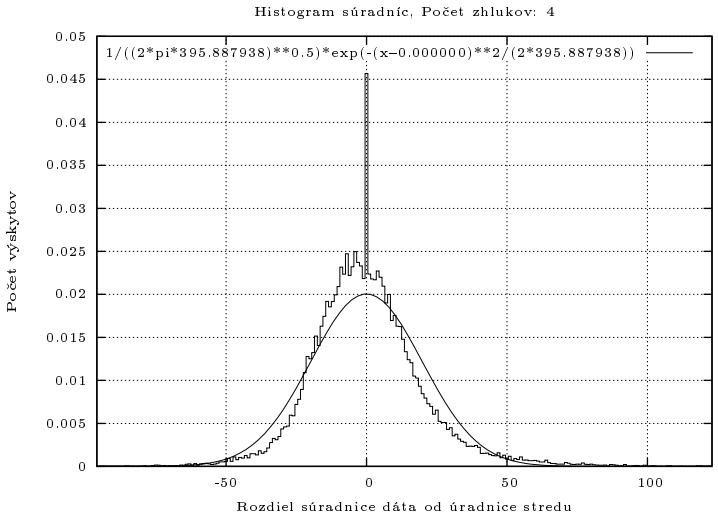
4.2.1 Metóda Gauss

Máme dáta v tvare 4.1, zabudnime na to, že sú to vektory, zapíšme vektory za seba a pozerajme na postupnosť súradníc $y_l{}_{l=1}^{N \cdot dim}$. Hypotéza H^g je, že postupnosť čísel je generovanú zdrojom - nor-

Obrázok 4.4: Výsledok MDL metódy: 1 rozmerný filozof, na 3-slovných vetách, súhrnná dĺžka kódu



Obrázok 4.5: Histogram hodnôt súradníc vektorov $\boldsymbol{dy}_i$: na 3-slovných vetách, 4 zhluky



málnym rozdelením pravdepodobnosti. Nájdeme parametre rozdelenia najlepšie reprezentujúceho postupnosť, nájdením priemeru μ a variancie σ^2 postupnosti [Bis06, p.93].

$$\mu = \frac{1}{N} \sum_{l=1}^{N \cdot dim} y_l \tag{4.7}$$

$$\sigma^2 = \sum_{l=1}^{N \cdot dim} (y_l - \mu)^2 \tag{4.8}$$

Kód dát teraz bude pozostávať z Shannonovho kódu pre jednotlivé súradnice, pričom dĺžka kódu jedného čísla x je

$$-\log_2 P(x)$$

kde $P(x)$ je pravdepodobnosť výskytu x v normálnom rozdelení danom parametrami μ, σ^2. Dĺžku cez všetky dáta označujeme

$$-\log_2 P(D|H^g) \tag{4.9}$$

Kód hypotézy H^g obsahuje kódovanie dvoch parametrov rozdelenia: stred a variancia. Oba parametre sú neohraničené a preto musíme podľa definície 2 kódovať maximum a minimum pre každý parameter ako celé čísla. Keďže chceme pomocou intervalu $[a, b]$ zakódovať vždy len jedno číslo, stačí kódovať len minimum, dekóder už vie, že maximum b je vždy $b = a + 1$. Nech

$$a^1 = \lfloor \mu \rfloor$$

$$a^2 = \lfloor \sigma^2 \rfloor$$

Dĺžka kódu hypotézy H_g je:

$$L^*(d) + L^*(|a^1|) + L^*(|a^2|) + 2 + 2d \tag{4.10}$$

dvojka pochádza z kódovania dvoch znamienok pre a^1, a^2. $2d$ sú dva, d bitov dlhé, kódy vymeriavajúce pozíciu v mriežkach. Počet parametrov $k = 2$ rozdelenia nemusíme kódovať (chýba člen $L^*(2)$, pretože používame rozdelenie s pevným počtom parametrov, a na tom sa enkóder a dekóder dohodnú dopredu.

Ďalej musíme kódovať hypotézu H: súradnice stredov, ktoré kódujeme najprv crude formou ako 4.3 a následne, použijeme na súradnice diskretizáciu mriežkou s rovnakou presnosťou d ako používame na diskretizáciu parametrov rozdelenia. Nájdeme minimum a a maximum b cez všetky súradnice všetkých stredov (pod stredmi sa teraz už rozumie ich rozdiel od ťažiska $\boldsymbol{T}_s$). Položíme mriežku s dĺžkou dielika 2^{-d}. Zakódujeme minimum a maximum pomocou kódu pre celé čísla a samotné súradnice, zaokrúhlené na pozície stredov dielikov mriežky, dostanú uniformný kód. Ešte musíme zakódovať dimenziu dim, aby dekóder vedel, kde sú oddelené vektory, počet stredov k (presnosť d sa kóduje v pravdepodobnostnej časti, netreba ju kódovať dva krát). Dĺžka kódu hypotézy H je:

