FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY
UNIVERZITY KOMENSKÉHO

ÚSTAV INFORMATIKY

Konceptuálna reprezentácia, konekcionizmus a jazyk

Diplomová práca

Vypracoval: Čestmír Hýbl
Školiteľ: Prof. Ing. Vladimír Kvasnička, DrSc.

BRATISLAVA 2001

Čestne vyhlasujem, že predkladanú prácu som vypracoval samostatne, len s použitím literatúry uvedenej v zozname.

.....

Bratislava, máj 2001.

Abstrakt

V práci sa zaoberáme potrebou a vlastnosťami konceptuálnej reprezentácie, ktorú popri symbolickej a konekcionistickej navrhuje Peter Gärdenfors ako ďalšiu úroveň reprezentácie informácie v inteligentných systémoch. Venujeme sa sémantike jazyka, geometrickým aspektom konceptuálnych priestorov a popisujeme praktickú aplikáciu paradigmy konceptuálnej reprezentácie v oblasti počítačového videnia. V experimentálnej časti demonštrujeme schopnosť emergencie spoločného slovníka nad konceptuálnou reprezentáciou v agentovom systéme, kde každý agent disponuje jednoduchou rekurentnou sieťou, kódujúcou a dekódujúcou symbolické výrazy.

Obsah

Kapitola 1

Úvod

Reprezentácia informácií v mysliach inteligentných agentov, či už živých alebo umelých, je kľúčovou témou pre umelú inteligenciu a kognitívnu vedu špeciálne. Reprezentácia úzko súvisí s myslením a jazykom a je nevyhnutná na vykonávanie zložitejších kognitívnych úloh. Každý zrejme podvedome očakáva, že ňou disponuje väčšina systémov, o ktorých povieme, že sú inteligentné. Alebo skôr rozmýšľajúce? Tu narážame na istý spor, pretože položiť reprezentáciu ako nutnú podmienku inteligentnosti systému je prisilné. Aj subdeliberatívne (reaktívne) agenty môžu v mnohých situáciách generovať inteligentné správanie, a pritom už z definície reprezentáciou nedisponujú. Nevraviac potom už o kolóniách reaktívnych agentov. Asi by sa dalo ukázať, že ľudské telo je tiež takouto kolóniou, a pritom o jeho inteligencii málokto pochybuje. Na druhej strane, zväzovať reprezentáciu s „rozmýšľaním“ sa možno zdá byť prisilné tiež. Môže byť „rozmýšľanie“ nutná podmienka pre reprezentáciu? Pokiaľ pod rozmýšľaním budeme chápať úplne ľubovoľný druh inferencie, aj ten najtriviálnejší, nemusí to byť zase až také absurdné.

Vlastných definícií reprezentácie možno nájsť u rozličných autorov mnoho. Niektorí chápu už rozdiel medzi „slabou“ reaktívnosťou, ktorá agentovi umožňuje pri voľbe akcie zohľadniť vnútorný stav (a naopak, s akciou môže byť spojená zmena vnútorného stavu) a čistou reaktívnosťou, keď agent volí akcie výhradne na základe podnetov z prostredia. Celkovo však prevláda názor, že pojem reprezentácie si vyžaduje silnejšie predpoklady. Určite súvisí aspoň s vlastnosťami ako (1) nezávislosť od konkrétnych senzorických vstupov - schopnosť asociovať jednotlivé druhy senzorických vnemov (a samozrejme schopnosť na niečo takúto asociovanosť využiť). Silnejšou vlastnosťou je (2) nezávislosť od aktuálnej prítomnosti senzorického vnemu vôbec. Ďalšou relevantnou vlastnosťou je (3) schopnosť očakávať, že nastane určitý jav v prostredí (od najjednoduchších až po predpovedanie akcií iného agenta s potenciálnou potrebou vnorenej reprezentácie).

Peter Gärdenfors, ktorého teórii konceptuálnej reprezentácie a pojmu konceptuálneho priestoru sa v našej práci venujeme, podľa nášho názoru prináša prinajmenšom zaujímavé námety na uvažovanie o povahe reprezentácie a spôsoboch realizácie reprezentácie vo vhodných kognitívnych architektúrach.

Kapitola 2

Reprezentácia

V umelej inteligencii (do)dnes prevládajú dva prístupy k modelovaniu reprezentácie[1] - symbolický a konekcionistický. Popíšeme si teda v skratke tieto dva prístupy a ich vzájomný vzťah.

2.1 Symbolická reprezentácia

Táto paradigma tvorí pozadie „klasickej“ umelej inteligencie. Je založená na predpoklade, že základné stavebné jednotky reprezentácie sú symboly. Z nich sa syntaktickým zreťazovaním podľa určitých pravidiel vytvárajú zložitejšie výrazy. Myslenie sa potom vníma ako výpočet realizovateľný na Turingovom stroji. Praktické aplikácie tejto paradigmy vyústili do samostatnej oblasti - expertných systémov. Dôležité vlastnosti symbolickej paradigmy okrem iného sú:

- výpočet je symbolický (a nie analógový)
- symboly, relácie a operácie sú *systematicky interpretovateľné* (Fodor, Pylyshyn). Možno im teda systematicky priradiť význam (sémantika).
- výpočet je nezávislý na implementácii

Zhrnieme si ďalej najvýznamnejšie obmedzenia symbolickej paradigmy, dôležité z kontextu našej práce. Sú nimi:

- *pôvod a evolúcia predikátov a symbolov.* V prvorádovej logike sa predikáty chápu ako dané. Otázkou je, či postačuje schopnosť „definovať“ nové predikáty len pomocou starých, alebo je pre inteligentného agenta nevyhnutné zavádzať už „počas života“ úplne nové predikáty, prípadne netriviálne modifikovať význam existujúcich. Nejasný potom zostáva aj evolučný pôvod predikátov a symbolov v populácii agentov. Z evolučného hľadiska sa dá očakávať, že lingvistické symboly sa nejakou formou vyvinuli zo signálov jednoduchšej formy komunikácie (a súčasne zrejme aj reprezentácie ...), pričom tieto signály samozrejme mali svoj jednoznačný význam.

[1] Úmyselne nehovoríme o modelovaní inteligencie. Inteligencia je totiž ťažko uchopiteľný pojem a zdá sa, že je skôr o správaní ako o „myslení“. Je tu teda skôr vhodné delenie z aspektu multiagentových systémov.

- *slabé oddelenie domén informácií.* V čistom jazyku prvorádovej logiky je každý predikát aplikovateľný na ľubovoľný term (resp. kartézsky súčin termov). Existujú síce *viacsortové* logiky, zavádzajúce formálne rozdelenie univerza na domény rôznych sort, toto rozdelenie je však formálne definovateľné aj prostriedkami prvorádovej logiky.

- *frame problem.* Dá sa neformálne sformulovať ako problém počítania „čo zostalo nezmenené“ po vykonaní nejakej operácie. Úzko súvisí s oddelením domén informácií a s výpočtovou zložitosťou.

- *vypočítateľnosť a zložitosť.* Potreba zaoberať sa nimi je daná tým, že sa symbolická paradigma opiera o pojem algoritmu (symbolického výpočtu). Nanešťastie, pokiaľ sa neobmedzíme na veľmi jednoduchý jazyk (bez operátora negácie, Hornove klauzy, ...), budeme narážať na NP-úplné, prípadne ešte ťažšie problémy [19].

- *revízie.* Pripustenie nemonotónneho usudzovania[2] so sebou prináša potrebu revízií bázy znalostí. Realizácia revízie môže byť v plne symbolickom systéme z výpočtového hľadiska dosť netrivialna.

2.2 Konekcionizmus

Konekcionistická paradigma ponúka predstavu, že myslenie je vytvárané dynamikou aktivácií neurónov. Neurón môžeme všeobecne vnímať ako plastickú výpočtovú jednotku transformujúcu n vstupov na jeden výstup. Sieť zloženú zo vzájomne poprepájaných neurónov možno potom chápať ako plastické vektorové zobrazenie. Táto neformálna definícia vyznieva možno trochu zjednodušujúco pre neurovedca, zaoberajúceho sa biologickými neurónmi a sieťami, aj tam však podľa nás obstojí . Zložky vektorov môže tvoriť napr. frekvencia impulzov akčného potenciálu (spike) a v prípade potreby aj zložitejšie kvantity, popisujúce povahu prenášaného signálu.

2.3 Vzťah konekcionizmu a symbolickej paradigmy

Fodor a Pylyshyn v známej publikácii „Connectionism and Cognitive Architecture: A Critical Analysis“ (1988) sformulovali kritiku konekcionizmu, ktorú možno zhrnúť nasledovne:

> Konekcionistické modely (1) buď nedokážu vysvetliť elementárne kognitívne zákonitosti mysle ako *produktívnosť, systematickosť a inferenčnú koherentnosť*, alebo (2) sú len implementáciou klasických symbolických architektúr.

Reakcie konekcionistov sa dajú rozdeliť na názory:

[2]Monotónnosť sa tu chápe vzhľadom na operátor dedukcie Cn ($A \subseteq B \Rightarrow Cn(A) \subseteq Cn(B)$). Prakticky to znamená, že monotónny usudzovateľ zachováva všetky už odvodené dôsledky aj napriek pridávaniu nových predpokladov do bázy znalostí. Robí to aj za cenu toho, že po vložení predpokladu, ktorý je v rozpore s už odvodeným dôsledkom, odvodí všetky vety jazyka. Nemonotónny usudzovateľ naopak, ak ešte pred chvíľou odvodil dôsledok A, po pribratí nových predpokladov do bázy znalostí ho už odvodiť nemusí. Ktoré chovanie je viac v súlade s bežným „sedliackym rozumom“ je dosť zrejmé.

1. spochybňujúce už Fodorove predpoklady o nutných vlastnostiach kognitívnej architektúry. Do tejto kategórie spadá najmä *eliminatívny materializmus*, zástanci ktorého tvrdia, že aktivačné vektory tvoria kľúčovú (jedinú) formu reprezentácie a vektorové zobrazenia hlavný (jediný) druh výpočtu v mozgu. Propozičnú reprezentáciu, symbolický výpočet, ako aj „ľudovú psychológiu“, založenú prevažne na logickej dedukcii, odmietajú ako prebytočnú konštrukciu. Očakávajú vývoj, podobný napr. redukcii klasickej termodynamiky na časticovú.

2. akceptujúce Fodorove predpoklady a ďalej:

   (a) akceptujúce záver (2) - že konekcionizmus ponúka nanajvýš implementáciu klasických architektúr. Títo konekcionisti vidia konekcionizmus, ako most medzi biologicko-fyzikálnou neurovedou a kognitívnou vedou. Neurónové siete tak podľa nich vypĺňajú priestor medzi vysokými úrovňami vedomia a fyzikálnou realitou.

   (b) spochybňujúce (2). Tento tábor tvrdí, že Fodorov záver sa zakladá na predpoklade zreťaziteľnej (concatenative) reprezentácie. Konekcionistické modely však pracujú s distribuovanou reprezentáciou. Tak možno tvrdiť, že potenciálne môžu spĺňať Fodorove požiadavky na kognitívnu architektúru a zároveň nebyť implementáciou klasickej architektúry.

2.4 Tretia úroveň reprezentácie

Peter Gärdenfors stojí niekde medzi názormi 2a a 2b z delenia v predošlom odseku. Uvedomujúc si obmedzenia symbolickej paradigmy navrhuje zaviesť tretiu formu modelovania informácie a nazývať ju *konceptuálna reprezentácia*, keďže očakáva, že práve na tejto úrovni možno vysvetľovať s modelovať vznik konceptov a ich vývoj. Tri úrovne reprezentácie teda podľa neho sú:

- subkonceptuálna (konekcionizmus)
- konceptuálna
- lingvistická (symbolizmus)

Gärdenfors zdôrazňuje potrebu vnímať všetky tieto tri úrovne ako rôzne uhly pohľadu na reprezentovanie informácie v inteligentnom systéme.

Medzi hlavné úlohy konceptuálnej úrovne reprezentácie patrí:

- *poskytovať sémantiku pre symbolickú reprezentáciu* (symbol grounding). Ako sme uviedli vyššie, symbolická úroveň základné koncepty len pomenúva - nemodeluje ich.
- *umožniť explicitnú reprezentáciou konceptov*, dovoľujúcu modelovať ich vznik a vývoj

2.5 Sémantika jazyka

Úplne neformálne možno povedať, že sémantika definuje význam jazyka: je to relácia medzi množinou výrazov nejakého jazyka a množinou entít (nazvime ju $\tilde{U}$), na ktoré sa jazyk vzťahuje.

Podľa spôsobu voľby množiny $\tilde{U}$[3] môžeme prístup k definícii sémantiky rozdeliť na *realistický* a *kognitivistický*.

Realistická sémantika

Realisti chápu sémantiku ako zobrazenie do abstraktného *univerza* – jej hlavným účelom je určovať *pravdivostnú hodnotu* vety jazyka. Konštantám (menám) sa priradia objekty z univerza, funkčným symbolom zobrazenia nad objektami, predikátom relácie medzi objektami. Na základe týchto zobrazení možno rozhodnúť o pravdivosti celej vety jazyka.

Pre príklad realistickej sémantiky môžeme siahnuť po predikátovej logike prvého rádu. Abeceda jej jazyka L^{FOL} pozostáva z nasledujúcich tried symbolov:

- premenné $(X, Y, Z, \ldots)$
- funkčné symboly $(f, g, \ldots)$, pričom funkčné symboly s nulovou aritou sú konštanty
- predikátové symboly (obvykle $p, q, r, \ldots$), pričom predikátové symboly s nulovou aritou sú výrokové premenné
- logické operátory $(\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow, \exists, \forall)$

Interpretácia jazyka L^{FOL} je potom usporiadaná dvojica (I, U), kde:

- U je univerzum - akákoľvek množina objektov
- I je sémantická funkcia, ktorá priradí:
  - každej konštante práve jeden prvok z U
  - každému funkčnému symbolu s nenulovou aritou n zobrazenie $\varphi : U^n \rightarrow U$
  - každej výrokovej premennej (predikátový symbol s nulovou aritou) práve jednu hodnotu z množiny pravdivostných hodnôt $\{t, f\}$
  - každému predikátovému symbolu s nenulovou aritou n podmnožinu $\sigma \subseteq U^n$.

    Uzavretej atomárnej formule $p(t_1, t_2, \ldots, t_n)$ je potom v interpretácii (I, U) priradené t (formula je *pravdivá*) vtedy a len vtedy, ak $(I(t_1), I(t_2), \ldots, I(t_n)) \in I(p)$.
  - logickým operátorom $\wedge, \vee, \Rightarrow, \Leftrightarrow$ zobrazenie $\{t, f\}^2 \rightarrow \{t, f\}$:

[3] Veľmi zjednodušene povedané. Toto delenie totiž úzko súvisí s realistickým resp. antirealistickým postojom k vzťahu ľudskej teórie a „objektívnej“ reality. Tento závažný filozofický problém nechceme na tomto mieste otvárať. Budeme sa zaoberať skôr povahou a modelovými aplikáciami samotnej kognitívnej sémantiky.

		$\wedge$	$\vee$	$\Rightarrow$	$\Leftrightarrow$
f	f	f	f	t	t
f	t	f	t	t	f
t	f	f	t	f	f
t	t	t	t	t	t

operátoru $\neg$ zobrazenie $\phi : \{t, f\} \rightarrow \{t, f\}$; $\phi(t) = f, \phi(f) = t$
Formula $\exists XF$ je interpretovaná ako t vtedy a len vtedy, ak $\exists x \in U$ také, že po jeho priradení do premennej X je formula F interpretovaná ako t.

Závažný dôsledok realistického prístupu je, že takto chápaný význam viet (ľudského) jazyka je *nezávislý od ich interpretácie konkrétnym agentom* (človekom) používajúcim jazyk.

Kognitívna sémantika

Naproti tomu *kognitívna* sémantika prichádza s nasledujúcou ústrednou myšlienkou:

- význam výrazov jazyka tvoria mentálne entity používateľa jazyka

Takáto sémantika je zobrazením z množiny výrazov jazyka do vhodných kognitívnych štruktúr agenta[4], pričom podľa Gärdenforsa práve konceptuálna úroveň reprezentácie by mala poskytovať takéto štruktúry. Významný rozdiel od realistického prístupu je, že kognitívna sémantika nepotrebuje na definovanie významu výrazu jeho vzťah k pravde. Dôležitá je len vnútorná prijateľnosť interpretácie výrazu pre agenta.

Kognitivistická paradigma je možno na prvý pohľad v niečom podobná Fodorovej „Mentálčine“ („Language of thought“, Fodor (1981)). Podľa Fodora je Mentálčinou to, čo človek používa, keď inferuje resp. keď formuluje lingvistické výrazy v prirodzenom jazyku. Čo sa týka sémantiky (Mentálčiny), je však Fodor realista[5] – Mentálčinu teda zavádza len ako ďalšiu syntaktickú úroveň medzi prirodzený jazyk a univerzálnu „ríšu významov“.

[4]Gärdenfors v [8] argumentuje, že aj na samotný jazyk sa môžeme pozerať ako na „kognitívnu štruktúru“ a že teda kognitívnu sémantiku možno chápať ako reláciu nad kognitívnymi štruktúrami

[5]nakoniec, veď je to Fodor!

Kapitola 3

Konceptuálny priestor

Pojem *konceptuálny priestor* (conceptual space – CS) zavádza Peter Gärdenfors na prelome 80. a 90. rokov a má tvoriť možnú geometrickú štruktúru pre konceptuálnu úroveň reprezentácie. Konceptuálny priestor ma tieto základné vlastnosti:

- tvoria ho kvalitatívne rozmery (dimenzie[1])
- je to metrický priestor – umožňuje zaviesť metriku, ktorá udáva mieru podobnosti reprezentovaných objektov

Dimenzie tvoriace konceptuálny priestor Gärdenfors primárne delí na:

- *vrodené* – dané konštrukciou senzorického aparátu a spôsobom spracovania jeho výstupov. Medzi takéto dimenzie môžeme u človeka s určitou mierou zjednodušenia zaradiť zrakové (farba, farebná sýtosť, jas, poloha na scéne), sluchové (výška a intenzita tónu, fázový posun medzi ľ./p. uchom), čuchové a chuťové dimenzie, polohu na povrchu tela a pod.
- *získané*
  - *sociálne*. Sem možno zaradiť napr. dimenziu vnímania času. Gärdenfors argumentuje, že rôzne ľudské kultúry vnímajú čas rôznym spôsobom. Okrem štruktúry časovej dimenzie blízkej západnej civilizácii (celá reálna os so stredom „v súčasnosti“) to môže byť ešte poloos $\mathbb{R}^-$, alebo cyklické vnímanie času.
  - *funkcionálne* – popisujúce vlastnosti komplexných objektov a javov. Tieto môžu mať rôznu štruktúru (okrem lineárneho usporiadania je predstaviteľné aj čiastočné usporiadanie, diskrétne domény) a je otázne, ako by sa dala zaviesť metrika priestoru, ktorého zložkami by boli takéto domény. Zatiaľ sme nenašli v žiadnej nám dostupnej literatúre zmienku o úvahách týmto smerom. Všetci autori (zdá sa, že vrátane Gärdenforsa) len „povinne“ opakujú, že treba brať do úvahy

[1] Pojem dimenzie v práci používame v dvoch rôznych významoch. Prvým je, ako aj v tomto prípade, jeden konkrétny rozmer priestoru. Druhým je štandardný pojem „dimenzia vektorového priestoru“ z algebry: počet prvkov niektorej z báz priestoru (alebo inak: maximálny počet lineárne nezávislých vektorov). V euklidovskom vektorovom priestore $\mathbb{R}^n$ je dimenzia rovná n.

aj takéto domény, taktne však pracujú ďalej s analógiou euklidovského priestoru, ktorá je v tomto prípade dosť zavádzajúca.

– *vedecké* – dimenzie, zavedené vedou. Majú umožňovať klasifikáciu objektov sveta podľa dimenzií, zavádzaných vedou.

Ústredná myšlienka geometrie konceptuálnych priestorov, postulovaná Gärdenforsom, je nasledovná:

- *„Kritérium P“*: prirodzenému konceptu zodpovedá v konceptuálnom priestore *konvexná množina bodov.*

Takýmto konvexným oblastiam zodpovedajú u človeka prirodzené pojmy, ako „červená“, „zelená“, „horký“, „guľatý“ a pod. Znova sa tu berie do úvahy analógia euklidovského priestoru, kde medzi každými dvoma bodmi existuje súvislá najmenšia cesta a je ňou priamka. Konvexnosť množiny sa potom chápe v bežnom zmysle, že s každými dvoma bodmi množiny do nej patria aj všetky body, ležiace na ich spojnici.

Generické body (reprezentácie objektov) v konceptuálnom priestore budeme ďalej nazývať *knoxely*[2] a značiť $k \in CS$, prípadne $\vec{k} \in CS$, pokiaľ budeme potrebovať zdôrazniť ich vektorovú povahu. Knoxely v blízkosti *prototypu* (centrálny bod konceptu) budeme nazývať *inštanciami prototypu.*

Príklad jednoduchého konceptuálneho priestoru vidíme na obr. 3.1.

3.1 Geometrický model

V zmysle prototypovej teórie sa ponúka možnosť reprezentovať prirodzené koncepty ich centrálnymi bodmi - prototypmi. Uvažujme množinu prototypov $C = \{c_1, c_2, \ldots, c_n\}$. Generický bod v konceptuálnom priestore potom spadá pod koncept, ktorého prototyp je k nemu najbližšie[3] (vzhľadom na metriku):

$$Concept(x) = \arg\min_{c \in C} d(x, c)$$

Fukncia $Concept(x)$ segmentuje priestor na disjunktné konvexné podmnožiny. Hranicu medzi dvoma najbližšími centrami tvorí množina bodov:

$$\{x \in CS \mid d(x, c_1) = d(x, c_2)\}$$

V EVP[4] je touto množinou hyperrovina (pričom spojnicu centier pretína v jej strede a je na ňu kolmá). Popisované rozdelenie sa nazýva *Voronoiho mozaika* (tesselation). Príklad uvádzame na obr. 3.2.

