



FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY
UNIVERZITA KOMENSKÉHO, BRATISLAVA

HIERARCHICKÁ KATEGORIZÁCIA DÁT

FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY
UNIVERZITA KOMENSKÉHO, BRATISLAVA

HIERARCHICKÁ KATEGORIZÁCIA
DÁT
(Diplomová práca)

Študijný program : aplikovaná informatika

Študijný odbor: 2511 aplikovaná informatika

Školiace pracovisko: Katedra aplikovanej informatiky

Školiteľ: RNDr. Martin Takáč, PhD.

Bratislava 2013

Autor: Bc. Jozef Čechovský



Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Jozef Čechovský
Študijný program: aplikovaná informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: 9.2.9. aplikovaná informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: slovenský

Názov: Hierarchická kategorizácia dát


Cieľ: Cieľom práce je preskúmať možnosti hierarchickej kategorizácie dát. Navrhnuť a implementovať algoritmus na hierarchickú kategorizáciu dát inšpirovaný algoritmom IHDR (Weng & Hwang, 2007), preskúmať vlastnosti implementovaného algoritmu, citlivosť na parametre, sledovať úspešnosť kategorizácie/klasifikácie na štandardných dátových sadách a porovnať s výsledkami ďalších existujúcich algoritmov. Analyzovať schopnosť algoritmu vytvárať rozumné hierarchie.

Literatúra: Weng J, Hwang W.S.: Incremental hierarchical discriminant regression. IEEE Transactions on Neural Networks 2007 Mar18(2):397-415.

Poznámka: pasívna znalosť angličtiny, schopnosť a ochota používať matematický aparát :), vlastná iniciatíva a ochota pracovať priebežne

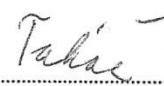
Vedúci: RNDr. Martin Takáč, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: doc. PhDr. Ján Rybár, PhD.
Dátum zadania: 07.10.2011

Dátum schválenia: 10.10.2011


doc. RNDr. Roman Ďurikovič, PhD.
garant študijného programu


.....

študent


.....

vedúci práce

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

.....

Ďakujem môjmu vedúcemu RNDr. Martinovi Takáčovi, PhD. za jeho vedenie, odborné rady, trpezlivosť a ústretovosť počas tvorby mojej diplomovej práce.

Abstrakt

V našej práci sa zaoberáme hierarchickým kategorizovaním dát. Konkrétne sa zameriavame na hierarchický Bayesovský model HBM a inkrementálnu hierarchickú diskriminačnú regresiu IHDR. Tieto dva biologicky motivované výpočtové modely nám popisujú, ako rozumne vytvárať hierarchie určené na kategorizáciu alebo klasifikáciu dát. V oboch modeloch hierarchické kategorizovanie vykonávame na základe viac abstraktných znalostí v jednotlivých vrstvách a uzloch hierarchie, ktoré nazývame metahypotézy alebo najviac diskriminačné črty. V práci popisujeme teoretické znalosti o týchto modeloch a návrh a implementáciu algoritmu založeného na IHDR, ktorý dokáže v reálnom čase vytvárať rozumné hierarchie kategórií vysokorozmerných dát. Následne sme s použitím bežných dátových súbier otestovali funkčnosť algoritmu, vizualizovali vytvorené hierarchie, popísali vlastnosti algoritmu a úspešnosť pri kategorizovaní a klasifikácii dát.

Kľúčové slová:

hierarchická kategorizácia, hierarchický Bayesovský model, HBM, metahypotéza, inkrementálna hierarchická diskriminačná regresia, IHDR, klasifikácia dát, kategorizácia dát

Abstract

The thesis deals with hierarchical categorisation of data trying to reach the main target of hierarchical Bayesian model HBM and incremental hierarchical discriminant regression IHDR. These biologically motivated computational models describe possibility of creation a reasonable hierarchy with purpose of data categorisation and classification. Hierarchical categorization is executed upon more abstract knowledge in several layers and nodes of hierarchy. These abstract facts are referred to as over-hypothesis and most discriminant features. The thesis describes theory of these models, design and implementation of algorithm based upon IHDR, which can build reasonable hierarchy of multidimensional data in real time. Consequently, functionality of algorithm on common datasets was tested, the created hierarchy visualized and behaviour of algorithm and percentages of successful data classification described.

Key words:

hierarchical categorization, hierarchical Bayesian model, HBM, overhypothesis, incremental hierarchical discriminant regression, IHDR, data classification, data categorization

Obsah

1	ÚVOD	9
2	HIERARCHICKÝ BAYESOVSKÝ MODEL	10
2.1	Induktívna generalizácia z príkladov.	10
2.2	Učenie označení kategórií	13
2.3	Induktívne obmedzenia	16
2.4	Hierarchický Bayesovský model	18
2.4.1	Metahypotézy v HBM	18
2.4.2	Získavanie metahypotéz	20
2.5	Zhrnutie	22
3	INKREMENTÁLNA HIERARCHICKÁ DISKRIMINAČNÁ REGRESIA	23
3.1	Biologická motivácia	23
3.2	Dátové štruktúry	24
3.3	Klasifikácia a regresia	25
3.4	Východiská IHDR	26
3.4.1	Diskriminačná analýza pre numerický výstup	26
3.4.2	Dvojitá klasterizácia	27
3.4.3	Štatistické parametre klastrov a vrcholov	28
3.4.4	Amnézický priemer v IHDR	29
3.4.5	Podpriestor najviac diskriminačných črt	31
3.4.6	Metriky založené na pravdepodobnosti	34
3.4.7	Prechod medzi metrikami	35

3.5	IHDR algoritmus	37
3.5.1	Procedúra Update-Tree	38
3.5.2	Procedúra Update-Node	38
3.5.3	Procedúra Update-ClusterPair	39
3.6	Porovnanie IHDR a HBM	40
4	IMPLEMENTÁCIA	42
4.1	Návrh	42
4.1.1	Projektová štruktúra	42
4.1.2	Objektová štruktúra	43
4.2	Použité algoritmy	44
4.2.1	GSO algoritmus	44
4.2.2	K-means algoritmus	45
4.2.3	K-means++ algoritmus	45
4.3	Časti algoritmu a modifikácie	46
4.3.1	Rozvetvenie vrchola	46
4.3.2	Parametre vrcholov podľa hĺbky	47
4.3.3	Vytváranie klastrov	48
4.3.4	Klasifikácia a regresia	50
4.3.5	Klasifikovanie dát	50
4.4	Metodika vývoja	51
4.5	Použité nástroje	52
4.5.1	Math.Net Numerics	52
4.5.2	ILNumerics	53
4.5.3	Git a GitHub	54
5	EXPERIMENTÁLNE VÝSLEDKY	56
5.1	Parametre algoritmu	56
5.2	Syntetické dáta	57

5.3	Hierarchie výrazov tváří	58
5.4	Klasifikácia dát	63
5.4.1	Dátová sada MNIST	63
5.4.2	Klasifikácia príkladov	67
5.5	Správanie algoritmu a jeho častí	70
5.5.1	Počet klastrov a stupeň rozvetvenia	70
5.5.2	Porovnanie metrík	71
5.6	Porovnanie výsledkov	72
6	ZÁVER	74
7	PRÍLOHY	75
	Príloha A – CD	75
	Príloha B – Zobrazenie hierarchie	76
8	REFERENCIE	77

1 Úvod

Jedna z najhlavnejších otázok kognitívneho vývinu je, ako sa naučiť čo najviac z nejakej limitovanej množiny dát. V učení detí o reláciách medzi objektmi, o kategóriách objektov, vlastnostiach jednotlivých kategórií, pri získavaní slov jazyka alebo konštruovaní intuitívnych teórií, detské uvažovanie siaha až za hranicu dát, ktoré mali doposiaľ k dispozícii [1]. Naučiť sa kategorizovať dáta nesúvisí teda len s naučením sa nejakých špecifických faktov charakteristických pre individuálne kategórie, ale môžeme sa naučiť aj viac abstraktné znalosti o tom, do akej kategórie objekt patrí, alebo o tom, ako môžu byť kategórie organizované do nejakých štruktúr. Tieto abstraktné znalosti nám potom umožňujú naučiť sa kategorizovať príklady nových kategórií rýchlejšie, ako keď sa učíme kategorizovať nové objekty bez nich. Abstraktné znalosti môžeme získať na základe induktívnych krokov, ktoré sa dajú popísať pravdepodobnostnými modelmi formou Bayesovskej inferencie. Jedným z týchto modelov je hierarchický Bayesovský model HBM [2], pomocou ktorého môžeme vytvárať hierarchické štruktúry založené na induktívnych obmedzeniach alebo abstraktných znalostiach vo viacerých úrovniach. Tieto znalosti nazývame metahypotézy a sú to vlastne distribúcie pravdepodobností vybraných atribútov dát, s ktorými následne vyjadrujeme podmienenú pravdepodobnosť hovoriacu o príslušnosti do kategórie v nižšej vrstve. Na podobnom princípe je založený aj iný výpočtový model inkrementálna hierarchická diskriminačná regresia IHDR [3], ktorý je určený na vytváranie regresných stromov s možnosťou hierarchickej kategorizácie. V tomto modeli používame ako induktívne obmedzenia vo vyšších vrstvách najviac diskriminačné črty a kategorizovanie dát sa vykonáva na základe pravdepodobnosti určenej pomocou štatistických vlastností jednotlivých vrcholov vo vytvorenej hierarchii.

V nasledujúcich kapitolách najskôr rozoberieme teoretické základy metód HMB a IHDR, na základe ktorých tieto dve metódy porovnáme. Ďalej popíšeme návrh a podrobnosti implementácie algoritmu založenom na IHDR, vyskytnuté problémy, rôzne modifikácie prípadne možné vylepšenia a správanie jednotlivých častí algoritmu. V experimentálnej časti overíme funkčnosť nami naimplementovaného algoritmu na bežných dátových sádach, otestujeme jeho rôzne konfigurácie a výsledky porovnáme s výsledkami už existujúcich metód.

2 Hierarchický Bayesovský model

2.1 Induktívna generalizácia z príkladov.

Za jednu z najzákladnejších otázok Bayesovskej inferencie považujeme otázku, ako aktualizovať domnienky a vykonávať inferenciu na daných dátach. Hlavný predpoklad pritom je, že stupeň pravdivosti nejakej domnienky reprezentujeme ako pravdepodobnosť, čiže naše presvedčenie o pravdivosti hypotézy h môže byť vyjadrené ako reálne číslo v intervale 0 až 1, kde 0 znamená, že h je kompletne nepravdivá a 1 znamená, že je kompletne pravdivá. O tom, ako vypočítať stupeň pravdivosti hypotézy h_i , ak máme dané dáta d , nám hovorí teória pravdepodobnosti.

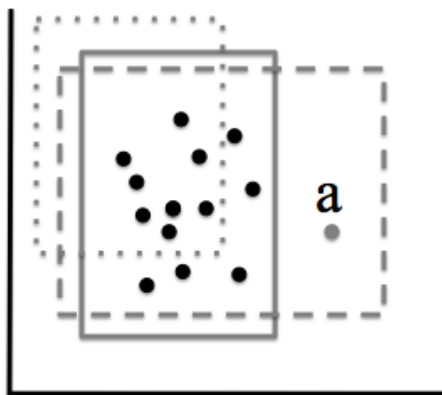
Výpočet stupňa pravdivosti ako pravdepodobnosť závisí na dvoch zložkách. Prvá sa nazýva apriórna pravdepodobnosť označená ako $P(h_i)$, ktorá zachytáva, ako veľmi veríme v pravdivosť h_i pred tým, ako máme k dispozícii nejaké dáta. Ďalšia zložka vierohodnosť alebo likelihood, ktorá sa označuje $P(d|h_i)$ zachytáva pravdepodobnosť, s ktorou môžeme očakávať práve také dáta d , ak je hypotéza h_i pravdivá. Keď tieto dve zložky skombinujeme, dostávame posteriórnu, podmienenú pravdepodobnosť hypotézy h_i , ktorej výpočet je definovaný podľa základného Bayesovského pravidla (2.1).

$$P(h_i|d) = \frac{P(d|h_i)P(h_i)}{\sum_{h_j \in H} P(d|h_j)P(h_j)} \quad (2.1)$$

Súčin apriórnej pravdepodobnosti a vierohodnosti v čitateli je intuitívny. Tento súčin je následne vydelený sumou cez pravdepodobnosti všetkých hypotéz z dôvodu normalizácie. Bayesovské pravidlo vyjadruje veľkosť pravdepodobnosti jednej hypotézy, pričom pravdepodobnosť jednotlivých hypotéz je rozdistribuovaná tak, aby ich súčet bol 1. To znamená, že keď je veľmi pravdepodobná jedna hypotéza, znižuje sa tak pravdepodobnosť iných.

Bayesovský prístup je generatívny. To znamená, že sú skúmané dáta generované nejakým základným procesom alebo mechanizmom, ktorý je zodpovedný za ich vytvorenie. Napríklad slová v jazyku môžu byť generované gramatikou v kombinácii so sociálnymi a pragmatickými faktormi, alebo generovanie náhodných čísel môže byť náhodné vyberanie prvkov z nejakej danej množiny čísel. V prípade kategorizácie sa pod hypotézou v Bayesovskej inferencii rozumie príslušnosť do kategórie. Bayesovským pravidlom teda určíme veľkosť pravdepodobnosti príslušnosti do tejto kategórie. Úlohou učiaceho sa subjektu (agenta) je vyhodnotiť rôzne hypotézy o generatívnom procese a robiť predikcie založené na najviac pravdepodobných hypotézach. Pravdepodobnostný model je jednoducho špecifikácia zúčastnených generatívnych procesov, teda identifikovanie krokov (a s nimi asociovaných pravdepodobností), ktoré generujú dáta [1]. Generatívny proces môže vo veľkej miere ovplyvniť inferenciu učiaceho sa agenta, pretože tá závisí od toho aký proces generuje dáta. Napríklad, ak okolo nás kašle veľa ľudí, myslíme si, že sa rozšíril nejaký vírus. Záleží však, aký generatívny proces generuje týchto ľudí. Pretože ak sme v čakárni u lekára, je normálne, že sa tam vyskytujú chorí ľudia a nemusí to znamenať rozšírenie vírusu.

Na nasledujúcom obrázku 2.1 sú ako bodky graficky znázornené dáta generované generatívnym procesom a ako obdĺžniky jednotlivé hypotézy, kategórie.

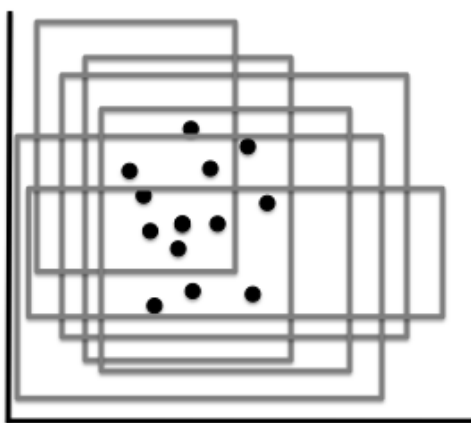


Obrázok 2.1 Grafické znázornenie dát a hypotéz [1].

Obrázok 2.1 zobrazuje viacero dát a tri hypotézy h_{solid} , h_{dashed} , h_{dotted} . Pred uvidením dát má učiaci sa agent istú domnienku, ktoré hypotézy sú najviac pravdepodobné, alebo či sú

všetky hypotézy rovnako pravdepodobné, prípadne pre ne má nejaké preferencie. Napríklad, či preferuje menšie alebo väčšie obdĺžniky. Veľkosť pravdepodobnosti domnienky sa vyjadruje apriórnu pravdepodobnosťou: $P(h_{solid})$, $P(h_{dashed})$, $P(h_{dotted})$. Na obrázku 2.1 sú dáta konzistentné s hypotézami h_{solid} , h_{dashed} , ale nie s h_{dotted} , pretože niektoré dáta sú mimo bodkovaného obdĺžnika. Tento jav sa odrazí vo vierohodnosti. $P(d|h_{solid})$ a $P(d|h_{dashed})$ by mali byť nenulové, ale $P(d|h_{dotted})$ môže byť aj nulové. Pridaný nový bod a má vierohodnosť $P(a|h_{solid}) = 0$, ale $P(a|h_{dashed}) > 0$, pretože leží v prerušovanom obdĺžniku.

Všetky možné hypotézy združujeme do priestoru hypotéz H . Priestor H je definovaný ako štruktúra problému, ktorým sa učiaci sa agent zaoberá. Na obrázku 2.2 je znázornený priestor hypotéz. Veľkosť priestoru hypotéz je nekonečná, na obrázku je pre ukážku znázornený len obmedzený počet.



Obrázok 2.2: Podmnožina priestoru hypotéz [1].

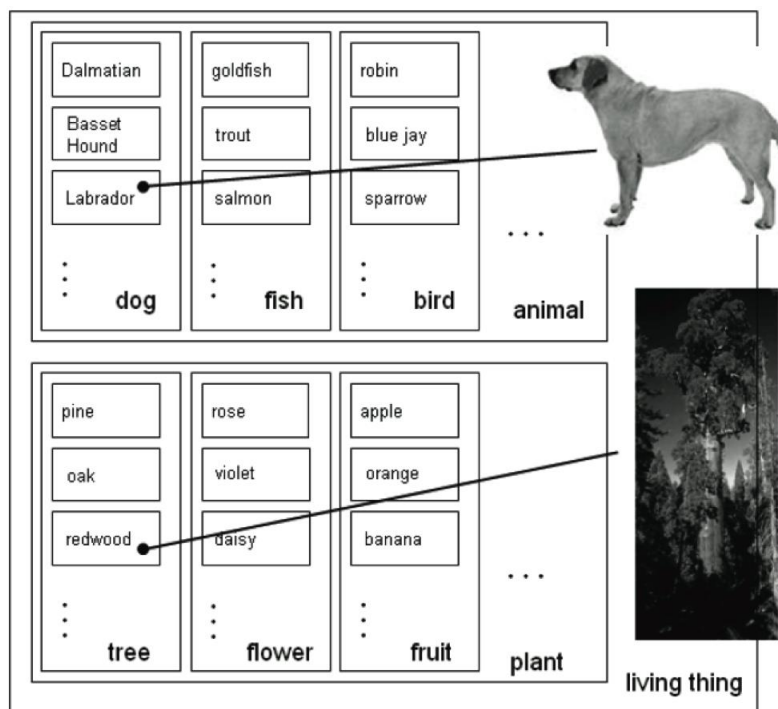
Priestor hypotéz môže byť ohraničený nejakými obmedzeniami. V príklade na obrázku 2.2 to môžu byť napríklad obmedzujúce intervaly pre súradnice (x, y) jedného rohu obdĺžnika, dĺžka l a šírka w obdĺžnika. Obmedzenia môžu byť taktiež určené aj distribúciou pravdepodobnosti. Apriórna pravdepodobnosť hypotézy $P(h_i)$ môže byť daná pravdepodobnostným generatívnym procesom, čo je proces operujúci o jednu úroveň vyššie od procesu generujúceho dáta. Hypotézy teda ovplyvňujú pravidlá, vlastnosti alebo obmedzenia generatívneho procesu. Keďže je generatívny proces o úroveň vyššie nad hypotézami, môžeme začať určovať príslušnosť do kategórie prechádzaním nejakej hierarchie, taxonómie,

kde máme v jednotlivých vrstvách alebo uzloch všeobecnejšiu informáciu o možnom priradení do kategórie na nižšej vrstve. Preto môžeme pri kategorizovaní dát zaviesť hierarchickú kategorizáciu, pri ktorej si pomáhame hierarchicky zoradenými znalosťami.

2.2 Učenie označení kategórií

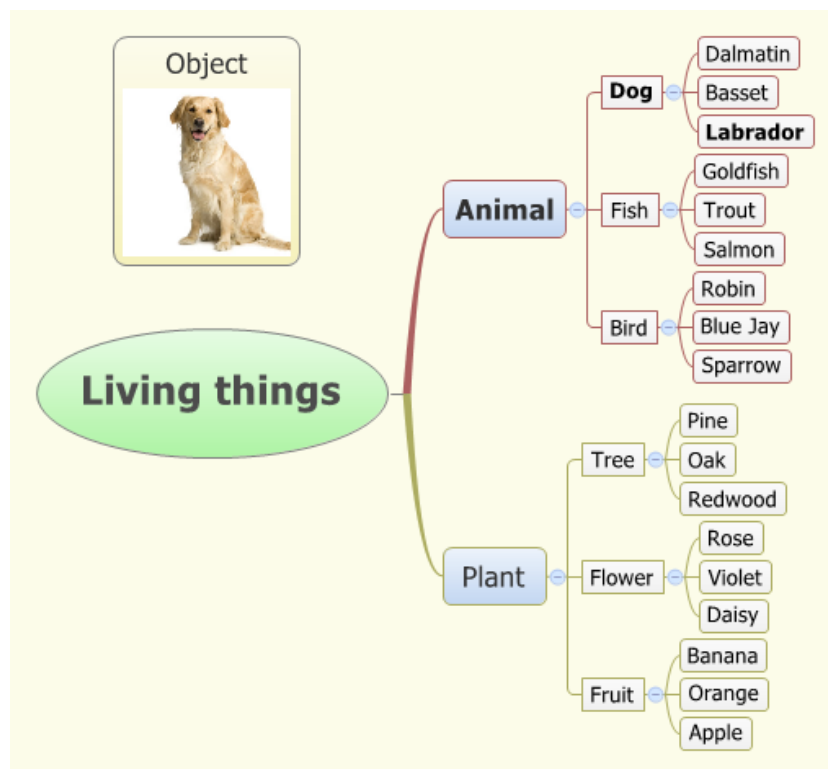
Pre ilustráciu Bayesovskej analýzy indukтивной generalizácie aplikovanej v praxi si zoberme naučenie sa pomenovať jednotlivé kategórie objektov. Táto úloha je často oblasť výskumu kognitívneho vývinu u detí. Jednoduché slovo ako „pes“ môže označovať hociktorú z nekonečne veľkej množiny kategórií, hypotéz. Napríklad hypotézy zahŕňajúce všetkých psov, labradorov, cicavcov, zvieratá, štvornohé živočíchy, objekty so srst'ou a iné. Napriek veľkému počtu možných extenzií sveta, malé deti dokážu zaradiť objekt do kategórie, aj keď videli veľmi málo príkladov z nej [4].

Xu a Tenenbaum predstavili Bayesovský model učenia sa slov, ktorý ponúka presný popis, ako si učiaci sa agenti môžu vytvoriť zmysluplné generalizácie z jedného alebo z pár príkladov nejakej novej kategórie [5]. Na obrázku 2.3. je zobrazená hierarchická taxonómia, pomenované koncepty alebo kategórie, ktoré patria do viacerých rôznych extenzií. Napríklad labrador patrí do kategórie „dog“, „animal“ a „living things“.