$$L^*(k) + L^*(dim) + L^*(a) + L^*(b) + k \cdot dim \cdot d \tag{4.11}$$

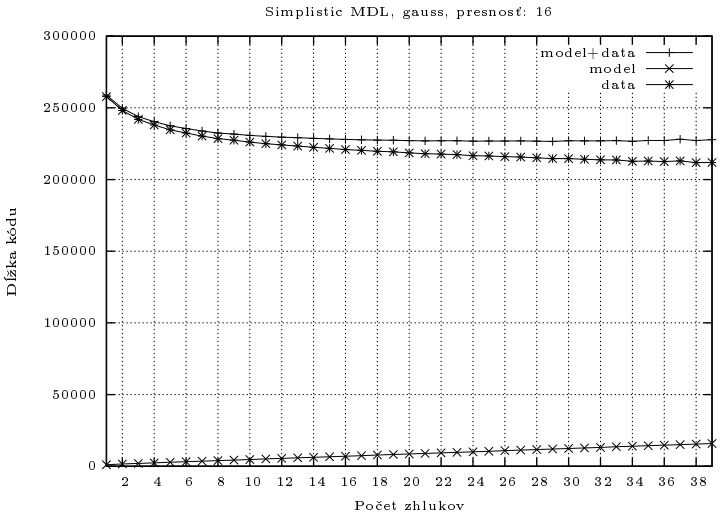
Celková dĺžka kódu je súčet 4.11 + 4.10 + 4.9:

$$\begin{gathered} L^*(k) + L^*(dim) + L^*(a) + L^*(b) + k \cdot dim \cdot d \\ + L^*(d) + L^*(|a^1|) + L^*(|a^2|) + 2 + 2d \\ - \log_2 P(D|H^g) \end{gathered} \tag{4.12}$$

Pri tejto metóde by sme podľa definície 2 mali minimalizovať dĺžku kódu vzhľadom na d a hranice a, b, a^1, a^2. Hranice a^1, a^2 sa nedajú ďalej zmenšovať. Hranice a, b sú tiež pevne dané rozsahom súradníc stredov. Ostáva nám možnosť hýbať parametrom d a tým akoby určovať sklon grafu dĺžky hypotézy, ako vidieť na obrázkoch 4.6 a 4.8. Podľa definície pre čisto pravdepodobnostné hypotézy nás Grünwald navádza vybrať za správnu tú hodnotu parametra d, kde dosiahneme aj najmenšiu

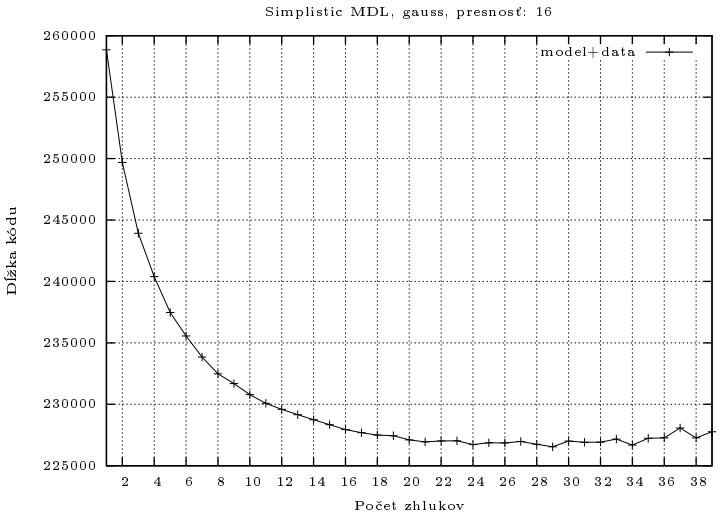
dĺžku súhrnného kódu. To by však znamenalo vybrať $d = 2$ (pričom pri tejto presnosti prestáva Gaussove rozdelenie dobre vystihovať dáta - stred a variancia sú príliš zaokrúhlené mriežkou) a pri nízkych hodnotách d dostávame výsledok 39 zhlukov (resp. potenciálne viac). Je možné, že v dátach skutočne existuje omnoho viac zhlukov, avšak riešime trochu aj aplikačný problém, musíme nájsť dostatočne dobrý dostatočne malý počet zhlukov. Tento problém zatiaľ nechávame otvorený, na konci kapitoly navrhujeme riešenie na aplikačný problém.

Obrázok 4.6: Výsledok MDL metódy: gauss, na 3-slovných vetách, parameter $d = 16$

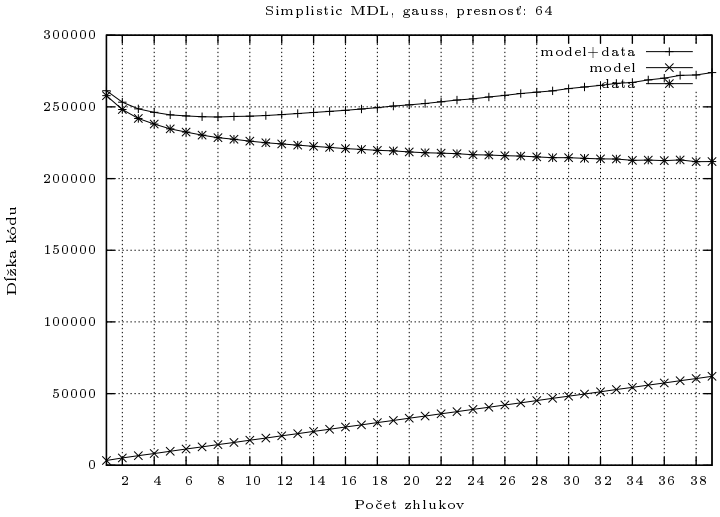


Obrázok 4.7: Výsledok MDL metódy: gauss, na 3-slovných vetách, parameter $d = 16$, súhrnná dĺžka kódu