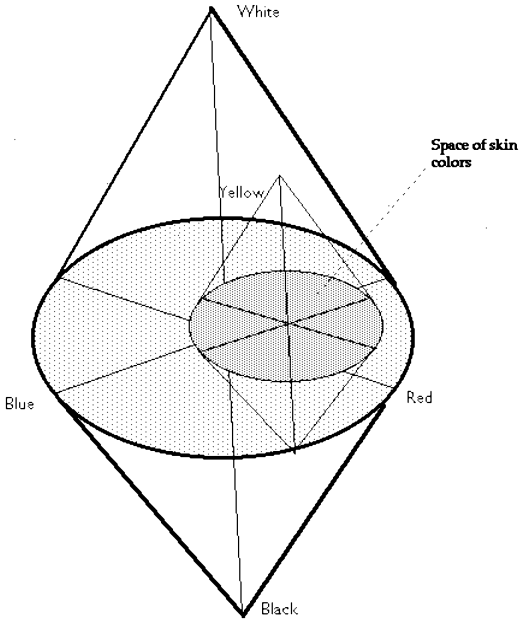
3.2 Nedostatky modelu

Takáto reprezentácia konceptov však má nedostatky, z ktorých najvýznamnejšie podľa nášho názoru sú:

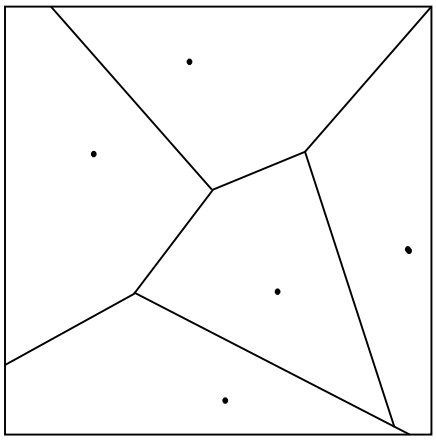
[2] Inšpirácia vychádza z analogického pojmu „pixel“ používaného pri digitálnom spracovaní obrazu. Pojem preberáme z Chella, Frixione, Gaglio (1999) [5]

[3] Hovoríme, že objekt je inštanciou prototypu

[4] Euklidovský vektorový priestor



Obr. 3.1: Notoricky známy HSB (hue, saturation, brightness) kužeľ percepcie farieb s podpriestorom farieb pokožky. Tento priestor sa dá samozrejme premietnuť aj na iný farebný model (RGB, CMYK), vždy však zostáva dimenzia 3 (CMYK redundantne reprezentuje čiernu).



Obr. 3.2: Ukážka Voronoiho mozaiky generovanej 5 centrami v dvojrozmernom pristore

- koncepty sú z definície disjunktné - nemožno reprezentovať prekrývajúce sa kategórie. Pritom zjavne možno očakávať, že napr.:

$$Concept(\text{„Sova pálená“}) \subseteq Concept(\text{„sova“}) \subseteq$$
$$\subseteq Concept(\text{„vták“}) \subseteq Concept(\text{„živý objekt“})$$

  Tento prípad, kde sú koncepty do seba vnorené, sa dá modelovať zavedením *hierarchie* v segmentovaní. Región konceptu „živý objekt“ by sme mohli rozdeliť vnorenou Voronoiho mozaikou, ktorej segment prislúchajúci konceptu „vták“ by sme delili ďalej atď. . . .

  Ale napr. pre $Concept(\text{„živý objekt“}) = A$, $Concept(\text{„lietajúci objekt“}) = B$ platí:

$$A \cap B \neq \emptyset \ \wedge \ A \not\subseteq B \ \wedge \ B \not\subseteq A$$

  Otázne však je, či by sa pre uvedené koncepty vôbec dal navrhnúť konceptuálny priestor, v ktorom by boli oba konvexné. Pre jednoduché priestory – napr. parametrický priestor superkvadrík[5] to predstaviteľné je. Môžeme uvažovať napr. koncepty „kváder“ a „zvislý objekt“.

- veľkosť regiónu každého konceptu je určená len vzdialenosťou a polohou najbližších susedných prototypov.

  Gärdenfors v [6] navrhuje generalizáciu Voronoiho mozaiky tak, aby bolo možné váhovať prototypy. Navrhuje zaviesť namiesto protypového bodu *prototypovú oblasť* tvaru hypergule s definovateľným polomerom. *Vzdialenosť generického bodu od prototypu by sa potom nemerala od centrálneho bodu, ale od hranice hypergule* (3.1). Zväčšovaním polomeru prototypovej hypergule by sa tak dala spojito zväčšovať celá oblasť konceptu na úkor susedných oblastí. Hranicu medzi dvoma najbližšími centrami tvorí množina bodov (3.3):

$$d_g(x, c) = d(x, c) - r_c \tag{3.1}$$

$$Concept_g(x) = \arg\min_{c \in C} \ d_g(x, c) \tag{3.2}$$

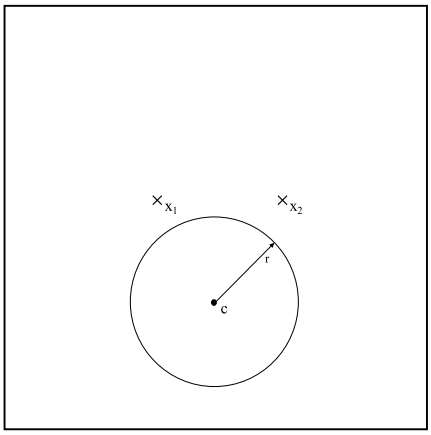
$$\{x \in CS \mid d(x, c_1) - r_{c_1} = d(x, c_2) - r_{c_2}\} \tag{3.3}$$

  Problém je, že „metrika“ d_g, pokiaľ sme správne pochopili Gärdenforsov popis, nespĺňa definíciu metriky. Nie je totiž splnená vlastnosť $(\forall x, y \in E : (d(x, y) = 0) \Rightarrow (x = y))$[6] (všetky body na prototypovej kružnici majú od centra vzdialenosť 0) ani trojuholníková nerovnosť (obr. 3.3).

  Okrem toho hranice oblastí, definované rovnosťou (3.3) nie sú vo všeobecnom prípade hyperroviny. Oblasti priestoru, generované zobrazením

[5] Geometrické modelovacie primitíva s parametrizovateľným tvarom. Definíciu uvádzame v odseku 5.1.1 na strane 24

[6] Toto by sa dalo obísť vhodnou definíciou priestoru. Konkrétne takou, ktorá by z neho explicitne vylúčila body patriace hraniciam hypergúľ.



Obr. 3.3: Pre body blízko prototypovej hypergule platí: $d_g(x_1, c) + d_g(c, x_2) < d_g(x1, x2)$ (keďže vzdialenosť od centrálneho bodu sa meria od hranice hypergule, z jedného bodu do druhého je bližšie cez centrum ako priamo). Nie je teda splnená trojuholníková nerovnosť - nutná podmienka pre to, aby zobrazenie d_g bolo metrikou priestoru.

$Concept_g$ v dôsledku toho nemusia byť konvexné[7] (obr. 3.4). To je samozrejme v rozpore s „Kritériom P“. Vhodnosť takto generalizovanej Voronoiho mozaiky na geometrické modelovanie konceptov je teda dosť rozporuplná.

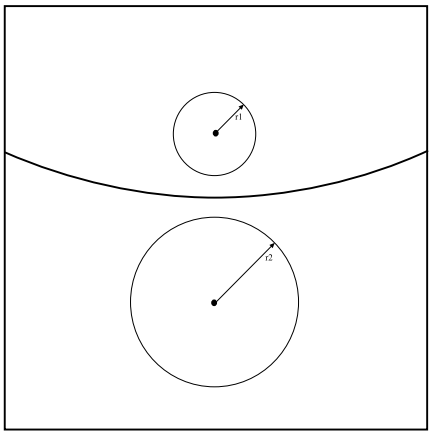
- Každý bod (objekt) priestoru formálne prislúcha nejakému konceptu, bez ohľadu na to, aká vzdialenosť medzi nimi je. Táto skutočnosť celkom korešponduje s klasifikovaním objektov malými deťmi, ktoré keď poznajú niekoľko málo prototypov, majú tendenciu čokoľvek, čo vidia zaklasifikovať na základe nich. Dospelý človek však s istotou rozoznáva, že ide o niečo „neznáme“, ak vníma objekt, ktorý je priveľmi vzdialený všetkým jemu známym prototypom.

3.3 Operácie nad konceptuálnym priestorom

Medzi najdôležitejšie formálne operácie nad konceptuálnym priestorom patria:

- *relácia podobnosti* knoxelov. Základná operácia nad CS, využívajúca priamo jeho metrické vlastnosti.
- *adresácia konceptov*, reprezentovaných ich prototypom. Využíva reláciu podobnosti, dá sa realizovať autoasociatívnou pamäťou.
- *asociovanie knoxelov* do páru resp. množiny, *zreťazovanie knoxelov* do postupnosti
- *zobrazenia nad podpriestormi CS* (metafory, analógie)

[7] Pokiaľ zostávame pri konvexnosti, založenej na lineárnej kombinácii (M je konvexná množina, ak $\forall \vec{u}, \vec{v} \in M : k\vec{u} + (1-k)\vec{v} \in M,\ k \in (0,1)$)



Obr. 3.4: Príklad nekonvexnosti oblasti, definovanej metrikou na Gärdenforsovej generalizovanej Voronoiho mozaike pre dve centrá (prototypy) s rôznymi prototypovými polomermi. Hranice oblastí tu vo všeobecnom prípade nie sú hyperroviny.

- *interpretačná funkcia* – zobrazenie z množiny lingvistických výrazov do konceptuálneho priestoru
- *denominačná funkcia* – zobrazenie z konceptuálneho priestoru do množiny lingvistických výrazov

3.4 Konceptuálny priestor ako kognitívna sémantika

Základná myšlienka interpretácie do konceptuálneho priestoru spočíva v tom, že *prirodzeným predikátom jazyka sa priradia prirodzené koncepty*, ktoré sú v CS tvorené konvexnými množinami („Kritérium P“). Platnosť týchto predikátov je potom overiteľná analyticky, priamo na základe metriky konceptuálneho priestoru.

3.5 Výhodné vlastnosti konceptuálnych priestorov

- Výhoda metriky. Dá sa na základe nej vyjadrovať vzájomná podobnosť objektov, reprezentovaných v CS. To je silným základom pre kľúčové druhy operácií nad CS.
- Fakt, že výrazy jazyka nie sú interpretované na homogénnej, neštruktúrovanej množine prvkov prináša značné výhody. Bohatá štruktúra konceptuálneho priestoru umožňuje „implicitnú“ – analytickú splniteľnosť primitívnych predikátov, ktoré by v realistickej sémantike museli byť definované „zvonka“.

- dobré narábanie s *metaforami*. Dajú sa tu dobre založiť ako zobrazenia nad podpriestormi CS. Realistická sémantika má problémy.
- biologická relevantnosť. CS je perceptuálne založený.

3.6 Problémy

- *aké dimenzie* by mal konceptuálny priestor obsahovať, okrem percepčne založených
- ako chápať konceptuálnu reprezentáciu - ako reálne existujúce štruktúry v stavbe kognitívneho aparátu, alebo skôr ako konštrukciu prisudzovanú systému pozorovateľom (teda niečo ako pojem „ťažisko telesa“)
- kde hľadať hodnoty zložiek dimenzií konceptuálneho priestoru. S tým úzko súvisí aj napojenie na subkonceptuálnu vrstvu. Základné námety, ako vyformovať z holistickej reprezentácie, vlastnej neurónovým sieťam, hodnoty vhodných dimenzií konceptuálneho priestoru ponúka napríklad Balkenius v [1]. Vzťahu aktivačného priestoru neurónovej siete a konceptuálneho priestoru sa venujeme v odseku 4.2.
- *pribúdanie dimenzií* je možný spôsob, navrhovaný Gärdenforsom, ako reprezentovať zložené objekty a ako prisudzovať objektom nové vlastnosti. Otázna je však relevantnosť takéhoto prístupu z hľadiska praktických aplikácií.
- koncepty nad konceptami, podobne ako v predikátovej logike potreba narábať s prvorádovými predikátmi (kvantifikovať, stavať nad nimi predikáty, ...), ktorá vedie k logikám vyšších rádov)
- reprezentácia relácií nad objektami

Kapitola 4

Neurónové siete

4.1 RAAM – rekurentná autoasociatívna pamäť

4.1.1 Definícia architektúry

Model popísal Pollack (1990) [17], pričom jeho úmyslom bolo navrhnúť mechanizmus, umožňujúci konekcionistickým architektúram spracovať symbolicky reprezentované dáta – konkrétne postupnosti a stromy premenlivej veľkosti nad konečnou abecedou. Jeho zámerom bolo navrhnúť generický *kodér*, schopný transformovať strom „bottom-up“ na reálny vektor konštantnej šírky a *dekodér*, schopný naopak dekódovať tento vektor na pôvodný strom.

Uvažujme teda binárne stromy. Pripusťme existenciu kodéra, realizujúceho zobrazenie $\Phi : \mathbb{R}^k \times \mathbb{R}^k \to \mathbb{R}^k$ a dekodéra, realizujúceho inverzné zobrazenie $\Phi' : \mathbb{R}^k \to \mathbb{R}^k \times \mathbb{R}^k$. Uvažujme strom $T = ((AB)(CD))$, kde A, B, C, D sú symboly vstupnej abecedy reprezentované ako rôzne (najlepšie ortogonálne) vektory z $\{0,1\}^k$. Ak chceme získať reprezentáciu stromu T, aplikujeme kodér rekurentne odspodu hore na uzly stromu:

$$v_t = \Omega(T) = \Phi(\Phi(A, B), \Phi(C, D))$$

Zároveň dostaneme aj reprezentácie podstromov (AB) a (CD).

Reálny vektor, reprezentujúci strom dekódujeme dekodérom rekurentne späť „up-down“:

$$\begin{aligned} \Phi'(v_t) &= (v_{t_1}, v_{t_2}) \\ \Phi'(v_{t_1}) &= (A', B') \\ \Phi'(v_{t_2}) &= (C', D') \end{aligned}$$

Postupnosti symbolov dĺžky n spracovávame tak, že ich chápeme ako na jednu stranu sa vetviace binárne stromy hĺbky n:

$$a_1 a_2 \ldots a_n \Leftrightarrow ((((a_1 nil) a_2) \ldots) a_n)$$

Konekcionistická implementácia

Kodér aj dekodér sú v Pollackovom RAAM-e implementované ako štandardné dvojvrstvové dopredné neurónové siete so sieťovou funkciou v tvare polynómu

1. rádu a sigmoidálnou aktivačnou funkciou. Využíva sa tu skutočnosť, že troj- a viacvrstvová dopredná sieť je univerzálnym aproximátorom spojitých funkcií $\mathbb{R}^m \to \mathbb{R}^n$. Pokiaľ teda existujú spojité zobrazenia Φ, Φ' s vlastnosťami uvedenými v odseku 4.1.1, existuje dopredná sieť (a priradenie váh) aproximujúca zobrazenia s dostatočnou presnosťou.

4.1.2 Učenie siete

Pri učení sa kodér a dekodér spojí do jedinej trojvrstvovej siete, realizujúcej zobrazenie $(\hat{\Phi} \equiv \Phi'(\Phi) : \mathbb{R}^k \times \mathbb{R}^k \to \mathbb{R}^k \times \mathbb{R}^k$. Prostrednú vrstvu, ktorá je výstupnou vrstvou kodéra a zároveň vstupnou vrstvou dekodéra, nazývame *kontextová vrstva*. Túto výslednú sieť potom učíme štandardným učením s učiteľom metódou spätného šírenia chybového gradientu (Backpropagation, Rumelhart et al. (1986)). Učenie prebieha po pároch, pričom na výstupe siete sa očakáva rovnaký pár ako je predložený na vstupe. Príklad pre náš strom $((AB)(CD))$:

vstup		kontext		výstup	požadovaný výstup
(A, B)	$\to$	v_{t_1}	$\to$	(A', B')	(A, B)
(C, D)	$\to$	v_{t_2}	$\to$	(C', D')	(C, D)
(v_{t_1}, v_{t_2})	$\to$	v_t	$\to$	(v'_{t_1}, v'_{t_2})	(v_{t_1}, v_{t_2})

Váhy v celej sieti sa modifikujú po každom predloženom páre na základe chybových gradientov tak, aby sa minimalizovala chyba. Významný rys popísaného algoritmu učenia je, že hodnoty, na ktoré kodér mapuje podstromy, v našom príklade v_{t_1} a v_{t_2}, sa využívajú v ďalších krokoch učenia a so zmenami váh v priebehu učenia sa samozrejme menia – mení sa teda aj cieľ učenia. Takýto druh učenia sa nazýva „moving target learning“ (BPMT - backpropagation, moving target). Konvergencia algoritmu BPMT nie je zaručená a je citlivá na počiatočné podmienky, ako aj parametre učenia. Learning rate sa používa čo najnižší ($\alpha \in [0.1, 10^{-5}]$) a moment sa zvyčajne nepoužíva vôbec.

4.1.3 Výhody a nevýhody

- výhody:
  - schopnosť nielen klasifikovať, ale aj výhodne reprezentovať konečné postupnosti a stromy vektorov z $\mathbb{R}^N$ s využitím regularít predkladaných dát
  - schopnosť realizovať *inverzné zobrazenie* - dekódovať reprezentáciu štruktúry (stromu, postupnosti) späť na pôvodnú štruktúru. Treba podotknúť, že túto schopnosť má len veľmi málo iných neurálnych architektúr.
  - jednoduchá architektúra, postavená na modeli doprednej siete aj s využitím štandardného algoritmu učenia, symetria páru kodér/dekodér, efektívna výpočtová realizácia
  - neformálne preukázaná určitá netriviálnosť reprezentácie (v zmysle, že RAAM nepracuje len ako posuvný register, ktorý „nasúka“ bity symbolickej informácie do bitov reprezentácie reálnych čísel) a generalizačná schopnosť (správna klasifikácia syntakticky odlišných podštruktúr, ktoré neboli v trénovacej množine) a aj schopnosť extrahovať významné črty tried predkladaných štruktúr.

  - schopnosť reprezentovať do určitej hĺbky niektoré nie regulárne bezkontextové jazyky z triedy jazykov, akceptovateľných počítadlovými automatmi (napr. $a^n b^n$)

- nevýhody:
  - silná závislosť vytváranej reprezentácie a úspešnosti učenia od počiatočných podmienok a parametrov učenia
  - neexistencia formálneho popisu ekvivalentnej triedy jazykov (vzťah s formálnymi jazykmi Chomského hierarchie) a teda nejasná sila modelu ako formálno-jazykovej abstrakcie
  - problematický test ukončenia pri dekódovaní (jednoduché metódy sú nepoužiteľné na zložitejšie štruktúry)
  - problematická rozšíriteľnosť – s rastúcim rozsahom symbolickej informácie (mohutnosť abecedy, hĺbka stromu) pochopiteľne rastú nároky na presnosť reprezentácie (reálnej aritmetiky použitej na výpočty), počiatočné podmienky, parametre algoritmu učenia a test ukončenia dekódovania.
  - problematická biologická relevantnosť modelu/reprezentácie, najmä čo sa týka použitia učenia s učiteľom, reálnych čísel s veľkými nárokmi na presnosť a fraktálnych zovšeobecnení [15] pre nekonečné štruktúry.

4.1.4 Demonštračný príklad

Na demonštráciu vlastností modelu uvádzame jednoduchý príklad aplikácie RAAM-u na spracovanie štruktúrovaných symbolických dát. Ukážeme schopnosť modelu, vytvoriť jednoznačnú reprezentáciu krátkych slov z bezkontextového jazyka $L = a^n b^n \cup c^m d^m$. Použili sme sieť s topológiou (4+2)-2-(4+2), abecedu $\Sigma = \{a, b, c, d\}$ a štandardné ortonormálne kódovanie symbolov metódou jeden z n ($v_a = (0,0,0,1), v_b = (0,0,1,0), \ldots, v_d = (1,0,0,0)$). Trénovacia množina obsahovala 60 slov z uvedeného jazyka (krátke sa v nej nachádzali častejšie). Treba povedať, že pre sieť s dvoma kontextovými neurónmi je táto úloha veľmi náročná. Kontextová vrstva dimenzie 2 je však veľmi výhodná pre možnosť jednoducho vizualizovať jej aktivačný priestor.

BPMT učenie, popísané v predošlých odsekoch nebolo schopné ani pri niekoľkých desiatkach opakovaných spustení natrénovať sieť tak, aby 100% reprezentovala všetky slová v trénovacej množine (učenie dosahovalo max. 63% úspešnosť). Váhy siete sme preto adaptovali genetickým algoritmom bez kríženia, s operátorom mutácie pracujúcim formou pripočítavania náhodných čísel s gaussovskou distribúciou ($\mu = 0, \sigma = 0.08$) k zložkám totálneho váhového vektora siete. Hodnotou účelovej funkcie pre GA bol priamo počet zle zaklasifikovaných príkladov z trénovacej množiny (ďalej zjemnený pripočítaním vhodného podielu sum-of-sqaures chyby aktivácií výstupnej vrstvy, aby bol povrch účelovej funkcie aspoň čiastočne spojitý). Týmto spôsobom možno riešenie, 100% pokrývajúce trénovaciu množinu (aj so schopnosťou generalizácie na dlhšie reťazce), nájsť veľmi rýchlo (hoci je samozrejme otázna relevantnosť takejto formy adaptácie z hľadiska napr. on-line učenia, o biologickej plausibilnosti ani nehovoriac).

Povahu reprezentácie ako aj dynamiky adaptovanej siete vidno z obr. 4.1. Graf (b) obsahuje „konvergenčné pole“ kodéra v kontextovom priestore siete.