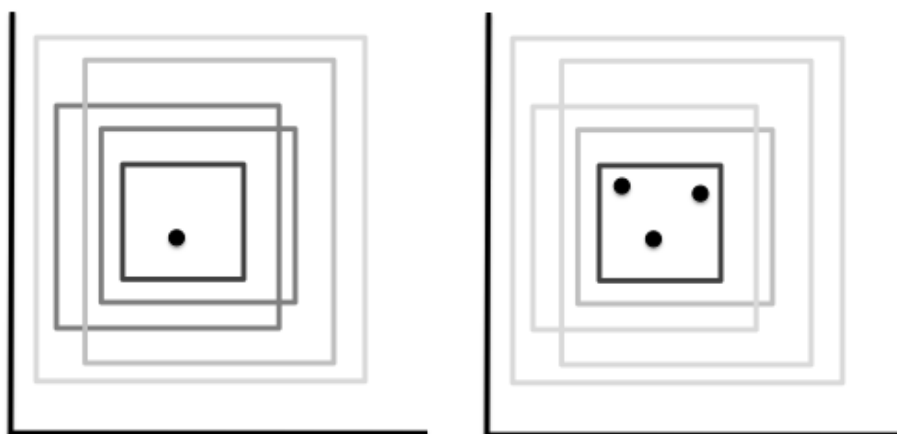
Obrázok 2.3 Hierarchická taxonómia hypotéz [1].

Pre lepšie predstavenie je na ďalšom obrázku 2.4 tá istá taxonómia zobrazená pomocou stromu, kde koreň stromu obsahuje najvšeobecnejšie objekty a vnútorné vrcholy viac konkrétne objekty. Najviac konkrétne sú samozrejme v listoch stromu. Zo stromového zobrazenia je lepšie vidno, na akej vrstve v hierarchii sú jednotlivé kategórie.



Obrázok 2.4 Stromové zobrazenie hierarchie kategórií.

Problém pri kategorizácii je rozhodnúť sa, ktorá kategória je pre dané dáta tá najlepšia v rámci hierarchie, teda na ktorej úrovni sa v rámci stromovej hierarchie nachádza. Ak máme k dispozícii jeden objekt, učitelia sa agent nemôže preferovať jednu z kategórií do ktorých patrí, aj keď niektoré môže mierne zvýhodniť. Až pokiaľ máme k dispozícii viac objektov toho istého názvu, je možné konkrétnejšie určiť, do ktorej kategórie v rámci hierarchie patrí. Pre lepší názorný príklad, ak dostaneme jedného labradora, ako príklad novej kategórie napríklad s názvom „*abc*“, nie je pre nás jasné či patria objekty „*abc*“ do kategórie „*living things*“, „*animal*“, „*dog*“ alebo „*labrador*“. Keď ale dostaneme viac labradorov s názvom „*abc*“, pravdepodobne „*abc*“ budú patriť do viac konkrétnej kategórie, ako len do kategórie „*animal*“. Pri viacerých príkladoch z novej kategórie môžeme s vysokou pravdepodobnosťou určiť konkrétnu kategóriu, pod ktorú nové objekty patria a následne hierarchiu rozšíriť o túto novú kategóriu. Na nasledujúcom obrázku 2.5 sú graficky znázornené hypotézy a pridávané nové objekty. Pravdepodobnosť príslušnosti nového objektu do kategórie sa dá vyjadriť pomocou vierohodnosti $P(d|h)$, čiže pravdepodobnosť skúmania dát d patriacich do kategórie za predpokladu, že hypotéza h je pravdivá.



Obrázok 2.5 Zobrazenie hypotéz a veľkosť pravdepodobnosti príslušnosti objektov [1].

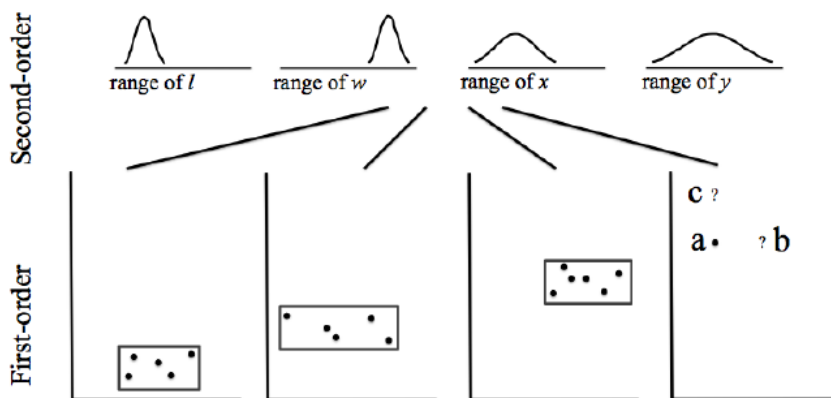
Pravdepodobnosť príslušnosti do kategórie je znázornená tmavosťou obdĺžnika. Vo všeobecnosti, hypotézy s viac obmedzeniami, čiže ktoré sú viac konkrétne, odpovedajú menším obdĺžnikom a dávajú väčšiu vierohodnosť pre dané dáta. Aj väčšie obdĺžniky, hypotézy zahŕňajú dáta z menších vnorených hypotéz, ale tie menej konkrétne vyjadrujú príslušnosť do kategórie. Formálne, pravdepodobnosť príslušnosti jedného príkladu do hypotézy h je nepriamo úmerná veľkosti hypotézy. Toto je dôvod prečo sú na obrázku 2.5 preferované menšie obdĺžniky. Keď ale máme dát viac, tak vierohodnosť hypotézy h s n odpovedajúcimi príkladmi je nepriamo úmerná k veľkosti h a ešte je umocnená na n . Preferencia pre menšie regióny oproti veľkým teda stúpa exponenciálne s počtom príkladov. Tento predpoklad je často označovaný ako princíp veľkosti [6].

2.3 Induktívne obmedzenia

Pri kategorizácii dát často vzniká potreba viacstupňovej generalizácie a s tým súvisiaci problém, ako z malého počtu dát získať nejaké vlastnosti, ktoré sú spoločné pre všetky príklady kategórie. Kategóriu môžeme počas učenia ohraničiť nejakými vlastnosťami, induktívnymi obmedzeniami. V prípade učenia detí je možné pozorovať, že sa dokážu naučiť aj z veľmi malého počtu príkladov zaradiť objekty do správnych kategórií a tak isto ich správne generalizovať. Tento jav je umožnený vďaka nadobudnutým induktívnym obmedzeniam, ktoré deti získali počas predchádzajúceho učenia. Jedným z príkladov takýchto obmedzení je takzvaný „*shape bias*“ vysvetľujúci učenie konceptov [7], alebo univerzálna

gramatika používaná pri získavaní lingvistických znalostí [8]. Tieto indukčné obmedzenia sú abstraktné znalosti nad kategóriami na rôznych úrovniach. Preto môžeme vytvárať hierarchie indukčných obmedzení a na základe nich potom dáta priradovať do kategórií v nižších vrstvách.

Hierarchické indukčné obmedzenia ukážeme na nasledujúcom príklade znázornenom na obrázku 2.6. Máme priestor hypotéz, ktoré predstavujú obdĺžniky. Každý obdĺžnik má nejaké vlastnosti, atribúty ako súradnice x a y a ďalej výšku h a šírku w . Ďalej máme dané body a, b, c , pričom iba o bode a vieme, do akej kategórie patrí. Snažíme sa rozhodnúť, ktorý z bodov b a c patrí s väčšou pravdepodobnosťou do tej istej kategórie ako a .



Obrázok 2.6 Indukčné obmedzenia na vyššej úrovni [1].

Na vyššej úrovni, nad priestorom hypotéz máme priestor indukčných obmedzení, ktorý obsahuje distribúcie pravdepodobnosti jednotlivých atribútov hypotéz z nižšej úrovne. Tieto distribúcie sú nadobudnuté z už poznaných kategórií na tej istej úrovni. Z distribúcie pravdepodobnosti o výške a šírke je zrejmé, že obdĺžniky zobrazujúce hypotézy majú väčšiu šírku ako výšku. Na základe týchto indukčných obmedzení určíme, že b s väčšou pravdepodobnosťou patrí do tej istej kategórie ako a . V spomenutom príklade môžeme sledovať dve úrovne hypotéz. Prvá úroveň obsahuje základné hypotézy, kategórie a druhá, vyššia úroveň, obsahuje abstraktné hypotézy – indukčné obmedzenia. Pre indukčné obmedzenia vyššej úrovne bol zavedený názov metahypotézy (z angl. overhypothesis) [2].

2.4 Hierarchický Bayesovský model

Vo formálnych modeloch vznikajú problémy ako definovať metahypotézy. Je zložité pochopiť, ako je možné sa naučiť niečo také abstraktné, ako je metahypotéza. Taktiež sa naskytuje otázka, ako sa naučiť indukzívne obmedzenia ešte na vyššej úrovni ako metahypotéza, takzvané meta-metahypotézy. Niektoré modely konekcionizmu [9] už prekonali a vyriešili tieto otázky, ako napríklad modely, ktoré dokázali samy získavať „*shape bias*” alebo aj iné typy abstraktných znalostí. Tieto modely však nedokázali jasne rozlíšiť znalosti na rozličných stupňoch abstrakcie a nedokázali určiť, ktoré metahypotézy sú zodpovedné za ich úspešné predikovanie priradovania do kategórií.

Pre vysvetlenie princípov ako je možné naučiť sa metahypotézy, bol navrhnutý hierarchický Bayesovský model (HBM) [2]. Tento model zahŕňa reprezentácie metahypotéz na viacerých úrovniach abstrakcie a tiež ukazuje, ako môžu byť znalosti na viacerých úrovniach získavané. HBM taktiež zjednocuje dva rozdielne prístupy ku kognitívnemu vývinu a to

- „*bottom - up*“ prístup – navrhuje, že konkrétne znalosti sú dostupné pred abstraktnými a abstraktné znalosti sú získané generalizáciou konkrétnych znalostí [10]
- „*top - down*“ prístup – navrhuje, že abstraktné znalosti sú dostupné skôr ako konkrétne znalosti [11]

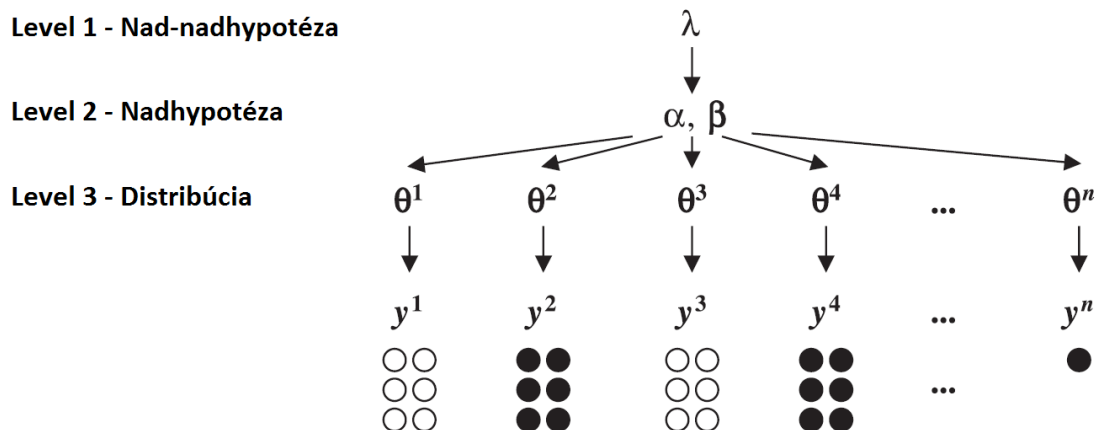
Pre príklad v „*top - down*” prístupe sa deti najskôr naučia napríklad kategóriu „*rastlina*“ alebo „*zvier*a“ a až potom konkrétnejšie kategórie ako „*strom*“ alebo „*ryba*“.

2.4.1 Metahypotézy v HBM

Ako prvý prišiel s pojmom metahypotéza Goodman a princíp tohto pojmu bol vysvetlený na ťahaní farebných guľčiek z vreciek [12]. Predstavme si množinu S , ktorá obsahuje viacero vreciek naplnených guľčkami. Pozrieme sa do niektorých vreciek a zistíme, že niektoré obsahujú iba čierne guľčky a iné zasa iba biele. Guľčky sú teda v každom vrecku jednotné vo farbe. Potom si vyberieme nové vrecko n a vytiahneme z neho guľčku, ktorá je čierna. Vytiahnutie iba jednej guľčky by nám malo povedať iba malú informáciu o tom aké budú zostávajúce, ale po predchádzajúcom skúmaní môžeme vysloviť hypotézu h : „*Všetky guľčky*

vo vrecúšku sú čierne.“ Túto hypotézu tvrdíme na základe vytvorenej metahypotézy z predchádzajúceho skúmania a tou je metahypotéza θ : „Každé vrecúško v množine S obsahuje guľičky s jednotnou farbou.“ Za metahypotézu teda považujeme hocijakú formu abstraktných znalostí, ktoré ovplyvňujú priestor hypotéz na nižšej úrovni. V našom prípade metahypotéza hovorí priestoru hypotéz o guľičkách vo vrecúšku n , že sú jednotnej farby.

Popíšeme tento príklad formálne. Máme k dispozícii dáta, z ktorých chceme získať metahypotézy. Dáta sú v našom prípade pozorovania viacerých vrecúšok. Pozorovanie pre i -te vrecúško označme y_i . Teraz chceme získať možnosť predikovať farbu každej ďalšej vytiahnutej guľičky z vrečka n . Prvý krok je získať znalosť na prvej úrovni, level 1. Táto znalosť bude predstavovať distribúciu farieb θ v každom vrecku. θ^i označuje distribúciu v i -tom vrecku. Na získanie znalosti na úrovni 1 nám pomôže viac metahypotéza na úrovni 2. Táto metahypotéza je distribúcia premenných v θ . V našom prípade to je len premenná „farba“, ale v iných prípadoch pri viacrozmerných dátach to môže byť distribúcia viacerých premenných napríklad „farba“, „tvar“, „veľkosť“. Metahypotéza, a teda distribúcia farby na úrovni 2 bude reprezentovaná dvoma parametrami α a β . α vyjadruje rozsah, v akom sú guľičky v konkrétnom vrecku jednotné vo farbe, inými slovami reprezentuje variabilitu farieb. β vyjadruje priemer distribúcií farieb vo všetkých vreckách, ktoré máme k dispozícii. Hodnota týchto parametrov je založená na znalosti λ z ďalšej úrovne, a to z úrovne 3. Táto znalosť nám hovorí o hodnotách α a β . Parameter λ a dvojica parametrov (α, β) sú metahypotézy, pretože nám vravia niečo viac o priestore o úroveň nižšie. V našom prípade pre zjednodušenie predpokladáme, že znalosť na úrovni tri máme k dispozícii popredu, čiže poznáme jej apriórnu pravdepodobnosť. Na nasledujúcom obrázku 2.8 je znázornená hierarchická štruktúra znázorňujúca jednotlivé vrstvy a premenné v HBM.



Obrázok 2.6 Grafické znázornenie HBM [2].

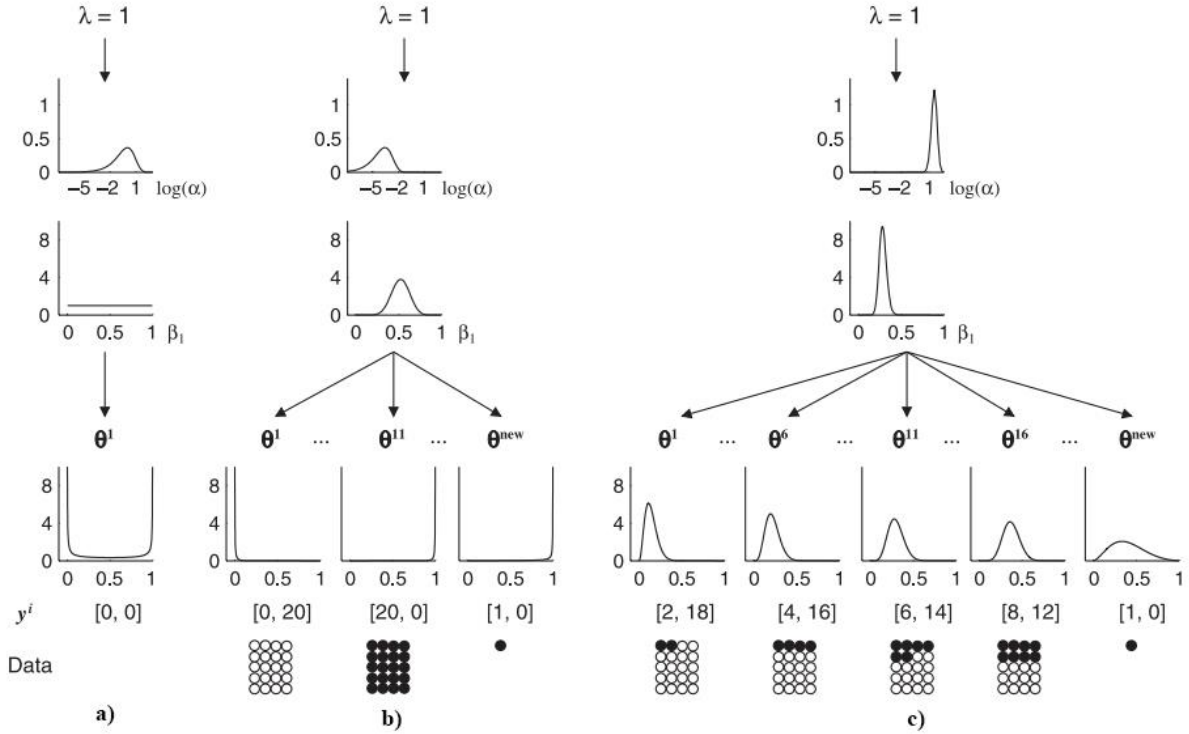
Model zobrazený na obrázku 2.8 je takzvaný Dirichletov-multinomiálny model [13].

2.4.2 Získavanie metahypotéz

Je známe, že hierarchické modely sú vhodné na ukázanie získavania metahypotéz, napríklad v príkladoch učenia gramatiky [2]. Výhoda HBM je, že tento model vysvetľuje, ako je možné získať metahypotézy pomocou štatistickej inferencie. Z obrázka 2.8 vyplýva, že získanie metahypotézy je vlastne získanie nejakých abstraktných znalostí na úrovni 2 v hierarchii. To, ako sú guľičky vo vreckách jednotné vo farbe, môžeme v našom prípade vyjadriť pomocou posteriórnej alebo podmienenej pravdepodobnosti $P(\alpha, \beta | y)$ – ak máme dáta y , ako je pravdepodobné, že farba guľičiek je jednotná. Na výpočet posteriórnej pravdepodobnosti použijeme numerickú integráciu alebo Markovovské reťazové Monte Carlo. Túto pravdepodobnosť potom použijeme pri výpočte pravdepodobnosti θ^i na úrovni 3, ktorá bude určená vzťahom 2.2.

$$P(\theta^i | y) = \int_{\alpha, \beta} P(\theta^i | \alpha, \beta, \lambda) P(\alpha, \beta | y) d\alpha d\beta \quad 2.2$$

Pomocou vzťahu 2.2 určíme pravdepodobnosť distribúcie θ^i , za prítomnosti metahypotéz α, β a λ . Analogicky sa dá tento vzťah použiť aj pri viacerých metahypotézach. Na obrázku 2.9, sú zobrazené konkrétne príklady generalizácií hierarchického Bayesovského modelu a jednotlivé distribúcie pravdepodobností.



Obrázok 2.7 Konkrétne príklady HBM [2].

Na prvej časti a) sú znázornené apriórne distribúcie $\log(\alpha), \beta$ a θ^1 , ktoré indikujú očakávania modelu ešte pred uvidením nejakých dát. Na časti b) sú znázornené podmienené pravdepodobnosti, ak sme videli už zloženie 10 vreciek iba s bielymi a 10 vreciek iba s čiernymi guľčikami. Na poslednej časti c) sú znázornené podmienené distribúcie po videní 20 vreciek so zmiešaným obsahom. Hodnota β sa pohybuje okolo 0.25, pretože iba približne 25% guľčiek je čiernych. Tieto podmienené pravdepodobnosti β nám predikujú, že nové vrečko bude mať s vyššou pravdepodobnosťou viac bielych guľčiek.

2.5 Zhrnutie

Predstavili sme hierarchický Bayesovský model určený na hierarchické kategorizovanie dát, ktorý je založený na Bayesovskej inferencii vo viacerých vrstvách. Popísali sme metahypotézy, na základe ktorých sa hierarchie vytvárajú a vysvetlili sme, ako tento model môže na základe metahypotézy priradovať dáta do kategórií. Nevýhoda HBM je, že metahypotézy musia byť dané, čiže si model nevie sám určiť, podľa čoho je vhodné vykonávať kategorizáciu. Na najvyššej vrstve model vychádza zo znalosti apriórnej pravdepodobnosti, ktorú nemusíme mať vždy k dispozícii. Tieto nedostatky do istej miery rieši iný typ výpočtového modelu na hierarchickú kategorizáciu predstavený v nasledujúcej kapitole - inkrementálna hierarchická diskriminačná regresia.

3 Inkrementálna hierarchická diskriminačná regresia

V tejto kapitole predstavíme ďalšiu z metód používaných na hierarchickú kategorizáciu dát a to hierarchickú diskriminačnú regresiu (IHDR) [3]. Táto metóda je vlastne inkrementálna verzia algoritmu HDR [14], obohatená o možnosť učenia online v reálnom čase. Metóda IHDR je založená na vytváraní regresného stromu. Vo vrcholoch vytvárame dva typy klastrov na základe vstupov a výstupov z vstupných príkladov. Vrcholy stromu okrem klastrov predstavujúcich kategórie obsahujú model distribúcie pravdepodobnosti, štatistické vlastnosti a vlastný podpriestor najviac diskriminačných črt. Hĺbkou v strome sa zvyšuje jemnosť distribúcie pravdepodobnosti a znižuje sa rozptyl dát v jednotlivých vrcholoch a klastroch. Na rozdiel od tradičných techník diskriminačnej analýzy používa negative-log-likelihood (NLL) prispôsobený počet pridávaných príkladov. Metóda je určená najmä pre vytváranie modelov hierarchií kategórií vysokorozmerných dát.