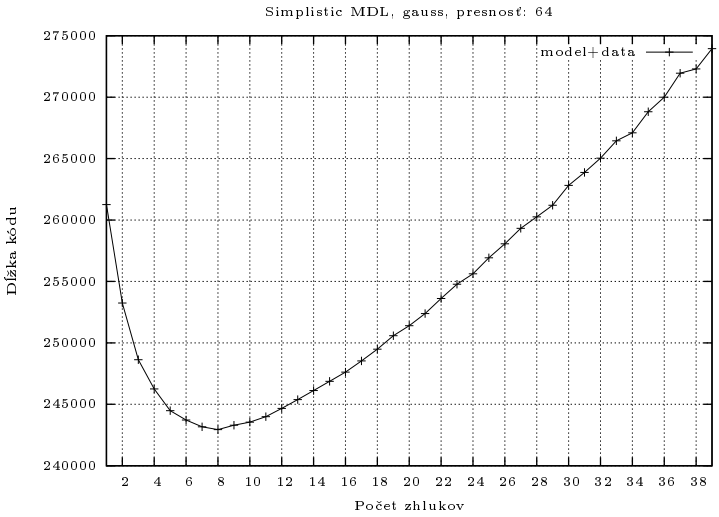
-



Obrázok 4.8: Výsledok MDL metódy: gauss, na 3-slovných vetách, parameter $d = 64$



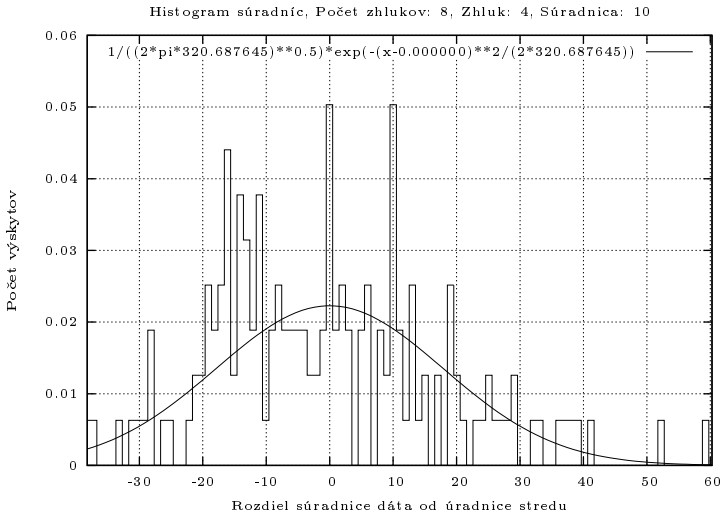
Obrázok 4.9: Výsledok MDL metódy: gauss, na 3-slovných vetách, parameter $d = 64$, súhrnná dĺžka kódu



4.2.2 Metóda Gauss pre každý stred

Naďalej skúšame nájsť zákonitosti v dátach. Skúsili sme si nakresliť aj histogram súradníc dát pre jednotlivé zhluky (stredy). Očakávame, že pre jednotlivé stredy môže byť rozptyl dát od stredu rôzny a preto aj parametre normálneho rozdelenia budú lepšie zodpovedať skutočnému rozloženiu dát. V kompresii by sme potom dostali o čosi kratšie kódy, lebo pravdepodobnostné rozdelenia lepšie vystihnú dáta. A keď už skúmame pre každý stred, mali by sme skúmať aj pre každú súradnicu. Nakreslili sme histogramy súradníc pre každý stred a každú súradnicu, nie je priestor uvádzať viac obrázkov, aspoň jeden ilustratívny je na obrázku 4.10

Obrázok 4.10: Histogram hodnôt súradníc y: na 2-slovných vetách, 8 zhlukov, 4. zhluk, súradnica 10



Keďže histogramy nevyzerali zle na predpoklad dim-dimenzionálneho normálneho rozdelenia pre každý stred, vytvorili sme ďalšie tri metódy: gauss_stred - rozdelenie cez všetky súradnice pre každý stred, gauss_ndim - rozdelenie pre každú súradnicu (dim-dimenzionálna superpozícia) cez všetky stredy, a napokon gauss_stred_ndim - rozdelenie pre každú súradnicu a každý stred. Očakávame dĺžku popisu dát v poradí od najväčšej po najmenšiu: gauss, gauss_stred, gauss_ndim, gauss_stred_ndim.