Šípky vedú z pevne zvoleného bodu $\vec{c}_0 = (c_0^1, c_0^2)$ do bodu $\vec{c}_1 = (c_1^1, c_1^2)$, kde $\vec{c}_1 = \Phi(char_{const}, \vec{c}_0)$ je kontext získaný jednou časovou iteráciou kodéra siete pre pevný znak na znakovom vstupe. Graf (a) obsahuje to isté pole, šípky sú však normalizované (majú zachovanú orientáciu, ale normalizovanú dĺžku). Zobrazené polia sú pre znak „b“. Z grafov je vidno fixpointovú povahu dynamiky siete, aj približnú polohu pevného bodu kodéra pre znak „b“. Pevné body pre zvyšné 3 znaky boli rozmiestnené vo zvyšných rohoch priestoru. Na grafe (c) je nenormalizované konvergenčné pole dekodéra siete. Dynamika dekodéra je trochu inej povahy, čo vyplýva z jeho inverznej funkcie. Pole je približne symetrické okolo priamky $y = 0.5$. Dva rôzne pevné body dekodéra ležia pri vrchnom a spodnom pravom rohu priestoru. Na grafe (d) je príklad trajektórie, po ktorej sa pohybuje kontextový vektor kodéra pri kódovaní (čierna) a následne dekodéra pri dekódovaní (modrá) reťazca a^6b^6. Na obr. 4.2 je Voronoiho mozaika kontextového priestoru, definovaná hyperrovinami neurónov výstupnej vrstvy. Z grafu je zrejmé, aký znak bude dekodér generovať v závislosti od kontextu v čase t.

Matematické pozadie, vhodné na analýzu dynamiky rekurentnej siete „akceptujúcej“ počítadlové jazyky môže prípadný záujemca nájsť napr. v [18].

4.2 Porovnanie vlastností aktivačného priestoru neurónovej siete a konceptuálneho priestoru

Majme množinu neurónov $\{N_1, N_2, \ldots, N_n\}$ s oborom hodnôt aktivačnej funkcie (a, b). Pod pojmom *aktivačný priestor* množiny neurónov máme na mysli kartézsky súčin $(a, b)^n \subseteq \mathbb{R}^n$, ktorého zložky tvoria aktivácie jednotlivých neurónov.[1]

Uvažujúc napr. štandardnú doprednú 3-vrstvovoú neurónovú sieť typu n_i-n_h-n_o, natrénovanú spätným šírením chyby na aproximáciu zobrazenia $R^i \rightarrow R^o$, má zmysel skúmať aktivačný priestor skrytej vrstvy, kde sa (po nastavení vstupu siete a následnom doprednom šírení aktivácií) nachádza „vnútorná reprezentácia“ predkladaných vstupných vektorov.

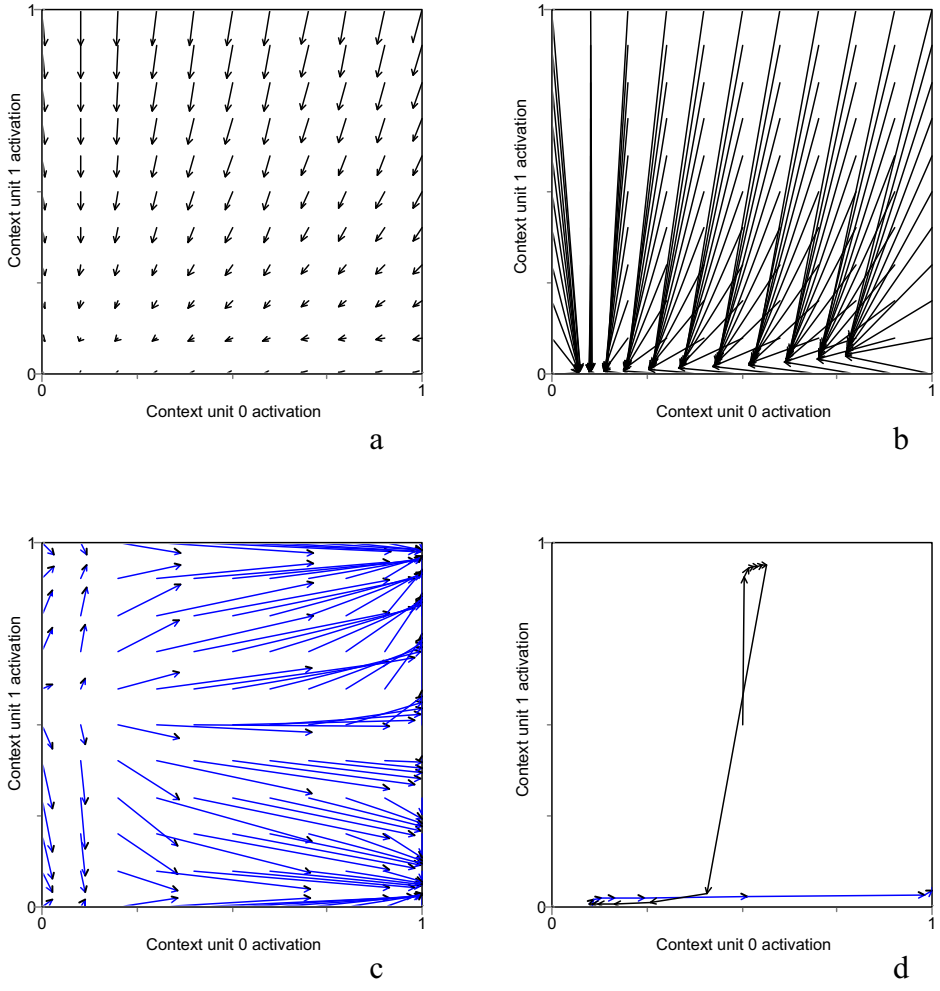
Ak si uvedomíme, že (1) body aktivačného priestoru predstavujú reprezentáciu vektorov (objektov) predkladaných sieti a (2) neuróny výstupnej vrstvy siete klastrujú tento priestor hyperrovinami na konvexné oblasti, môže nás to navádzať k hľadaniu analógií medzi aktivačnými a konceptuálnymi priestormi. Zamerajme sa teda v skratke na analogické a rozdielne vlastnosti týchto priestorov.

Konceptuálny priestor:

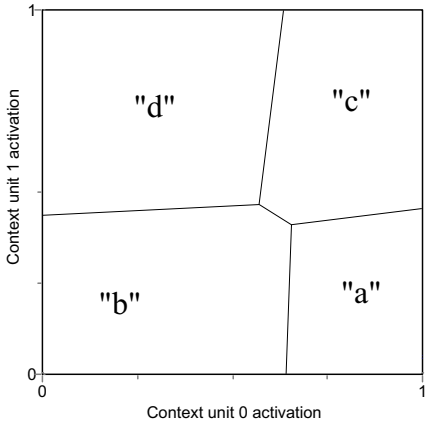
- dimenzie sú kvalitatívne - každá popisuje inú, jednoznačne definovanú kvalitu reprezentovaného objektu. Je bližší skôr priestoru atribútov rámca z rámcovej reprezentácie (so zavedenou metrikou)
- konvexné podmnožiny odpovedajú prirodzeným konceptom - „Kritérium P“.

Aktivačný priestor NN:

[1] Kartézsky súčin intervalov $(a, b)^n$ nie je podpriestorom, ale len podmnožinou vektorového priestoru $\mathbb{R}^n$. Nie je totiž uzavretý vzhľadom na súčet a násobenie skalárom



Obr. 4.1: Dynamické vlastnosti RAAM-u, reprezentujúceho jazyk $L = a^n b^n \cup c^m d^m$. Jednotlivé grafy sú popísané v texte na strane 18.



Obr. 4.2: Segmentácia kontextového priestoru hyperrovinami „znakových“ neurónov výstupnej vrstvy dekodéra siete. Ak dekodéru odovzdáme na vstupnú - kontextovú vrstvu vektor zo segmentu, prislúchajúcemu niektorému znaku, bude aktivačný vektor na znakovom výstupe dekodéra najbližší (v zmysle euklidovskej vzdialenosti) reprezentácii daného znaku.

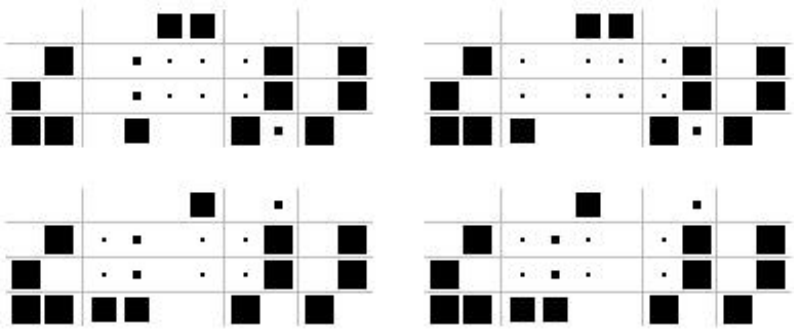
- dimenzie zvyčajne nie sú kvalitatívne – informácia môže byť reprezentovaná distribuovane (žiadna dimenzia nedefinuje jednu kvalitu úplne, informácia je rozptýlená medzi viacerými dimenziami) a redundantne (nie je možné nájsť dimenziu, vymazanie ktorej by znemožnilo prenos úplnej informácie – príklad je na obr. 4.3).

  Využitie aktivačného priestoru siete, aj keď ju necháme aproximovať nejaké dobre definované zobrazenie, silno závisí od takmer všetkých parametrov modelu - chybovej funkcie, aktivačných funkcií, počiatočných váh, použitého algoritmu a parametrov učenia, voľby trénovacej množiny a jej permutácií pri predkladaní.

- konvexné podmnožiny hrajú významnú úlohu.

  Tento fakt vyplýva z povahy dopredných sietí. Pri obmedzení na lineárne aktivačné funkcie delí každý neurón aktivačný priestor predošlej vrstvy (prvá skrytá vstupný priestor) hyperrovinou (pre nejaký zvolený prah aktivácie). Vektorový priestor je potom vrstvou s k neurónmi rozdelený Voronoiho mozaikou na k jednoduchých konvexných oblasti, pričom vo vnútri i-tej je aktivácia i-teho neurónu maximálna.

- euklidovská vzdialenosť bodov tu nehrá najdôležitejšiu úlohu. Aj body s malou vzdialenosťou môžu byť od seba oddelené hyperrovinou neurónu nasledujúcej vrstvy a v dôsledku toho spracovávané odlišne.



Obr. 4.3: *Demonštrácia redundantnosti reprezentácie na skrytej vrstve doprednej siete.*
Sieť typu 2-4-2 so sigmoidálnymi aktivačnými funkciami aproximuje 1-bitové binárne sčítanie s prenosom ($y_1 = x_1 \operatorname{xor} x_2,\ \ y_2 = x_1 \operatorname{and} x_2$). Sieť funguje správne aj v prípade, keď sa aktivácia ktoréhokoľvek neurónu skrytej vrstvy nastaví na konštantnú 0 (infimum aktivačnej funkcie).
Stĺpce diagramov obsahujú po rade: vstupný vektor, aktivačný vektor skrytej vrstvy, aktivačný vektor výstupnej vrstvy, očakávaný výstup. Plocha štvorcov je úmerná aktivácii.

4.3 Architektúry, vhodné na implementovanie výpočtov nad konceptuálnym priestorom

Na tomto mieste uvádzame voľný popis možných oblastí aplikácie najznámejších konekcionistických architektúr:

- *FFN* (feed forward network) je vhodná na aproximáciu „slušných“ (spojité, spojitá prvá derivácia) generických zobrazení $\mathbb{R}^N \to \mathbb{R}^M$ a teda aj zobrazení nad CS

- *RAAM* (recursive auto-associative memory), je vhodná na asociovanie párov, konečných postupností a stromov bodov z $\mathbb{R}^N$, dokáže vytvárať vnútornú reprezentáciu týchto štruktúr. Po vhodnom zakódovaní jej možno predkladať postupnosti a stromy symbolov nad konečnou abecedou. Je použiteľná aj napr. na systematické priradenie taxonomických kategórií bodom (oblastiam) priestoru a pod.

- *SRN* (simple recurrent network) je jednoduchšia ako RAAM. Pri tejto architektúre pracuje rekurentne len kodér. Dekodér tu iba transformuje kontextový vektor na výstupný. Architektúra je vhodná na klasifikovanie symbolických postupností, predikčné modely (na výstupe sa v čase t nachádza odhad nasledujúceho znaku, ktorý by mal byť predložený) jednoduchých bezkontextových jazykov. V našej práci takúto sieť používame na mieste interpretačnej funkcie jazykových výrazov. Interpretujeme pritom do konceptuálneho priestoru.

- *Hopfieldov model* autoasociatívnej pamäte je dynamický systém, ktorý je v režime s atraktorovou dynamikou vhodný na asociatívne adresovanie konečného počtu vektorov z $\{0,1\}^N$ (po zakódovaní aj $\mathbb{R}^M$). Okrem toho je možné ho použiť vo verzii s „time-delayed“ váhami ako generátor postupnosti vektorov. Dá sa predstaviť jeho aplikácia najmä pri manipuláciách s prototypmi a teste príslušnosti objektu k prototypu, teda pri prepájaní symbolickej a konceptuálnej úrovne reprezentácie (implementácia interpretačnej a denominačnej funkcie).

- Kohonenov *SOM* (self-organizing map) redukuje dimenzionalitu dát – realizuje zobrazenie $\mathbb{R}^N \to \mathbb{R}^M$, kde $M \ll N$ (zvyčajne sa používa organizácia laterálnej vrstvy do dvojrozmernej mriežky, vtedy $M = 2$), pričom zachováva nelineárne topologické a metrické vlastnosti dát. Dá sa použiť ako klastrovací algoritmus. Jeho aplikácia môže byť zaujímavá napr. pri implementácii denominačnej funkcie.

Kapitola 5

Súčasný stav výskumu v oblasti a aplikácie

5.1 Kognitívna architektúra pre počítačové videnie

Traja talianski autori – Chella, Frixione a Gaglio (1997) navrhli elegantnú aplikáciu paradigmy konceptuálneho priestoru na problém rozpoznávania scény, najmä najťažších fáz – počnúc segmentáciou obrazu. Systém spadá do triedy architektúr modelujúcich scénu pomocou geometrických primitív. Keďže ide o peknú ukážku využitia vlastností parametrického priestoru primitív a keďže by architektúra mohla poslúžiť ako základ na experimenty v oblasti mobilnej robotiky, popíšeme ju v krátkosti v nasledujúcich odsekoch.

5.1.1 Geometrické primitíva

Ako základné geometrické primitíva zvolili autori tzv. *superkvadriky*. Ich parametrický tvar je:

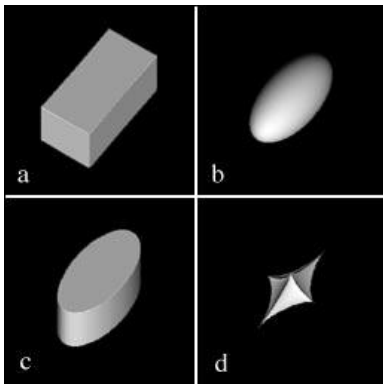
$$f(\eta,\omega) = \begin{bmatrix} a_x \cos^{\varepsilon_1}\eta \cos^{\varepsilon_2}\omega \\ a_y \cos^{\varepsilon_1}\eta \sin^{\varepsilon_2}\omega \\ a_z \sin^{\varepsilon_1}\eta \end{bmatrix} \qquad \begin{matrix} -\pi/2 \le \eta \le \pi/2 \\ -\pi \le \omega \le \pi \end{matrix}$$

Konštanty a_x, a_y, a_z definujú dĺžky osí a $\varepsilon_1, \varepsilon_2$ tvar superkvadriky pozdĺž dvoch osí. Pokiaľ je $\varepsilon_i \ll 1$, superkvadrika má v danom smere pravouhlý tvar, pre $\varepsilon_i \approx 1$ oblý tvar a pre $\varepsilon_i \gg 1$ zahrotený tvar. Príklady vizualizovaných superkvadrík sú na obr. 5.1.

Ak chceme superkvadrikou aproximovať teleso v 3D priestore s generickými rozmermi (a_x, a_y, a_z), tvarom $(\varepsilon_1, \varepsilon_2)$, , polohou (p_x, p_y, p_z) a orientáciou $(\varphi, \vartheta, \psi)$, dostaneme nasledovný vektor z $\mathbb{R}^{11}$:

$$[a_x, a_y, a_z, \varepsilon_1, \varepsilon_2, p_x, p_y, p_z, \varphi, \vartheta, \psi]^T = k$$

Konceptuálny priestor bude teda v prípade takejto reprezentácie tvorený parametrami superkvadrík, každý knoxel v ňom určuje jedno primitívne teleso scény.



Obr. 5.1: Príklady tvarov superkvadriky v závislosti od parametrov $(\varepsilon_1, \varepsilon_2)$.
$a : (0.01, 0.01),\ b : (1, 1),\ c : (0.01, 1),\ d : (5, 5)$

Na definovanie vzdialenosti knoxelov možno v tomto prípade použiť Euklidovskú metriku:

$$d(k, k') = ||k - k'|| = \sqrt{\sum_{i=1}^{n} (k_i - k'_i)^2}$$

Veľkou výhodou uvedenej reprezentácie je, že prirodzené prototypy, v tomto prípade pre človeka „prirodzené“ geometrické tvary – kváder, valec, guľa a pod. spĺňajú „Kritérium P“ – tvoria konvexné množiny v konceptuálnom priestore.

5.1.2 Reprezentácia zložených objektov

Aby bol systém počítačového videnia prakticky použiteľný, mal by byť schopný rozpoznať a klasifikovať aj komplexnejšie objekty na scéne. V tomto prípade ide o klasifikáciu objektov, ktorých prototypy sú zložené z viacerých superkvadrických primitív. Na tento účel autori rozšírili model o pojem konečnej množiny knoxelov, ktorú nazývajú *percepčný klaster*:

$$pc = \{k_1, k_2, \dots\ k_l\}$$

Množina všetkých percepčných klastrov nad CS je definovaná:

$$PC = \{\{k_1, k_2, \dots\ k_l\} \mid l \in \mathbb{N}, k_i \in CS; 1 \leq i \leq l\}$$

Percepčný klaster autori postavili ako základnú sémantickú jednotku, na ktorú sa mapujú lingvistické konštanty. Ide tu o dôležitý posun oproti Gärdenforsovmu modelu, ktorý predpokladal interpretačnú funkciu, mapujúcu konštanty priamo na knoxely v CS. Zavedením množiny knoxelov ako sémantickej jednotky sa však dá vyhnúť pribúdaniu dimenzií CS pri potrebe reprezentovať komplexnejšie objekty. Na druhej strane, reprezentácia množiny vektorov patrí medzi ťažké problémy pre neurónové siete.

5.1.3 Lingvistická úroveň a interpretácia symbolov

Na lingvistickej úrovni modelu autori použili predikátovú logiku prvého rádu a kompletne monotónnu logiku.

Interpretačná funkcia zobrazuje základné termy (konštanty) jazyka na percepčné klastre:

$$\Phi : L^{const} \rightarrow PC$$

a predikáty na relácie nad percepčnými klastrami:

$$\Phi : L^{n-ary\_pred} \rightarrow PC^n$$

Aj tu možno s výhodou využiť štruktúru sémantickej domény na definovanie implicitných predikátov. V prípade percepčných klastrov je takouto implicitnou vlastnosť „byť časť“ – `has-part(Const1, Const2)`. Pokiaľ by sme interpretovali realistickou sémantikou, bola by pravdivosť uvedeného predikátu rozhodnuteľná len na základe extenzie:

$$\texttt{has-part(Const1, Const2)}$$
$$\Leftrightarrow$$
$$\texttt{(}\Phi\texttt{(Const1), }\Phi\texttt{(Const2))} \in \Phi\texttt{(has-part)}$$

V zavedenej sémantike možno na interpretáciu atómu využiť priamo percepčné klastre, na ktoré sú interpretované konštanty:

$$\texttt{has-part(Const1, Const2)} \Leftrightarrow \Phi\texttt{(Const2)} \subseteq \Phi\texttt{(Const1)}$$

5.1.4 Zameriavanie pozornosti a generovanie asercií

Na nižších úrovniach spracovania obrazu, kde všetky informácie majú rovnakú dôležitosť, možno s výhodou použiť masívny paralelizmus (predspracovanie – transformácie obrazovej matice, detekcia hrán, ...). Vyššie úrovne pracujú s dátami, ktoré sú navzájom previazané a majú rôznu dôležitosť. Vyžadujú skôr sekvenčné spracovanie a z časových a pamäťových dôvodov je nevyhnutné z rôznych alternatívnych postupov výpočtu zvoliť čo najvhodnejší. Do popredia sa dostáva pojem *zaostrenia pozornosti*.

Definujeme CS^* ako množinu všetkých konečných postupností knoxelov z CS:

$$CS^* = \{k_{i_1}k_{i_2}\ldots k_{i_l} \mid l, i_j \in \mathbb{N}; k_{i_j} \in CS\}$$

a *percepčný akt* ako generickú postupnosť knoxelov z CS:

$$k_1k_2\ldots k_l = p \in CS^*$$

Hovoríme, že percepčný akt p je *asociovaný* s percepčným klastrom pc ak p je ľubovolnou postupnosťou ľubovolného počtu bodov z pc. p nemusí obsahovať všetky knoxely z pc a každý knoxel môže obsahovať ľubovolný počet krát.

Na zakladanie nových asercií na základe úspešných percepčných aktov slúži *denominačná funkcia*, ktorá tvorí jadro popisovanej architektúry. Je definovaná ako:

$$\Theta : CS^* \rightarrow L^{FOL}$$

kde L^{FOL} je jazyk prvorádovej logiky. V prípade úspešného percepčného aktu spôsobí volanie funkcie $\Theta(p)$ založenie novej konštanty pre objekt (inštanciu konceptu) reprezentovaný percepčným aktom p a vráti aserciu obsahujúcu túto novú konštantu. Napr.:

$$\Theta(k_1 k_2) = \texttt{Hammer(Hammer\#1)}$$

Popisovaná architektúra používa tri režimy zaostrovania pozornosti:

- *reaktívny* režim. Využíva len samotnú prítomnosť objektov na scéne s preferovaním blízkych objektov na exhaustívne generovanie percepčných aktov, ktorých počet je z toho dôvodu exponenciálny v závislosti od počtu objektov na scéne. Tento režim je najjednoduchší a používa sa len v prípade, že sa nedá aplikovať iný režim.