3.1 Biologická motivácia

Tak ako hierarchický Bayesovský model popísaný v predchádzajúcej kapitole, aj metóda IHDR je inšpirovaná človekom, v tomto prípade konkrétne ľudským mozgom. Správanie výpočtového modelu IHDR je podobné ako automatický vývoj asociatívnej kôry v ľudskom mozgu, ktorá uchováva asociácie medzi vnemami a motorickými akciami. Pri pohľade na ľudský mozog sa môžeme inšpirovať a snažiť sa zostrojiť analogický výpočtový model, ktorý dokáže simulovať prirodzenú kategorizáciu dát a vytváranie asociácií medzi vonkajšou a vnútornou reprezentáciou objektov. Mozgová kôra sa vyvíja, prispôsobuje a samoorganizuje postupne s prichádzajúcimi vstupnými signálmi [3]. Mozgovú kôru rozdeľujeme na viac častí. Jednou z nich je časť, v ktorej máme uložené asociácie medzi primárnou a motorickou kôrou. Matematický model, ktorý opíšeme v tejto kapitole, zachytáva približný model pre vývoj asociatívnej kôry, pričom hlavným cieľom je zachytiť vzťahy medzi senzorickými vnemami a motorickými akciami. Ako konkrétny príklad si môžeme predstaviť snahu o vytvorenie

mapovania medzi vstupmi zo senzorov robota a výstupmi, čo môžu byť nejaké motorické akcie, ktoré vykoná. Taktiež samotné vytváranie hierarchie kategórií je inšpirované kategorizovaniu vnemov a objektov u ľudí, pretože v ľudskom kategorizovaní je prirodzené si vytvárať medzi jednotlivými kategóriami relácie, ktoré popisujú hierarchické vzťahy medzi kategóriami. Na základe objektov priradených do konkrétnej kategórie dokážeme určiť najviac špecifické vlastnosti pre túto kategóriu. V IHDR ich označujeme pojmom najviac diskriminačné črty a sú analógiou metahypotéz v HBM. Pomocou týchto črt potom dokážeme prechádzať hierarchiu kategórií a zaradiť nový objekt do správnej kategórie.

3.2 Dátové štruktúry

Algoritmus IHDR vytvára automaticky kategórie, ktoré sú organizované do hierarchie. Pre reprezentáciu hierarchie je najlepšie použiť ako dátovú štruktúru strom. Klasifikačné a regresné stromy majú dva základné účely: indexovanie a predikciu [3]. Stromy určené na indexovanie dát ako K-D stromy alebo R-stromy sú napríklad veľmi využívané v databázach a zaručujú logaritmickú zložitosť pri vyhľadávaní dát. Predikčné alebo rozhodovacie stromy sú zasa využívané v strojovom učení na predikciu neznámych dát. Rozhodovacie stromy sú väčšinou skonštruované cielene s menšou hĺbkou pre redukciu času prehľadávania. Keď zvážime vlastnosti stromov a požiadavky ktoré boli kladené na IHDR algoritmus, pre logaritmickú zložitosť a jednoduchú reprezentáciu generalizácie a konkretizácie sú stromy najlepšia voľba.

V jednoduchých klasifikačných a regresných stromoch (CART [6]) sa používa v každom internom vrchole univariantné rozdelenie. Toto znamená, že na rozdelenie priestoru každým vrcholom sa používajú nadroviny ktoré sú ortogonálne k osiam vstupného priestoru X [3]. Pri multivariantnom rozdelení nemusí byť nutne nadrovina ortogonálna ku žiadnej osi vstupného priestoru a preto môže rozdeliť priestor lepšie a tým dosiahnuť lepšiu predikciu. Stromy ktoré používajú multivariantné rozdelenie priestoru sa nazývajú šikmé stromy (oblique trees). Existuje viacero algoritmov na vytváranie šikmých stromov napríklad OC1 [2] alebo SHOLIF [3]. OC1 používa iteratívne vyhľadávanie v rovine k nájdeniu rozdelenia a SHOLIF používa na výpočet rozdelenia metódu hlavných komponentov (PCA) a lineárnu diskriminačnú analýzu (LDA). Pomocou LDA a PCA môžeme určiť hranicu medzi kategóriami a odfiltrovať

atribúty, ktoré nie sú dôležité pre výsledok. Problém sa stáva ale ťažší, keď pracujeme s vysokorozmernými dátami, alebo jednotlivé dáta prichádzajú postupne v reálnom čase. Pokiaľ nemáme pri dátach názov triedy ako výstup, LDA nie je priamo aplikovateľná.

V algoritme IHDR je prezentovaná metóda pri ktorej sa vytvárajú klastre v dvoch priestoroch naraz a to vo vstupnom X a vo výstupnom priestore Y . Klastre vo výstupnom priestore predstavujú virtuálne označenia tried pomocou ktorých sa formujú klastre vo vstupnom priestore. V IHDR je použité multivariantné nelineárne rozdelenie, s multivariantným lineárnym rozdelením v špeciálnych prípadoch [3].

3.3 Klasifikácia a regresia

Diskriminačná analýza môže byť rozdelená na dva typy podľa výstupu. Výstup môže byť:

- označenie triedy
- numerický výstup alebo vektor

Ak je výstup označenie triedy, nazývame úlohu klasifikácia. V druhom prípade je nazývaný proces predikcie regresia. Pre IHDR je potrebné, aby bol typ úlohy regresia, pretože formujeme klastre aj podľa výstupného priestoru a musíme dokázať určiť nejakú vzdialenosť alebo podobnosť dvoch príkladov na základe výstupu. Pri klasifikácii nemáme informáciu ako je trieda A podobná triede B, preto musíme klasifikáciu transformovať nejakým spôsobom na regresiu. Klasifikácia je formálne určená nasledovne : Ak máme množinu príkladov L (3.1)

$$L = \{(x_i, l_k) | i = 1, 2 \dots n, k = 1, 2 \dots c\} \quad (3.1)$$

kde $x_i \in X$ a l_k je označenie triedy alebo kategórie, do ktorej x_i patrí, pod klasifikáciou rozumieme určenie l_k neznámeho vstupu $x_i \in X$. Definícia regresie je veľmi podobná. Jediný rozdiel je zmena označenie triedy l_k na vektor y_i .

Existuje viacero spôsobov ako transformovať klasifikáciu na regresiu [14].

1. Ak máme k dispozií maticu $[c_{ij}]$, kde c_{ij} určuje mieru podobnosti medzi triedami i a j , n označení tried môžeme vložiť do $n - 1$ rozmerného výstupného priestoru priradením vektoru y_i k triede $i, i = 1, 2, \dots, n$ tak, že $\|y_i - y_j\|$ je najbližšie k c_{ij} ako je možné. Táto metóda ale nie je veľmi často používaná, pretože maticu $[c_{ij}]$ nie je jednoduché určiť.
2. Kanonické mapovanie – namapovanie c označení tried do c -dimenzionálneho výstupného priestoru tak, že i -te označenie triedy zodpovedá vektoru, kde je i -ty prvok 1 a ostatné prvky sú nuly. V tomto prípade je vzdialenosť medzi jednotlivými vektormi vždy rovnaká.
3. Mapovanie značiek kategórií do vstupného priestoru. Každý príklad (x_i, l_k) , ktorý patrí do triedy k je premenený na (x_i, y_k) kde je y_k priemer všetkých x_i patriacich do rovnakej triedy. Pomocou tejto metódy máme ako výstup dáta, ktoré majú medzi sebou rôzne vzdialenosti. Táto metóda je vhodná v mnohých prípadoch.

Je však zrejmé, že je nemožné namapovať výstupný priestor do obmedzenej množiny značiek kategórií bez straty nejakých vlastností medzi nekonečným množstvom číselných vektorov príkladov. Pre to je regresia viac všeobecný problém ako klasifikácia.

3.4 Východiská IHDR

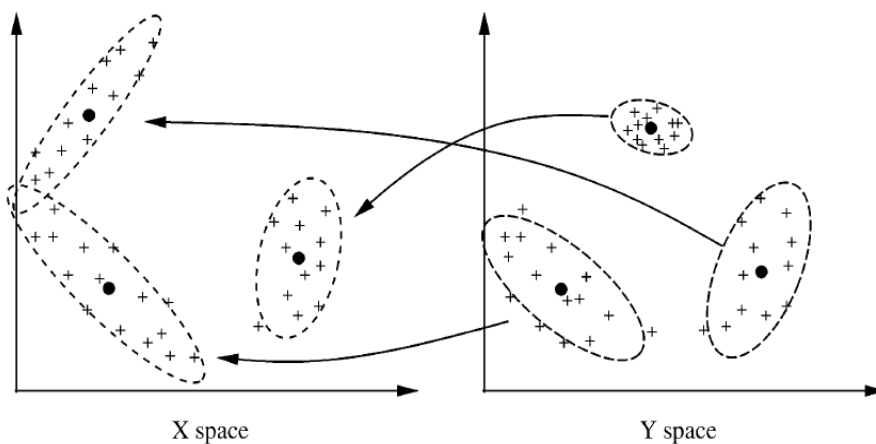
3.4.1 Diskriminačná analýza pre numerický výstup

Definujeme všeobecný problém regresie, a to aproximačné mapovanie $h : X \rightarrow Y$ z množiny trénovacích príkladov $\{(x_i, y_i) | x_i \in X, y_i \in Y, i = 1, 2, \dots, n\}$ medzi vstupom a výstupom. Ak y_i je označenie triedy, môžeme použiť známu metódu lineárnu diskriminačnú analýzu (LDA) s využitím rozptylových matíc vrámci triedy a medzi triedami (within-class scatter, between-class scatter matrix) [15]. Pokiaľ ale niektoré kategórie obsahujú málo príkladov, rozptylová matica vrámci triedy je veľmi nepresne určená a LDA potom nie je efektívna a presná. Ak premeníme klasifikačný problém na problém regresie, je možné vytvárať viac všeobecné kategórie s väčším počtom príkladov zlúčením príbuzných kategórií s menším počtom

príkladov. Táto procedúra nám umožní lepšie určenie rozptylovej matice vrámci triedy a teda kategorizovať aj príklady, ktorých kategória nie je ešte dostatočne určená.

3.4.2 Dvojitá klasterizácia

Výstupný vektor y_i môže mať na každom prvku ľubovoľnú hodnotu, ktorá môže, ale aj nemusí hovoriť o príslušnosti do konkrétnej kategórie. Je ťažké určiť efektívnu diskriminačnú analýzu atribútov a tak odfiltrovať atribúty irelevantné pre výsledok. Práve pre tento prípad bola navrhnutá hierarchická štatistická metóda hierarchická diskriminačná regresia IHDR. Táto metóda vytvára regresný strom, v ktorom sa automaticky determinujú najviac diskriminačné črty vysokorozmerných príkladov bez prítomnosti označenia, do ktorej triedy patria. V IHDR metóde sa formujú klastre v dvoch priestoroch a to v priestore X a Y . V X priestore sú to x -klastre a analogicky v Y priestore y -klastre. Mapovanie h je prítomné práve medzi týmito klastrami. Na obrázku 3.1 sú znázornené oba priestory X a Y s ich klastrami a mapovaním h medzi klastrami.



Obrázok 3.1 Zobrazenie klastrov v dvoch priestoroch a mapovanie medzi klastrami [3].

V každom vrchole vytváraného stromu sa nachádza maximálne q klastrov každého typu. Y -klastre určujú takzvané virtuálne označenie triedy každého vstupného príkladu (x, y) na základe y častí z príkladov. Virtuálne označenie je použité na selekciu x -klastra pomocou mapovania h , ktorý má byť modifikovaný pridaním vstupu x . Pokiaľ máme v klastroch

dostatočné množstvo príkladov a potrebujeme jemnejšiu aproximáciu, môžeme vrchol rozvetviť do nových synov, pričom príklady rozdistribuuujeme do synov a tak vytvoríme nové klastre, ktoré predstavujú konkrétnejšiu triedu.

V každom listovom vrchole hľadáme najbližší alebo najpodobnejší y -klastre jednoducho pomocou Euklidovskej vzdialenosti. Pomocou najbližšieho y -klastra určíme, ktorý x -klastre má byť aktualizovaný. Aktualizácia klastra znamená prepočítanie štatistických vlastností klastra. Štatistické vlastnosti uchovávané v x -klastroch sú kovariančná matica a priemer. Aby nebolo potrebné počítat' priemer a kovariančnú maticu zo všetkých príkladov v x -klastri, IHDR používa inkrementálne počítanie. Štatistické vlastnosti klastrov sú použité k určení pravdepodobnosti príslušnosti nového príkladu do daného klastra. Táto pravdepodobnosť je modelovaná ako viacrozmerný Gausián, ktorý je zložený z menších Gausiánov z nižších úrovní. Pomocou q x -klastrov je možné určiť $(q - 1)$ -rozmerný priestor najviac diskriminačných črt' dát a následne pravdepodobnosť príslušnosti do kategórie počítat' iba na tomto priestore.

3.4.3 Štatistické parametre klastrov a vrcholov

Postupným pridávaním príkladov do stromu upravujeme a počítame štatistické vlastnosti jednotlivých klastrov alebo celých vrcholov. Tieto štatistické vlastnosti budú použité na výpočet vzdialeností a pravdepodobností príslušnosti príkladu do klastra. Najdôležitejšie vlastnosti sú priemer \bar{x} (3.2), variancia σ^2 (3.3) a kovariančná matica Γ (3.4).

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i \quad 3.2$$

$$\sigma^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2 \quad 3.3$$

$$\Gamma = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})(x_i - \bar{x})^T \quad 3.4$$

Priemer klastra počítame ako aritmetický priemer všetkých príkladov, variancia je stredná kvadratická odchýlka od priemeru a kovariančná matica je matica kovariancií medzi jednotlivými prvkami vektora. Na pozícii $[i, j]$ v kovariančnej matici Γ máme kovarianciu i -teho a j -teho atribútu zo všetkých vstupných dát. Na diagonále kovariančnej matice dostaneme spomínanú varianciu.

Keďže pri každom novom príklade potrebujeme prepočítať štatistické vlastnosti vrchola, bolo by veľmi neefektívne počítať ich vždy nanovo. Preto je vhodné použiť pre výpočet inkrementálne verzie vzťahov. V inkrementálnej verzii IHDR je vhodné zohľadňovať aj v akom čase boli príklady pridané. Preto je v IHDR použité amnézické inkrementálne počítanie štatistických vlastností, v ktorom na staršie príklady zabúdame a nové príklady berieme viac do úvahy.

3.4.4 Amnézický priemer v IHDR

Pri inkrementálnom vytváraní hierarchických kategórií sú počiatočné centrá klastrov najviac determinované prvými príkladmi. Keď prídu nové dáta, centrá sú viac alebo menej posúvané na novú lokáciu. Ak sú nové centrá potom použité na určenie klastrov a ich hraníc, niektoré, skôr pridané dáta môžu byť pridané do nesprávneho klastra, pretože v čase ich pridania ešte nebol klaster dostatočne určený. Inými slovami centrum alebo stred klastra je vypočítaný aj pomocou dát, ktoré nepatria do tohto klastra. Aby sme redukovali vplyv skorších dát na výpočet stredu klastra, používame na jeho výpočet amnézický priemer namiesto obyčajného, pretože lepšie reaguje na dynamické zmeny medzi dátami. Nasledujúci vzťah (3.5) vyjadruje výpočet priemeru z dát $x_1, x_2, x_3, \dots, x_t$

$$\bar{x}^{(t)} = \frac{1}{t} \sum_{i=1}^t x_i = \sum_{i=1}^t \frac{1}{t} x_i \quad 3.5$$

V predchádzajúcom vzťahu má vo výsledku každý vstup tú istú váhu $1/t$. Keď prichádzajú príklady postupne, potrebujeme inkrementálny výpočet priemeru kvôli redukcii času výpočtu. Z predchádzajúceho vzťahu ľahko odvodíme vzťah na inkrementálny výpočet (3.6)

$$\bar{x}^{(t)} = \frac{(t-1)\bar{x}^{(t-1)} + x_t}{t} = \frac{t-1}{t} \bar{x}^{(t-1)} + \frac{1}{t} x_t \quad 3.6$$

Vo vzťahu na výpočet amnézického priemeru (3.7) je pridaný amnézický parameter $\mu \geq 0$. Vďaka tomuto parametru bude nový príklad pripočítaný s väčšou váhou ako predchádzajúce príklady.

$$\bar{x}^{(t)} = \frac{t-1-\mu}{t} \bar{x}^{(t-1)} + \frac{t+\mu}{t} x_t \quad 3.7$$

Napríklad pre $\mu = 1$ bude váha pre nový príklad zdvojnásobená. Ak bude t veľmi veľké číslo, amnézická váha pre nové dáta $(1+\mu)/t$ sa ale bude blížiť k 0. To znamená, že ak máme neohraničený počet príkladov, nové dáta nebudú takmer vôbec použité na úpravu štatistických vlastností klastra, preto je nutné meniť μ dynamicky. V IHDR sa z tohto dôvodu používa namiesto μ funkcia $\mu(t)$, ktorá dynamicky ovplyvňuje váhu príkladu pri výpočtoch na základe času. Pri výpočte $\mu(t)$ sa používajú dva hraničné body t_1 a t_2 a to nasledovne

- Ak $t \leq t_1$ $\mu(t) = 0$
- Ak $t_1 < t \leq t_2$ $\mu(t) = c(t - t_1) / (t_2 - t_1)$
- Ak $t_2 \leq t$ $\mu(t) = c + (t / t_2) / m$

V druhom prípade je dôležitý parameter c , pomocou ktorého sa mení μ lineárne od 0 po c . Ak máme neohraničený počet príkladov, po prekročení hranice t_2 je váha príkladu $1/m$ ako vidno z nasledujúcej limity .

$$\lim_{t \rightarrow \infty} \left(1 + \frac{\mu(t)}{t}\right) = \frac{1}{m} \quad 3.8$$

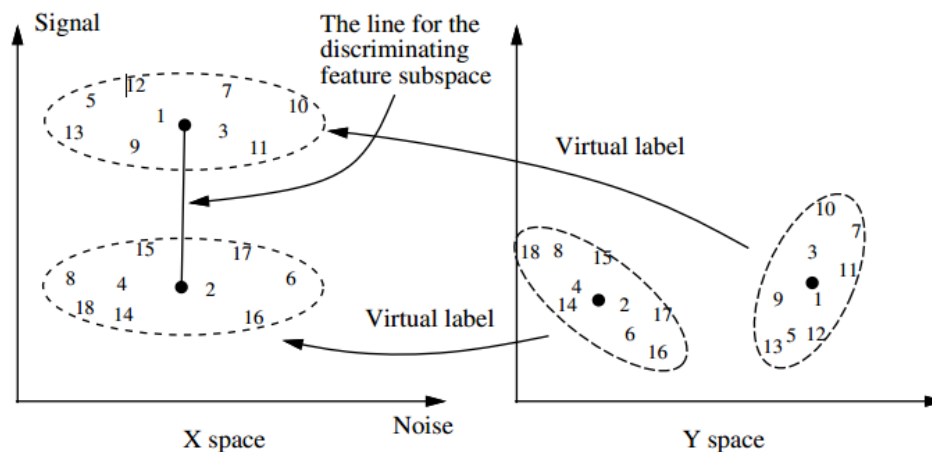
Tak isto ako počítanie priemeru potrebujeme inkrementálne počítať aj kovariančnú maticu. Vzorec na inkrementálny výpočet kovariančnej matice (3.9) je obdobný ako pri priemere. Do vzťahu môžeme už zahrnúť aj funkciu $\mu(t)$.

$$\Gamma_x^{(t)} = \frac{t-1-\mu(t)}{t} \Gamma_x^{(t-1)} + \frac{t+\mu(t)}{t} (x_t - \bar{x}^{(t)})(x_t - \bar{x}^{(t)})^T \quad 3.9$$

Predchádzajúce vzťahy používajú t stupňov voľnosti namiesto $t-1$ z nasledujúceho dôvodu. Keď máme k dispozícii jeden príklad x_1 , výsledná kovariančná matica by nemala byť určená ako nulový vektor, pretože často môže určovať nejakú udalosť, ktorá sa nedá presne odmerať. Preto iniciálna kovariančná matica $\Gamma_x^{(1)}$ bude určená ako $\sigma_2 I$, kde σ_2 je očakávaný šum a I je identická matica príslušnej dimenzie.

3.4.5 Podpriestor najviac diskriminačných črt

Pri kategorizovaní dát sa často stretávame s vysokorozmernými dátami. Pri práci s takýmito dátami je zrejmé, že výpočty môžu trvať oveľa dlhšie ako pri menej rozmerných dátach. Často však niektoré atribúty nemajú vôbec výpovednú hodnotu o tom, do ktorej kategórie príklad patrí. Preto je vhodné tieto atribúty ignorovať a následne určovať príslušnosť do kategórie iba na podmnožine atribútov dôležitých pre príslušnosť do kategórií. Na obrázku 3.2 je zobrazený príklad, pri ktorom je možné ignorovať jeden z atribútov. V tomto prípade môžeme ignorovať horizontálny komponent, pretože o príslušnosti do kategórie nám hovorí len vertikálny.



Obrázok 3.2 Zobrazenie príkladu, v ktorom je možné pri kategorizovaní ignorovať atribút „Noise“ [3].

Pri výpočtovom modeli IHDR máme v každom vrchole stromu uložený priemer a kovariančnú maticu. Často sa stretávame s prípadmi, keď je ako vstup napríklad obrázok, čiže vstupný vektor bude obsahovať ako atribúty jednotlivé pixle obrázka. Keď má vstupný obrázok veľkosť napríklad 100 x 80 pixelov tak vstupný vektor má dimenziu 8000. Z toho vyplýva, že kovariančná matica je veľkosti 8000 x 8000. Pri takejto veľkosti matic a vektorov sú matematické operácie nad nimi veľmi zdĺhavé a preto je prakticky nemožné pracovať v pôvodnom priestore. V strome, ktorý sa vytvára pomocou IHDR algoritmus, má každý vrchol svoj podpriestor najviac diskriminačných črt, na ktorom počítame príslušnosť do kategórie. Počítanie v tomto podpriestore nám rapídne zníži výpočtovú zložitosť operácií v algoritme.