Opíšeme v krátkosti prvú metódu gauss_stred. Crude časť kódujeme rovnako ako v predošlom, s dĺžkou 4.11. Nebudeme meniť ani kód pre jednu inštanciu normálneho rozloženia, iba zakódujeme k inštancií. Potrebujeme k priemerov a k variancií:

$$\mu_j = \sum_{i=1}^{N} \sum_{l=1}^{dim} r_{ij} y_{il} \tag{4.13}$$

$$\sigma_j^2 = \sum_{i=1}^{N} \sum_{l=1}^{dim} r_{ij} (y_i l - \mu_j)^2 \tag{4.14}$$

Dostaneme tak kód pre dáta (podľa k normálnych rozdelení):

$$\sum_{i=1}^{k} -\log_2 P(D|H_i^g) \tag{4.15}$$

A kód pre k hypotéz H^g:

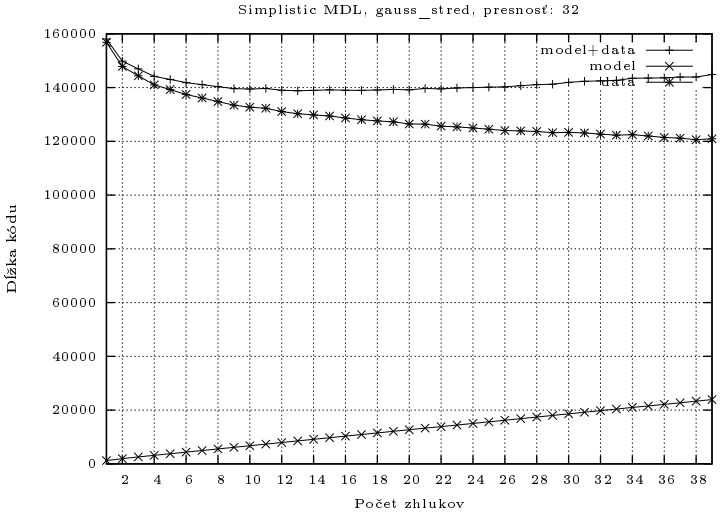
$$L^*(d) + \sum_{i=1}^{k} (L^*(|a_i^1|) + L^*(|a_i^2|)) + 2k + 2kd \tag{4.16}$$

Celkový kód je teda:

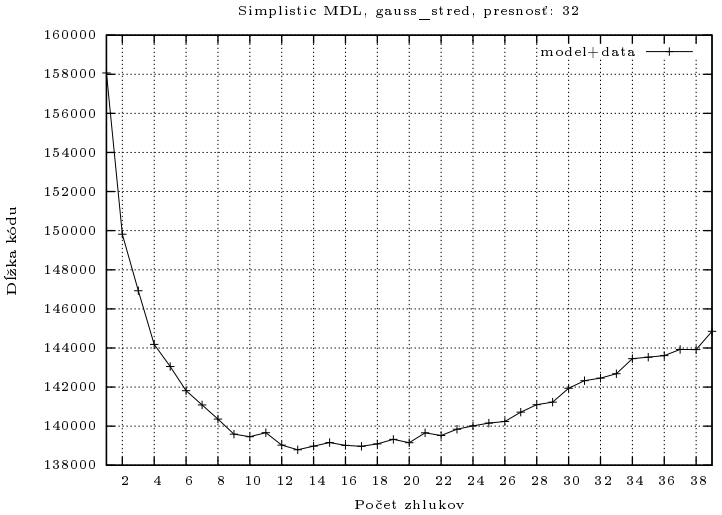
$$\begin{array}{c} L^*(k) + L^*(dim) + L^*(a) + L^*(b) + k \cdot dim \cdot d \\ + L^*(d) + \sum_{i=1}^{k} (L^*(|a_i^1|) + L^*(|a_i^2|)) + 2k + 2kd \\ \sum_{i=1}^{k} (-\log_2 P(D|H_i^g)) \end{array} \tag{4.17}$$

Podľa očakávaní nám dĺžka kódu dát v experimentoch vyšla všeobecne menšia ako v metóde Gauss. Stále však ostáva parameter d a aplikačný problém známy už z časti 4.2.1 Jeden z výstupov metódy je na obrázku 4.11

Obrázok 4.11: Výsledok MDL metódy: gauss_stred, na 2-slovných vetách, parameter $d = 32$



Obrázok 4.12: Výsledok MDL metódy: gauss_stred, na 2-slovných vetách, parameter $d = 32$, súhrnná dĺžka kódu



4.2.3 Metóda Gauss cez každú súradnicu

Rozdiel oproti metóde Gauss je len v tom, že jednorozmerné normálne rozdelenie nahradíme superpozíciou *dim* rozdelení. Podľa [Bis06, p.93] na nájdemie *dim*-rozmernej hustoty potrebujeme

nájsť kovariančnú maticu vektorov dát. Spôsob jej hľadania, vlastnosti a ďalšie podrobnosti sú nad rámec tejto práce. Preto sa venujeme len implementácii tejto (aj nasledujúcej) metódy s pomocou predprogramovanej funkcie cov v knižniciach. Samotnú kovariančnú maticu zahŕňame do kódu hypotézy H^g ďalším ad-hoc kódovaním. Podobne ako stredy zhlukov, nájdeme aj v kovariančnej matici minimum a^m a maximum b^m, položíme mriežku, opäť s použitím rovnakej presnosti d ako na ostatné diskretizácie. Dĺžka kódu hypotézy H^g sa zmení v dvojici parametrov, tie zameníme za dĺžku kódu diskretizovanej kovariančnej matice a dĺžku kódu pre priemer $\boldsymbol{\mu}$. Vektor $\boldsymbol{\mu}$ získame:

$$\boldsymbol{\mu}_j = \sum_{i=1}^{N} r_{ij} \boldsymbol{y}_i \tag{4.18}$$

Dĺžka kódu pre dáta:

$$-\log_2 P(D|H^g) \tag{4.19}$$

Kde H^g je superpozícia rozdelení určená diskretizovanou kovariančnou maticou.