- *lingvistický* režim. Využíva symbolickú informáciu z lingvistickej úrovne. Pri zaostrení na určitý objekt `O_init` scény sa v báze znalostí hľadá vhodný komplexný objekt `O_cplx`, ktorého časťou objekt `O_init` môže byť. Pokiaľ taký existuje, vygeneruje sa percepčný akt p_{cplx}, overujúci prítomnosť celého objektu `O_cplx` na scéne.

- *asociatívny* režim. Funguje rovnako ako lingvistický s tým rozdielom, že koncept, asociovaný s `O_init`, hľadá v pamäti voľných asociácií objektov. Táto pamäť sa učí Hebbovským učením počas práce systému – vždy keď sa na scéne spoločne vyskytnú inštancie ľubovolných dvoch konceptov, väzba medzi nimi sa zosilní.

5.1.5 Implementácia denominačnej a interpretačnej funkcie

Jadrom implementácie denominačnej a interpretačnej funkcie je Hopfieldov model autoasociatívnej pamäte (Hopfield (1982)). Autori používajú diskrétny model, hodnoty spojitých veličín vhodne kódujú (požiadavky na kódovanie vzorov pre Hopfieldovu sieť nájdeme napr. v [14]).

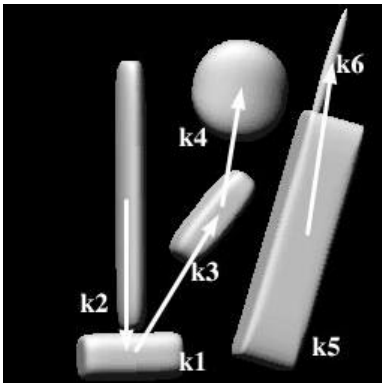
5.1.6 Nedostatky architektúry

- invariantnosť voči afinným transformáciám (všetky linérane tranformácie + všetky posunutia) komplexných objektov (percepčných klastrov). Tomuto problému sa autori v [5] venujú len úplne náznakovo, hoci ide o jeden z najťažších problémov pri analýze reálnej scény. Príklady, ako možno pomocou Hopfieldovej siete invariantne rozoznávať jednoduché vzory nájdeme napr. v [14].

- dynamická scéna. Aby bola architektúra počítačového videnia použiteľná napr. ako dominantný senzorický systém mobilného robota, je potrebné aby okrem priestorových relácií, bola schopná pracovať aj s časovými reláciami nad objektami scény.

- určitá prebytočnosť použitia Hopfieldovho modelu. Aplikácia siete by sa v tomto prípade možno dala nahradiť priamymi analytickými výpočtami (založenými na vzdialenosti bodov). Na overenie tohto predpokladu by

Obr. 5.2: Komplexná scéna pozostávajúca z kladiva, telefónu, myši, a dreveného kvádra



Obr. 5.3: Rekonštrukcia komplexnej scény z obr. 5.2 spätnou vizualizáciou rozpoznaných superkvadrík. Šípky znázorňujú presun pozornosti pri analýze scény.

však bolo treba vykonať experimenty s dátami podobnej povahy, aké sa spracovávali v popisovanej architektúre.

```
Knoxel (#k1)
Knoxel (#k2)
Knoxel (#k3)
Knoxel (#k4)
Knoxel (#k5)
Knoxel (#k6)
Cylinder-shaped(#k2)
Box-shaped(#k1)
Hammer (Hammer#1)
has-handle(Hammer#1,#k2)
has-head(Hammer#1,#k1)
Box-shaped(#k3)
Block(Block#1)
has-part(Block#1,#k3)
Next-to(Next-to#1)
participant(Next-to#1,Hammer#1)
participant(Next-to#1,Block#1)
Ellipsoid-shaped(#k4)
Mouse(Mouse#1)
has-part(Mouse#1,#k4)
Above (Above#1)
is-above(Above#1,Mouse#1)
is-below(Above#1,Block#1)
Parallelepiped-shaped(#k5)
Thin-cylinder-shaped(#k6)
Telephone(Telephone#1)
has-body(Telephone#1,k#5)
has-antenna(Telephone#1,#k6)
```

Obr. 5.4: Symbolická reprezentácia scény z obr. 5.2, vygenerovaná popisovaným systémom na rozpoznávanie obrazu

Kapitola 6

Experimenty

6.1 Experimenty s Marrovou reprezentáciou

V snahe experimentovať s konceptuálnou reprezentáciou nejakých konkrétnych objektov, siahli sme po reprezentácii troj-rozmerných objektov pomocou geometrických primitív. Reprezentáciu navrhol Marr (1982) [8]. Valec, použitý v tejto reprezentácii ako primitívum je definovaný dĺžkou, priemerom (l, d), polohou (p_x, p_y, p_z) a orientáciou $(\alpha_x, \alpha_y, \alpha_z)$. Parametre tvoria priestor $E_{Marr} = \mathbb{R}^8$:

$$[l, d, p_x, p_y, p_z, \alpha_x, \alpha_y, \alpha_z]^T = k$$

Ukážky vizualizovaných zložených objektov sú na obr. 6.2, pôvodný náčrt reprezentácie je na obr. 6.1.

Nevýhoda tejto reprezentácie je, že neposkytuje veľa kandidátov na prirodzené koncepty v zmysle „Kritéria P“. Konvexné množiny tvoria v priestore E_{Marr} len koncepty typu „Malý objekt“, „Tenký objekt“, „Vodorovný objekt“ apod. Výhodnejšie by boli superkvadriky, popísané v odseku 5.1.1, v parametrickom priestore ktorých možno demonštrovať viac prirodzených konceptov. Napriek tomu sme nad týmto priestorom zrealizovali niekoľko experimentov, ktoré možno rozdeliť zhruba do nasledovných kategórií:

- *autoasociatívna reprezentácia primitív pomocou FFN.* Použili sme sieť so symetrickou topológiou (8-c-8) a učením smerom k autoasociatívnej schopnosti, podobne ako pri RAAM-e. Dá sa ukázať, že výsledná reprezentácia (pri výraznej redukcii dimenzie) zachováva topologické vlastnosti priestoru primitív. Môžeme to analyzovať napr. rozklastrovaním priestoru E_{Marr} zjednotenia množín primitív vhodného počtu komplexných objektov pomocou Kohonenovho SOM-u. Keď potom takým istým spôsobom rozklastrujeme redukovanú reprezentáciu na kontextovej (skrytej) vrstve FFN, autoasociatívne reprezentujúcej primitíva, môžeme pozorovať vznik podobnej mapy. To je však dosť očakávateľný výsledok.

- *extrakcia jednoduchých priestorových relácií* („blízko“, „nad“, „pod“, ...) nad pármi primitív pomocou FFN. Jeden demonštračný príklad uvádzame v nasledovnom odseku.

- *reprezentácia postupnosti primitív pomocou RAAM-u.*

Tieto experimenty, ani ich výsledky, nebudeme v práci rozoberať, nakoľko ich nepokladáme za významné a jadro práce tvoria iné experimenty. Počas experimentov sa vynorili niektoré pozorovania, ktoré možno zhrnúť nasledovne:

- silná potreba rozdeliť architektúru, spracúvajúcu informácie z vizuálneho senzorického systému na bloky „Čo“ a „Kde“. Bez tohto nie je možné zvládať invariantné narábanie s objektami.
- spracovanie (klasifikácia, reprezentácia) konečnej množiny bodov $A \subseteq \mathbb{R}^n$, ktoré by sa žiadalo na prácu s geometrickými primitívami nie je pomocou štandardných konekcionistických modelov dobre možná. Rekurentné siete zvládnu pracovať s konečnou postupnosťou bodov $\vec{k}_1\vec{k}_2\ldots\vec{k}_n;\ \vec{k}_i \in \mathbb{R}^n$, nie však s množinou. Hopfieldov model síce dokáže konvergovať k najbližšiemu z množiny atraktorov, avšak informáciu, že sa jeho stav nachádza v blízkosti nejakého vzoru je možné získavať jedine externe, na základe globálnej veličiny, zvanej *prekryv* stavu s daným vzorom. Túto veličinu treba rátať analyticky, a keď raz pripustíme takúto cestu, môžeme od použitia neurónovej siete už upustiť úplne. Otázne je, či vôbec má zmysel požadovať od konekcionistickej platformy takúto funkciu.

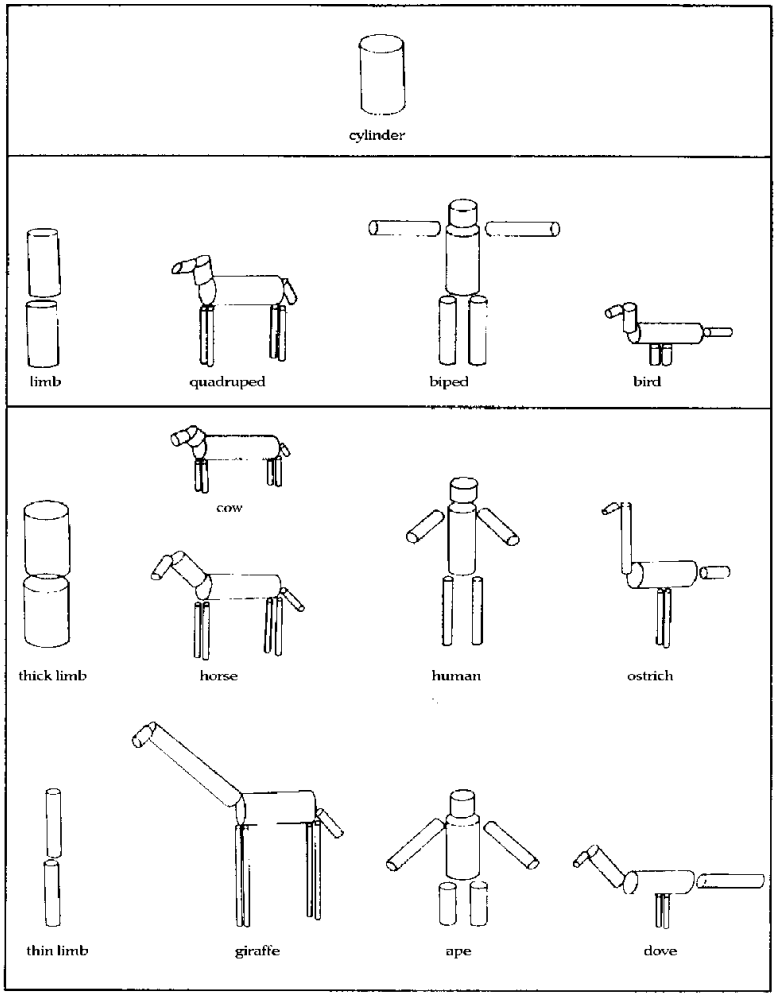
Okrem toho je zrejme rozumné poučiť sa v tomto prípade z organizácie zrakovej kôry a modelovať aj tieto fázy rozpoznávania scény pomocou celulárnych sietí (CNN) s Hebbovským učením. Vhodnú inšpiráciu na takéto modelovanie nájdeme v [10], kde autori popisujú platformu, implementujúcu práve jednoduché priestorové relácie objektov pomocou CNN (v tomto prípade zovšeobecnený Hopfieldov model - okrem stavu každá bunka obsahuje aj fázový uhol. Ovplyvňovanie sa buniek navzájom potom závisí okrem stavov aj od miery zhodnosti ich fázových uhlov. Tak je možné modelovať stav, kedy vlastnosti rovnakého objektu scény reprezentujú grafy buniek s rovnakým fázovým uhlom a celé to má pomerne vtipné vlastnosti.).

6.1.1 Reprezentovanie priestorovej relácie nad CS doprednou sieťou

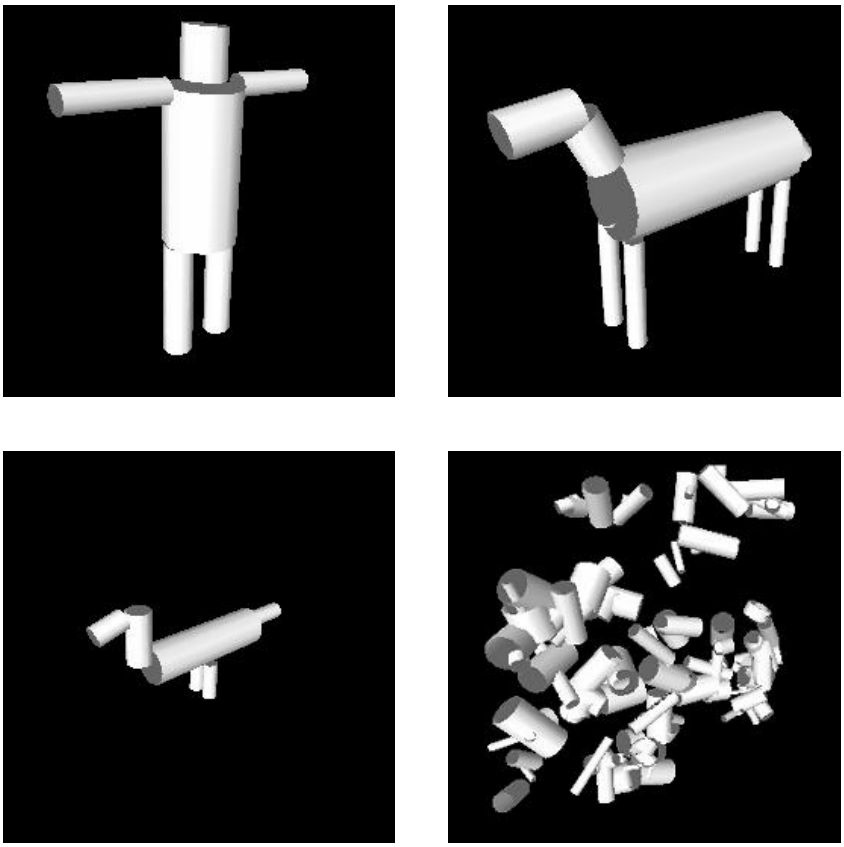
Tento príklad uvádzame len čisto na demonštráciu. Majme množinu parametrických vektorov Marrových primitív $K = \{k_i\}$ a funkciu:

$$Near(k_1, k_2) = \begin{cases} 0 & \text{ak } d(k_1^{[p_x,p_y,p_z]}, k_2^{[p_x,p_y,p_z]}) > c \\ 1 & \text{ak } d(k_1^{[p_x,p_y,p_z]}, k_2^{[p_x,p_y,p_z]}) \leq c \end{cases}$$

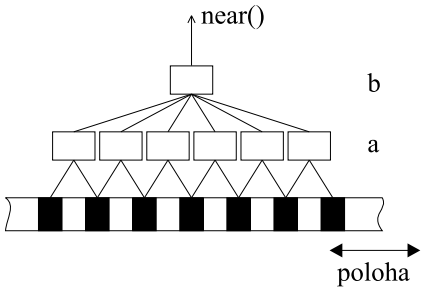
Ak predkladáme sieti s topológiou (8 + 8)-h-1 na vstup páry parametrických vektorov a na výstupe očakávame hodnotu $Near(k_1, k_2)$, je schopná pri dostatočnej veľkosti trénovacej množiny aproximovať priestorovú reláciu (obr. 6.4, 6.5). Podobné výsledky dostaneme aj pre ostatné jednoduché relácie. Nejde tu o ťažko aproximovateľnú závislosť, sieť však musí „prísť na to“, ktoré dimenzie (zložky) vektorov sú pre reláciu relevantné a adaptovať váhy tak, aby výsledok nebol ovplyvňovaný irelevantnými zložkami. Ak je však príkladov dosť, sieť sa ľahko adaptuje. Príklad uvádzame aj preto, že pri CNN prístupoch k modelovaniu relácií treba na rozpoznávanie inštancií takýchto relácií hierarchickú štruktúru „spolupracujúcich“ neurónov (obr. 6.3) a ich učenie nie je ľahké.



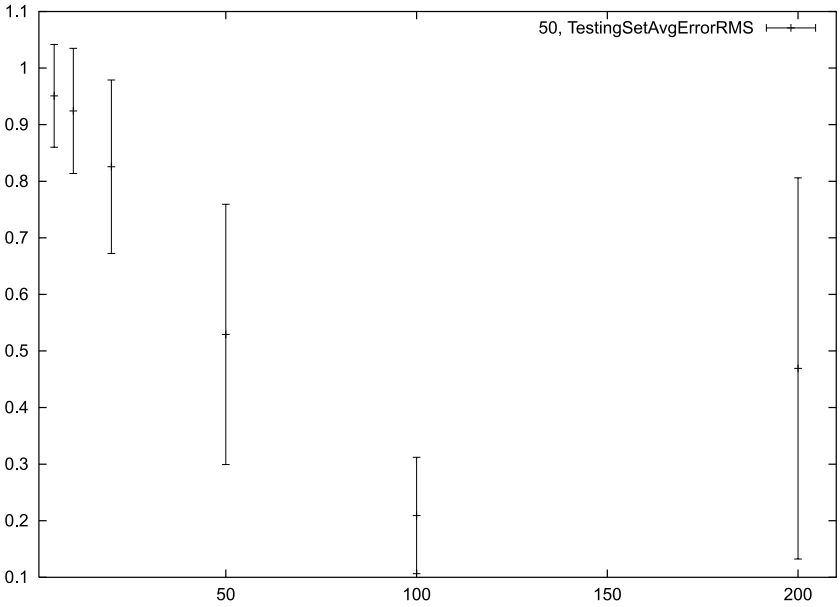
Obr. 6.1: Reprezentácia živých objektov pomocou valca ako geometrického primitíva. (Marr (1982))



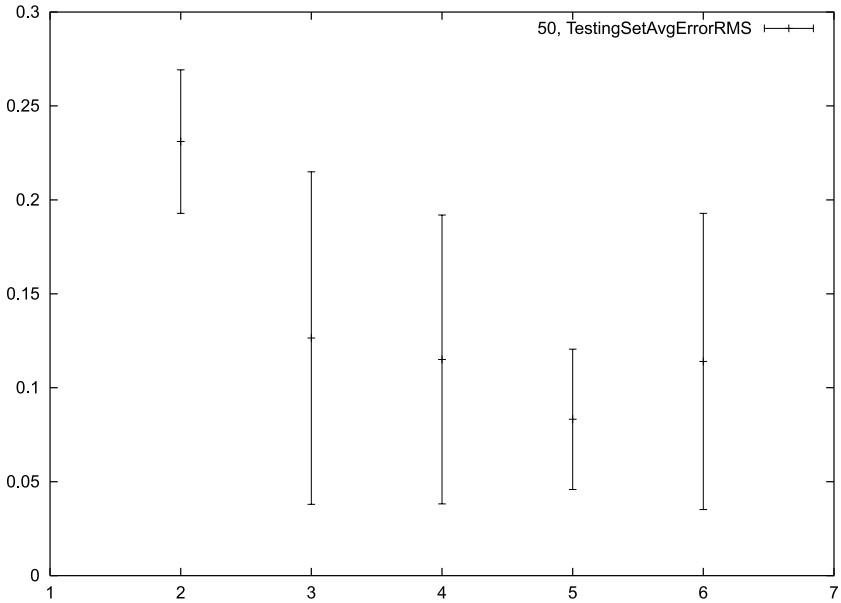
Obr. 6.2: Marrova reprezentácia. Objekty zložené z valcov, reprezentovaných, ako vektory v 8 rozmernom priestore. Na editáciu a predkladanie sieťam sme si pripravili jednoduchý testbed, vizualizujúci pomocou OpenGL.



Obr. 6.3: Reprezentácia priestorovej relácie Near(x,y) v architektúre s celulárnymi neurónmi (CNN). Výstupy absolútnych korelačných buniek (a) treba skombinovať v relačnej bunke (b).



Obr. 6.4: Závislosť chyby testovacej množiny od veľkosti trénovacej množiny. V súlade s očakávaním treba veľa príkladov párov objektov, aby sa vynorila korektná aproximácia relácie.



Obr. 6.5: Závislosť chyby testovacej množiny od počtu skrytých neurónov. Na uspokojivú aproximáciu stačia už 3 neuróny.

6.2 Emergencia koordinovaného slovníka nad CS

Nasledujúca séria experimentov sa týka prepojenia lingvistickej a konceptuálnej reprezentácie. Uvažujeme v ňom agentov, *disponujúcich konceptuálnou reprezentáciou* určitých objektov. Cieľom je dosiahnuť schopnosť koordinovanej komunikácie v populácii takýchto agentov. Komunikácia by mala prebiehať pomocou konečných sekvencií znakov nad konečnou abecedou a mala by umožniť čo najpresnejší prenos konceptuálnej reprezentácie.

Takýto agentový model filozoficky vychádza z pohľadu na jazyk ako na *dynamický komplexný systém*. Jazyk je chápaný ako emergentný fenomén, vznikajúci samoorganizáciou dynamického komplexného systému interagujúcich agentov. Agenti jazyk používajú aj spoluvytvárajú zároveň. Kľúčová je tu dynamika.

Steels [21] uvádza najdôležitejšie kritériá, ktoré by mal spĺňať realistický systém, modelujúci vývoj jazyka týmto spôsobom:

- *distribuovanosť* a *lokálnosť*. Každý agent by mal v prostredí vnímať a ovplyvňovať len svoje lokálne okolie. Týka sa to aj jazyka - žiaden agent by nemal mať globálny prehľad o celom jazyku a na ďalší vývoj jazykových konvencií v populácii by mal mať len nepriamy vplyv prostredníctvom svojho okolia.

- *obmedzená racionalita*. Žiaden agent by nemal mať priamy prístup k vnútornému stavu iného agenta. Poznávanie a ovplyvňovanie vnútorného stavu iných musí prebiehať prostredníctvom jazyka.

- *otvorenosť*. Populácia agentov by nemala byť statická. Malo by dochádzať k úbytku a prírastku nových agentov. To isté sa týka aj významov (objektov, situácií), ktoré sú predmetom jazykovej komunikácie. Tieto fluktuácie sú dôležité, nakoľko predstavujú energiu, vychyľujúcu dynamický systém z rovnovážneho stavu. To zabraňuje zotrvávaniu v triviálnych atraktorových stavoch a podporuje samoorganizáciu, smerujúcu k rastu komplexnosti. Z teórie dynamických systémov je známe, že k najzaujímavejším javom dochádza práve „na hrane chaosu“.