IHDR algoritmus sa dokáže vysporiadať s vysokými dimenziami nasledujúcim spôsobom. Ako bolo spomenuté, každý vnútorný vrchol uchováva q x -klastrov. Z týchto klastrov zoberieme množinu stredov C (3.10).

$$C = \{ c_1, c_2, c_3, \dots, c_q \mid c_i \in X, i = 1, 2, \dots, q \} \quad 3.10$$

Pomocou pozícií týchto q stredov klastrov môžeme určiť podpriestor D , v ktorom tieto klastre ležia. Podpriestor najviac diskriminačných črt určíme nasledovne

- Vypočítame váhovaný priemer \bar{C} všetkých stredov q klastrov (3.11)

$$\bar{C} = \frac{1}{n} \sum_{i=1}^q n_i c_i \quad 3.11$$

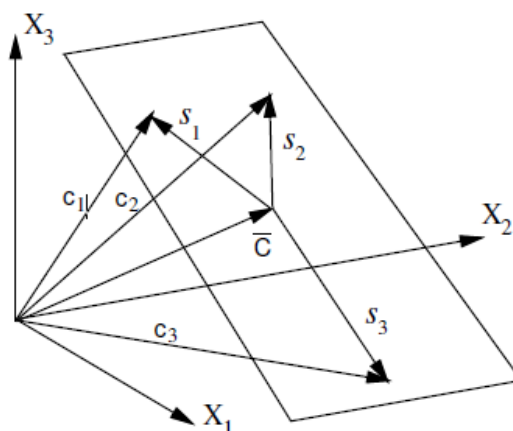
- Určíme rozptylové vektory. Rozptylový vektor s_i je určený ako rozdiel stredu i -teho klastra a váhovaného priemeru všetkých klastrov (3.12).

$$s_i = c_i - \bar{C}, \text{ kde } i = 1, 2, \dots, q \quad 3.12$$

Máme množinu S všetkých rozptylových vektorov s_i . Podpriestor určený množinou vektorov S pozostáva zo všetkých lineárnych kombinácií z týchto vektorov. Tieto vektory nie sú lineárne nezávislé.

- Pomocou Gram-Schmidtovho ortogonalizačného procesu (GSO) [16] určíme ortonormálnu bázu a_1, a_2, \dots, a_{q-1} nového podpriestoru. Pri výpočte použijeme len $q - 1$ rozptylových vektorov, pretože potrebujeme, aby vektory boli lineárne nezávislé.

Na obrázku 3.3 je znázornený príklad s tromi rozptylovými vektormi a lineárnou varietou určenou týmito vektormi.



Obrázok 3.3 Zobrazenie lineárnej variety určenej rozptylovými vektormi [3].

Keď máme vypočítanú ortonormálnu bázu podpriestoru D môžeme vstupný príklad premietnuť do tohto podpriestoru a ďalej počítať už iba s vektorom najviac diskriminačných črt f . Vektor f určíme projekciou vektora $s = x - \bar{C}$ na lineárnu varietu M jednoduchým vynásobením $f = M^T s$, kde M je matica obsahujúca báзовé vektory vypočítané pomocou GSO (3.12).

$$M = [a_1, a_2, \dots, a_{q-1}] \quad 3.12$$

Vďaka tejto procedúre môžeme ďalej pracovať s dátami, ktoré sú oveľa menšieho rozmeru. Pre príklad, ak máme vstupné dáta ľubovoľného rozmeru a v každom vrchole uchováваме 10 klastrov, po výpočte podpriestoru D ďalej transformujeme vstupné dáta do rozmeru 9. Potom nám stačí uchovávať a pracovať s maticami rozmeru iba 9×9 , čo nám umožní oveľa rýchlejšie výpočty. Na druhej strane však strácame presnosť. Vo veľkej miere však záleží od typu vstupných dát.

3.4.6 Metriky založené na pravdepodobnosti

V každom vrchole uchováваме podpriestor najviac diskriminačných črt, ktorý je dimenzie $q - 1$. V tomto podpriestore je potrebné automaticky určovať hranice medzi triedami, založené na distribúcii pravdepodobnosti. Pre rôzne metriky vzdialeností sú určené rozličné modely distribúcie pravdepodobnosti. Pre niektoré metriky je dôležité mať dostatok príkladov

na určenie štatistických vlastností klastrov alebo vrcholov v strome. Berieme do úvahy tri metriky na určenie príslušnosti príkladu do kategórie.

- Gausovská negative-log-likelihood vierohodnosť (Gausovská NLL)
- Mahalanobisova negative-log-likelihood vierohodnosť – (Mahalanobisova NLL)
- Euklidovská negatívna negative-log-likelihood vierohodnosť – (Euklidovská NLL)

Gaussovská NLL (3.13) sa vypočíta pomocou kovariančnej matice Γ_i každého klastra. Na určenie kovariančnej matice potrebujeme dostatočný počet príkladov, inak by bol výpočet značne nepresný. Pri Mahalanobisovej NLL (3.14) sa použije váhovaný priemer všetkých kovariančných matíc klastrov Γ , čo je vlastne rozptylová matica vrámci triedy používaná v LDA. Pri Euklidovskej NLL (3.15) použijeme maticu $\rho^2 I$, ktorá má na diagonále varianciu alebo iba chybu merania. Pre výpočet príslušnosti príkladu x do klastra so stredom c_i použijeme pre jednotlivé metriky nasledujúce vzťahy. Vzťahy sú takmer totožné, mení sa iba matica vyjadrujúca štatistické vlastnosti.

$$G(x, c_i) = \frac{1}{2}(x - c_i)^T \Gamma_i^{-1}(x - c_i) + \frac{q-1}{2} \ln(2\pi) + \frac{1}{2} \ln(|\Gamma_i|) \quad 3.13$$

$$M(x, c_i) = \frac{1}{2}(x - c_i)^T \Gamma^{-1}(x - c_i) + \frac{q-1}{2} \ln(2\pi) + \frac{1}{2} \ln(|\Gamma|) \quad 3.14$$

$$E(x, c_i) = \frac{1}{2}(x - c_i)^T \rho^2 I^{-1}(x - c_i) + \frac{q-1}{2} \ln(2\pi) + \frac{1}{2} \ln(|\rho^2 I^{-1}|) \quad 3.15$$

3.4.7 Prechod medzi metrikami

Keďže v každom vrchole a klastri máme rôzny počet príkladov, musíme sa rozhodnúť, ktorú metriku použijeme. Pre určenie, ktorú metriku zvýhodníme potrebujeme vedieť koľko príkladov potrebujeme na dostatočné určenie matíc použitých v definovaných vzťahoch na výpočet. Pre to je pre každú maticu Γ , Γ_i a $\rho^2 I$ určený estimačný parameter NSPP (z angl. number of scales per parameter). NSPP pre maticu $\rho^2 I$ použitú v Euklidovskej NLL je $(n-1)(q-1)$, pretože jeden príklad nám nedáva žiadnu hodnotu variancie a pre varianciu

potrebujeme určiť $q - 1$ prvkov, čo je vlastne veľkosť vektora variancie v našom podpriestore. Euklidovská NLL je preto určená predovšetkým pre prípad, keď máme v klastri veľmi málo príkladov. Pre Mahalanobisovu vzdialenosť je nutných viac príkladov. Je to z dôvodu, že máme q klastrov a v každej matici potrebujeme určiť $(q - 1)/2$ prvkov, teda pre Γ je nutných $(q - 1)q/2$ prvkov. Počet príkladov je $n - q$, keďže každý klaster má minimálne jeden príklad na určenie svojho priemeru. Každý vektor má dimenziu $q - 1$, preto je potrebné učiť ďalších $(n - q)(q - 1)$ nezávislých skalárov. Z tejto úvahy vyplýva, že NSPP pre Γ budeme počítat podľa vzťahu (3.16)

$$NSPP_{\Gamma} = \frac{(n - q)(q - 1)}{(q - 1)q/2} = \frac{2(n - q)}{q} \quad 3.16$$

Podobné je vyjadrené aj NSPP pre Γ_i

$$NSPP_{\Gamma_i} = \frac{1}{q} \sum_{i=1}^q \frac{2(n_i - 1)}{q} = \frac{2(n - q)}{q^2} \quad 3.18$$

Hodnotu NSPP pri Euklidovskej a Mahalanobisovej NLL je potrebné ešte ohraničiť nejakou hodnotou, aby sme pri väčšom počte príkladov pri prechode medzi metrikami zvýhodnili Gausovskú NLL, keďže jej NSPP rastie najpomalšie. Preto je potrebné zvoliť nejakú hranicu, ktorá obmedzí vplyv matice Γ a $\rho^2 I$ na výpočty vzdialeností. V IHDR sa NSPP pri Euklidovskej a Mahalanobisovej NLL hodnoty ohraničujú parametrom $n_s = 1/\alpha + 1$ [3]. Pri každej metrike je potom určené, s akou váhou prispeje k výsledku. Euklidovská a Mahalanobisova NLL budú zhora ohraňované (3.19, 3.20), Gausovská NLL bude neohraňovaná (3.21). Nasledujúce vzťahy vyjadrujú jednotlivé váhy

$$b_e = \min\{(n - 1)(q - 1), n_s\} \quad 3.19$$

$$b_m = \min\left\{\frac{2(n - q)}{q}, n_s\right\} \quad 3.20$$

$$b_g = \frac{2(n - q)}{q^2} \quad 3.21$$

Nasledovne je prevedená normalizácia týchto váh, pri ktorej dostaneme výsledné váhy $w_e = b_e/b$, $w_m = b_m/b$ a $w_g = b_g/b$, kde b je súčet všetkých váh $b = b_e + b_m + b_g$. Pri určovaní, do ktorého klastra i patrí nový príklad alebo do ktorej časti stromu má pokračovať prehľadávanie, je použitá váhovaná matica vypočítaná podľa nasledujúceho vzťahu

$$W_i = w_e \rho^2 I + w_m \Gamma + w_g \Gamma_i \quad 3.22$$

Dosadením do už známeho vzťahu dostávame váhovanú SDNLL (z angl. názvu size-dependent negative-log-likelihood) (3.23).

$$L(x, c_i) = \frac{1}{2} (x - c_i)^T W_i^{-1} (x - c_i) + \frac{q-1}{2} \ln(2\pi) + \frac{1}{2} \ln(|W_i|) \quad 3.23$$

Vďaka b_e, b_m a b_g , ktoré sa postupným pridávaním príkladov automaticky menia, získame prechod medzi jednotlivými metrikami. Premenné vyjadrujú vlastne vzťah medzi LDA a SDNLL. Vďaka tomu, že sa matica upravuje dynamicky od počtu príkladov v klastroch a vrchoch, môžeme pracovať s dátovými sadami, ktoré napríklad nemajú vyvážené počty príkladov v jednotlivých kategóriách.

3.5 IHDR algoritmus

Samotný IHDR algoritmus je zložený na vytváraní regresného stromu zo vstupu tréningových dát (x_t, y_t) , kde x_t označuje vstupnú časť a y_t výstupnú časť príkladu v čase t . Po každom pridaní príkladu sa prechádza doposiaľ vytvorený strom a na svojej ceste k listom aktualizuje potrebné štatistické parametre vrcholov. Keď príde do listu stromu, tak sa pridá do príslušného klastra. Ak vrchol po pridaní spĺňa podmienky na rozvetvenie, vytvorí q synov a rozdistribuuje medzi nich svoje dáta. Pri rozvetvení zároveň určí vrchol svoj podpriestor najviac diskriminačných čŕt a premietne svoje štatistické parametre do tohto priestoru. Takto

nám vzniká zjemňujúca hierarchická štruktúra, teda čím je vrchol hlbšie v strome, tým má menšiu variáciu priradených príkladov. V nasledujúcej časti popíšeme slovne jednotlivé procedúry algoritmu IHDR. Základné procedúry algoritmu sú *Update-Tree*, *Update-Node* a *Update-ClusterPair*.

3.5.1 Procedúra Update-Tree

Základná procedúra vytvárania IHDR stromu. Na vstupe dostane príklad (x, y) , na základe ktorého sa upraví strom. V tejto procedúre sa používajú nasledujúce parametre a označenia

- q – maximálny počet synov vrcholov
- b_s – hraničná hodnota NSPP používaná pri rozhodovaní, či má byť vrchol rozvetvený.
- A – aktívny vrchol, vrchol ktorý bude ďalej prehľadávaný

Update-tree (x,y):

- 1) Inicializácia – ako A je nastavený koreň stromu
- 2) Vykonaj príslušnú akciu pre aktívny vrchol
 - a) Ak je A vnútorný vrchol a je označený ako plastický zavolaj procedúru *Update-Node*
 - b) Ak je A listový vrchol zavolaj *Update-ClusterPair*
 - c) Ak je A listový vrchol a spĺňa podmienku $NSPP > b_s$, označ A ako vnútorný vrchol a rozvetvi vrchol do q synov, pričom klastre rozdistribuuješ do synov.
- 3) Vypočítaj pomocou SDNLL pravdepodobnosť príslušnosti vloženého príkladu pre všetky klastre vo vrchole
- 4) Syn koreňspodujúci s najpravdepodobnejším klastrom je označený ako nový aktívny vrchol A
- 5) Opakuj kroky 2, 3, 4 pokiaľ nie je aktívny vrchol listový

3.5.2 Procedúra Update-Node

Táto procedúra je určená na aktualizovanie jedného vrchola. Je volaná z predchádzajúcej procedúry *update-tree*. V procedúre *Update-Node* sa používajú nasledujúce parametre a označenia

- n – počet klastrov vo vrchole
- q_c – maximálny počet klastrov vo vrchole
- δ_y - rozlíšenie vo výstupnom priestore Y . Zabraňuje vznikaníu veľmi podobných klastrov
- p – aká časť y -klastrov má byť aktualizovaná (napríklad pre $p = 0.2$ bude aktualizovaných 20 % najbližších y -klastrov

Update-Node(x, y)

- 1) Nájdi klaster y_j , ktorý je najbližší k y pomocou Euklidovskej vzdialenosti

$$j = \operatorname{argmin}_{1 \leq j \leq n} \{ ||y_j - y|| \}$$

- 2) Ak $n < q$ a zároveň $\delta_y > ||y_j - y||$, vytvor nový pár klastrov s jedným príkladom (x, y) .
Inak vykonaj krok 3 a 4.
- 3) Aktualizuj p časť najbližších y -klastrov (aktualizácia priemeru pomocou amnézického výpočtu)
- 4) Aktualizuj x -klaster odpovedajúci najbližšiemu y -klastru (aktualizácia priemeru a kovariančnej matice pomocou amnézického priemeru)
- 5) Aktualizuj podpriestor najviac diskriminačných črt

3.5.3 Procedúra Update-ClusterPair

Posledná z metód je metóda *Update-Cluster-Pair*, ktorá je určená iba pre listové vrcholy. Je podobná predchádzajúcej metóde s rozdielom, že aktualizujeme klastre na základe najbližšieho x -klastra. V procedúre *Update-Cluster-Pair* sa používajú nasledujúce parametre.

- n – počet klastrov vo vrchole
- q_c – maximálny počet klastrov vo vrchole
- δ_x - rozlíšenie vo vstupnom priestore X . Zabraňuje vznikaníu veľmi podobných klastrov

Update-Cluster-Pair(x, y)

- 1) Nájdi klaster x_j , ktorý je najbližší k x , pomocou Euklidovskej vzdialenosti

$$j = \operatorname{argmin}_{1 \leq j \leq n} \{ ||x_j - x|| \}$$

- 2) Ak $n < q$ a zároveň $\delta_x > ||x_j - x||$, vytvor nový pár klastrov s jedným príkladom (x, y) .
Inak vykonaj krok 3 a 4.
- 3) Aktualizuj najbližší x -klaster odpovedajúci najbližšiemu y -klastru (aktualizácia priemeru pomocou amnézického priemeru)
- 4) Aktualizuj y -klaster odpovedajúci najbližšiemu x -klastru (aktualizácia priemeru pomocou amnézického priemeru)

3.6 Porovnanie IHDR a HBM

Predstavili sme dve metódy používané pri hierarchickej kategorizácii dát. Prvá metóda bola viac abstraktná a inšpirovaná kognitívnym vývinom a založená na Bayesovskej inferencii. Druhá metóda je motivovaná vývojom asociatívnej mozgovej kôry ľudského mozgu a je založená hlavne na štatistických metódach. Aj IHDR aj HBM sa snažia v rôznych vrstvách hierarchie získavať nejaké abstraktné znalosti o podkategóriách. V HBM sme používali ako abstraktné znalosti metahypotézy a dáta sme kategorizovali na základe posteriornej pravdepodobnosti podmienenej metahypotézami z vyšších úrovní. V IHDR si môžeme pod viac všeobecnými znalosťami predstaviť najviac diskriminačné črty a príslušnosť do kategórií sme určovali pomocou SDNLL. V oboch metódach sme tiež mali nejaké reprezentácie pre kategóriu, či už hypotézy v HBM alebo klastre v IHDR. Spomínané metódy sa líšili najmä spôsobom získavania abstraktných znalostí. V HBM museli byť metahypotézy dané a ich distribúcie pravdepodobností sa aktualizovali postupne počas učenia. Ďalej sa v tejto metóde vychádza z dopredu danej apriórnej pravdepodobnosti metahypotéz na najvrchnejšej úrovni. V IHDR sme získavali ako abstraktné znalosti najviac diskriminačné črty pomocou rozptylových vektorov, z ktorých sme potom vytvárali podpriestory. IHDR algoritmus je oproti HBM viac dynamický a všeobecne použiteľný na viac typov dát. Jeho dynamickosť sa prejavuje najmä v automatickom získavaní hlavných črt dát, pričom v HBM musia byť črty

alebo metahypotézy špecifikované dopredu. Ďalšou výhodou IHDR je primárne určenie na vysokorozmerné dáta, teda ako vstup môžu byť obrázky ktoré sa dajú ľahko vizualizovať.

V nasledujúcich kapitolách bude nasledovať popis a postup implementácie algoritmu založenom na IHDR algoritme, popis správania pri rozličných scenároch, rozličných dátových sadách a podrobnejší popis jednotlivých jeho častí.

4 Implementácia

V nasledujúcej kapitole bude popísaný postup implementácie algoritmu. Počas implementácie sme vychádzali hlavne z teoretických znalostí o algoritme IHDR. Neskôr sme pridali rôzne modifikácie za účelom zefektívnenia algoritmu a preskúmania rôznych vlastností. Ďalej budú v tejto kapitole popísané nástroje použité na vývoj, použité metodiky, externé knižnice, problémy vzniknuté počas implementácie a rôzne možnosti vylepšenia a zefektívnenia algoritmu.

4.1 Návrh

Pri návrhu algoritmu sme vychádzali z teoretických znalostí z predchádzajúcich kapitol. Ako základ nám slúžil IHDR algoritmus, ktorý sme sa pokúsili do detailov reprodukovat' a neskôr vylepšovateľ, skúšať rôzne modifikácie a testovať algoritmus na rôznych dátových sadách. Zostali sme pri rovnakom pomenovaní algoritmu, teda IHDR. Nasleduje stručný popis jednotlivých častí projektu a objektivej štruktúry samotného algoritmu.

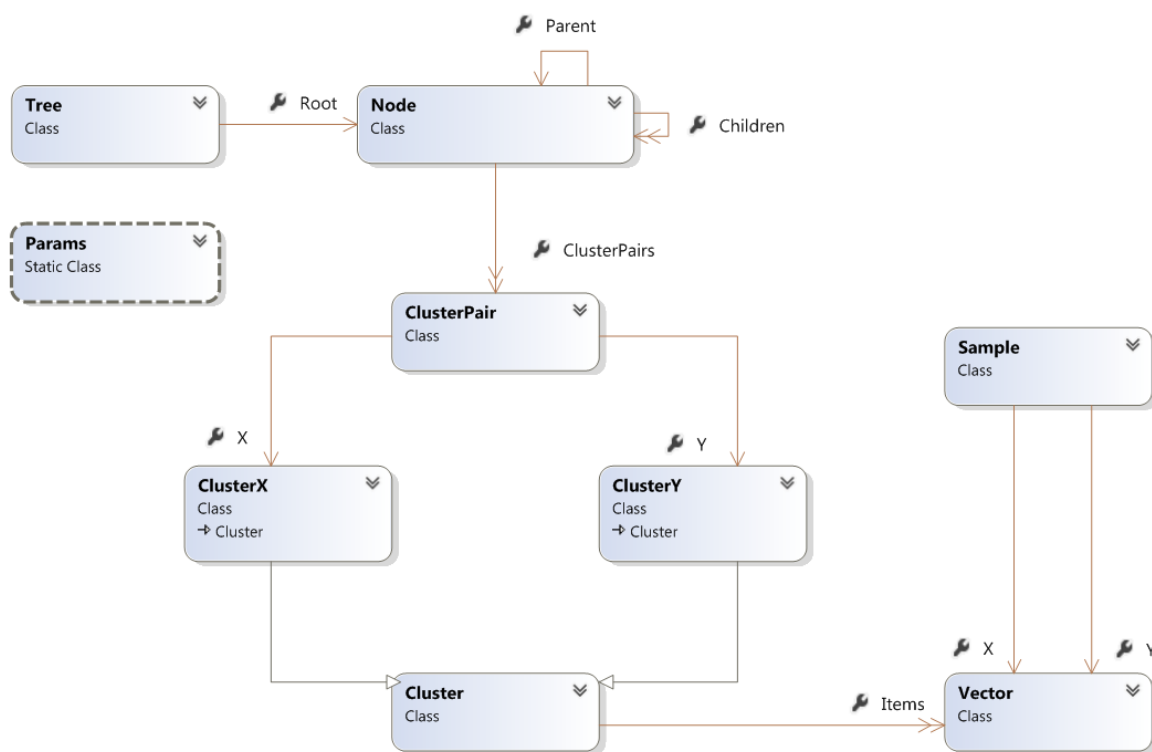
4.1.1 Projektová štruktúra

Samotný algoritmus sme vyvíjali ako knižnicu s možnosťou použitia v hocijakom inom kompatibilnom projekte. Nasleduje zoznam jednotlivých podprojektov a ich stručný popis.

- IHDRApplication – aplikácia s užívateľským rozhraním určená na ovládanie a testovanie algoritmu
- IHDRLib – knižnica obsahujúca celý algoritmus IHDR
- IHDRLibTest – knižnica obsahujúca testy jednotlivých metód a funkcií z IHDRLib
- SyntheticDataGenerator – knižnica na generovanie syntetických dát, určených na testovanie očakávaného správania celého algoritmu
- MNISTParserLib – knižnica určená na načítavanie dátovej sady MNIST
- Faces2ParserLib – knižnica určená na načítavanie dátovej sady „Výrazy tvárí“

4.1.2 Objektová štruktúra

Základná dátová štruktúra je strom, ktorého vrcholy predstavovali jednotlivé kategórie. Objektový návrh s jednotlivými objektmi a reláciami medzi nimi, ktorý je použitý v algoritme je na nasledujúcom obrázku 4.1. Znáznornené sú iba základné a dôležitejšie časti.



Obrázok 4.1 Diagram tried našej implementácie.