Kód pre kovariančnú maticu má dĺžku:

$$L^*(d) + L^*(|a^{kov}|) + L^*(|b^{kov}|) + 2 + d \cdot dim^2 \tag{4.20}$$

Kde 2 pochádza z kódovania znamienok hraníc a^{kov}, b^{kov}. Rozmery kovariančnej matice sú dim^2 a každé číslo v nej má d bitov, takže celkový kód pre pozície v mriežke má dĺžku $d \cdot dim^2$

Kód pre vektor $\boldsymbol{\mu}$ má dĺžku:

$$L^*(|a^{\mu}|) + L^*(|b^{\mu}|) + 2 + d \cdot dim \tag{4.21}$$

Celkový kód je teda:

$$\begin{gathered} L^*(k) + L^*(dim) + L^*(a) + L^*(b) + k \cdot dim \cdot d \\ + L^*(d) + L^*(|a^{kov}|) + L^*(|b^{kov}|) + 2 + d \cdot dim^2 \\ + L^*(|a^{\mu}|) + L^*(|b^{\mu}|) + 2 + d \cdot dim \\ + \textstyle\sum_{i=1}^{k} (-\log_2 P(D|H_i^g)) \end{gathered} \tag{4.22}$$

Pri tejto metóde sa začali objavovať numerické problémy pri zaokrúhľovaní na diskrétnu mriežku. Niekedy sa stalo, že zaokrúhlením sme získali delenia nulov pri počítaní determinantu matice. Zdá sa však, že tento model je už príliš komplexný, kód pre hypotézu bol pomerne dlhý.

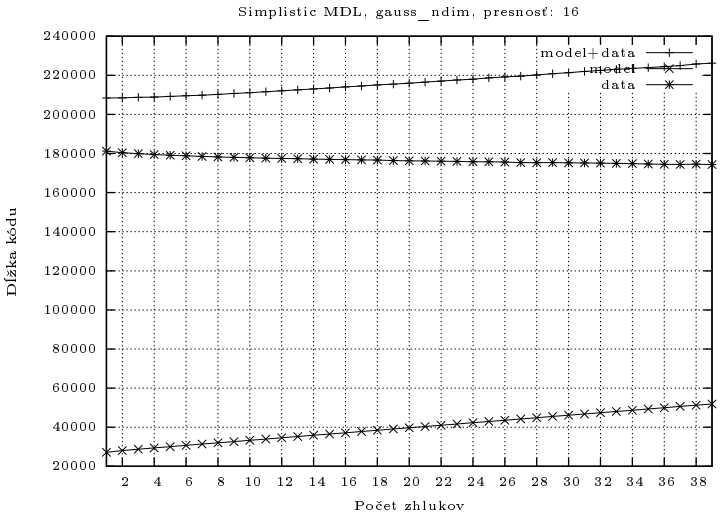
Výsledok metódy je na obrázku 4.13

4.2.4 Metóda Gauss cez každú súradnicu pre každý stred

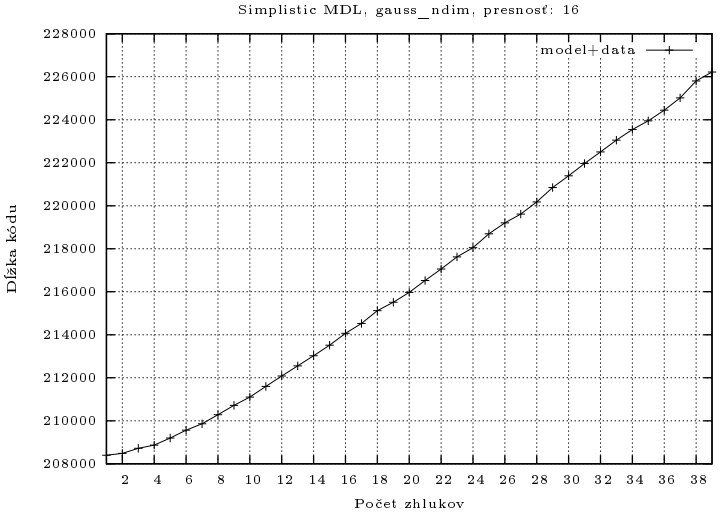
Rozdiel oproti metóde Gauss-stred je v tom, že jednorozmerné normálne rozdelenia nahradíme superpozíciou dim rozdelení. Potrebujeme k stredov a k kovariančných matíc, avšak stredy sú nulové vektory - tak sme dáta zhlukovali, nepotrebujeme ich ani kódovať. Kód pre každú z k kovariančných matíc použijeme rovnaký ako 4.20. Dĺžka kódu pre dáta:

$$\sum_{j=1}^{k} -\log_2 P(D|H_j^g) \tag{4.23}$$

Obrázok 4.13: Výsledok MDL metódy: gauss_ndim, na 5-slovných vetách, parameter $d = 16$