6.2.1 Predpoklady

My vychádzame z modelu, navrhnutého Batalim [2]. Predpokladáme, že agent disponuje „artikulačným“ a „sluchovým“ aparátom, umožňujúcim bezchybnú komunikáciu s iným agentom (vyslanie, prijatie signálu vo forme postupnosti znakov z konečnej abecedy Σ). Pre celú populáciu predpokladáme pre jednoduchosť jednotnú konceptuálnu reprezentáciu vo forme vektorov z $(0,1)^d$. Agent je ďalej vybavený kognitívnym orgánom, implementujúcim denominačnú (F_{den}) a interpretačnú (F_{int}) funkciu. F_{den} zobrazuje konceptuálnu reprezentáciu objektu (meaning vektor) na reťazec znakov a F_{int} realizuje inverzné zobrazenie:

$$F_{den} : CS \rightarrow \Sigma^*$$

$$F_{int} : \Sigma^* \rightarrow CS$$

Perspektívny námet na realizáciu týchto zobrazení ponúkajú rekurentné neurónové siete. Modely, spracovávajúce štruktúrovanú symbolickú informáciu

```
function TSaussureAgent.InterpretationFunction(
  input: ASequence; output: AMeaningVector
  )
{
  LContextBuffer := BlankContext;
  foreach char in ASequence do {
    NN.Layers[Input].ContextInput := LContextBuffer;
    NN.Layers[Input].CharInput := CharToVector(char);

    NN.FeedForwadActivations;

    LContextBuffer := NN.Layers[Hidden].Activation;
    }
  AMeaningVector := RNN.Layers[Output].Activation;
}
```

Obr. 6.6: Pseudokód interpretačnej funkcie.

(RAAM, SRN) sú známe relatívne dlho [17]. Experimenty však boli zvyčajne realizované nad pevnou trénovacou množinou a navyše takmer vždy smerom od symbolických dát k distribuovanej reprezentácii (sieť klasifikuje alebo reprezentuje človekom vhodne zvolenú množinu symbolických štruktúr - reťazcov alebo stromov). Z hľadiska modelovania evolúcie jazykových schopností je však zaujímavejší opačný prístup, ktorý využíva aj popisovaný model.

6.2.2 Interpretácia symbolov

Batali implementoval interpretačnú funkciu agenta pomocou jednoduchej rekurentnej siete (obr. 6.7). Sieť možno chápať ako zobrazenie $\Phi = \Phi_{ih} \circ \Phi_{ho}$. Majme postupnosť znakov $x_1 x_2 \dots x_n$ a príslušnú postupnosť vektorov $v_{x_1} v_{x_2} \dots v_{x_n}$. Nasledujúca definícia špecifikuje interpretačnú funkciu. Prvé dva riadky induktívne definujú prácu siete a posledný ukazuje, ako interpretačná funkcia „zapúzdruje“ sieť.

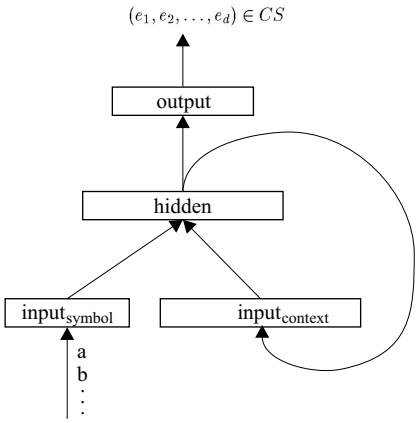
$$\begin{aligned} c_1 &= \Phi_{ih}(v_{x_1}, c_0) \\ c_{t+1} &= \Phi_{ih}(vx_{t+1}, c_t) \\ F_{int}(x_1 x_2 \dots x_n) &= \Phi_{ho}(c_n) \end{aligned}$$

Ide tu o štandardné použitie jednoduchej rekurentnej siete na spracovanie reťazca symbolov. Pseudokód nájdeme na obr. 6.6.

6.2.3 Produkcia symbolov

Zaujímavejšia je realizácia denominačnej funkcie. Na tvorbu symbolického výrazu Batali prijíma tzv. *Saussureanovskú*[1] konvenciu, ktorá predpokladá, že agent pri *generovaní* signálu využíva spätnú väzbu zo svojho vlastného aparátu

[1] Ferdinand de Saussure (1857-1913), lingvistik, zakladateľ lingvistického štrukturalizmu



Obr. 6.7: Jednoduchá rekurentná sieť, implementujúca interpretačnú funkciu.

na interpretáciu signálu. Metaforicky možno povedať, že agent sa pri „rozprávaní“ zároveň „počúva“ a snaží sa „rozprávať“ tak, aby si sám čo najlepšie „rozumel“. Iní autori už experimentálne ukázali, že táto vlastnosť je evolučne výhodná a môže v evolúcii emergovať zo slabších predpokladov [21].

V prípade popisovaného modelu je Saussureanovský predpoklad „natvrdo zabudovaný“ do definície denominačnej funkcie. Pri odosielaní sa totiž reťazec reprezentujúci $k_{meaning}$ buduje po znakoch, pričom každý ďalší znak je zvolený tak, aby doteraz vygenerovaný reťazec pri spätnom interpretovaní tým istým agentom dával význam čo najpodobnejší odosielanému významu $k_{meaning}$. Pseudokód popísanej Saussureanovskej denominačnej funkcie uvádzame na obr. 6.8. Formálna definícia je nasledovná:

$$F_{den}(k_{meaning}) = x_1 x_2 \ldots x_n,$$

$$x_t = \begin{cases} \arg\min_{\hat{x}\in\Sigma} d(F_{int}(\hat{x}), k_{meaning}) & \text{pre } t = 1, \\ \arg\min_{\hat{x}\in\Sigma} d(F_{int}(x_1 x_2 \ldots \hat{x}), k_{meaning}) & \text{pre } t > 1 \end{cases}$$

Koncept, ktorý dostaneme zo zdrojového konceptu k transformáciou „cez jazyk“ (kompozícia transformácie na symbolický výraz a spätnej interpretácie), budeme ďalej označovať ako $k' = F_{int} \circ F_{den}(k)$. Keďže máme záujem o jazyk s konečnými reťazcami, je potrebné definovať *podmienku ukončenia* generovania. Na tento účel používame dve ohraničujúce konštanty. $MaxDist$ udáva maximálnu euklidovskú vzdialenosť medzi konceptami k a k', pre ktorú sa ešte $F_{int} \circ F_{den}$ považuje za *korektné* vzhľadom na koncept k. $MaxLength$ udáva maximálnu prípustnú dĺžku slova. Denominačná funkcia teda generuje znaky, pokiaľ výsledný koncept k' nie je dostatočne podobný zdrojovému, alebo sa dosiahol limit pre dĺžku slova. Formálne vyzerá podmienka ukončenia generovania nasledovne:

```
function TSaussureAgent.DenominationFunction(
  input: AMeaningVector; output: ASequence
  )
{
  LContextBuffer := BlankContext;
  ASequence := '';

  repeat
    LMinDistance := +INFINITY;

    foreach char in Alphabet do {
      NN.Layers[Input].ContextInput := LContextBuffer;
      NN.Layers[Input].CharInput := CharToVector(char);

      NN.FeedForwadActivations;

      LCurrentCharContext := NN.Layers[Hidden].Activation;
      LCurrentCharMeaningVector := NN.Layers[Output].Activation;
      if d(LCurrentCharMeaningVector, AMeaningVector) < LMinDistance {
        LMinDistance := d(LCurrentCharMeaningVector, AMeaningVector);
        LMinDistanceChar := char;
        LMinDistanceContext := LCurrentCharContext;
        }
      }

    ASequence := ASequence + LMinDistanceChar;
    LContextBuffer := LMinDistanceContext;
  until StopCondition(AMeaningVector, ASequence);
}
```

Obr. 6.8: Pseudokód denominačnej funkcie.

$$p_{stop\_condition}(k, x_1x_2\ldots x_n) \equiv d(k, F_{int}(x_1x_2\ldots x_n)) \leq MaxDist \vee |x_1x_2\ldots x_n| >= MaxLength$$

6.2.4 Abeceda

Používame štvorprvkovú abecedu $\Sigma = \{a, b, c, d\}$ a štandardné ortonormálne kódovanie symbolov metódou jeden z n ($v_a = (0, 0, 0, 1), v_b = (0, 0, 1, 0), \ldots, v_d = (1, 0, 0, 0)$). Stačí síce už dvojprvková abeceda, je to však na úkor rýchlosti a spoľahlivosti konvergencie (sieť sa musí učiť „nelineárnejšie“ zobrazenie). Keďže požívame najmenej 3-rozmerný konceptuálny priestor, sieť nemôže adaptovať „naivné“ riešenie, kde každý znak abecedy posúva výstupnú aktiváciu pozdĺž jednej z dimenzií k infimu alebo supremu aktivačnej funkcie.

6.2.5 Simulácia

Na úvod je založená populácia agentov, ktorých neurónové siete obsahujú náhodne vygenerované váhové (a prahové) koeficienty (uniformne, z intervalu $(-0.5, 0.5)$). Príklad slovníka, generovaného takýmto agentom, je na obr. 6.11. Distribúcia chýb komunikačných epizód (pojem definujeme ďalej) v úvodnej populácii je na obr. 6.27a.

Interakcia medzi agentami v populácii prebieha v *trénovacích kolách*, počas ktorých agenti medzi sebou komunikujú formou *komunikačných epizód*.

Počas komunikačnej epizódy:

- rozprávajúci vygeneruje meaning vektor a pretransformuje ho (F_{den}) na postupnosť znakov
- postupnosť znakov vyšle počúvajúcemu
- počúvajúci postupnosť príjme a pretransformuje späť na meaning vektor (F_{int})

Trénovacie kolo potom pozostáva z nasledovných krokov:

- z populácie je náhodne vybraný počúvajúci
- k nemu sú rovnako náhodne vyberaný rozprávajúci, rôzni od počúvajúceho. Každý pár podstúpi určitý počet komunikačných epizód.

Pseudokód trénovacieho kola uvádzame na obr. 6.9.

6.2.6 Adaptácia interpretačnej funkcie

Jadrom simulácie je adaptácia interpretačnej funkcie tak, aby sa zmenšovala chyba interpretácie. Na tento účel používame štandardné učenie s učiteľom metódou spätného šírenia chybového gradientu (Backpropagation, Rumelhart et al. (1986)). Adaptácia je implementovaná priamo v tele interpretačnej funkcie. Upravený pseudokód je na obr. 6.10.

Kľúčová požiadavka na povahu adaptácie interpretačnej funkcie pritom vyplýva zo Saussureanovskej konštrukcie denominačnej funkcie. Denominačná funkcia, ako sme popisovali vyššie, volí každý ďalší znak a_{t+1} na odoslanie výhradne na základe kritéria minimálnej vzdialenosti $d(k'_{t+1}, k)$ výstupnej aktivácie v danom časovom kroku od odosielaného meaning vektora. Ak teda má denominačná funkcia pracovať správne, je nevyhnutné aby trajektória v aktivačnom priestore výstupnej vrstvy konvergovala k cieľovému meaning vektoru monotónne po celý čas interpretácie - teda nie až ku koncu sekvencie. Táto skutočnosť prakticky vylučuje použitie učiaceho algoritmu, pracujúceho na sieti, rozvinutej v čase (BPTT), aj keď tento je na učenie jednoduchých rekurentných sietí vhodnejší.

Čo sa týka modifikácie váh po spätnom šírení chybových gradientov (.UpdateWeights() volanie v pseudokóde), nie je samozrejme nevyhnutné modifikáciu aplikovať ihneď. Namiesto toho môžeme vektory diferencií váh, získané delta pravidlom kumulovať a pripočítať k aktuálnemu vektoru váh siete až na konci komunikačnej epizódy (prípadne ešte neskôr). Dá sa tak očakávať hladší priebeh učenia (odstránia sa oscilácie vo váhovom priestore, ktoré môžu nastať v

```
function CommunicationEpisode(
  input: AAgentSpeaker, AAgentListener
  ) {

  LMeaning := AAgentSpeaker.GenerateMeaning();
  LSequence := AAgentSpeaker.DenominationFunction(LMeaning);
  LMeaning' := AAgentListener.InterpretationFunction(LSequence);
  }

function LearningRound() {
  LAgentListener := Population.RandomAgent();
  for i = 1 to CSpeakerPerListenerCount do {
    LAgentSpeaker := Population.RandomAgentOtherThan(LAgentListener);

    for j = 1 to CCommunicationEpisodePerAgentPairCount do
      CommunicationEpisode(LAgentSpeaker, LAgentListener);
    }
  }
```

Obr. 6.9: Pseudokód komunikačnej epizódy a trénovacieho kola

dôsledku „protirečivých“ zmien váh siete počas spracúvania sekvencie). V našich experimentoch nepoužívame momentový člen v delta pravidle a zmeny váh aplikujeme vždy na konci komunikačnej epizódy.

Keď si uvedomíme povahu popísanej interakcie v populácii agentov, je zrejmé, že konvergencia jazykovej evolúcie nie je ničím priamo zaručená. Na úvod simulácie agent, hrajúci úlohu počúvajúceho v trénovacom kole, „počuje“ od každého rozprávajúceho celkom odlišné pomenovania pre ten istý meaning vektor. Konvergencia (v prípade, že nastane) je teda skôr štatistickej povahy. Keďže výber agentov, účinkujúcich v trénovacom kole je stochastický, rovnako ako priradenie počiatočných váhových koeficientov sieťam, dochádza k drobným asymetriám v homogenite[2] párov (sekvencia, význam), ktoré počúvajúci interpretuje. V systéme ďalej existuje zrejmá pozitívna spätná väzba, zvyšujúca asymetriu v pomenovávaní meaning vektorov. Akonáhle totiž začne v populácii výraznejšie prevládať určitý trend v povahe sekvencií, priraďovaných jednému významu, sú agenti, keďže túto sekvenciu/vzor „počujú“ častejšie ako iné, vystavení adaptačnému tlaku, vedúcemu k ďalšiemu rozširovaniu tohoto vzoru. Táto pozitívna spätná väzba, pokiaľ sú použité siete dostatočne plastické, časom ústi do konvergencie k spoločnému slovníku.

6.2.7 Hodnotenie priebehu simulácie

Na posudzovanie miery konvergencie počas simulácie používame niekoľko veličín, ktoré si teraz popíšeme. Budeme ich nazývať anglickými názvami, ktoré sú stručnejšie a používame ich aj v legendách grafov.

[2]Nejde tu o homegenitu v zmysle, že by pomenovania jedného meaning vektora boli rovnaké. Ide skôr o to, nakoľko sú všetky zastúpené rovnomerne a nakoľko rovnomerne sa počúvajúcemu objavujú „na vstupe“ pri trénovacích kolách.

```
function TSaussureAgent.InterpretationFunctionWithLearnig(
  input: ASequence, AMeaningVectorTemplate;
  output: AMeaningVector
  )
{
  LContextBuffer := BlankContext;
  foreach char in ASequence do {
    NN.Layers[Input].ContextInput := LContextBuffer;
    NN.Layers[Input].CharInput := CharToVector(char);

    NN.FeedForwadActivations;

    LContextBuffer := NN.Layers[Hidden].Activation;

(*) Adaptation:
      LMeaningVector := NN.Layers[Output].Activation;
      NN.Layers[Output].Error := (AMeaningVectorTemplate - LMeaningVector);
      NN.ErrorBackPropagation();
      NN.UpdateWeights(CLearningRate, CMomentum);

    }
  AMeaningVector := NN.Layers[Output].Activation;
}
```

Obr. 6.10: Pseudokód interpretačnej funkcie s adaptáciou neurónovej siete. Učenie je v kóde označené symbolom (*).

Index	Sequence	Meaning	MeaningNN
0	cccccccccccccccccc		
1	bbbbbbbbbbbbbbbbbb		
2	bbbbbbbbbbbbbbbbbb		
3	bbbbbbbbbbbbbbbbbb		
4	cccccccccccccccccc		
5	bbbbbbbbbbbbbbbbbb		
6	aaaaaaaaaaaaaaaaaa		
7	dddddddddddddddddd		
8	dddddddddddddddddd		
9	aaaaaaaaaaaaaaaaaa		
10	bbbbbbbbbbbbbbbbbb		
11	cccccccccccccccccc		
12	cccccccccccccccccc		
13	cccccccccccccccccc		
14	dddddddddddddddddd		
15	bbbbbbbbbbbbbbbbbb		
16	dddddddddddddddddd		
17	bbbbbbbbbbbbbbbbbb		
18	ccaaaaaaaaaaaaaaaaa		
19	cccccccccccccccccc		

Obr. 6.11: Slovník, generovaný jedným z agentov inicializačnej populácie

ErrorRMS: priemerná chyba komunikačného aktu dvoch rôznych agentov, založená na vzdialenosti konceptov k a $k' = F_{int} \circ F_{den}(k)$:

$$E_{RMS} = \frac{\sum_i d(k_i, k'_i)}{\sum_i d(k_i, \langle k \rangle)}; \quad \langle k \rangle = \frac{1}{n} \sum_{i=1}^{n} k_i$$

SelfErrorRMS: ako predošlá, udáva však mieru toho, ako priemerný agent „rozumie sám sebe“ (chyba interpretácie reťazcov, generovaných ním samým). Pritom znovu zdôrazňujeme, že agent sa v trénovacích kolách na interpretáciu ním samým vygenerovaných pomenovaní neadaptuje.

Distance: priemer euklidovských vzdialeností (ako ErrorRMS, avšak bez normalizácie):

$$E_{distance} = \frac{1}{n} \sum_i d(k_i, k'_i)$$

Correctness: relatívny počet úspešných komunikačných epizód. Za úspešnú sa považuje epizóda, pri ktorej vzdialenosť cieľového konceptu k' od zdrojového nie je väčšia ako určitá zvolená konštanta c.

$$\begin{aligned} \text{Correctness} &= \frac{1}{n} \sum_i f_{correct}(k_i, k'_i) \\ f_{correct}(k, k') &= \begin{cases} 0 & \text{ak } d(k, k') > c \\ 1 & \text{ak } d(k, k') \leq c \end{cases} \end{aligned}$$

Distinctness: priemerný počet rôznych reťazcov na jeden koncept. Hodnota 1 znamená, že pre n rôznych konceptov má agent n rôznych názvov (optimum). $1/n$ znamená, že generuje rovnaký reťazec pre všetky koncepty (najhorší možný stav).

Length: relatívna priemerná dĺžka reťazcov. Hodnota 1 znamená, že všetky generované reťazce dosahujú hraničnú dĺžku $MaxLength$ (najhorší stav).

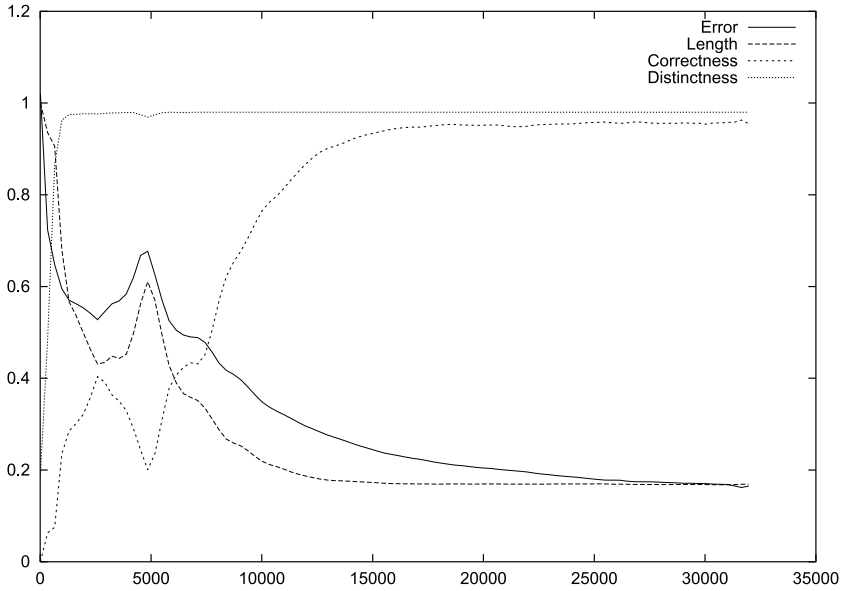
IteratedDistance: túto veličinu definujeme nižšie (v odseku 6.2.12)

6.2.8 Binárne meaning vektory

Úplne na úvod sme zreprodukovali experiment za podmienok, aké použil Batali. Priebeh hodnotiacich veličín vidno na obr. 6.12 a je takmer zhodný s priebehmi, zaznamenanými Batalim v [2], rovnako aj vzniknuté slovníky majú podobné vlastnosti. Binárnym meaning vektorom sme sa potom už ďalej nevenovali.

6.2.9 Všeobecne o reálnych meaning vektoroch

Vytváranie slovníka nad priestorom reálnych vektorov je už na prvý pohľad podstatne ťažší problém ako pre binárne vektory. Dôvody sú nasledovné:

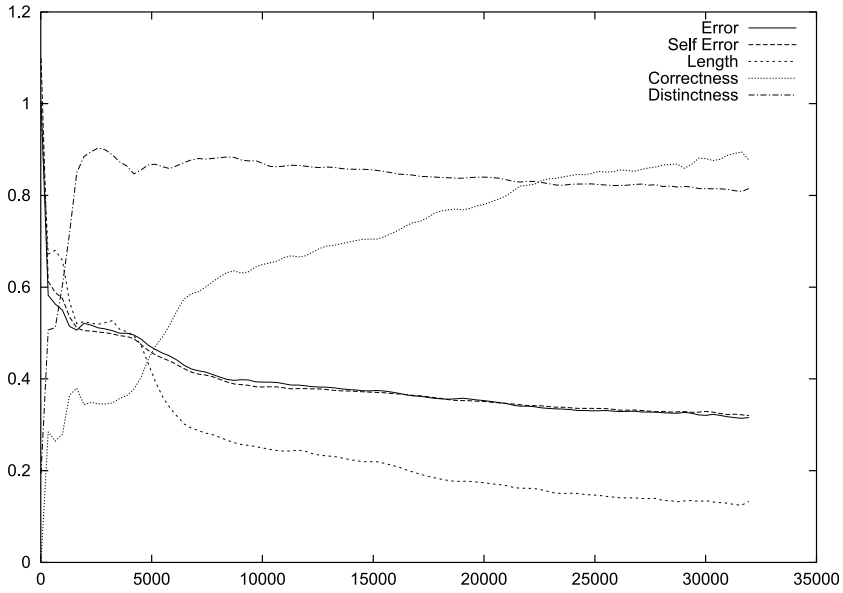


Obr. 6.12: Priebeh globálnych kvantitatívnych parametrov evolúcie slovníka. Na osi x sú vynesené trénovacie kolá.
Reprodukcia Bataliho experimentu s binárnymi meaning vektormi na našej experimentálnej platforme.