V každom vrchole (**Node**) máme množinu párov klastrov (relácia **ClusterPairs** na triedu **ClusterPair**). Jeden pár klastrov obsahuje x -klaster (relácia **X** na **ClusterX**), ktorý predstavoval dáta v X priestore a y -klaster (relácia **Y** na **ClusterY**) predstavujúci dáta v Y priestore. Klastre obsahovali všetky priradené príklady (relácia **Items** na **Vector**). Každý vrchol mohol obsahovať ďalších q vrcholov (relácia **Children**), teda synov. Vstupný príklad (**Sample**) pozostáva z X časti a Y časti (relácie **X** a **Y** na triedu **Vector**). Všetky nastavenia algoritmu sú uložené v parametroch (**Params**).

4.2 Použité algoritmy

Pri implementácii algoritmu sme použili aj ďalšie známe algoritmy. Boli nimi hlavne algoritmus Gram-Schmidtovho ortogonalizačného procesu spomínaný v predchádzajúcej kapitole a algoritmy určené na klasterizáciu k-means a k-means++.

4.2.1 GSO algoritmus

Pri rozvetvení vrcholov bolo nutné určiť podpriestor najviac diskriminačných čŕt. Jednotlivé klastre v priestore nám určujú smery čŕt dát. Tieto smery vyjadrujú rozptylové vektory od celkového priemeru k priemerom klastrov. Z týchto vektorov nasledovne vypočítame ortogonálnu bázu pre podpriestor najviac diskriminačných čŕt pomocou Gram-Schmidtovho ortogonalizačného procesu (GSO). Na vstupe GSO algoritmus dostane množinu n -dimenzionálnych vektorov $U = \{u_1, u_2, u_3, \dots, u_n\}$ a na výstupe dostaneme ortogonálnu bázu $V = \{v_1, v_2, v_3, \dots, v_n\}$. GSO algoritmus je iteratívny a jeho kroky sú nasledovné (rozpísané sú len prvé štyri kroky)

$$1. \quad v_1 = u_1 / (||u_1||)$$

$$2. \quad v_2 = u_2 - \text{proj}_{W_1} u_2 = u_2 - \frac{\langle u_2, v_1 \rangle}{||v_1||} v_1,$$

kde W_1 priestor určený bázovým vektorom v_1 a $\text{proj}_{W_1} u_2$ je ortogonálna projekcia u_2 na W_1

$$3. \quad v_3 = u_3 - \text{proj}_{W_2} u_3 = u_3 - \frac{\langle u_3, v_1 \rangle}{||v_1||} v_1 - \frac{\langle u_3, v_2 \rangle}{||v_2||} v_2,$$

kde W_2 priestor určený bázovými vektormi v_1 a v_2

$$4. \quad v_4 = u_4 - \text{proj}_{W_3} u_4 = u_4 - \frac{\langle u_4, v_1 \rangle}{||v_1||} v_1 - \frac{\langle u_4, v_2 \rangle}{||v_2||} v_2 - \frac{\langle u_4, v_3 \rangle}{||v_3||} v_3,$$

kde W_3 priestor určený bázovými vektormi v_1, v_2 a v_3 ...

GSO napísaný všeobecným pseudokódom

$$1. \quad v_1 = u_1 / (||u_1||)$$

2. Pre $i = 2, 3, \dots, n$, rob nasledovné:

$$a. \quad v_1' = u_i - \sum_{j=1}^{i-1} u_i^T v_j v_j$$

$$b. \quad a_i = v_1' / ||v_1'||$$

4.2.2 K-means algoritmus

V prípade rozvetvenia vrcholov je potrebné roz distribuovať jednotlivé páry klastrov do nových synov. Ak sme mali menší počet synov ako párov klastrov, musíme použiť algoritmus na vytváranie skupín klastrov. Klastre z jednotlivých skupín klastrov následne vkladáme do nových synov. Pre túto potrebu používame k-means algoritmus určený na klasterizáciu dát [17]. K-means používame taktiež pri modifikácii algoritmu, keď sme potrebovali reorganizovať klastre pri rozvetveniach, kvôli presnejšiemu určeniu podpriestoru najviac diskriminačných črt (4.3.3). Ako vstup do k-means algoritmu máme príklady patriace vrcholu. Príklady $X = \{x_1, \dots, x_n\}$ potrebujeme rozdeliť do q klastrov, čo je maximálny počet klastrov pre vrchol. Algoritmus je nasledovný:

1. Ľubovoľne zvolíme q príkladov z X ako iniciálne centrá klastrov $C = \{c_1, \dots, c_k\}$
2. Všetky príklady $x_i \in X$ priradíme k najbližším centrá a tak vytvoríme klastre C_i , ktorým patria príklady najbližšie k centrá c_i
3. Posunieme centrá klastrov vypočítaním priemeru z príkladov v C_i . Vzniknú tak nové centrá $c_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$
4. Opakuj kroky 2 a 3 pokiaľ nie sú priradenia príkladov k centrá stabilné, čiže po posunutí centier nevzniknú žiadne zmeny v priradeniach

Algoritmus k-means nie je však vhodný na všetky prípady rozmiestnenia príkladov v priestore. Je ovplyvnený iniciálnym zvolením centier a keďže sú centrá priradené náhodne, nemusí byť klasterizácia optimálna. Náhodný výber iniciálnych centier robí problémy aj pri testovaní algoritmu, pretože pri testovaní rovnakých konfigurácií s rovnakými dátami algoritmus dáva rôzne výsledky. Preto používame zoptimalizovanú verziu algoritmu k-means a to k-means++ [18].

4.2.3 K-means++ algoritmus

Tento algoritmus je takmer totožný s k-means algoritmom. Jediná zmena je voľba centier na začiatku. Iniciálne centrá zvolíme predvýpočtom nasledovne:

1. Zvolíme prvé iniciálne centrum c_1 ľubovoľne z príkladov $X = \{x_1, \dots, x_n\}$
2. Zvolíme ďalšie centrum $c_i = x' \in X$ s pravdepodobnosťou

$$P = \frac{D(x')^2}{\sum_{x \in X} D(x)^2}$$

kde $D(x)$ je vzdialenosť k najbližšiemu centru

3. Opakujeme kroky 1 a 2 kým nemáme q iniciálnych centier
4. Pokračujeme, ako v klasickom k-means algoritme

Potrebovali sme však stály algoritmus, bez žiadnych náhodných krokov. Dôvodom je hlavne lepšia možnosť testovania. Preto sme v prvom kroku volili nie ľubovoľné centrum, ale ako centrum sme zvolili prvý príklad z množiny X .

4.3 Časti algoritmu a modifikácie

Kostra algoritmu je podobná algoritmu IHDR. Keďže tento algoritmus je popísaný v predchádzajúcej kapitole, podrobnejšie popíšeme a vysvetlíme len niektoré časti algoritmu. Taktiež popíšeme iné možnosti implementácie jednotlivých častí.

4.3.1 Rozvetvenie vrchola

Jednou z najkritickejších častí algoritmu bolo rozvetvenie vrchola. Pri rozvetvení vrchola sa vykonávalo hlavne určenie podpriestoru najviac diskriminačných čŕt, transformácia štatistických vlastností vrcholov do tohto podpriestoru a rozdistribuovanie príkladov do nových synov vrchola. Pri neinkrementálnej verzii pribudlo ešte reorganizovanie klastrov pomocou k-means++. V nasledujúcom kóde je popísaná postupnosť akcií vykonaných pri rozvetvení vrchola


```

// test, či má vrchol dostatok príkladov na rozvetvenie
if (this.GetNSPP() > Params.bs)
{
    // rozvetvenie vrchola na q synov
    this.Swap();
    // výpočet váhovaného priemeru všetkých klastrov
    this.CountC();
    // výpočet lineárnej variety MDFS
    this.CountGSOManifold();
    // transformácia všetkých príkladov jednotlivých klastrov do MDFS
    this.CountMDFOfVectors();
    // výpočet priemerov v klastroch v MDFS
    this.CountMDFMeans();
    // výpočet kovariančných matíc v MDFS
    this.CountCovarianceMatricesMDF();
    // výpočet váhovaného priemeru kovariančných matíc v MDFS
    this.CountCovarianceMatrixMeanMDF();
}

```

4.3.2 Parametre vrcholov podľa hĺbky

V IHDR algoritme sa vytvárajú klastre s rozlíšením δ_x a δ_y . Tieto rozlíšenia vlastne určujú, akú najmenšiu vzdialenosť môžu mať medzi sebou klastre. Čiže čím vyššie rozlíšenie, tým máme menšiu hodnotu δ_x a δ_y . V IHDR sú hodnoty rozlíšení tie isté v každom vrchole. Nie je však výhodné mať tieto rozlíšenia všade rovnaké, pretože nám ide o zjemňovanie kategorizácie smerom k listom stromu. V koreni stromu máme hrubé kategórie vytvárané s nižším rozlíšením. Ak robíme rozvetvenie, potrebujeme nastaviť synom vyššie rozlíšenie a následné vytvorenie klastrov z príslušných príkladov vykonať podľa tohto vyššieho rozlíšenia. Preto zavedieme poklesy rozlíšení $\varepsilon_{\delta_x}, \varepsilon_{\delta_y} \in (0,1)$. δ_x a δ_y v závislosti od hĺbky vrchola l sa budú počítat nasledovne:

- $\delta_x(l) = \varepsilon_{\delta_x}^l \delta_x$, prípadne inkrementálne $\delta_x(l) = \varepsilon_{\delta_x} \delta_x(l-1)$, kde $l > 1$
- $\delta_y(l) = \varepsilon_{\delta_y}^l \delta_y$, prípadne inkrementálne $\delta_y(l) = \varepsilon_{\delta_y} \delta_y(l-1)$, kde $l > 1$

V každom vrchole máme ohraničený počet x -klastrov a y -klastrov parametrami b_x a b_y , z ktorých vyplýva aj dimenzia podpriestoru najviac diskriminačných črt. V koreni je preto v našom algoritme výhodnejšie mať viac klastrov, pretože dynamické vytváranie podpriestoru najviac diskriminačných črt pri nízkom počte klastrov nebude presné. Pri nízkom počte klastrov by sme pri prehľadávaní stromu nevedeli veľmi spoľahlivo povedať, do ktorej vetvy má prehľadávanie pokračovať. Keď bude vrchol vo väčšej hĺbke, už nepotrebuje podpriestor

najviac diskriminačných črt veľkej dimenzie, pretože príklady majú už veľmi málo rozdielov a stačí nám ich rozlišovať na základe menšieho počtu atribútov a taktiež rozdeľovať do menšieho počtu kategórií. Podobný princíp, ako pri rozlíšeníach použijeme aj pri dynamickom menení maximálneho počtu klastrov a zavedieme poklesy $\varepsilon_{b_x}, \varepsilon_{b_y} \in (0,1)$. Hodnota b_x a b_y musí byť zdola ohraničená hodnotou 2, pretože pri jednom klastri $q = 1$ nedokážeme vytvárať podpriestor najviac diskriminačných črt, keďže ho vytvárame pri GSO z $q - 1$ klastrov:

- $b_x(l) = \max\{\text{round}(\varepsilon_{b_x}^l b_x), 2\}$,
prípadne inkrementálne $b_x(l) = \max\{\text{round}(\varepsilon_{b_x} b_x(l-1)), 2\}$, kde $l > 1$
- $b_y(l) = \max\{\text{round}(\varepsilon_{b_y}^l b_y), 2\}$,
prípadne inkrementálne $b_y(l) = \max\{\text{round}(\varepsilon_{b_y} b_y(l-1)), 2\}$, kde $l > 1$

Zvyšovanie rozlíšení, a teda zmenšovanie parametrov δ_x a δ_y nám zaručí model vytvárajúci hrubé kategórie vo vrchných úrovniach stromu, ktoré sa s rastúcou hĺbkou v strome zjemňujú. Menenie počtu klastrov nám nebude vytvárať zbytočne veľa nových kategórií vo veľkých hĺbkach stromu, čo má vplyv aj na rýchlosť prehľadávania. Význam použitia dynamického menenia parametrov však veľmi závisí od charakteru a vlastností vstupných dát. Pri niektorých typoch dát by sme potrebovali skôr opačnú zmenu počtov klastrov. Napríklad v prípade, že ak by nám na rozdeľovanie veľmi konkrétnych kategórií nestačilo len zopár najviac diskriminačných črt.

4.3.3 Vytváranie klastrov

V každom vrchole môžeme vytvárať klastre rôznym spôsobom. Ako bolo spomínané, v IHDR sa vytvárajú klastre v dvoch priestoroch a to v X a Y priestore. Určenie klastrov, ktoré budú aktualizované pri pridaní príkladu vo vnútorných vrchoch, robíme vždy na základe Euklidovskej vzdialenosti y časti príkladu od y-klastrov. Pri vytváraní klastrov však máme na výber, v ktorom priestore, a tak isto akým spôsobom bude robená klasterizácia.

- x – klasterizácia - vytváranie klastrov podľa priestoru X . Nemáme žiadnu informáciu o tom do akej kategórie príklady patria. Pri istých dátach sa môže stať, že sa vytvárajú

klastre, ktoré potom zle určia podpriestor najviac diskriminačných črt. Lepšie pri vytváraní kategórií, ktoré nemusia mať veľmi konkrétny výstup.

- y – klasterizácia - vytváranie klastrov podľa priestoru Y . Klastre sa vytvárajú podľa výstupných dát z príkladov. Takto presnejšie určíme, ako má vyzeráť správny podpriestor najviac diskriminačných črt pre konkrétnu triedu. Vhodnejšie pre problém klasifikácie.

V našej implementácii sa nachádzajú oba typy klasterizácie a typ vyberáme podľa vstupných dát, ktoré chceme kategorizovať alebo klasifikovať. Pri klasifikácii vyberáme skôr druhý prístup, pretože chceme vytvárať klastre prevažne jednej triedy a potrebujeme také podpriestory v priestore X , ktoré presnejšie určujú diskriminačné črty jednotlivých tried. Tento prístup však nie je možný pri príkladoch s malou množinou typov výstupov. Napríklad ak máme výstup z množiny piatich vektorov predstavujúcich jednotlivé triedy, kategórie, nevytvoríme v Y priestore viac ako päť klastrov. To by nám zabráňovalo rozvetveniu stromu, lebo by sme nemali dostatok klastrov na vytvorenie nových vrcholov.

Ďalšie rozdelenie klasterizovania je podľa spôsobu vytvárania klastrov. Pri IHDR sú klastre vytvárané jednoduchým, ale nie veľmi presným spôsobom a to podľa rozlíšenia. Jednoducho povedané, ak máme príklad a jeho vzdialenosť od najbližšieho klastra je väčšia ako rozlíšenie, vytvoríme nový klaster. Táto metóda môže robiť značné problémy pri zvolení zlého rozlíšenia, prípadne pri zvolení zlého poklesu rozlíšenia. Pokiaľ nemáme dostatočnú informáciu o vstupných dátach, dobrá voľba rozlíšenia je takmer nemožná. Do istej miery však tieto problémy rieši amnézický priemer, pomocou ktorého sa staré, zle zaradené dáta ignorujú a centrá klastrov sa posúvajú do správnych lokácií. Veľmi veľkou výhodou tejto klasterizácie je, že nám umožňuje mať algoritmus inkrementálny. V algoritme sme ale zvolili aj druhú možnosť klasterizácie a to pomocou spomínaného algoritmu k -means++. Tento spôsob vytvára klastre presnejšie a tak získame aj lepšie výsledky. Problémom však je, že pri tomto spôsobe nie je algoritmus inkrementálny, pretože pri každom rozvetvení sa vytváranie stromu pozastaví kvôli reorganizovaniu klastrov vo vrchole. Tento problém zhoršuje aj skutočnosť, že k -means++ algoritmus je veľmi časovo náročný.

4.3.4 Klasifikácia a regresia

Pomocou nášho algoritmu dokážeme riešiť dva problémy a to regresiu a klasifikáciu. Pri regresii máme mnohorozmerný vstup aj výstup. Pre príklad si môžeme zobrať robota, ktorý má na vstupe obrázok prostredia, v ktorom sa nachádza a regresný strom mu určí vektor akcií, ktoré má vykonať. Pri klasifikácii máme však ako výstup iba označenie triedy alebo kategórie, do ktorej vstupný príklad patrí, čiže máme iba jednorozmerný výstup. Tento výstup však nijako nevyjadruje, ako sú jednotlivé triedy podobné. Mohol by nám teda vzniknúť problém určiť, ktorý klaster je najviac podobný novému príkladu, prípadne aké výstupy majú jednotlivé klastre, ak obsahujú príklady rôznych tried. Teoreticky by bolo možné určiť tieto údaje podľa percentuálneho zastúpenia jednotlivých tried v klastri, ale táto metóda by bola značne nepresná. V príklade klasifikácie rukou písaných čísel je napríklad triede “8” bližšie klaster, ktorý obsahuje 49 % príkladov triedy “8” a 51 % príkladov triedy “3”, ako klaster obsahujúci 51 % príkladov triedy “8” a 49 % príkladov triedy “1”, pretože triedy “3” a “8” sa vizuálne viac podobajú. Z tohto dôvodu potrebujeme problém klasifikácie previesť na problém regresie tak, že označenie triedy nahradíme viacrozmerným vektorom. Možnosti prevodu sú spomenuté v časti 3.3. V našom algoritme sme výstup príkladov pri klasifikácii nastavovali dvoma spôsobmi:

- Priemer všetkých vstupov tej istej triedy – v prípade, že poznáme všetky dáta dopredu
- Priemer všetkých doposiaľ videných vstupov tej istej triedy – v prípade postupného prichádzania príkladov

Pri prvej možnosti vzniká problém spomínaný pri klasterizácii a to ten, že máme iba toľko typov výstupov, koľko máme tried. To bude viesť k znemožneniu robiť klasterizáciu podľa Y priestoru. Problém sme v našom prípade riešili pridaním náhodného jemného šumu.

4.3.5 Klasifikovanie dát

Pomocou regresného stromu vytvoreného našim algoritmom, dokážeme vstupné príklady zaraďovať do kategórií v X priestore a následne im určiť virtuálny výstup pomocou priradeného klastra v Y priestore. Pomocou tohto výstupu potrebujeme určiť, do akej triedy daný príklad patrí, klasifikovať ho. Na základe klasifikácie môžeme následne overiť aj popísať

výsledky algoritmu. Nasleduje algoritmus prehľadávania v strome. Aktívny vrchol v algoritme označuje vrchol, v ktorom sa momentálne nachádzame:

1. Ako aktívny vrchol nastav koreň stromu
2. Ak aktívny vrchol nie je listový vrchol tak :
 - a. Získaj najbližší x -klastor pomocou SDNLL
 - b. Označ vrchol odpovedajúci najbližšiemu x -klastoru ako aktívny (je to ten vrchol, do ktorého bol klastor pridaný pri rozvetvení)
 - c. Opakuj krok 2.
3. Ak je aktívny vrchol listový tak :
 - a. Získaj najbližší x -klastor pomocou Euklidovskej vzdialenosti
 - b. Vráť ako výsledok priemer y -klastra, ktorý prislúcha tomuto x -klastoru

Príklady potom klasifikujeme jednoduchým spôsobom a to na základe Euklidovskej vzdialenosti medzi výstupom z prehľadávania a priemeru vstupných dát jednotlivých tried.

Možná modifikácia je pridanie šírky prehľadávania, čiže by sme v algoritme nehľadali pomocou SDNLL iba najbližší klastor ale w najbližších klastrov a prehľadávanie by pokračovalo do prislúchajúcich vrcholov. Toto prehľadávanie by mohlo priniesť presnejšie výsledky, pretože by nebolo také citlivé na voľbu do ktorej vetvy stromu sa pokračuje. Pri stromoch s veľkou hĺbkou by však bolo toto prehľadávanie časovo náročné.

4.4 Metodika vývoja

Algoritmus obsahuje značné množstvo matematických výpočtov, preto bola implementácia veľmi náchylná na chyby pri programovaní a taktiež na efektivitu jednotlivých častí. Predpokladali sme, že bude nutnosť kód často testovať, upravovať, vylepšovať a refaktorovať. Z týchto dôvodov sme zvolili ako metodiku vývoja takzvaný Test Driven Development (TDD) a počas implementácie sme sa snažili pokrývať kód testami kontrolujúcimi korektnosť jednotlivých metód a funkcií. Ako sa neskôr ukázalo, táto metodika nám pomohla vo veľa prípadoch nájsť chyby vzniknuté pri programovaní, modifikáciách algoritmu a refaktoringu. Na nasledujúcom obrázku 4.2 je znázornený zoznam testov pričom jeden test nebol úspešný.

Medium > 100 ms (3)	
✗ CountC_CountCCorrect	317 ms
✓ ClusterXCreate_CreateCorrectClusterX	169 ms
✓ UpdateMean_UpdatingIsCorrect	205 ms
Fast < 100 ms (26)	
✓ Add_AddToVectorCorrectVector	< 1 ms
✓ CountCovarianceMatrix_CovarianceMatrixIsCorrect	< 1 ms
✓ CountCovarianceMatrixMDF_CountCorrectCM	75 ms
✓ CountGSOManifold_ProvideCorrectCounting	30 ms
✓ CountLabelOfCluster_LabelIsCorrect	4 ms
✓ CountMDFMean_CountCorrectMean	61 ms
✓ CountMeanOfNode_CountCorrectMean	8 ms
✓ CountOutputsFromClassLabels_CountCorrectOutputs	4 ms
✓ Equals_VectorEqualsToVector	< 1 ms
✓ Equals_VectorNotEqualsToVector	< 1 ms

Obrázok 4.2 Zobrazenie časti testov.

4.5 Použité nástroje

Očakávali sme, že implementácia nebude triviálna a bude vyžadovať časté úpravy, niekedy aj väčších častí. Hlavne z tohto dôvodu sme zvolili programovací jazyk C# kvôli jednoduchšiemu objektovo orientovaného vývoju ako pri skriptovacích jazykoch a dobrej znalosti tohto jazyka. Ako vývojové prostredie sme použili Visual Studio. Výhodou tejto voľby je, že Visual Studio priamo obsahuje framework podporujúci metodiku TDD a v prípade potreby máme k dispozícii jednoduchú implementáciu užívateľského rozhrania. Na druhej strane jazyk C# nie je veľmi bežný pri implementáciách takýchto typov algoritmov a matematických výpočtov, najmä výpočtov z lineárnej algebry. Pokiaľ by sa nepodarilo nájsť vhodné knižnice na tento typ algoritmu, mohli by vzniknúť značné implementačné komplikácie a taktiež problémy s časovou efektivitou algoritmu. Pri vývoji sme vyskúšali dve externé knižnice, a to Math.Net Numerics a ILNumerics, z ktorých sme nakoniec zvolili ILNumerics.