Obrázok 4.14: Výsledok MDL metódy: gauss_ndim, na 5-slovných vetách, parameter $d = 16$, súhrnná dĺžka kódu



Kód pre k hypotéz H^g zahŕňa kovariančné matice:

$$L^*(d) + \sum_{j=1}^{k}(L^*(|a_j^{kov}|) + L^*(|b_j^{kov}|)) + 2k + kd \cdot dim^2 \tag{4.24}$$

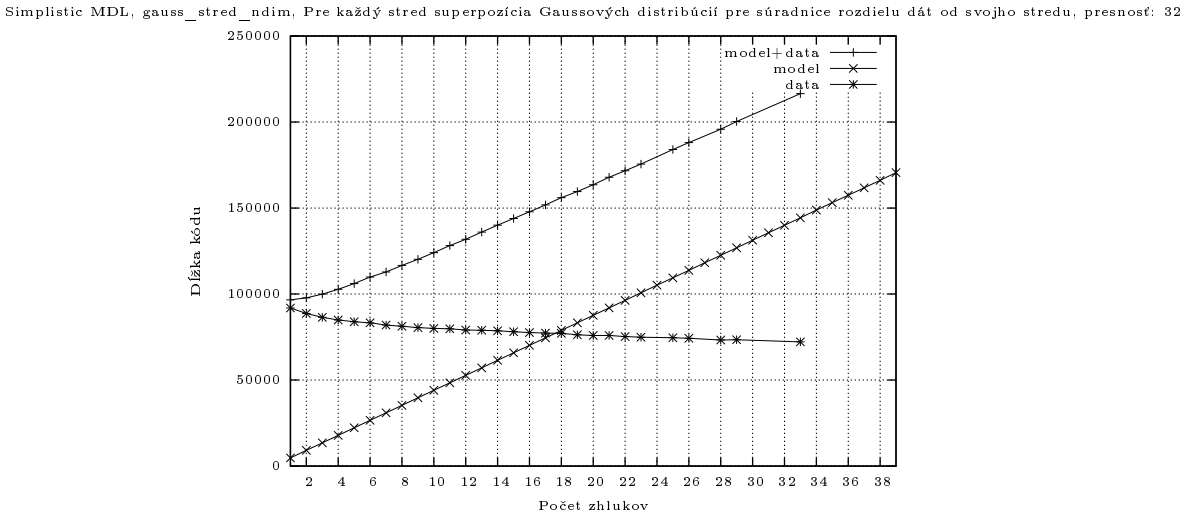
Celkový kód je teda:

$$\begin{gathered} L^*(k) + L^*(dim) + L^*(a) + L^*(b) + k \cdot dim \cdot d \\ +L^*(d) + \textstyle\sum_{j=1}^{k}(L^*(|a_j^{kov}|) + L^*(|b_j^{kov}|)) + 2k + kd \cdot dim^2 \\ + \textstyle\sum_{i=1}^{k}(-\log_2 P(D|H_i^g)) \end{gathered} \tag{4.25}$$

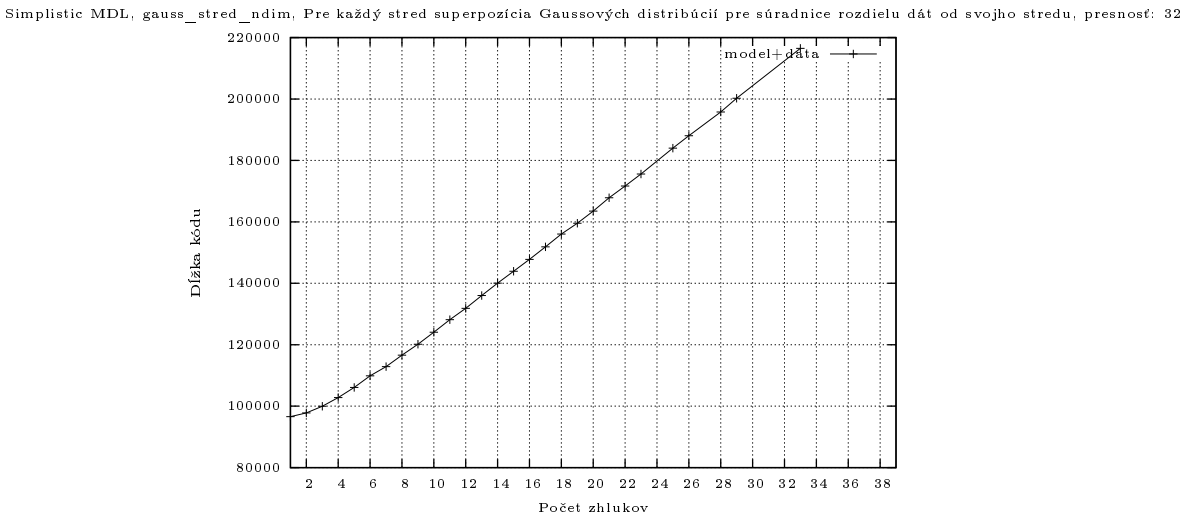
Dĺžky kódov pred dáta boli vo všeobecnosti najmenšie zo všetkých metód. Avšak ukázalo sa, že kódovanie kovariančných matíc je veľkým bremenom a súhrnné dĺžky kódov nadobúdali minimum spravidla pre 1 zhluk - následne už komplexnosť modelu „prebila“ regularitu dát.