- aktivácie neurónov výstupnej vrstvy siete sa musia nachádzať takmer vždy v lineárnej oblasti aktivačnej funkcie. Z toho vyplývajú väčšie nároky na presnosť práce siete. Pri binárnych vektoroch má proces učenia tendenciu „usádzať“ postsynaptické potenciály výstupných neurónov v priemere čo najďalej od stredu, čím sa aktivácie blížia k infimu/supremu aktivačnej funkcie. Z hľadiska správnosti výstupu je jedno, či je potenciál rovný $+0.1$, $+1$, alebo $+1000$, vždy ide o ten istý binárny stav. Pri vektoroch z $(0,1)^n$ však o správnosti rozhodujú aj veľmi malé intervaly potenciálu. Sieť sa teda musí učiť podstatne „nelineárnejšie“ zobrazenie.

- konštanta $MaxDist$, definujúca podmienku ukončenia generovania reťazca (určuje vzdialenosť, pre ktorú je výsledný meaning vektor už „dosť dobrý“) musí byť podstatne nižšia, ako pri interpretácii na binárne vektory, kde stačí ak sú aktivácie neurónov do vzdialenosti 0.5 od hodnoty príslušnej zložky binárneho vzoru. Dá sa očakávať, že na dosiahnutie potrebnej vzdialenosti cieľového meaning vektoru od zdrojového bude treba dlhšie reťazce, ako v prípade binárnych vektorov.

Aby sme sa presvedčili o schopnosti rekurentnej siete pracovať v takomto režime, navrhli sme experiment podľa nasledovnej tabuľky:



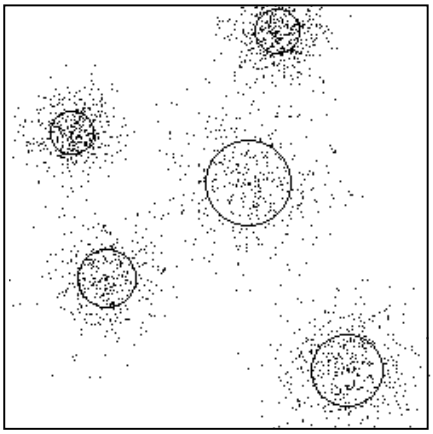
Obr. 6.13: Priebeh globálnych kvantitatívnych parametrov evolúcie slovníka. Na osi x sú vynesené trénovacie kolá.
Overovací experiment myšlienky, spracovávať reálne vektory namiesto binárnych. Agenti „pomenúvajú“ 20 náhodne (s uniformnou distribúciou) vygenerovaných bodov z $(0,1)^3$. Topológia siete je (4+15)-15-3, veľkosť populácie 20, learning rate 0,01, moment 0, BPMT.

topológia siete:	(4+15)-15-3
Σ:	$\{a, b, c, d\}$
učenie, learning rate, moment:	BPMT, 0.01, 0
$MaxLength$, $MaxDist$:	20, 0.2
veľkosť populácie:	20
generovanie meaning vektorov:	pevná trénovacia množina. 20 vektorov, náhodne vygenerovaných z $(0,1)^3$ s uniformnou distribúciou
trénovacie kolo:	k náhodne zvolenému počúvajúcemu sa náhodne vyberá 10 rozprávajúcich
komunikačná epizóda:	meaning vektory z tr.m. sa prekomunikujú v náhodnom poradí

Priebehy veličín sú na obr. 6.13 a ako vidno, agenti si s týmto problémom poradili úspešne. Pevná množina meaning vektorov, navyše s uniformnou distribúciou však nie cieľom našich experimentov. Ďalej sa teda budeme venovať inej metóde generovania meaning vektorov.

6.2.10 Model konceptuálnej reprezentácie

„Zostrojiť“ vhodný konceptuálny priestor s dostatočným množstvom prototypov s vhodnými vlastnosťami nie je ľahké. Pre potreby experimentov sme preto navrhli jednoduchý model abstraktného konceptuálneho priestoru.



Obr. 6.14: Model dvojrozmerného konceptuálneho priestoru ($d = 2,\ n = 5,\ c = 0.1,\ r_i \in (0.01, 0.1)$)

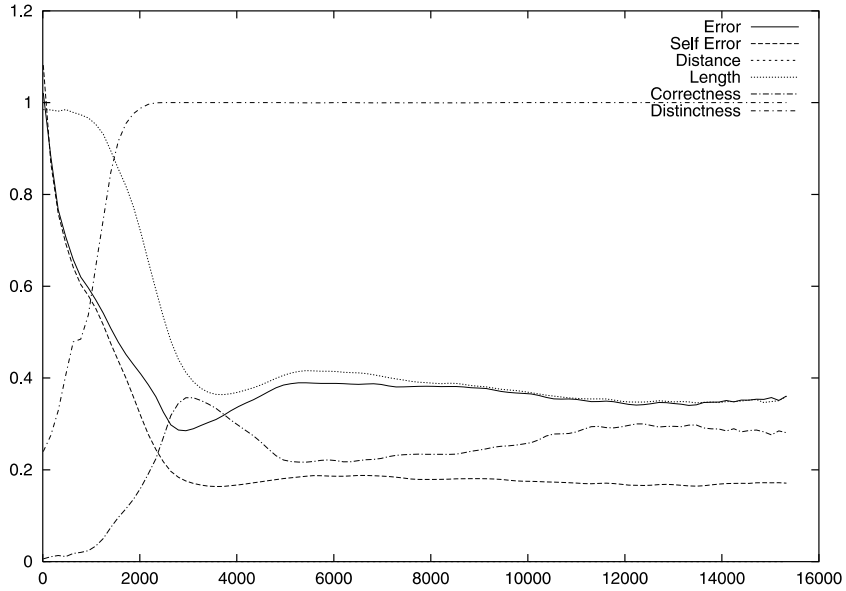
Postupujeme nasledovne: zvolíme počet dimenzií d, počet prototypov n, ku každému priradíme prototypový polomer a v $(0, 1)^d$ náhodne, s uniformnou distribúciou vygenerujeme n centrálnych bodov tak, aby polomery každých dvoch boli od seba vzdialené najmenej c. Minimálna vzdialenosť ľubovolných dvoch prototypov je tak $r_1 + r_2 + c$.

Inštancie prototypov (vlastné objekty, resp. meaning vektory) potom generujeme generátorom náhodných čísel s Gaussovskou distribúciou so stredom v danom centre a so smerodajnou odchýlkou rovnou príslušnému prototypovému polomeru (tzn. 50% bodov leží vo vnútri polomeru). Príklad takéhoto priestoru v 2D vidno na obr. 6.14

Pre potreby experimentov sme samozrejme používali priestor s väčším počtom rozmerov (dimenziou) aj prototypov, najčastejšie priestor s troma dimenziami, 20 prototypmi, s polomerom 0.05, centrá ktorých boli vzdialené najmenej 0.3. Pre overenie zovšeobecniteľnosti sme vykonali aj experimenty v priestoroch s dimenziou 6 a 8.

6.2.11 Základná štruktúra experimentu

S pomocou popísaného generátora meaning vektorov sme zrealizovali niekoľko experimentov. Najčastejšie použité parametre ukazuje nasledujúca tabuľka. Budeme sa na ne ďalej odvolávať, ako na referenčné:



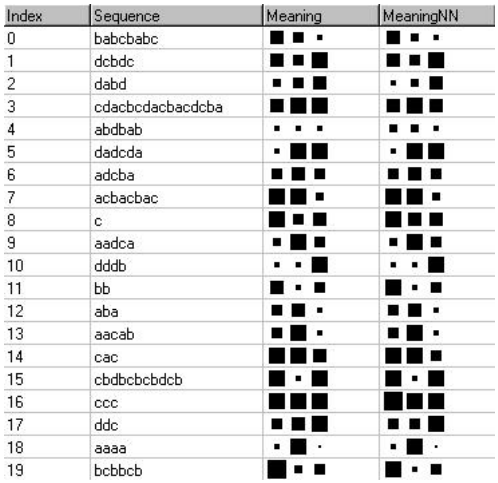
Obr. 6.15: Priebeh globálnych veličín evolúcie slovníka. Na osi x sú vynesené trénovacie kolá.
Simulácia bežala s referenčnými parametrami z odseku 6.2.11.

topológia siete:	(4+15)-15-3
Σ:	$\{a, b, c, d\}$
učenie, learning rate, moment:	BPMT, 0.005, 0
$MaxLength$, $MaxDist$:	20, 0.2
veľkosť populácie:	20
generovanie meaning vektorov:	generátorom vektorov z $(0, 1)^3$ s gaussovským rozdelením. 20 centier s polomerom 0.05 a minimálnou vzdialenosťou 0.3
trénovacie kolo:	k náhodne zvolenému počúvajúcemu sa náhodne vyberá 10 rozprávajúcich
komunikačná epizóda:	20 vygenerovaných meaning vektorov

Priebeh sledovaných veličín je na obr. 6.15. Z priebehu veličiny Correctness by sa zdalo, že jazyková koherencia nie je oproti modelu s binárnymi meaning vektormi veľmi veľká. Z distribúcie chýb komunikačných epizód (obr. 6.27d) vyplýva, že 63% komunikačných epizód dosahuje vzdialenosť výsledného meaning vektoru menšiu ako 0.15, čo je $\frac{1}{2}$ najmenšej možnej vzdialenosti prototypov. Takže najmenej 63% komunikačných aktov zachovalo príslušnosť meaning vektoru k správnemu prototypu. Ukážka slovníka je na obr. 6.16.

Priebehy veličín, ako ich vidíme na grafe sa ukázali byť charakteristické pre tento model. Priebehy, aj dosahované minimá „odolávajú“ rôznym úpravám parametrov experimentu. Skúšali sme meniť dimenziu kontextovej vrstvy, vkladať ďalšie vrstvy do rekurentnej siete - pred, aj za kontextovú vrstvu, dosahované výsledky to ovplyvňuje len málo a vždy smerom k horšiemu.

Významnú úlohu hrá konštanta $MaxDist$, určujúca (okrem počiatočných



Index	Sequence	Meaning	MeaningNN
0	babcbabc		
1	dcbdc		
2	dabd		
3	cdacbcdacbacdcba		
4	abdbab		
5	dadcda		
6	adcba		
7	acbacbac		
8	c		
9	aadca		
10	dddb		
11	bb		
12	aba		
13	aacab		
14	cac		
15	cbdbcbcbdcb		
16	ccc		
17	ddc		
18	aaaa		
19	bcbbcb		

Obr. 6.16: Slovník, generovaný jedným z agentov po stabilizácii vývoja v populácii

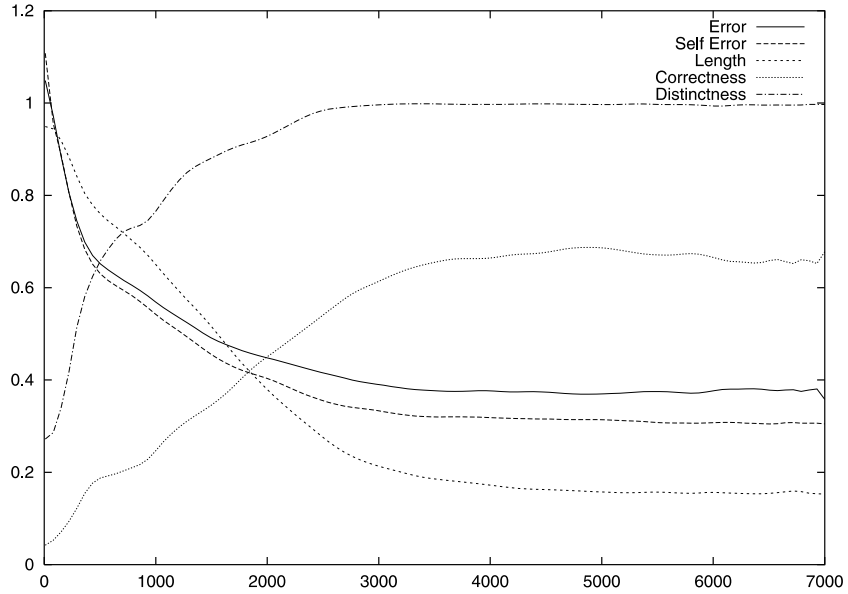
štádií evolúcie, kedy sekvencie dosahujú dĺžku $MaxLength$) podmienku ukončenia generovania sekvencie. Pokiaľ je príliš veľká (≥ 0.2), dosahovaná presnosť je nízka (obr. 6.17). Jedine maximálna dosiahnutá korektnosť samozrejme, keďže je definovaná priamo na základe tejto konštanty, narastie.

Zaujímavá je tiež veľkosť populácie. Model adaptácie samozrejme funguje už pre jediného agenta (s tým, že v tomto prípade agent účinkuje zároveň ako rozprávajúci, aj ako počúvajúci), dokonca pokiaľ úspešne skonverguje, dostane sa na lepšie minimum, ako v prípade veľkej populácie (obr. 6.27b). Evolúcia však má za týchto okolností tendenciu občas sa „zasekávať“ v lokálnych minimách povrchu chybovej funkcie. Je to v súlade s filozofiou jazyka, ako distribuovaného dynamického systému, kde hrá primárnu úlohu jazyková interakcia mnohých agentov.

6.2.12 Zreťazenie komunikačných epizód

Zaujímalo nás ďalej, nakoľko sú agenti schopní zachovať prenášaný význam vzhľadom na iteráciu vzťahu $k_{t+1} = F_{int} \circ F_{den}(k_t)$. Ide teda o hľadanie pomyselného pevného bodu pri komunikovaní významu. Metaforicky možno povedať, že ide analógiu kolovania nejakej informácie „z úst do úst“. Reprezentatívny príklad takejto iterácie je na obr. 6.18. Vidno, že napriek náhodnému výberu agentov na „prenos ďalej“ z populácie, zostáva štruktúra meaning vektora relatívne stála. Objektívnejšie o priemernej odolnosti transformácií významov voči iterácii prezradí globálna veličina, ktorú sme na tento účel zaviedli. Nazvali sme ju *Iterated distance*, pričom definícia je nasledovná:

$$\begin{aligned} E_{ID} &= \frac{1}{n}\sum_{i=1}^{n} E'_{ID}(k_i), \qquad k_i \in \texttt{Prototypes} \\ E'_{ID}(k_0) &= d(k_{t_{max}}, k_0) \qquad \text{(ďalej používame } t_{max} = 20) \end{aligned}$$



Obr. 6.17: Priebeh globálnych veličín evolúcie slovníka. Na osi x sú vynesené trénovacie kolá.
Referenčné parametre; $MaxDist = 0.2$.

$$\begin{aligned} k_{t+1} &= F_{int}^{a_{t+1}} \circ F_{den}^{a_t}(k_t) \\ a_{t+1} &= \texttt{Population.RandomAgentOtherThan}(a_t) \end{aligned}$$

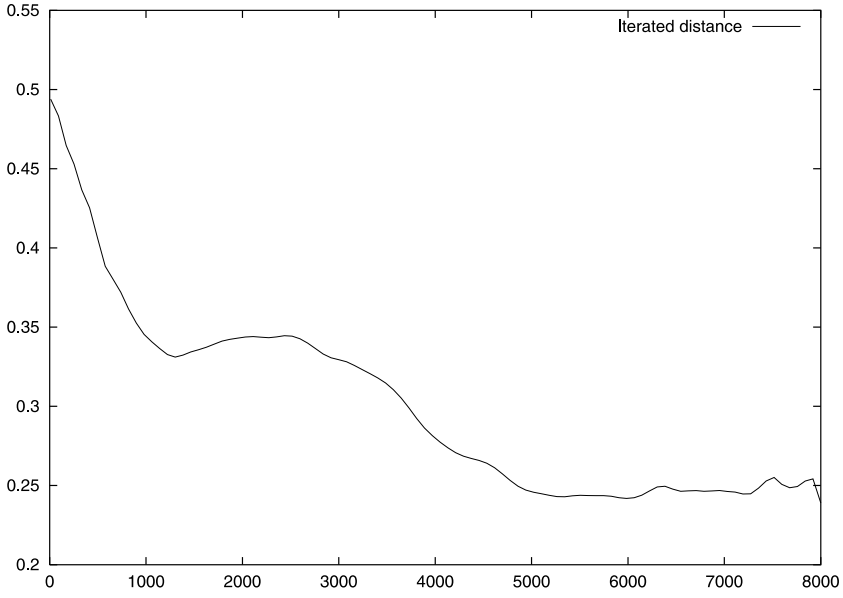
Veličina teda meria priemernú vzdialenosť, do akej „doputuje“ meaning vektor, keď koluje medzi agentami. Priebeh veličiny počas jazykovej evolúcie podľa referenčných podmienok je na obr. 6.19. Veličina klesá približne na hodnotu 0.23. Je to veľmi slušná hodnota, vzhľadom na spôsob, akým je získaná. Vypovedá o konvergencii jazykovej evolúcie, ako aj o povahe interpretačnej/denominačnej funkcie. Pripomíname, že najmenšia vzdialenosť prototypov v našom modelovom priestore je 0.3. Meaning vektor sa teda v priemernom prípade ani po 20 zreťazených komunikačných epizódach neodchýli viac, než k susednému prototypu.

6.2.13 Konceptuálne priestory s vyššou dimenziou

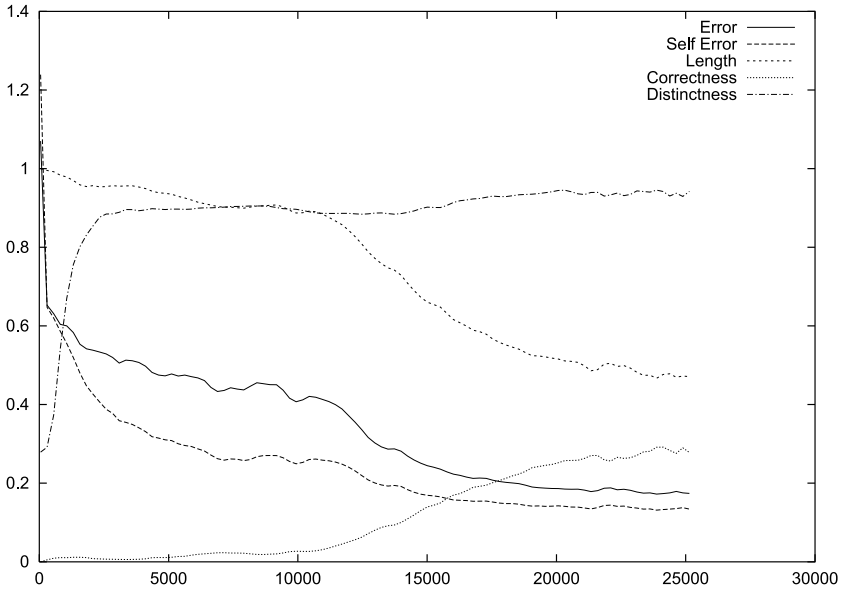
Čiastočne zo zvedavosti, čiastočne kvôli vysokej dimenzionalite priestoru, sme nechali bežať aj simuláciu, kde boli meaning vektormi parametrické vektory geometrických primitív Marrovej reprezentácie (popisujeme ju v odseku 6.1). Simulácia bežala s nasledovnými parametrami:

Index	Agents	Sequence	Meaning	Meaning'	Dist.f.org	Dist.
0	4->6	bbcb			0,08484	0,08484
1	6->13	bbcb			0,08253	0,05639
2	13->18	bbc			0,14966	0,13957
3	18->19	bbc			0,17521	0,05773
4	19->6	bbc			0,15092	0,08095
5	6->1	bbc			0,16281	0,01804
6	1->0	bbc			0,15315	0,03883
7	0->14	bbc			0,15521	0,04244
8	14->17	bbc			0,10175	0,08113
9	17->15	bbc			0,31542	0,26203
10	15->3	bcb			0,27722	0,05542
11	3->15	bcb			0,27194	0,06420
12	15->8	bcb			0,12784	0,15203
13	8->0	bbc			0,15315	0,08103
14	0->11	bbc			0,20604	0,08691
15	11->14	bbc			0,15521	0,06606
16	14->3	bbc			0,28899	0,17070
17	3->10	bcb			0,14745	0,27413
18	10->15	bcb			0,27194	0,21603
19	15->16	bcb			0,10954	0,17826

Obr. 6.18: Iterácia $k' = F_{int} \circ F_{den}(k)$ cez postupnosť rôznych agentov. Stĺpce po rade: (1) hĺbka iterácie; (2) komunikujúci agenti v tvare rozprávajúci → počúvajúci; (3) názov, vygenerovaný rozprávajúcim; (4) meaning vektor k_t, denominovaný rozprávajúcim; (5) meaning vektor k_{t+1}, dekódovaný počúvajúcim; (6) vzdialenosť významu od inicializačného, $d(k_t, k_0)$; (7) $d(k_t, k_{t+1})$



Obr. 6.19: Priebeh iterovanej vzdialenosti. Na osi x sú vynesené trénovacie kolá. Simulácia s referenčnými parametrami z odseku 6.2.11.