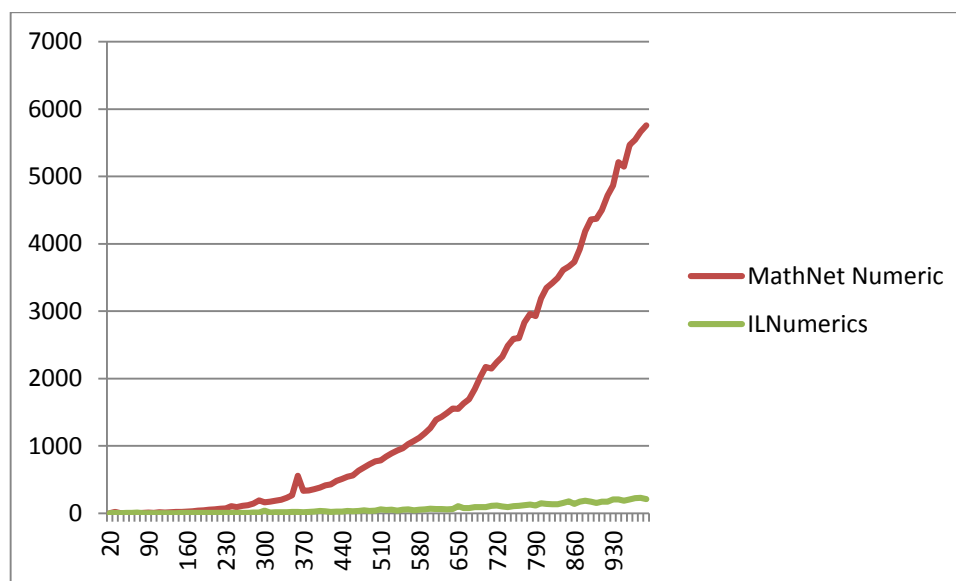
4.5.1 Math.Net Numerics

Knižnica Math.Net Numerics [19] je opensource knižnica určená pre vývojárov pod .NET platformou. Hlavným dôvodom použitia bola podpora výpočtov lineárnej algebry a všetkých potrebných maticových operácií ako násobenie, transponovanie, výpočet inverznej matice

a iných. Po implementácii časti algoritmu, sme zistili, že knižnica nevyhovuje našim potrebám. Obsahovala nedokončené časti, nebola efektívna pri práci s viacrozmernými maticami a vektormi, v niektorých prípadoch mala dokonca problém so zaokrúhľovaním. Z týchto dôvodov sme museli zvoliť inú knižnicu určenú pre platformu .NET a to ILNumerics.

4.5.2 ILNumerics

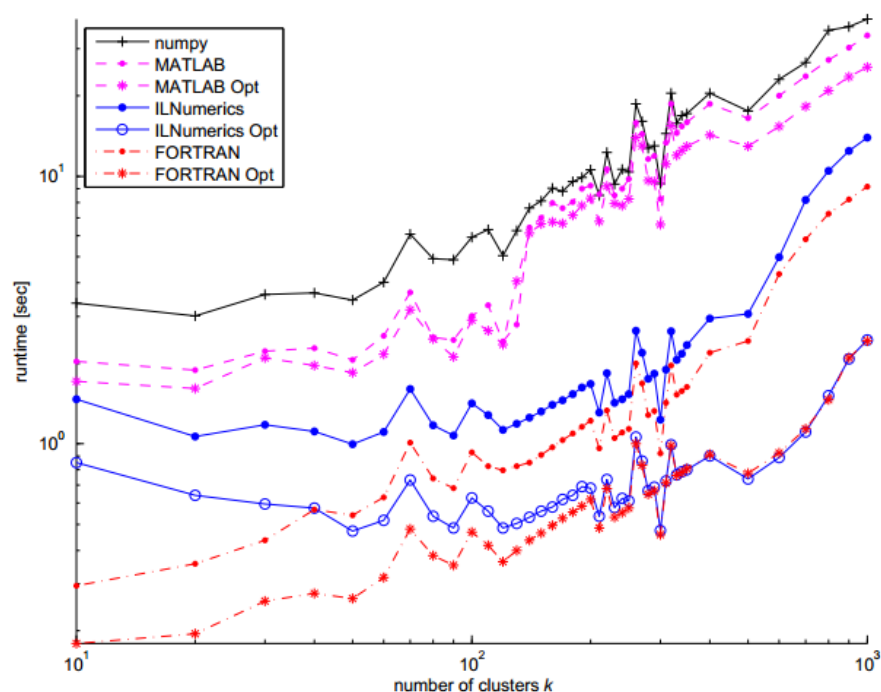
Ďalšou vyskúšanou knižnicou bola ILNumerics [20] [21]. Oproti MathNet Numerics je táto knižnica vo veľa ukazovateľoch lepšia. Hlavne jednoduchšia inicializácia matíc, korektnosť a rýchlosť výpočtov. Na nasledujúcom grafe 4.1 je znázornené porovnanie rýchlosti dvoch používaných knižníc pri násobení matíc.



Graf 4.1 Porovnanie rýchlostí *MathNet Numeric* a *ILNumerics* pri násobení matíc. Na osi *x* je dimenzia násobených matíc, na osi *y* čas násobenia v milisekundách.

Ako bolo spomenuté, vyvíjať podobné algoritmy v jazyku C# nie je veľmi bežné. Najčastejšie používané sú najmä jazyky ako Fortran, Matlab, Python, C alebo C++ a to hlavne kvôli rýchlosti výpočtov a syntaxi jazyka. Ďalšou možnosťou pri vývoji bolo použitie skriptovacieho jazyka F#, ktorý je primárne určený na matematické operácie pod .NET platformou. Pre jednoduchšiu implementáciu bola ako konečná voľba použitie knižnice ILNumerics, ktorá dokonca nezaostáva v rýchlosti výpočtov za bežnými jazykmi. Na grafe 4.2

je znázornené porovnanie časov algoritmu k-means pri rôznych jazykoch. Z grafu 4.2 vidno, že optimalizovaná verzia ILNumerics je takmer na úrovni Fortranu a je dokonca efektívnejšia ako Matlab.

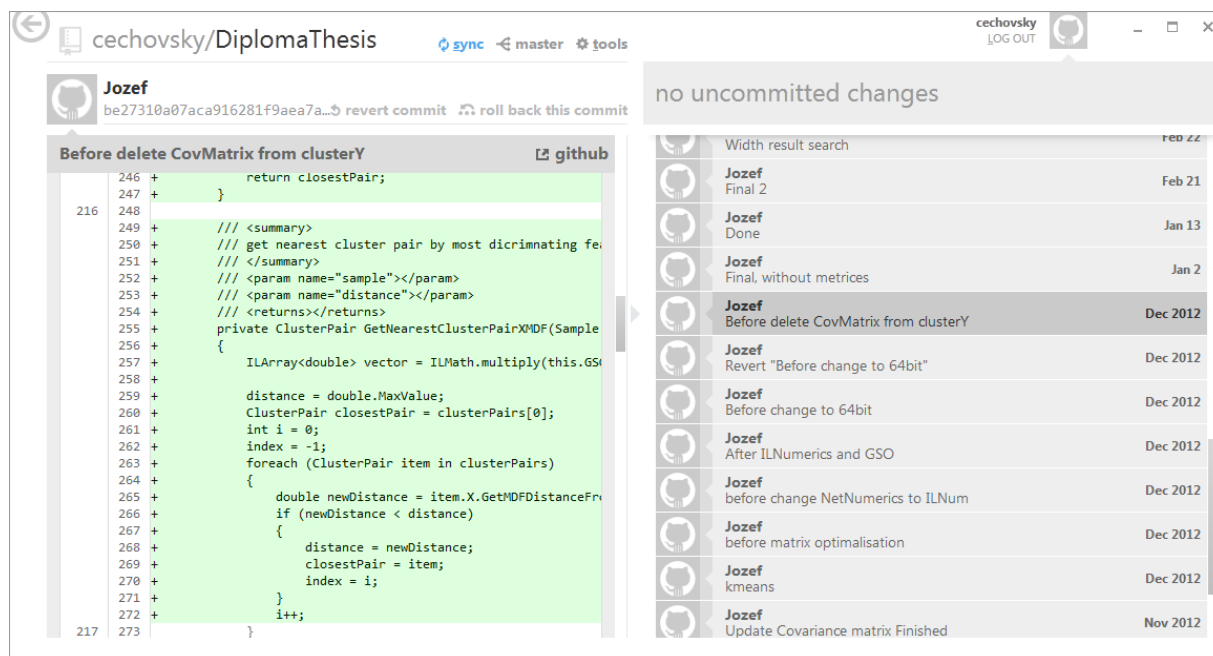


Graf 4.2 Porovnanie rýchlostí algoritmu k-means v jednotlivých jazykoch pri parametroch počet klastrov $m = 500$, počet príkladov $n = 2000$ [21].

4.5.3 Git a GitHub

Počas vývoja sme sa stretli s častými zmenami architektúry, použitých knižníc, typmi premenných a s väčšími modifikáciami algoritmu, kedy mohli vzniknúť rôzne nekonzistentnosti v kóde. Algoritmus sme vyvíjali zároveň na dvoch počítačoch, na ktorých boli vykonávané aj testy. Z týchto dôvodov sme použili distribuovaný systém riadenia revízií (sub-version system) Git [22]. Počas implementácie sme ho využívali na zaznamenávanie histórie kódu, zjednodušil nám vývoj na dvoch počítačoch zároveň a slúžil nám aj na časté zálohovanie súborov. Git repozitár sme publikovali prostredníctvom web služby GitHub, kde je dostupný celý kód algoritmu aj so všetkými revíziami. Url odkaz repozitáru je

<https://github.com/cechovsky/DiplomaThesis>. Na nasledujúcom obrázku 4.3 je zobrazené grafické rozhranie služby GitHub so zoznamom revízií.



Obrázok 4.3 Zobrazenie grafického rozhrania služby GitHub.

5 Experimentálne výsledky

Výsledný algoritmus obsahuje viac možností nastavení parametrov a konfigurácií. Pre získanie najlepších výsledkov sme vykonali sadu testov algoritmu, aby sme otestovali, či je jeho celkové správanie a správanie jednotlivých jeho častí korektné a taktiež, aby sme zistili vhodné nastavenie parametrov a zvolili vhodnú konfiguráciu. V nasledujúcej kapitole sú popísané a zobrazené rôzne testy algoritmu a ich výsledky. Na testovanie sme najskôr použili nami vygenerované dáta a ďalej voľne dostupné dátové sady používané pri metódach klasifikovaní alebo kategorizovaní dát.

5.1 Parametre algoritmu

Algoritmus obsahoval pomerne veľké množstvo parametrov, hlavne pri inkrementálnej verzii. V implementácii je ich možné nastaviť v statickej triede *Params*. Nasleduje ich zoznam a stručný popis:

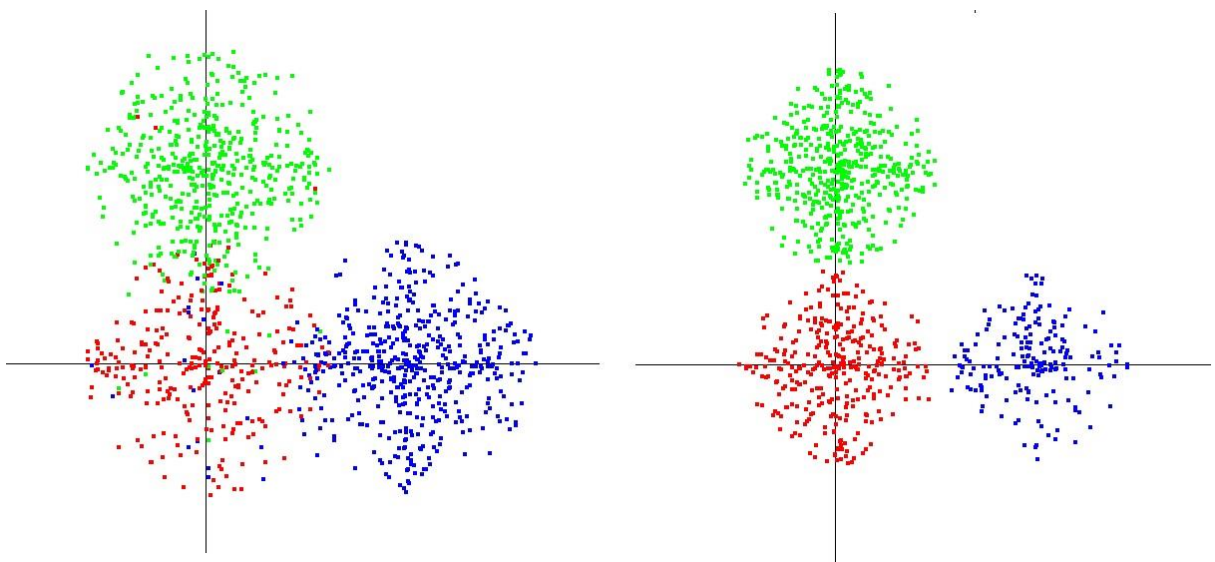
- q – počet synov vrchola
- blx – maximálny počet klastrov vo vrchole v priestore X
- bly – maximálny počet klastrov vo vrchole v priestore Y
- δX – rozlíšenie v X priestore
- δY – rozlíšenie v Y priestore
- bs – hraničná hodnota NSPP pri ktorej dochádza k rozvetveniu vrchola
- $t1, t2, c, m$ – parametre použité pri amnézických výpočtoch priemeru a kovariančnej matice. Použité pri inkrementálnej verzii
- l – hranica vzdialenosti vrchola od listov, pri ktorej sa ešte aktualizujú štatistické vlastnosti.
- *InputDataDimension* – veľkosť rozmeru vstupu
- *OutputDataDimension* – veľkosť rozmeru výstupu
- *Epochs* – počet epôch pri vytváraní stromu
- *DigitizationNoise* (DN) – šum dát. Použité, ak očakávame dáta pri ktorých môže byť chyba pri meraní. Napríklad chyby meraní pri získavaní dát.

- *ConfidenceValue* (CV) – parameter použitý pri výpočte parametra n_s pre horné ohraničenie váh kovariančných matíc pri SDNLL

Pri jednotlivých testoch budeme uvádzať aj hodnoty parametrov. Pri rôznych verziách algoritmu sú však niektoré parametre zbytočné, preto budeme uvádzať iba tie, ktoré majú pre výsledky algoritmu význam.

5.2 Syntetické dáta

Najskôr sme algoritmus overili na syntetických dátach, ktoré sme si vygenerovali sami. Vytvorili sme si tri triedy a náhodné dáta, ktoré k nim prislúchali a mali daný nejaký rozptyl. Z hľadiska klasifikácie sme očakávali dobré výsledky, keďže sme dáta vytvárali zámerné tak, aby klasifikácia prebehla úspešne. Tento test bol dôležitý pre overenie správnosti algoritmu a hlavne správnosti redukcie dimenzií. Vstupné dáta boli trojrozmerné a v algoritme sa dimenzia redukovala na dva rozmery. Vo vrcholoch sme teda mali dvojrozmerný podpriestor najviac diskriminačných črt. Klasifikáciu sme previedli na regresiu s dvojrozmerným výstupom pomocou náhodného vygenerovania súradníc v okolí nejakých zvolených bodov. Výstup regresie sme potom mohli zobrazit' jednoducho do 2D grafu. Na obrázku 5.1 sú znázornené dva grafy zobrazujúce úspešnosť klasifikácie.



Obrázok 5.1 Grafické zobrazenie klasifikácie podľa farieb.

Farba určuje triedu zistenú podľa Euklidovskej vzdialenosti výstupu regresie od priemeru výstupov konkrétnej triedy. Pri prvom grafe obrázku 5.1 bol vyšší rozptyl vstupných príkladov a všetky dáta neboli správne klasifikované. Ako nesprávne klasifikované môžeme vidieť tie príklady, ktoré sú zobrazené v rámci troch klastrov s inou farbou. Na druhom grafe obrázku 5.1 sú správne klasifikované všetky príklady.

5.3 Hierarchie výrazov tváří

Po otestovaní na nami vygenerovaných dátach sme testovali algoritmus na reálnych dátach. Pri prvom testovaní sme použili dátovú sadu výrazov tváří [23]. Táto dátová sada obsahuje pomerne málo, iba 84 obrázkov, na ktorých je zachytených 7 typov výrazov tváří. Obrázky tváří majú fixnú pozíciu očí. Na obrázku 5.2 je zobrazená časť dátovej sady.



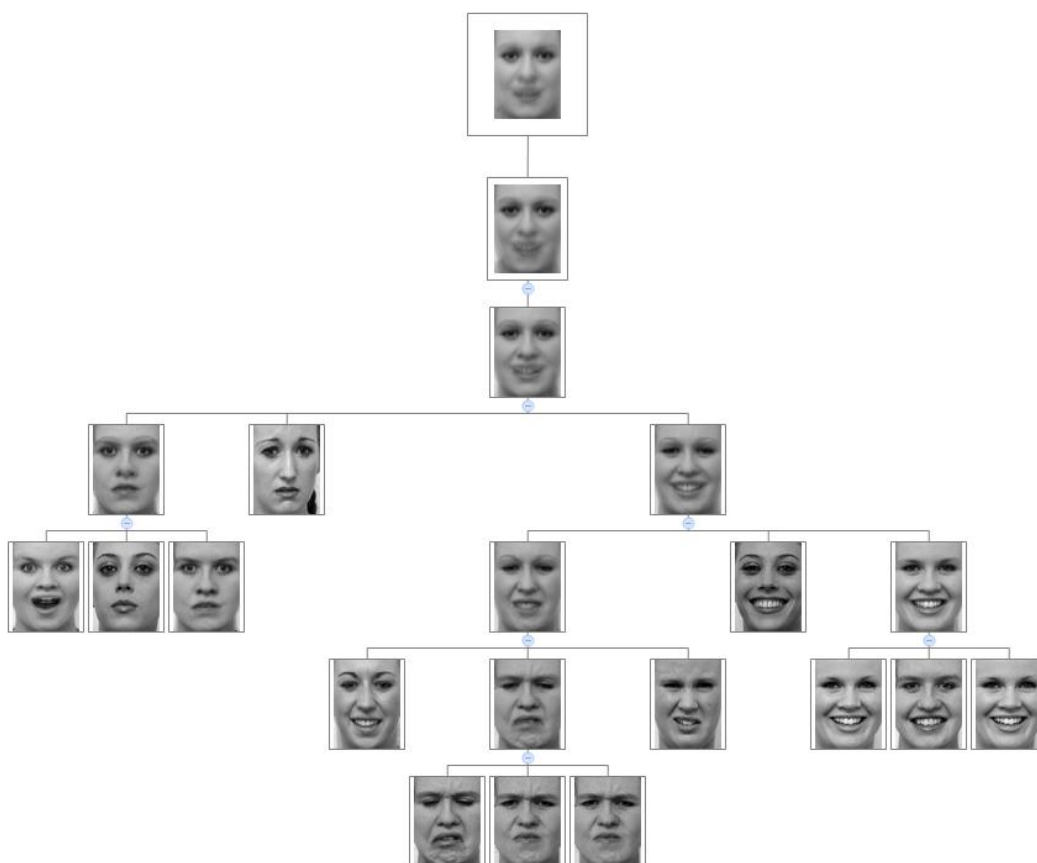
Obrázok 5.2 Ukážka príkladov z dátovej sady výrazov tváří [23].

Rozmer obrázkov sme si upravili na veľkosť 54 x 72 pixlov, čiže sme dostali ako vstup 3888 rozmerný vektor. Keďže bol počet príkladov malý, dáta sme zopakovali v našej upravenej dátovej sade viackrát s pridaným šumom, aby neboli vstupy úplne totožné. Ako výstupy sme zvolili priemery vstupov jednotlivých tried a pridali tiež náhodný jemný šum, pre možnosť klasterizácie aj v y-priestore. Na nasledujúcom obrázku 5.3 sú znázornené výstupy pre jednotlivé triedy.



Obrázok 5.3 Výstupy jednotlivých tried výrazov tvárí.

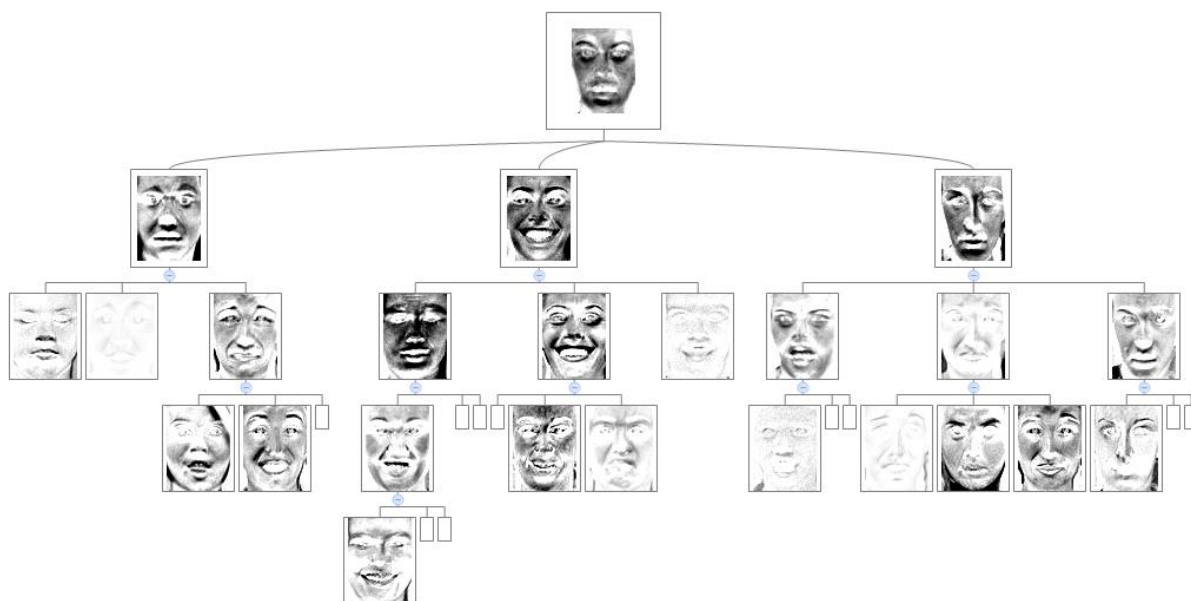
Ako je vidno, jednotlivé triedy tvárí sa líšia najmä ústami, očami a oblasťou okolo nosa. Náš algoritmus by mal dokázať tieto črty identifikovať a na základe nich rozlišovať medzi jednotlivými kategóriami, triedami. S použitím týchto dát sme vytvorili regresný strom. Parametre sme najskôr nastavili tak, aby nebol strom veľmi hlboký, kvôli potrebe jeho zobrazenia. Časť vytvoreného stromu je zobrazený v prílohe B. Pre lepšiu viditeľnosť a názornú ukážku je na nasledujúcom obrázku 5.4 zobrazený menší podstrom vytvorenej hierarchie.