Navyše pre numerické problémy so zaokrúhlením na mriežku dopadlo veľa hodnôt v delení nulou. Metóda sa tak stala vlastne nepoužiteľnou. Výsledok metódy je na obrázku 4.15

Obrázok 4.15: Výsledok MDL metódy: gauss_stred_ndim, na 1-slovných vetách, parameter $d = 32$



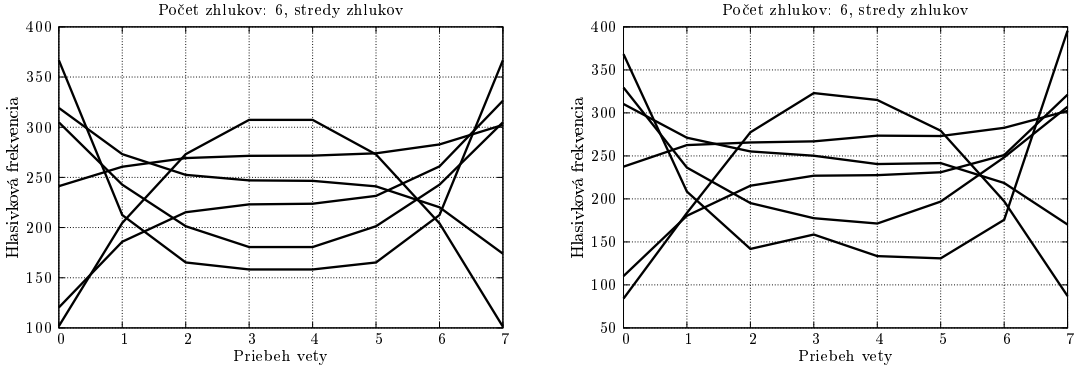
Obrázok 4.16: Výsledok MDL metódy: gauss_stred_ndim, na 1-slovných vetách, parameter $d = 32$, súhrnná dĺžka kódu



4.2.5 Aplikačný problém

Aplikačný problém nášho MDL skúmania dát spočíva v tom, že je celkom dobre možné, že v dátach prirodzený „správny“ počet stredov neexistuje, alebo že je väčší ako naše praktické obmedzenie. Hľadáme tak skôr druhý najlepší, alebo akceptovateľne dobrý počet stredov a nie ideálny. Aby sme tento problém riešili, vyskúšali sme trénovať rôzne metódy simplistic MDL (filozofi nemajú žiadne menné parametre, nie je na nich čo ladiť) na pokusných, testovacích dátach. Naprogramovali sme generátor pokusných dát. Generujú sa s nejakým počtom stredov, majú tvar polynómov a jednotlivé zašumené (Gaussovským šumom) dáta rôznych stredov sa prekrývajú. Na tieto dáta sme spúšťali rovnaký proces analýzy, najprv K-means, ktorý našiel stredy veľmi podobné syntetickým stredom, čo môžeme vidieť na obrázku 4.17

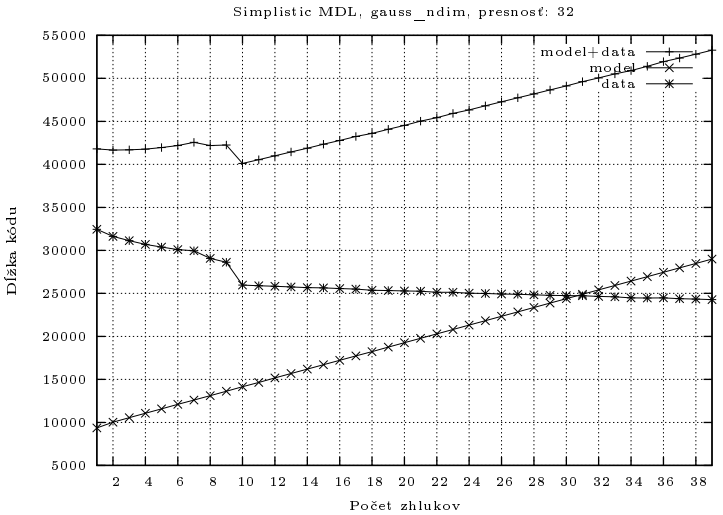
Obrázok 4.17: Syntentické stredy a stredy nájdené algoritmom K-means