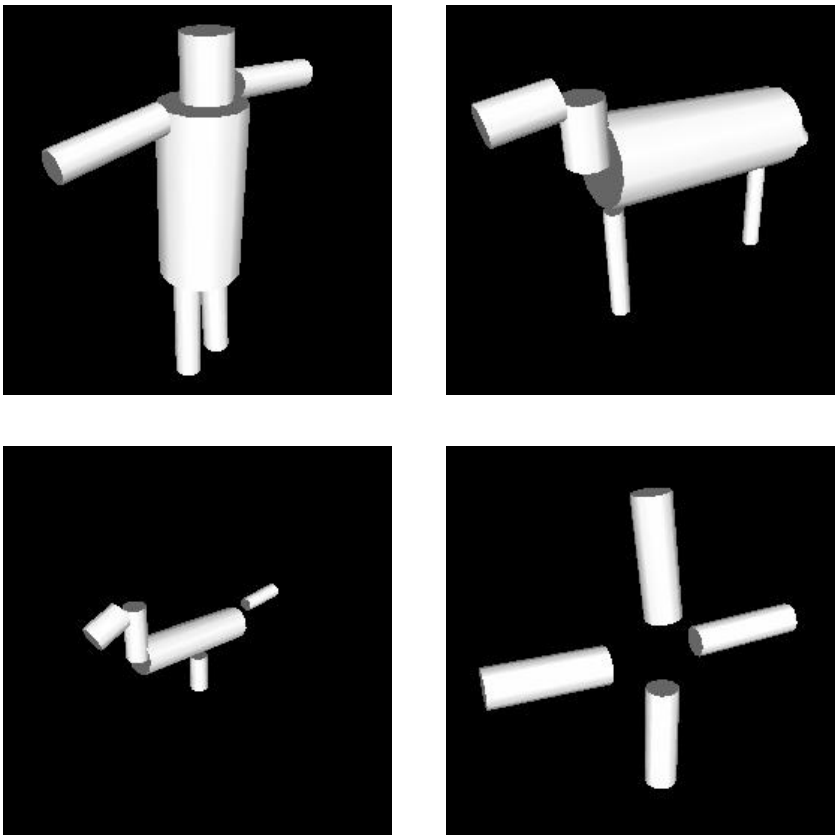
Obr. 6.20: Priebeh globálnych kvantitatívnych parametrov evolúcie slovníka. Na osi x sú vynesené trénovacie kolá.
Agneti „pomenúvajú“ primitíva objektov Marrovej reprezentácie (6.1). Topológia siete je (4+30)-30-8, veľkosť populácie 20, learning rate 0,005, moment 0, BPMT.

topológia siete:	(8+30)-30-8
Σ:	$\{a, b, c, d\}$
učenie, learning rate, moment:	BPMT, 0.005, 0
$MaxLength$, $MaxDist$:	20, 0.2
veľkosť populácie:	20
generovanie meaning vektorov:	pevná trénovacia množina. 24 vektorov z $(0, 1)^8$ - parametrických vektorov geometrických primitív Marrových komplexných objektov
trénovacie kolo:	k náhodne zvolenému počúvajúcemu sa náhodne vyberá 10 rozprávajúcich
komunikačná epizóda:	meaning vektory z tr.m. sa prekomunikujú v náhodnom poradí

Priebeh evolúcie uvádzame na obr. 6.20, vizualizované objekty, zložené z primitív, vygenerovaných na základe prekomunikovaných meaning vektorov vidíme na obr. 6.21.

6.2.14 Vlastnosti komunikácie v okolí prototypov

Keďže naším „tajným“ predpokladom ohľadne tohto modelu bol potenciálny vznik slovníka, jednoznačne priraďujúceho jedinečné názvy jednotlivým konceptom, pozrime sa teraz bližšie na to, ako agenti pomenúvajú meaning vektory v okolí prototypov. Už z priebehov veličín, zaznamenávaných počas evolúcie je



Obr. 6.21: Ukážka toho, „ako dopadli“ Marrove komplexné objekty po prekomunikovaní ich geometrických primitív (reprezentovaných v 8 rozmernom „konceptuálnom“ priestore) medzi dvoma agentami. Na poslednom obrázku je prekomunikovaná verzia symetrického terča (valec, rotovaný okolo súradnicového stredu postupne o 90^o, všetko v rovine X-Y). Vzory ostatných objektov sú na obr. 6.2.

1	2	3	4	5	6	7	8
Index	Shift	Shift NN	Meaning	Meaning NN	Sequence	Dist.f.ref	Dist.
0	(0,00, 0,00, 0,00)	(0,04, -0,04, -0,01)	(0,77, 0,88, 0,51)	(0,81, 0,84, 0,50)	cac	0,00000	0,05308
1	(0,03, -0,02, 0,00)	(0,04, -0,04, -0,01)	(0,80, 0,86, 0,51)	(0,81, 0,84, 0,50)	cac	0,03611	0,02126
2	(-0,02, 0,07, 0,01)	(-0,09, 0,03, 0,06)	(0,75, 0,95, 0,52)	(0,68, 0,91, 0,57)	cacacad	0,07701	0,09401
3	(-0,02, -0,02, -0,01)	(-0,06, -0,10, -0,05)	(0,76, 0,86, 0,50)	(0,71, 0,78, 0,46)	ca	0,02975	0,09844
4	(-0,01, -0,01, -0,03)	(0,04, -0,04, -0,01)	(0,77, 0,87, 0,48)	(0,81, 0,84, 0,50)	cac	0,02858	0,05463
5	(-0,06, 0,04, -0,02)	(-0,11, 0,04, 0,02)	(0,71, 0,93, 0,49)	(0,66, 0,92, 0,53)	cacadac	0,07545	0,06378
6	(0,02, -0,06, -0,01)	(0,04, -0,04, -0,01)	(0,79, 0,82, 0,50)	(0,81, 0,84, 0,50)	cac	0,06237	0,02755
7	(0,00, -0,02, 0,12)	(-0,01, 0,00, 0,15)	(0,77, 0,86, 0,63)	(0,76, 0,88, 0,66)	ccadac	0,11693	0,04432
8	(0,02, -0,01, 0,05)	(0,04, -0,04, -0,01)	(0,79, 0,87, 0,56)	(0,81, 0,84, 0,50)	cac	0,05033	0,06154
9	(-0,02, 0,05, 0,10)	(-0,09, 0,01, 0,08)	(0,75, 0,93, 0,61)	(0,69, 0,89, 0,59)	caccada	0,11333	0,07427
10	(-0,04, 0,01, -0,07)	(0,02, 0,02, -0,12)	(0,73, 0,89, 0,44)	(0,79, 0,90, 0,39)	caca	0,08078	0,08154
11	(0,07, -0,01, 0,03)	(0,04, -0,04, -0,01)	(0,84, 0,87, 0,54)	(0,81, 0,84, 0,50)	cac	0,07755	0,05367
12	(0,03, 0,02, 0,03)	(0,04, -0,04, -0,01)	(0,80, 0,90, 0,54)	(0,81, 0,84, 0,50)	cac	0,04914	0,07299
13	(0,02, 0,00, -0,05)	(0,04, -0,04, -0,01)	(0,79, 0,88, 0,46)	(0,81, 0,84, 0,50)	cac	0,05480	0,05866
14	(-0,07, -0,03, 0,03)	(-0,15, -0,02, 0,08)	(0,70, 0,85, 0,54)	(0,62, 0,86, 0,59)	cacad	0,08253	0,09875
15	(-0,05, 0,03, 0,00)	(-0,11, 0,04, 0,02)	(0,73, 0,91, 0,51)	(0,66, 0,92, 0,53)	cacadac	0,05686	0,06725
16	(-0,08, -0,02, 0,04)	(-0,15, -0,02, 0,08)	(0,70, 0,86, 0,55)	(0,62, 0,86, 0,59)	cacad	0,08771	0,08699
17	(0,01, -0,01, 0,06)	(0,04, -0,04, -0,01)	(0,79, 0,87, 0,57)	(0,81, 0,84, 0,50)	cac	0,06166	0,07329
18	(0,11, -0,03, -0,05)	(0,04, -0,04, -0,01)	(0,88, 0,85, 0,46)	(0,81, 0,84, 0,50)	cac	0,12694	0,08780
19	(-0,08, 0,03, 0,04)	(-0,15, -0,02, 0,08)	(0,69, 0,91, 0,55)	(0,62, 0,86, 0,59)	cacad	0,09445	0,09720

Obr. 6.22: Pomenúvanie meaning vektorov v okolí prototypu $(0.77, 0.88, 0.51)$. Stĺpce po rade: (1) č. iterácie; (2) diferenčný vektor $k_c - k$; (3) $k_c - k'$; (4) k, náhodný meaning vektor z okolia k_c (Gauss s $\mu = k_c,\ \sigma = 0.2$); (5) $k' = F_{int} \circ F_{den}(k)$; (6) $\alpha = F_{den}(k)$; (7) $d(k, k_c)$; (8) $d(k, k')$. 0. riadok obsahuje prototyp k_c.

zrejmé, že agenti dokážu samotné prototypy (centrálne body konceptov) jednoznačne rozlíšiť (Distinctness = 1.0).

Z množiny prototypov referenčného experimentu sme ďalej náhodne vybrali prototyp k_0 a jeho najbližších susedov k_1, k_2, k_3. Vzdialenosť od k_0 k najvzdialenejšiemu zvolenému susedovi bola 0.374, ide teda o naozaj blízke prototypy (minimálna možná vzdialenosť je 0.3). Na obr. 6.22-6.25 vidno, ako náhodne zvolený agent pomenúva meaning vektory v okolí týchto prototypov. Vidíme, že aj príslušnosť blízkych inštancií (meaning vektorov) k ich prototypom agent jednoznačne zohľadňuje v tvare generovaných sekvencií. V okolí každého prototypu dominuje jeden prefix, sekvencie majú pre jednotlivé prototypy najčastejšie tvar cac*, cda*, ac*, cc*). Z tabuliek zároveň vidíme, ako vyzerá množina sekvencií, generovaných na okolí prototypu.

Ďalšia skutočnosť, ktorá nás zaujímala bola, nakoľko sa pri komunikácii zachováva relatívna poloha inštancie prototypu voči centrálnemu bodu (teda orientácia, prípadne dĺžka vektora $k_{instance} - k_{prototype}$) . V prípade dimenzií, reprezentujúcich polohu objektu na scéne by sme problém mohli neformálne priblížiť nasledovne: keď namiesto prototypovej pozície v priestore vezmeme bod "vľavo hore" od prototypu a vyšleme jeho pomenovanie druhému agentovi, získa interpretáciou tohto pomenovania vo svojom konceptuálnom priestore tiež bod "vľavo hore" od príslušného prototypu? Tento aspekt je dôležitý aj z hľadiska prípadnej snahy, rozšíriť systém nejakým spôsobom o modelovanie metafor. Tie, ako rád a často demonštruje Gärdenfors, závisia práve na zachovávaní orientácie v podpriestoroch CS.

Znova sme teda náhodne zvolili prototyp v CS a vygenerovali okolité body (inštancie) posúvaním pozdĺž jednotlivých rozmerov priestoru $(k + (0, 0, c),\ k +$

1	2	3	4	5	6	7	8
Index	Shift	Shift NN	Meaning	Meaning NN	Sequence	Dist.f.ref	Dist.
0	(0,00, 0,00, 0,00)	(-0,08, -0,03, 0,00)	(0,60, 0,72, 0,72)	(0,52, 0,68, 0,72)	cda	0,00000	0,08543
1	(-0,03, -0,01, 0,00)	(-0,08, -0,03, 0,00)	(0,57, 0,70, 0,73)	(0,52, 0,68, 0,72)	cda	0,03579	0,05045
2	(-0,04, 0,10, -0,03)	(-0,12, 0,11, 0,02)	(0,56, 0,81, 0,69)	(0,48, 0,83, 0,74)	cdacad	0,10880	0,09665
3	(0,04, 0,06, 0,02)	(0,09, 0,06, 0,01)	(0,63, 0,77, 0,74)	(0,69, 0,78, 0,73)	cdac	0,06877	0,05347
4	(-0,07, -0,06, -0,02)	(-0,08, -0,03, 0,00)	(0,53, 0,65, 0,71)	(0,52, 0,68, 0,72)	cda	0,09516	0,03316
5	(0,07, -0,01, 0,11)	(0,09, 0,02, 0,17)	(0,67, 0,71, 0,83)	(0,69, 0,73, 0,89)	cdcadcdcabadc	0,12936	0,06712
6	(-0,05, -0,02, 0,04)	(-0,08, -0,03, 0,00)	(0,55, 0,70, 0,76)	(0,52, 0,68, 0,72)	cda	0,06229	0,05075
7	(0,13, -0,01, 0,06)	(0,16, 0,05, 0,04)	(0,73, 0,71, 0,78)	(0,76, 0,77, 0,76)	cdca	0,14307	0,07115
8	(-0,01, 0,02, 0,02)	(-0,08, -0,03, 0,00)	(0,59, 0,73, 0,74)	(0,52, 0,68, 0,72)	cda	0,02585	0,08405
9	(0,02, -0,03, -0,01)	(-0,08, -0,03, 0,00)	(0,62, 0,68, 0,72)	(0,52, 0,68, 0,72)	cda	0,03928	0,09996
10	(0,02, 0,09, -0,11)	(0,07, 0,14, -0,09)	(0,62, 0,80, 0,61)	(0,67, 0,85, 0,63)	cadca	0,14007	0,07872
11	(0,01, 0,07, 0,06)	(0,09, 0,06, 0,01)	(0,61, 0,78, 0,79)	(0,69, 0,78, 0,73)	cdac	0,09300	0,09248
12	(0,00, 0,00, 0,00)	(-0,08, -0,03, 0,00)	(0,59, 0,72, 0,72)	(0,52, 0,68, 0,72)	cda	0,00555	0,08086
13	(-0,02, -0,05, 0,04)	(-0,08, -0,03, 0,00)	(0,57, 0,66, 0,76)	(0,52, 0,68, 0,72)	cda	0,07055	0,07103
14	(0,04, 0,02, -0,01)	(0,09, 0,06, 0,01)	(0,64, 0,74, 0,71)	(0,69, 0,78, 0,73)	cdac	0,04723	0,06698
15	(0,06, -0,07, 0,07)	(0,08, -0,11, 0,08)	(0,65, 0,64, 0,79)	(0,68, 0,60, 0,81)	cd	0,11536	0,04773
16	(0,01, -0,08, -0,05)	(-0,05, -0,08, -0,02)	(0,61, 0,64, 0,67)	(0,55, 0,63, 0,70)	cdacbad	0,08988	0,06139
17	(0,05, 0,05, -0,06)	(0,07, 0,14, -0,09)	(0,65, 0,77, 0,66)	(0,67, 0,85, 0,63)	cadca	0,09588	0,09287
18	(0,03, 0,00, 0,00)	(0,09, 0,06, 0,01)	(0,63, 0,71, 0,72)	(0,69, 0,78, 0,73)	cdac	0,03014	0,08711
19	(0,01, 0,12, -0,10)	(0,07, 0,14, -0,09)	(0,61, 0,84, 0,63)	(0,67, 0,85, 0,63)	cadca	0,15763	0,06493

Obr. 6.23: Pomenúvanie meaning vektorov v okolí prototypu $(0.60, 0.72, 0.72)$.

1	2	3	4	5	6	7	8
Index	Shift	Shift NN	Meaning	Meaning NN	Sequence	Dist.f.ref	Dist.
0	(0,00, 0,00, 0,00)	(-0,02, 0,04, 0,07)	(0,79, 0,79, 0,18)	(0,76, 0,83, 0,25)	acabca	0,00000	0,08767
1	(0,06, -0,04, -0,05)	(0,07, -0,08, 0,02)	(0,84, 0,75, 0,12)	(0,86, 0,70, 0,20)	acbacabcaabc	0,08591	0,09280
2	(-0,02, -0,07, 0,01)	(-0,04, -0,12, 0,00)	(0,77, 0,72, 0,18)	(0,75, 0,67, 0,18)	acbacab	0,06901	0,05563
3	(-0,07, -0,05, 0,02)	(-0,03, -0,14, 0,01)	(0,72, 0,73, 0,20)	(0,76, 0,65, 0,19)	acabcab	0,08829	0,09596
4	(-0,08, -0,10, 0,05)	(-0,13, -0,11, 0,07)	(0,71, 0,69, 0,23)	(0,66, 0,68, 0,25)	acba	0,13498	0,05790
5	(0,03, -0,09, -0,01)	(-0,04, -0,12, 0,00)	(0,82, 0,70, 0,16)	(0,75, 0,67, 0,18)	acbacab	0,09394	0,07881
6	(0,01, -0,04, 0,06)	(-0,01, -0,03, 0,11)	(0,80, 0,75, 0,23)	(0,77, 0,76, 0,29)	cabac	0,06704	0,05973
7	(0,03, 0,01, 0,03)	(-0,01, -0,03, 0,11)	(0,81, 0,80, 0,20)	(0,77, 0,76, 0,29)	cabac	0,03941	0,09915
8	(0,04, -0,07, -0,08)	(0,04, -0,08, -0,02)	(0,83, 0,72, 0,10)	(0,83, 0,70, 0,16)	acbacabacabcabac	0,11480	0,06101
9	(-0,13, 0,02, -0,06)	(-0,10, 0,05, 0,02)	(0,66, 0,80, 0,11)	(0,69, 0,84, 0,20)	acabac	0,14584	0,09686
10	(0,03, 0,03, 0,10)	(0,00, -0,01, 0,12)	(0,82, 0,82, 0,28)	(0,79, 0,77, 0,30)	cacab	0,11068	0,05581
11	(0,00, -0,03, 0,05)	(-0,01, -0,03, 0,11)	(0,79, 0,76, 0,23)	(0,77, 0,76, 0,29)	cabac	0,05848	0,05944
12	(-0,02, 0,00, -0,11)	(0,01, -0,03, -0,02)	(0,77, 0,79, 0,07)	(0,79, 0,76, 0,15)	acabacabcabac	0,11261	0,09591
13	(0,00, 0,08, -0,09)	(-0,04, 0,10, 0,00)	(0,79, 0,87, 0,09)	(0,75, 0,89, 0,18)	acabcaacabac	0,11889	0,09788
14	(0,08, 0,05, 0,00)	(0,06, 0,06, 0,08)	(0,87, 0,84, 0,18)	(0,85, 0,84, 0,26)	cacabacabca	0,10038	0,08299
15	(0,04, -0,01, -0,04)	(0,01, -0,01, 0,01)	(0,82, 0,77, 0,14)	(0,79, 0,77, 0,19)	acabcabac	0,05241	0,05410
16	(0,03, -0,13, -0,06)	(0,03, -0,20, -0,01)	(0,82, 0,66, 0,11)	(0,82, 0,59, 0,17)	acbacbacab	0,14710	0,09035
17	(0,02, -0,05, 0,00)	(-0,04, -0,12, 0,00)	(0,80, 0,73, 0,18)	(0,75, 0,67, 0,18)	acbacab	0,05581	0,08438
18	(-0,04, -0,01, -0,03)	(-0,12, -0,08, -0,04)	(0,75, 0,77, 0,15)	(0,67, 0,71, 0,14)	acabcaba	0,05164	0,09996
19	(0,03, -0,10, -0,07)	(-0,04, -0,12, 0,00)	(0,82, 0,69, 0,11)	(0,75, 0,67, 0,18)	acbacab	0,12179	0,09721

Obr. 6.24: Pomenúvanie meaning vektorov v okolí prototypu $(0.79, 0.79, 0.18)$.

1	2	3	4	5	6	7	8
Index	Shift	Shift NN	Meaning	Meaning NN	Sequence	Dist.f.ref	Dist.
0	(0,00, 0,00, 0,00)	(0,03, 0,04, -0,06)	(0,92, 0,76, 0,83)	(0,95, 0,80, 0,77)	ccc	0,00000	0,07654
1	(0,06, -0,03, 0,11)	(-0,03, -0,05, 0,07)	(0,97, 0,73, 0,94)	(0,89, 0,70, 0,91)	ccdc	0,12530	0,09809
2	(0,05, 0,02, -0,04)	(0,03, 0,04, -0,06)	(0,96, 0,77, 0,79)	(0,95, 0,80, 0,77)	ccc	0,06243	0,03495
3	(0,01, 0,00, 0,01)	(0,03, 0,04, -0,06)	(0,93, 0,75, 0,84)	(0,95, 0,80, 0,77)	ccc	0,01490	0,08303
4	(-0,01, -0,01, 0,00)	(0,03, 0,04, -0,06)	(0,91, 0,75, 0,84)	(0,95, 0,80, 0,77)	ccc	0,01041	0,08587
5	(0,05, 0,01, 0,01)	(0,03, 0,04, -0,06)	(0,97, 0,77, 0,85)	(0,95, 0,80, 0,77)	ccc	0,05454	0,08209
6	(0,00, -0,07, -0,07)	(-0,02, -0,01, -0,13)	(0,92, 0,69, 0,76)	(0,89, 0,75, 0,70)	cc	0,10190	0,08786
7	(0,05, 0,00, -0,06)	(0,03, 0,04, -0,06)	(0,97, 0,76, 0,78)	(0,95, 0,80, 0,77)	ccc	0,07842	0,04307
8	(0,06, -0,02, 0,02)	(-0,01, -0,03, 0,05)	(0,98, 0,74, 0,85)	(0,90, 0,73, 0,88)	cccd	0,06734	0,08169
9	(0,07, 0,02, 0,05)	(0,05, 0,07, -0,02)	(0,99, 0,78, 0,88)	(0,97, 0,83, 0,82)	cccc	0,08795	0,08601
10	(0,00, 0,08, 0,04)	(0,05, 0,07, -0,02)	(0,92, 0,84, 0,87)	(0,97, 0,83, 0,82)	cccc	0,08677	0,07162
11	(0,06, -0,02, -0,03)	(0,03, 0,04, -0,06)	(0,98, 0,74, 0,81)	(0,95, 0,80, 0,77)	ccc	0,07094	0,07646
12	(0,04, -0,05, -0,10)	(-0,02, -0,01, -0,13)	(0,96, 0,71, 0,73)	(0,89, 0,75, 0,70)	cc	0,12078	0,07959
13	(-0,14, 0,08, 0,05)	(-0,13, 0,00, 0,06)	(0,78, 0,84, 0,88)	(0,78, 0,76, 0,89)	ccdacd	0,16366	0,07713
14	(0,04, 0,02, 0,05)	(-0,01, -0,03, 0,05)	(0,96, 0,78, 0,88)	(0,90, 0,73, 0,88)	cccd	0,06724	0,07378
15	(0,02, 0,03, -0,06)	(-0,02, -0,01, -0,13)	(0,94, 0,79, 0,78)	(0,89, 0,75, 0,70)	cc	0,06724	0,09356
16	(-0,01, -0,01, -0,01)	(0,03, 0,04, -0,06)	(0,90, 0,75, 0,83)	(0,95, 0,80, 0,77)	ccc	0,01860	0,08564

Obr. 6.25: Pomenúvanie meaning vektorov v okolí prototypu $(0.92, 0.76, 0.83)$.