Obrázok 5.4 Časť hierarchie vytvorenej algoritmom.

Na predchádzajúcom obrázku 5.4 dobre vidno, ako strom vytvára kategórie, ktoré sa s rastúcou hĺbkou zjemňujú. V koreni máme teda nejasnú, všeobecnú kategóriu a v listoch stromu sú už konkrétne kategórie. Niektoré dáta neboli pri vytváraní zaradené do správnej vetvy stromu a tak sa vytvorili niektoré vrcholy aj v nesprávnych miestach v hierarchii. Niektoré kategórie sú vytvorené dvakrát, pretože vstupné dáta sa opakujú. Síce majú pridaný jemný šum, ale ten nie je viditeľný.

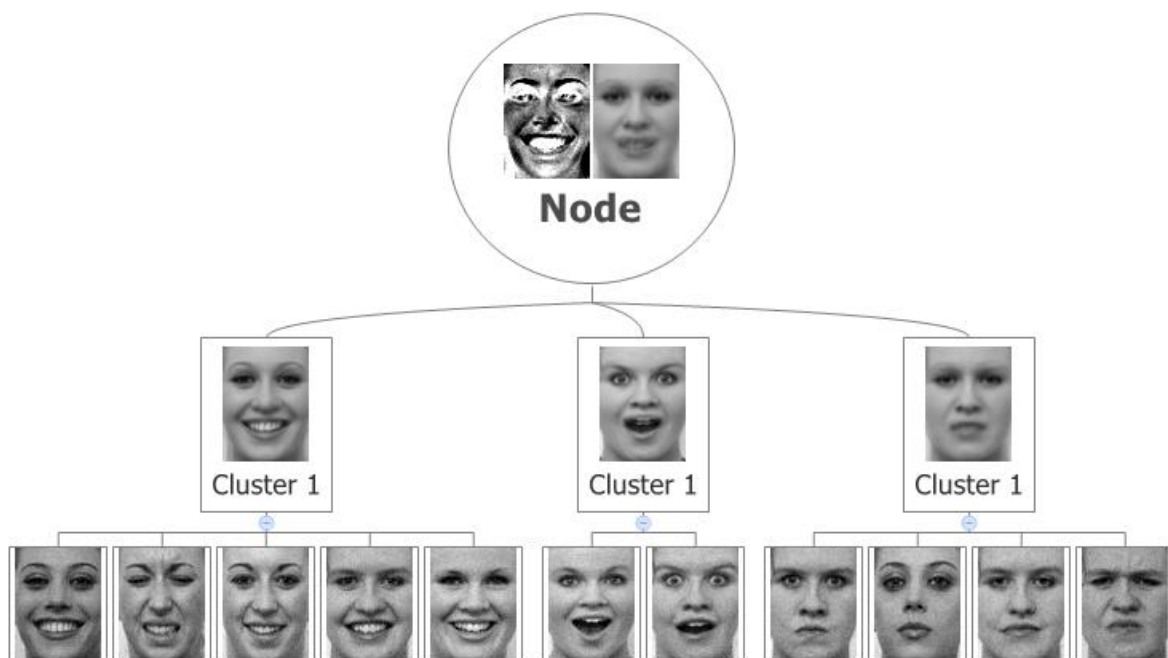
V každom vnútornom vrchole stromu máme iný podpriestor najviac diskriminačných črt, do ktorého sa všetky pridávané príklady premietajú. V tomto podpriestore sú taktiež potom počítané všetky štatistické vlastnosti. Na nasledujúcom obrázku 5.5 je zobrazený strom podpriestorov najviac diskriminačných črt. Pri tomto príklade počítame konkrétne trojrozmerné podpriestory.



Obrázok 5.5 Zobrazenie podpriestorov najviac diskriminačných črt v hierarchii.

Na obrázku je dobre vidno podľa ktorých črt sa najviac rozhoduje o príslušnosti do kategórie. Oblasti najviac diskriminačných črt sú bielej farby, naopak ignorované atribúty sú čierne. Aj na tomto zobrazení je možnosť sledovať zjemňovanie kategorizovania smerom k listom. V koreni stromu sú podpriestory viac tmavé, pretože je málo črt, ktoré sú na všetkých

obrázkoch rozdielne. Čím sme bližšie k listom, tým je kategorizovanie jemnejšie a presnejšie. Preto sú podpriestory bledšie, takmer biele. Tento jav zapríčiňuje fakt, že viac konkrétne obrázky vo väčšej hĺbke stromu sa medzi sebou rozlišujú vo veľa atribútoch. Grafické zobrazenie podpriestorov je vykonané prostredníctvom premietnutia príkladu so všetkými atribútmi čiernej farby a jeho následná inverzia (odčítanie od 255, čo je maximálna hodnota stupňa šedej). Na nasledujúcom obrázku 5.6 je ukážka jedného z vrcholov a jeho klastre s príkladmi a priermi, z ktorých sa vytvára podpriestor najviac diskriminačných črt.

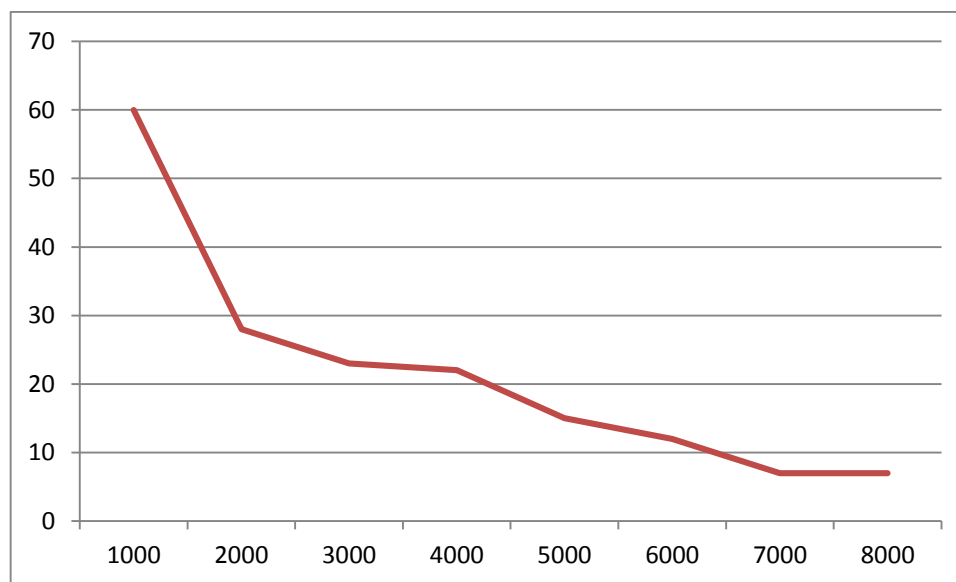


Obrázok 5.6 Zobrazenie jedného vrchola, jeho klastrov a priradených príkladov.

Zo zobrazeného podpriestoru je zrejmé, že príslušnosť do kategórií, teda do jednotlivých klastrov sa určuje hlavne na základe atribútov prislúchajúcich ústam a očiam. Podľa priemerov klastrov a jednotlivých príkladov priradených do klastrov je vidno, že príklady sú priradené do správnych kategórií.

Po overení správania algoritmu a vytvárania regresného stromu sme vykonali testy klasifikácie príkladov. Dátová sada však nebola vhodná na detailné otestovanie algoritmu, hlavne kvôli obmedzenému počtu dát v jednotlivých triedach. Z toho dôvodu sme nemali dostatočne určené

štatistické parametre jednotlivých klastrov. Aj napriek tomu algoritmus dokázal hierarchicky klasifikovať väčšinu dát. Na nasledujúcom grafe je znázornený graf závislosti percentuálnej chyby klasifikácie od počtu pridaných príkladov v strome.

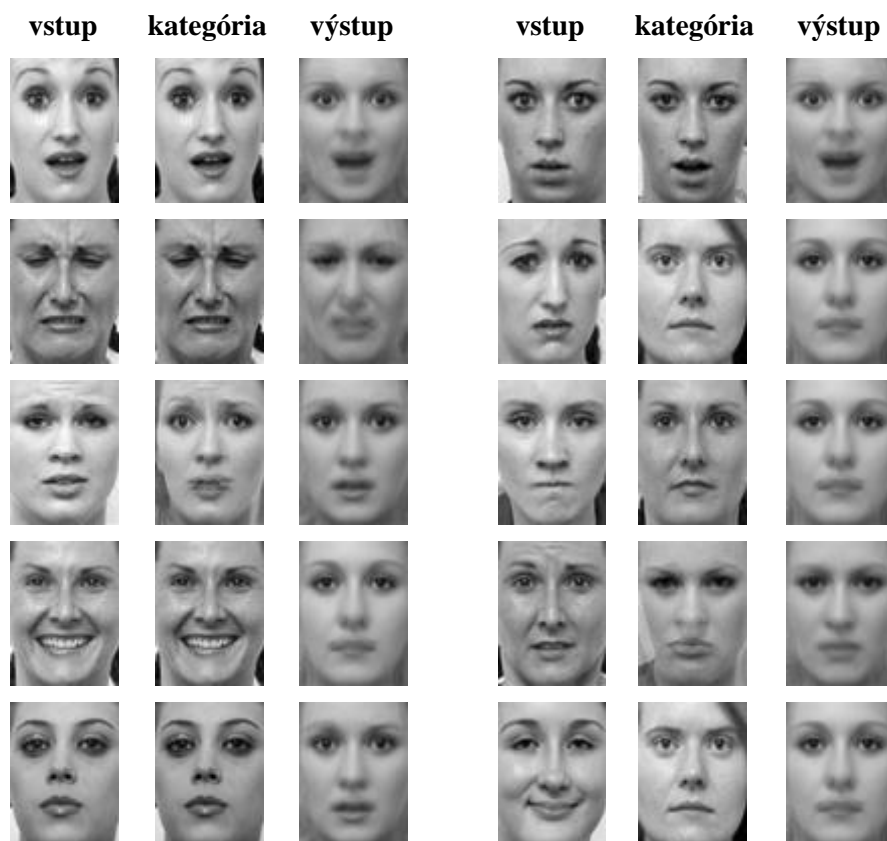


Graf 5.1 Vývoj percentuálnej chyby klasifikácie výrazov tváří pri zmene počtu príkladov.

Tieto výsledky sme dosiahli nasledovnou konfiguráciou algoritmu:

- Počet klastrov vo vrchole 20
- Stupeň rozvetvenia 5
- Vytváranie klastrov v priestore X
- Pri rozvetvení reorganizácia klastrov s k-means++

Na nasledujúcom obrázku 5.8. sú znázornené výstupy pre jednotlivé vstupy. Na ľavej strane sú ukážky správne klasifikovaných a na pravej nesprávne klasifikovaných príkladov. Kategóriu určuje priemer x -klastra do ktorého algoritmus zaradil vstupný príklad. Výstup je prislúchajúci y -klastor k tomuto klastru. Správne klasifikovanie zistíme porovnaním výstupu určeného algoritmom s výstupmi jednotlivých tried. Triedu určíme pomocou najbližšieho výstupu z tried.



Obrázok 5.7 Zobrazenie priradení vstupov do kategórií a príslušné výstupy.

Z predchádzajúcich dát vidno, že príklady boli zaradené do klastrov, kategórií, ktorých priemer je takmer totožný so vstupom. Je to z dôvodu, že tie isté dáta sme pridávali do stromu viackrát a vytvárali sa klastre s takmer totožnými príkladmi. Ďalej preto, že sme nemali dátovú sadu rozdelenú na testovaciu a trénovaciu sadu. Pri rozdelení dát na testovacie a trénovacie príklady algoritmus dával veľmi nepresné výsledky. Dôvodom bola veľká variabilita atribútov príkladov pri veľmi malom počte príkladov. Podrobnejšie testovanie algoritmu prevedieme na dátovej sade MNIST.

5.4 Klasifikácia dát

5.4.1 Dátová sada MNIST

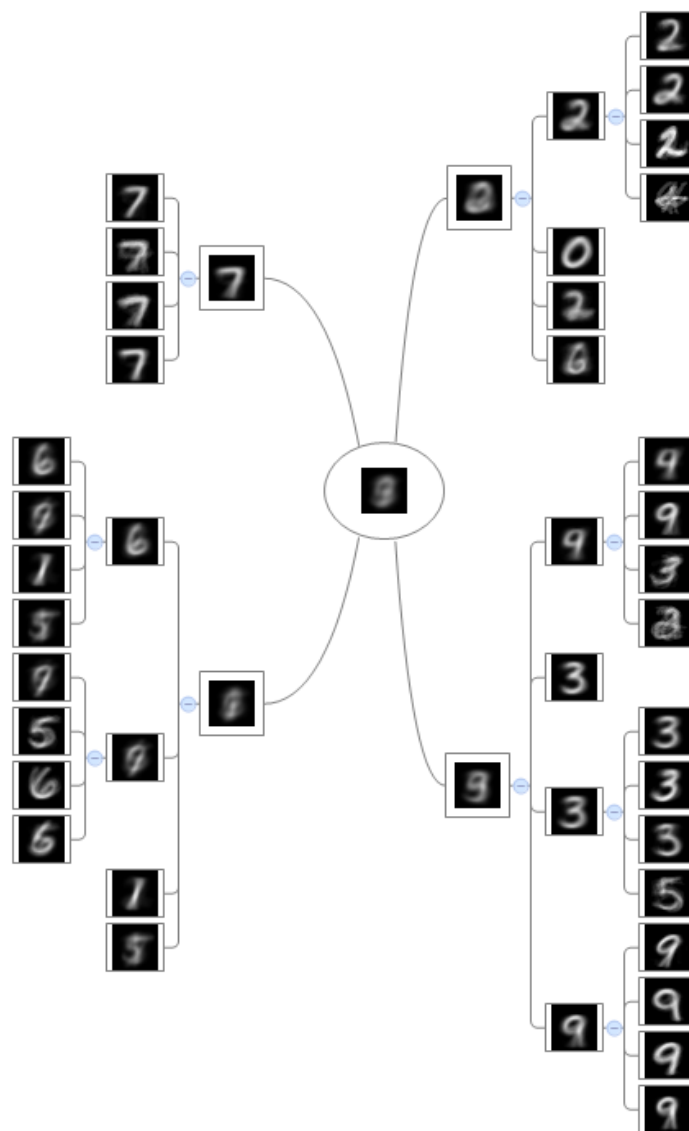
Druhou dátovou sadou, na ktorej sme testovali algoritmus bola MNIST [24]. MNIST je dátová sada pozostávajúca z obrázkov rukou písaných čísel, ktorá sa bežne používa na testovanie

rôznych techník klasifikácie dát. Obsahuje 60000 príkladov v trénovacej a 10000 príkladov v testovacej sade. Príklady sú obrázky rozmeru 28 x 28 a veľkosť čísel je normalizovaná. Jednotlivé pixle vyjadrujú stupne šedej, čiže hodnoty 1 až 255. Na nasledovnom obrázku 5.9 je názorná ukážka vybraných príkladov z dátovej sady.



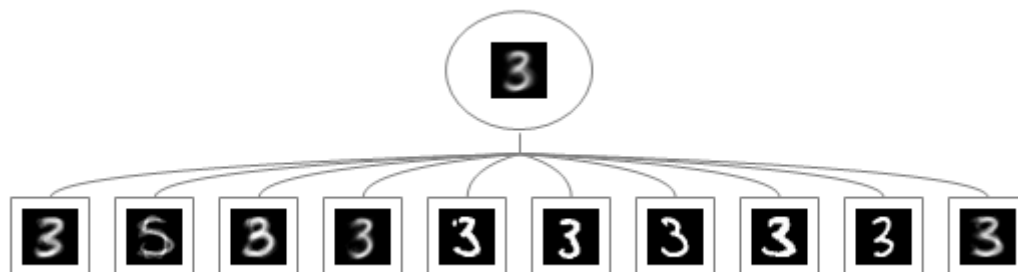
Obrázok 5.8 Ukážka príkladov z dátovej sady MNIST [24].

Táto dátová sada je oveľa vhodnejšia na otestovanie algoritmu ako predchádzajúca, pretože obsahuje veľký počet dát. Z tohto dôvodu môžeme vytvárať veľké stromy a hlavne jednotlivé klastre budú mať dostatočne presne určené štatistické vlastnosti. Na obrázku 5.10 je graficky znázornený strom podobne ako pri predchádzajúcej dátovej sade. Jednotlivé vrcholy na obrázku obsahujú priemery vrcholov alebo klastrov. Niektoré vrcholy sú takzvané hluché. Sú to tie, ktoré sa dostali do nesprávnej časti stromu a už nikdy sa k nim nové dáta nedostanú. Z toho dôvodu sa nebudú ďalej rozvetvovať.



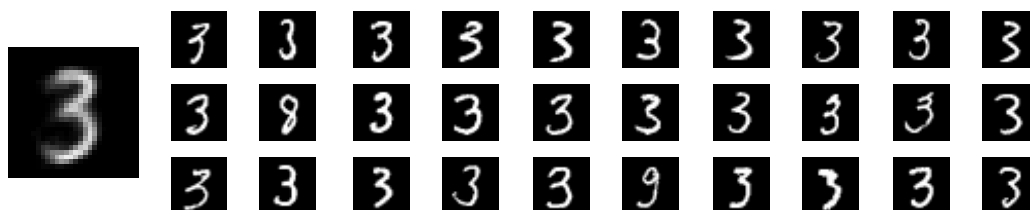
Obrázok 5.9 Zobrazenie vytvoreného stromu kategórií príkladov MNIST dátovej sady.

Stromy vytvárané algoritmom dosahujú oveľa väčšie rozmery ako zobrazený strom. Na obrázku 5.10 je zobrazená len veľmi malá časť stromu blízko koreňa. Preto sú jednotlivé kategórie ešte veľmi nejasné. Na nasledujúcom obrázku 5.11 je znázornený vrchol a jeho klastre vo väčšej hĺbke. Presnejšie na pozícii s cestou v strome *root-node1-node2-node3-node1-node4*.



Obrázok 5.10 Zobrazenie viac konkrétnej kategórie a priemerov klastrov.

Vrchol na obrázku 5.11 obsahuje prevažne klastre obsahujúce trojky, a jeden hluchý klastor, v ktorom sú päťky. Na nasledujúcich obrázkoch 5.12 a 5.13 sú znázornené konkrétne príklady priradené jednému klastru. V prvom prípade je listový klastor, v druhom klastor vo vrchole blízko koreňa, čiže väčšina dát je priradených podľa SDNLL.



Obrázok 5.11 Zobrazenie priemeru listového klastra a podmnožiny jemu priradených príkladov.



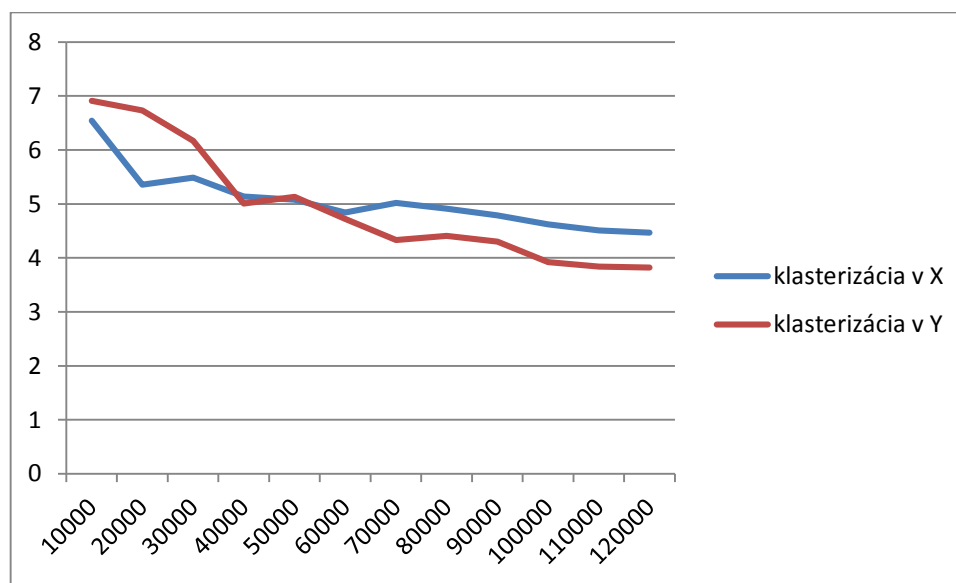
Obrázok 5.12 Zobrazenie priemeru klastra blízko koreňa a jemu priradených príkladov.

Podobne ako pri predchádzajúcej dátovej sade zobrazíme podpriestory najviac diskriminačných čŕt. Na nasledujúcom obrázku S5.14 je zobrazený podstrom s podpriestormi najviac diskriminačných čŕt. Prázdne vrcholy sú listové, teda ešte nemajú určený tento podpriestor. Z týchto zobrazených podpriestorov vyplýva, že pri kategorizovaní a klasifikovaní dát sú čierne časti úplne ignorované. Kategorizovanie je založené teda len na bielych častiach, čo sú najviac diskriminačné črty.

5.4.2.1 Vplyv klasterizácie na klasifikáciu

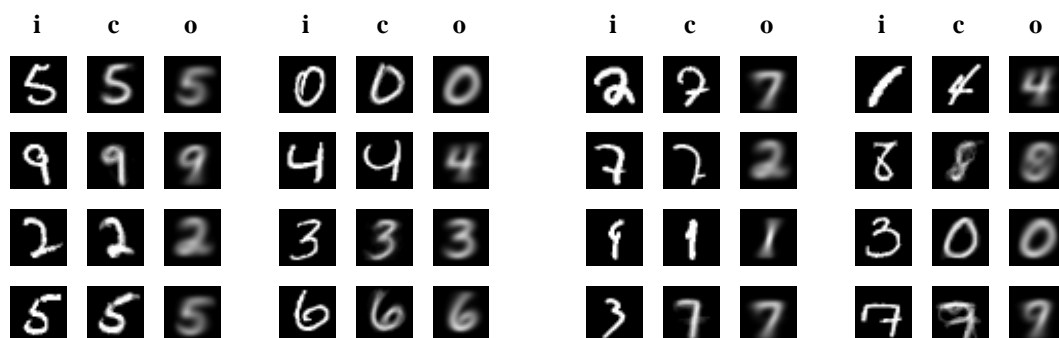
V časti 4.3.3 sme popisovali možnosti klasterizácie a ich použitie. Na nasledujúcom grafe 5.2 je zobrazený vývoj chyby klasifikácie pri dvoch typoch klasterizácie. Tieto výsledky sme získali na neinkrementálnej verzii a nasledovnej konfigurácii parametrov :

q	20	blx	40	DN	1
bs	1	l	10	CV	0.1



Graf 5.2 Vývoj chyby klasifikácie pri dvoch typoch klasterizácie.