Následne sme sa snažili naladiť parameter d pre každú metódu na syntetické dáta tak, aby dávala správny výsledok. Avšak, pre rôzne testovacie dáta sme dostali rôzne intervaly možných hodnôt d, dokonca aj bez prieniku. Vyskúšali sme ešte skúmať úplne syntetické dáta: dáta, pri ktorých sa zhluky vôbec neprekrývajú. Algoritmus K-means našiel správne stredy a skúšali sme MDL. Ako vidieť na obrázku 4.18, dĺžka kódu dát spoľahlivo klesá až po dosiahnutie správneho počtu stredov. Každý ďalší stred pridaný algoritmom K-means je iba overfitting a nezlepší dĺžku kódu dát. Dĺžka kódu hypotézy naproti tomu lineárne stúpa s počtom stredov. Parameter d má na správanie výsledku takýto vplyv: čím väčšia presnosť d, tým ostrejšie rastie dĺžka kódu hypotézy (presnejšia diskretizačná mriežka) a tým lepšie klesá dĺžka kódu dát. Ibaže dĺžka kódu dát sa už pri malej hodnote $d = 4$ zastabilizuje a pre väčšie d už parametre rozdelenia nie sú kódovaná oveľa presnejšie. V podstate d ovplyvňuje v našom prípade (našich inštanciách kódov) najmä sklon dĺžky hypotézy.

Pri prezeraní obrázkov pre stúpajúce d sme dostali nasledujúci nápad, ako riešiť náš aplikačný problém. Ak d stúpa, rastie síce hodnota hypotézy, ale správny počet zhlukov ako minimum súčtu dĺžok kódov odoláva pomerne širokému intervalu parametra d (od 2 do 64). Pri priveľkom d už dĺžka kódu hypotézy „prebije“ aj prudko klesajúcu dĺžku kódu dát. Mohli by byť „správne“ počty zhlukov tie počty, ktoré dobre odolávajú rastúcemu d? Je najsprávnejší počet zhlukov ten, ktorý pre najviac rôznych hodnôt d je minimom funkcie? Zdá sa nám, že táto úvaha je celkom logická a neskoršie skúmania, ktoré spomíname vo výsledkoch to len potvrdzujú. Úvaha si však pýta

Obrázok 4.18: Výsledok MDL metódy: gauss_ndim, na 2-slovných syntetických dátach, parameter $d = 32$



dôslednejšie formalizovanie.

Kapitola 5

Záver

Naprogramovali sme prostredie pre zbiehanie zhlukovania a MDL v jazyku python. Navrhli sme, a následne aj vyskúšali rôzne formy Crude MDL aplikovať na dáta - melodické obálky viet. Na vzniknutý aplikačný problém sme navrhli praktické, aplikované riešenie. Pôvodne položenú otázku, koľko zhlukov sa nachádza v dátach sme čiastočne zodpovedali - ťažko možno súdiť kvalitu aplikovaného riešenia. Pre potreby syntézy reči je táto odpoveď veľmi dobrá, o samotných dátach totiž ťažko niečo predpokladať a cieľ nebol nájsť ideálnu odpoveď, ale odpoveď, ktorá sa dá pri syntéze použiť. Praktický výsledok je teda navrhnutie správneho rozdelenia do zhlukov pre kolegov zo Žiliny, ktorí tento výsledok použijú pri syntéze reči. Odovzdávame im celý projekt, so zdrojovými kódmi, touto prácou, grafmi a dátovými súbormi. Budeme s nimi ďalej spolupracovať na riešení ďalších otázok, prípadne na vylepšení riešenia.

Metódy gauss_ndim (časť 4.2.3) a gauss_stred_ndim (časť 4.2.4) sa ukázali ako overfitting, samotný kód pre hypotézy bol príliš dlhý, hoci sme objavili viac regularít v dátach. Tieto dve metódy nám nakoniec rozumný výsledok nepovedali. Naše doterajšie pokusy s upravenou metódou filozof (časť 4.1.1) a metódou gauss (časť 4.2.1) ukazujú, že metódy sa na správnom výsledku zhodnú. Je možné, že pre účely získania optimálneho počtu zhlukov nie je vôbec potrebné skúmať pravdepodobnostné rozdelenie dát kódovaných pomocou stredov. Odpovede v rozsahu ± 1 od metódy filozof a gauss nám dala aj metóda gauss_stred (časť 4.2.2).

V ďalšej práci je možné sa venovať formalizácii aplikačnej úvahy z časti 4.2.5. Po prevedení všetkých metód, si myslíme, že lepšie než normálne rozdelenie pravdepodobnosti dáta vystihuje Student-t, pretože dáta majú ťažké chvosty. V práci by sa dalo pokračovať aj vyskúšaním EM algoritmu na zhlukovanie (inicializovaného pomocou algoritmu K-means) a následne Student-t rozdelenie v MDL časti aplikácie.

Literatúra

[Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, August 2006.

[ČŠ] Mária Čerňanská and Ondrej Škvarek. Sentence melody analysis for speech production in the TTS system.

[CT06] T.M. Cover and J.A. Thomas. *Elements of information theory*. Wiley-Interscience, 2006.

[Grü04] P. Grünwald. A tutorial introduction to the minimum description length principle. *Arxiv preprint math/0406077*, 2004.

[Grü07] P.D. Grünwald. *The minimum description length principle*. Mit Press, 2007.

[PPTL98] A.D. Patel, I. Peretz, M. Tramo, and R. Labreque. Processing prosodic and musical patterns: a neuropsychological investigation. *Brain and Language*, 61(1):123–144, 1998.

[Ptá92] Miroslav Ptáček. *Úvod do fonetické akustiky*. Univerzita Karlova v Praze, Fakulta filozofická, 1992.