Index	Shift	Shift NN	Shift	Shift NN
0	(0,00, 0,00, 0,00)	(0,12, -0,02, 0,06)		
1	(0,20, 0,00, 0,00)	(0,22, 0,01, -0,01)		
2	(0,00, 0,20, 0,00)	(0,06, 0,28, -0,01)		
3	(0,00, 0,00, 0,20)	(0,12, -0,02, 0,20)		
4	(-0,20, 0,00, 0,00)	(-0,03, -0,06, 0,10)		
5	(0,00, -0,20, 0,00)	(0,14, -0,16, 0,04)		
6	(0,00, 0,00, -0,20)	(0,06, -0,03, -0,29)		

Obr. 6.26: Transformácia meaning vektorov, posunutých od prototypu pozdĺž jednotlivých rozmerov priestoru. Stĺpce (3) a (5) obsahujú meaning vektory k', ktoré získal počúvajúci interpretáciou sekvencií rozprávajúceho. Vidno, že $F_{int} \circ F_{den}$ v tomto prípade „prejavuje náznaky“ zachovávania orientácie.

$(0, 0, -c)$, $k + (0, c, 0)$, ...). Ukázalo sa, že pod vzdialenosť dvojnásobku polomeru prototypu (0.1) sa orientácia týchto posunov pri komunikácii ani po zpriemerovaní cez veľa komunikačných epizód vôbec nezachováva. Badateľná začína byť až od hodnoty približne 0.2 (tabuľka je na obr. 6.26).

6.2.15 Vplyv fluktuácie agentov

V úvode sekcie spomíname, že otvorenosť populácie je jednou z rozumných požiadaviek na agentový systém. V snahe zistiť teda vplyv prírastku a úbytku agentov na priebeh evolúcie a stabilitu výsledného slovníka v našom modeli, upravili sme trénovací cyklus. Definovali sme konštantu $BirthPeriode$, určujúcu počet trénovacích kôl, po uplynutí ktorých sa nahradí náhodne zvolený agent z populácie novovygenerovaným. Nový agent má samozrejme náhodné váhové koeficienty.

Po vyhodnotení niekoľkých behov sa ukázalo, že populácia znáša aj dosť nízke hodnoty $BirthPeriode$. Konkrétne $BirthPeriode = 100$ už nemalo viditeľný vplyv na priebeh globálnych veličín. Táto hodnota pri populácii veľkosti 20

Interval	Count	Cumulative
< 0,01	0,0%	0,0%
< 0,05	0,0%	0,0%
< 0,10	0,4%	0,4%
< 0,15	1,2%	1,5%
< 0,20	1,8%	3,3%
< 0,30	6,7%	10,0%
< 0,50	32,5%	42,5%
>= 0,50	57,5%	100,0%

a

Interval	Count	Cumulative
< 0,01	0,0%	0,0%
< 0,05	10,0%	10,0%
< 0,10	85,0%	95,0%
< 0,15	0,0%	95,0%
< 0,20	0,0%	95,0%
< 0,30	5,0%	100,0%
< 0,50	0,0%	100,0%
>= 0,50	0,0%	100,0%

b

Interval	Count	Cumulative
< 0,01	0,0%	0,0%
< 0,05	3,3%	3,3%
< 0,10	17,5%	20,8%
< 0,15	24,2%	45,0%
< 0,20	25,8%	70,8%
< 0,30	22,5%	93,3%
< 0,50	5,8%	99,2%
>= 0,50	0,8%	100,0%

c

Interval	Count	Cumulative
< 0,01	0,0%	0,0%
< 0,05	7,5%	7,5%
< 0,10	28,4%	35,9%
< 0,15	27,6%	63,5%
< 0,20	16,0%	79,5%
< 0,30	12,3%	91,9%
< 0,50	7,4%	99,3%
>= 0,50	0,7%	100,0%

d

Obr. 6.27: Distribúcia chýb $d(k, k')$ komunikačných epizód. Distribúciu sme rátali cez všetky možné trojice (rozprávajúci, počúvajúci, prototyp), kde rozprávajúci $\neq$ počúvajúci (teda pre 20 agentov a 20 prototypov $19 \times 20 \times 20$ epizód). Výsledky sú (okrem *a*) zhromaždené po stabilizácii vývoja v populácii.
(a) úvodná, náhodne vygenerovaná populácia 20 agentov, (b) 1 agent (v tomto prípade komunikujúci sám so sebou), (c) 3 agenti, (d) 20 agentov v populácii

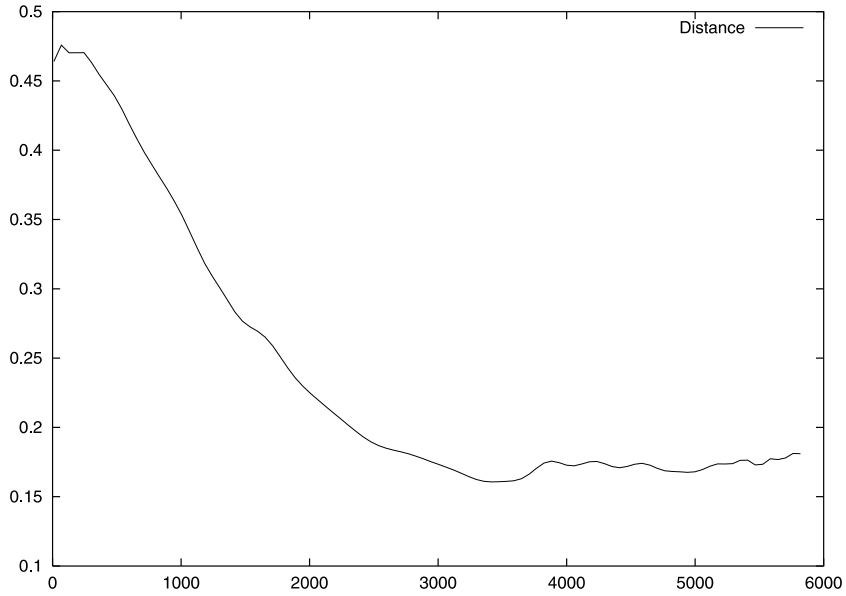
agentov znamená, že každý agent sa dožíva v priemere 2000 kôl. Kritickou bola hodnota približne $BirthPeriode = 20$. Pod touto hodnotou už nastáva dekoherencia a všetky veličiny klesajú/stúpajú na hodnoty ako pre novovygenerovanú populáciu.

Z výsledkov experimentu vidno veľkú robustnosť systému vzhľadom na fluktuácie populácie. Na druhej strane, nespozorovali sme ani žiaden signifikantný prínos fluktuácií na dosahované optimá globálnych veličín, hoci sme ich pozorovali ešte dostatočne dlho po zablokovaní obmeny agentov (prítomnosť „nezapracovaných“ agentov v populácii by samozrejme mohla zhoršovať hodnoty globálnych veličín).

Zaujímal nás aj priebeh „zapracovávania sa“ nového agenta v stabilizovanej populácii. Na obr. 6.28 uvádzame príklad. Vidno, že agent potrebuje asi 3000 trénovacích kôl simulácie (z ktorých účinkuje ako počúvajúci v priemere v $\frac{1}{20}$), aby dosiahol stabilnú chybu.

6.2.16 Dynamika rekurentnej siete

Keďže v prípade nášho modelu používame siete s kontextovou vrstvou s niekoľkými desiatkami neurónov, nie je celkom jednoduché exaktne analyzovať jej dynamiku. Preto aspoň jednoduchou vizuálnou metódou demonštrujeme, že dynamika kodéru je štandardná, fixpointová. Ako vidno z tabuľky na obr. 6.29, vektor aktivácií kontextovej vrstvy pre konštantný vzor na znakovom vstupe pomerne rýchlo konverguje k pevnému bodu kodéra. Na obr. 6.30 je zoznam pevných bodov pre jednotlivé znaky abecedy. Nie je však v našich silách vyšet-



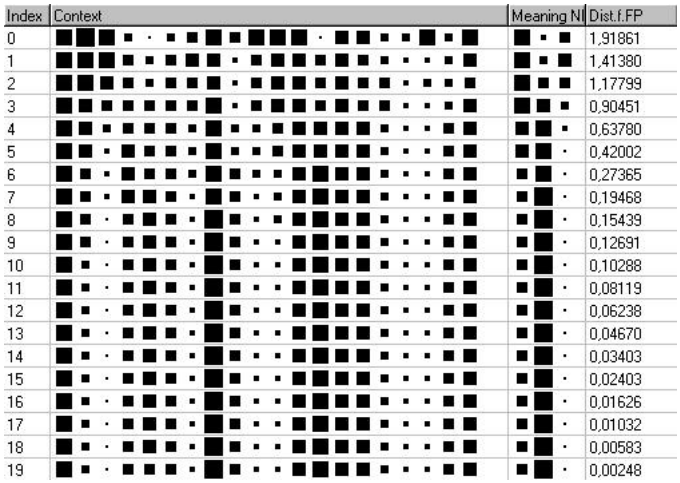
Obr. 6.28: Priebeh chyby komunikačných epizód nového agenta, vloženého do stabilnej populácie.

riť, či sú to jediné pevné body kodéra pre jednotlivé znaky.

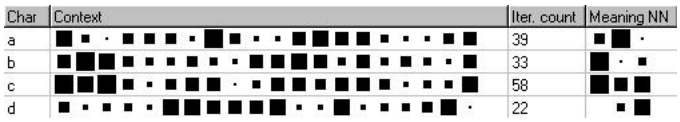
6.3 Stručne o použitej simulačnej platforme

Simulačné nástroje, na ktorých sme spúšťali experimenty pochádzajú prevažne z vlastnej dielne. Väčšina kódu je napísaná v Object Pascale, experimenty bežali na platforme Win32. Grafy generujeme zväčša GNUPlot-om, zabehaným multiplatformným nástrojom, disponujúcim interpréterom matematických výrazov a príkazov a umožňujúcim okrem iného kvalitný export grafov do rôznych formátov vrátane postscriptu.

Ako základ pre všetky testovacie aplikácie nám slúžia backgroundové knižnice pre jednotlivé konekcionistické architektúry (FFN, SRN/RAAM, SOM, Hopfield, TD(λ), ...), ktoré máme napísané čo možno najabstraktnejšie, sú dobre odladené a tak široko použiteľné pri experimentoch v oblasti umelej inteligencie. Tieto knižnice ďalej stoja na „nižších“ knižniciach, poskytujúcich utility typu generátory náhodných čísel s rôznymi distribúciami, kolekcie, abstraktné manipulácie s permutáciami, chybové a aktivačné funkcie pre dopredné siete a pod. Dôležitou bázovou knižnicou je aj zbierka tried, implementujúcich lineárnu algebru (TLinAlgFloatVector, TLinAlgFloatMatrix). Tieto triedy poskytujú všetky základné algebraické operácie a umožňujú veľmi rozumnú dekompozíciu zložitosti najmä rôznych konekcionistických modelov, kde možno prevažnú väčšinu výpočtov sformulovať pomocou lineárnej algebry. Triedy používajú referenčný prístup k zdieľaným dátam (tak je možné vždy keď sa dá kopírovať len referencie na dáta, uvoľňovanie pamäte je riešené reference countingom). Umožňujú aj efektívne blokové operácie s údajmi, čo je veľmi výhodné



Obr. 6.29: Konvergencia trajektórie v aktivačnom priestore kontextovej vrstvy rekurentnej siete, použitej na mieste interpretačnej funkcie. Pre ľubovoľný, náhodne vygenerovaný inicializačný kontext a konštantný znak na vstupe nastáva konvergencia k rovnakému pevnému bodu. To však neznamená, že iné pevné body neexistujú (nakoľko sme samozrejme skúšali len konečný počet inicializačných kontextov. . .).



Obr. 6.30: Tabuľka pevných bodov v kontextovom priestore siete, ku ktorým konverguje iterácia zobrazenia $c_{t+1} = F_{ih}(char_{const}, c_t))$ pre jednotlivé znaky abecedy. V stĺpci (3) uvádzame počet iterácií, potrebných na splnenie podmienky $|c_{t+1} - c_t| < 0.001$. V poslednom stĺpci je meaning vektor, ktorý by odpovedal príslušnému fixpointovému kontextu.

pre aplikácie ako napr. genetický algoritmus nad váhovými vektormi neurónovej siete, kde treba extenzívne presúvať bloky dát.

Veľmi praktickým nástrojom je aj abstraktný genetický algoritmus (OpenGA), umožňujúci simulovať umelú evolúciu nad ľubovoľným typom chromozómov. Niekoľko špecializovaných tried (GABinary, GAFloat, GANN) potom umožňuje hľadať (sub)optimálne riešenia rôznych problémov a slúži ako dobrý referenčný základ pre posudzovanie kvality riešení, získaných iným spôsobom. Aplikácie sme navrhovali tak, aby ich bolo možné navzájom linkovať k iným vyvíjaným testbedom, resp. medzi sebou navzájom. Môžu si tak poskytovať služby, najmä tie, ktoré treba „na každom kroku“ (OpenGA, SOM, ...).

Kapitola 7

Záver

V úvodnej časti práce sme sa venovali geometrickým vlastnostiam konceptuálnych priestorov, poukázali sme na nedostatky geometrických modelov konceptov, navrhovaných niektorými autormi.

V experimentálnej časti sme implementovali agentový systém, modelujúci emergenciu spoločného slovníka objektov, reprezentovaných v konceptuálnom priestore. Model je založený na jednoduchých rekurentných sieťach a je zameraný výhradne na simuláciu kulturálnej evolúcie. Zásadnou myšlienkou, ktorú sme v práci overili, je možnosť pomenúvať rekurentnou sieťou vektory z celého objemu priestoru $(a, b)^n$ - teda nie len binárne vektory, ležiace v rohoch príslušnej hyperkocky. Ukazujeme, že siete vhodnej topológie tento režim práce zvládajú a agentový systém preukazuje schopnosť konvergencie jazykovej evolúcie.

Ďalej sme sa venovali vlastnostiam takto adaptovanej komunikácie z hľadiska vzťahu ku konceptuálnej reprezentácii. Ukázali sme niekoľko výhodných vlastností tohto typu komunikácie.

Heslovite možno závery, vyplývajúce z výsledkov experimentov, zhrnúť do nasledovných bodov:

- emergencia koordinovaného slovníka konceptov v populácii agentov za skúmaných podmienok spoľahlivo nastáva. Miera koherentnosti slovníka medzi agentami je veľmi slušná, najmä s ohľadom na náročný režim práce sietí.
- model je rozšíriteľný aj na konceptuálne priestory s veľkou dimenzionalitou
- slovník, ktorý je výsledkom jazykovej evolúcie v populácii, preukazuje vlastnosti, výhodné z hľadiska reprezentovania konceptov. Ide najmä o veľkú podobnosť názvov pre inštancie toho istého prototypu a signifikantnú odlišnosť názvov pre inštancie rôznych prototypov.
- populácia preukazuje veľkú odolnosť voči úbytku agentov a prírastku nových agentov (s náhodnými počiatočnými parametrami siete)

Aj naďalej však zostáva ešte pomerne veľký priestor na prácu v oblasti agentového modelovania evolúcie slovníka. Niektoré námety, znovu len heslovite:

- použitie geograficky rozptýlenej populácie agentov s tým, že komunikácia by prebiehala len medzi „zemepisne blízkymi“ agentami. Týmto by bol model bližší Steelsovým požiadavkám na agentové modelovanie jazykových fenoménov, uvedeným v odseku 6.2 (lokálnosť). Dal by sa tak očakávať vznik dialektov slovníka, používaného agentami v zemepisne vzdialených oblastiach. Vznikol by priestor na memetickú evolúciu v populácii agentov - rôzne „dialekty“ jazyka by mohli „súťažiť“ medzi sebou o „priazeň agentov“.

- modelovanie Darwinovskej evolúcie v populácii agentov. Ak by fitness agentov závisela od ich komunikačnej úspešnosti, prípadne od rýchlosti, akou sú schopní úspešnú komunikáciu „po narodení“ dosahovať, dala by sa v prípade dlhodobo stabilného slovníka očakávať jeho genetická asimilácia (Baldwinov efekt). V prípade, že by slovník nebol dlhodobo stabilný vzhľadom na epochy genetickej evolúcie (taká situácia by sa dala očakávať napr. v prípade konkurujúcich si dialektov v populácii, viď predošlý bod, kde by dynamika dialektov mohla byť rýchlejšia ako dynamika Darwinovskej evolúcie), mohla by evolúcia viesť k zvyšovaniu predispozícii na jazykovú interakciu. Dal by sa tiež modelovať stav, kde by Darwinovská evolúcia využívala (aj) iný druh plasticity agentov než kulturálna evolúcia. Do úvahy prichádza v prvom rade zmena topológie rekurentnej siete, prípadne ďalšie vlastnosti agentov pri vhodnom, menej priamom modeli jazykovej interakcie.

- použitie nestatickej konceptuálnej reprezentácie - modelovať posun polohy prototypov a veľkosti konceptu, ako aj vznik celkom nových konceptov. Z hľadiska vierohodnosti modelovania by bolo vhodné namiesto konceptuálnej reprezentácie, modelovanej náhodnými zhlukmi objektov, použiť reprezentáciu skutočných objektov, vnímaných senzormi agentov. Toto všetko by zároveň umožnilo sledovať dynamiku jazyka - ako si poradí so vznikom nových významov.

- zoslabenie predpokladov, použitých pri adaptácii agenta. To sa týka najmä priameho prístupu počúvajúceho k meaning vektoru rozprávajúceho pre potreby adaptácie. Tento proces by mal byť viac nepriamy.

- modelovanie metaforického pomenúvania s využitím vlastností konceptuálneho priestoru

- použitie účinnejšieho algoritmu učenia siete, napr. BPTT (backprop na sieti, rozvinutej v čase). Problém je, že pri použití BPTT zaniká tlak na monotónnu konvergenciu aktivácií výstupnej vrstvy k cieľovému meaning vektoru počas predkladania znakov. Učenie by len nútilo sieť dosiahnuť po poslednom znaku požadovaný aktivačný vektor, bez ohľadu na tvar trajektórie v aktivačnom priestore výstupnej vrstvy. Monotónna konvergencia počas celého predkladania znakov je však nutným predpokladom úspešného generovania znakov (Saussureanovská konštrukcia denominačnej funkcie), vynucujúc si tak korekciu chyby po každom znaku - a teda BPMT učenie. Bolo by teda nevyhnutné upraviť aj konštrukciu denominačnej funkcie.

Dúfame, že práca prispeje, hoci aj malou mierou, k porozumeniu povahy rôznych úrovní reprezentácie a ich vzájomnému vzťahu.

Literatúra

[1] Balkenius, Ch. (1999), Are There Dimensions in the Brain? Lund University Cognitive Science, Lund, Sweden.

[2] Batali, J. (1998), Computational Simulations of the Emergence of Grammar. Department of Cognitive Science, University of California, San Diego.

[3] Berkeley, I.S.N. (1997), Some Myths of Connectionism. The University of Southwestern Louisiana.

[4] Bickle, J., The Philosophy of Neuroscience. Stanford Encyclopedia of Philosophy.

[5] Chella, A., Frixione, M., Gaglio, S. (1997), A cognitive architecture for artificial vision. Artificial Intelligence. 89:73-111

[6] Gärdenfors, P. (1992). A Geometric Model of Concept Formation, Lund University Cognitive Science, Lund, Sweden.

[7] Gärdenfors, P. (1994). Cued and detached representations in animal cognition. Technical report 38, Lund University Cognitive Science, Lund, Sweden.

[8] Gärdenfors, P. (1995). Meaning as conceptual structures. Technical report 40, Lund University Cognitive Science, Lund, Sweden.

[9] Gärdenfors, P. (1995). Language and the evolution of cognition. Technical report 41, Lund University Cognitive Science, Lund, Sweden.

[10] Gasser, M., Colunga, E., Smith, L.B. (1998). Developing Relations. Cognitive Science Program, Indiana University.

[11] Harnad, S. (1993) Grounding Symbols in the Analog World with Neural Nets. Think 2: 12-78 (Special Issue on "Connectionism versus Symbolism" D.M.W. Powers & P.A. Flach, eds.).

[12] Hlaváč, V., Šonka, M. (1992). Počítačové vidění. Grada, Praha.

[13] Hristev R.M. (1998). The ANN Book. Edition 1.

[14] Kvasnička, V., Beňušková, Ľ, Pospíchal, J., Farkaš, I., Tiňo, P., Kráľ, A. (1997). Úvod do teórie neurónových sietí, Iris, Bratislava.

[15] Levy, S., Melnik, O. and Pollack, J. (2000). Infinite RAAM: A Principled Connectionist Basis for Grammatical Competence, CogSci 2000.

[16] Pearlmutter, B. A. (1995). Gradient calculations for dynamic recurrent neural networks: A survey.

[17] Pollack, J.B. (1990). Recursive distributed representations. Artifical Intelligence, 36:77-105.

[18] Rodriguez, P., Wiles, J., and Elman, J. L. (1999). A recurrent neural network that learns to count. Connection Science, 11(1):5–40.

[19] Šefránek, J. (1999). Reprezentácia znalostí a inferencia. Preprint publikácie Inteligencia ako výpočet, Iris, Bratislava.

[20] Simula, O. et.al. (1999). The self-organizing map in industry analysis. Industrial applications of neural networks. CRC Press, 1999.

[21] Steels, L. (1997). The synthetic modeling of language origins. Sony computer science laboratory, Paris.

[22] Wassermann, R. (1998). Revising concepts. In Proceedings of the Fifth Workshop on Logic, Language, Information and Comunication (WoLLIC), Sao Paulo.