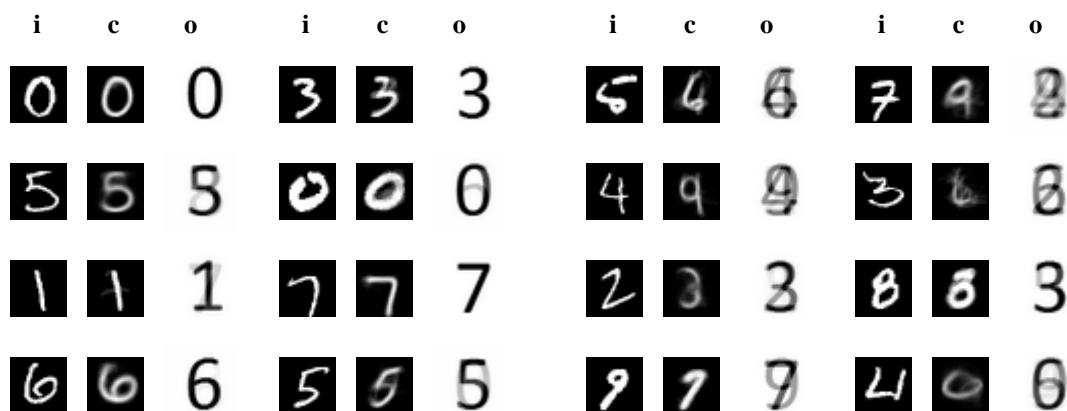
Vyššia chyba pri klasterizácii v *Y* priestore na začiatku vytvárania stromu je spôsobená výstupom príkladov, pretože ten bol počítaný ako priemer doposiaľ videných príkladov jednotlivých tried. Keď boli neskôr výstupy príkladov dostatočne určené, chyba klasifikácie bola nižšia ako pri *X* klasterizácii. Nasleduje obrázok 5.16 zobrazujúci dobre a zle klasifikované dáta. V prvých dvoch častiach sú správne klasifikované dáta, v ďalších dvoch nesprávne klasifikované.



Obrázok 5.14 Ukážka správneho a nesprávneho klasifikovania príkladov dátovej sady MNIST. (i = vstup, c = kategória, o = výstup).

5.4.2.2 MNIST a vytvorené výstupy

Pri predchádzajúcej klasifikácii nám ako výstupy slúžili priemery všetkých príkladov alebo len všetkých doposiaľ videných príkladov tej istej triedy. Tieto priemery síce jednoznačne určujú o akú triedu sa jedná, ale ako vidno z výstupov na obrázku 5.16, nie sú tieto priemery veľmi ostré. Pre ďalšie testovanie sme si vyrobili vlastné výstupy pre jednotlivé triedy a to obrázky číslíc odpovedajúcich triede. Následne sme vykonali testy klasifikácie s použitím týchto ostrejších výstupov. Veľkosť výstupov sme zvolili 40 x 40, takže sme zároveň otestovali fungovanie algoritmu pri dvoch rozdielnych rozmeroch X a Y priestoru. Pri nami vytvorenom výstupe sme dosahovali lepšie výsledky klasifikácie. Na obrázku 5.17 sú zobrazené príklady správneho a nesprávneho klasifikovania podobne ako v predchádzajúcom prípade.



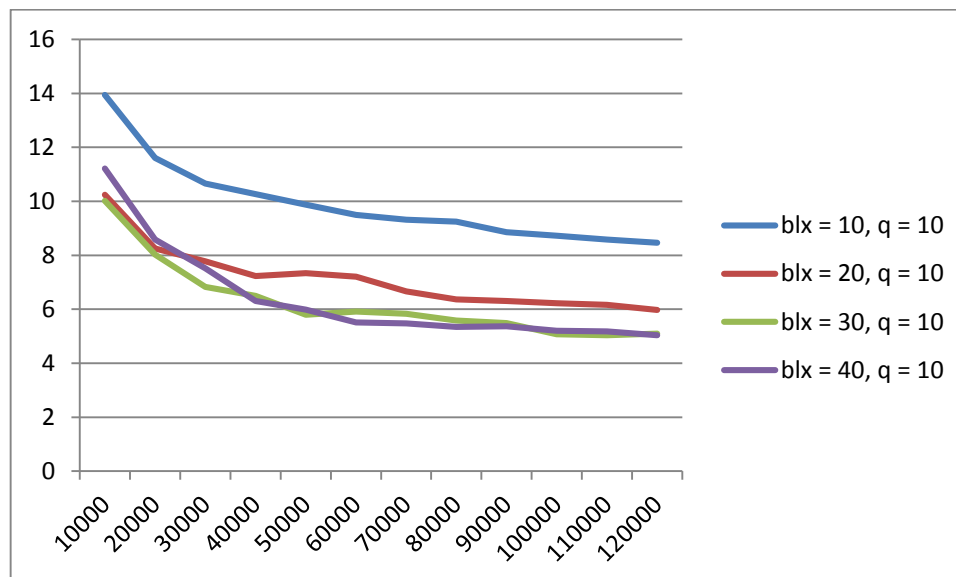
Obrázok 5.15 Ukážka správneho a nesprávneho klasifikovania príkladov dátovej sady MNIST s pridanými výstupmi. (i = vstup, c = kategória, o = výstup).

5.5 Správanie algoritmu a jeho častí

Algoritmus obsahuje viacero parametrov a konfigurácií. V nasledujúcej časti je popis testov jednotlivých konfigurácií a častí algoritmu, ktoré ovplyvňujú výsledky.

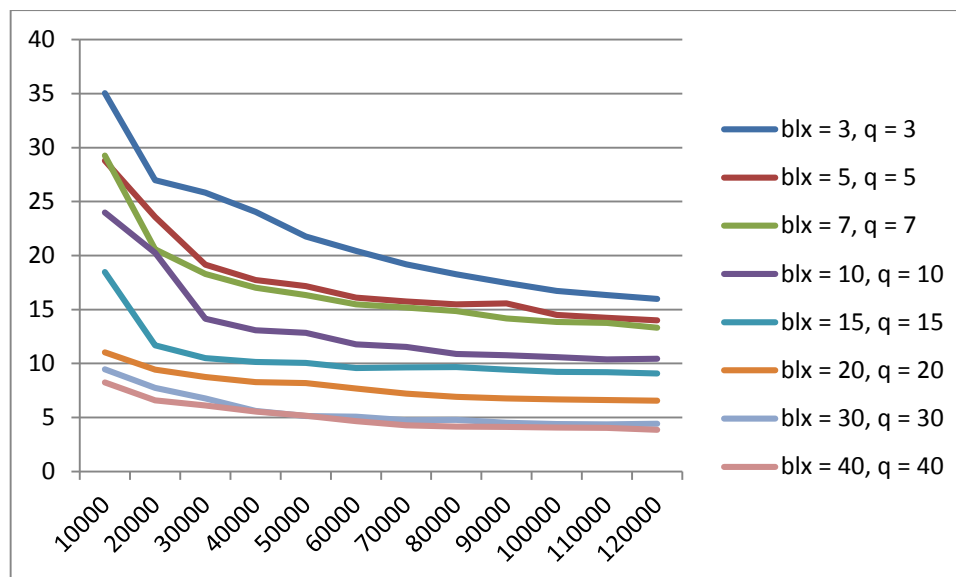
5.5.1 Počet klastrov a stupeň rozvetvenia

Parametre, ktoré najviac určujú vlastnosti vytvoreného stromu sú počet klastrov a počet nových vrcholov pri rozvetvení. Tieto parametre do veľkej miery ovplyvňujú aj následnú klasifikáciu. Na grafe 5.3 sú zobrazené chyby pri klasifikácii pri rôznych konfiguráciách. Počet synov q pri rozvetvení sme ponechali rovnaký, menili sme len počet klastrov blx .



Graf 5.3 Zobrazenie chyby klasifikácie pri rôznych konfiguráciách počtu klastrov a stupňa rozvetvenia.

S rastúcim počtom klastrov bola chyba menšia. Zmenšovanie chyby sa zastavilo pri počte klastrov 40. Pri ďalšom testovaní sme zvolil rovnaké hodnoty pre q a blx , čiže pri rozvetvení vrchola bol do každého nového vrchola pridaný práve jeden klaster. Na grafe 5.4 sú znázornené výsledky.



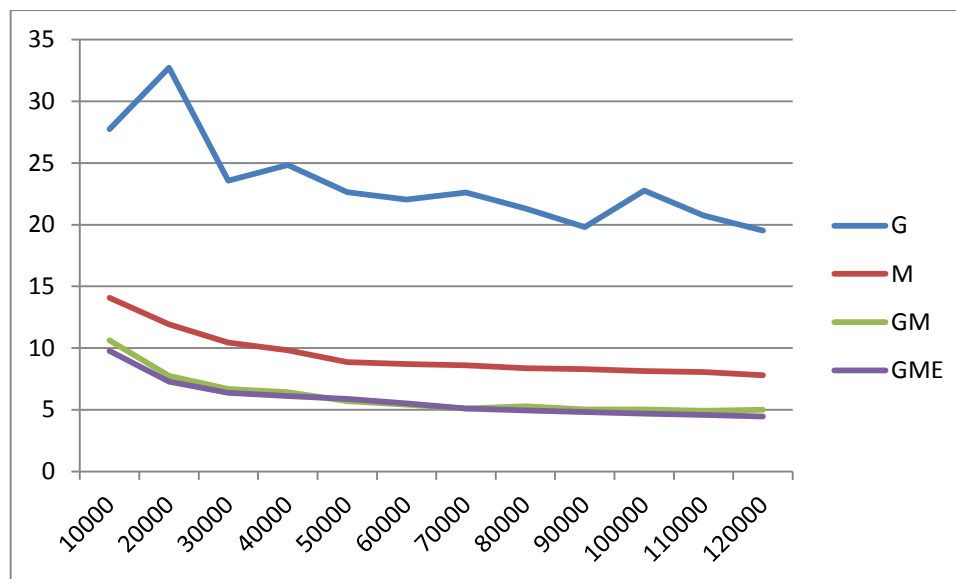
Graf 5.4 Zobrazenie chyby klasifikácie pri rôznych konfiguráciách počtu klastrov a stupňa rozvetvenia

Zaujímavé sú výsledky pri prvej konfigurácii v grafe. V tejto konfigurácii sa o príslušnosti do kategórie rozhoduje iba v dvojrozmernom podpriestore najviac diskriminačných črt. Pri poslednej konfigurácii sme dosiahli najlepší výsledok a to chybu iba 3.85 %. Pri vyššom počte klastrov a počte nových vrcholoch pri rozvetvení sme už dostávali horšie výsledky.

5.5.2 Porovnanie metrík

V časti 3.4.6 bolo popísané, že v algoritme sa používajú tri typy metrík. Vplyv použitia jednotlivých metrík na výsledky klasifikácie sú znázornené na nasledujúcom grafe 5.5. Jednotlivé metriky sú označené písmenami

- G – Gausovská NLL
- M – Mahalanobisova NLL
- E – Euklidovská NLL



Graf 5.5 Zobrazenie chyby klasifikácie pri rôznom použití metrík
(G – Gausovská NLL, M – Mahalanobisova NLL, E – Euklidovská NLL).

Zo zobrazeného grafu 5.5 vyplýva, že pri použití všetkých troch metrík dostávame najlepšie výsledky klasifikácie. Samotná Gausovská NLL dáva veľmi zlé výsledky, pretože strom obsahuje veľa vrcholov s malým množstvom príkladov. Samotná Mahalanobisova NLL je na tom oveľa lepšie, ale pri použití v kombinácii s Gausovskou dáva viditeľne presnejšie výsledky. Pridaním Euklidovskej NLL získame mierne zlepšenie, ktoré indikuje lepšie priradenia príkladov pri veľmi malom počte príkladov na začiatku vytvárania, čo ovplyvní neskoršie výsledky. Pri voľbe váh metrík je veľmi dôležité nastavenie parametra *ConfidenceValue*. V našom prípade sme tento parameter mali nastavený na hodnotu 0,1.

5.6 Porovnanie výsledkov

Na dátovej sade MNIST bolo vyskúšaných mnoho metód na klasifikovanie dát. Z existujúcich metód sme pre porovnanie s našim algoritmom vybrali tie viac známe. V našom algoritme sme najlepší výsledok klasifikácie dosiahli pri neinkrementálnej verzii s reorganizovaním klastrov a to 3.85 %. Pri inkrementálnej verzii to bol o niečo horší výsledok a to 4.18 %. Tieto výsledky sú porovnateľné s výsledkami iných metód. Existuje však množstvo vylepšení a modifikácií týchto metód, ktoré dávajú pri MNISTe výsledky pod 1%. V nasledujúcej tabuľke je pre porovnanie prehľad jednotlivých metód a ich percentuálna chyba pri klasifikácii [24].

Metóda	Percentuálna chyba
náš algoritmus (s reorganizáciou)	3.85 %
náš algoritmus	4.18 %
1-vrstvová neurónová sieť	12.0 %
K-najbližších susedov	5.0 %
40 PCA + kvadratický klasifikátor	3.3 %
SVM, Gaussian Kernel	1.4 %
2-vrstvová neurónová sieť, 1000 skrytých neurónov	4.5 %

6 Záver

V diplomovej práci sme popísali teoretické znalosti dvoch biologicky motivovaných výpočtových modelov určených na hierarchickú kategorizáciu dát. Prvý z nich bol hierarchický Bayesovský model založený na Bayesovskej inferencii a metahypotézach, ktoré máme dané dopredu a poznáme ich distribúcie pravdepodobností. Ďalším modelom bola inkrementálna hierarchická diskriminačná regresia inšpirovaná vývojom asociatívnej mozgovej kôry. V tomto modeli vytvárame stromovú hierarchiu kategórií na základe najviac diskriminačných čŕt, získavaných postupne pri rozvetveniach stromu. V jednotlivých vrcholoch hierarchie, ktoré predstavujú kategórie, vykonávame klasterizáciu v dvoch priestoroch, pričom klastre v jednom priestore vravia o kategórií a v druhom priestore o virtuálnom výstupe pre kategóriu. Pomocou tejto metódy dokážeme vytvárať regresné stromy, na ktorých môžeme vykonávať regresiu, klasifikáciu alebo kategorizáciu objektov.

Hlavným prínosom našej práce bola implementácia algoritmu založeného na IHDR, jeho praktické otestovanie a preskúmanie jeho vlastností a rôznych konfigurácií. Po overení korektnosti algoritmu na nami vygenerovaných dátach sme ďalej testovali algoritmus na štandardnej dátovej sade číslíc MNIST [24] a na dátovej sade fotografií výrazov tvárí [23]. Hierarchie vytvorené algoritmom sme vizualizovali a vykonali na nich klasifikáciu dát. Výsledky klasifikácie ukazujú, že nami naimplementovaný algoritmus dosiahol výsledky porovnateľné s bežnými známymi metódami. Koncept, ktorému sme sa v práci venovali je otvorený ďalším rozšíreniam, ako napríklad pridanie prvkov spomínaného hierarchického Bayesovského modelu [2] alebo sémantiky rozlišovacích kritérií [25].

7 Prílohy

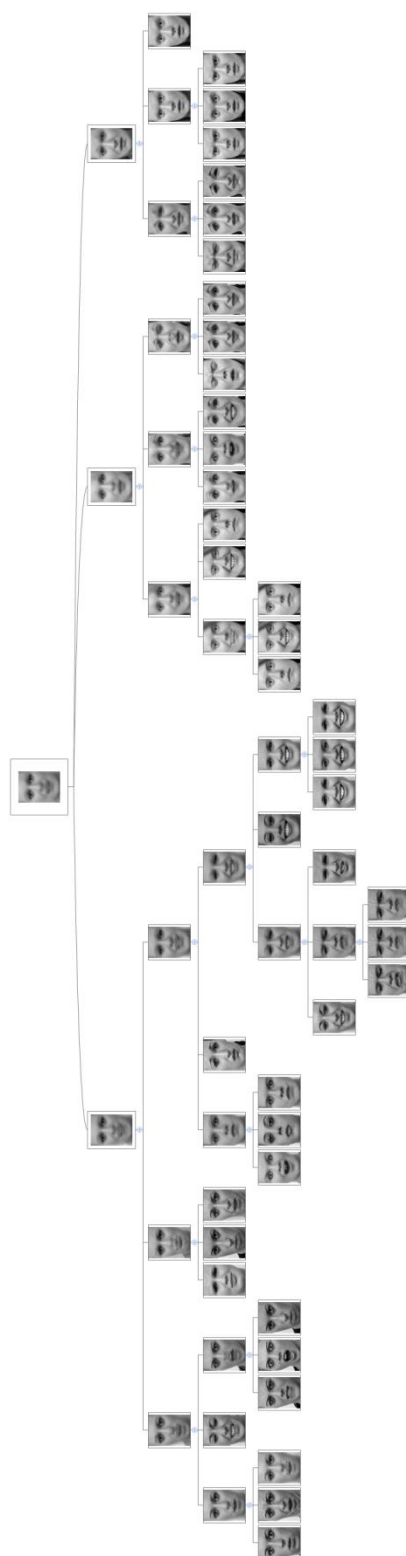
Príloha A – CD

Na priloženom CD sa v jednotlivých priečkach nachádzajú:

- DataSets – dátové sady použité pri testovaní algoritmu
- IHDRapplication – celá naimplementovaná aplikácia algoritmu. Návod na spustenie je popísaný v textovom súbore readme.txt
- IHDRresults – výstupy algoritmu, súborová hierarchia stromu, výsledky správne a nesprávne klasifikovaných dát
- Thesis – PDF verzia diplomovej práce

Príloha B – Zobrazenie hierarchie

Zobrazenie časti hierarchie výrazov tvárí.



8 Referencie

- [1] A. Perfors, J. Tenenbaum a T. Griffiths, „A tutorial introduction to Bayesian models of cognitive development.,“ *Cognition*, 2011.
- [2] K. Charles, P. Amy a T. B. Joshua, „Learning overhypotheses with hierarchical Bayesian models,“ *Developmental Science*, pp. 307-321, 2007.
- [3] W. J a H. WS., „Incremental hierarchical discriminant regression.,“ *IEEE Trans Neural Netw*, pp. 397-415, 2007.
- [4] E. M. Markman, *Categorization And Naming In Children*, MIT Press, 1989.
- [5] F. Xu a J. B. Tenenbaum, „Sensitivity to sampling in Bayesian word learning,“ *Developmental Science*, p. 288–297, 2007.
- [6] J. B. Tenenbaum a T. L. Griffiths, „Generalization, similarity and Bayesian inference,“ *Behavioral and brain sciences*, pp. 629 - 640, 2001.
- [7] T. H. Heibeck a E. M. Markman, „Word Learning in Children: An Examination of Fast Mapping,“ *Child Development*, pp. 1021-1034, 1987.
- [8] N. Chomsky, *Rules and Representations*, Oxford : Basil Blackwell, 1980, p. 299.
- [9] L. K. Samuelson, „Statistical Regularities in Vocabulary Guide Language Acquisition,“ *Developmental Psychology* 38, pp. 1016-1037, 2002.
- [10] J. Piaget, *The Psychology Of The Child*, New York: Basic Books, 1969.
- [11] J. Mandler, „Conceptual categorization,“ *Early category and concept development*, pp. 103 - 131, 2003.
- [12] N. Goodman, *Fact, fiction and forecast*, Harvard : Harvard University Press, 1983.
- [13] A. Perfors a J. Tenenbaum, „Learning to learn categories,“ *Proceedings of the 31st Annual Conference of the Cognitive Science Society*, pp. 136 - 141.
- [14] W.-S. Hwang a J. Weng, „Hierarchical Discriminant Regression,“ *Pattern Analysis and Machine Intelligence*, pp. 1277 - 1293, 2000.
- [15] H. Yu a J. Yang, „A direct LDA algorithm for high-dimensional data - with application to

- face recognition," *Pattern Recognition* 34, pp. 2067-2070, 2001.
- [16] M. Hazewinkel, *Encyclopaedia of Mathematics*, 1994.
- [17] D. MacKay, *Information Theory, Inference, and Learning Algorithms*, Cambridge : Cambridge University Press, 2003.
- [18] D. Arthur a S. Vassilvitskii, „K-means++: the advantages of careful seeding," *SODA '07 Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 1027-1035, 2007.
- [19] C. Ruegg, „Math.NET Numerics," [Online]. Available: <http://mathnetnumerics.codeplex.com/>.
- [20] H. Kutschbach, „ILNumerics," [Online]. Available: <http://ilnumerics.net/>.
- [21] H. Kutschbach, „High Performance Computing With .NET," TU-Berlin, Berlin, 2012.
- [22] „Git - Build software better, together," [Online]. Available: <https://github.com/>.
- [23] P. Hancock, „Psychological Image Collection at Stirling (PICS)," Psychology School of Natural Sciences, 2008. [Online]. Available: <http://pics.stir.ac.uk/>.
- [24] Y. LeCun a C. Cortes, „The MNIST database of handwritten digits," Courant Institute NYU, Google Labs New York, [Online]. Available: <http://yann.lecun.com/exdb/mnist/>.
- [25] M. Takáč, „Autonomous construction of ecologically and socially relevant semantics," *Cognitive Systems Research* 9 (2008), p. 293–311, 2007.
- [26] L. Breiman, J. Friedman, R. Olshen a C. Stone, *Classification and regression trees.*, Wadsworth Books, 1984.
- [27] D. L. Swets a J. Weng, „Hierarchical discriminant analysis for image retrieval," *Pattern Analysis and Machine Intelligence*, pp. 386 - 401, 1999.
- [28] S. K. Murthy, S. Kasif a S. Salzberg, „A System for Induction of Oblique Decision Trees," *Journal of Artificial Intelligence Research* 2, pp. 1-32, 1994.
- [29] O. A. Abbas, „Comparisons between data clustering algorithms," *The international Arab journal of information technology* vol. 3, 2007.
- [30] T. Finch, „Incremental calculation of weighted mean and variance," University of Cambridge Computing Service, Cambridge , 2009.

- [31] W.-S. Hwang, „Incremental hierarchical discriminating regression for indoor,“ Department of Computer Science and Engineering, Michigan State University, 2001.
- [32] E. Cantu-Paz a C. Kamath, „Inducing Oblique Decision Trees with Evolutionary Algorithms,“ Lawrence Livermore National Laboratory, Livermore, 2003.
- [33] P. Cunningham, „Dimension Reduction,“ *Technical Report UCD-CSI*, 2007.
- [34] P. E. Utgoff a C. E. Brodley, „An incremental method for finding multivariate splits for decision trees,“ Department of Computer and Information Science, University of Massachusetts, 1990.